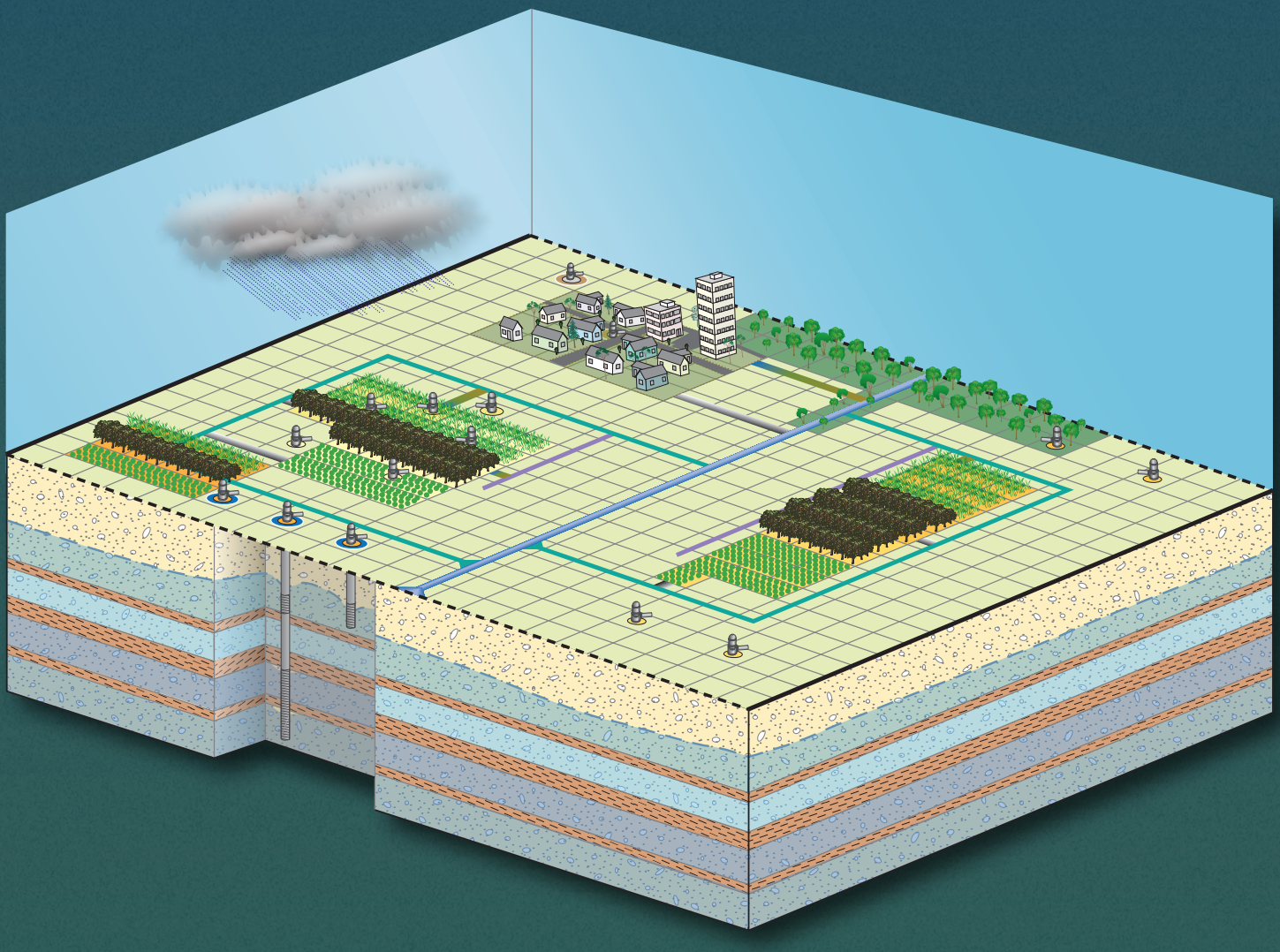
USGS
science for a changing world

Prepared in cooperation with the Bureau of Reclamation

One-Water Hydrologic Flow Model: A MODFLOW Based Conjunctive-Use Simulation Software



Techniques and Methods 6–A60

U.S. Department of the Interior
U.S. Geological Survey

Cover. Generalized view of the model domain

One-Water Hydrologic Flow Model: A MODFLOW Based Conjunctive-Use Simulation Software

By Scott E. Boyce, Randall T. Hanson, Ian Ferguson, Wolfgang Schmid, Wesley Henson, Thomas Reimann, Steffen M. Mehl, and Marisa M. Earll

Prepared in cooperation with the Bureau of Reclamation

Techniques and Methods 6–A60

U.S. Department of the Interior
U.S. Geological Survey

U.S. Department of the Interior
DAVID BERNHARDT, Secretary

U.S. Geological Survey
James F. Reilly II, Director

U.S. Geological Survey, Reston, Virginia: 2020

For more information on the USGS—the Federal source for science about the Earth, its natural and living resources, natural hazards, and the environment—visit https://www.usgs.gov or call 1–888–ASK–USGS.

For an overview of USGS information products, including maps, imagery, and publications, visit https://store.usgs.gov.

Any use of trade, firm, or product names is for descriptive purposes only and does not imply endorsement by the U.S. Government.

Although this information product, for the most part, is in the public domain, it also may contain copyrighted materials as noted in the text. Permission to reproduce copyrighted items must be secured from the copyright owner.

Suggested citation:
Boyce, S.E., Hanson, R.T., Ferguson, I., Schmid, W., Henson, W., Reimann, T., Mehl, S.M., and Earll, M.M., 2020, One-Water Hydrologic Flow Model: A MODFLOW based conjunctive-use simulation software: U.S. Geological Survey Techniques and Methods 6–A60, 435 p., https://doi.org/10.3133/tm6a60.

ISSN 2328-7055 (online)

Preface

This report presents a computer program for simulating the movement and conjunctive use of surface water and groundwater by the U.S. Geological Survey (USGS) hydrologic model, MODFLOW One-Water Hydrologic Model (MF-OWHM) version 2.

All MODFLOW code developed by the USGS is available to download on the Internet from a U.S. Geological Survey software repository. The repository is accessible on the World Wide Web from the USGS Water Resources information Web page at

https://www.usgs.gov/software/modflow-owhm-one-water-hydrologic-flow-model

and a git repository at

https://code.usgs.gov/modflow/mf-owhm

Although this program has been used by the USGS, no warranty, expressed or implied, is made by the authors, the USGS, or the United States Government as to the accuracy or functioning of the program and related program material, nor shall the fact of distribution constitute any such warranty, and no responsibility is assumed by the authors or the USGS in connection with it. Future applications might reveal errors that were not detected in the test simulations. Users are requested to notify the USGS of any errors found in this document or the computer program by using the email address available on the aforementioned Web site. When important updates are made either to the MF-OWHM program or to the documentation, these updates are uploaded to the USGS Web site. Users are encouraged to check the Web site periodically and read the release.txt document provided with the distribution of the software.

The computer program described here is based, in part, on copyrighted scientific methodologies originally obtained from the copyright holder (Schmid, 2004). The copyright holder has granted full permission to the USGS and to the public to quote, copy, use, and modify these methods, as well as publish modified methods. Whereas MF-OWHM includes all the features and core elements of MODFLOW-2005 (rev 1.12), we request that if you use MF-OWHM version 2, you also include proper citation to this document (Boyce and others, 2020) in any related reports, articles, or presentations.

In the download of this computer program is a readme.txt, release.txt, and supplemental documentation. The .txt files can be viewed in any ASCI/UNICODE text viewer. The readme.txt contains an overview of the release and describes the contents of the software download. The release.txt file describes any changes or bug fixes made since the initial release of MF-OWHM. The supplemental documentation provides background knowledge about the software and any features incorporated since the release of this report.

To provide a "living" reference to all input features from all the currently supported packages and processes, an online manual (guide) is available at

https://ca.water.usgs.gov/modeling-software/one-water-hydrologic-model/users-manual/

At the time of this report's publication, the online guide is maintained by Richard Winston (rbwinst@usgs.gov).

A user-group email, MODFLOW_OWHM@usgs.gov, is available for users to electronically report potential software issues (bugs); users may also request to be on a mailing list that sends notices related to the MF-OWHM simulation-software distribution.

Specific correspondence regarding this report, its documented simulation program, notification of simulation issues or bugs, or future feature suggestions can be sent electronically to

MODFLOW_OWHM@usgs.gov

Acknowledgments

A simulation engine of this magnitude could not be completed without the help and advice of many individuals, organizations, and beta testers. The development of this second version One-Water Hydrologic Flow Model was supported by the Rio Grande Project of the Bureau of Reclamation and funds from the U.S. Geological Survey (USGS) California Water Science Center.

The authors would like to thank and acknowledge the following technical reviewers for suggestions that improved this report: Dr. Markus Disse of the Technical University of Munich, Germany; Dr. Thomas Maddock III of the University of Arizona; and Dagmar Llewellyn from the Bureau of Reclamation.

The authors thank Matt Baillie, Robert Berger, and Les Chau of Amec Foster Wheeler for their comments and suggestions with the MF-OWHM2 and Surface-Water Operations Process; Nicholas Newcomb of Luhdorff and Scalmanini for bug fixes and conceptual suggestions; Justin A. Clark, Dale Mason, Keith Nelson, Kyle Richards and Frank Corkhill from the Groundwater Modeling Unit of the Arizona Department of Water Resources for their comments and discussions on subsidence; and Dr. Mary C. Hill of the University of Kansas.

We further acknowledge and thank Howard Franklin from Monterey County Water Resources Agency, Brian Lockwood from the Pajaro Valley Water Management Agency, Dr. Andrew Rich from Sonoma County Water Agency, Dr. Gabriele Chiogna from the University of Innsbruck, Austria, and Dr. Girma Yimer Ebrahim from the International Water Management Institute for groundwater simulation discussions and suggestions for improvement of the beta releases of the software. The authors further acknowledge the California Water and Environmental Modeling Forum for hosting OWHM short courses and having MF-OWHM2 conference sessions during their annual meeting. The dialogues that came in response to those short courses and sessions led to many features within this release.

The authors acknowledge Yong Lai from the Bureau of Reclamation for both advice on surface-water flow and for providing the opportunity for a research exchange workshop discussing MF-OWHM2 in Taiwan. This led to advice and technical conversations from Chung-Hung Chen, Paul Tseng, and Yi Fung Wang from the Taiwan Water Resources Planning Institute; Yun-Da Jack Cheng from the Taiwan Water Resources Agency Groundwater Research Center; Dr. Harold Yih-Chi Tan, Dr. Howard Hao Che Ho, Dr. Gene Jing-Yun You from National Taiwan University; Dr. Liang-Cheng Chang and Dr. Jacky Yu-Wen Chen from National Chiao Tung University; and Dr. Yung-Chia Chiu from National Taiwan Ocean University.

We further acknowledge the FREEWAT developers, Dr. Laura Foglia, Iacopo Borsi, Dr. Steffen Mehl, Dr. Giovanna De Filippis, Dr. Massimiliano Cannata, Dr. Enric Vasquez Suñe, Dr. Rotman Criollo, and Rudy Rossetto for providing QGIS support in MF-OWHM.

The authors thank and acknowledge Dr. Richard B. Winston from the USGS for suggestions that improved this report and for his work developing and maintaining the online user's guide to MF-OWHM input files and supporting MF-OWHM in the graphical user interface ModelMuse. The authors also acknowledge Debra Curry of the USGS for her assistance with the report review process and ensuring that it was published in a timely manner. The authors thank Jeff Falgout from the Core Science Analytics, Synthesis, and Libraries (CSASL) Advanced Research Computing (ARC) group at the USGS for Unix computing assistance. The following USGS personnel have provided invaluable feedback and beta testing of MODFLOW One-Water Hydrologic Flow Model version 2 (MF-OWHM2): Dr. Claudia Faunt, Jonathan Traum, Jonathan Traylor, Joshua Larsen, Andre Ritchie, Amy Galanter, Matt Ely, and Joseph Hevesi.

Contents

Figures

Tables

Conversions, Datums, Abbreviations, Acronyms, and Definitions of Variables

Common Unit Conversions Between International System of Units and U.S. customary units

Multiply	By	To obtain
	Length	
centimeter (cm)	0.3937	inch (in.)
meter (m)	3.281	foot (ft)
	Area	
square meter (m^2)	10.7639104	square feet (ft^2)
square meter (m^2)	0.0001	hectare (ha)
hectare (ha)	2.4710538	acre
acre	43560.0	square feet (ft^2)
	Volume	
cubic meter (m^3)	35.3146667	cubic feet (ft^3)
cubic meter (m^3)	0.0008107	acre-foot (acre-ft)
acre-foot (acre-ft)	43560.0	cubic feet (ft^3)
	Flow rate	
cubic meter per second (m^3/sec)	70.0456199	acre-foot per day (acre-ft/d)

Acre-foot (acre-ft) is a unit of volume used commonly in water resources. Its volume equals one acre of surface area to a depth of one foot

decisiemens (dS) is a unit of electric conductance and is the inverse of electrical resistance.

Elevation, as used in this report, refers to distance above the vertical datum.

Parts per million (ppm), is a unit of concentration in water. It is the same quantity same as milligram per liter.

The symbols “L” and “T” represent any accepted unit for length and time, respectively.

For example, L^3/T represents any volumetric flow rate (such as, m^3/sec or acre-ft/d).

The symbol “M” represents any accepted unit for mass (such as, kilogram or pound-mass).

For example, M/LT^2 represents a pressure (such as, kilogram/meter-second^2).

For variable and input definitions, the symbol “(-)” is used to indicate that the variable or input is unitless.

Datum

Vertical coordinate information is referenced to the North American Vertical Datum of 1988 (NAVD 88).

Horizontal coordinate information is referenced to the North American Datum of 1983 (NAD 83).

Elevation, as used in this report, refers to distance above the vertical datum.

Stage, as used in this report, refers to distance above the vertical datum.

Abbreviations and Acronyms

ASCII	American Standard Code for Information Interchange (basic text file file)
BAS	Basic Package
BCM	Basin Characterization Model
CFP	Conduit Flow Process
CFPM1	Conduit Flow Process Mode 1
CFPM2	Conduit Flow Process Mode 2
CIMIS	California Irrigation Management Information System
CIR	crop irrigation requirement
CU	consumptive use
$D_{irrigation}$	necessary irrigation to meet a land use's consumptive use (CIR/OFE)
DIS	Discritization Package
DRN	Drain Package
DRT	Drain return-flow package
DP	deep percolation, water that infiltrates beneath the root zone
ET	Evapotranspiration
ET_{ref}	Reference evapotranspiration flux [L/T]
FAO	Food and Agriculture Organization of the United Nations
FEI	Fraction of evaporation from irrigation
FIESWI	Fraction of inefficient losses from irrigation to surface water
FIESWP	Fraction of inefficient losses from precipitation to surface water
FMP1	Farm Process, version 1; from MODFLOW-FMP
FMP2	Farm Process, version 2; from MODFLOW-FMP2
FMP3	Farm Process, version 3; from MF-OWHM
FMP	Farm Process, version 4; from MF-OWHM2
FTR	Fraction of transpiration
GCM	Global Climate Model

GHB	General Head Boundary Package
GIS	Geographic Information System
HFB	Hydrologic Flow Barrier Package
HOB	Head Observation Process
HUF	Hydrogeologic-Unit Flow Package
HYDMOD	computer program for calculating hydrograph time series data for MODFLOW
IHM	integrated hydrologic model
IRR	amount of applied, irrigated water to crop
ISO	International Standard Organization
K_c	Crop coefficient for evapotranspiration
K_{cb}	Basal crop coefficient for transpiration
LAI	List-Array Input Style
LGR	Local Grid Refinement
LIST	Listing File, a transcript of all operations in a MODFLOW, MF-OWHM, and MF-OWHM2 simulation
LPF	Layer Property Flow Package
MF	MODFLOW
MF2005	MODFLOW-2005
MODFLOW-FMP	MODFLOW-2000 with the Farm Process version 1
MODFLOW-FMP2	MODFLOW-2005 with the Farm Process version 2
MF-OWHM	MODFLOW-One-Water Hydrologic Model Version 1
MF-OWHM2	MODFLOW-One-Water Hydrologic Model Version 2
MNW1	Multi-Node Well Package version 1 Package
MNW2	Multi-Node Well Package version 2 Package
MULT	Multiplier Package
NAME	MF-OWHM2 Name file
NFARM	number of FMP water-balance subregions
NWBS	number of FMP water-balance Subregions
NWT	Newton-Raphson Solver Package
OFE	on-farm efficiency
PCG	preconditioned conjugate gradient solver package
PVAL	Parameter Value Package
RES	Reservoir Package
RIV	River Package
SFAC	Scale Factor—keyword to indicate advanced scale factors are read

SFR	Streamflow Routing Package
SGMA	Sustainable Groundwater Management Act of California
SUB	Subsidence Package
SWI	Seawater Intrusion Package
SWO	Surface-Water Operations Process
SWR	Surface Water Routing Process
TFDR	total farm delivery requirement
TFR	Transient File Reader
$T_{irrigation}$	transpiration from irrigation
T_p	transpiration from precipitation
TSF	Time-Series File
T_{uptake}	transpiration from groundwater-root uptake
ULOAD	universal input-loading utility
UPW	Upstream weighting flow package
USGS	U.S. Geological Survey
UZF	Unsaturated Zone Flow Package
WBS	Water-Balance Subregions—previously called FMP "Farm"s
WEL	Well Package
ZON	Zone Array Package
ZONEBUDGET	computer program for calculating subregional water budgets for MODFLOW

One-Water Hydrologic Flow Model: A MODFLOW Based Conjunctive-Use Simulation Software

By Scott E. Boyce[1], Randall T. Hanson[1], Ian Ferguson[2], Wolfgang Schmid[3], Wesley Henson[1], Thomas Reimann[4], Steffen M. Mehl[1,5], and Marisa M. Earll[1]

Executive Summary

The U.S. Geological Survey's (USGS) Modular Ground-Water Flow Model (MODFLOW-2005) is a computer program that simulates groundwater flow by using finite differences. The MODFLOW-2005 framework uses a modular design that allows for the easy development and incorporation of new features called processes and packages that work with or modify inputs to the groundwater-flow equation. A process solves a flow equation or set of equations. For example, the central part of MODFLOW is the groundwater-flow process that solves the groundwater-flow equation; the surface-water routing process is an additional process that solves the surface-water flow equation. Packages are code related to the groundwater-flow process. For example, the subsidence package modifies the groundwater-flow process by including aquifer compaction effects on flow. With the development of new packages and processes, the MODFLOW-2005 base framework diverged into multiple independent versions designed for specific simulation needs. This divergence limited each independent MODFLOW release to its specific purpose, so that there was no longer a single, comprehensive, general-purpose hydraulic-simulation framework.

The MODFLOW One-Water Hydrologic Flow Model (MF-OWHM, also informally known as OneWater) is an integrated hydrologic flow model that combines multiple MODFLOW-2005 variants in one cohesive simulation software; changes were made to enable multiple capabilities in one code. This fusion of the MODFLOW-2005 versions resulted in a simulation software that can be used to address and analyze a wide class of conjunctive-use, water-management, water-food-security, and climate-crop-water scenarios. As a second core version of MODFLOW-2005, MF-OWHM maintains backward compatibility with existing MODFLOW-2005 versions, with features that include the following:

- Process-based simulation.
  - Saturated groundwater flow (three-dimensional).
  - Surface-water flow (one- and two-dimensional).
    - Stream and river flow.
    - Lake and reservoir storage.
  - Landscape simulation and irrigated agriculture.
    - Land-use and crop simulation.
    - Root uptake of groundwater.
    - Precipitation.
    - Actual evapotranspiration.
    - Runoff.
    - Infiltration.
    - Estimated irrigation demand.
  - Reservoir operations.
  - Aquifer compaction and subsidence by vertical model-grid deformation.
  - Seawater intrusion by a sharp-interface assumption.
  - Karst-aquifer and fractured-bedrock flow.
  - Turbulent and laminar-pipe network flow.
  - Unsaturated groundwater flow (one-dimensional).
- Internal linkages among the processes that couple hydraulic head, flow, and deformation.
- Redesigned code for faster simulation, increased user-input options, easier model updates, and more robust error reporting than in previous models (approximately 75,000 new lines of Fortran code were added to MF-OWHM).

[1]U.S. Geological Survey.

[2]Bureau of Reclamation.

[3]Commonwealth Scientific and Industrial Research Organization.

[4]Technische Universität Dresden.

[5]California State University at Chico.

MF-OWHM is a MODFLOW-2005 based integrated hydrologic model that can simulate and analyze varying environmental conditions to allow for the evaluation of management options from many components of human and natural water movement through a physically based, supply and demand framework. The term "integrated," in the context of this report, refers to the tight coupling of groundwater flow, surface-water flow, landscape processes, aquifer compaction and subsidence, reservoir operations, and conduit (karst) flow. Another benefit of this integrated hydrologic model is that models developed to run by MODFLOW-2005, MODFLOW-NWT, MODFLOW-CFP, or MODFLOW-FMP can also be simulated with MF-OWHM. At the time of this report's publication, MF-OWHM version 2 (MF-OWHM2) does not include a direct internal simulation of snowmelt, advanced mountainous watershed rainfall-runoff simulation, detailed shallow soil-moisture accounting, or atmospheric moisture content. Atmospheric moisture may be accounted for indirectly by, optionally, specifying a pan-evaporation rate, reference evapotranspiration, and precipitation. These features are not included to ensure that simulation runtime remains short enough to enable the use of automated methods of calibrating model parameters to field observations, which typically require many simulation model runs. The MF-OWHM approach is to include as much detail as possible to simulate hydrological processes, providing the simulation runtimes remain reasonable enough to allow for robust parameter estimation and model calibration.

To represent both natural and human-influenced flow, MF-OWHM integrates physically based flow processes derived from MODFLOW-2005 in a supply and demand framework. From this integration, the physically based movement of groundwater, surface water, imported water, and precipitation serve as supply to meet consumptive demands associated with irrigated and non-irrigated agriculture, natural vegetation, and urban water uses. Water consumption is determined by balancing the available water supply with water demand, leading to the concept of a demand-driven, supply-constrained simulation.

The MF-OWHM Supply-and-Demand Framework is especially useful for the analysis of agricultural water use, where there are often few data available to describe changes in land-use through time, such as crop type and distribution, and the associated changes in groundwater pumpage. This framework attempts to satisfy each land-use water demand with available water supplies—that is, groundwater uptake, precipitation, and irrigation. An option provided in MF-OWHM2 is to automatically increase groundwater pumping for irrigation, which often is unknown, by the calculated residual between demand and the other available sources of supply. From large- to small-scale applications, the physically based supply and demand framework provides key capabilities for simulating and analyzing historical, current, and future conjunctive-use of surface water and groundwater.

To achieve the physically based supply and demand framework, the MODFLOW-2005 standard of no inter-package and -process communication was relaxed for MF-OWHM2. Traditional MODFLOW simulation models required that all packages and processes interact through the groundwater-flow equation or by removing the water flow from the simulation domain. For example, the MODFLOW-2005 representation of a groundwater well extracts water from the groundwater-flow equation (by subtraction) and removes it from the simulation domain. This feature is available in the MF-OWHM framework, but options have been added to allow the specification of a use or destination of pumped groundwater within the model domain, for example, it can be used for irrigation, managed aquifer recharge, or return-flow to streams.

This report documents the new features and capabilities associated with the second release of the One-Water Hydrologic Flow Framework (MF-OWHM2), which expands upon the features of the MF-OWHM by introducing new packages and processes, improving linkages between them, and updating the overall software. The major MF-OWHM2 enhancements include the following:

- Inclusion of a Conduit-Flow Process (CFP) for simulating karst aquifers, leaky pipe networks, and secondary porosity.
- Updates to the Farm Process (FMP).
  - The ability to specify multiple land-use types (crops) within a model cell.
  - The ability to specify additional demand types not associated with land use.
  - Calculation of additional irrigation for soil-salinity flushing.
  - A direct-recharge option to represent infiltration ponds.
  - A "sand" soil type and bare-soil or fallow land-use option.
  - Allow for enabling or disabling, by land-use category, root uptake of groundwater, crop anoxia, or crop-soil stress.
  - Updating of base code from FORTRAN 95 to FORTRAN 2008.
  - Complete redesign of the input structure for easy maintenance and calibration.
- Additions to General Head Boundary (GHB) that include head-dependent conductance and automatic calculation of conductance based on aquifer properties.
- Inclusion of a Calendar Date and Time format for model input and output by specifying a starting simulation date; the model then tracks when each time-step occurs on the calendar.

- Inclusion of alternative package input structures that facilitate easier model maintenance.
  - LineFeed—Stress-period-based input that is structured like a spreadsheet.
  - Time-Series Files— uses calendar dates to assign input by the start and ending dates of each model time step.
  - TabFiles—tabulated data input that have a time stamp and data.
  - Addition of a second well package with enhancements to its TabFile input.
  - New Warning Package (WARN)
  - The Listing file (LIST) is now optional.
  - Designation of multiple budget groups for select packages that are passed to the cell-by-cell output and volumetric budget information for advanced post-processing.
- Upgrades and modifications to the Basic (BAS), Discretization (DIS), Stream Flow Routing (SFR), Multi-Node Well v2 (MNW2), Reservoir (RES), Parameter Value (PVAL), Multiplier Array (MULT), Zone Array (ZONE), and Head Observation (HOB) to improve the execution speed and add features that support conjunctive-use simulation and analysis.
- Code-specific changes include buffering of input and output files for decreasing simulation runtimes, the ability to split output files into multiple files to prevent excessively large file sizes, an advanced program-stop utility that provides detailed information, a universal input-loading utility (`ULOAD`) that uses a transient file reader for more flexible input, and a generic block-input tool that loads block-style input automatically.

The MF-OWHM2 framework also is designed to support the concept of "self-updating" models. This was implemented by separating the MODFLOW input-file structure into spatial (or structural) and temporal components. This allows the input files to be easier to read and modify by model developers, users, and reviewers; thus, the models are easier to use and update. This separation also allows automated programs to query databases, websites, or spreadsheets for data to update the input files (for example, streamflow or specified pumping rates). This automation, or self-updating, of the input files allows for the simulation model to be readily used after its initial construction, thus increasing the longevity and value of the simulation model.

This report includes a hypothetical example using MF-OWHM2. The example problem illustrates how to answer typical conjunctive-use questions by addressing additional irrigation requirements due to salinity flushing. When needed, salinity flushing results in an additional irrigation demand that is met by increased groundwater pumpage.

Introduction

The Modular Ground-Water Flow Model (MODFLOW-2005) is a computer program that uses the finite difference method to simulate the groundwater-flow equation (Harbaugh, 2005). The MODFLOW-2005 framework used a modular design that allows for the easy development and incorporation of features called packages and processes that work with the groundwater-flow equation solver. Packages are features related to the groundwater-flow process (for example, the Well package, "WEL"), whereas a process solves flow equations and can represent an ancillary process that interacts with the groundwater-flow equation (for example, Farm Process, "FMP"). Typically, popular packages were incorporated into the base MODFLOW-2005 code, whereas processes were released as separate, independent versions of MODFLOW. The separate, independent versions resulted in a divergence of MODFLOW development, limiting each independent release to the specific purpose for its design. The simulation of conjunctive management of water resources using MODFLOW-2005 required unifying the various MODFLOW-2005 variants into a single general-purpose hydraulic-simulation framework. This core update to MODFLOW-2005 yielded the integrated hydrologic flow model software called the One-Water Hydrologic Flow Model (MF-OWHM). Beyond "integrating" the separate MODFLOW-2005 variants into one cohesive simulation software, MF-OWHM incorporated new capabilities to the unified code. This fusion of MODFLOW-2005 versions resulted in analysis and simulation software capable of addressing, and thereby advancing, the understanding of a broad class of water-use and sustainability problems, including conjunctive-use, water-management, water-food-security, and climate-crop-water scenarios. Another benefit of this fusion is that existing models developed using the various predecessors—MODFLOW-2005 (Harbaugh, 2005), MODFLOW-NWT (Niswonger and others, 2011), MODFLOW-CFP (Shoemaker and others, 2008), and MODFLOW-FMP (Schmid and others, 2006; Schmid and Hanson, 2009a)—can also be simulated using MF-OWHM and have access to features of other MODFLOW releases.

The improvements, new features, modifications to MODFLOW-2005, and newly developed processes described in this report continue the MF-OWHM goal of retaining and tracking as much water as is feasible in the simulation domain. This provides the scientific and engineering community with confidence in the water accounting and a technically sound foundation to address broad classes of problems for the public. Because complex questions are being asked about the sustainability of water resources and sophisticated tools are required to answer difficult conjunctive-use management questions, the Bureau of Reclamation (Reclamation) cooperated with the U.S. Geological Survey (USGS) to develop this updated version of MF-OWHM incorporating the new capabilities of software and availability of data.

Report Organization

This report is organized as a main text and set of appendixes. The main text of the report presents an overview of the MODFLOW-2005 based hydrologic modeling software and the features and concepts incorporated in MF-OWHM version 2 (MF-OWHM2). The concepts of integrated hydrologic modeling, the MF-OWHM2 supply and demand framework, and the self-updating model structure are introduced for users new to this approach to integrated conjunctive-use modeling. The main text of the report then describes improvements to MODFLOW specific to the MODFLOW-2005 base code and new landscape features, which include the updates and improvements to the Farm Process (FMP). Next, the report introduces the revised Conduit Flow Process (CFP), with emphasis on improvements to the original MODFLOW-CFP (Shoemaker and others, 2008). The report includes a hypothetical example problem to illustrate the application of a subset of the features of MF-OWHM2. The current limitations and future improvements to MF-OWHM2 are the final topics.

The report contains nine appendixes. The first appendix, appendix 0, discusses the meaning and types of syntax highlighting used throughout this report. The next two appendixes (appendix 1 and 2) introduce new utilities for input and temporal separation in MF-OWHM2. In particular, the input and output file utilities *Generic_Input*, *Generic_Output*, the *Universal Loader* utility (`ULOAD`), and how they relate to the new *List-Array Input* syntax (`LAI`) are described appendix 1. Appendix 2 provides details about the LineFeed input format, the *Time-Series File* input format, improvements to the TabFiles, and concludes with suggestions on how to effectively build a *Transient File Reader* (`TFR`). Appendix 3 describes specific updates and improvements to the MODFLOW-2005 part of MF-OWHM2—including improvements to the Basic (BAS), Discretization (DIS), General Head Boundary (GHB), and WEL packages. Appendixes 4 and 5 provide an overview of the theory behind the Farm Process and the data requirements for simulating land use and calculating consumptive use. Appendix 6 describes the new FMP input options and related upgrades. Lastly, appendixes 7 and 8 describe the addition of the Conduit Flow Process (MODFLOW-CFP) in MF-OWHM2 and new features incorporated in it.

MODFLOW-2005 Framework Descendants Relationship to MF-OWHM2

The MODFLOW-2005 modular framework facilitated the development of independent releases designed for specific applications. A diagram of the MODFLOW-2005 variant descendants (fig. 1) shows the relationship of each one to MF-OWHM2. Each of the major independent releases of MODFLOW-2005 variants are discussed in the remainder of this section as context for the development of the MF-OWHM2 software.

MODFLOW-FMP (Schmid and others, 2006; Schmid and Hanson, 2009) was one of the early attempts to develop the ability of MODFLOW-2005 to simulate conjunctive use by including landscape processes. It introduced the Farm Process (FMP1), which simulates crop growth, root uptake of groundwater, water deliveries, and runoff within a supply and demand framework. It also incorporated climate information in the form of reference evapotranspiration and precipitation; climate influences on water consumption by crops and the use of irrigation. MODFLOW-FMP resulted in the capability for dynamic estimation of surface-water diversions and groundwater pumpage, neither of which are necessarily known quantities.

MODFLOW-2005 is based on the concept of structured finite volumes, which are calculated by the finite difference method. Structured grids have the limitation that for a single model row, column, or layer, there must be a constant width, length, or height, respectively. This means that two rows may have different widths, but for any one row, all the columns and layers passing through it must have the same width. This limitation prompted the development of MODFLOW-LGR, Local Grid Refinement (Mehl and Hill, 2005, 2013). MODFLOW-LGR couples a coarse “parent” model grid with a refined “child” grid. The parent and child models both follow the structured finite-volume scheme, but flows at the parent–child boundary are dynamically coupled. The parent grid can, for example, be associated with a regional model that accounts for bulk flows, and a child model embedded within the regional domain can provide a more detailed simulation in a subarea of the regional model. The parent- and child-model coupling may be one-way, passing flows only from parent to child, or two-way, passing flows back and forth iteratively between the parent and child. This coupling allows for a detailed simulation of local areas in a model domain and provides hydrologically reasonable boundary conditions for problems on a local scale.

An assumption of MODFLOW-2005 was that groundwater flow is laminar, fully saturated, and follows Darcy’s law. This assumption is not valid for karst aquifers that are characterized by dual porosity and turbulent flow. This led to the development of the MODFLOW-CFP, Conduit Flow Process (Shoemaker and others, 2008), which can simulate short-circuit “conduits” in groundwater systems. These conduits represent fractures in porous media, karst topography, or a pipe-network distribution system, thereby enabling simulation of turbulent flow.

A MODFLOW-2005 simulation consists of a three-dimensional structured grid analogous to stacked cubes. Each cube has defined groundwater properties and may have flow into or out of the cube, either directly from its interior—such as groundwater pumping—or through any of its six faces. A single MODFLOW cube is called a model cell and is identified by model row, column, and layer. According to the model cell properties and location, the MODFLOW-2005 groundwater-flow equation is solved using the Picard method of successive approximations by integration

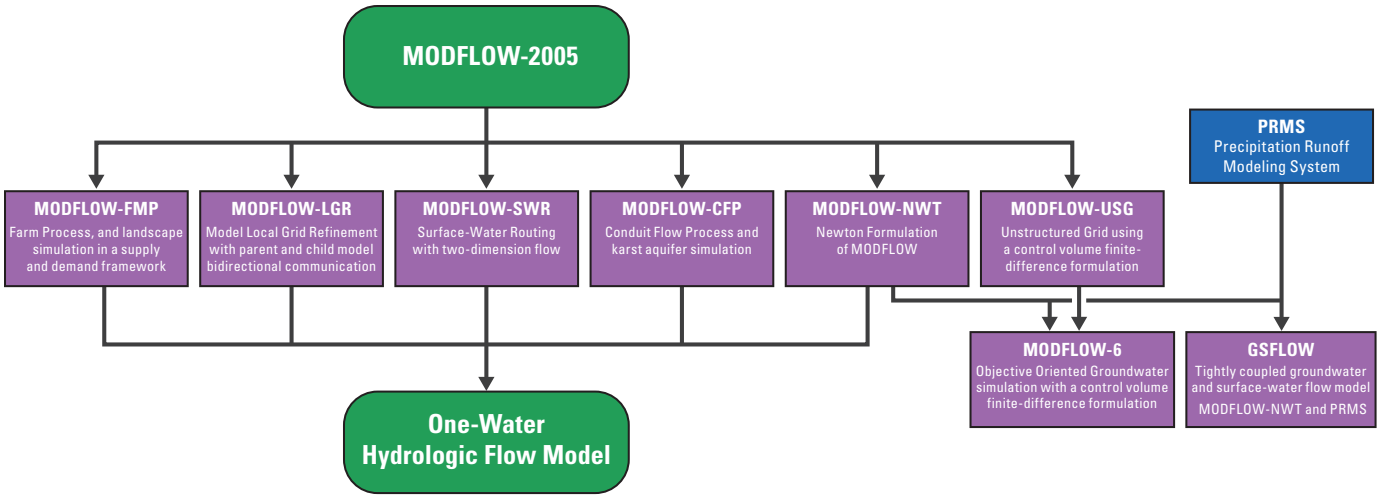


Figure 1. Overview of the MODFLOW-2005 framework and its descendants.

(Harbaugh, 2005). The Picard method iteratively solves the groundwater-flow equation by applying the head estimated in the previous iteration to all nonlinear features to linearize them. This procedure continues until the previous head value, used to linearize the groundwater-flow equation, converges to have the same value as the solution from linear groundwater equations. A limitation of this method is that when a water-level drops beneath the bottom of a cell, the cell is removed from the simulation. When a model cell is removed from the simulation, it is called a "dry" model cell; conversely, when it contains water (having a water level above its bottom) it is called a "wet" model cell. Under some certain circumstances, the Picard method can cause the solution for a model cell to oscillate between wet and dry, thereby preventing the groundwater-flow equation from converging to a final solution (Keating and Zyvoloski, 2009). This is referred to as the MODFLOW wet-dry problem. One of the early attempts to resolve this problem was the development of MODFLOW-NR (Painter and Başağaoğlu, 2007; Painter and others, 2008), which applied upstream weighting, well smoothing, and a Newton-Raphson formulation to MODFLOW, but was only applicable to single-layer models. The USGS generalized this formulation to develop MODFLOW-NWT (Niswonger and others, 2011), which solved the wet-dry problem for multi-layer systems. This variant recast the groundwater-flow equations in a Newton-Raphson framework and called the Newton-Raphson solver package (NWT) of MODFLOW. This formulation requires an asymmetric matrix solver that precludes the use previous default solver methods, so MODFLOW-NWT incorporated two alternative matrix solvers: generalized minimal residual method solver (GMRES) and a specialized solver called χMD (Ibaraki, 2005). By eliminating the wet-dry issue, MODFLOW-NWT enabled a stable simulation of flow in unconfined and perched groundwater systems.

The Surface-Water Routing process (SWR) is an advanced surface-water flow simulator for the MODFLOW-2005 framework (Hughes and others, 2012). The SWR solves the continuity equation for one-dimensional and two-dimensional surface-water flow routing by a simple level- and tilted-pool reservoir routing and a diffusive-wave approximation of the Saint-Venant equations. At the time of writing this report, the SWR is available within MODFLOW-NWT, MODFLOW-2005, and MF-OWHM.

The MODFLOW unstructured grid (MODFLOW-USG; Panday and others, 2013) is a special, branched version of MODFLOW that relaxes the structured-grid requirement, allowing for a true finite-volume representation. This relaxation resulted in a control-volume finite-difference formulation that allows a model cell to be connected to an arbitrary number of adjacent cells—with its resulting cell shape, such as a cube, based on the number of connections. This unstructured feature allows for models to incorporate local grid refinements in areas that require more detailed groundwater-flow simulation by increasing the number of connections, such as a quadtree refinement. This method improves solution stability by using a “ghost-node correction” that interpolates the head value within a model cell to a location that is orthogonal to its adjacent cell. MODFLOW-USG included several of the key MODFLOW-2005 packages to provide a comprehensive groundwater-flow simulation platform without the structured-grid requirement. Building upon the concepts of MODFLOW-USG and MODFLOW-2005, a new groundwater-flow framework, called MODFLOW-6 (Langevin and others, 2017), was developed using an object-oriented programming paradigm. At the time of this publication, MF-OWHM2 does not include the unstructured-grid formulation, because of fundamental differences in how the groundwater-flow equation is solved. It is mentioned here because it provides a flexible model grid for groundwater-flow models that require refinement beyond the capabilities of MODFLOW-LGR.

To accurately represent hydrologic systems where groundwater flow is tightly coupled with surface-water flow and runoff modeling, the Precipitation-Runoff Modeling System (PRMS; Leavesley and others, 1983, Markstrom and others, 2015) was merged with MODFLOW-2005 to form GSFLOW (Markstrom and others, 2008). GSFLOW is a variant of MODFLOW-2005 with a tight coupling between PRMS and MODFLOW-2005 that enables detailed simulation of watersheds holistically, linking streams, lakes, and groundwater flow. A subsequent release of GSFLOW included the advanced flow solvers of MODFLOW-NWT, allowing for a robust representation of unconfined flow. GSFLOW simulations use daily time steps and, optionally, include solar-radiation balances, snowmelt, and air temperature.

In comparison to MF-OWHM2, the GSFLOW incorporation of PRMS provides a more detailed runoff model at the expense of simulation runtime. MF-OWHM2, through the Farm Process, has a simpler runoff model and often faster simulation runtimes. MF-OWHM2 has no inherent limit to time-step length, but a time step greater than or equal to 1 day is recommended when using the FMP. GSFLOW is therefore more physically based for runoff calculations, but may be more challenging to calibrate owing to the confluence of two inherent features: its daily time step and its requirement for convergence of both solutions (PRMS's and MODFLOW's) for each time step. Conversely, MF-OWHM2's simpler runoff model results in models that may be easier to calibrate and verify in terms of runoff. This makes MF-OWHM2 suitable for simulating long historical periods (months to 1,000 plus years) and for short and long-term future projections for evaluating sustainability and conjunctive-use management scenarios, as well as land-use changes.

GSFLOW most accurately represents, and is recommended for, simulation of mountainous headwater regions or valleys that have strong interaction between groundwater and surface runoff. MF-OWHM2 is more appropriate for simulation of aquifer systems in valley settings, especially in arid and semi-arid regions, that do not require a detailed runoff simulation or snowmelt (radiation balance) calculations. MF-OWHM2 simulation domains typically have a lateral boundary at the base of a mountain (or foothill area) or other bedrock feature and may, optionally, use a larger regional rainfall-runoff model to estimate the stream inflows and mountain-block recharge along that boundary. For more details about rainfall-runoff models please see the "Optional Use of Separate Rainfall-Runoff and Hydraulic Models" section, which briefly describes potential companion simulation models. A rainfall-runoff model is not required by MF-OWHM2, and any model domain or aquifer system that can be simulated by MODFLOW-2005, MODFLOW-NWT, MODFLOW-CFP, or MODFLOW-FMP can also be simulated by MF-OWHM2.

Overview of MF-OWHM2

Like its MF-OWHM predecessor, the MF-OWHM2 software can be used to simulate and analyze a wide class of conjunctive-use, water management, water-food-security, sustainability, and climate-crop-water scenarios.

MF-OWHM2 uses a physically based simulation that is connected to a supply and demand framework (fig. 2). This framework starts with the landscape's demand for water consumption that originates from either an administrative requirement—such as urban consumption or managed aquifer recharge—or from the landscape surface's potential evaporation and transpiration. This "landscape water demand" is then satisfied from available supplies of water—such as precipitation, surface water, groundwater, and imported water. Water supply can be limited by physical constraints from the natural and engineered water systems. These constraints result from the physics of natural groundwater and surface-water flow and to physical limits of engineered systems, such as diversion canals or well-production capacity. The landscape water demand can affect both surface water and groundwater because of their interconnectivity. Further, the supply of groundwater and surface water can be controlled by water rights, managed through reservoir operations, or limited by regulations.

MF-OWHM2 is well suited for simulation of agricultural settings because it includes the dynamic estimation of agricultural water consumption and groundwater pumpage for irrigation, routing and management options for surface-water diversions, detailed water-budget output, and embedded functionality for reservoir operations. This dynamic estimation makes MF-OWHM2 a powerful tool for evaluation of present and future agricultural scenarios and assessing conjunctive-use sustainability.

MF-OWHM2 provides a simulation engine for assessing conjunctive use and groundwater sustainability, which may be part of a water agreement, transboundary compact or treaty, or legislation. For example, the California Sustainable Groundwater Management Act of 2014 (SGMA; State of California, 2014) requires that the management and use of groundwater is done without causing "undesirable results." The SGMA specified "undesirable results" are "(1) chronic lowering of groundwater levels that is independent of drought, indicating a significant and unreasonable depletion of supply, (2) significant and unreasonable reduction in groundwater storage, (3) significant and unreasonable seawater intrusion, (4) significant and unreasonable degraded water quality, (5) significant and unreasonable land subsidence, and (6) reduction in surface-water flow, due to groundwater use, that has significant and unreasonable adverse impacts on beneficial uses of the surface water" (http://leginfo.legislature.ca.gov/faces/codes_displayText.xhtml?lawCode=WAT&division=6.&title=&part=2.74.&chapter=2.&article=).

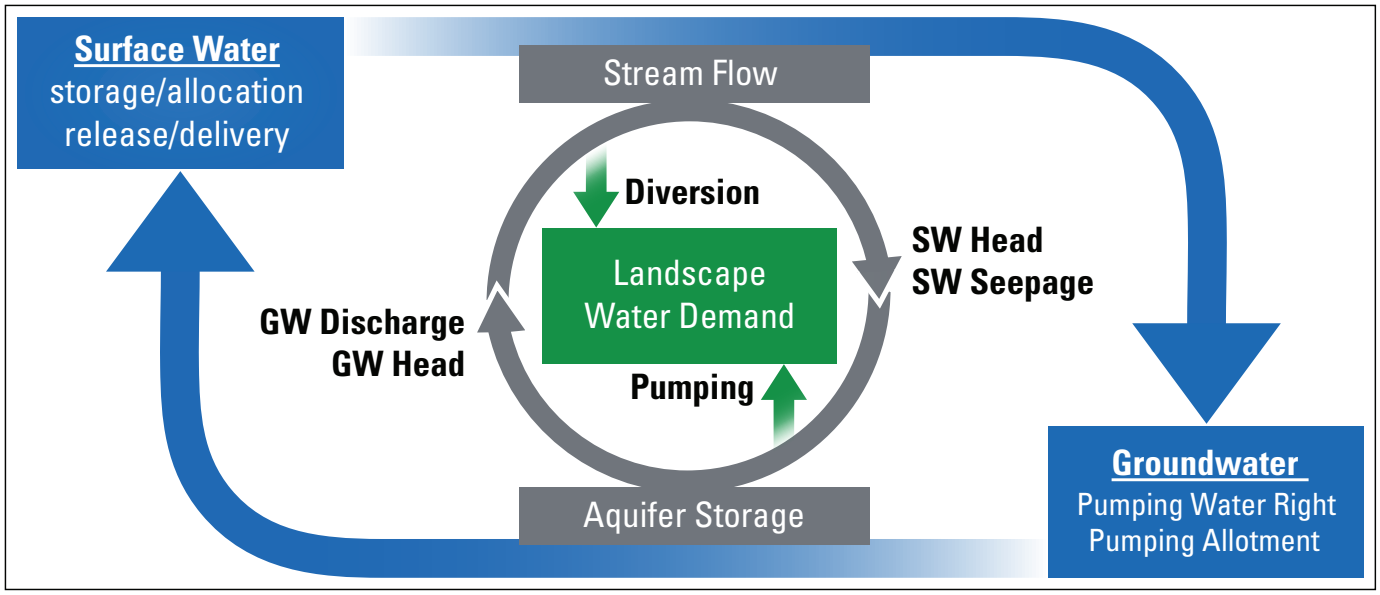


Figure 2. Water flow and use is interconnected through physically based processes and management processes. [SW stands for surface water and GW stands for groundwater; precipitation, not included in figure, is a source of water that could potentially reduce the landscape water demand and increase aquifer storage and stream flow.]

Five of the six SGMA "undesirable results" (1, 2, 3, 5, and 6) can be directly simulated in the MF-OWHM framework, and the sixth, degraded water quality (4), can be simulated indirectly. (1) *Chronic lowering of groundwater levels* can be evaluated through analyzing aquifer heads output with the Head-Observation Process (HOB) or HydMod (HYD). (2) *Significant and unreasonable reduction in groundwater storage* can be evaluated by using MF-OWHM2's detailed water-budget outputs or using standard post-processing tools, such as Zonebudget (Harbaugh, 1990), that analyze the Cell-By-Cell (CBC) flow output. (3) *Significant and unreasonable seawater intrusion* can be evaluated with the Seawater Intrusion (SWI) Process (Bakker and others, 2013) using an equivalent freshwater head boundary condition or addressed indirectly by linking MF-OWHM2 to a mass transport simulation model. (4) *Significant and unreasonable degraded water quality* cannot be directly simulated in MF-OWHM2; however, using MF-OWHM2 with the Link-MT3DMS (LMT) Package can generate flow input for MT3DMS (Zheng and Wang, 1999), and MT3DMS-USGS (Bedekar and others, 2016) can be used to determine water-quality changes in groundwater and surface water. (5) *Significant and unreasonable land subsidence* can be evaluated with the Subsidence and Aquifer-System Compaction (SUB) Package (Hoffmann and others, 2003) or Subsidence for Water Table Aquifers (SWT) Package (Leake and Galloway, 2007). (6) *Reduction in surface-water flow* is part of understanding conjunctive use and can be determined using water budgets from the Stream Flow Routing (SFR) Package (Prudic and others, 2004) or the Surface-Water Routing (SWR) Process (Hughes and others, 2012).

The first release of MF-OWHM (Hanson and others, 2014d) was selected by the World Bank Water Resource Software Review as one of three recommended simulation programs for "integrated surface and ground water simulations required for conjunctive management" (Borden and others, 2016). MF-OWHM incorporated and built on the features and source code of MODFLOW-2005 (Harbaugh, 2005), MODFLOW-FMP2 (Schmid and Hanson, 2009a), MODFLOW-NWT (Niswonger and others, 2011), MODFLOW-SWR (Hughes and others, 2012), MODFLOW-SWI (Bakker and others, 2013), and MODFLOW-LGR version 2 (Mehl and Hill, 2005, 2013) and included the Riparian Evapotranspiration (RIP-ET) package (Maddock and others, 2012). Visual development of MF-OWHM simulation models is possible through the USGS graphical user interface ModelMuse (Winston, 2009, 2014). MF-OWHM is also used as the primary simulation engine for FREEWAT, a European Union sponsored open-source water-management software environment and geographic information system (GIS) user interface (Rossetto and others, 2015; De Filippis and others, 2017).

The capabilities incorporated in MF-OWHM2 have been designed to maintain backward compatibility with the assimilated MODFLOW versions. Any existing model that runs with MODFLOW-2005 should run with MF-OWHM and MF-OWHM2 with little to no modification. There are two exceptions to this. The first is that the variants descended from MODFLOW-2005 automatically set the last input variable on a line to zero if it was not present. For example, if an input file contained a line expected have three integers (layer, row, column) and there were only two integers on the line, then MODFLOW would automatically set the third integer to zero (that is, column is equal to 0). In contrast, MF-OWHM2 raises an error stating that a required input variable is missing, and the user must include it for the simulation to continue. The second exception is that the default input for MODFLOW-2005 is fixed formatted, which required the Basic (BAS) package keyword, **FREE**, to use free formatted input. In practice it is not common, nor is it recommended, for any of the MODFLOW versions to use the fixed formatted input. Consequently, MF-OWHM2 reads all input with free format by default. MF-OWHM2 still accepts the BAS keyword **FREE**, but also offers the keyword **NOFREE** to use fix formatted input for legacy models.

MF-OWHM2 includes a conduit-flow process for karst aquifers and leaky pipe networks; a variety of improvements to all the MODFLOW packages, including head-dependent conductance for GHB cells; a new Well Package (WEL); and a complete redevelopment of the FMP. It also includes additional features to facilitate easier model updates, faster execution, better runtime-error messages and reporting, a new Warning package (WARN), and more cross-communication between the traditional MODFLOW packages.

MF-OWHM2 also includes a new "self-updating" structure. The self-updating aspect refers to conversion of the MODFLOW input-file structure to one that separates the spatial and structural input from the temporal input; for example, well location is separate from its pumping rate. This allows the input-file structure to be to be easier to read and modify by the model developers, users, and reviewers; thus, the temporal files are easier to use and update. This separation also was implemented to allow automated programs to query databases, websites, or spreadsheets for new data to update the input files (for example, streamflows or specified pumping rates). This automation, or self-updating, of the input files allows a simulation model to more readily be used after its initial construction, thus increasing the longevity and value of the simulation model.

To facilitate easy maintenance of models, three new types of input files were introduced. The first an alternate input-file type, called LineFeed (appendix 2) is now available for the WEL, GHB, and MNW2 packages. LineFeed separates a package's spatial and construction information from its temporal information. This allows the user to predefine all the static model properties once and have a separate file of the transient stress-period properties that is easy to maintain. The second type of input file is called a TabFile (time-tabulated input). TabFiles have been linked to an ExpressionParser to allow the tabulated values to be passed to a user-defined function (appendix 2). This function requires fewer TabFiles to describe a set of model features; for example, sea-level data can be applied to a user-defined equation that translates it to a freshwater equivalent boundary condition across an ocean boundary. Similar to TabFiles, the third new type of input file, called a Time-Series File (TSF), is introduced. The TSF is tied to calendar dates and offers more options to specify the handling of the time-series data (for example, interpolate, nearest value, time-weighted mean, or step function) than are available in TabFiles.

A new set of input and output utilities (appendix 1) offer a more general and user-friendly input structure called the List-Array Input (`LAI`). This input structure allows for keyword-based input that supports two-dimensional array input and record-based list inputs, including advanced scale-factor options. The `LAI` can either load input once or load data by stress period using the *Transient File Reader* (`TFR`), which is a pointer file that directs how the input is loaded every stress period. Additional new input and output file utilities, called *Generic_Input* and *Generic_Output*, provide standard methods of opening and handling input and output files, respectively. The *Generic_Input* is used by the Universal Loader utility (`ULOAD`) to read any style of input file. `ULOAD` also is used by `LAI` to provide a generic input-format framework. In MF-OWHM2, `LAI` is only available to the FMP and SWO. *Generic_Input* and *Generic_Output* files are used by the new input and output options for multiple packages.

The traditional input structure for MODFLOW-2005 was based on model-dependent coordinates—row, column, layer, and simulated time. Although this was convenient for coding purposes, simulation outputs required translation to real-world coordinates (that is, geographical and date-time coordinates). MF-OWHM2 includes the optional specification of a Cartesian coordinate system that is linked to the model-grid and calendar-date systems. This option lets the user specify a model feature—for example, WEL or GHB—by X, Y, and Z coordinates. If an initial calendar date is specified, then it is propagated forward with each time step, considering leap and non-leap years. For example, for an FMP supply well—Farm Well—the user can specify a starting and ending date to represent when it is available to supply irrigation water. In addition, when a starting calendar date is specified, the volumetric budget, hydraulic head observation package (HOB), and the FMP include the calendar date in the output. The HydMod package (HYD) now only uses the MF-OWHM2 Cartesian coordinate system to spatially position each time-series output for a point (XL and YL) in the model domain. If the MF-OWHM2 Cartesian coordinate system is not specified, then the origin of the model domain is automatically set to the lower-left corner—which is the HydMod coordinate system used in MF-OWHM.

MF-OWHM2 includes a set of new features that facilitate parameter estimation. The Basic Package (BAS) offers an option that cycles through the input files to check, before parameter estimation, if they loaded correctly. Also, the "FastForward" feature allows the input files to be cycled through to a specified starting stress period, which allows parameter estimation for a specified simulation window without having to rebuild an entirely new input dataset.

A variety of new features improve users' control over simulation outputs. To reduce excessive writing to a hard drive, writing the CBC flow file can be turned off, all output files can be optionally buffered in RAM, and writing the Listing File (LIST) is optional (total suppression). Packages can use sub-budget groups for improved tracking and reporting of the volumetric budget. The HOB package allows head observations to be written at the end of each time step using null values for observations yet to be simulated.

MF-OWHM2 Package and Process Support

This section presents short descriptions of the packages and processes supported in MF-OWHM2 (table 1). Please check the Online Guide to MODFLOW-OWHM version 2 (https://ca.water.usgs.gov/modeling-software/one-water-hydrologic-model/users-manual/) for any other packages supported after the release of this report. The packages and processes have been grouped according to a common functionality. The following are the groups specified in table 1:

- Parameter
  - Packages associated with a MODFLOW Parameter Process. Parameter values, once loaded, can alter the properties of another package (for example, rescaling the hydraulic conductivity in a flow package).
- Flow Package
  - Packages that specify the aquifer flow properties. For example, hydraulic conductivity and specific storage.
  - Only one flow package may be used during a MF-OWHM simulation.
- Flow Modification
  - Packages that modify the flow package; they are not required for a simulation.
- Land Use
  - Farm Process related group.
- Karst/Pipe Flow
  - Conduit Flow Process related group.
- Transport
  - Group represents the package that produces the flow-link binary input file to MT3DMS (Zheng and Wang, 1999) and MT3DMS-USGS (Bedekar and others, 2016) for transport simulation.
- Fixed Boundary
  - Packages that represent constant hydraulic head or constant flux
  - The CHD package is not recommended for conjunctive use because of the lack of head-dependence.
- Head-Dependent Boundary
  - Packages that calculate boundary flows based on hydraulic head.
  - GHB is the most commonly used boundary condition. (Note that most head-dependent packages are variations of GHB.)
  - For a conjunctive-use simulation, the drain return-flow package (DRT) is recommended instead of the drain (DRN) package. The DRN package always removes water from the simulation domain, whereas the DRT package has the option to return runoff to the landscape or groundwater.
  - Caution is advised when simulating evapotranspiration using FMP, RIP, evapotranspiration (EVT), and evapotranspiration segments (ETS). Only one should be used in each model cell because they all simulate evapotranspiration, and there no internal check for double-accounting evapotranspiration.
- Subsidence
  - Packages that simulate interbed storage and aquifer compaction.
  - Only one Subsidence package may be used in a simulation.
- Surface Flow
  - Packages that simulate surface-water flow and storage.
  - Caution is advised when using the river package (RIV) for integrated or conjunctive-use simulations because it is more like GHB (Head-Dependent Boundary) than surface-water flow.
- Groundwater Well
  - Packages that represent groundwater well functionality through either extraction or injection.
  - Note that the WEL package is technically a Fixed Boundary package, but it is designed to represent groundwater pumping or injection. Models can use the WEL package to represent boundary recharge or inflow.
- Observation
  - Packages that support other packages by writing output in a convenient form for post-processing or calibration.

- Solver
  - Packages that solve the groundwater flow equation.
  - Only one solver may be used during a simulation.
- HUF Extension
  - Specialized packages that only function when the HUF flow package is in use.

Table 1. MODFLOW One-Water Hydrologic Flow Model version 2 (MF-OWHM2) supported packages and processes grouped by common functionalities.

[LGR, Local Grid Refinement; MT3DMS, Modular Three-Dimensional Multispecies Transport Model Dimensional Multispecies Transport Mode; MT3D-USGS, U.S. Geological Survey update to MT3DMS; 1D, One-dimensional; 2D, Two-dimensional; —, not applicable]

Package	Package or process name	Short description
NAM	Name file that lists all packages in use	Not a package, but loaded at start of simulation to declare packages and processes used by user's model application.
LIST	Listing file	Contains transcript of package output, warnings, and errors.
WARN	Warning file	Contains a transcript of package warnings and errors that are raised and written to the listing file.
BAS	Basic	Defines global options, active model cells, and initial head.
DIS	Discretization	Specifies model time and space discretization.
OC	Output control	Specifies writing of output to list and cell-by-cell flow file.
		Parameter
ZONE	Zone file	Parameter process—specify parameter zones of application.
MULT	Multiplier file	Parameter process—specify parameter multiplication arrays.
PVAL	Parameter value file	Parameter process—specify global parameters.
		Flow package
BCF	Block-centered flow	Defines aquifer flow properties.
LPF	Layer-property flow	Defines aquifer flow properties.
UPW	Upstream weighting	Defines aquifer flow properties.
HUF	Hydrogeologic-unit flow	Defines aquifer flow properties.
		Flow modification
HFB	Horizontal flow barrier	Barriers to flow between model cells (for example, faultline or slurry walls).
UZF	Unsaturated-zone flow	Vertical flow of water through the unsaturated zone to water table.
SWI	Seawater intrusion	Vertically integrated, variable-density groundwater flow and seawater intrusion in coastal multi-aquifer systems.
		Land use
FMP	Farm Process	Dynamic simulation of land use, evapotranspiration, surface-water diversions, and estimation of unknown pumpage.
		Karst/pipe flow
CFP	Conduit Flow Process	Simulation of turbulent flow through karst conduits or pipe networks.
		Transport
LMT	Link-MT3DMS	Produces a binary flow file that is used for MT3DMS and MT3D-USGS for transport simulation.
		Fixed Boundary
BFH	Boundary flow and head	LGR child model only—couples parent model's flows and heads to child model.
CHD	Time-variant specified-head	Specifies model cells that have a constant head (not recommended for conjunctive use).
FHB	Flow and head boundary	Specifies model cells that have a constant head or constant flux in or out.
RCH	Recharge	Specified flux distributed over the top of the model domain.

Table 1. MODFLOW One-Water Hydrologic Flow Model version 2 (MF-OWHM2) supported packages and processes grouped by common functionalities.—Continued

[LGR, Local Grid Refinement; MT3DMS, Modular Three-Dimensional Multispecies Transport Model Dimensional Multispecies Transport Mode; MT3D-USGS, U.S. Geological Survey update to MT3DMS; 1D, One-dimensional; 2D, Two-dimensional; —, not applicable]

Package	Package or process name	Short description
	Head-dependent boundary	
GHB	General head boundary	Simulates head-dependent flux boundaries.
DRN	Drain	Simulates head-dependent flux boundaries that remove water from domain if head is above a specified elevation.
DRT	Drain return	Simulates head-dependent flux boundaries that move water from model cell if head is above a specified elevation.
RIP	Riparian evapotranspiration	Simulates evapotranspiration separately for multiple plant functional groups in a single model cell.
EVT	Evapotranspiration	Simulate a head-dependent flux out of the model distributed over the top of the model domain.
ETS	Evapotranspiration segments	Simulates evapotranspiration with a user-defined relation between evapotranspiration rate and hydraulic head.
RES	Reservoir	Simulates leakage between a reservoir and the underlying groundwater.
	Subsidence	
IBS	Interbed-storage	Simulates compaction of low-permeability interbeds within layers (legacy code—recommended to use SUB instead) (not recommended for conjunctive use).
SUB	Subsidence and aquifer-system compaction	Simulates drainage; changes in groundwater storage; and compaction of aquifers, interbeds, and confining units that constitute an aquifer system.
SWT	Subsidence for water table aquifers	Simulates compaction for changes in water table by including geostatic stresses as a function of water-table elevation.
	Surface flow	
RIV	River	Simulates head-dependent flux boundaries by specifying a river stage (not recommended for conjunctive use).
LAK	Lake	Simulates lake storage and flow.
STR	Stream	Flow in a stream is routed instantaneously to downstream streams (legacy code—recommended to use SFR instead).
SFR	Streamflow-routing	Simulates streamflow either by instantaneously routing to downstream streams and lakes or routed using a kinematic wave equation.
SWR	Surface-Water Routing Process	Simulates surface-water routing in 1D and 2D surface-water features and surface-water and groundwater interactions.
	Groundwater well	
WEL	Well (Version 2)	Specified flux to model cells in units; revised TABFILE input.
WEL1	Well (Version 1)	Specified flux to model cells in units; original TABFILE input.
MNW1	Multi-node, drawdown-limited well	Simulates wells that extend to more than one cell (legacy code—recommended to use MNW2 instead).
MNW2	Multi-node well	Simulates “long” wells that are connected to more than one model cell; calculates well head and well potential production.
	Observation	
MNWI	Multi-node well information	Provides detailed output from MNW2 wells.
HYD	HydMod	Provides time series of observations from SFR, SUB, and Head.
GAGE	Stream gaging (monitoring) station	Provides output for specified SFR segments and LAK lakes.
HOB	Head-observation	Specifies observations of head in aquifer.
DROB	Drain (DRN) observation	Specifies observations of DRN related flows.
DRTOB	Drain Return (DRT) observation	Specifies observations of DRT related flows.
GBOB	GHB observation	Specifies observations of GHB related flows.

Table 1. MODFLOW One-Water Hydrologic Flow Model version 2 (MF-OWHM2) supported packages and processes grouped by common functionalities.—Continued

[LGR, Local Grid Refinement; MT3DMS, Modular Three-Dimensional Multispecies Transport Model Dimensional Multispecies Transport Mode; MT3D-USGS, U.S. Geological Survey update to MT3DMS; 1D, One-dimensional; 2D, Two-dimensional; —, not applicable]

Package	Package or process name	Short description
	Observation—Continued	
CHOB	CHD observation	Specifies observations of CHD related flows.
RVOB	RIV observation	Specifies observations of RIV related flows.
	Solver	
NWT	Newton-Raphson groundwater formulation	Solves groundwater-flow equation with Newton-Raphson method; requires UPW or LPF as flow package.
PCG	Preconditioned conjugate-gradient	Primary MODFLOW-2005 solver.
PCGN	PCG solver with improved nonlinear control	Solver with advanced dampening and relaxation for highly nonlinear groundwater models.
GMG	Geometric multigrid solver	Geometric multigrid preconditioner to conjugate gradient solver.
DE4	Direct solution solver	Use Gaussian elimination solver for the groundwater-flow equation.
SIP	Strongly implicit procedure	Legacy code—recommended to use PCG or PCGN.
	HUF extension	
KDEP	Hydraulic-conductivity depth-dependence	HUF extension that allows for the automatic calculation of depth-dependent horizontal hydraulic conductivity.
LVDA	Variable-direction horizontal anisotropy	HUF extension that allows for the automatic variable-direction horizontal anisotropy.

Integrated Hydrologic Modeling

Simulation and mathematical representation of the hydrologic cycle often are based on the assumption that each process in the cycle is independent. The most common assumption is that groundwater flow is decoupled from surface-water flow, such that surface water is treated as a simple boundary condition for the groundwater system (for example, the river package, RIV). Similarly, some surface-water models treat groundwater inflow—called base flow—and outflow as a constant value. When it is important to understand the relationship between groundwater and surface water, such as for conjunctive-use management, the decoupling assumption breaks down.

The groups of physical systems represented in integrated hydrologic modeling (IHM) vary depending on the document describing the IHM. Typically, IHM involves the simulation of multiple hydrological processes across the hydrologic cycle. The term "integrated," in the context of this report, refers to the tight coupling of groundwater flow, surface-water flow, landscape processes, subsidence and aquifer compaction, reservoir operations, and conduit or karst flow. To run MF-OWHM2, a groundwater flow package (LPF, UPW, HUF) must be specified; the rest of the integrated features are optional. For example, if there is neither subsidence nor conduit flow in an aquifer system, then there is no need to include them in the simulation.

The original MODFLOW-2005 had one flow process: groundwater flow. This necessitated that all packages in MODFLOW-2005 communicated through the groundwater-flow equation as boundary conditions, such as specified hydraulic head or specified flows. This yielded what is called "head-dependent flow," which is controlled by the hydraulic head in a model cell and in the cells adjacent to it. This approach prevented communication among packages and did not allow the transfer of water from one package to another, except through groundwater flow.

MODFLOW-FMP relaxed this requirement by allowing the transfer of water outside of the groundwater-flow process. MF-OWHM2 incorporated MODFLOW-FMP and extended this ability by allowing cross-communication among the other assimilated MODFLOW versions. The additional cross-communication introduced the potential for simulation capabilities associated with what is called "flow-dependent flow," "consumption-dependent flow," and "deformation-dependent flow" (Hanson and others, 2014d). Flow-dependent flows are calculated from flows that originated from other flow processes. For example, drain flow from the DRT package, which originated as a head-dependent flow, could move drained water to a SWR segment or SFR stream reach as a flow-dependent flow. Another example is surface-water runoff that originates from delivery losses from irrigation water. Consumption-dependent flows originate from the demand for water by irrigated and non-irrigated agriculture, natural vegetation, and urban water uses. The water demand is then satisfied by "flows" either from natural sources or from human sources. Examples of natural sources are root uptake from groundwater or precipitation that falls over the landscape.

Examples of human sources are groundwater pumping, surface water diverted for delivery, and reclaimed water. Deformation-dependent flows result from the vertical deformation associated with compaction of aquifer materials during subsidence. The user can configure the subsidence package (SUB) to adjust the simulation domain's vertical discretization in response to aquifer compaction. The change in vertical discretization alters the aquifer-flow properties (transmissivity and conductance terms). Lastly, the change in surface gradient and aquifer-flow properties may alter the surface-flow features (such as SFR streamflow).

Each of the previously described dependent flows can be interlinked. For example, surface-water diversions to meet irrigation requirements can decrease streamflow (consumption-dependent flow); conversely, irrigation surface runoff, which may include groundwater pumping, can increase streamflow (flow-dependent flow). Groundwater pumping could be characterized as a head-dependent flow (for example, MNW2 well) and might affect the groundwater hydraulic head. The associated change in head could cause subsidence, resulting in deformation of the vertical model grid; in turn, the change in slope of the deformed land surface could alter streamflow (deformation-dependent flow). In addition to being interlinked by their dependencies, flows can also be climate dependent. The climate dependence plays a dominant role because it typically affects major sources of water, specifically precipitation, and losses of water through evapotranspiration.

The subsections that follow provide an overview of the main processes represented in the MF-OWHM2 conceptual framework. The MF-OWHM2 framework builds upon the MODFLOW-2005 framework and incorporates the other processes and packages from other MODFLOW versions in one platform for the simulation of conjunctive use. MF-OWHM2 has incorporated major processes that can be selectively coupled to the fundamental groundwater-flow process combined with surface-water, landscape, reservoir, and conduit-flow processes. The MODFLOW framework defines a process as an operation that solves a major equation or set of equations (for example, groundwater flow, surface-water flow, on-farm water use), and a package is the part of the model addressing a single aspect of simulation (for example WEL, GHB). Overall, the suite of processes and potential couplings results in an integrated hydrologic modeling toolbox that allows the simulation of head-dependent flows and the associated boundary conditions, along with flow-dependent, consumption-dependent, and deformation-dependent flows.

The MF-OWHM2 simulation time frame is specified as a set of stress periods, which are composed of a set of time steps. A stress period is a length of time for which MF-OWHM2 simulates groundwater flow. At the start of each stress period, all model related stresses are specified and applied for the duration of the stress period. For example, at the start of each stress period, the WEL package declares all the wells in use for the stress period and their pumping rate. The number of stress periods, and their associated length of time, determines the total simulation time frame. Typically, stress periods are in line with the months of the year; using the appropriate number of days for each month and take into account 29 days in February for a leap year. For example, a 3-month time frame in a non-leap year would have three stress periods of 31, 28, and 31 days, respectively, to represent January, February, and March. Within each stress period are a set of time steps that subdivide solving the MF-OWHM2 flow equations at a shorter interval of time. The concepts of the stress period and time step are a fundamental to MODFLOW; simplify the input to each stress period while solving the MODFLOW equations at shorter time length, the "time step." MF-OWHM2 does differ from MODFLOW in that some input can be specified at the time step level by using override keywords or defining input with TabFiles or Time-Series Files (TSF). Also, MF-OWHM2 does have packages that dynamically change at the time-step level based on user input at the stress-period level. For example, an FMP agricultural supply well's maximum pumping capacity is specified by stress period, but the actual pumping rate is determined every time step by agricultural demand. Another difference is that MF-OWHM2 allows for time-step lengths to be directly specified, rather than being a subdivision of a stress period. This modification was necessary to allow the user greater control over the time-step length and ensure that time steps are simulated using whole numbers, such as 5 days, instead of 3.1416 days.

Groundwater Flow

The fundamental component of head-dependent flow in MF-OWHM2 is the calculation of the groundwater flow through the main flow package. This package establishes the groundwater-flow equations for a given set of aquifer properties, which can then be modified by any additional packages. For a complete formulation of the MODFLOW-2005 groundwater-flow equation and how it is translated into a block-centered finite-difference scheme, please see Harbaugh (2005). This section provides a brief overview of hydraulic head and groundwater flow.

Hydraulic head, sometimes called piezometric head, total head or head, represents the mechanical energy per unit weight of fluid in the system, or simply is the potential for water flow through a porous media. Hydraulic head is measured as the elevation of freshwater above a datum that can be supported by the hydraulic pressure at a given point in a groundwater system. For consistency, head is typically referenced to a standard elevation datum, such as the North American Vertical Datum of 1988 (NAVD 88). Hydraulic head (Hemond and Fechner, 2015) is calculated as the sum of a pressure term ($P/\rho_{fw}g$), an elevation term (Z), and a kinetic energy term ($v^2/2g$):

$$h = \frac{P}{\rho_{fw} g} + Z + \frac{v^2}{2g} \quad (1)$$

where

- h is the hydraulic head for a given pressure (L),
- P is the gauge pressure measured at elevation Z (M/LT^2),
- ρ_{fw} is the freshwater density (M/L^3),
- Z is the elevation that the gauge pressure is measured at (L),
- v is the velocity of water at the referenced point (L/T), and
- g is the acceleration due to gravity (L/T^2).

The pressure term in the context of MODFLOW assumes constant freshwater density and represents an equivalent gauge pressure at elevation Z within a column of fresh water. For a continuous, nonmoving body of water, the kinetic energy term can be assumed to be zero ($v = 0$). Because of the low flow rate of groundwater compared to the pressure and elevation terms, this simplifies equation 1:

$$h = \frac{P}{\rho_{fw} g} + Z \quad (2)$$

In MODFLOW, and consequently MF-OWHM, the change in hydraulic head across an aquifer determines where groundwater flows. Specifically, groundwater flows toward regions of the aquifer with a lower hydraulic head.

Groundwater flow in MODFLOW relies on the application of Darcy's law to the conservation of mass (continuity) equation. Darcy's law, originally formulated on the basis of empirical evidence by Henry Darcy (1856), can be derived from the Navier-Stokes equations (Whitaker, 1986). Darcy's law is formally defined as follows:

$$q = -K \frac{\Delta h}{D} \quad (3)$$

where

- Δh is change in hydraulic head between two points (L),
- D is the distance between the two points (L),
- K is the hydraulic conductivity of the aquifer material along the distance D (L/T), and
- q is the specific discharge or Darcy flux (L/T).

The physical parameter of hydraulic conductivity, K, describes the resistance to flow in the porous medium that makes up the groundwater system. Hydraulic conductivity is a function of the mean grain diameter, a shape constant of the soils called specific permeability, dynamic viscosity, and specific gravity (Willis and Yeh, 1987; Boyce, 2015).

The representation of hydraulic conductivity in a groundwater simulation model is hampered by the difficulty of direct measurements of K at scales applicable to regional hydrologic models. Hydraulic conductivity is a non-uniformly distributed property; direct measurements at the field scale do not capture all the spatial variability present at the larger scales; hence, uncertainty increases when extrapolating K to typical scales of a simulation domain. In addition, hydraulic conductivity stochastically varies in the subsurface. This stochastic variation can be described by the log-normal probability distribution (Freeze, 1975) or the gamma and log-gamma distributions (Loáiciga and others, 2006). Because of spatial variability in hydraulic conductivity, representative values in hydrologic models are often inferred by solving inverse problems, such as an aquifer test, or using optimization techniques, called parameter estimation. If hydraulic conductivity is treated as stochastic, then it can be solved with a Bayesian inverse problem (further discussion of this approach is beyond the scope of this report). For most situations, the estimated hydraulic conductivity is distributed over zones of model cells on the basis of available information and is calibrated to field observations.

In MODFLOW, groundwater flow between model cells is calculated using hydraulic properties, saturated thickness, and the associated hydraulic head. MODFLOW assumes that groundwater flow between model cells is primarily horizontal and laminar, which originates from a modified Dupuit-Forchheimer assumption. To account for vertical flow, MODFLOW specifies a vertical hydraulic conductivity and applies Darcy's law in the vertical direction. This results in a full three-dimensional formulation of groundwater flow.

Because flow passes through each face of a model cell, it is advantageous to align the model grid in the primary direction of groundwater flow. This ensures that groundwater flow passes smoothly between model cells. This is illustrated in figure 3, which has a model grid that is not aligned with the general groundwater flow and another grid that has been rotated to have the cell faces aligned with the general direction of groundwater flow.

The mathematical reason for aligning the model grid in the dominant-flow directions is that hydraulic conductivity (K) is a second-order, symmetric tensor (eq. 4). This tensor contains a set of off-diagonal terms (K_{xy}, K_{xz}, K_{yz}) that represent the rotational offset of the model grid (or model cell faces) from the principal directions of flow (fig. 3*A*).

$$K = \begin{bmatrix} K_{xx} & K_{xy} & K_{xz} \\ K_{xy} & K_{yy} & K_{yz} \\ K_{xz} & K_{yz} & K_{zz} \end{bmatrix} \quad (4)$$

where

K_{xx} is hydraulic conductivity in the x-principal direction (L/T),

K_{yy} is hydraulic conductivity in the y-principal direction (L/T),

K_{zz} is hydraulic conductivity in the z-principal direction (L/T),

K_{xy} is hydraulic conductivity in the x-principal direction due to the gradient in head in the y-direction (L/T),

K_{xz} is hydraulic conductivity in the x-principal direction due to the gradient in head in the z-direction (L/T), and

K_{yz} is hydraulic conductivity in the y-principal direction due to the gradient in head in the z-direction (L/T).

By aligning the model grid—that is, aligning the Cartesian principal axes (x, y, z) with the general flow directions—the off-diagonal terms approach zero. If the model grid is rotated so that the cell faces align with the principal directions of flow (fig. 3*B*), then mathematically, K changes to equation 5:

$$K = \begin{bmatrix} K_{xx} & 0 & 0 \\ 0 & K_{yy} & 0 \\ 0 & 0 & K_{zz} \end{bmatrix} \quad (5)$$

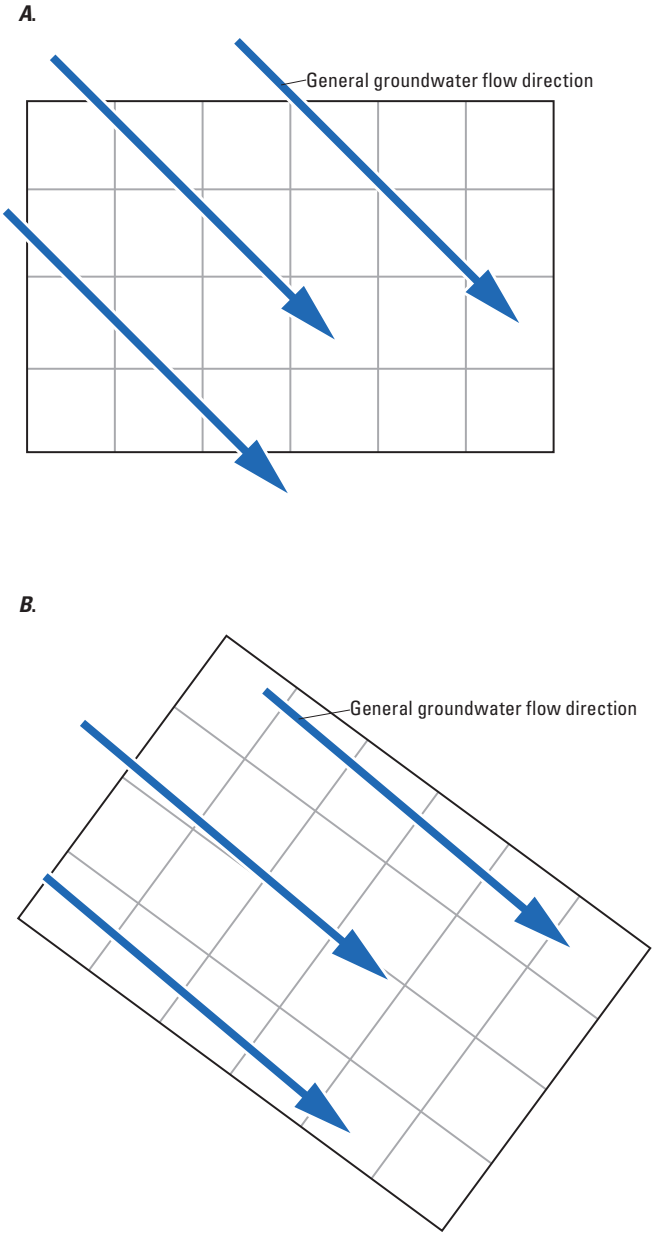


Figure 3. Example model grids that differ by orientation to groundwater-flow diration: *A*, not aligned with the general groundwater flow, and *B*, aligned with the general groundwater flow.

A single model cell is a representative elementary volume (REV) of a porous medium such as an aquifer, aquitard, or aquiclude. Model cells are grouped by model layer, and each layer is defined as confined or convertible. The difference between these two layer types is the way that aquifer properties are treated when the hydraulic head is less than the model cell's top elevation—specifically, how the model cell's saturated thickness is calculated. The saturated thickness is the vertical thickness of the model cell in which the pore spaces are filled (saturated) with water. Given that the cell's hydraulic head represents the saturated water level, if the head is at or above the top of a model cell, it is considered fully saturated and has a saturated thickness equal to the cell thickness. For a confined layer, the saturated thickness is independent of hydraulic head and always equal to the model cell's vertical thickness—that is, the model cell is always fully saturated. A convertible layer acts as a confined or unconfined aquifer. If the hydraulic head is above the model cell's top elevation, then the convertible cell functions identically to a confined cell—that is, the model cell is fully saturated. If the hydraulic head drops below the top of a convertible model cell, but is above the cell bottom, then the saturated thickness is equal to the head minus the elevation of the bottom of the cell (that is, vertical saturated distance within the cell). If the head drops below the model cell, then the cell becomes "dry" with zero saturated thickness and no longer contributes to groundwater flow. The confined or convertible designation of a model layer affects the flow properties that are dependent on transmissivity—which is equal to the hydraulic conductivity multiplied by the saturated thickness—and storativity—which is equal to the specific storage multiplied by the saturated thickness.

The storativity represents the volume of water released from storage per unit decline in hydraulic head in the model cell per unit area of the model cell. Confined layers require specifying a specific storage (S_s, sometimes called volumetric specific storage), which is the volume of water that a model cell releases from storage per volume of model cell per unit decline in hydraulic head. Specific storage represents water that can be removed from a model cell without changing the saturation—that is, the model cell remains fully saturated yet releases water.

Convertible layers require specifying a specific storage and specific yield (S_y). When a convertible layer is fully saturated, then specific storage is used as its storage property; if it becomes unsaturated, then specific yield is used. Specific yield is always substantially larger than specific storage. Specific yield represents the volume of water that can be drained by gravity from a fully saturated model cell relative to the volume of the model cell. For example, if the hydraulic head in a model cell changes from the cell's top elevation to the cell's bottom elevation, then the specific yield is the cell's gravity-drained volume of water divided by the volume of the model cell. Convertible layers can increase simulation runtime, so it is common to initially develop a model in which all layers are defined as confined. For layers defined as confined that are known to always be partially saturated (that is, not fully saturated, unconfined conditions), then the specific yield can be approximated by specifying a specific storage equal to the specific yield divided by the cell thickness. This is a common modeling technique that takes advantage of the speed of the confined formulation but represents the unconfined storage response.

Groundwater flow in model layers designated as "confined" is solved using the confined, anisotropic, saturated groundwater-flow equation. This governing equation is developed by combining the continuity equation with Darcy's law, and it can be expressed by the following parabolic partial differential equation (Willis and Yeh, 1987; Boyce and Yeh, 2014):

$$\frac{\partial}{\partial}\left(\mathrm{K_x}\frac{\partial \mathrm{h}}{\partial \mathrm{x}}\right)+\frac{\partial}{\partial}\left(\mathrm{K_y}\frac{\partial \mathrm{h}}{\partial \mathrm{y}}\right)+\frac{\partial}{\partial}\left(\mathrm{K_z}\frac{\partial \mathrm{h}}{\partial \mathrm{z}}\right)\pm \mathrm{W}=\mathrm{S_s}\frac{\partial \mathrm{h}}{\partial \mathrm{t}} \quad (6)$$

where

x, y, z	are distances in the respective Cartesian coordinate directions (L);
t	is the time (T);
h	is the hydraulic head at locations x, y, and z (L);
K_x, K_y, K_z	is the hydraulic conductivity in the x, y, and z directions (L/T);
W	is a volumetric flux per unit volume in or out of the system (1/T); and
S_s	is the specific storage (1/T).

Groundwater flow in model layers designated as "convertible" is solved with the confined flow equation (eq. 6) or the unconfined flow equation (eq. 7), depending on whether the model cell is fully saturated. The confined flow equation is used when the hydraulic head is above the model cell's top (fully saturated), and the unconfined flow equation when the hydraulic head is below the model cell's top but above the cell's bottom (water table conditions, variable saturated thickness). Because unconfined flow has a variable saturated thickness, its governing saturated groundwater-flow equation requires a slight modification to its confined-flow counterpart. The key change is that the upper boundary condition is now a free surface, so it must be included in the governing equation by integrating across the z-direction using Leibniz's integral rule. This yields the following equation (Willis and Yeh, 1987; Boyce and others, 2015):

$$\frac{\partial}{\partial}\left(\mathrm{K_x h}\frac{\partial \mathrm{h}}{\partial \mathrm{x}}\right)+\frac{\partial}{\partial}\left(\mathrm{K_y h}\frac{\partial \mathrm{h}}{\partial \mathrm{y}}\right)\pm \mathrm{W}=\mathrm{S_y}\frac{\partial \mathrm{h}}{\partial \mathrm{t}} \quad (7)$$

where

S_y	is the specific yield (-).

The way vertical flow is handled in MODFLOW between an unconfined model cell and a confined model cell depends on the flow package. Each of the packages use Darcy's law to calculate the flow in the vertical direction, but differ in how they calculate the distance, D, between the model cells. The two most commonly used flow packages are Layer Property Flow package (LPF, table 1) and Upstream Weighting (UPW, table 1). The LPF package varies the distance by the saturated thickness of the upper, unconfined cell; that is, vertical conductance varies with saturated thickness. The UPW package mimics confined, vertical flow by holding the distance constant; that is, vertical conductance remains constant. It is important to note that MF-OWHM2 allows using the Newton-Raphson Groundwater Formulation (NWT solver, table 1) with the LPF package. Because the NWT solver is designed to only work with UPW, if the NWT solver is selected, the LPF package input is translated to UPW, resulting in a constant vertical conductance for convertible layers despite using the LPF package—that is, LPF acts like UPW when the NWT solver is used. Conversely, the UPW package input is translated to LPF if a solver other than NWT is used.

A limitation of the MODFLOW-2005 framework is that water removed by a package is removed from the model domain rather than connected to another package or process. For example, the WEL package extracts by pumping water from groundwater storage, and that water is removed from the model domain. By contrast, in MF-OWHM2 the extracted water often is applied to the surface as irrigation, where it can become groundwater recharge, runoff to streams, or satisfy some of the potential evapotranspiration demand.

The groundwater flow equation in MODFLOW-2005 was advanced by including vertical, unsaturated flow in the unsaturated-zone flow package (UZF; Niswonger and others, 2006), which solves a one-dimensional approximation to the Richards equation using the method of characteristics for solving partial differential equations.

Improvements to these fundamental components and features of the basic MODFLOW-2005 groundwater-flow process and associated boundary-condition packages are summarized in the appendixes of MF-OWHM (Hanson and others, 2014d). New features that enhanced the MODFLOW-2005 flow process and its basic packages are documented in the section "Fundamental MODFLOW Enhancements" and the associated new input-data structures are summarized in appendixes 1, 2, and 3.

Seawater Boundary Representation—Equivalent Freshwater Head

Coastal groundwater basins with an ocean boundary generally contain a density related pressure difference between seawater and freshwater. An ocean boundary can be simulated with the GHB Package by a boundary head (BHead) set to the equivalent freshwater head of the sea level. Because of additional dissolved solids, sea water has a higher density than freshwater. Consequently, seawater hydraulic head represents a larger pressure than freshwater hydraulic head. The difference in seawater density (and viscosity) also affects hydraulic conductivity (K) compared to freshwater, but it is assumed to be negligible.

Using the seawater pressure and density, equation 2 can be recast to represent the seawater hydraulic head (which is equal to the sea level):

$$h_{sw} = \frac{P_{sw}}{\rho_{sw} g} + Z \tag{8}$$

where

h_{sw} is the seawater hydraulic head (L);
P_{sw} is the gauge pressure, as a result of seawater, at elevation Z (M/LT^2);
ρ_{sw} is the seawater density (M/L^3);
g is the acceleration due to gravity (L/T^2); and
Z is the elevation that the gauge pressure is measured at (L).

MODFLOW uses the change in freshwater hydraulic head in Darcy's law to determine groundwater flow. Because of this, an ocean boundary condition in MODFLOW requires the sea level (h_{sw}) be converted to an "equivalent freshwater head" (h_{fw}). To derive an equivalent freshwater head for seawater, MODFLOW recasts equation 8 in terms of seawater pressure (eq. 9):

$$P_{sw} = \rho_{sw} g \left(h_{sw} - Z \right) \tag{9}$$

The seawater pressure (eq. 9) is then substituted for pressure in the freshwater hydraulic head equation (eq. 2) to yield the following:

$$h_{fw} = \frac{P_{sw}}{\rho_{fw} g} + Z \tag{10}$$

$$h_{fw} = \frac{\rho_{sw} g \left(h_{sw} - Z \right)}{\rho_{fw} g} + Z \tag{11}$$

This is simplified to produce the equivalent freshwater head equation:

$$h_{fw} = \frac{\rho_{sw}}{\rho_{fw}} h_{sw} - \left(\frac{\rho_{sw} - \rho_{fw}}{\rho_{fw}} \right) Z \quad (12)$$

where

h_{fw} is the seawater's equivalent freshwater hydraulic head at elevation Z (L),
ρ_{sw} is the seawater density (M/L^3),
ρ_{fw} is the freshwater density (M/L^3), and
Z is the elevation point where the equivalent freshwater head is calculated (L).

To specify an ocean boundary condition with the GHB, the sea level is converted to an equivalent freshwater head at the model cell's center. The density of seawater is not constant and changes with depth, but is typically assumed to have an average value of 1,025 kg/m^3. Similarly, the density of freshwater is assumed to be 1,000 kg/m^3. An ocean boundary head can be determined using equation 12 and these two densities:

$$BHead = 1.025 \times h_{sw} - 0.025 \times Z_p \quad (13)$$

where

BHead is the GHB ocean boundary head (L),
h_{sw} is the ocean sea-level elevation (L), and
Z_p is the elevation at the center of the model cell (L).

There are three methods to evaluate seawater intrusion using MF-OWHM2. The first is to assume purely advective seawater transport and use a freshwater equivalent for ocean boundary heads (eq. 11); the seawater intrusion flow pathways can then be determined using particle tracking with MODPATH (Pollock, 2016) or MODPATH-OBS (Hanson and others, 2013). The second method is to assume a sharp interface (no concentration mixing) between the seawater and freshwater flow and use the Seawater Intrusion (SWI) Process (Bakker and others, 2013) to simulate the location of the sharp interface. SWI requires that a GHB specify BHead as an equivalent freshwater head (eq. 11), but defines Z as the model cell's top elevation. The third option uses MF-OWHM2 with the Link-MT3DMS (LMT) Package to generate flow input for MT3DMS (Zheng and Wang, 1999) or MT3DMS-USGS (Bedekar and others, 2016) for a full seawater transport simulation.

Surface-Water Processes

Interactions between groundwater and surface water are common near surface-water bodies, such as streams, wetlands, and lakes, and also for irrigated landscapes. When groundwater elevations are below those of a surface-water system, surface water drains into the groundwater. When groundwater elevations are above those of a surface-water system, groundwater discharges to surface water. This coupling results in a unique set of dynamics between the groundwater and surface-water system that can be characterized through simulation. For example, groundwater pumping can lower the groundwater level enough that nearby streambed leakage reduces streamflow; conversely, recharge ponds can raise groundwater elevations, increasing streamflow. Surface-water flows can be controlled by an upstream reservoir, augmented by imported irrigation water, retained locally (for example, urban runoff basins and farm ponds), or exported for off-stream storage, which then could contribute to groundwater recharge, evaporative losses, or be released as local or transbasin-diverted streamflow. Simulation is a common approach for developing an understanding of these complex dynamics and associated temporal lags between stresses and responses.

In the MF-OWHM2 model, rainfall-runoff processes are considered in a conjunctive-use context. Surface-water flows are computed using the spatial and temporal distribution of precipitation, the excess irrigated water that becomes runoff, surface-water diversions and deliveries, subsurface drain flows, and groundwater discharge to the surface. Runoff is either directly delivered or prorated throughout a stream network, where it can be routed through stream channels and further interact with the groundwater system. Recharge that leaves the soil root zone either is passed to the Unsaturated-Zone Flow (UZF) package to become delayed recharge or is routed instantaneously to the water table (or uppermost simulated model cell).

Streamflow routing is represented as a head-dependent-flow boundary condition in MODFLOW; two packages are available to offer a range of routing options. These packages account for flows in channels, elevation of water in the streams (that is, stream stage), and two-way interactions between streams and groundwater. The greatest difference between these two packages is the manner in which each computes the stream stage. The SFR package (Niswonger and others, 2006) assumes the slope of the water surface is equal to the surface of the streambed (that is, kinematic assumption), whereas the SWR (Hughes and others, 2012) does not make this assumption. The simplifying assumption in the SFR provides a robust and computationally efficient representation of streamflow for most applications, but for complex systems where the kinematic assumption is invalid—such as tidal areas or otherwise inundated channels—the SWR package is more suitable.

Conceptually, the method of computing flow between streams and aquifers in SFR and SWR is the same as that used for the standard river package (RIV; McDonald and Harbaugh, 1988, chapter 6), but it is important to note that the RIV package is not recommended for a conjunctive-use simulation. Flow between streams and aquifers in the groundwater model is computed using Darcy's law and assuming uniform flow between a stream and aquifer over a given section of stream and corresponding volume of aquifer. This flow is computed as follows:

$$Q_L = \frac{K_{sb} w L}{m}\left(h_s - h_a\right) \quad (14)$$

where

Q_L is a volumetric flow between a given section of stream and volume of aquifer (L^3/T),
K_{sb} is the hydraulic conductivity of streambed sediments (L/T),
w is a representative width of stream (L),
L is the length of stream corresponding to a volume of aquifer (L),
m is the thickness of the streambed deposits (L),
h_s is the stream head computed as the stream depth plus elevation of streambed (L), and
h_a is the aquifer hydraulic head beneath the streambed (L).

In this formulation, transient leakage across the streambed could change depending both on the stream head and on the aquifer head that is calculated during each time step. The volume of water that seeps from a stream is calculated by multiplying the infiltration rate by the wetted area of the stream. The wetted area of the stream may be held constant or determined on the basis of stream cross-sectional dimensions, discharge, and stage. The relation between stage and discharge is calculated using Manning's equation. The SFR and SWR packages have both been previously described for MF-OWHM (Hanson and others, 2014d); new and updated features in MF-OWHM2 are summarized in appendixes 1, 2, and 3.

Landscape Processes

Landscape processes in MF-OWHM2 involve the simulation of consumption, use, and unchannelized flow of water across the land surface and vertically from the bottom of the root zone of plants—or soil zone—to the top of agricultural or natural vegetation. The development of landscape processes was an important change from the traditional MODFLOW approach, in that processes such as evapotranspiration (ET) and other demands for water from various sources were driven by the estimation of the consumption, movement, and even reuse of water across the landscape. This was initially implemented through the FMP (Schmid, 2004; Schmid and others, 2006; Schmid and Hanson, 2009a; Hanson and others, 2014d). The features of FMP and their connection with other features and processes broadened the couplings between landscape processes and related groundwater and surface-water processes. The fundamental advance contributing to the FMP was the capability to incorporate "flow-dependent" flows, whereby packages and processes can pass flows from one model feature to another on the basis of estimated or specified water demands and supplies. This also facilitated simulation of water management in a demand-driven and supply-constrained context throughout the entire model framework (Hanson and others, 2010; Hanson and Schmid, 2013). An overview of the core concepts for the landscape simulation in the FMP is provided in appendix 4 ("Consumptive Use and Evapotranspiration in the Farm Process") and appendix 5 ("Landscape and Root-Zone Processes").

Landscape modeling improves the understanding of landscape, groundwater, and surface-water interactions. The Landscape process couples irrigation-water losses, precipitation, imported water, and evapotranspiration to groundwater flow and surface-water flow. The Landscape process simulates surface-water diversions and estimates the unknown pumpage to meet irrigation demands, infiltration to groundwater, and runoff return to surface water. In cases where groundwater is shallow enough for crop-root uptake, the Landscape process accounts for this source, thereby reducing the demand for diverted surface water and groundwater pumping. Lastly, surface water is coupled indirectly to receive deep groundwater, representing runoff of irrigation water extracted from deep groundwater wells. Incorporating landscape processes provides understanding of the feedback and interrelations between the land surface, groundwater, and surface-water flows.

Land Subsidence

The process of aquifer-system compaction, and associated land subsidence, can have important effects on the surface-water and groundwater systems. Representing these processes in simulations can help to avoid misleading results. Land subsidence is an often-overlooked hazard and an environmental consequence of ground-water withdrawal (Hoffmann and others, 2003; Galloway and others, 1999). The arid and semi-arid regions that contain compressible, unconsolidated basin-fill deposits are especially vulnerable to subsidence because of a reliance on groundwater in dry climates with limited surface-water supplies. Coastal regions may also be at risk if they are underlain by unconsolidated, compressible coastal plain and shallow-marine sediments (Hoffmann and others, 2003). Land subsidence can result in environmental consequences that include damage to engineered structures, such as buildings, roadways, pipelines, aqueducts, sewerages, and groundwater well casings; earth fissures; increased coastal and riverine flooding; and loss of saltwater- and freshwater-marsh ecosystems.

With respect to the groundwater system, inelastic aquifer compaction yields a one-time, but often substantial, source of water. Simulation of the compaction process ensures that this contribution from groundwater storage is calculated appropriately; simulation of the same system without accounting for water released by compaction would result in the incorrect estimation of elastic storage properties. For example, without simulation of aquifer compaction, the water released from inelastic compaction would result in an overestimation of the aquifer-storage properties (specific storage and specific yield). Simulation of subsidence in MF-OWHM2 is limited to one-dimensional, vertical compaction. For MF-OWHM2 simulation, compaction refers to the change in vertical thickness that accompanies changing stresses on the aquifer system.

MF-OWHM2 has two packages, SUB and SUB-WT packages (Hoffmann and others, 2003; Leake and Galloway, 2007) that simulate aquifer compaction and subsidence. A limitation is that only one subsidence package may be used during a simulation. MF-OWHM2 SUB also includes the improvements that enable separate accounting of elastic and inelastic subsidence (Schmid and others, 2009), parameterization of selected subsidence parameters (Hanson and others, 2014d) and the option to simulate deformation-dependent flows using the SUB-link feature (Hanson and others, 2014d; Schmid and others, 2014). The SUB-link feature allows dynamic modification of the vertical model discretization in response to aquifer compaction and expansion, and it correspondingly adjusts any packages affected by land-surface elevation change or aquifer thickness. For example, the SUB-link alters the elevation of the stream beds in SWR, which can change the streamflow rate or reverse the flow direction.

Reservoir Operations

Reservoir operations represent human influence on natural river systems through retention and release of surface water for multiple downstream purposes. Storage reservoirs provide a buffer during dry periods, enabling release of water for irrigation, domestic consumption, other uses, and for environmental flows. Reservoir operations typically are based on a set of rules associated with downstream water uses and other considerations, including flood protection, environmental flows for fish passage or habitat maintenance, stream and reservoir habitats and recreation, and hydro-electric power production. At the time of this publication, MF-OWHM2 does not simulate reservoir power production directly (Ferguson and others, 2016).

The altered flow regime that results from reservoir operations influences the timing of interactions among surface water, groundwater, and the landscape. For example, if a reservoir releases water to meet irrigation demand during a period when insufficient flow in a river is typical, then some of the released water can infiltrate to groundwater during transit to the irrigation point.

Recent work on methods for incorporating reservoir operations in the MODFLOW framework involved linking MODSIM software (Labadie and Larson, 2007) with MODFLOW-NWT to simulate reservoir operations using the SFR package (Morway and others, 2016). An iterative linking between the two simulators allowed MODSIM to adjust releases to groundwater and flow conditions simulated by MODFLOW-NWT. Iterative, in this case, refers to the adjustments of MODSIM and MODFLOW at every model solver iteration (called the outer iteration).

The ability to dynamically simulate reservoir operations directly in MODFLOW, such that operations are tightly coupled to streamflow gains and losses and downstream demand, has become increasingly important to reservoir managers. This led to joint development by the USGS and Reclamation of the Surface-Water Operations Process (Ferguson and Llewellyn, 2015; Ferguson and others, 2016). The Ferguson and others (2016) Surface-Water Operations Process simulated single-reservoir system operations that efficiently released water to meet downstream irrigation demand; this demand was either user specified or calculated by the FMP dynamically.

Conduit Flow

The Conduit Flow Process (CFP) simulates dual-porosity aquifers that can be mathematically approximated by coupling the traditional groundwater-flow equation with a discrete network of cylindrical pipes (CFPM1) or inserting a preferential-flow layer (CFPM2) that uses a hydraulic conductivity based on turbulent flow to simulate turbulent horizontal-flow conditions. The pipes can represent dissolution features or fractures and can be fully saturated or partially saturated under laminar or turbulent conditions. The preferential-flow layers may represent (1) a porous media through which flow is turbulent flow or (2) horizontal preferential-flow zones in an aquifer for which the explicit geometry of the secondary porosity is not well defined. The CFP simulates steady-state and transient hydraulics of the dual-porosity system (Shoemaker and others, 2008).

The CFP was initially developed as an individual variant of MODFLOW (MODFLOW-CFP; Shoemaker and others, 2008) to address the issues of preferential flow in karst aquifers; however, the coupling with the groundwater-flow process was limited. The concepts behind the new upgrades to CFP are summarized in appendix 7 (“Conduit Flow Process Updates and Upgrades”); the associated new input data structures are summarized in appendix 8 (“Conduit Flow Process Input Data”).

Optional Use of Separate Rainfall–Runoff and Hydraulic Models

It can be advantageous to develop, along with a MF-OWHM2 model, companion models that use other simulation approaches. Such companion models may be valuable for providing boundary conditions, improving the understanding of areas beyond the MF-OWHM2 domain, or providing calibration targets (such as streamflows in areas that are ungaged).

The domain of MF-OWHM2 models typically represents aquifer systems in valley settings; thus, the domain boundaries are often near the foothills of a mountain range. A rainfall–runoff model that includes the MF-OWHM2 domain and extends into the upland watersheds can provide estimates of boundary inflow from the upland watersheds to the MF-OWHM2 simulation domain. In such a case, runoff from the rainfall–runoff model is applied as a boundary inflow to the MF-OWHM2 model at the stream reaches that intersect the model boundary. The intersection at which flows are passed from the upland watershed model to the MF-OWHM2 model is called a "pour point" and is typically where SFR intersects the model-domain boundary. The SFR, or other packages or processes, then simulates the streamflow in the MF-OWHM2 model domain.

A rainfall–runoff model may also provide potential or reference evapotranspiration estimates (ET_{ref}) and supply streamflows as calibration targets in regions of the MF-OWHM2 model where measurements are sparse (that is, ungaged streams). If ET_{ref} is available, then it can be used as part of the FMP input to provide a more accurate calculation of actual evapotranspiration. Because the temporal resolution of rainfall–runoff models (hourly to daily) is finer than that of a MF-OWHM2 model (typically more than a day), the detailed streamflow in ungaged regions of the model can be aggregated and used as a calibration target. Conversely, the MF-OWHM2 model can provide better estimates of baseflow to the rainfall–runoff model. This may be particularly important if the rainfall–runoff model is used for sediment transport and flood prediction.

Companion rainfall–runoff models that have been developed along with MF-OWHM2 models are the Basin Characterization Model (BCM; Flint and others, 2013; Flint and Flint, 2014), Hydrologic Simulation Program—Fortran (HSPF; Donigian and others, 1995; Bicknell and others, 2001), and the Precipitation-Runoff Modeling System (PRMS; Markstrom and others, 2015). The BCM simulates the interactions of climate (rainfall and temperature) with empirically measured landscape attributes, including topography, soils, and the underlying geology. It is a grid-based model that calculates the water balance in a given watershed. Some BCM-simulated datasets are publicly available for download (https://www.usgs.gov/centers/ca-water/science/basin-characterization-model-bcm?qt-science_center_objects=0#qt-science_center_objects). The HSPF simulates watershed-hydrology and water-quality processes on pervious and impervious land surfaces and in streams and well-mixed impoundments. The U.S. Environmental Protection Agency (EPA) maintains HSPF and a variety of databases at its Basins webpage (U.S. Environmental Protection Agency, 2019, https://www.epa.gov/ceam/basins-framework-and-features). The PRMS is a deterministic, distributed-parameter modeling system that simulates streamflow and general watershed hydrology in response to climate and land use. USGS maintains this model and access to associated datasets at the Modeling of Watershed Systems (MoWS) webpage (U.S. Geological Survey, 2019, https://wwwbrr.cr.usgs.gov/projects/SW_MoWS/index.html).

Where MF-OWHM2 is applied to simulation of regions that require detailed river hydraulics beyond the capabilities of SFR and SWR, the use of an independent river hydraulics simulator could prove beneficial. MF-OWHM2 can, in turn, provide an accurate base-flow estimate for the hydraulic simulator, or a more formal two-way coupling can be developed. Examples of hydraulic flow software include the Sedimentation and River Hydraulics—Two-Dimensional model (SRH-2D; Lai, 2008, 2010) and Hydrologic Engineering Center, River Analysis System (HEC-RAS; U.S. Army Corps of Engineers, 2016). The SRH-2D is a two-dimensional hydraulic, sediment, temperature, and vegetation model for river systems. It is capable of simulating flows involving in-stream structures, bends, perched rivers, side-channel and agricultural returns, and braided-channel systems. Similarly, the HEC-RAS can simulate a network of channels, a dendritic system, or a single river reach.

Supply and Demand Framework

One of the core goals of MF-OWHM2 is representing water supply, management, and use in a demand-driven and supply-constrained framework. Water supply can include surface water, groundwater, precipitation, imported water (non-routed deliveries), reclaimed water, or a combination of these and other sources, such as desalinated water. Water demands can originate from irrigation needs; managed aquifer-recharge operations; environmental needs; water-supply needs for domestic, municipal, and industrial uses; power production; and other uses.

The demand-driven and supply-constrained framework provides a physically based context for simulating water management. The physical constraints of supply, such as maximum-capacity surface-water flows or groundwater elevations and well-production capacity, are combined with management constraints, such as allocations, water rights, and administrative pumpage restrictions. Similarly, drivers of demand define the amount of water needed, such as irrigation efficiency, soil and water salinity, and how the landscape is used. The demand-driven and supply-constrained framework allows for the dynamic simulation of the landscape demands that can be satisfied with supplies from precipitation, surface water, groundwater, and imported water.

The dynamic representation of management constraints is important for scenario analyses involving changes in climate and associated climate variability; land use; socioeconomic conditions; governance; and management actions. The dynamic representation of demand and supply necessitates that demand be estimated as part of the simulation. Previous simulation practices required pre-calculating demand externally from the simulation model (the so-called "spreadsheet method") and then specifying those demands directly as surface-water diversions and groundwater pumping. Spreadsheet methods directly specify surface-water diversions and groundwater pumping, resulting in a loss of the dynamic response that real, managed systems have. These methods are unable to dynamically change surface-water diversions and groundwater pumping in response to changes in land usage, current groundwater and surface-water conditions, climate variability, reservoir operations, and availability of external water sources.

The ability to dynamically represent management constraints in an integrated hydrologic model is a unique feature of MF-OWHM2 that makes it well suited for analysis and planning of conjunctive-use systems. FMP, attempts to satisfy the demand with available water sources. Water sources (the supply) include natural sources (precipitation and groundwater uptake) as well as supplies of groundwater pumpage, surface water, imported and reused waters. The supply can be constrained by physical infrastructure (such as well- or diversion-capacities), and management constraints (such as allotments or water rights).

Local landscape demands are estimated by defining accounting regions in the model domain that are called a water-balance subregion (WBS). The WBS was originally called a Farm in the first release of the FMP, so the term WBS and "Farm" are used interchangeably in this report. For example, the number of WBSs are defined by the keyword **NWBS** or **NFARM**. For a given WBS, its total demand is balanced with the available water supplies. One or more types of native vegetation, natural vegetation, or agricultural crops can be simulated as a landscape type referred to as a "Crop." A Crop can represent any land-use type for which consumptive use of water can be represented as the sum of evaporation and transpiration of any combination of groundwater uptake, precipitation, and irrigation (appendix 4 and 5). Crops generally have additional demands associated with runoff or infiltration of irrigation water (that is, the additional irrigation requirement resulting from inefficient irrigation practices). For this report, the term consumptive use (CU) is defined as the total water consumed by a land-use type (a defined "Crop"), including natural and anthropogenic sources. When CU is in reference to vegetation or agricultural crops, then it is synonymous with actual evapotranspiration. Additional irrigation water applied to meet the CU demand, needed because of inefficiencies in land management and irrigation methods in an area, is part of the farm delivery requirement.

The name "Farm" is potentially misleading because a WBS does not necessarily refer to a specific agricultural farm, nor does a "Crop" strictly represent an agricultural crop. A WBS represents a region that has a common set of supply and demand calculations applied. This region does not have to be contiguous and can be defined for any surface model cell in the model grid (that is, it is not required to have active groundwater cells beneath it). A WBS can range from a municipality, to a large collection of agricultural fields, to a ranch within a single model cell—anything that has common sources of water supply. During a simulation, each WBS aggregates its associated surface model cells to compute a collective water demand that is then satisfied by common sources of water supply. These sources of water do not have to be in the WBS. For example, an MNW2 well that serves a WBS can be anywhere in the active simulation domain.

A Crop may represent any type of native or natural, agricultural, urban, industrial, or other land use that consumes water. Crops may also represent solely evaporative land-uses, such as a water body, rock quarries, or dairy farm (a dairy farm would use evaporation losses to represent general consumption from care of the herd, such as drinking and washing of cows). Another use for Crops is to represent a wastewater-treatment plant, which would have a specified water-delivery requirement, zero consumptive use (no evapotranspiration), and all delivered water that is discharged to a river set to become runoff.

Water-Balance Subregions

The design of the water-balance subregion (WBS) for a valley or project region is an important consideration for the analysis of conjunctive-use issues. Figure 4 indicates that the questions to be addressed (step 1) and extent of the analysis region (step 2) guide the division of the region into WBSs (step 3a) on the basis of conjunctive-use issues and the present and future framework of supply. A WBS can represent virtually any type of water use (or uses) for any specified area; however, in order to serve the broader understanding of conjunctive use, it is recommended that WBSs are defined by areas that have consistent water supplies and uses (fig. 4, steps 3b, c). The WBSs can also represent areas that have consistent local elements of policy, governance, or treaties (fig. 4, step 3). To achieve this, the design of WBSs benefits from consultation with local water purveyors or governing bodies that administer or control surface water, groundwater, or water-dependent features such as riparian habitat. The delineation of WBSs is also linked to the design of the land-use categories to be used to represent demand in each WBS (fig. 4, step 4).

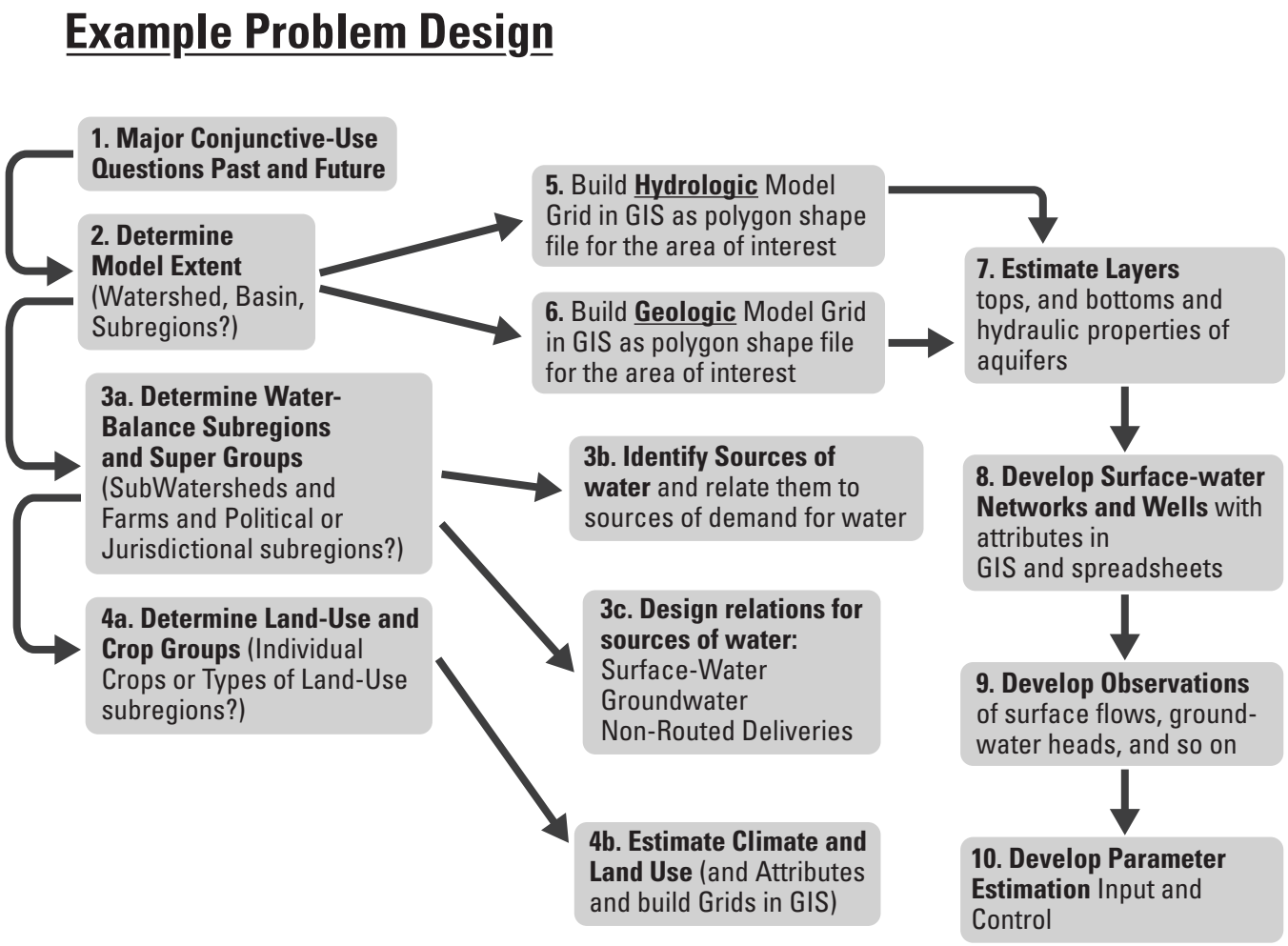


Figure 4. Example of the steps in a workflow process for developing a conjunctive-use model design.

A WBS can represent an individual spatial location (a single model cell) or a multi-cell area where water is consumed or intermittently stored from a common or shared supply or from any potential combination of water sources. Spatially contiguous and non-contiguous multi-cell subregions can be represented by WBSs. Examples of this were included in the USGS Central Valley (of California) Hydrologic Model (CVHM), in which the WBSs represent state-delineated depletion accounting units (DAUs) that each receive one or more surface-water diversions (Faunt, 2009; Faunt and others, 2009; Hanson and others, 2012). Similarly, WBSs can be grouped to represent all areas and beneficiaries that receive water from a common supply, such as a reservoir or groundwater basin. An example of this is an application of the FMP and SFR to the Rincon and Mesilla Valleys in southern New Mexico and western Texas, which includes parts of the federal Rio Grande Project (Hanson and others, 2019). Water supplies and demands for each irrigation district and for the Rio Grande Project as a whole were analyzed by aggregating fine-scale WBSs representing areas of the water-delivery service area. Alternatively, a WBS can represent a mixture of parts of irrigation districts, municipalities, and subwatersheds, as was done for California applications in Pajaro Valley (Hanson and others, 2014a) and Borrego Springs (Faunt and others, 2015). The WBS can also represent individual locations where managed aquifer-recharge operations or supplemental or blended-groundwater wells are augmenting supply and demand; this was done for Pajaro Valley, California (Hanson and others, 2014a). A WBS can be oriented toward specific native, natural, or agricultural vegetation types as completely discontiguous sets of cells heterogeneously distributed across a region.

Finally, WBS subregions can change through time as administrative boundaries, land ownership, land use, and water-related infrastructure change and induce corresponding changes in water demands, water rights, and well and diversion locations. This can require adjusting the extent and shape of WBSs through the course of a model simulation. For example, for the simulation of the Cuyama Valley in California (CVIHM), it was necessary to represent the transition from about a dozen large ranches and native lands parcels in the 1940s to more than 80 smaller parcels by 2010 (Hanson and

others, 2014c). The change in the size and number of WBSs can be represented in several ways, including using results of land-use or economic models, known changes in land ownership or zoning, or projections of land use into the future. The latter method was applied to the Sonoma Valley (Andrew Rich, Sonoma County Water Agency, written commun., 2015) to assess the potential expansion of agricultural lands.

Ultimately, for purposes of jurisdiction, decision making, or physical infrastructure, groups of WBSs may be combined into larger subregions, or super groups of WBSs, that facilitate analyses appropriate to such scales. This allows the analyses of water demand and use scenarios for developing best management practices, sustainability, or adaptation plans. Examples of combining local-scale WBSs in large, regional-scale groups include applications for California's Central Valley, Pajaro Valley, and Cuyama Valley.

Land-Use Types

A fundamental function of FMP is the calculation of consumptive use for each type of land use. For MF-OWHM2, potential consumptive use (potential CU) is defined as the amount of water from any source needed to meet a land use's (or Crop's) water demand as evaporation and transpiration. It is synonymous with a crop's potential evapotranspiration (PET), and consumption can range from solely transpiration (potential transpiration) to only evaporation (potential evaporation). If a land use has access to irrigation, then after the consumption from natural sources (groundwater-root uptake and precipitation), any remaining CU is satisfied with the available irrigation water. Appendix 4 provides a detailed explanation of the consumptive-use calculations.

As described previously, the term "land use" in this report is synonymous with the word "Crop" and may represent different kinds of consumptive uses on the landscape. The term "Crop" can be used either to aggregate multiple crops into one group of similar properties or to split a crop type into multiple crops of differing properties. An example of aggregation would be to lump different sets of berries—such as blackberries, blueberries, and raspberries—together as one crop having a common set of properties. Crop classification is used to nest groups of input properties for a given land use. Crops typically have a common user-specified consumptive use or crop coefficient. Additional properties associated by Crop are whether it is irrigated, irrigation efficiency, fraction of delivery losses from irrigation and precipitation, and the ratio of a basal crop coefficient to crop coefficient. A Crop may also have a specified additional irrigation demand. The additional demand can be used to represent urban demand or additional irrigation for salinity flushing.

The geographic location and area that a Crop occupies affects its consumptive use because of the spatial variability in climate conditions, crop coefficients, and reference evapotranspiration. A Crop may occupy an entire model cell and be specified as a two-dimensional array of Crop identities (IDs), or there can be multiple Crops in a single cell, the footprints of which can be specified as fractions of the model cell. The ability to simulate multiple Crops in a model cell provides a more accurate representation of the land use but has the additional computational expense of a more complex input dataset.

Virtual Crops can be used to represent a specific demand or non-plant-based consumption, such as urban demand, dairies, lumber mills, or rock quarries. Non-plant-based consumption requires that no root uptake of groundwater is included in the consumptive use—this is specified by setting the FMP keyword **`GROUNDWATER_ROOT_INTERACTION`** to `1` for the Virtual Crop. By removing groundwater-root uptake, the Virtual Crop's CU is only fulfilled by irrigation and precipitation. Typically, non-plant-based crops' CU are defined as either having solely transpiratory or only evaporative consumptive demand. Two examples of transpiratory Virtual Crops are irrigation demand for urban consumption or drinking water for a dairy. Examples of purely evaporative consuming Virtual Crops are rock quarries or animal agriculture, such as livestock feedlots, dairies, or poultry farms.

Any Crop may have a specified additional irrigation demand using the FMP keywords **`ADDED_CROP_DEMAND`** and **`ADDED_DEMAND`**. This additional irrigation water is not consumed by the crop, but instead becomes either runoff or deep percolation (infiltration). Additional irrigation demand requires that the Crop is irrigated with an irrigation type to incorporate the irrigation type's irrigation efficiency. The actual amount of additional irrigation water applied is equal to the specified additional irrigation demand divided by the irrigation efficiency. Additional irrigation could represent an urban demand that becomes discharge from a wastewater treatment plant—this is done by specifying that all of the additional irrigation water becomes runoff that flows to a stream network.

As described previously, the demand calculations are based on the location of the WBS and its associated land uses (Crops). The association between a WBS and Crop is made by collocation wherever the WBS locations are coincident with the Crop's location. The locations are ordered to proceed from demand to supply, however; that is, the demand for water is calculated by the Crop's location (given reference evapotranspiration, crop coefficient, precipitation, and groundwater levels). Each crop is designated as irrigated or not, and if irrigated, associated with an irrigation type. Crops that are irrigated rely on their coincident WBS to provide water to meet their deficit consumptive use (the irrigation demand after consumption of groundwater uptake and precipitation). This allows for a single Crop type to be reused across multiple WBSs.

To illustrate the relationship among a WBS, Crop, and irrigation, consider a model that has two surface cells, where Cell 1 is associated with WBS 1, Cell 2 is associated with WBS 2, and both cells contain Crop 1 that is irrigated. The potential consumptive use of Crop 1 in Cell 1 is satisfied by WBS 1 and that of Cell 2 is satisfied by WBS 2. The Crop first consumes the natural sources of water (groundwater and precipitation) in the cell. If the natural sources are less than the potential consumptive use, then the crop has an irrigation demand. If Crop 1 in Cell 1 has an irrigation demand, then WBS 1 uses its sources of irrigation water to attempt to meet the irrigation demand. Similarly, if Crop 1 in Cell 2 has an irrigation demand, then WBS 2 uses its sources of irrigation water to attempt to meet the irrigation demand.

Water-Balance Subregion Supply Wells—The Farm Well

The supply and demand framework includes water supplied from groundwater-extraction wells. For each WBS, the model attempts to fulfill the total aggregated demand by imported water, surface water, and groundwater uptake, in that order; any remaining demand is supplied by groundwater pumping. This is how an unknown pumping component is calculated as part of the FMP (appendixes 4 and 5). For simplicity, the groundwater-extraction wells are henceforth referred to as Farm Wells. Farm Wells either are defined in the FMP input or are linked to a well that is defined in the Multi-Node Well package, version 2 (MNW2). Farm Wells are associated with a WBS and have a specified maximum pumping capacity. The sum of maximum capacities for all Farm Wells serving a WBS represents the total potential pumping capacity that supplies groundwater to meet the WBS irrigation demand.

Farm Wells either can be represented as a direct-sink term to groundwater flow—as is the WEL package—or can be linked to MNW2 to more accurately represent the well's construction and production potential. Farm Wells that mimic the WEL package extract or inject water at the FMP requested rate. This extraction rate can be curtailed if a saturated thickness smoothing function is applied. Smoothing decreases the pumping rate linearly as the head approaches the bottom of a model cell to represent the loss of production due to a cell going dry. This simulates the dewatering of the well and serves to minimize solution convergence issues associated with cells alternating between wet and dry. If a Farm Well is linked to MNW2, then the well's location and construction are defined by the MNW2 input, and the MNW2 well's maximum desired pumping rate (Qdes) is set by FMP (typically in response to demanded pumpage from a WBS). MNW2 determines the actual pumping rate on the basis of Qdes, the well construction and current aquifer conditions. If MNW2 cannot meet the FMP-specified Qdes to meet a WBS's demand, then FMP adjusts the WBS's supplies accordingly. Specifically, if there is insufficient supply from a Farm Well, a WBS may shift its demanded pumping to another linked MNW2 well or direct-sink well or the WBS may end up in a deficit irrigation situation because of inadequate well production (water supply does not meet water demand). For more details about farm wells, see the "Supply Well (Farm Well) Redesign and Implementation" section.

Supply and Demand Hierarchy for Surface-Water Operations

When using the Surface-Water Operations Process (SWO; Ferguson and others, 2016), the concept of the water-balance subregion (WBS) is extended to a supply and demand hierarchy. This hierarchy, from highest to lowest, is composed of a Project, District, Unit, and Beneficiary, and a WBS is considered a type of Beneficiary.

A Beneficiary consists of an entity served by a surface-water delivery that is controlled by a set of surface-water operation rules to meet the entity's water-consumption demand. That is, a Beneficiary is a water consumer that directly benefits from reservoir operations managing the surface-water delivery. Beneficiaries are nested in groups, called Units, that keep track of the water accounting. In particular, each Unit manages a diversion location that serves a group of Beneficiaries and keeps track of water consumption and bypassed flow. The Units are aggregated in groups, called Districts, that have a common water allocation to which the Unit water consumption is charged and, optionally, bypassed water is credited. Districts are aggregated in a water project that represents water storage from a single or set of reservoirs.

Figure 5 presents a conceptional example schematic of the supply and demand hierarchy. This example has a single reservoir that serves project water to three water districts. The reservoir releases water to a single main river channel that runs through the center of the figure. The first district served is District 1, which is on both sides of the main river channel. District 1 is composed of two units (Units 1 and 4), because the district is on both sides of the main river (each unit manages a diversion on either side of the main river). Within Units 1 and 4 are a set of Farmland beneficiaries that use the released project water for consumption. The second district, District 2, contains Unit 5, which manages the diversion that serves three beneficiaries (Municipal consumption, Farmland, and an Industrial Plant). Lastly, District 3 is composed of two units (Units 2 and 3). Unit 2 has the same main diversion location as Unit 1 from the main river system and receives water bypassed from Unit 1 to deliver water to two beneficiaries (Greenhouse and Farmland). Unit 3 from District 3 receives water from its own diversion, which delivers water to a single beneficiary (Farmland).

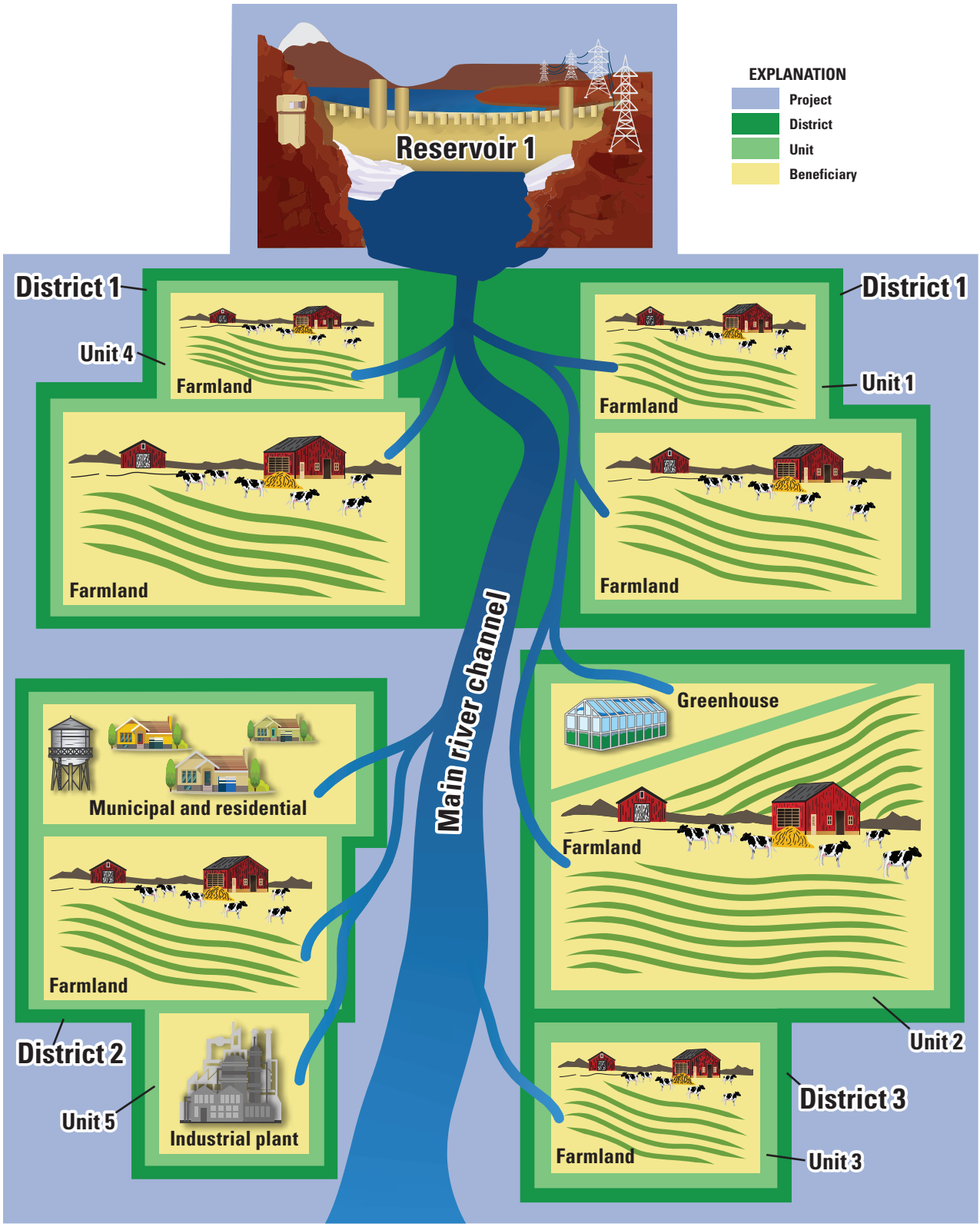


Figure 5. A conceptual example of the supply and demand hierarchy. The project area is served by a single reservoir that releases water to serve 3 districts, 5 units, and 10 beneficiaries (modified from Ferguson and others, 2016).

Self-Updating Model Structure

The MF-OWHM2 framework design implements the concept of "self-updating" models. The self-updating design refers to the conversion of the MODFLOW input-file structure to separates the spatial and structural input from the temporal input. This makes the input files clearer to the model developers, users, and reviewers; thus, the models are easier to use and update. This separation also allows automated programs to query databases, websites, or spreadsheets for new data to be downloaded and appended as updates to the model input files (for example, streamflows or specified pumping rates). This automated self-updating of the input files allows the simulation model to be re-used after its initial construction, thus increasing its potential longevity and value.

The construction of a self-updating model facilitates the use of data streams from land-based sensors and satellite-based imagery, which can provide estimates of properties that vary spatially and temporally. Examples of data streams that are temporally varying point measures are municipal pumping and gaged streamflow. Spatiotemporal varying examples are climate-related data (for example, temperature, precipitation, reference evapotranspiration) and sea-level gage extrapolation across an ocean boundary condition. Part of the MF-OWHM2 framework involved the progressive restructuring of the MODFLOW packages and processes—namely, SFR, FMP, GHB, DRT, MNW2, and WEL—to accommodate the variety of data streams as separate input files. This facilitates a more efficient transfer of these data streams into model input without having to restructure them.

Separation of Non-Spatial, Temporal Input (Point-Data Stream)

A point-based data stream is applied to a specific location or has its single value replicated to multiple points in space. Point-based data streams are applied to MF-OWHM2 using TabFiles, Time-Series Files (TSF), and LineFeed files. The fundamental difference in these types is that a LineFeed does not have a specific temporal component, but automatically loads the next input line for every stress period (or time step); conversely, TabFiles and TSFs contain a time stamp with the input data that determines when the data are applied.

The LineFeed input can load an arbitrary number of "FeedFiles" that structure the temporal input in a spreadsheet style. Each column of the FeedFile represents a data point, and each row represents one stress period (or time step) of data input. The columns can be separated by multiple blank spaces, multiple tabs, and single commas. The keyword "NaN" is used as a place holder for a missing point measurement (for example, an MNW2 well that has not been drilled or one that has been destroyed). The NaN value allows differentiation between stress periods in which a feature does not exist and those in which the applied rate is equal to zero. For example, a zero-value specified for a well in the MNW2 package indicates intraborehole flow is possible, whereas a NaN means that the well does not exist during that stress period. The LineFeed input style is currently supported only for the SFR, WEL, GHB, and MNW2 packages. For the input structure details and package support of LineFeed, please see the "LineFeed—Alternative Temporal Input" section in appendix 2.

TabFiles (time-tabulated input) have been linked to an MF-OWHM2 ExpressionParser to allow their tabulated values to be passed to a user-defined function. A TabFile is a time-series-like text file that contains a date and associated datum point on each line that uses a simulation's time-step date (or total simulation time) to set a model input property. By linking the TabFile result to the ExpressionParser, fewer text files are required to describe a set of model features. For example, a group of model cells that represent a GHB ocean boundary can be linked to a single TabFile composed of a time series from a sea-level gage, and use the ExpressionParser to translate the sea level to a freshwater equivalent for each ocean boundary GHB cell. As with TabFiles, a Time-Series File (TSF) has been introduced that is tied to calendar dates and offers more options to process the data (for example, interpolate, nearest value, time-weighted mean, step function). TabFiles are optimized for one file that is applied to many features (such as one TabFile linked to hundreds of GHB cells), whereas the Time-Series File is optimized for setting a property of a single feature (such as specifying a single SFR stream inflow or diversion). These new input structures facilitate integration into a self-updating structure of data streams, simulation, and analysis useful for evaluating changing water-resource management problems.

TabFiles and TSFs are similar in that both are single files that contain a time stamp and associated data input for each timestep. At the start of a simulation time step (not stress period) MF-OWHM2 uses the time step's starting and ending dates to parse the appropriate input data from the TabFiles and TSF. TabFiles and TSF both support time stamps in the form of a decimal year or calendar dates (in International Standard Organization, ISO, or American style), and TabFiles also support the model simulated time (TOTIM).

TabFiles (described in appendix 2) are optimized for reuse across multiple model features—such as one TabFile applied to multiple wells—and can optionally pass the time-tabulated data point to a custom expression. Each model feature that the TabFile is linked to can have its own custom expression. For example, a single sea-level gage record can be used as a TabFile input that is applied to all model cells that represent the ocean boundary condition simulated with the GHB. The sea-level gage is then made unique to each model cell by including an expression that translates the sea level to its freshwater equivalent and incorporates the bathymetry of the location. Tabfiles' data are parsed using the time interval from a time step's starting and ending times.

The TSFs (described in appendix 2) are optimized for use by a single model feature, such as a TSF that specifies the inflow to a single SFR segment. As with TabFiles, the TSF uses the time step's starting and ending calendar dates to determine which input data are applied (note that it does not use the simulated time, TOTIM). Because a TSF uses a time step's starting and ending calendar date, its use requires that a starting calendar date is specified as a BAS package option (see appendix 3, **STARTDATE** keyword). In MF-OWHM2, the data in a TSF are processed on the basis of a time step's starting and ending calendar date and using either interpolation, averaging, or resampling (appendix 2). TSF files either contain a complete time record in which the current time step's calendar date window matches the appropriate data, or the TSF may specify a single annual time series that is parsed based on the day and month of the time step. The single annual time series is advantageous if there is a repeating annual input. As a TSF, this repeating annual input would only require specifying the month and day of the month, and the input automatically finds the appropriate part of the file on the basis of the time step's month and day.

Separation of Temporally Varying Spatial Input (Array or Raster Data Stream)

Separation of spatial data through time is most useful for compiling and managing input for features such as FMP that require large data streams of climate and land-use data. This is facilitated with the Transient File Reader (`TFR`, appendix 1 and 2), which is a pointer file that directs when and from where input is loaded. In a `TFR` input file, each uncommented row of text has a keyword that directs where to load input or to reuse previous input that is applied for the current stress period. The `TFR` allows the temporal input to be split among multiple files. This allows the specification of multiple arrays of climate or land-use attributes coincident with the stress-period intervals. For example, a set of monthly precipitation arrays can each be in a separate file that is loaded with the `TFR` for the appropriate stress period. This type of separation makes it easier to build and manage climate data for historical periods or for future climate change evaluations derived from downscaled Global Climate Model data. The `TFR` also supports multiple scale factors that provide flexibility to rescale the input data from simulation scenarios and model calibration. These scale factors also provide a method of altering raw input data without having to rebuild a package input. For details about the input structure of a `TFR`, see "Transient File Reader and Direct Data Files" in appendix 1. For input and calibration examples, see "Transient File Reader—Spatial-Temporal Input" in appendix 2.

Fundamental MODFLOW Improvements

MF-OWHM2 is based on MODFLOW-2005 and consequently can run any models developed for it. With each release of MF-OWHM2, the original MODFLOW base code is updated and improved. This section briefly introduces some of the updates, improvements, and new features of the MODFLOW part of MF-OWHM2. Although not specific to the MODFLOW-2005 packages, this section discusses how error messages have been altered to be more user friendly and how to interpret their meaning. The MODFLOW-2005 part includes notable code changes and improvements, which maintain backward compatibility, to the BAS, DIS, WEL, WEL1, MNW2, HydMod, and HOB packages; modifications and improvements to the rest of the MODFLOW base packages; additional spatial coordinate and temporal information; improved and advanced file operations, input and output (I/O) options, and budget features; and new clear and understandable warning and error message handling. Appendix 3 provides a detailed explanation of the new features for the MODFLOW-2005 part of MF-OWHM2.

Error Messages and How to Interpret Them

The MODFLOW base-read utilities (*U2DREL*, *U2DINT*, *U1DREL*, *URWORD*, and *ULSTRD*) were modified to provide user-friendly error information. Traditional MODFLOW errors would raise Fortran-style debug information that could be of limited value in determining the cause of the error. In MF-OWHM2, the error messages provide the user with clues to the reason the simulation stopped. Typical output includes the name of the input file that contains the error-producing input, the line of text that was processed, and the operation that was not executed (for example, failure to load a number or open a file). Figure 6 shows an example of an error message for the GBH package where the expected input is three integers (model layer, row, column) and two floating point numbers (BHead, the boundary head, and Cond, the boundary conductance), but the last floating-point number—the boundary conductance—is missing.

This error is written to the command prompt, then the Listing file (LIST) file, and the WARN file. One important clue in the example ("`FAILED TO CONVERT TEXT TO DOUBLE PRECISION NUMBER`"; fig. 6) error message is that MF-OWHM2 was attempting to convert **"#"** to a number. If there was no number present and the end of the line was reached, then the message would convey failure attempting to convert **" "**, indicating that the number is missing. Figure 7 shows an example error loading input that requires the user to specify a file to open and load data, but the path to the specified file is incorrect.

In rare situations, the user-friendly error messages are not able to identify the error, and Fortran debug information is written to the screen instead (fig. 8). MF-OWHM2 has been compiled so that the Fortran error debug messages provide the error type and the call stack to indicate where the problem is. The call stack is a list, which begins at the error, of Fortran routine names, source-code line numbers, and source-code file names. The call stack uses the word "Unknown" for routines that it cannot identify. In the call stack, the first source-code file name that is not written as "Unknown" is typically where the error is. Often the MF-OWHM2 package that raised the error is in the source-code file.

In figure 8, the first line indicates the error occurred while utilizing the file "D:\SFR_Input.sfr". Ignoring rows with "Unknown", the first readable Routine and Source are GWF2SFR7AR and gwf2sfr7_OWHM.f, respectively, which indicate that the error originated from SFR because the letters "sfr" are in the Source name. If the error cannot be determined from the SFR input, the number specified under "Line" represents the line number in the file gwf2sfr7_OWHM.f where the error occurred. In the Fortran file around line 1194 (fig. 9), it can be discerned that the error is in the command

```
READ (In, *) ITMP, IRDFLG, IPTFLG
```

that has three input variables defined in the SFR manual (Niswonger and others, 2006). This indicates that the error must be caused by input file "D:\Model\Input.sfr" while loading one or more of the input variables ITMP, IRDFLG, or IPTFLG. If this error cannot be determined from the previously described input variables, then the nearby code comments can help determine the source of the error. In Fortran, a comment begins with either an "!" or contains a letter in the first column (typically a C in the first column). For example, on line 1192, there is the following comment: "C14----READ SEGMENT INFORMATION FOR FIRST STRESS PERIOD." that provides helpful information for identifying the error. The comment indicates that the subsequent code is designed to load the first stress-period segment input for SFR. This indicates that the error must have occurred at some point while loading the first stress period's segment information.

```
                          ONE-WATER ERROR

         THE FOLLOWING COMMENTS WERE PASSED TO THE ERROR ROUTINE

THIS ERROR IS BELIEVED TO HAVE ORIGINATED FROM WITHIN THE FOLLOWING FILE:
"D:\ExampleModel\GHB\GHB_Input.ghb"

THE GUESSED LINE THAT THE ERROR OCCURED ON IS:

"1  3  5    10.     # Layer Row Column Bhead  Cond"

THE DESRIPTION OF THE ERROR IS:

GET_NUMBER READ UTILITY ERROR: FAILED TO CONVERT TEXT TO DOUBLE PRECISION NUMBER.

THE FOLLOWING IS THE TEXT ATTEMPTED TO BE CONVERTED "#".

THE FOLLOWING IS AN ADDITIONAL COMMENT PASSED TO GET_NUMBER:

ULSTRDSTRUCT ERROR: FAILED TO LOAD FLOATING POINT NUMBER AFTER LAYER, ROW, COLUMN.
INPUT EXPECTS 2 FLOATING POINT NUMBERS AFTER LAYER, ROW, COLUMN,
BUT FAILED TO LOAD THE 2 FLOATING POINT NUMBER ON THE CURRENT INPUT LINE.
```

Figure 6. Example of a MF-OWHM2 error message for missing input in the general head boundary (GHB) package. In this example the input failed to supply the second floating point input (fifth input number of the line) and instead found a #, which is why the error message was raised.

```
FILE I/O ERROR:

FOR FILE UNIT 20

WHICH IS ASSOCIATED WITH FILE: D:\ExampleModel\FMP\FMP_Main.fmp

WHILE READING OR WRITING LINE
"SURFACE_ELEVATION  OPEN/CLOSE ./ExampleModle/Land_Surface_Elevation.txt"

THE FOLLOWING IS AN ADDITIONAL COMMENT INCLUDED WITH ERROR:

FAILED TO OPEN FILE WITH GENERIC_INPUT_FILE_INTERFACE.

FOUND KEYWORD "OPEN/CLOSE",
BUT FAILED TO OPEN THE FOLLOWING FILE:

"./ExampleModle/Land_Surface_Elevation.txt"

PLEASE CHECK TO SEE IF THE PATH AND FILE NAME CORRECT.

 ***NOTE THAT THE "/" WORKS FOR BOTH WINDOWS AND LINUX,
    BUT THE "\" ONLY WORKS ON WINDOWS.
```

Figure 7. Example of error message for incorrect file path for input data specified by the user. This error is raised because the path to the file "./ExampleModle/Land_Surface_Elevation.txt" is invalid. This error results when the file does not exist or there is a mistake or spelling error in the directory path. In this example, the directory "ExampleModle" should have been spelled "ExampleModel."

```
forrtl: severe (59): list-directed I/O syntax error, unit 4, file D:\SFR_Input.sfr

Image              PC                Routine            Line        Source
OneWater.exe       00007FF60FEABF87   Unknown           Unknown     Unknown
OneWater.exe       00007FF60FF06624   Unknown           Unknown     Unknown
OneWater.exe       00007FF60FF045A9   Unknown           Unknown     Unknown
OneWater.exe       00007FF60EAB3555   GWF2SFR7AR          1194      gwf2sfr7_OWHM.f
OneWater.exe       00007FF60F70C5FC   MAIN__               181      OWHM_Main.f90
KERNEL32.DLL       00007FFB80E71FE4   Unknown           Unknown     Unknown
ntdll.dll          00007FFB8132CB31   Unknown           Unknown     Unknown
```

Figure 8. Example of a crash of MF-OWHM2 that returns Fortran call stack information. In the call stack, the first source code line number (Line) and file name (Source) that is not written as "Unknown" is typically where the error is. In this example, the call stack indicates that the error is on line 1194 in the source file gwf2sfr7_OWHM.f.

```
1188
1189       CALL ERR%CHECK(HED='SFR FATAL ERRORS',
1190      +                    INFILE=IN, OUTPUT=IOUT, KILL=TRUE)
1191  C
1192  C14-----READ SEGMENT INFORMATION FOR FIRST STRESS PERIOD.
1193       IF ( NSFRPAR.EQ.0 ) THEN
1194         READ (In, *) ITMP, IRDFLG, IPTFLG
1195         NP = 0
1196         nlst = NSS
1197         lb = 1
1198         ichk = 1
1199         CALL SGWF2SFR7RDSEG(nlst, lb, In, Iunitgwt, Iunituzf, NSEGCK,
1200      +                      NSS, ichk, 1, Nsol)
1201       END IF
```

Figure 9. An example excerpt of Fortran code from `gwf2sfr7_OWHM.f` that includes the line numbers in gray. If there was a Fortran error that stated the error was on line number 1194, then it would indicate the error occurred while trying to load `ITMP`, `IRDFLG`, or `IPTFLG`.

Calendar Dates

MF-OWHM2 supports, for select packages, calendar dates both for input and for output. The input structure for calendar dates is very flexible, allowing for ISO 8601 format, American (United States) style structure, or a decimal year. If the year is specified, then it must be a four-digit Gregorian year, unless it represents a year before the 11th century.

The ISO 8601 format has the following input structures:

- `yyyy-mm`
- `yyyy-mm-dd`
- `yyyy-mm-ddThh:MM:ss`

The `yyyy` is the four-digit Gregorian year, `mm` is a two-digit month number, and `dd` is a two-digit day of the month. If the day (dd) is not specified, then it is automatically set to the first of the month (`dd=1`). The time separator **`T`** is used to initiate the start of the 24-hour clock time input, where `hh` is the hour in 24-hour format, `MM` is minutes, and `ss` is seconds. If the input only uses the calendar part (yyyy-mm-dd), then the assumed time is midnight (00:00:00). Note that the hyphen, -, is used as the delimiter between parts of the date for the ISO style. To use American date format, a forward slash is used instead, resulting in the following input formats:

- `mm/yyyy`
- `mm/dd/yyyy`
- `mm/dd/yyyyThh:MM:ss`

Several of the date-aware input structures allow for input of a month and day, but exclude the year. The input structure then automatically appends the year at a later time. Internally, the year is set to zero and then updated with the correct year when in use. Note that February 29th should be avoided in this input structure because it can become ambiguous by automatically becoming March 1st during non-leap years. The following are different methods of specifying a month and day, but not the year (note the use of a backslash is required if the month and day are specified as numbers):

- `mm\dd`
- `mmm`
- `mmm-dd`
- `mmm/dd`

The mmm represents either a three-letter representation of the month (for example, JAN) or the full month name (for example, January). If the day of the month is not specified, then it is automatically set to the first of the month. Note that the month-day input structure also supports a 24-hour clock if the **`T`** separator is present, but it is not recommended. If a starting calendar date is specified, then MF-OWHM2 keeps track of the date of each stress period and provides the calendar date to the volumetric budget in the LIST file and for select package-output options. The following are examples of acceptable calendar date inputs:

- `1979-4-23`
- `1979-4-23T16:20:01`
- `4/23/1979`
- `4/23/1979T16:20:01`

Calendar date output is usually in the ISO 8601 standard format of `yyyy-mm-ddThh:MM:ss`, which some programs may not recognize. In particular, the spreadsheet program Microsoft Excel does not auto-recognize it as a date unless the letter **`T`** is removed. To fix this, "search and replace" the letter **`T`** with a blank space; Excel then auto-converts it to a date format.

Simulation Starting Date and Variable Time Steps

Two improvements to the temporal features of MF-OWHM2 are the ability to specify a starting calendar date and define custom time-step lengths. Defining a starting date is required when using inputs that only support calendar dates or decimal years, such as a TSF. Custom time-step lengths allow the user to predefine each times step's length so that they can be aligned with observations or ensure that time-step lengths are in whole numbers (such as a 11-day stress period with time step lengths as 5 and 6, instead of 5.5 and 5.5).

Starting Date

In MF-OWHM, dates were included by specifying a starting decimal year using the DIS package keyword **`STARTIME`**. The problem with this input is that it assumed a 365.2425-day year and did not support common and leap years (365- and 366-day years, respectively). To overcome this limitation, calendar dates were introduced in MF-OWHM2. Calendar dates are initiated by specifying the keyword **`STARTDATE`** (appendix 3, BAS package options) followed by the starting calendar date of the model.

When **`STARTDATE`** is included, MF-OWHM2 uses the starting calendar date to keep track of every time step's starting and ending calendar date and the corresponding decimal year. A decimal year in this case has its decimal part (the numbers to the right of the decimal point) representing the fraction of a 365-day or 366-day year, depending on if it is a common or leap year, respectively. For example, the calendar date April 23, 1979, equals decimal year 1979.306849, where 0.306849 represents (113 – 1) / 365, whereas April 23, 1980, equals decimal year 1980.308743, where 0.308743 represents (114 – 1) / 366.

If **STARTDATE** is specified, then MF-OWHM2 provides the calendar date to the volumetric budget in the LIST file and for select package-output options. In particular, when using the HOB package and specifying calendar dates, the HOB output file includes the decimal year and the calendar dates with each observation.

Specifying Time-Step Lengths

The DIS package was modified to allow the user to specify the exact time-step length. The time-step lengths are loaded on the same line as the stress-period information (PERLEN NSTP TSMULT SS/TR). This feature is initiated when the time step count (NSTP) is specified as a negative number and the multiplier is set to 1. The absolute value of the time step count represents the number of time-step lengths read to the right of the stress-period type (SS/TR), and the sum of the time-step lengths is the stress-period length (over-writes PERLEN). This allows the user to customize time-step lengths to match observation times or to create an acceleration factor that uses simpler-integer numbers (for example, 1, 2, 7, 10, 80 to accelerate to a total of 100 days). The compact numbering can be used to prevent simulation times with decimal parts by specifying time-step lengths to be whole numbers.

The capability for user-specified time-step lengths is particularly advantageous when the stress periods mimic calendar months and the month can be broken into different counts of days. For example, a month with 31 days can be represented by a stress period with four time steps with lengths defined as 7, 8, 8, and 8 days.

Improved Coordinate System

Previous MODFLOW simulation models did not provide a link between spatial coordinates and the model grid. This lack of explicit connection caused the user to rely on a separate GIS, graphical user interface (GUI), or comments in the input files to keep track of where spatially the model resided. This limitation was partially overcome with the release of MF-OWHM by allowing the user to specify a Cartesian coordinate system for the model grid.

In MF-OWHM2, the Cartesian coordinate system is connected to a new input format, called LineFeed (see appendix 2), that, for the GHB and WEL packages, accepts either the traditional layer, row, and column input or the Cartesian (X, Y, and Z) coordinates. If coordinates are specified, then MF-OWHM2 automatically determines the layer, row, and column in which the coordinate resides. The coordinate system is also connected to HydMod, such that the hydrograph location's point coordinates (XL and YL) use the specified coordinate system to determine the model row-and-column location of the hydrograph (note that if a coordinate system is not defined, then it defaults to the same coordinate system as HydMod).

Basic Packages Improvements

Modifications were made to most of the MODFLOW core packages to advance the supported input styles toward the self-updating concept, extend the features, and reduce simulation runtimes. One important improvement is that the BAS package has the new option, **INPUT_CHECK**. When this option is activated, MF-OWHM2 cycles through the simulation's input files—that is, the simulation is run, but without the solver, to check all input. Any problems with the input files result in MF-OWHM2 either crashing or recognizing erroneous input and writing a message to the LIST file. This option is especially useful to quickly check the input files for long-running simulations that could have an input error toward the end.

The WEL package source code was entirely rewritten to restructure the location of the TabFiles in the input file. The new WEL package also expands the optional smoothing of the pumping rate as the pumping cell goes dry by providing this option to all the solver packages (previously only available when using the NWT solver). The original WEL package remains, allowing the user to have two separate WEL packages during a simulation (declared as WEL for the new version and WEL1 for original).

An improvement to the GHB package provides the option to automatically build its boundary conductance (BCOND) from the hydraulic conductivity used by the flow package (LPF or UPW packages). This avoids having to specify BCOND as part of the input and allows the GHB to implicitly use the aquifer properties when determining boundary flow. The GHB package also can vary a GHB cell's BCOND with saturated thickness, which can be applied to the user specified BCOND or the flow package calculated version. Lastly, the inputs to PVAL, MULT, and ZON packages offer automatic counting of the number of parameters, multiplier arrays, and zone arrays, respectively, by setting their count variable to –1, and all these packages may include text comments anywhere, preceded by a "#" character.

File Operation Improvement

The file operations of MODFLOW can be challenging for new users, especially those without a background in Fortran programming languages. One of the limitations of Fortran 95, used for MODFLOW-2005, is requiring an identification number, called a unit number, to be assigned to all files opened by the program. Consequently, MODFLOW-2005 requires, as part of its NAME file input, a unit number that is assigned to each package-input file opened by it. Part of the MODFLOW base code was modified to make use of Fortran 2003 and 2008 features, which include automatic unit-number assignment. This modification allows the unit numbers in the MF-OWHM2 Name file (NAME file) to be optional for packages. If the NAME file only has the package name followed by a file

name, it auto-assigns a unit number that will not conflict with another. It is still required to use a unit number in the NAME file for the keywords DATA and DATA(BINARY) because the unit number is used for identification by other packages, such as the CBC number or input that uses the keyword **EXTERNAL** to load input (see appendixes 1 and 3 for more information).

A transcript of all operations is written to the Listing file in a MODFLOW and MF-OWHM2 simulation. This file is called LIST in the NAME file and was previously required to run a MODFLOW simulation. For large simulation models the LIST file can become very large. The large file size may affect hard-drive performance, slowing down the overall simulation runtime. This is particularly important during calibration, when multiple copies of the listing file can occupy a large amount of hard drive space. In MF-OWHM2, the LIST file is now optional. If LIST is not specified in the NAME file, then it is not used in the simulation. LIST suppression was included in MF-OWHM with the LSTLVL feature, but this feature has been removed from MF-OWHM2 now that the LIST file is optional.

The NAME file itself was modified to include a set of new optional keywords that alter how the file is opened and processed. To maintain backward compatibility, the keywords are specified to the right of the file name in the NAME file. Another useful option is the ability to buffer the files opened for input and output in random access memory (RAM; appendix 1). The buffered file is either preloaded in RAM if it is an input file or, for an output file, written to RAM until the buffer is full and then written to the hard drive. This is initiated by the keyword **BUFFER**, followed by the buffer size in kilobytes (KB). By default, all files in the MF-OWHM2 NAME file are opened with a buffer of 0 KB for the listing file, for immediate writing; 128 KB for all packages; and 32 KB for all files opened with DATA and DATA(BINARY). There are two limitations to buffering for output files. The first is that the file is not written until the buffer is full, causing results to be written in chunks equal to the buffer size, which delays the actual writing. This can be a problem if the user wants to view results during runtime. The second is that if there is a power interruption to the computer, then the information stored in the buffer is lost and never written to the file. For this reason, the LIST file has a default buffer of 0 KB, but if this is not an issue, it is recommended to have the LIST buffer set to 1024 KB to buffer the file in one-megabyte chunks.

For the LIST file and DATA and DATA(BINARY) files that are used for output, there is an option to split the file into a set of smaller files. This is done with the optional keyword **SPLIT**, followed by a split size in megabytes (MB). This is advantageous when output files become too large to be opened in a text editor. If a file is specified by including the keyword **SPLIT** and its file size exceeds the split size limit, a new file is created with the same name, but has a number appended to the name to make it unique. This new file has the same header on the first line as the original file. The new file is used until the new file size exceeds the split limit; then, another file is created.

New Budget Features

MF-OWHM2 includes a set of new budget options for certain packages that allow the user to analyze and understand model results better or to make the connection to calibration software simpler. These modifications include the ability to define multiple budget groups and to write detailed budget information to a separate file. Traditional MODFLOW-2005 Volumetric Budgets and CBC file outputs only write the total flowrate in and out of the groundwater flow equations as simulated by each package. Lumping all the rates for an entire package does not allow the user to see how different parts of the package may interact with groundwater flow.

One common example of this problem is FMP linked MNW2 wells. In this case, FMP determines the desired pumping rate MNW2 wells should have, and MNW2 determines the actual pumping rate and includes this rate in its budget terms. To single out the FMP linked wells would require using the MNWI package and reconstructing its output information.

The new feature allows two budget groups to be defined for output in the MODFLOW Volumetric Budget from the CBC. Having two distinct groups in the CBC allows for programs such as the ZoneBudget post-processor (Harbaugh, 1990) to tabulate water budgets that represent each group. This feature is available for the MNW2, RIP-ET, WEL, GHB, DRN, DRT, and RIV packages.

MODFLOW-2005 simulations only write budget information to the LIST file (or WBGT file) if requested by the Output Control (OC) package. This led to two potential issues. The first is that the mass balance errors and cumulative mass balance errors were only calculated when the budget information is requested in the OC. This could result in underestimates of cumulative mass errors, because only the time steps specifically requested by the OC for a budget calculation are summed. The second issue is that time steps that reached convergence may have a large mass error that is unknown to the user. This occurs when the convergence criteria are not strict enough (specifically, the solver's HCLOSE and RCLOSE are too large). In MF-OWHM2, the BAS package was modified to always calculate the Volumetric Budget for every time step and raise a warning if the mass rate balance error ever exceeds 5 percent. The BAS package also includes an external output file that contains detailed budget information for every time step. The budget information is specified as a time-step number, calendar date, simulated time, time-step length, and then the rates of groundwater inflow and outflow for all packages in use during the specified time-step. The

format of this file is tabular (columns of rate values for each package and rows of time-step records), in which the first row contains a header and subsequent rows contain the data. This format can easily be loaded to a spreadsheet or database software for post-processing.

For the packages SFR, GHB and FMP, there is a new option similar to the column-based volumetric budget output, which prints properties specific to each package for each time step and includes a calendar date, the rate information, and head-dependent properties. For SFR, the output is identical to the file created by using the flag "`ISTCB2>0`" (the input keyword used in MODFLOW-2005), with the addition of two columns—the date and the streambed elevation at the start of the reach. The GHB package output provides the conductance used in the simulation, which in MF-OWHM2 can be a function of the water-table height or of the flow-package hydraulic conductivity. FMP has multiple output options that provide detailed information about Crops and Farm Wells. The Crop output provides detailed information for actual transpiration, actual evaporation, anoxia losses, fallowed-land evaporation, and, optionally, the crop root pressures. The Farm Wells output gives detailed information for demanded pumpage; final simulated pumpage; and the reason for reduced pumping capacity of wells, such as scale factors or seepage faces in MNW2 wells.

Warning Package (WARN)

MODFLOW-2005 writes all errors and warnings to the LIST file. Because of the length of the LIST file, it can be difficult to find important warnings that various packages might raise. The Warning Package is an optional output package that presents all package warnings in one location. To initiate the Warning Package, it must be declared in the NAME file by the keyword WARN, followed by a unit number, and the filename to which to write the warnings. If the warning package is used, then warnings are written to the LIST and WARN files.

Landscape Features—Farm Process (FMP)

In MF-OWHM2, the Farm Process, version 4 (FMP), has important upgrades that include modification to some of the structural relationships of selected features. The concepts and features that are the foundation for simulating the use and movement of water through the landscape by FMP (Schmid, 2004; Schmid and others, 2006; Schmid and Hanson, 2009a; and Hanson and others, 2014d) are summarized in appendixes 4 and 5 ("Consumptive Use and Evapotranspiration in the Farm Process" and "Landscape and Root-Zone Processes"). Upgrades to FMP improve simulation runtimes, simplify the input structure, remove features that are seldom used, add features that represent newly modeled relationships between selected features, and make the addition of other features easier to incorporate. The following sections briefly summarize the newly added concepts and features. Features that were removed from FMP are also listed. Appendix 6 describes the input structure for FMP in detail.

Concepts New to the Farm Process

A variety of concepts have been added or modified in FMP. These include options for additional demand related to leaching requirements for salinity (salinity demand) and urban consumption; capability to specify multiple land-use (crop) types in a model cell; revised crop consumptive-use concepts; options for defining how each crop type's roots interact with groundwater; a redefined irrigation efficiency and deficit irrigation framework; revised methods to define how pumpage in a WBS is distributed; and new options for simulating managed aquifer recharge for water-banking operations.

Crops can have an additional irrigation demand (leaching requirement) for flushing salts out of the soil zone based on the salinity of the irrigation water. The MF-OWHM2 implementation of a leaching requirement, called a salinity demand, is described in the "Salinity Irrigation Demand" section.

The deficit irrigation "deficiency scenario" now has two options for how irrigation efficiency is handled when there is insufficient water supply. Previously, irrigation efficiency increased up to perfect efficiency to ensure supply met the demand. This assumes that during water shortages agricultural entities became more efficient with their irrigation. This previous option is still available, but now, by default, irrigation efficiencies are held constant during deficit irrigation. This assumes that irrigation equipment does not change or improve during a water shortage. The implementation of deficit irrigation is described in the "Implementation of Deficit Irrigation" section. The "Non-Irrigation Flag" has been redefined as an irrigation type with an associated efficiency, rather than being specified by land use (Crop). It should be noted that in FMP a crop is not irrigated when the "Irrigation Flag" is set to zero, and if set to a positive integer, the flag refers to the irrigation type. The irrigation types—such as flood, sprinkler, soaker hose, or drip irrigation—can be

associated with specific crops or crop groups to indicate that the crop is irrigated and has the irrigation type's efficiency.

Deliveries of water for managed aquifer recharge were newly implemented into FMP. This is described in the "Direct Recharge Options" section.

As part of the FMP upgrades, a revised input format and related set of input read utilities were developed. The FMP input now utilizes a template file structured as block-style input using keywords that indicate the property to load or feature to enable. The input blocks simplify the user's choice of FMP options. Active features are grouped into the following named blocks in the template (appendix 6):

1. **GLOBAL DIMENSION**: Global properties used by other FMP blocks
2. **WATER_BALANCE_SUBREGION**: Properties that pertain to defining WBS
3. **OUTPUT**: Ancillary output files
4. **OPTIONS**: Global modifier options
5. **SOIL**: Soil-specific properties
6. **CLIMATE**: Climate-related properties
7. **SURFACE_WATER**: Surface-water deliveries and runoff properties
8. **SUPPLY_WELL**: Groundwater-supply well properties
9. **ALLOTMENT**: Apply a limit to different water supplies
10. **LAND_USE**: Crop or land-use specific properties
11. **SALINITY_FLUSH_IRRIGATION**: Addition irrigation demand for salinity leaching

There is no requirement for the order of the blocks in the FMP input file, but the numbered list order provided here is recommended for consistency with different model applications. The only blocks required to run the FMP simulation are the **GLOBAL DIMENSION** and **WATER_BALANCE_SUBREGION** blocks. The remaining blocks can be retained as needed to specify the input data and the desired FMP simulation.

Water-Balance Subregion (Farm) Water Sources

To increase simulation speed, each water-balance subregion (WBS) can have its water sources specified. These water sources represent the available sources of water used for irrigation of crops. The available sources are non-routed deliveries (NRD) that represent imported water, semi-routed deliveries (SRD) that represent surface water delivered from an SFR diversion, and groundwater pumping. The input is specified in the **WATER_BALANCE_SUBREGION** block by the WATERSOURCE keyword (see appendix 6). Use of the water-balance block is advantageous if a WBS does not have any imported or surface-water sources. The block essentially declares this as a groundwater-only WBS, which prevents FMP from using any of the surface-water routines when determining the available water supplies for the WBS.

Supply-Constraint Options (Allotments)

Supply constraints are applied by WBSs in FMP and can be specified as a surface-water or groundwater allotment (if not defined, then the allotments are set to infinity for all WBS). Allotments are useful for representing water rights, operating agreements, legislation, adjudication, or analyzing sustainability. A surface-water allotment imposes a limit for a WBS on the amount of surface water that can be delivered. This limit only restricts water delivered as a semi-routed delivery (SRD) from SFR. Surface-water allotments must be specified as a maximum volume of water that can be delivered in a stress period or a maximum height per stress period (the height is converted to a volume by multiplying it by the associated WBS's irrigated area). The volume per stress period becomes a rate limit by dividing it by the stress period duration. This volumetric flow rate (L^3/T) becomes the maximum delivery rate that is allowed. A groundwater allotment is a volume per stress period limit imposed on a WBS's collective pumping. As with the surface-water allotment, the groundwater allotment is divided by the stress period to obtain a maximum allowed total WBS groundwater pumping flow rate (L^3/T).

Salinity Flush Irrigation Demand

Managing soil salinity is essential to avoid salt accumulation in the soil zone and loss of arable lands for agriculture. Dissolved salts in irrigated water remain in the soil after the water is removed through evapotranspiration. For example, applying 1 acre-foot of water with a total dissolved salt concentration of 735 parts per million could increase the soil-salinity mass by one ton of salt (Cahn and Bali, 2015). The increase in soil salinity reduces potential transpiration of the crops and lowers potential yields. The crop yield is the quantity of crop, by mass, that is harvested per unit area of land cultivated. A maximum yield can be determined for a unit of cropped area by assuming ideal water supply, climate, and soil salinity. Morway and Gates (2012) estimated the reduction in yield due to salinity for agricultural lands of the Lower Arkansas River Valley, Colorado, ranged from 6 to 17 percent over a 9-year period. In addition to the loss of productivity and irrigable lands, the supplemental water required for salt flushing as part of irrigation with saline waters can also greatly

increase water demand. Salinity-flush demand was added to FMP to account for the additional applied water necessary to flush salts out of the soil zone.

The Food and Agriculture Organization of the United Nations (FAO) includes detailed information about soil salinity, crop salt tolerances, and guidelines for increasing crop yields (Tanji and Kielen, 2002). Soil salinity can be measured using the average electrical conductivity (EC) of a soil sample in units of decisiemens per meter (dS/m). Within the range of 0.1 to 5 dS/m, an EC of 1 dS/m represents 640 milligrams per liter (mg/L) of total dissolved solids (TDS). For measurements greater than 5 dS/m, 1 dS/m represents approximately 800 mg/L of TDS. Table 2 presents a set of thresholds for various crops that represent the average soil salinity tolerated by the crop, as measured in a saturated soil-paste extract (EC_e), without a loss in potential yield (Tanji and Kielen, 2002). These values are guides, and the actual value can vary depending upon climate, soil conditions, agricultural practices, and the stage during a life cycle of a crop. For example, if a crop is grown in a soil rich in gypsum, then the crop-salinity tolerance threshold (EC_e) can be increased by 2 dS/m (Tanji and Kielen, 2002).

To account for the reduction in yields due to soil salinity, the FAO designated crops as sensitive, moderately sensitive, moderately tolerant, and tolerant to salt build up. Figure 10 presents the EC_e ranges of these designations for different values of relative crop yield. For example, 80-percent relative yield indicates that for the EC_e ranges indicated, there is a 20-percent reduction from the potential yield. The crop-specific soil-salinity tolerances in table 2 represent a relative yield of 100 percent in figure 10.

The most common method for determining the necessary irrigation for salinity flushing is the Rhoades equation (Rhoades, 1972, 1977, 2012; Rhoades and Merril, 1976; Ayers and Wescott, 1985; Cahn and Bali, 2015), which is composed of two parts. The first part involves determining the fraction of total irrigation (applied) water that must pass through the soil to prevent the soil salinity from reaching the tolerance of the crop. This is a unitless fraction called the leaching requirement (LR). The LR is determined from the salinity concentration of irrigation water (EC_w) and the crop tolerance to soil salinity (EC_e). The EC_w and EC_e are both measured as electrical conductivity (dS/m). From Ayers and Wescott (1985), the leaching requirement can be calculated as follows:

$$LR = \frac{EC_w}{(5 \times EC_e) - EC_w} \qquad \forall \ (5 \times EC_e) > EC_w \qquad (15)$$

where

LR is minimum leaching requirement needed to control salts, with $0 \leq LR < 1$ (-);

EC_w is salinity of the applied irrigation water (dS/m); and

EC_e is average soil salinity (dS/m) tolerated by the crop as measured in a saturated soil-paste extract, and it can be viewed as the desired soil salinity after additional irrigation is applied.

The leaching requirement must be less than 1, so salinity flushing is possible only when EC_w is less than 5 times the EC_e. The choice of EC_e is based on the desired, or obtainable, relative yields of the crop. For example, table 2 represents EC_e values that calculate a leaching requirement to obtain 100 percent relative yield. Once the leaching requirement is determined, the Rhoades equation uses the crop irrigation requirement (the irrigation necessary to satisfy evapotranspiration for a crop) and specifies the total irrigation necessary for salinity flushing as follows:

$$AW = \frac{CIR}{1 - LR} \qquad (16)$$

$$D_{irrigation} = AW / OFE \qquad (17)$$

where

CIR is crop-irrigation requirement under perfect irrigation efficiency (L^3/T);

AW is applied water necessary for salinity flushing under perfect irrigation efficiency (L^3/T);

OFE is the irrigation efficiency, with $0 < OFE \leq 1$ (-); and

$D_{irrigation}$ is the irrigation necessary to satisfy evapotranspiration for a crop and also sufficiently provide salinity flushing (L^3/T).

MF-OWHM2, using FMP, can calculate the crop-irrigation requirement and determine the additional irrigation necessary, given a set of crop-salinity tolerances (EC_e) and the salinity of each of the sources of irrigation (EC_w). With this information, the FMP determines the leaching requirement from a composite EC_w (calculated from the mixture of available irrigation sources) and the corresponding additional irrigation to prevent salt build up. The determination can either be made through the Rhoades equation or through user-supplied expressions that calculate the additional irrigation. Additional details and examples for using salinity-demand input options are summarized in appendix 6 in the **`SALINITY_FLUSH_IRRIGATION`** block input option.

Table 2. List of common agricultural crops and their soil salinity (EC_e) threshold (Tanji and Kielen, 2002).

[The threshold value represents the point when the potential crop yield is decreased because of soil salinity. **Abbreviations**: dS/m, decisiemens per meter, a unit measurement of electrical conductivity; EC_e, mean electrical conductivity of a saturated soil paste taken from the crop's root zone]

Crop common name	Soil salinity threshold, EC_e (dS/m)	Crop common name	Soil salinity threshold, EC_e (dS/m)
Alfalfa	2.0	Garlic	3.9
Almond	1.5	Lemon	1.5
Barley	8.0	Lettuce	1.3
Broccoli	2.8	Peach	1.7
Cabbage	1.8	Potato	1.7
Carrot	1.0	Spinach	2.0
Celery	1.8	Strawberry	1.0
Corn	1.7	Tomato	2.5
		Wheat, durum	5.9

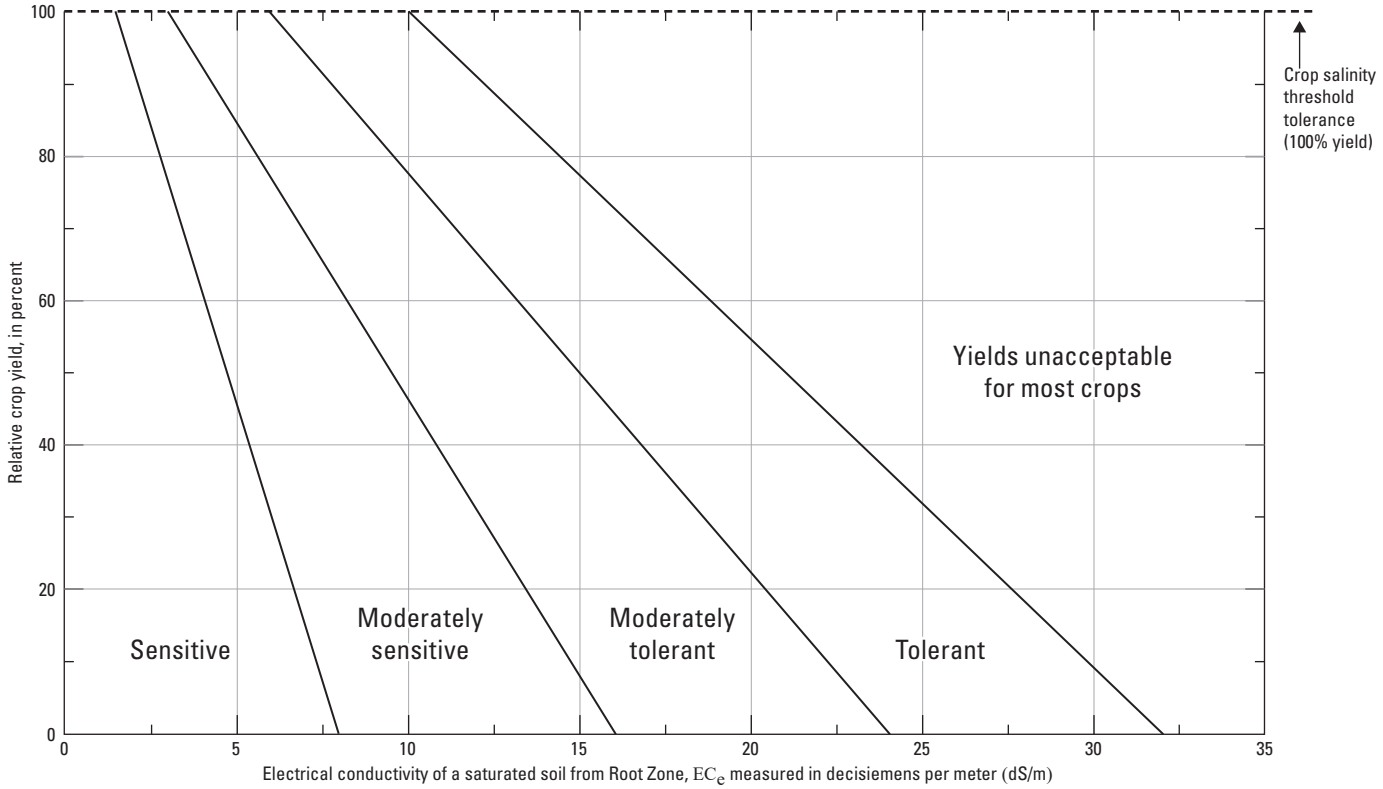


Figure 10. The United Nations Food and Agriculture Organization classification of crop tolerance to salinity (modified from Tanji and Kielen, 2002). The crop specific soil-salinity-threshold value, EC_e, represents 100-percent yield, and larger salinities result in decreases in the percentage yield.

Land-Use Grouping and Spatial Definition

Land use is an important aspect of an integrated hydrologic model. Along with climate, land use influences changes in water demand, use, and movement as well as sources of water supply and reuse. Each land use or "Crop" is specified using a numeric identifier (land-use ID). A land-use ID serves as a pointer to a set of common land-use properties (such as crop coefficient or potential consumptive use), and the ID is used to designate the land use's location in the surface model grid. The maximum number of land-use ID's must be declared at the start of a simulation (keyword **NCROP**), but the use of any land-use ID during a simulation is optional. That is, not all land-use IDs are required to be used during a single "stress period."

The land-use's available water for consumption and runoff calculations are determined by the WBS that a land-use is associated with. The association of the land-use ID with a WBS is by collocation, that is, where the land-use ID is spatially coincident with the WBS. For example, land use 1 (wheat) may be specified in WBS 1 and WBS 2, but only the wheat in WBS 1 contributes to WBS 1's total demand and the wheat in WBS 2 contributes to WBS 2's total demand. Additionally, if the wheat is irrigated, then WBS 1 only provides irrigation water to wheat in its domain; similarly, WBS 2 is for wheat in its own domain. The runoff calculations for wheat in WBS 1 are determined by WBS 1 and similarly for the wheat areas in WBS 2.

Allowing the user to define a set of land-use ID's that have a set of associated properties and spatial location provides better flexibility in model development. Although it could be preferable to some modelers if FMP used a pre-defined set of crop identification and properties, such as having a "wheat" category with preloaded properties or links to existing databases, it provides less flexibly and limits the applicability of MF-OWHM2. For example, agricultural catalogues such as CropScape (Mueller and others, 2011) do not include natural or urban categories of land use and related vegetation and only provide selected recent years of annual agricultural land use. As a result, each modeled region typically requires its own catalogue of land use that includes the vegetation and crop types specifically growing in that region. For example, a model can declare wheat, fruit trees, natural vegetation, urban irrigation, and strawberries as land-use IDs 1, 2, 3, 4, and 5, respectively. Although the same land-use ID can be used in any WBS in a model, the user may have reasons to segregate the same crop to different IDs that represent different plant varieties, agricultural practices, landscapes, or hydrologic conditions. For example, strawberries can be broken into two regional groups to capture different climate and growing condition effects, such that the new land-use IDs are 5 and 6 to represent coastal strawberries and inland strawberries, respectively.

Multiple Land Uses (Crops) in a Model Cell (New Feature)

The input for the spatial distribution of a land use ID in FMP is very flexible, ranging from a single model cell to every model cell in the entire model domain. The spatial location of land use in FMP can be specified using one or two possible methods. The first method, FMP keyword **SINGLE_LAND_USE_PER_CELL**, limits the spatial resolution to one land-use location of each land use to one per surface model cell. This is how previous versions of FMP declared the location of each land use (Crop). This method is still supported and is the recommended method for specifying the spatial location of the land uses because of its simple input structure. A new feature in FMP is an adjustment fraction, keyword **LAND_USE_AREA_FRACTION**, that reduces the surface area of a land use in a model cell. For example, an almond tree farm that covers half the surface area of a model cell would have a land-use fraction of 0.50—previous versions of FMP required that the almond tree land use cover the entire model cell. It should be noted that the land-use fraction is based on the farmed area and not the almond tree covered area—that is, the fraction includes tree canopy-covered area and the open space between the trees.

The second method for specifying the spatial location of land uses, FMP keyword **MULTIPLE_LAND_USE_PER_CELL**, allows for more than one land use to be defined per surface model cell. This option is useful when the land uses in a surface model cell are too different to be combined in a composite land use. The method that FMP uses to define multiple land uses per model cell is similar to the mixed riparian vegetation method in the RIP-ET package (Maddock and others, 2012; Hanson and others, 2014d). The input structure for multiple land uses per surface cell requires defining an area fraction array for each land-use type. The fraction represents the part of the model cell's surface area covered by the land use. For example, if a surface model cell is 30-percent wheat farm and 60-percent almond tree farm, then the fraction for wheat and almond trees for that that model cell would be 0.30 and 0.60, respectively. If the fractions of all crops defined for a model cell do not sum to 1, then the remaining area is assumed to be fallowed land (bare soil). In the previous example, 10 percent is assumed to be bare soil.

The choice of the appropriate input, one land use per model cell or multiple lands uses per cell, should be based on the spatial and temporal resolution of available land-use data, the objectives of model application, and the potential benefit of using multiple land uses compared to composite land use. Selecting land uses as **MULTIPLE_LAND_USE_PER_CELL** increases the complexity of input and total simulation run time, but this option may be essential for regions with mixed cropping and vegetation or complex topologies of land use.

To have multiple land uses (Crops) in a model cell, the user must specify a two-dimensional array of fractions, in a domain from zero to one, for each Crop instead of a Crop ID array (appendix 6). It is not required to have Crop fractions sum to 1 per model cell; if the Crop fractions do not sum to 1, then the unspecified part of the model cell follows the bare soil calculations (appendix 4). Therefore, if fractions are used, it is required to specify either a bare-soil evaporation rate or reference evapotranspiration rate. Additional details for using Crop-input options are summarized in appendix 6 in the **LAND_USE** block input.

Land-Use Grouping

Potential considerations for developing land-use categories include appropriately balancing loss of detail for the model's scale or generality, allowing flexibility for future land-use categories, and reducing the amount of model input and number of land-use categories. A useful approach to developing land-use IDs is to determine (1) the important land uses in the model domain, (2) how land use varies spatially and temporally during the simulation period, and (3) the relationship of land uses to the hydrologic budgets needed from the model. Equally important can be whether certain crops or vegetation types need to be simulated as separate entities, or whether groups of crops that have similar planting dates, harvest dates, growth cycles, crop attributes, and methods of irrigation can be grouped together into "virtual crops." Consequently, if specific land uses are integral to the desired model analysis, it is best to aggregate them in groups for analysis. For example, land uses can be grouped in stable land uses that are relatively permanent, such as natural or urban vegetation; land uses that can change on multi-year time frames, such as orchards and vineyards; land uses that are annual or seasonal, such as wheat-corn-fallow rotation; land uses with high-frequency multi-cropping, such as spinach; or land uses that represent non-traditional growing techniques, such as indoor nurseries. Grouping of more detailed land uses allows a simpler input structure and reduces the data input needs. Furthermore, representing the water demands of a group of similar land uses in the model can result in more efficient model execution. If it is found that a more detailed representation of a specific land use within a group is needed, it can be removed and made into its own group.

Crop Consumptive-Use (CU) Concepts

The potential consumptive use (CU) of a land use (crop) is defined as the consumption of water necessary to meet the land use's potential evapotranspiration (PET). In the context of MF-OWHM2, it can be assumed that CU is the same as PET, which is satisfied from water that originates directly from groundwater, precipitation, and applied irrigation under perfect irrigation efficiency. The necessary irrigation to meet the CU under perfect efficiency is called the crop irrigation requirement (CIR). After the CIR is determined, the additional water demand caused by inefficient irrigation can be determined from the irrigation type's irrigation efficiency.

Previously, FMP specified two consumptive-use flags as part of its input, `ICUFL` and `ICCFL`. The first flag, `ICUFL`, defines how the consumptive use input is specified. In FMP, if a crop's CU is to be directly specified, `ICUFL` is set to `1` or `2`. If CU is to be calculated using a crop coefficient (K_c) and reference evapotranspiration (ET_{ref}), such that $CU = K_c \times ET_{ref}$, `ICUFL` is set to `-1`. Appendix 4 provides a detailed explanation of the consumptive use concepts used in FMP.

The second flag, `ICFFL`, defines two consumptive use concepts previously called Concept 1 and Concept 2. These "Concepts" define how a crop's anoxia- or wilting-related pressure heads are calculated, the potential quantity of groundwater that a crop can consume directly, and whether deep percolation is simulated by UZF for delayed recharge. In this report, to identify these two concepts, the concept formerly called Concept 1 (`ICCFL` set to 1 or 3) is referred to as the *analytical root response* for root uptake of groundwater and anoxia, and Concept 2 (`ICCFL` set to 2 or 4) is referred to as the *linear root response* for root uptake of groundwater and anoxia. For a detailed explanation of the concepts, please see "Consumptive Use and Evapotranspiration in the Farm Process" (appendix 4) and "Concepts of Landscape and Root Zone Processes" (appendix 5).

The FMP can determine which of the two concepts is applied for simulating consumptive use of a given crop on the basis of amount of input provided. The consumptive-use concepts are no longer global properties (`ICUFL` and `ICCFL`) applied to all crops, but instead are specified on a crop-by-crop basis. The optional linkage to the UZF package—that is, FMP deep percolation becomes infiltration to UZF for delayed recharge —is still a global option that is specified with the keyword **UZF_LINK** in the **GLOBAL DIMENSION** block (appendix 6). If the **UZF_LINK** is enabled, it is important that the UZF package's ET option is disabled to prevent double accounting from FMP and UZF.

All consumptive-use concepts in the new input structure require, at a minimum, specifying the soil capillary fringe in the **SOIL** block. The *linear root response* concept (formally Concept 2 with `ICCFL = 2` or `4`) requires specifying the root depth (ROOT), fraction of transpiration (FTR), fraction of evaporation from irrigation (FEI), and fractions of delivery losses to surface water from precipitation and irrigation in the **LAND_USE** block for each crop. The *analytical root response* concept additionally requires a soil-type coefficient (silt, silty clay, sandy loam, or sand) in the **SOIL** block, four root pressures that define the threshold for anoxia, the range of optimal root uptake of groundwater, and the threshold for wilting in the **LAND_USE** block.

If there are crops that receive applied water or irrigation, then the **WATER_BALANCE_SUBREGION** block must specify each irrigation type's OFE, and the **LAND_USE** block must include an irrigation flag (zero to indicate no irrigation and non-zero to indicate the specific irrigation type being used). FMP defines the number of irrigation types with the keyword **NIRRIGATE**. Consequently, if a crop is irrigated it should be specified with an irrigation flag from 1 to **NIRRIGATE** to indicate irrigation type used. It should be noted that earlier FMP versions specified a "non-irrigation flag," where 0 (zero) indicated the crop was irrigated and 1 (one) meant it was not. This feature is no longer supported; instead, the "non-irrigation flag" is replaced with an "irrigation flag" that is set to 0 (zero) for no irrigation and greater than zero for irrigation and to identify the irrigation type. Irrigation types are discussed in more detail in the "Redefinition of Irrigation Efficiency" section.

Any combination of consumptive-use concepts is valid as long as all the necessary input information is provided. For example, a user may elect to directly specify the consumptive use of crop 1 and simulate crop uptake using the *linear root response* concept, but for crop 2, to use crop coefficients and the *analytical root response* concept. FMP distinguishes among the different combinations of concepts applied by the user setting flags to zero in the input to signify features not wanted. For example, if one crop has all its soil-water pressures (ψ) set to zero, then FMP uses the *linear root response* concepts rather than the *analytical root response* concepts.

Redefinition of Irrigation Efficiency

Irrigation efficiency is required to determine the irrigation water demand for a given crop irrigation requirement. The irrigation water demand (D) is determined as the crop irrigation requirement divided by the irrigation efficiency (D = CIR/OFE). In the previous versions of FMP, the irrigation demand is referred as the farm delivery requirement (FDR). The specification of irrigation efficiency (OFE) has changed in FMP. The OFE input for FMP was an array-based input specifying a set of efficiencies for each crop in each WBS, resulting in the input-array dimensions NFARM by NCROP. This definition and input structure of OFE limited crops to one irrigation style based on the crop's irrigation efficiency—that is, one OFE per crop. To provide a more realistic way to represent OFE, it has been redefined to allow specification of OFE by irrigation types (number of types, with the keyword **NIRRIGATE**) for each WBS. The number of irrigation types is specified along with the other global dimensions, NWBS, NCROP, and NSOIL, and represents the number of irrigation types that are specified for each WBS. This changes the OFE input from defining NCROP efficiencies for each WBS to NIRRIGATE efficiencies for each the WBS, thereby changing the input-array dimensions to NWBS by NIRRIGATE. For example, if the user specifies three irrigation types representing drip, sprinkler, and flood irrigation, then an OFE can be specified for each irrigation type in each WBS. Actual irrigation-efficiency values are dependent on the local practices, but previous publications (Hanson and others, 2014a) have set irrigation efficiencies within the range of 0.8 to 0.9 for drip, 0.6 to 0.7 for sprinkler, and 0.5 to 0.6 for flood. If desired, users may specify temporal changes in efficiencies in the model input—that is, the OFE can change by stress period.

The redefinition of irrigation efficiency necessitated changing the meaning of the irrigation option in FMP. Previously, it was referred to as "non-irrigation," for which a value of 0 indicated the crop was irrigated and 1 indicated no irrigation. In this version, the irrigation flag of 0 indicates no irrigation, and greater than zero indicates the type of irrigation used for the crop. For example, if there are three irrigation types and the third type is used for a crop, then its irrigation flag would be set to "3" to indicate that it is irrigated using the efficiency defined for irrigation type number 3.

The irrigation efficiency is included as part of several outputs, including the output files that result from the keywords **FARM_BUDGET** (FB_Details.out) and **FARM_DEMAND_SUPPLY_SUMMARY** (FDS.out). For output files that summarize the efficiency by WBS, the output efficiency is an aggregate of the total efficiency in the WBS (dividing the sum of all requirements, ΣCIRs, by the sum of the irrigation demands, ΣD). It should be noted that when a simulation includes deficit irrigation, OFE can either remain constant under deficit irrigation (default in FMP) or if the keyword **EFFICIENCY_IMPROVEMENT** is included, then OFE can increase to represent farmers being more efficient under water shortages. The "Irrigation Efficiency under Deficit Irrigation" section includes a detailed description of efficiency improvement.

Groundwater–Root Interaction Options

A crop simulated in FMP has a specified root depth that interacts with the water table. This interaction is dependent on the distance between the water table with its associated overlying capillary fringe and the bottom of the roots, and it can result in root groundwater uptake or an anoxic reduction in transpiration. Previously, root groundwater uptake, anoxia, and soil-moisture stress were calculated and applied to the crop's final consumptive use. This feature is applied using keyword **GROUNDWATER_ROOT_INTERACTION** and is now optional on a crop-by-crop basis. There are five levels of groundwater–root interaction. There is also a zero-level to indicate the crop is dead and, consequently, has no consumption. Table 3 presents each of the crop root–groundwater interaction levels and indicates whether FMP is applying root groundwater uptake, anoxia, and soil stress for each one. See appendix 6 for a detailed overview and explanation of **GROUNDWATER_ROOT_INTERACTION** and how it is applied in the FMP input **LAND_USE** block.

Table 3. Crop root–groundwater interaction levels.

[—, not applicable]

Level	Groundwater root uptake	Anoxia reduction	Soil stress reduction
0	—	—	—
1	No	No	No
2	No	Yes	Yes
3	Yes	No	No
4	Yes	No	Yes
5	Yes	Yes	Yes

At the first level of groundwater–root interaction, there is no interaction between the groundwater and the crop roots. This prevents any reduction in consumptive use due to anoxia or soil stress conditions and does not allow the crop to consume any groundwater directly (it can be indirectly satisfied through precipitation and applied irrigation). This method is most appropriate for crops that are always disconnected from groundwater or do not suffer from anoxia, such as rice.

At the second level, anoxia and soil stress can reduce transpiration by the crop and, consequently, the consumptive use. Root groundwater uptake is not allowed; thus, all water consumption is from precipitation and irrigation. This option is the least useful, and not recommended, because it allows for high groundwater conditions to affect the crop while disconnecting any actual consumption. This option is most appropriate if most of the crop's consumption is supplied primarily from precipitation and irrigation, but the root groundwater uptake is negligible. This allows FMP to determine the anoxia and soil-stress the crop is experiencing, but not any consumption of groundwater.

The third level is the opposite of the second; root groundwater uptake is allowed, but transpiration is not decreased by anoxia or from soil stresses. To achieve this, the anoxia and soil-stress quantities are calculated; then, any root groundwater uptake is added to the uptake amount rather than deducted from the consumptive use. If there is no root groundwater uptake, then the anoxia and soil-stress quantities are added to the water demand from surface sources of precipitation and irrigation.

At the fourth level, root groundwater uptake and soil-stress losses are allowed, but anoxia is not. This is similar to the third level, except that soil stress is deducted from the total consumptive use, but anoxia is not added either to the root groundwater uptake or to the water needed from surface sources.

At the fifth level, full interaction between groundwater and the crop root is allowed. If anoxia and soil stress are present, then the crop's reduced transpiration consumes only groundwater by root uptake (not precipitation or irrigation consumption). In previous releases of FMP, this fifth level was the only option; now it is the default interaction for all crops if the groundwater-root interaction flag is missing from the FMP input.

Implementation of Deficit Irrigation

If FMP does not have enough water supply to meet the water demand for a WBS, one of two options is taken: water from an external source is used to meet the supply shortfall, or deficit irrigation is implemented. External source water is called the Zero-Scenario and indicates that a WBS can obtain an unlimited supply of water from outside the simulation domain to meet an irrigation-supply shortfall. Conversely, deficit irrigation allows for a deficit between the demanded irrigation and the available water supply (actual applied water). The supply shortfall then results in a reduction in crop transpiration. The reduced transpiration is accompanied by reduced water uptake, which can result in wilting conditions and a reduction in crop yield.

When a WBS has a supply shortfall and is under deficit irrigation, FMP must determine how water is distributed among the crops. In previous versions of FMP, water was distributed by an average supply flow for each irrigated-crop area (WBS supply divided by the irrigated area). Crops that had a demand below this average received no reduction in water. The remaining water supply was then evenly distributed among the remaining crops. The formal equations for this method of deficit irrigation for all crops (N) in a WBS are as follows:

$$\mathrm{QAVF} = \frac{\mathrm{Supply}}{\sum_{i=1}^{N} \mathrm{Area}_i} \tag{18}$$

$$\begin{aligned} \mathrm{QDEF} &= \sum_{i=1}^{N} \mathrm{QAVF} \times \mathrm{Area}_i - \mathrm{D}_i \\ &\qquad \forall\ \mathrm{i}\ \text{ with }\ \mathrm{D}_i \le \mathrm{QAVF} \times \mathrm{Area}_i \\ \mathrm{QEXC} &= \sum_{i=1}^{N} \mathrm{D}_i - \mathrm{QAVF} \times \mathrm{Area}_i \\ &\qquad \forall\ \mathrm{i}\ \text{ with }\ \mathrm{D}_i > \mathrm{QAVF} \times \mathrm{Area}_i \end{aligned} \tag{19}$$

$$\begin{aligned} \tilde{\mathrm{D}}_i &= \mathrm{D}_i \\ &\qquad \forall\ \mathrm{i}\ \text{ with }\ \mathrm{D}_i \le \mathrm{QAVF} \times \mathrm{Area}_i \\ \tilde{\mathrm{D}}_i &= \mathrm{QAVF} \times \mathrm{Area}_i + \frac{\mathrm{QDEF}}{\mathrm{QEXC}} \times \left(\mathrm{D}_i - \mathrm{QAVF} \times \mathrm{Area}_i\right) \\ &\qquad \forall\ \mathrm{i}\ \text{ with }\ \mathrm{D}_i > \mathrm{QAVF} \times \mathrm{Area}_i \end{aligned} \tag{20}$$

where

QAVF is average supply flow for each irrigated-crop area for a WBS (L/T),

Supply is total water supply available to the WBS (L^3/T),

i is the crop ID (-),

N is the number of crops (-),

Area_i is Crop *i*'s area (L^2),

D_i is Crop *i*'s initial irrigation demand (L^3/T),

QDEF is sum of area-weighted supply less demand when demand is less than the area-weighted supply (L^3/T),

QEXC is sum of demand less area-weighted supply when demand is greater than the area-weighted supply (L^3/T), and

$\tilde{D}_i$ is Crop *i*'s deficit irrigation demand that was satisfied by supply (L^3/T).

This method of prorating irrigation supply to simulated crops tends to shift the available irrigation water to crops that have lower irrigation demand and causes a larger deficit (supply shortfall) for the high-irrigation demand crops. This method is most suitable if it is preferable to fully satisfy the demand of low-irrigation demand crops at the expense of high-irrigation demand crops. To distinguish this method, it is referred to as ByAverage Deficit Irrigation.

A new deficit irrigation option is the ByDemand Deficit Irrigation, which is now the default option in FMP. ByDemand Deficit Irrigation prorates each crop's irrigation demand by the ratio of the total water supply divided by the WBS irrigation demand. This causes all crops to have an equal percentage reduction in transpiration, allowing for a smaller deficit for high-irrigation demand crops. This can spread the wilting evenly among all the crops. Equations 21 and 22 describe how the ByDemand deficit demand is calculated.

$$DEF_{ratio} = \frac{Supply}{\sum_{i=1}^{N} D_i} \tag{21}$$

$$\tilde{D}_i = D_i \times DEF_{ratio} \quad \forall i \tag{22}$$

where

DEF_{ratio} is deficit ratio used to reduce all the demands equally (-).

To illustrate the difference between ByAverage and ByDemand Deficit Irrigation, table 4 presents a hypothetical set of demands from four crops and the resulting deficit demand. In this example, the crop area and initial demand are given, and the necessary components for the deficit demand were calculated using equations 18–22.

In this example, crops 1 and 2 have low demand per unit area (both are 0.5), so they receive their full demand when the ByAverage method is used, at the expense of crops 3 and 4. Conversely, the ByDemand method reduces all the crop irrigation demands by half, satisfying a larger fraction of the initial demand for crops 3 and 4. ByAverage and ByDemand Deficit Irrigation are specified in the **WATER_BALANCE_SUBREGION** block by the keyword **PRORATE_DEFICIENCY** followed by the keyword "**ByAverage**" or "**ByDemand**." If **PRORATE_DEFICIENCY** is not specified, then the FMP defaults to the ByDemand Deficit Irrigation option.

Table 4. Example of the difference in final irrigation demand using ByAverage and ByDemand Deficit-Irrigation simulation methods.

[Note that "ByAverage" deficit demands are rounded to the nearest integer. **Abbreviations**: L^2, area in model units; L^3/T, volume per time in model units; D, irrigation demand (L^3/T); i, designates the ith crop out of N total specified land use types; ID, identification number]

Crop ID	$Area_i$ (L^2)	Initial demand (D_i)	Deficit demand ($\tilde{D}_i$)	
			ByAverage	ByDemand
1	100	50	50	25
2	200	100	100	50
3	100	200	73	100
4	100	250	77	125

Supply	300
QAVF	0.6
DEF_{ratio}	0.5
QEXC	330
QDEF	30

Crop ID	QAVF × Area
1	60
2	120
3	60
4	60

Irrigation Efficiency Under Deficit Irrigation

Previous versions of FMP assumed when deficit irrigation was enabled and a WBS did not have enough supply to meet its demand, the irrigation efficiency would increase linearly to unity. Increasing the efficiency assumed that irrigation practices improve when there is less water available for irrigation—that is, a farmer is more conservative with his water use during a shortage. For a detailed discussion about this, please see appendix 4, "Satisfying the Potential Transpiration Component." The original method calculated the increase in efficiency by holding the crop irrigation requirement (CIR) constant and setting the total water demand (D_i) equal to the total available water supply for irrigation ($\tilde{D}_i$). The efficiency is recalculated by dividing the CIR by the available supply for irrigation:

$$OFE_i = CIR_i / D_i \tag{23}$$

$$\tilde{OFE}_i = CIR_i / \tilde{D}_i \tag{24}$$

where

- i is any one crop (-),
- OFE_i is crop *i*'s initial irrigation efficiency (-),
- $\tilde{OFE}_i$ is crop *i*'s improved irrigation efficiency (-),
- CIR_i is crop *i*'s crop irrigation requirement (L^3/T),
- D_i is crop *i*'s initial irrigation demand (L^3/T), and
- $\tilde{D}_i$ is crop *i*'s deficit irrigation demand that is equal to the available supply (L^3/T).

If the adjusted efficiency was greater than `1.0`, then it was changed to `1.0`, resulting in a deficit irrigation—that is, the crop is irrigated with the available water at perfect efficiency (OFE = `1.0`). This method is appropriate when it is known that the agricultural irrigation implementation in a WBS becomes more efficient under deficit irrigation.

Typically, it is not easy to change existing irrigation infrastructures or to modify irrigation practices to improve efficiency. Consequently, FMP now has the option to hold irrigation efficiency constant irrespective of water supply. By holding OFE constant, the applied water is either equal to the demanded water (D_i) or to the available irrigation that can be applied to the crop (IRR) taking into account irrigation efficiency (IRR = Supply). To keep the math consistent, when $D_i > IRR$, a new CIR is calculated based on the available irrigation water ($\tilde{CIR}$). The new $\tilde{CIR}$ then would result in a reduction in the crop transpiration due to wilting (eq. 14).

$$\begin{aligned} &\text{If} \quad D_i > IRR_i \\ &\text{then} \quad \tilde{CIR}_i = OFE_i \times IRR_i \\ &\text{and} \quad \tilde{D}_i = IRR_i \end{aligned} \tag{25}$$

$$T_{irrigation} = \frac{\tilde{CIR}_i}{1 + FEI / FTR} \tag{26}$$

$$W_i = \frac{CIR_i - \tilde{CIR}_i}{1 + FEI / FTR} \tag{27}$$

where

- IRR_i is the amount of irrigation water available to crop i (L^3/T);
- OFE_i is crop i's initial irrigation efficiency (-);
- CIR_i is crop i's initial crop irrigation requirement (L^3/T);
- $\tilde{CIR}_i$ is crop i's deficit irrigation requirement (L^3/T);
- FEI is the fraction of evaporation from irrigation, which is the fraction of total cropped area where irrigated water is applied to bare soil (-);
- FTR is the fraction of transpiration, which is the ratio of the basal crop coefficient divided by full crop coefficient that represents the fraction of total cropped area covered by the crop canopy (-);
- $T_{irrigation}$ is the proportion of crop transpiration that originated from irrigation, reduced by the available water supply (L^3/T); and
- W_i is the deficit in a crop's potential transpiration resulting from insufficient water supply (L^3/T).

Holding the efficiency constant simulates irrigation practices that do not become more efficient under deficit irrigation. This is more representative of irrigation practices that have a relatively low efficiency, such as flood or sprinkler irrigation.

To account for different irrigation practices for each WBS and irrigation type, the method of increased efficiency remains available as an option in FMP. The method can be specified in the FMP input in the **`WATER_BALANCE_SUBREGION`** block. This requires including the keyword **`EFFICIENCY_IMPROVEMENT`** and specifying a flag for each WBS and each irrigation type. If the keyword is not present, then FMP defaults to holding efficiency constant and reducing transpiration if there is deficit irrigation.

Supply Well (Farm Well) Redesign and Implementation

A water-balance subregion's water demand is determined by the total consumptive use of all the land uses in it. This water demand is first satisfied by natural sources of water, which are direct uptake from groundwater and precipitation. If the water demand is not fully satisfied from the natural sources, then the remaining demand is met with irrigation water that originates from imported sources (non-routed delivery, NRD), from surface-water sources (semi-routed delivery, SRD), and from supply wells that pump groundwater (Q_{WBS}), in that order. A WBS can be associated with a set of irrigation supply wells. These wells were called "Farm Wells" in previous publications, but have been renamed in this release as WBS Supply Wells (Q_{WBS}).

The implementation of FMP supply wells was redeveloped to increase speed, simplify user-input, and include new features and output options. Supply wells are defined in the FMP input's **SUPPLY_WELL** block. The supply wells have two possible methods for extracting water from groundwater and three potential input configurations.

Traditional FMP Supply Well

The first method by which a WBS supply well can extract water functions identically to the WEL package. This method sets a demanded pumping rate to a model cell. The demanded extraction rate is always satisfied unless the model cell becomes "dry" or if the FMP well-capacity smoothing option is enabled. Capacity smoothing reduces a supply well's capacity (Q_{cap}) if the cell's saturated thickness (eq. 28) is less than a user specified threshold (MT). When the saturated thickness is less than the specified threshold, then Q_{cap} is multiplied by a smoothing factor (Q_{smf}, eq. 29) to determine the supply well's smoothed pumping capacity. This mimics the loss of well production and improves the stability of the groundwater simulation. The threshold can be specified as a fraction of the cell thickness or as a length above the cell bottom:

$$\begin{array}{lll} \text{If} & h_{cell} > TOP_{cell} & b_{cell} = 1.0 \\ \text{else-if} & h_{cell} > BOT_{cell} & b_{cell} = h_{cell} - BOT_{cell} \\ \text{else} & & b_{cell} = 0.0 \end{array} \qquad (28)$$

where

cell is a model cell at a model layer, row, and column (-);

h_{cell} is the simulated hydraulic head of the model cell (-);

TOP_{cell} is the top elevation of the model cell (-); and

b_{cell} is the model cell's saturated thickness (L).

$$Q_{smf} = b_{cell}^{\,2} \times \left(\frac{3}{MT^2} - \frac{2b_{cell}}{MT^3} \right) \quad \forall\, b_{cell} \in 0 \le b_{cell} \le MT \qquad (29)$$

where

MT is saturated thickness threshold that enables smoothing (L), and

Q_{smf} is smoothing factor multiplied by the supply well capacity and is only applied if b_{cell} is in the range of $0 \le b_{cell} \le MT$ (-).

To enable well-capacity smoothing, the **SUPPLY_WELL** block must include the keyword **SMOOTH** followed by the secondary keyword **ByFraction** or **ByThick**. The saturated thickness threshold (MT) can be set as a minimum cell thickness (**ByThick**) or as a minimum cell fraction (**ByFraction**). If the input is specified as a fraction, then it is converted to a cell thickness for each model cell. The threshold can also be specified as a single number applied to all wells, by WBS, or by model layer. For more details on the usage of the keyword **SMOOTH**, see appendix 6.

FMP-MNW2 Linked Supply Well

The second method by which a WBS supply well can extract water uses the MNW2 package to simulate the actual pumping. If a supply well is linked to MNW2, the well's spatial location and construction are defined by MNW2, but the MNW2 well's desired pumping rate (Qdes) is set by FMP (typically in response to demanded pumpage from a WBS). MNW2 determines the actual pumping rate on the basis of Qdes from FMP, the well construction, and aquifer conditions (such as hydraulic head and horizontal hydraulic conductivity). If MNW2 cannot meet the FMP specified Qdes, then FMP adjusts the WBS's supplies accordingly. Specifically, if there is insufficient supply from an FMP-MNW2 linked well, then a WBS may shift its demanded pumping to a different supply well or the WBS may end up in a deficit irrigation situation due to insufficient water supply.

Supply Well QMAXRESET and NOCIRNIQ Options

The Farm Well keywords **QMAXRESET** and **NOCIRNOQ**—previously included in the "flags for auxiliary variables"—are now specified in the **SUPPLY_WELL** block as a global option by WBS rather than by a supply well. Supply wells with **QMAXRESET** flag indicate that if they are linked to MNW2, then the supply well's maximum production capacity (Q_{cap}) is reset to the value specified in the input at the start of each time step instead of at the start of each stress period. MNW2

can reduce Q_{cap} when calculating the production potential of the WBS supply well. The advantage of not resetting Q_{cap} is that typically subsequent time steps have the same production potential, resulting in an improvement on simulation execution time. In practice, this improvement was negligible; consequently, **QMAXRESET** is enabled by default for all WBS. The keyword **QMAXRESET** is only necessary if it is desired to specify it for only certain WBS—or, optionally, the keyword **NO_QMAXRESET** can be specified to entirely disable it.

NOCIRNOQ indicates that a supply well contributes groundwater to meet a WBS demand—through pumping—only if the model cell that it resides in has a crop irrigation requirement (CIR). This is advantageous for representing local deliveries of groundwater when every cell that has an irrigated crop in a WBS also contains a supply well. By default, **NOCIRNOQ** is disabled.

Prorating Farm Supply Well Pumpage

A new option is the ability to define how WBS demanded pumpage is spread across its Farm Wells if the total demand for groundwater pumping is less than the total summed pumping capacity. It is defined for each WBS as follows:

$$QTOTcap_j = \sum_{i=1}^{N} Qcap_i \tag{30}$$

where

$QTOTcap_j$ is the total pumping capacity of the jth WBS (L^3/T),
j is the WBS index ($1 \le j \le NWBS$),
N is the number of wells associated with the WBS (-),
i is the index for one of the wells associated with the WBS ($1 \le i \le N$), and
$Qcap_i$ is maximum pumping capacity of well i, (L^3/T).

This option is initiated by including the keyword **PRORATE_DEMAND** in the **SUPPLY_WELL** block, followed by a method keyword—either **ByAverage** or **ByCapacity**. This proration functions similarly to the way deficit irrigation is applied to crops, as described in appendix 6.

The `ByAverage` option is the way that previous releases of FMP spread pumpage and is the default option if **PRORATE_DEMAND** is not specified. The formal equations for `ByAverage` proration for all wells associated with one WBS are as follows:

$$QAVF = DMD_j / N \tag{31}$$

$$\begin{aligned} QDEF &= \sum_{i=1}^{N} QAVF - QCAP_i \quad \forall\ i \text{ with } QCAP_i \le QAVF \\ QEXC &= \sum_{i=1}^{N} QCAP_i - QAVF \quad \forall\ i \text{ with } QCAP_i > QAVF \end{aligned} \tag{32}$$

$$\begin{aligned} Q_i &= QCAP_i \quad \forall\ i \text{ with } QCAP_i \le QAVF \\ Q_i &= QAVF + \frac{QDEF}{QEXC} \times (QCAP_i - QAVF) \quad \forall\ i \text{ with } QCAP_i > QAVF \end{aligned} \tag{33}$$

where

$QAVF$ is average demanded pumping rate (L^3/T),
DMD_j is the j^{th} WBS total pumping demand (L^3/T),
N is the number of wells associated with the WBS (-),
i is the index for one of the wells associated with the WBS ($1 \le i \le N$),
$QCAP_i$ is maximum pumping capacity of well i (L^3/T),
$QEXC$ is total excess pumping relative to QAVF (L^3/T),
$QDEF$ is total deficit pumping relative to QAVF (L^3/T), and
Q_i is the final pumping rate assigned to the well i (L^3/T).

This proration tends to keep the pumping rate even for all wells, but may under-utilize the large production wells that can be operated at higher pumping rates.

The `ByCapacity` option uses the ratio of the demanded pumping rate (DMD_j) to the total pumping capacity of a WBS to prorate the pumpage across wells. Equations 34 and 35 describe the way the `ByDemand` deficit demand is calculated.

$$QPRO_j = \frac{DMD_j}{QTOTcap_j} \tag{34}$$

$$Q_i = QCAP_i \times QPRO_j \quad \forall\ i \tag{35}$$

where

$QPRO_j$ is ratio used to prorate equally the well capacities for the j^{th} WBS (-).

To illustrate the difference between `ByAverage` and `ByCapacity` methods of distributing pumpage, table 5 presents a hypothetical total pumpage demanded from four Farm Wells that have varied capacities and lists the resulting pumping rates. It should be reiterated that this algorithm is only applied if a WBS-demanded pumping rate is less than the WBS summed maximum pumping capacity ($DMD_j < QTOTcap_j$); otherwise, all the wells in the WBS pump at the maximum rate.

Direct Recharge Option

In FMP, water not consumed by the land-use type (crop), as evaporation and transpiration, becomes either surface-water runoff or deep percolation to groundwater. The deep percolation can be handled by the Unsaturated Zone Flow Package (UZF), to simulate delayed recharge and rejected infiltration, or flow directly to the water table as recharge. A previous limitation was having no way to specify additional deep percolation beyond what was calculated to result from efficiency losses from precipitation and irrigation water. Additional deep percolation could be specified with the Recharge Package (RCH), but this recharge was not included in the WBS budget information and did not offer the option of delayed recharge. Another limitation was that if FMP was linked to UZF, then the UZF input FINF (infiltration rate at land surface) is set to zero; subsequently, FINF is set to the FMP-calculated deep percolation.

To overcome such limitations in FMP, an input called Direct Recharge is now available. Direct Recharge may be used to represent a set of infiltration ponds that obtain their water from external sources or it could represent natural recharge to the groundwater system that is not consumed by the land use. This option is useful for simulating water banking of managed aquifer recharge (MAR).

Direct Recharge is specified as part of the **`CLIMATE`** block and is composed of a two-dimensional array (`NROW` by `NCOL`) that represents water intended to be directly recharged, bypassing crop consumption (transpiration) and bare soil evaporation. This recharge is simulated similarly to the methods used in the Recharge Package, but differs in that the Recharge Package is a user-specified flux that becomes a volumetric rate across the entire model grid, whereas the FMP Direct Recharge array can be either specified as a flux or volumetric rate. The resulting recharge is passed to deep percolation. From there, it is sent to the UZF or directly recharges the water table. This recharge is not a source of demand, but it could be a source of supply through root groundwater uptake. Because Direct Recharge is a source of water—that is, it enters the landscape budget—it is included as a new column in the FB_Details.out file and has the heading Q-drch-in. Direct Recharge leaves the landscape through deep percolation, so it is included in the column Q-dp-out. For separate accounting of deep percolation from crops, this can be calculated as Q-dp-out minus Q-drch-in.

Table 5. Example illustrating the difference in final pumping rate, in L^3/T, between calculations using ByAverage and ByDemand proration methods.

[ID, identification; QCAP, maximum pumping capacity rate of a well; L^3/T, volume per time in model units]

Well ID	QCAP	Well pumping rate	
		ByAverage	ByCapacity
WELL_1	100	100	60
WELL_2	175	175	105
WELL_3	325	160	195
WELL_4	400	165	240

DMD_j	600
QAVF	150
$QPRO_j$	0.6
QEXC	425
QDEF	25

Farm Process Features Removed

Several seldom used features were removed from FMP. These features included **`FLAG_BLOCKS`** input data structure (the new input for FMP is not backward compatible with previous versions), the prior appropriation scheme for ranked appropriation by farms that represented a water rights hierarchy of preferred deliveries, deficit irrigation options for acreage optimization, conservation pool, water stacking, and the LGR "P" flag that automatically translated farm and crop properties from a parent LGR grid to its associated child-model's farms and crops. Functions equivalent to many of these features can be more effectively performed through external optimization wrappers and preprocessing of child-model data. For example, acreage optimization, formerly initialized by `IDEFFL>0`, is no longer included as an option. Instead, it is recommended to use an external optimizer for determining optimal crop placement. This type of crop optimization has been successfully used by Fowler and others (2014 and 2016). Another important change is that the prior appropriation system between FMP and SFR, formerly initialized by `IALLOTSW>0`, is no longer included. If the removed features are desired, the user can use MF-OWHM (Hanson and others, 2014d). The removed features are summarized in appendix 6.

Several relatively minor features also were removed. Farm Well Parameters (NPFWL) have been removed and replaced with an entirely different input structure for Farm Wells (appendix 6). This new input structure supports a variety of scale factors that can provide the same benefit that was included with NPFWL. The daily crop coefficient and root-depth time series for an entire simulation (ICUFL=3 and IRTFL=3) have been removed. An alternative to this setup is to develop equivalent crop coefficients by upscaling the daily values to the length of stress period (or time step). Additionally, crop properties can be specified by time step or stress period. The efficiency behavior flag (IEBFL) options have been removed; efficiency is now held constant until a new efficiency value is loaded. Declaring a non-irrigation season (IROTFL>0) is no longer an option; this can now be set directly with irrigation flags or fallowing cropland. Lastly, the fraction of evaporation from precipitation (FEP) is no longer required input. Its definition required that its sum with the fraction of transpiration (FTR) was always equal to one. Instead of requiring the user to specify it, it is automatically calculated as FEP = 1 – FTR.

Conduit Flow Process (CFP)

Dual-porosity aquifers consist of a primary interstitial porosity of the aquifer or soil and a secondary porosity that may be due to secondary solution, regional fracturing, or both (Freeze and Cherry, 1979). If there are relatively large interconnected voids or fractures, there can be rapid laminar or turbulent groundwater flow through the aquifer. Dual-porosity aquifers are often associated with karst aquifer systems, but could also include volcanic aquifers or anthropogenic settings, such as a system of mine shafts and tunnels. Approximately 10 to 20 percent of the Earth's surface is underlain by karst carbonate aquifers that supply about 25 percent of the world's population with drinking water (Ford and Williams, 1989). The rapid flow in dual porosity aquifers makes them vulnerable to contamination. The contaminants can be rapidly transported through the aquifer through the larger voids of the secondary porosity (that is, conduits; Ewers, 2006). For example, in the Floridan aquifer of the southeastern United States, karst windows and sinks provide direct connections between the surface and aquifer, resulting in increased aquifer vulnerability. Quantification of local to regional porous media flow and preferential flow in conduits is essential to quantifying this vulnerability. This setting provides an ideal example of how quantitative simulation of flow and exchanges between both porosity domains in hydrologic decision models could improve water- and land-management strategies.

Adequate quantification of dual-porosity flow in hydrologic models requires developing a mathematical formulation based on a physical representation of flow in both domains. Traditional groundwater-flow models using the groundwater flow equation to simulate flow through porous media cannot effectively account for the potentially rapid transport of water and solutes. The Conduit Flow Process (CFP; Shoemaker and others, 2008) can simulate turbulent groundwater-flow conditions of dual porosity aquifers. In systems that are strongly influenced by karst conduit flow or fracture flow, the CFP provides a means to represent these embedded flow systems through a porous media, resulting in a more complete representation of flow.

Overview

The CFP was developed in response to the need for the simulation of karst and dual porosity aquifers (Shoemaker and others, 2008). Incorporation of the CFP into MF-OWHM2 allows the simulation of flow processes in highly conductive structures, like pipe networks. The CFP simulates one-dimensional laminar and turbulent steady flow in discrete pipes according to the Darcy-Weisbach equation. Flow in discrete pipes is iteratively solved in the CFP. Discrete pipes are coupled to the matrix continuum through a head-dependent transfer function. In this way, the CFP allows the consideration of karst systems (for example, Saller and others, 2013; Xu and others, 2015a, b; Xu and Hu, 2017) or other highly conductive discrete elements, such as drainage systems or abandoned mines.

There are three modes of operation for the CFP to simulate karst and dual-porosity aquifers (Shoemaker and others, 2008). The CFP mode 1 allows simulation of relevant discrete flow structures, like karst conduits, drainage systems, or abandoned mining shafts. Conduit flow pipes can represent dissolution or biological burrowing features in carbonate aquifers, voids in fractured rock, or lava tubes in basaltic aquifers. These pipes can also be fully or partially saturated under laminar or turbulent flow conditions. Mode 1 couples the traditional groundwater-flow equation with the formulation for a discrete network of cylindrical pipes. Mode 2 may be used to represent any of the three feature types: a porous media in which turbulent flow is suspected under the observed hydraulic gradients; a single secondary-porosity subsurface feature, such as a well-defined, laterally extensive underground cave; or a horizontal preferential flow layer consisting of many interconnected voids. Mode 3 can simultaneously simulate modes 1 and 2 by coupling the discrete pipe-network formulation with a high-conductivity flow layer.

Mode 2 simulates a high-conductivity flow layer that can switch between laminar and turbulent flow and allows the representation of a dual-porosity system without definition of the individual conduit elements. This can be especially useful if knowledge of the distribution and location of karst conduits is limited and there is a regional aquifer that may represent non-discrete conduits as a secondary porosity. In addition, this mode can be useful to represent other types of secondary porosity settings, such as fractured igneous or volcanic rocks or unknown distributions of conduits in anthropogenic settings, such as networks of mine shafts and addits, water or sewer transmission tunnels, or even fractures from land subsidence or tensional faulting.

Previously, the CFP was only available in a separate release of MODFLOW-2005 called MODFLOW-CFP (Shoemaker and others, 2008). A modified CFP code has been integrated in MF-OWHM2 to allow for the simulation of dual porosity. This integrated approach increases flexibility for the application of MF-OWHM2 to a variety of hydrologic settings. Furthermore, the ability to simulate single- and dual-porosity flow using one code allows the evaluation of the importance of conduit-flow processes to model objectives. The CFP is not linked to any of the advanced packages, such as the MNW2, UZF, SFR, HFB, or to other processes such as FMP and SWR.

Improvements

The revised CFP includes selected upgrades and modifications to the original CFP (Shoemaker and others, 2008), including additional boundary conditions, input, storage, and linkages for the modular, three-dimensional, multispecies transport modeling (MT3D):

- Additional mixed boundary conditions (Cauchy, combined Dirichlet-Neumann).
- Time series as input files (for example, for boundaries—important because of highly conductive pipes).
- Additional direct storage for discrete pipe structures (CADS).
- Water release through dewatering discrete elements considered by partially filled pipe storage.
- Improved input routines (conduit height, exchange).
- New linkage between the CFP and post-processing transport routines (modified version of MT3D).

The CFP input routines and data structures are described by Shoemaker and others (2008), and the revised and upgraded features are summarized in more detail in appendix 7. Slight modifications of selected input files are described in appendix 8. Although additional examples are available in the release package of MF-OWHM2, the use of CFP is not included in the example problem presented in this report.

MF-OWHM2 Example Problem

To demonstrate the functionality of MF-OWHM2, an example problem is presented that uses the SUB, FMP, SFR2, UZF1, NWT/UPF, and MNW2 packages to demonstrate the new linkages and flow interdependencies. This example model was originally distributed with MODFLOW-FMP2 (Schmid and Hanson, 2009a), was later modified to demonstrate the effects of deformation-dependent flows (Hanson and others, 2014d), and here is further modified to show the additional linkage between FMP3 with NWT and MNW2. The problem is used to compare results with and without the new Salinity Demand function now available in FMP. Although not all features of these processes and packages are included in this example, it illustrates many of the fundamental features needed in regional hydrologic models to simulate and analyze water movement and conjunctive use by irrigated agriculture, natural vegetation, and urban areas in a supply and demand framework. A full suite of example problems that can be tested using MF-OWHM2 are included in the distribution package. This includes the LGR2 example (Mehl and Hill, 2013) with the boundary flow and head package (Mehl and Hill, 2013) and the SWR1 and Sublink example (Schmid and others, 2014; Hanson and others, 2014b). Selected input and output datasets are shown in appendix 6, and the complete datasets are included with the distribution package of MF-OWHM2. Additional example problems for CFP applications are also included in the release package for MF-OWHM2. All previous examples from all other packages are also included in the release package, including the example problems from the previous versions of FMP.

Model Structure and Input

The spatial discretization, boundary conditions, and structure of wells, rivers, canals, drains, farms, and other landscape features are summarized in figure 11. The GHB at the upgradient and downgradient edge of the model domain were from the example problem accompanying the MODFLOW-FMP2 user guide (Schmid and Hanson, 2009a); these head boundaries were used for an initial steady-state stress period to develop the predevelopment boundary inflows and outflows. The steady-state stress period was followed by 10 years of the transient simulation that had monthly stress periods. The model grid consisted of 23 rows and 20 columns in a uniform, horizontal spacing of 500 meters (m) and of 7 layers having thicknesses ranging from 60 m to 94 m. The original version of this example problem (Schmid and Hanson, 2009a) used the Layer Property Flow package (LPF) and the Preconditioned Conjugate Gradient Solver package (PCG) for simulating the aquifers and solving the equations of surface-water, landscape, and groundwater flow. For this example, the combination of LPF and PCG was replaced with the Upstream Weighting package (UPW) used in concert with the Newton-Raphson solver package (NWT; Niswonger and others, 2011).

The movement and use of water across the landscape simulated by FMP were represented by eight "virtual farms" or WBSs. These WBSs included five irrigated agricultural areas, an urban area, a non-irrigated riparian wetland, and a region of natural vegetation that represented a largely undeveloped landscape surrounding the other seven WBSs (fig. 11*A*). It should be noted that this example problem section uses the terms "virtual farm," Farm, and WBS synonymously. The landscape was covered by six vegetation types that represented vegetable row crops, orchards, winter grains, urban lawns and gardens, natural vegetation, and riparian vegetation. The remaining features used to simulate consumption, recharge, and runoff were summarized by Schmid and Hanson (2009a).

The model (Hanson and others, 2014d) included three model soil types (fig. 12*B*) and seven hydrostratigraphic layers representing four aquifers and three intercalated confining-bed layers (fig. 11*B*). The streambed elevations of diversion segments followed the slope of a variable ground surface at defined depths (Schmid and others, 2006; Schmid and Hanson, 2009a, p. 93), which allowed local variation in size and slope of streambeds and changes in slope resulting from land subsidence. Using the default interpolation of the SFR between streambed elevations at up- and downstream ends of diversion segments would create streambed elevations that either cut through variable morphological relief or were above the land surface. In addition, linear interpolation between different elevations would create relatively steep slopes that do not allow detection of code limitations that arise for minimal slopes using Manning's equation (slope in the denominator leads to overestimation of stream stages). FMP was also linked to MNW2 by multi-node wells screened across several layers that supply water to Farm 5 (UZF Farm) and Farm 6 (urban area; fig. 11*A*). The MNW1 wells that were in the original version of this example (Schmid and Hanson, 2009a) were replaced with MNW2 wells. FMP was also linked to UZF to simulate unsaturated-zone processes under farm 5 and farm 8 to include the effects of rejected infiltration and groundwater discharge to the surface in Farm 8 (riparian area; fig. 11*A*).

Although all model cells did not necessarily need to be assigned to specific model farms in FMP, in this example, all model cells of the model domain were assigned to eight "virtual farms" that represent water-accounting regions. Six of these "virtual farms" were associated with Farm Wells (supply wells) for the potential delivery of groundwater, if needed (fig. 11*A*). There were two additional non-irrigated, rain-fed water-accounting regions that represented a riparian wetland on the eastern boundary surrounding the river outflow (virtual farm 8) and the natural vegetation in the remainder of the model (virtual farm 7). The SWR canal that was in the MF-OWHM example (Hanson and others, 2014d) was removed from this example.

The example model included six virtual crop types that represented groups of crops aggregated by similar crop coefficients and growth-stage lengths (fig. 13). Although FMP provides the option to change the spatial distribution of crop types from stress period to stress period (representing crop rotation), in this example, the distribution of crop types did not change through time. Crop-type 1 represented vegetable row crops consisting of 20-percent cabbage, 50-percent lettuce, and 30-percent green beans. Crop-type 2 represented apple, cherry, and walnut orchards. Crop-type 3 represented winter grains, such as barley, wheat, and oats. The landscaping of the urban area, crop-type 4, represented lawns and gardens, which were simulated with crop coefficients for turf. Crop-type 5 represented natural vegetation comprising equal areas of grazed pasture, grass and clover, a wildlife area, and non-agricultural trees and vines. Crop-type 6 represented a riparian area of willows, which can take up water under variably saturated conditions.

For each crop group, weighted averages of individual crop coefficients and growth-stage periods were computed using the percentage contribution of each individual crop. The individual values for initial-, mid-, and end-season crop coefficients as well as the periods for initial, mid, and late growth stages were compiled from published databases in various sources of literature (Allen and others, 1998, 2005; Food and Agriculture Organization, 2007). For each crop group represented by its average growth and harvest attributes, a daily time series (365 days) of crop coefficients was calculated using the "composite" crop coefficients and "composite" growth-stage periods. Finally, crop coefficients were calculated for each month of the year using the daily time series, and the 12 average monthly crop coefficients were applied to the 10-year simulation period for stress periods 1 through 12 and 13 through 24 of the example model. The monthly crop coefficients allowed the different types of vegetation to be active at different times of the year as they each cycled through their seasonal growth stages (fig. 13*A*). The virtual crop coefficients for the virtual crop types (crop groups) described previously were preprocessed for the example model prior to assembling the FMP data input. The technique and algorithms applied were formulated in Excel spreadsheets that also contained a compilation of crop coefficients and growth-stage time spans obtained from published sources. These Excel spreadsheets are provided in the release package of MF-OWHM2. Other approaches to preprocess crop coefficients for each model stress period are possible.

Crop-specific parameters required by the FMP include fractions of transpiration (FTR) and fractions of evaporation (FEI) related to precipitation and irrigation for the six crop groups. The separate simulation of transpiration and evaporation is an essential difference between FMP and many other hydrologic models, which assume a common extinction depth for a composite evapotranspiration term. In FMP, evaporation from groundwater is extinct at a depth to water equal to a specified capillary fringe, and transpiration from groundwater is extinct at a depth to water equal to the root zone plus the capillary fringe. The example problem simulated crop transpiration under unsaturated conditions (crop-types 1 through 5) as well as saturated conditions (for example, crop-type 6 simulated as riparian willows) by analytical solutions. Fractions of transpiration and evaporation were varied by month (figs. 13*B–C*).

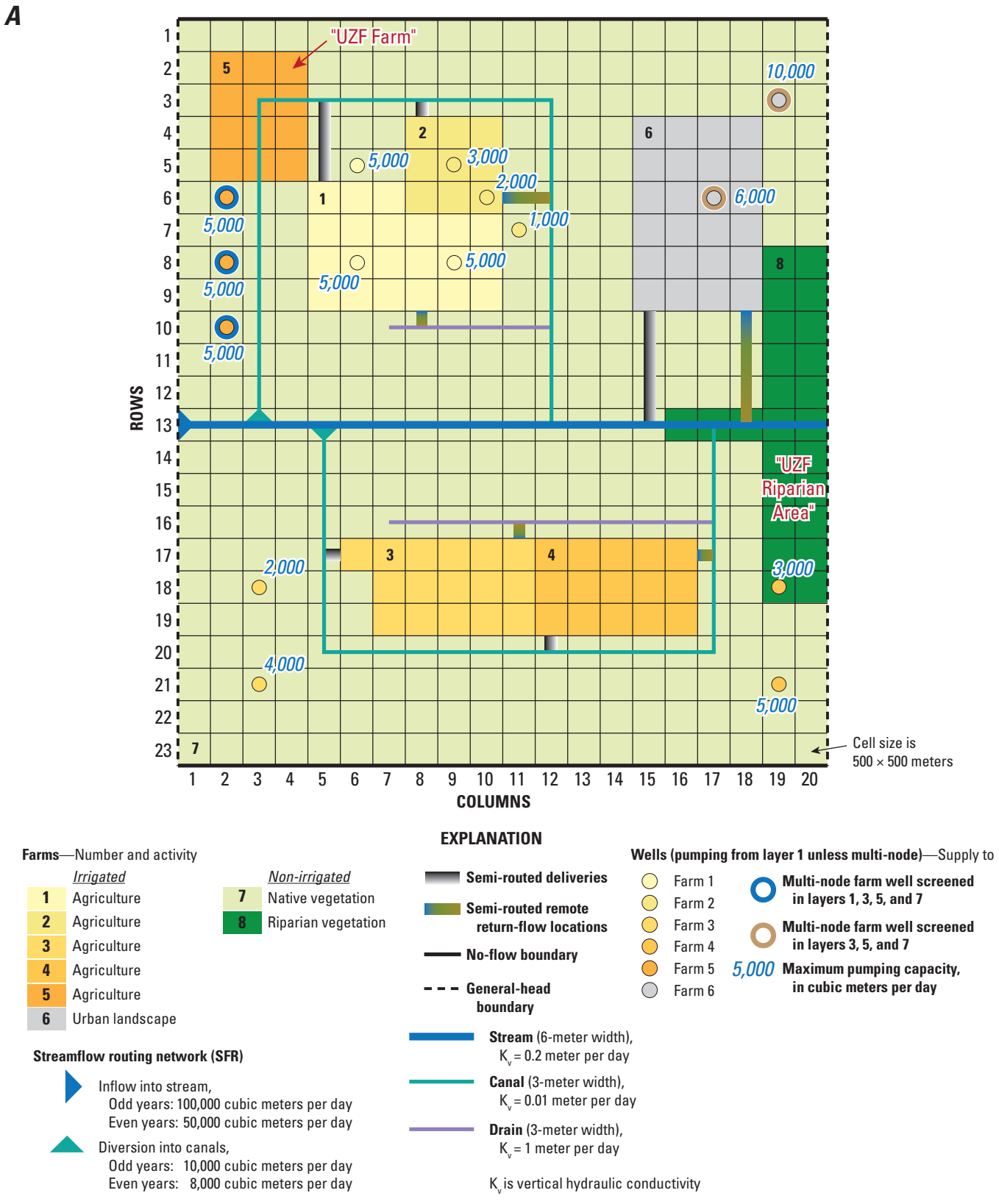
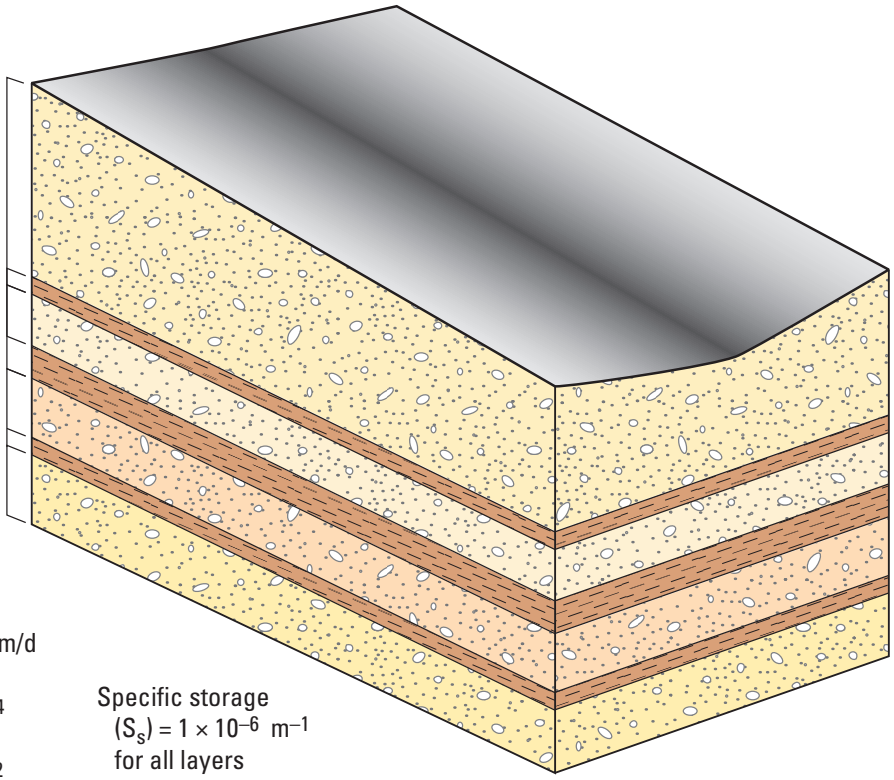


Figure 11. Example model structure and features: *A*, plan view of model domain, grid resolution, boundary conditions, distribution of farms and Farm Wells (supply wells), and streamflow-routing network with points of diversion to farms, points of return flow from farms, and surface-water canal traversing an urban area; *B*, block view of model layering; and *C*, simulated land subsidence (from Schmid and others, 2014; Hanson and others, 2014d).

B



Layer 1: (unconfied)
- Specific yield (S_y) = 0.12
- Hydraulic conductivity (K) = 3 m/d
- Elastic skeletal storage coefficient (S_{ke}) = 5.02×10^{-4}
- Inelastic skeletal storage coefficient (S_{kv}) = 5.02×10^{-2}

Specific storage (S_s) = 1×10^{-6} m^{-1} for all layers

Layer 2 (confining bed at bottom of Layer 1):
- K = 0.1 m/d
- $S_{ke} = 1.5 \times 10^{-5}$
- $S_{kv} = 1.5 \times 10^{-3}$

Layer 3: (confined)
- K = 2 m/d
- $S_{ke} = 3.6 \times 10^{-4}$
- $S_{kv} = 3.6 \times 10^{-2}$

Layer 4 (confining bed at bottom of Layer 3):
- K = 0.1 m/d
- $S_{ke} = 4.5 \times 10^{-5}$
- $S_{kv} = 4.5 \times 10^{-3}$

Layer 5: (confined)
- K =2 m/d
- $S_{ke} = 3.6 \times 10^{-4}$
- $S_{kv} = 3.6 \times 10^{-2}$

Layer 6 (confining bed at bottom of Layer 5):
- K = 0.1 m/d
- $S_{ke} = 1.5 \times 10^{-5}$
- $S_{kv} = 1.5 \times 10^{-3}$

Layer 7: confined
- K =2 m/d
- $S_{ke} = 3.6 \times 10^{-4}$
- $S_{kv} = 3.6 \times 10^{-2}$

C

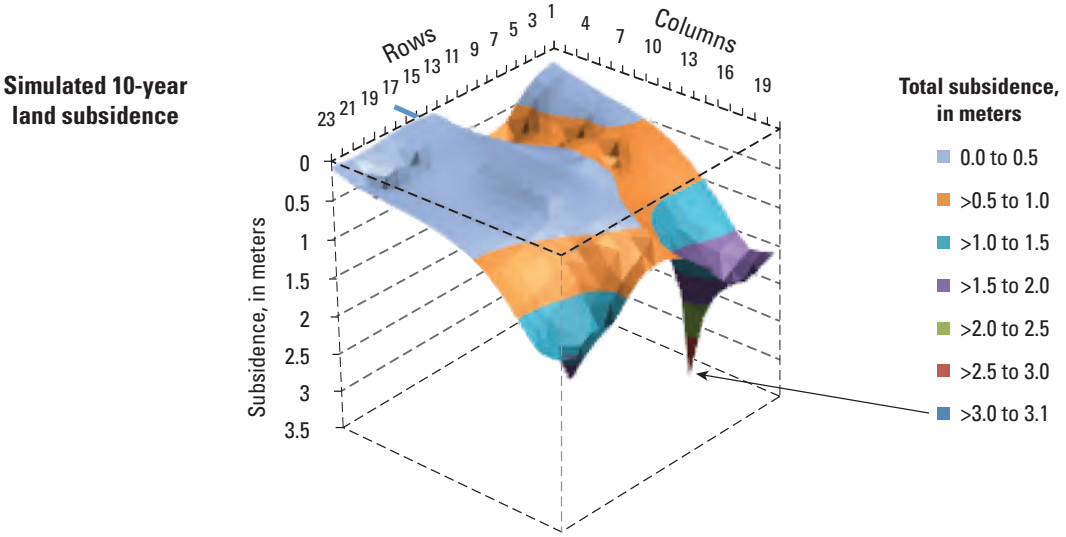


Figure 11. —Continued

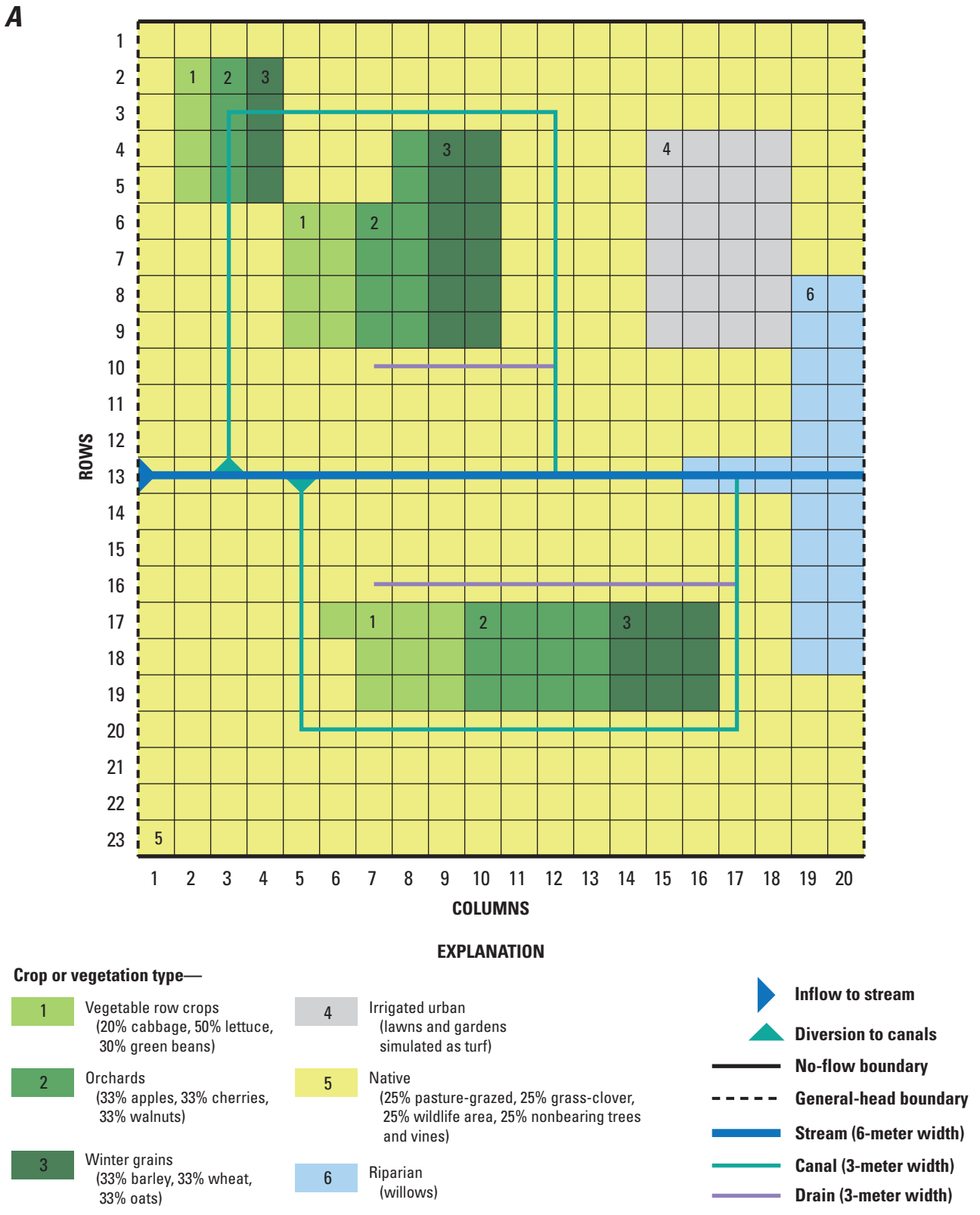


Figure 12. Model grid of MF-OWHM2 example problem showing *A*, crop and other vegetation distribution, and *B*, distribution of soils (Schmid and Hanson, 2009a).

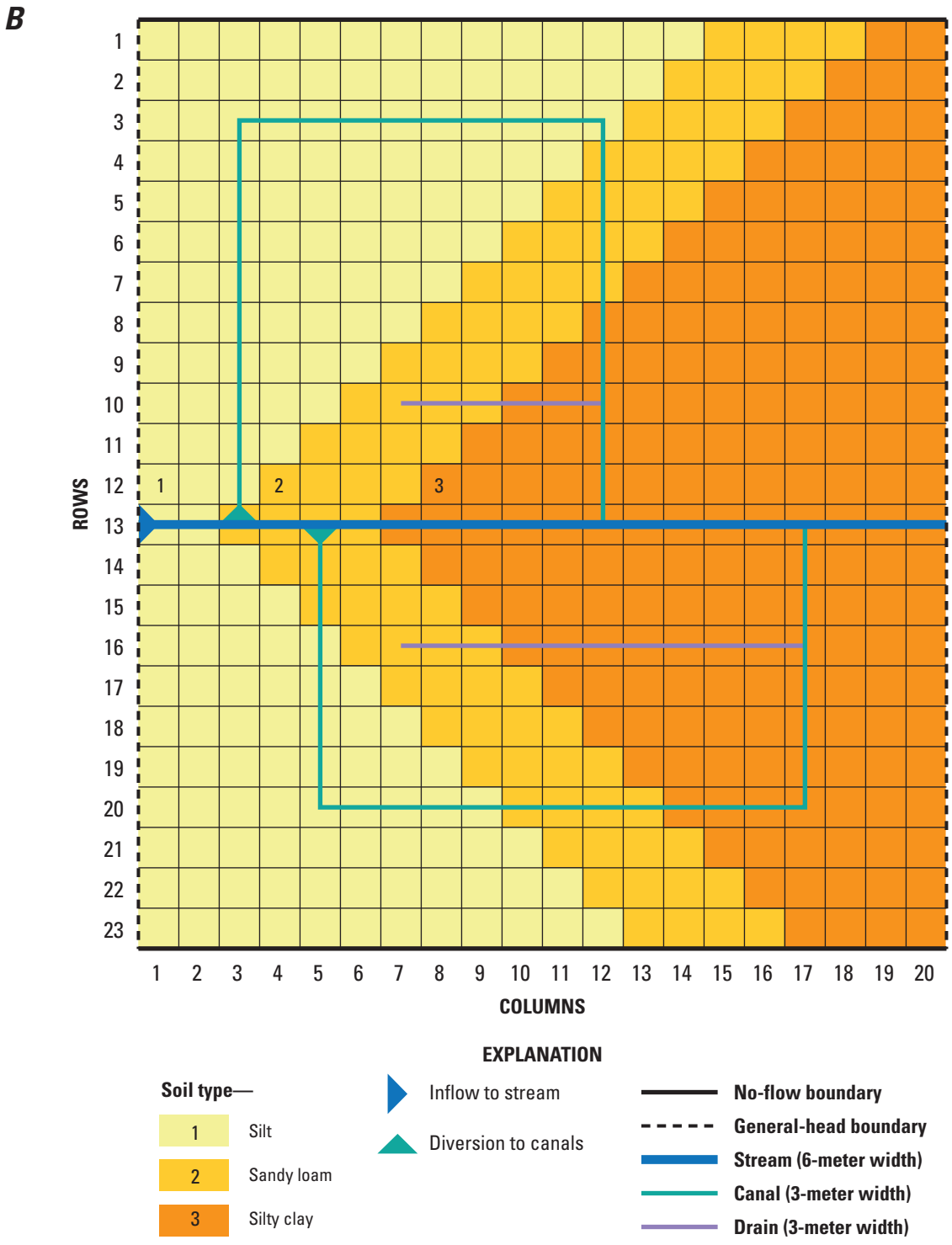


Figure 12. —Continued

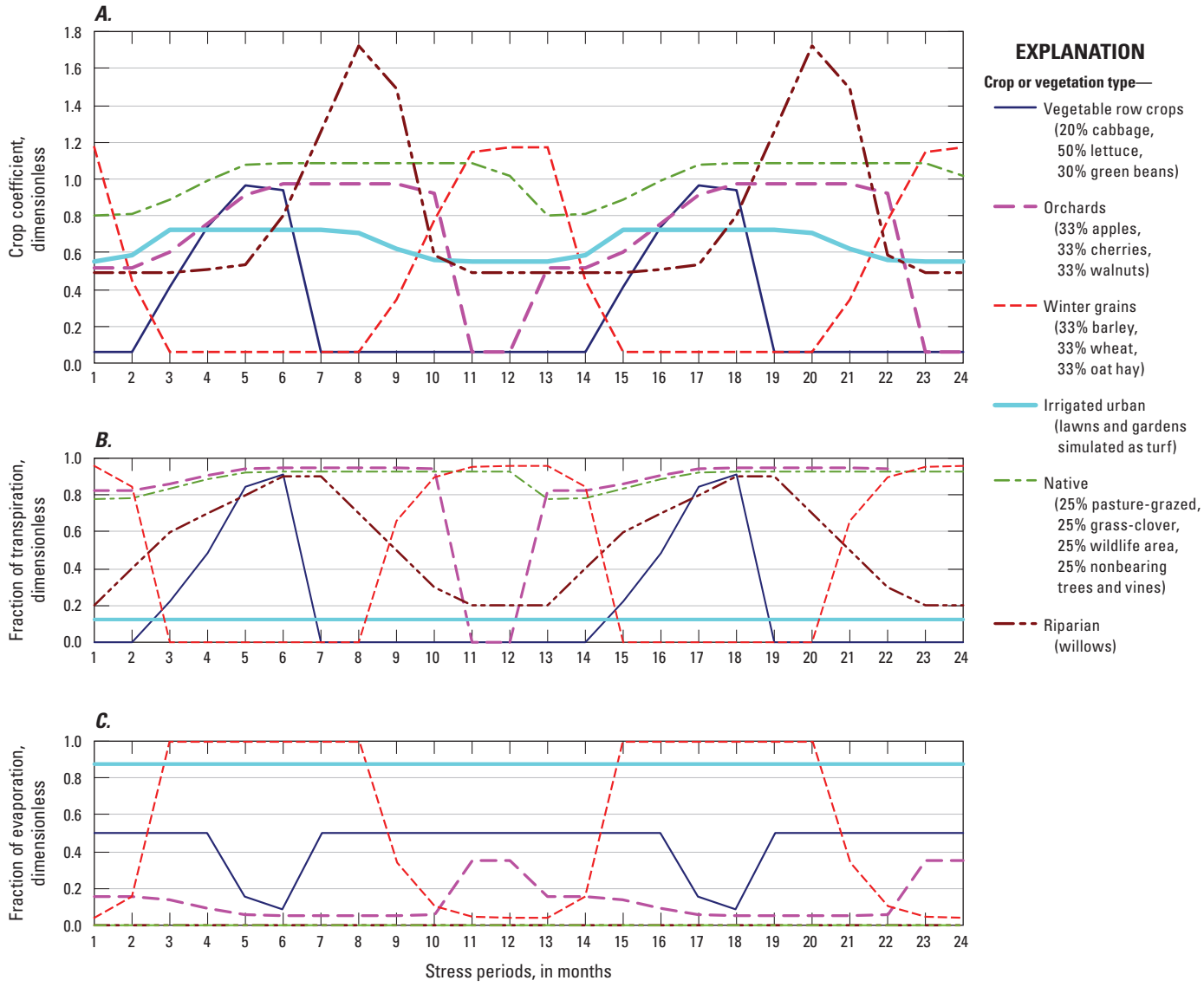


Figure 13. Results for the six virtual-crop types as monthly time series for the MF-OWHM2 model problem (Schmid and Hanson, 2009a): *A*, crop coefficients, K_c; *B*, fractions of transpiration, FTR; and *C*, fractions of evaporation related to irrigation, FEI.

The fraction of transpiration, FTR, can be derived as $FTR = K_{cb}/K_c$ if, in addition to the total crop coefficient, K_c, a "basal" crop-transpiration coefficient, K_{cb}, is available (Allen and others, 1998, 2005; Food and Agriculture Organization, 2007). The fraction of evaporation for exposed areas wetted by precipitation, FEP, depends on the exposed bare-soil surface wetted by precipitation. Even though transpiration and evaporation may be related nonlinearly, for the virtual crop-types 1 through 3 and 5 in this example model, we simplified the fraction of evaporation to be equal to the complement of the fraction of transpiration—that is, FEP = 1 – FTR. The fraction of evaporation related to irrigation (FEI) depends on the fraction of the bare-soil surface wetted by irrigation. Unlike the soil surface wetted by precipitation, the exposed areas wetted by irrigation may not be entirely wetted. The extent to which the exposed area is wetted depends on the irrigation method used, which commonly is related to the specific crop type. For the virtual crop-types 1 through 3 in the example model, the fraction of transpiration related to irrigation was assumed to be constrained by the lesser of complement of the fraction of transpiration or the wetted fraction, fw, for certain irrigation methods (Allen and others, 1998; Allen and others 2005; Food and Agriculture Organization, 2007). That is, FEI = min[1 – FTR, fw]. Fractions of transpiration and evaporation are FMP parameters that often are uncertain, and MF-OWHM2 models are sensitive to these parameters (Schmid and others, 2008). The demonstrated approach is one of many ways the fraction of transpiration and evaporation can be either physically based or based on published data. Rough initial estimates of these fractions may be specified, but the user is advised to refine these parameters using estimates derived during the model-calibration process.

For the urban area (crop-type 4), the fraction of transpiration was assumed to be equal to the fraction of the entire area in which there is transpiration (for example, turf and gardens). In many cases, land-use surveys specify the percentage of irrigated land in urban areas. In the example model, an average value of percentage range (for example, 12.5 percent as the average of 0 to 25 percent) was used to represent the fraction of the area (that is, 0.125) in which there was transpiration. The fraction of evaporation was then assumed to be equal to the fraction of the entire urban area that was open and exposed (such as housing and other buildings, parking lots, industrial sites, airports). For the natural vegetation (crop-type 5), the fraction of evaporation related to irrigation was specified using placeholder zero values because no irrigation was applied. For riparian vegetation (crop-type 6), the fractions of transpiration and of evaporation related to precipitation were assumptions. No basal crop coefficients, K_{cb}, were found in published sources that could be applied. The fractions of evaporation related to irrigation were also placeholder zero values because no irrigation was applied.

The model represented three soil types that are internally defined by the FMP as silt, sandy loam, and silty clay (fig. 12*B*). Root depths were specified for all crop types for every stress period (IRTFL = 2); the depths were varied for some of the crop types, such as vegetable row crops and winter grains, but were held constant for the others. For the example model, the maximum rooting depth used was the average of values available from Allen and others (1998, table 22) and Brush and others (2006). For perennial crops such as orchards and turf or for natural and riparian vegetation, the rooting depth was assumed to be constant through time. For annuals like vegetable row crops and winter grains, the root-zone depth was assumed to vary proportionally to the crop coefficient of each stress period using a proportionality factor equal to the ratio of maximum rooting depth to maximum crop coefficient. This algorithm is used when the crop coefficient increases or remains constant at its maximum or minimum.

$$RZ^t = \left(\frac{RZ_{max}}{K_{c-max}}\right) \times K_c^t, \text{ if } K_c^t \geq K_c^{t-1} \text{ or } K_c^t = K_{c-min}$$
$$RZ^t = RZ^{t-1}, \text{ if } K_c^t < K_c^{t-1} \text{ and } K_c^t \neq K_{c-min} \quad (36)$$

where

t	is the time index,
RZ^t	is the root-zone depth at time t (L),
K_c^t	is the crop coefficient at time t (-),
$K_{c\text{-}max}$	is the maximum crop coefficient (-), and
$K_{c\text{-}min}$	is the minimum crop coefficient (-).

During the final stress period of the growing season, the crop coefficient declined until harvest. Nevertheless, the maximum root zone reached during the growth mid-period was assumed to remain at the maximum until the crop coefficient dropped to the off-season minimum value corresponding to harvest or senescence.

Fractions of inefficient losses (delivery losses) to surface-water runoff were specified for each virtual crop type in each stress period. In the FMP, surface-water runoff is assumed to depend on irrigation methods, which in turn may depend, in part, on the crop type. Because rainfall intensity and irrigation application methods also influence runoff, the FMP requires input of two separate fractions of inefficient losses to surface-water runoff—one related to precipitation (FIESWP) and another related to irrigation (FIESWI)—which may be omitted or set to placeholder zero values for non-irrigated crop types, such as natural (crop-type 5) and riparian vegetation (crop-type 6). In the example model, FIESWP and FIESWI were held constant through time for crop-types 1 through 4. The FIESWP increases for natural (crop-type 5) and riparian vegetation (crop-type 6) during the winter–spring months, however, indicating an increased fraction of inefficient losses to runoff during the heavy winter–spring precipitation typical of the climate in Davis, California. Additional runoff components were calculated by the UZF-FMP link for farm 5 and the riparian area (farm 8) stemming from infiltration in excess of the saturated hydraulic conductivity, the groundwater discharge to land surface, and rejected infiltration for high groundwater levels. In the FMP, two flags indicate the design of the runoff return-flow routing system (see later). In the UZF1, a two-dimensional integer array, IRUNBD, specifies the SFR streamflow segment in which the potential runoff is returned to the river for each UZF-active cell (Schmid and Hanson, 2009a, appendix A).

Crop-specific parameters, such as crop coefficients, root-zone depths, fractions of transpiration and evaporation, and fractions of inefficient losses to surface-water runoff, can vary by stress period. In contrast, pressure heads that define stress-response function coefficients are the only crop-related set of parameters specified for the entire simulation. Notably, in the FMP, a stress-response function (appendix 5) can define unsaturated and saturated conditions by specifying negative and positive pressure heads, respectively. In the example model simulation, the stress response of riparian willow trees (crop-type 6) to water uptake was described by a stress-response function, in which the optimal uptake was in unsaturated conditions, but reduced uptake was still possible in saturated conditions, until the pressure head reached 20 centimeters and uptake became zero (Schmid and Hanson, 2009a, appendix A, file PSI.IN).

Reference evapotranspiration and precipitation were set to be constant within each monthly stress period, but to vary from stress period to stress period. The input data were derived from the California Irrigation Management Information System (CIMIS) data from the weather station at the University of California, Davis (http://wwwcimis.water.ca.gov/cimis/data.jsp, accessed April 20, 2009). For each month of the year, a median was determined from the monthly values from water year 1982 to 2008.

Surface-water deliveries to irrigated farms included non-routed water transfers from outside the model domain and equally appropriated semi-routed deliveries along a streamflow-routing network simulated using the SFR2 Package. Non-routed deliveries (NRDs) were assumed to be known volumes of deliverable water for each stress period (Schmid and Hanson, 2009a, appendix A, file NRDV.IN). The NRDs were supplied to all but the natural vegetation and riparian areas using a variable monthly scale factor that changed the volume of the NRDs through the course of each model year (Schmid and Hanson, 2009a, appendix A, file NRDFAC.IN). Semi-routed surface-water deliveries to irrigated farms were diverted from specified stream reaches (Schmid and Hanson, 2009a, appendix A, file SRD.IN) outside the farm domain. The term "semi" is used to describe the routing for two reasons:

A. Deliveries are routed along the stream network to a user-specified point of diversion.

B. Deliveries are non-routed (for example, pipe flow) from the user-specified point of diversion (perceived as 'remote head-gate') to the farm.

Semi-routed runoff was returned to the stream network (simulated by SFR2) at a specified location only for virtual farm 1 (Schmid and Hanson, 2009a, appendix A, file SRR.IN). For all virtual farms other than virtual farm 1 (that is all WBS other than 1), FMP automatically prorates runoff to all SFR stream reaches within the WBS. For three farms, virtual-farm 5, the natural vegetation (virtual-farm 7), and the riparian area (virtual-farm 8), stream segments were found within the domain of each farm, and each farm's return flow was prorated to those reaches accordingly. An output file, ROUT.OUT, was written that informs the user about the system of routing deliveries to, and runoff away from, each virtual farm (Schmid and Hanson, 2009a, of which appendix A contains the part of the file that pertains to stress-period 1, time-step 1).

The data input for linked packages is included with the model distribution package, and the reader is referred to the NWT, SFR2, UZF1, SWR1, and MNW2 input instructions for more complete explanations of the NWT, SFR2, UZF1, SWR1, and MNW2 data input used in the example model (Niswonger and Prudic, 2005; Niswonger and others, 2006, 2011; Hughes and others, 2012; Konikow and others, 2009). The streamflow network and its hydraulic properties are summarized in figure 11 along with the location and screening of multi-node wells.

The FMP input features for this example model included temporally distributed precipitation as a specified-flux boundary condition typical of the rainfall for Davis, California, in the Sacramento and San Joaquin Valleys (also known as the Central Valley). This helps facilitate delayed recharge following time-varying supplies from precipitation and irrigation to crops, urban areas, and natural vegetation. The FMP also used semi-routed deliveries and return flows to connect agriculture with surface water derived from the river (fig. 13*A*). The distribution of crops demonstrates the combined use of precipitation and irrigation for winter wheat as opposed to surface and groundwater supplies for irrigation of orchard and vegetable crops grown during the spring and summer.

The SUB Package used steady-state heads from the previous version of the example model as critical heads to enable simulation of subsidence with the onset of pumping. This implies the system is assumed to be normally consolidated at the beginning of the simulation. To ensure that the pumping provided sufficient drawdowns to drive subsidence, the transient model was extended to a 10-year model by repeating the 2 years of monthly stress periods from the FMP model five times. The subsidence package input dataset contains elastic and inelastic specific-storage coefficients (S_{ske} and S_{skv}) of 6×10^{-6} and 6×10^{-4} per meter, respectively, for fine-grained interbeds of all aquifer layers and of 3×10^{-6} and 3×10^{-4} per meter, respectively, for all confining bed layers (fig. 11*B*). Subsidence was assumed to be instantaneous, with no-delay interbeds or confining beds, and was active in all cells of all model layers. Land subsidence ranged from 0 to 3.1 m and was greatest under the city, near the urban supply wells, after the 10 years (fig. 11*C*).

Salinity Demand

This example demonstrates the simulation of additional demand for irrigation required for leaching salts from the soil zone. The salinity demand can be selectively applied to specific crops and to specific virtual farms (WBSs). In this example, the salinity demands were applied to the five agricultural virtual farms (WBS 1–5) for all the crop types (vegetable row crops, orchards, and winter wheat) and to the urban farm (WBS 6) for the urban landscape (turf grass). The RHOADES option was used in the salinity block to estimate the leaching requirement and the applied water. The crop salinity tolerances were specified from previously published values (Cahn and Bali, 2015; Ayers and Wescott, 1985), and vegetable row crops were represented by strawberries (640 mg/L), orchards represented by grapes and almonds (960 mg/L), grains represented by winter wheat (3,840 mg/L), and urban landscape represented by turf grass (704 mg/L). The salinity of the water sources was set to values typical of some of the coastal California basins with user-specified constant salinities for surface-water (309 mg/L), groundwater (216–340 mg/L), and nonrouted deliveries (510 mg/L for agriculture and 610 mg/L for the urban farm receiving recycled water). The irrigation uniformity represents how uniformly the irrigation is applied with respect to a crop's root depth, where a value of 1 indicates perfectly uniform, and 0.5 is 50 percent of uniform. The irrigation uniformity with respect to depth in the root-zone varied by irrigation type and was set to 0.80 for sprinkler-soaker hose, 0.85 for drip, 0.75 for pivot irrigation, and 0.9 for urban sprinkler.

Unsaturated Flow

The linkage to the UZF1 package facilitated delayed recharge through the unsaturated zone in the upgradient (western) part of the example model domain, such as at virtual farm 5 (fig. 11*A*). This linkage also allowed simulation of rejected infiltration in the riparian areas in the discharge region along the river outflow at the eastern part of the model domain (virtual farm 8; fig. 11*A*). The areas where this linkage was active (specified through the UZF Package input in the IUZFBND array) were coincident only with virtual farm 5 and the riparian area (virtual farm 8). The additional unsaturated-zone properties specified included a Brooks-Corey epsilon of 0.35, a saturated water content of 0.2, an initial water content of 0.16, and a saturated vertical hydraulic conductivity in the unsaturated zone of 0.001 meters per day. The relationship between the land surface and the initial water table at the peak of growing season when water demand caused the water table to lower in model-layer 1 for the unsaturated zone beneath virtual farm 5 is shown in figure 14.

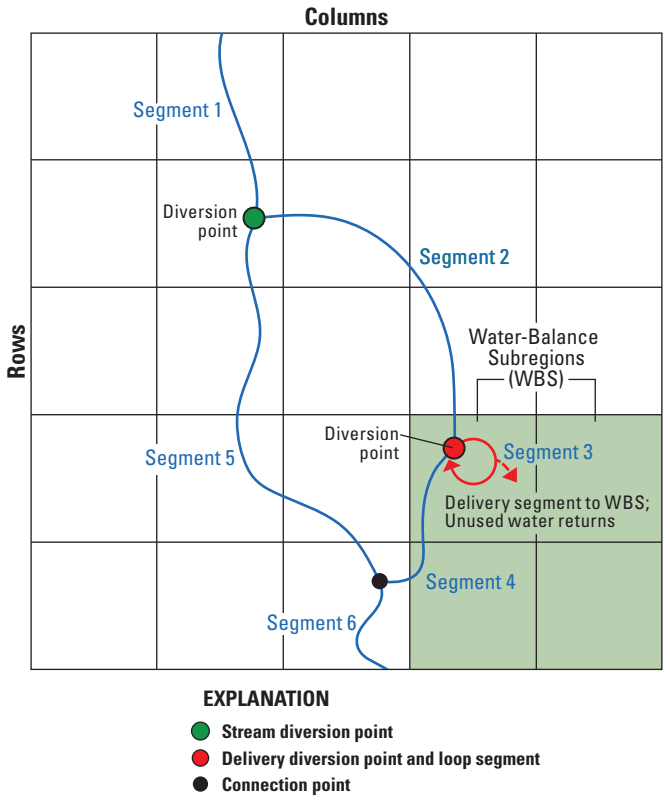


Figure 14. Relation between the land surface and the water table in an unsaturated zone from the MF-OWHM example (modified from Schmid and Hanson, 2009a).

Model Results

Results from the example model are used here to demonstrate how salinity demand, surface-water operations, and unsaturated-zone flow (or rejection) of infiltration were represented by the model.

Salinity Demand

The effects of salinity demand resulted in a large increase in irrigation demand for all the different landscapes and WBSs (fig. 15). The increases in irrigation demand to account for salinity leaching ranged from 22 to 38 percent and varied among the farms. Farms with more vegetable row crops had greater leaching requirements, with the most additional irrigation required for virtual farms 3 and 5 and least for the urban landscape of virtual farm 6. These different farms also required greater percentages of additional irrigation for leaching during the spring and fall (fig. 15*A*). Because each of these WBS received different amounts of surface-water and non-routed deliveries, the additional portions of groundwater needed to accommodate salinity leaching also varied among the WBS and from month to month (fig. 15*B*). The additional leaching demand can also result in variably increased irrigation among crops (fig. 15*C*). In this example, the increase was 22 to 43 percent for vegetable row crops in farms 1, 3, and 5, which have a lower salinity tolerance, whereas the increase was 24 to 34 percent for orchards in farms 2 and 4. Winter grain crops, which made up more than 60 percent of the land use for farms 2 and 4 and received winter precipitation to supplement irrigation, still required a 36-percent increase in irrigation for leaching. This additional irrigation demand can trigger adverse effects, such as reduced surface-water deliveries, land subsidence, and reduced streamflow, any of which can be an important consideration for management of conjunctive use. Such considerations are becoming increasingly relevant with the passage of groundwater laws, such as the Sustainable Groundwater Management Act (State of California, 2014). Other types of water demand for uses like dust control, frost protection, and pest control could also be simulated with this feature and user-specified equations that represent when those additional applications would be needed.

Unsaturated Flow

The effects of unsaturated flow are demonstrated in the example model beneath virtual farm 5 in the northwestern part of the model grid and the beneath the riparian area (virtual farm 8) where the UZF package was activated and beneath these water-balance subregions. Beneath virtual farm 5, there was a relatively large unsaturated zone that delayed recharge about 153 days in the middle of 2004 (fig. 16*A*). Similarly, the effects of rejected infiltration were apparent beneath the riparian area (virtual farm 8) for about a month during the same time (fig. 16*B*).

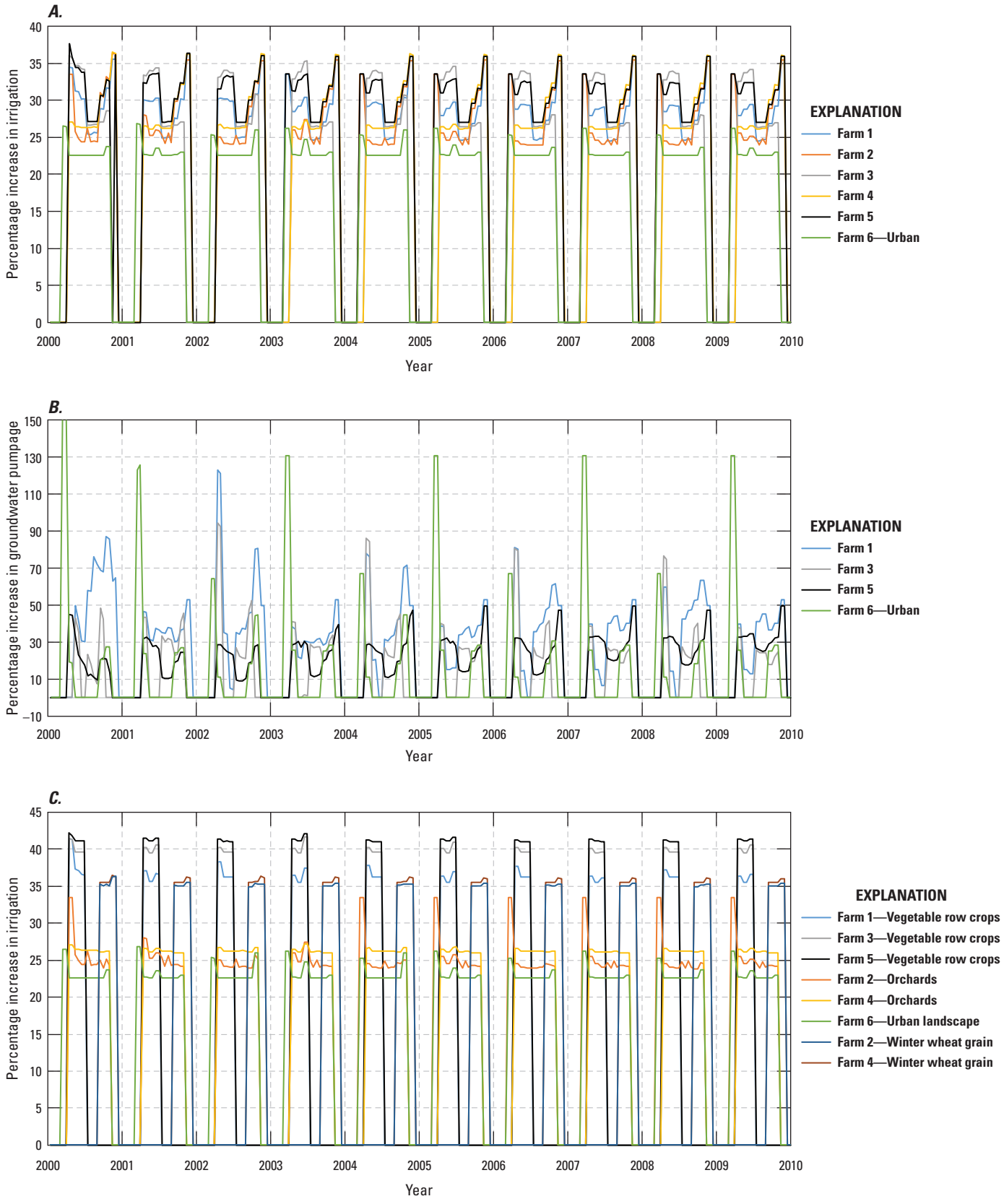


Figure 15. Relative increase with leaching among simulations from the example model with and without the salinity demand option: *A*, irrigation, *B*, groundwater pumpage, and *C*, additional irrigation for selected crops and farms.

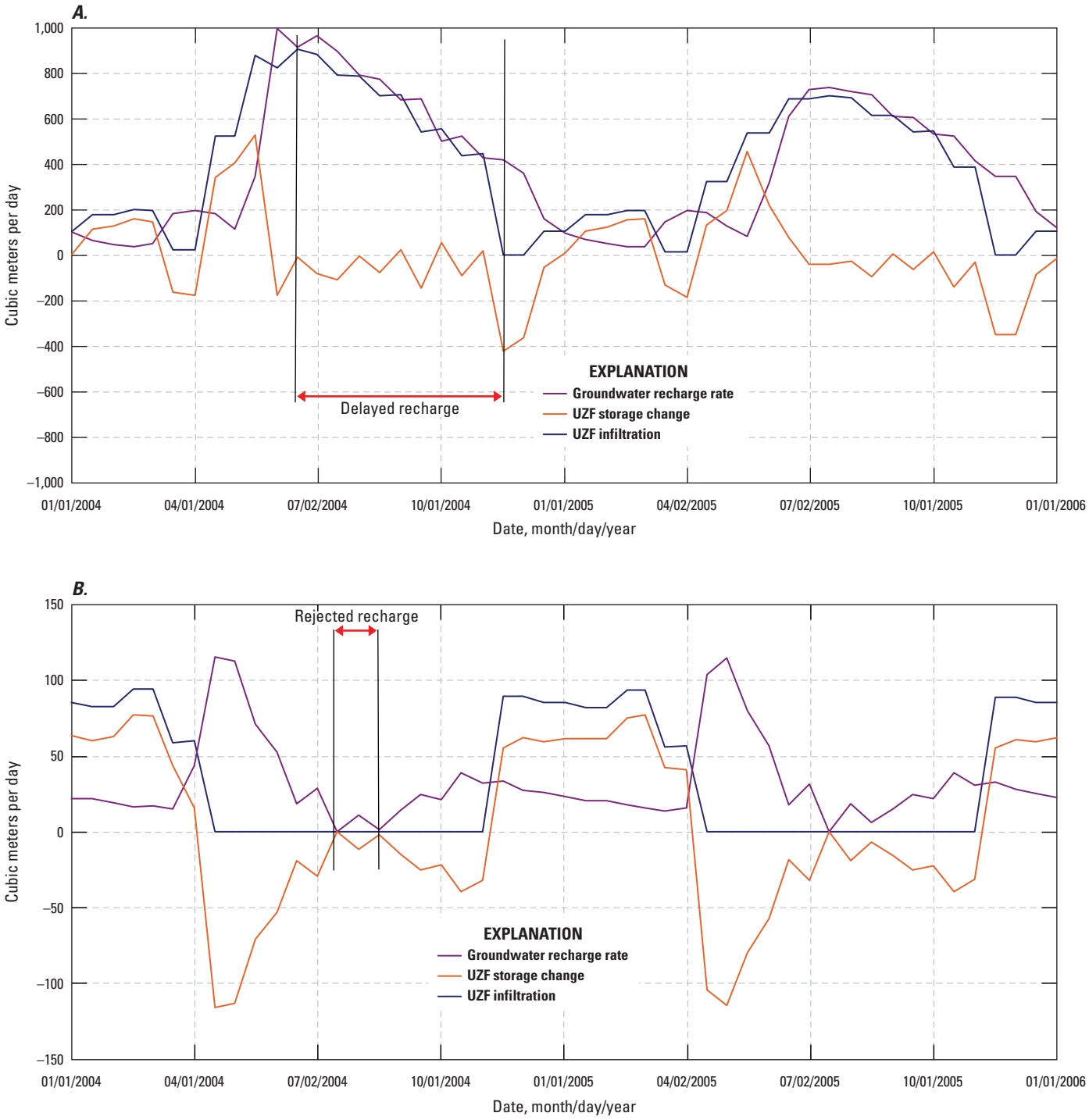


Figure 16. Results from the example model of the effects of unsaturated zone (UZF) on *A*, delayed recharge beneath farm 5, and *B*, rejected recharge beneath the riparian area (farm 8).

Limitations and Future Improvements

In MF-OWHM2, if the natural water supply—root groundwater uptake and precipitation—is not enough to satisfy demand, the remaining water is obtained from supplies in a specific order that cannot be changed. The order is always as follows: non-routed deliveries, that is water delivered without simulating its conveyance; surface-water delivery, that is, water diverted from a surface-water network; any remaining demand is supplied by groundwater pumping. This order of water-supply consumption is typical for the western United States, but it may not be applicable to other parts of the world or areas that give preference to groundwater pumping over surface-water deliveries. In a potential update to MF-OWHM2, a user-specified order for sources of supply would be possible.

Soil moisture is assumed to reach steady-state flow within a single time-step; unsaturated flow can be simulated with the unsaturated-zone flow package (UZF). Soil-moisture studies completed as part of the initial release of the farm process (FMP1; Schmid, 2004) indicated that this assumption is valid for time steps equal to or greater than 1 day (24 hours). It is recommended that time-step period be greater than 1 day for the FMP to simulate land use and the related soil moisture conditions. A separate software module which could be subsequently included in MF-OWHM2 is an approximation of the Richards equation for a more accurate simulation of soil-moisture dynamics.

This release of MF-OWHM2 does not simulate small scale on-farm storage (small ponds/storage tanks). This practice typically is used on farms to reuse water multiple times or capture runoff from precipitation events.

As of 2020, MF-OWHM2 does not include a direct, internal simulation of snowmelt, permafrost simulation, mountainous-watershed rainfall-runoff process, or atmospheric moisture-content effects. Snowmelt is treated as an inflow boundary condition that is calculated outside of the model. If detailed rainfall–runoff information is required, then a companion watershed model can be developed. Such a companion model may also serve to generate the inflow of the stream network along the model domain boundary. Atmospheric moisture may be accounted for indirectly by optionally specifying a pan evaporation rate, reference evapotranspiration, and precipitation. These features are not included in order to ensure that simulation runtime remains short enough to enable automated methods of calibrating model parameters to field observations, which typically require a large number of model runs. The MF-OWHM2 development approach is to include as much detail in hydrological processes as possible, while simulation runtimes remain reasonable enough to allow for robust model validation, verification, and predictability.

Some additional limitations and abilities have been summarized in several model comparisons of selected integrated hydrologic model (IHM) codes, such as MODFLOW-FMP and the Integrated Water Flow Model (IWFM; Dogrul and others, 2011; Schmid and others, 2011; Dogrul, 2009a, b) and MF-FMP, IWFM, and Hydrogeosphere (Therrien and others, 2010; Harter and Morel-Seytoux, 2013). Other fine-scaled comparisons of the MF-FMP simulated groundwater uptake as ET to empirical methods have also been done (Liu and Luo, 2012).

Water quality is only included with the LMT link to MT3DMS and MT3DMS-USGS. This link supports only groundwater and surface-water flow through the streamflow routing package (SFR). At this point, there is no link to soil-moisture transport, transport that results from evapotranspiration and its effect on actual evapotranspiration, or from applied water that has a different chemistry than the groundwater in the area where it is applied.

Although several versions of MODFLOW have been integrated in MF-OWHM2, there are some compatibility issues that remain between packages. The Sea Water Intrusion package (SWI) does not support groundwater wells represented by the MNW1 or MNW2 packages. Combining the NWT formulation and LGR can present difficulties for convergence of flows across parent and child model boundaries given the way that conductance in rows and columns is specified in the upstream weighting package. The HFB2 package flow-routing feature in MF-OWHM2 is incompatible with other post-processing programs, such as MODPATH and ZoneBudget. Note that compatibility issues with other packages have been fixed in MF-OWHM2. For instance, the NWT can operate properly with the subsidence packages SUB and SUB-WT. Because of potential linkages between packages and processes, certain program structures and programming features and protocols need to be followed if developers want to add other features to MF-OWHM2. For example, the addition of a landscape-based feature to MF-OWHM2 requires careful implementation to be connected to the subsidence-linkage option. A description of ways modifications can be made to MF-OWHM2 for specific applications is beyond the scope of this document.

Summary and Conclusions

The One-Water Hydrologic Model (MF-OWHM2) is an integrated hydrologic model (IHM). It is a nearly complete version of the MODFLOW family of hydrologic simulators. It includes comprehensive functionality for the analysis of a broad range of conjunctive water-use issues. MF-OWHM2 simulates and can aid analyses to improve management of multiple components of human and natural water movement and use in a physically based supply and demand framework. MF-OWHM2 is based on the farm process of MODFLOW-2005 (MF-FMP3) combined with local grid refinement to allow use of the Farm process (FMP) and streamflow routing (SFR) in embedded grids. The ability to use embedded models allows for the use and linkage of models developed by local water agencies in the framework of regional models that simulate the entire watershed.

MF-OWHM2 combines several existing capabilities, including the surface-water routing process (SWR) and riparian evapotranspiration (RIP-ET); a broad range of solvers, such as Newton-Raphson (NWT) and nonlinear preconditioned conjugate gradient (PCGN); and simulates reservoir operations through linkage to SWO (Ferguson and others, 2016). MF-OWHM2 can simulate deformation-, flow-, and head-dependent flows, and also includes an upgrade for the salinity demand for additional irrigation. Deformation-dependent flows are simulated through the optional linkage to simulate land subsidence by a vertically deforming mesh. Flow-dependent flows include linkages between the updated SWR with SFR and the FMP, as well as connection to embedded models for the SFR and FMP through the LGR and DRT (drain return flows). Head-dependent flow processes include a modified Hydrologic Flow Barrier Package that allows optional transient HFB capabilities and flow between any two layers adjacent along a depositional or erosional boundary or displaced along a fault. The expansion of the subsidence package allows easier parameterization and separation of the elastic and inelastic deformation, which enables better representation and estimation of land subsidence. Additional features include an ExpressionParser in the multiplier package, as well as a more systematic time-series input for SFR, GHB, SWR, WEL, and MNW packages. The salinity demand option is embedded in the FMP and allows for flow-dependent application of additional water to prevent salt accumulation. Finally, support for SWO allows for flow-dependent linkage between allocation of water from a reservoir-based project and the conveyance and demands at multiple levels on and off the model grid. These added features facilitate a more physically based parameterization and the fundamental input structures needed to build self-updating models for operational and forecasting analysis.

MF-OWHM2 represents a complete hydrologic model that fully links the use and movement of groundwater, surface water, and imported water for consumption by irrigated agriculture, as well as water used in urban areas and by natural vegetation. Supply and demand components of water use are analyzed under demand-driven and supply-constrained relationship. From large- to small-scale settings, MF-OWHM2 has capabilities to simulate and analyze historical, present day, and future conjunctive-use conditions. MF-OWHM2 is especially useful for the analysis of agricultural water use for which few data are available for pumpage, land use, or agricultural practices. MF-OWHM2 characteristically keeps water in the simulation and reduces the water not accounted for by the simulation. This facilitates a more comprehensive simulation and analysis of the conjunctive use and movement of precipitation, surface water, and groundwater, as well as water reuse. This allows a more complete representation of the hydrosphere and its potential connections to humanity, habitat, climate, agriculture, land use, and other related socioeconomic or physical elements that are affected by the distribution of water.

In addition to groundwater, surface-water, and landscape budgets, MF-OWHM2 provides additional options for observations of land subsidence, hydraulic properties, and evapotranspiration (ET). Detailed landscape budgets combined with output of estimates of actual evapotranspiration facilitate a linkage to remotely sensed observations as input or as additional observations for parameter estimation or water-use analysis. The features of the FMP have been extended to allow for temporally variable WBSs (farms) that can be linked to land-use models, defined surface-water and groundwater allotments to facilitate sustainability analysis, linked simulation-optimization that maximizes crop yield (for example, Fowler and others, 2014, 2016), and support for linking surface-water operations with FMP and SFR to analyze the complete reservoir-dependent project schemes for surface-water allotments.

The example model demonstrated the application of MF-OWHM2 in conjunction with land subsidence by a vertically deforming mesh, delayed recharge through an unsaturated zone, rejected infiltration in a riparian area, changes in demand due to deficiency in supply, changes in multi-aquifer pumping due to constraints imposed through the FMP and the MNW2 package, the simulation of unsaturated conditions by a combination of the NWT and UZF Packages, and changes in surface water such as runoff and streamflow. The example model was also used to show how the salinity demand can be represented in the FMP to simulate the potential reduction of salt accumulation in irrigated lands.

The effects of feedback to the land surface and aquifers from salinity demand in MF-OWHM2 were found to be relatively important with respect to simulations not using these linkages and additional demands. The inclusion of salinity demand in the simulation resulted in an even larger difference in flow terms relative to simulations that did not consider this additional demand and also resulted in additional secondary effects, such as increased pumping, storage depletion, streamflow infiltration, and land subsidence. Such linkages can be critical to a complete analysis of selected supply and demand components for conjunctive water use compared to simulations that do not consider these feedbacks, including simulations of the sustained agricultural and urban demands driving secondary effects, such as land subsidence, that can become the limiting factors for sustainability and further resource development. Therefore, these linkages are well suited for evaluating conjunctive water use where the vertical displacements or differential displacements can affect the availability of sources of water, the proportions of multiple sources of water, and the flow to and from aquifers.

References Cited

Allen, R.G., Pereira, L.S., Raes, D., and Smith, M., 1998, Crop evapotranspiration—Guidelines for computing crop water requirements: Rome, Italy, Food and Agriculture Organization of the United Nations, Irrigation and Drainage Paper 56, 300 p., http://www.fao.org/docrep/X0490E/X0490E00.htm.

Allen, R.G., Clemmens, A.J., Burt, C.M., Solomon, K., and O'Halloran, T., 2005, Prediction accuracy for projectwide evapotranspiration using crop coefficients and reference evapotranspiration: American Society of Civil Engineers, Journal of Irrigation and Drainage Engineering, v. 131, no. 1, p. 24–36, https://ascelibrary.org/doi/abs/10.1061/%28ASCE%290733-9437%282005%29131%3A1%2824%29.

Ayers, R.S., and Wescott, D.W., 1985, Water quality for agriculture: Rome, Italy, Food and Agriculture Organization of the United Nations Irrigation and Drainage Paper 29, rev. 1, variously paginated, http://www.fao.org/docrep/003/t0234e/t0234e00.htm.

Bakker, M., Schaars, F., Hughes, J.D., Langevin, C.D., and Dausman, A.M., 2013, Documentation of the Seawater Intrusion (SW12) package for MODFLOW: U.S. Geological Survey Techniques and Methods 6–A46, 47 p., https://pubs.usgs.gov/tm/6a46/.

Bedekar, V., Morway, E.D., Langevin, C.D., and Tonkin, M., 2016, MT3D-USGS version 1—A U.S. Geological Survey release of MT3DMS updated with new and expanded transport capabilities for use with MODFLOW: U.S. Geological Survey Techniques and Methods 6–A53, 69 p., https://doi.org/10.3133/tm6A53.

Bicknell, B.R., Imhoff, J.C., Kittle, J.L., Jr., Jobes, T.H., and Donigian, A.S., Jr., 2001, Hydrologic simulation program—FORTRAN HSPF—User's manual version 12: U.S. Environmental Protection Agency, 845 p., https://searchworks.stanford.edu/view/5650433.

Boyce, S.E., 2015, Model reduction via proper orthogonal decomposition of transient confined and unconfined groundwater flow: Los Angeles, Calif., University of California Los Angeles, PhD Thesis, 64 p., https://escholarship.org/uc/item/0k31q9qr.

Boyce, S.E., and Yeh, W.W.-G., 2014, Parameter-independent model reduction of transient groundwater flow models—Application to inverse problems: Advances in Water Resources, v. 69, p. 168–180, https://doi.org/10.1016/j.advwatres.2014.04.009.

Boyce, S.E., Nishikawa, T., and Yeh, W.W.-G., 2015, Reduced order modeling of the Newton formulation of MODFLOW to solve unconfined groundwater flow: Advances in Water Resources, v. 83, p. 250–262, https://doi.org/10.1016/j.advwatres.2015.06.005.

Brush, C.F., Belitz, K., Phillips, S.P., Burow, K.R., and Knifong, D.L., 2006, MODGRASS—Update of a groundwater flow model for the central part of the western San Joaquin Valley, California: U.S. Geological Survey Scientific Investigations Report 2005–5290, 81 p., https://doi.org/10.3133/sir20055290.

Cahn, M., and Bali, K., 2015, Managing salts by leaching: University of California Agriculture and Natural Resources Publication no. 8550, 8 p., http://anrcatalog.ucanr.edu.

Darcy, H., 1856, Exposition et application des principes a suivre et des formulesa employer dans les questions de distribution d'eau, *in* Dalmont, V., ed., Les Fontaines Publiques de la Ville de Dijon: Paris, 647 p.

De Filippis, G., Borsi, I., Foglia, L., Cannata, M., Velasco, V., Vasquez-Suñe, E., Ghetta, M., and Rossetto, R., 2017, Software tools for sustainable water resources management: the GIS-integrated FREEWAT platform: Rendiconti Online Società Geologica Italiana, v. 42, p. 59–61, https://doi.org/10.3301/ROL.2017.14.

Dogrul, E.C., 2009a, Integrated Water Flow Model (IWFM v3.1)—Theoretical documentation: Sacramento, Calif., Integrated Hydrological Models Development Unit, Modeling Support Branch, Bay-Delta Office, California Department of Water Resources, variously paginated.

Dogrul, E.C., 2009b, Integrated Water Flow Model (IWFM v3.1)—User's manual: Sacramento, Calif., Integrated Hydrological Models Development Unit, Modeling Support Branch, Bay-Delta Office, California Department of Water Resources, variously paginated.

Dogrul, E.C., Schmid, W., Hanson, R.T., Kadir, T.N., and Chung, F.I., 2011, Integrated water flow model and modflow-farm process—A comparison of theory, approaches, and features of two integrated hydrologic models: California Department of Water Resources Technical Information Record, TIR–1, 80 p.

Donigian, A.S., Jr., Bicknell, B.R., and Imhoff, J.C., 1995, Hydrological simulation program—FORTRAN, *in* Singh, V.P., ed., Computer models of watershed hydrology: Highlands Ranch, Colo., Water Resources Publications, p. 395–442.

Ewers, R.O., 2006, Karst aquifers and the role of assumptions and authority in science: The Geological Society of America Special Papers, v. 404, no. 19, p. 235–242, https://doi.org/10.1130/2006.2404(19).

Faunt, C.C., ed., 2009, Groundwater availability of the Central Valley aquifer, California: U.S. Geological Survey Professional Paper 1766, 225 p., https://doi.org/10.3133/pp1766.

Faunt, C.C., Hanson, R.T., Belitz, K., and Rogers, L., 2009, California's Central Valley groundwater study—A powerful new tool to assess water resources in California's Central Valley: U.S. Geological Survey Fact Sheet 2009–3057, 4 p., http://pubs.usgs.gov/fs/2009/3057/.

Faunt, C.C., Stamos, C.L., Flint, L.E., Wright, M.T., Burgess, M.K., Sneed, M., Brandt, J., Coes, A.L., and Martin, P., 2015, Hydrogeology, hydrologic effects of development, and simulation of groundwater flow in the Borrego Valley, San Diego County, California: U.S. Geological Survey Scientific Investigations Report 2015–5150, 135 p., https://doi.org/10.3133/sir20155150.

Ferguson, I.A., and Llewellyn, D., 2015, Simulation of Rio Grande project operations in the Rincon and Mesilla Basins—Summary of model configuration and results: Appendix C of Draft Environmental Impact Statement: Bureau of Reclamation Technical Memorandum no. 86–68210–2015–05, 171 p.

Ferguson, I.A., Llewellyn, D., Hanson, R.T., and Boyce S.E., 2016, User guide to the surface water operations process—An integrated approach to simulating large-scale surface water management in MODFLOW-based hydrologic models: Denver, Colo., Bureau of Reclamation Technical Memorandum no. 86-68210–2016-02, 96 p.

Flint, L.E., and Flint, A.L., 2014, California basin characterization model—A dataset of historical and future hydrologic response to climate change: U.S. Geological Survey Data Release, https://doi.org/10.5066/F76T0JPB.

Flint, L.E., Flint, A.L., Thorne, J.H., and Boynton, R., 2013, Fine-scale hydrological modeling for climate change applications; using watershed calibrations to assess model performance for landscape projections; Ecological Processes, v. 2, no. 25, 21 p., https://ecologicalprocesses.springeropen.com/articles/10.1186/2192-1709-2-25.

Food and Agriculture Organization, 2007, Chapter 8—ET_c under soil water stress conditions: Food and Agriculture web page, accessed September 9, 2008, http://www.fao.org/docrep/x0490e/x0490e0e.htm.

Ford, D.C., and Williams, P.W., 1989, Karst geomorphology and hydrology: London, England, Unwin Hyman, 601 p.

Fowler, K.R., Jenkins, E.W., Ostrove, C., Chrispell, J.C., Farthing, M.W., and Parno, M., 2014, A decision making framework with MODFLOW-FMP2 via optimization—Determining trade-offs in crop selection: U.S. Army Corps of Engineers, 12 p., http://digitalcommons.unl.edu/usarmyceomaha/151/.

Fowler, K.R., Jenkins, E.W., Parno, M., Chrispell, J.C., Colón, A.I., and Hanson, R.T., 2016, Development and use of mathematical models and software frameworks for integrated analysis of agricultural systems and associated water use impacts: AIMS Agriculture and Food, v. 1, no. 2, p. 208–226, https://doi.org/10.3934/agrfood.2016.2.208.

Freeze, R.A., 1975, A stochastic-conceptual analysis of one-dimensional groundwater flow in nonuniform homogeneous media: Water Resources Research, v. 11, no. 5, p. 725–741.

Freeze, R.A., and Cherry, J.A., 1979, Groundwater: Englewood Cliffs, N.J., Prentice-Hall, 604 p.

Galloway, D.L., Jones, D.R., and Ingebritsen, S.E., 1999, Land subsidence in the United States: U.S. Geological Survey Circular 1182, 175 p., https://doi.org/10.3133/cir1182.

Hanson, R.T., and Schmid, W., 2013, Economic resilience through "One-Water" management: U.S. Geological Survey Open-File Report 2013–1175, 2 p., https://pubs.usgs.gov/of/2013/1175/.

Hanson, R.T., Schmid, W., Faunt, C.C., and Lockwood, B., 2010, Simulation and analysis of conjunctive use with MODFLOW's Farm Process: Ground Water, v. 48, no. 5, p. 674–689, https://doi.org/10.1111/j.1745-6584.2010.00730.x.

Hanson, R.T., Flint, L.E., Flint, A.L., Dettinger, M.D., Faunt, C.C., Cayan, D., and Schmid, W., 2012, A method for physically based model analysis of conjunctive use in response to potential climate changes: Water Resources Research, v. 48, 23 p., https://doi.org/10.1029/2011WR010774.

Hanson, R.T., Kauffman, L.K., Hill, M.C., Dickinson, J.E., and Mehl, S.W., 2013, Advective transport observations with MODPATH-OBS—documentation of the MODPATH observation process: U.S. Geological Survey Techniques and Methods 6-A42, 96 p., https://doi.org/10.3133/tm6A42.

Hanson, R.T., Schmid, W., Faunt, C.C., Lear, J., and Lockwood, B., 2014a, Integrated hydrologic model of Pajaro Valley, Santa Cruz and Monterey Counties, California: U.S. Geological Survey Scientific Investigations Report 2014–5111, 166 p., https://doi.org/10.3133/sir20145111.

Hanson, R.T., Lockwood, B., and Schmid, W., 2014b, Analysis of projected water availability with current basin management plan, Pajaro Valley, California: Journal of Hydrology, v. 519, p. 131–147, https://ca.water.usgs.gov/pubs/2014/HansonLockwoodSchmid2014.pdf.

Hanson, R.T., Flint, L.E., Faunt, C.C., Gibbs, D.R., and Schmid, W., 2014c, Hydrologic models and analysis of water availability in Cuyama Valley, California: U.S. Geological Survey Scientific Investigations Report 2014–5150, 150 p., https://doi.org/10.3133/sir20145150.

Hanson, R.T., Boyce, S.E., Schmid, W., Hughes, J.D., Mehl, S.M., Leake, S.A., Maddock, T., III, and Niswonger, R.G., 2014d, One-water hydrologic flow model (MODFLOW-OWHM): U.S. Geological Survey Techniques and Methods 6–A51, 120 p., https://doi.org/10.3133/tm6A51.

Hanson, R.T., Ritchie, A.B., Boyce, S.E., Galanter, A.E., Ferguson, I.A., Flint, L.E., Flint, A., and Henson, W.R., 2019, Rio Grande transboundary integrated hydrologic model and water-availability analysis, New Mexico and Texas, United States, and northern Chihuahua, Mexico: U.S Geological Survey Scientific Investigations Report 2019–5120, 188 p., https://doi.org/10.3133/sir20195120.

Harbaugh, A.W., 1990, A computer program for calculating subregional water budgets using results from the U.S. Geological Survey modular three-dimensional ground-water flow model: U.S. Geological Survey Open-File Report 90–392, 46 p., https://doi.org/10.3133/ofr90392.

Harbaugh, A.W., 2005, MODFLOW-2005—The U.S. Geological Survey modular ground-water model—The ground-water flow process: U.S. Geological Survey Techniques and Methods 6–A16, variously paginated, https://pubs.usgs.gov/tm/2005/tm6A16/.

Harter, T., and Morel-Seytoux, H., 2013, Peer review of the IWFM, MODFLOW and HGS model codes—Potential for water management applications in California's Central Valley and other irrigated groundwater basins: Sacramento, Calif., Final Report, California Water and Environmental Modeling Forum, 112 p., http://www.cwemf.org/Pubs/index.htm.

Hemond, H.F., and Fechner, E.J., 2015, Chemical fate and transport in the environment: Oxford, UK, Elsevier, 486 p., https://doi.org/10.1016/C2011-0-09677-1.

Hoffmann, J., Leake, S.A., Galloway, D.L., and Wilson, A.M., 2003, MODFLOW-2000 ground-water model—User guide to the subsidence and aquifer-system compaction (SUB) package: U.S. Geological Survey Open-File Report 2003–233, 46 p., https://pubs.usgs.gov/of/2003/ofr03-233/pdf/ofr03233.pdf.

Hughes, J.D., Langevin, C.D., Chartier, K.L., and White, J.T., 2012, Documentation of the surface-water routing (SWR1) process for modeling surface-water flow with the U.S. Geological Survey modular ground-water model (MODFLOW–2005): U.S. Geological Survey Techniques and Methods 6–A40, 113 p., https://pubs.usgs.gov/tm/6a40/pdf/Hughes_TM6-A40.pdf.

Ibaraki, M., 2005, χMD user's guide—An efficient sparse matrix solver library, version 1.30: Columbus, Ohio, Ohio State University School of Earth Sciences.

Keating, E., and Zyvoloski, G., 2009, A stable and efficient numerical algorithm for unconfined aquifer analysis: Ground water, v. 47, no. 4, p. 569–579.

Konikow, L.F., Hornberger, G.Z., Halford, K.J., and Hanson, K.J., 2009, Revised multi-node well (MNW2) package for MODFLOW ground-water flow model: U.S. Geological Survey Techniques and Methods 6–A30, 67 p., https://pubs.usgs.gov/tm/tm6a30/.

Labadie, J.W., and Larson, R., 2007, MODSIM 8.1—River basin management decision support system, user manual and documentation: Fort Collins, Colo., Colorado State University, 123 p.

Lai, Y.G., 2008, SRH-2D version 2: Theory and user's manual: Denver, Colo., Bureau of Reclamation, Technical Service Center, 97 p., https://www.usbr.gov/tsc/techreferences/computer%20software/models/srh2d/downloads/Manual-SRH2D-v2.0-Nov2008.pdf.

Lai, Y.G., 2010, Two-dimensional depth-averaged flow modeling with an unstructured hybrid mesh: Journal of Hydraulic Engineering, American Society of Civil Engineers, v. 136, no. 1, p. 12–23, https://ascelibrary.org/doi/abs/10.1061/%28ASCE%29HY.1943-7900.0000134?journalCode=jhend8.

Langevin, C.D., Hughes, J.D., Banta, E.R., Niswonger, R.G., Panday, S., and Provost, A.M., 2017, Documentation for the MODFLOW 6 groundwater flow model: U.S. Geological Survey Techniques and Methods 6–A55, 197 p., https://doi.org/10.3133/tm6A55.

Leake, S.A., and Galloway, D.L., 2007, MODFLOW ground-water model—User guide to the subsidence and aquifer-system compaction package (SUB-WT) for water-table aquifers: U.S. Geological Survey Techniques and Methods 6–A23, 42 p., https://pubs.usgs.gov/tm/2007/06A23/.

Leavesley, G.H., Lichty, R.W., Troutman, B.M., and Saindon, L.G., 1983, Precipitation-runoff modeling system—User's manual: U.S. Geological Survey Water Resources Investigations Report 83–4238, 207 p., https://pubs.er.usgs.gov/publication/wri834238.

Liu, T., and Luo, Y., 2012, An empirical approach simulating evapotranspiration from groundwater under different soil water conditions: Journal of Environmental Earth Sciences, v. 67, p. 1345–1355, https://doi.org/10.1007/s12665-012-1577-3.

Loáiciga, H.A., Yeh, W.W.-G., and Ortega-Guerrero, M.A., 2006, Probability density functions in the analysis of hydraulic conductivity data: Journal of Hydrologic Engineering, v. 11, no. 5, p. 442–450.

Maddock, T., III, Baird, K.J., Hanson, R.T., Schmid, W., and Ajami, H., 2012, RIP-ET—A riparian evapotranspiration package for MODFLOW-2005: U.S. Geological Survey Techniques and Methods 6–A39, 76 p., http://pubs.usgs.gov/tm/tm6a39/.

Markstrom, S.L., Niswonger, R.G., Regan, R.S., Prudic, D.E., and Barlow, P.M., 2008, GSFLOW-coupled ground-water and surface-water FLOW model based on the integration of the precipitation-runoff modeling system (PRMS) and the modular ground-water flow model (MODFLOW-2005): U.S. Geological Survey Techniques and Methods 6–D1, 240 p., https://pubs.er.usgs.gov/publication/tm6D1.

Markstrom, S.L., Regan, R.S., Hay, L.E., Viger, R.J., Webb, R.M.T., Payn, R.A., and LaFontaine, J.H., 2015, PRMS-IV, the precipitation-runoff modeling system, version 4: U.S. Geological Survey Techniques and Methods 6–B7, 158 p., https://doi.org/10.3133/tm6B7.

McDonald, M.G., and Harbaugh, A.W., 1988, Chapter A1—A modular three-dimensional finite-difference ground-water flow model: Reston, Va., U.S. Geological Survey Open-File Report 83–875, v. 6, 588 p., https://pubs.usgs.gov/twri/twri6a1/pdf/TWRI_6-A1.pdf.

Mehl, S., and Hill, M.C., 2005, MODFLOW-2005—The U.S. Geological Survey modular ground-water model—Documentation of shared node Local Grid Refinement (LGR) and the Boundary Flow and Head (BFH) package: U.S. Geological Survey Techniques and Methods 6–A12, 68 p., https://doi.org/10.3133/tm6A12.

Mehl, S.W., and Hill, M.C., 2013, MODFLOW-LGR—Documentation of ghost node Local Grid Refinement (LGR2) for multiple areas and the Boundary Flow and Head (BFH2) package: U.S. Geological Survey Techniques and Methods 6–A44, 43 p., https://pubs.usgs.gov/tm/6a44/.

Morway, E.D., and Gates, T.K., 2012, Regional assessment of soil water salinity across an intensively irrigated river valley: Journal of Irrigation and Drainage Engineering, v. 138, no. 5, p. 393–405, https://doi.org/10.1061/(ASCE)IR.1943-4774.0000411.

Morway, E.D., Niswonger, R.G., and Triana, E., 2016, Toward improved simulation of river operations through integration with a hydrologic model: Environmental Modelling and Software, v. 82, p. 255–274, https://doi.org/10.1016/j.envsoft.2016.04.018.

Mueller, R., Yang, Z., Han, W., and Di, L., 2011, CropScape—A new web based visualization portal for the dissemination of NASS geospatial cropland products: U.S. Department of Agriculture, National Agricultural Statistics Service, 33 p., http://www.nass.usda.gov/Education_and_Outreach/Reports,_Presentations_and_Conferences/Presentations/Mueller_WSS11_CropScape.pdf.

Niswonger, R.G., and Prudic, D.E., 2005, Documentation of the streamflow-routing (SFR2) package to include unsaturated flow beneath streams—A modification to SFR1: U.S. Geological Survey Techniques and Methods 6–A13, 50 p., https://pubs.usgs.gov/tm/2006/tm6A13/pdf/tm6a13.pdf.

Niswonger, R.G., Prudic, D.E., and Regan, R.S., 2006, Documentation of the unsaturated-zone flow (UZF1) package for modeling unsaturated flow between the land surface and the water table with MODFLOW-2005: U.S. Geological Survey Techniques and Methods 6–A19, 62 p., https://pubs.usgs.gov/tm/2006/tm6a19/pdf/tm6a19.pdf.

Niswonger, R.G., Panday, S., and Ibaraki, M., 2011, MODFLOW-NWT—A Newton formulation for MODFLOW-2005: U.S. Geological Survey Techniques and Methods 6–A37, 44 p., https://pubs.usgs.gov/tm/tm6a37/pdf/tm6a37.pdf.

Painter, S., and Başağaoğlu, H., 2007, Robust representation of dry cells in MODFLOW: Southwest Research Institute Final Technical Report, SwRI Project 20–13003, 20 p., https://old.edwardsaquifer.org/documents/2007_PainterBasagaoglu_MODFLOW-DryCells.pdf.

Painter, S., Başağaoğlu, H., and Liu, A., 2008, Robust representation of dry cells in single-layer MODFLOW models: Groundwater, v. 46, no. 6, p. 873–881, https://doi.org/10.1111/j.1745-6584.2008.00483.x.

Panday, S., Langevin C.D., Niswonger, R.G., Ibaraki, M., and Hughes, J.D., 2013, MODFLOW–USG version 1—An unstructured grid version of MODFLOW for simulating groundwater flow and tightly coupled processes using a control volume finite-difference formulation: U.S. Geological Survey Techniques and Methods 6–A45, 66 p., https://pubs.usgs.gov/tm/06/a45/.

Pollock, D.W., 2016, User guide for MODPATH version 7—A particle-tracking model for MODFLOW: U.S. Geological Survey Open-File Report 2016–1086, 35 p., https://doi.org/10.3133/ofr20161086.

Prudic, D.E., Konikow, L.F., and Banta, E.R., 2004, A new streamflow-routing (SFR1) package to simulate stream-aquifer interaction with MODFLOW-2000: U.S. Geological Survey Open-File Report 2004–1042, 95 p., https://water.usgs.gov/nrp/gwsoftware/modflow2000/ofr2004-1042.pdf.

Rhoades, J.D., 1972, Quality of water for irrigation: Soil Science, v. 113, no. 4, p. 277–284.

Rhoades, J.D., 1977, Potential for using saline agricultural drainage waters for irrigation: Proceedings of the American Society of Civil Engineers Irrigation and Drainage Division Specialty Conference on Water Management for Irrigation and Drainage, Reno, Nev., July 20–22, 1977, p. 85–116.

Rhoades, J.D., 2012, Evidence of the potential use of saline water for irrigation: CIHEAM— Options Mediterraneennes, no. 133, 14 p., ftp://ftp.ecn.purdue.edu/vmerwade/class/GDT/uploaded_documents/India/use%20of%20saline%20water%20for%20irrigation.pdf.

Rhoades, J.D., and Merril, S.D., 1976, Assessing the suitability of water for irrigation—Theoretical and empirical approaches, *in* Prognosis of salinity and alkalinity: Rome, Italy, Food and Agriculture Organization Soils Bulletin, no. 31, p. 69–109, http://www.fao.org/3/a-ar112e.pdf.

Rossetto, R., Borsi, I., and Foglia, L., 2015, FREEWAT—FREE and open source software tools for WATer resource management: Rendiconti Online Della Società Geologica Italiana, v. 35, p. 252–255, https://doi.org/10.3301/ROL.2015.113.

Saller, S.P., Ronayne, M.J., and Long, A.J., 2013, Comparison of a karst groundwater model with and without discrete conduit flow: Hydrogeology Journal, v. 21, no. 7, p. 1555–1566, https://doi.org/10.1007/s10040-013-1036-6.

Schmid, W., 2004, A farm package for MODFLOW-2000—Simulation of irrigation demand and conjunctively managed surface-water and ground-water supply: Tucson, Ariz., University of Arizona, Department of Hydrology and Water Resources, PhD Dissertation, 278 p.

Schmid, W., and Hanson, R.T., 2009, The farm process version 2 (FMP2) for MODFLOW-2005—Modifications and upgrades to FMP1: U.S. Geological Survey Techniques and Methods 6–A32, 102 p., https://pubs.er.usgs.gov/publication/tm6A32.

Schmid, W., Hanson, R.T., Maddock, T., III, and Leake, S.A., 2006, User guide for the farm process (FMP1) for the U.S. Geological Survey's modular three-dimensional finite-difference ground-water flow model, MODFLOW-2000: U.S. Geological Survey Techniques and Methods 6–A17, 127 p., https://water.usgs.gov/nrp/gwsoftware/mf2005_fmp/tm6A17.pdf.

Schmid, W., Hanson, R.T., Faunt, C.C., and Phillips, S.P., 2008, Hindcast of water availability in regional aquifer systems using MODFLOW's farm process: Proceedings of Hydropredict 2008, Prague, Czech Republic, September 15–18, 2008, p. 311–314, https://pubs.er.usgs.gov/publication/70147052.

Schmid, W., King, J.P., and Maddock, T.M., III, 2009, Conjunctive surface-water/ground-water model in the southern Rincon Valley using MODFLOW-2005 with the farm process: Las Cruces, N. Mex., New Mexico Water Resources Research Institute Completion Report no. 350, 53 p.

Schmid, W., Dogrul, E.C., Hanson, R.T., Kadir, T.N., and Chung, F.I., 2011, Comparison of simulations of land-use specific water demand and irrigation water supply by MF-FMP and IWFM: California Department of Water Resources Technical Information Record TIR-2, 80 p.

Schmid, W., Hanson, R.T., Leake, S.A., Hughes, J.D., and Niswonger, R.G., 2014, Feedback of land subsidence on the movement and conjunctive use of water resources: Environmental Modelling and Software, v. 62, p. 253–270.

Shoemaker, W.B., Kuniansky, E.L., Birk, S., Bauer, S., and Swain, E.D., 2008, Documentation of a conduit flow process (CFP) for MODFLOW-2005: U.S. Geological Survey Techniques and Methods 6–A24, 50 p., https://pubs.usgs.gov/tm/tm6a24/pdf/tm6-A24.pdf.

State of California, 2014, Sustainable Groundwater Management Act [and related statutory provisions from SB1168 (Pavley), AB1739 (Dickinson), and SB1319 (Pavley) as chaptered]: State of California, 52 p., http://leginfo.legislature.ca.gov/faces/codes_displayText.xhtml?lawCode=WAT&division=6.&title=&part=2.74.&chapter=2.&article=.

Tanji, K.K., and Kielen, N.C., 2002, Agricultural drainage water management in arid and semi-arid areas: Rome, Italy, Food and Agricultural Organization of the United Nations, Food and Agriculture Organization Irrigation and Draining Paper 61, http://www.fao.org/docrep/005/y4263e/y4263e00.htm.

Therrien, R., McLaren, R.G., and Sudicky, E.A., 2010, HydroGeoSphere—A three-dimensional numerical model describing fully-integrated subsurface and surface flow and solute transport: Ontario, Canada, Groundwater Simulations Group, University of Waterloo, variously paginated, https://www.ggl.ulaval.ca/fileadmin/ggl/documents/rtherrien/hydrogeosphere.pdf.

U.S. Army Corps of Engineers, 2016, HEC-RAS river analysis system user manual, version 5.0: Davis, Calif., U.S. Army Corps of Engineers, Hydrologic Engineering Center, 960 p., http://www.hec.usace.army.mil/software/hec-ras/documentation/HEC-RAS%205.0%20Reference%20Manual.pdf.

U.S. Environmental Protection Agency, 2019, BASINS framework and features, environmental modeling community of practice Web page: https://www.epa.gov/ceam/basins-framework-and-features.

U.S. Geological Survey, 2019, Modeling of watershed systems (MOWS) home Web page: https://wwwbrr.cr.usgs.gov/projects/SW_MoWS/index.html.

Whitaker, S., 1986, Flow in porous media I—A theoretical derivation of Darcy's law: Transport in Porous Media, v. 1, no. 1, p. 3–25, https://doi.org/10.1007/BF01036523.

Willis, R., and Yeh, W.W.-G., 1987, Groundwater systems planning and management: Englewood Cliffs, N.J., Prentice-Hall, 416 p.

Winston, R.B., 2009, ModelMuse—A graphical user interface for MODFLOW-2005 and PHAST: U.S. Geological Survey Techniques and Methods 6–A29, 52 p., https://pubs.usgs.gov/tm/tm6A29/tm6A29.pdf.

Winston, R.B., 2014, Modifications made to ModelMuse to add support for the saturated-unsaturated transport model (SUTRA): U.S. Geological Survey Techniques and Methods 6–A49, 6 p., https://doi.org/10.3133/tm6a49.

Xu, Z., and Hu, B.X., 2017, Development of a discrete-continuum VDFST-CFP numerical model for simulating seawater intrusion to a coastal karst aquifer with a conduit system: Water Resources Research, v. 53, no. 1, p. 688–711, https://doi.org/10.1002/2016WR018758.

Xu, Z., Hu, B.X., Davis, H., and Cao, J., 2015a, Simulating long term nitrate-N contamination processes in the Woodville Karst Plain using CFPv2 with UMT3D: Journal of Hydrology, v. 524, p. 72–88, https://doi.org/10.1016/j.jhydrol.2015.02.024.

Xu, Z., Hu, B.X., Davis, H., and Kish, S., 2015b, Numerical study of groundwater flow cycling controlled by seawater/freshwater interaction in a coastal karst aquifer through conduit network using CFPv2: Journal of Contaminant Hydrology, v. 182, p. 131–145, https://doi.org/10.1016/j.jconhyd.2015.09.003.

Zheng, C., and Wang, P.P., 1999, MT3DMS—A modular three-dimensional multispecies transport model for simulation of advection, dispersion, and chemical reactions of contaminants in groundwater systems—Documentation and user's guide: Tuscaloosa, Ala., Alabama University, 220 p., https://hydro.geo.ua.edu/mt3d/mt3dmanual.pdf.

Appendix 0. Report Syntax Highlighting and Custom Font Styles

Syntax highlighting is used to help distinguish each input type and its function in this report. Figure 0.1 provides a description and examples of each of the font styles used.

Style	Description	Example
Capitalized Times New Roman Regular	MF-OWHM2 Package names.	DIS, GHB, FMP
Times New Roman Italic	Full name of input styles and read utilities.	*U2DREL, Generic_Input, Transient File Reader*
`Consolas Regular`	MF-OWHM2 code variable names; input variable names that require a number; acronyms to input styles; and read utilities referenced input to the NAME file.	`IBOUND`, `TSMULT`, `TFR`, `DATA(BINARY)`
`Consolas Bold`	Block "BEGIN," "END," and its name; non-block keywords that initiate an input; a package's OPTION block keywords; secondary keywords for an input.	**`BEGIN OPTION`**, **`TABFILE`**, **`FREE`**, **`INTERNAL`**
`Consolas Blue Bold`	Primary package input keyword; primary package keywords in a block input.	**`CROP_COEFFICIENT`**, **`ROOT_DEPTH`**
`Consolas Red Bold`	Surrogate input keyword that must be replaced by a keyword within a group of keywords.	**`INPUT_STYLE`**, **`TEMPORAL_KEY`**
Calibri Gray	Explanation and comments in examples and text boxes. Comments are preceded by a "#" symbol (called number, hash, or pound sign).	# A comment # or explanation

Figure 0.1. Explanation and examples of syntax highlighting.

Appendix 1. New Input Formats and Utilities

A set of new input utilities were developed for MF-OWHM2 and are used by most of the new features presented in this report. The new utilities simplify input by the use of keywords and generalize the input structure of input and output (I/O) files. This new structure simplifies the name file (NAM)—the file that specifies the names of all global I/O data files and packages used in the model simulation and controls the parts of the model program that are active—and allows for an easier understanding of the MF-OWHM2 input for someone who did not build the original simulation model. Further, it allows for a standardized input structure that is easier to develop and maintain for the original model developer. These tools are currently only fully supported in the Farm Process version 4 (FMP) and select packages. With each subsequent release of MF-OWHM2, the utilities can be folded more into the traditional MODFLOW packages to simplify their input.

This simplification is done with keywords that indicate the frequency with which the input is loaded (**TEMPORAL_KEY**), the input style that is loaded (**INPUT_STYLE**), where the input is located (*Generic_Input*), and a host of options for applying multiple scale factors (for example, to separate out conversion factors, transformation multipliers, and calibration parameters/multipliers). Each of the input utilities builds upon each other with the simplest being the *Generic_Input. Generic_Input* is used by the *Universal Loader* (ULOAD) to read input, which is then called by the *Transient File Reader* (TFR) for spatially and temporally varying input, which is then called by the *List-Array Input* utility that parses the input style and temporal frequency. If the data are loaded only once and used for the entire simulation, then the *List-Array Input* utility bypasses the *Transient File Reader* and directly uses ULOAD one time. The temporally varying input can be read at any time interval (for example, time step or stress period) defined by the Package that uses the utilities. The rest of this appendix uses the term "stress period" synonymously with "time interval" because the stress period is the fundamental time interval used by MODFLOW to receive and hold constant user-specified inflows and outflows.

This appendix begins by describing the use of comments in packages and describing a new "*Block Style*" input format. The rest of the appendix begins with a top-down flow chart of the *List-Array Input* utility's structure. This is to illustrate the effect each keyword has until the final input is loaded. The next section discusses the seven possible combinations of keywords available with the *List-Array Input* and briefly discusses them. The flow chart provides a road map of the subsequent sections that summarize the major input utilities developed for MF-OWHM2. Each of the utilities makes use of the previous ones in this appendix. The discussion of each utility includes a top-down overview flow chart of the keywords that make up the *List-Array Input*, a bottom up description of the input utilities for the *Generic_Input* and *Generic_Output*, the ULOAD and scale factors (defined by the keyword **SFAC**), and finally the *List-Array Input* Style and *Transient File Reader*. The appendix concludes with a formal description of the *IXJ Style* input and the *Lookup Style* input.

Comments in Package Input

A new basic read utility, *READ_TO_DATA*, was developed to allow comments in input files. It has been applied entirely to the Farm Process (FMP), General Head Boundary (GHB), and Well (WEL) packages and incorporated at select locations in other packages. A commented line contains the symbol, **#**, with only blank space to the left of it. Comments to the right of a line with input values should also be preceded by a "**#**" symbol. This ensures that the comment is not treated as an input value. Empty lines, although not a comment, are automatically skipped. Examples of commented lines are shown in figure 1.1.

Block-Style Input

MODFLOW 2005 (Harbaugh, 2005) input relied on positional integer flags (for example, 2, 4, 6) and floating-point numbers (for example, 2.0, 4.2, 6.E8) to construct a simulation model. The new block-style input was intended to increase user friendliness, ease of use and documentation, and flexibility. A block input nests model-package input properties in one location (both temporally varying and static inputs), providing a simple input structure that helps the user understand what is being supplied to a model. Each block begins with the word **BEGIN** followed by the block's **NAME**, which defines the input in the block, then the block terminates its input feed with the word **END**. To the right of the block name is an optional set of **GLOBAL_KEYWORD**s that alter the behavior of the entire block. In the block is its input, which relies on keywords to define each input type and its style (for example, transient input or static input). Block input allows keywords to be in any order; it automatically skips blank lines; and comments are accepted as long as they are preceded with a **#** symbol. Figure 1.2 presents the general structure of an input block.

The block specific keyword **BLOCK_INCLUDE** mimics the functionality of the C language `#include "file"` and Fortran `INCLUDE File` statements allowing the user to specify a part of a block in a separate file that is inserted at the **BLOCK_INCLUDE** location. This feature is useful if a block needs to be subdivided into individual files or to use the separate file as a calibration template file. Figure 1.3 is a simple example of **BLOCK_INCLUDE** that loads four keywords. At runtime, before any keywords are processed, the block input inserts all **BLOCK_INCLUDE** files, strips from the block any comments and blank lines, and adjusts all remaining lines in the block to be left justified (fig. 1.3*C*).

```
          Column Number                                                Column Number
 1234567890123456789012345678                                  12345678901234
|55  65        # Side comment |  → MF-OWHM2 only sees   "55  65        "
|   55  65     # Side comment |  → MF-OWHM2 only sees   "   55  65     "
|#  55  65     # Side comment |  → MF-OWHM2 skips the line and loads the next.
|  #55  65     # Side comment |  → MF-OWHM2 skips the line and loads the next.
|                             |  → MF-OWHM2 skips empty line and loads the next.
```

Figure 1.1. Example input lines, delineated with a |, that are parsed by MF-OWHM2 to remove comments and blank lines. To the right of each example input line is an explanation of what MF-OWHM sees as input. [Note that a "|" is used to delineate the start and end of each example input line. Comments must be preceded by a # (pound sign). Column number is the number of spaces within the file ranging from 1 to 28.]

A

```
BEGIN NAME     GLOBAL_KEYWORD
            #
            # Comments and blank lines are ignored

            #
            KEYWORD       # COMMENT

            # COMMENT
            KEYWORD       # COMMENT
            # COMMENT
            #
            BLOCK_INCLUDE  Generic_Input_OptKey
            #
END
```

Figure 1.2. Block-Style Input *A*, structure, and *B*, explanation of the items in part *A*.

B

BEGIN	initiates the block input.
NAME	is the block input name that identifies the expected input within the block.
#	the # and anything to the right of the # is ignored.
KEYWORD	is a keyword that is defined for use within the block.
END	must be present and terminates reading of the block input. Text after **END** is ignored, so it is recommended to add the block's name after the termination word for clarity (for example, **END NAME**).
GLOBAL_KEYWORDS	is optional; if present, a set of global keywords that affect the entire block. An example is **BEGIN LINEFEED XY**, where the block name, **LINEFEED**, indicates that the block contains LineFeed input and the global block keyword, **XY**, indicates that row and column are specified by a x-coordinate and y-coordinate instead.
BLOCK_INCLUDE	is optional; if present must be followed by *Generic_Input_OptKey* that points to a file that contains a set of input that is inserted into the block. *Generic_Input_OptKey* cannot use the keyword **INTERNAL**, since it must reference an external file to include. This feature allows for breaking large data input sections into multiple files that are then inserted into the block. This functions identically to C language `#include "file"` and Fortran `INCLUDE File` statements.

Figure 1.2. —Continued

A

```
BEGIN DUMMY_BLOCK
  #
  Keyword1
  #
  BLOCK_INCLUDE ./Keywords2n3.txt
  #
  # An indented Keyword
                Keyword4     # Comment
END
```

B

```
# Contents of Keywords2n3.txt

Keyword2

Keyword3

# note to self -- remember this note about...
```

C

```
BEGIN DUMMY_BLOCK
Keyword1
Keyword2
Keyword3
Keyword4
END
```

Figure 1.3. Block-Style Input example that defines four input keywords and uses the **BLOCK_INCLUDE** keyword to insert two of the four input keywords. *A*, Example block input with the name "DUMMY_BLOCK" that includes two input keywords and inserts the contents from the file "Keywords2n3.txt". *B*, The contents of the file "Keywords2n3.txt". *C*, The final version of the block that is read as input by MF-OWHM2. This version removes all blank lines and comments, then shifts to the left any remaining text. [Keyword1, Keyword2, Keyword3, and Keyword4 are example keywords that would be defined for use in the DUMMY_BLOCK block.]

Overview of the List-Array Input—Top-Down View

The *List-Array Input* is a simple input structure that relies on keywords to define the frequency with which input is loaded and the structure of the input. This section gives a broad top-down overview of the *List-Array Input*. First, a flow chart illustrates the keyword decision-guided tree that loads the input. After the flow chart, a summary of all the potential keyword combinations is discussed. This section is meant to be an overview of the input utilities that are defined in detail in the subsequent appendix sections.

Flow Chart

To visualize a top-down view of the *List-Array Input*, the flow chart in appendix figure 1.4 presents the decision tree behind each keyword. It begins with the Temporal keyword that defines the frequency that the input is loaded. The options for the Temporal keyword are **`STATIC`**, **`TRANSIENT`**, or **`CONSTANT`**. The next decision is the input style in which the data are formatted. The two input styles that are supported are a record-based *List Style* input and a model grid shaped *Array Style* input. The *List-Array Input* does support a third input style called *IXJ Style*—which is only mentioned here for completeness and not included in the flow chart nor in subsequent sections. This input style is an advanced input whose structure varies and serves as a surrogate for *List Style* and *Array Style*. The keyword for *List Style* is **`LIST`**, for *Array Style* is **`ARRAY`**, and *IXJ Style* is **`IXJ`**.

Once the input frequency and format are defined, then the actual data location is specified with a *Generic_Input* and loaded with the *Universal Loader* (`ULOAD`). The flow chart uses the word "stress period", corresponding to a typical frequency, but the input load frequency can be different depending how the calling package defines it.

Potential Input Combinations

The *List-Array Input* utility uses keywords as presented in the flow chart (fig. 1.5). The following are the seven keyword combinations and their basic descriptions. The detailed descriptions are provided in the remainder of this appendix.

If the *Transient File Reader* (`TFR`) is used, then it has a set of directive keywords that are specified on each row of its file to indicate each time interval's input (typically one keyword for each stress period, unless noted by the specific MF-OWHM2 input). Note that each uncommented, non-**`SFAC`** row in the *Transient File Reader* is loaded for each time interval. If **`SFAC`** is found (see "**`SFAC`**—Scale Factor Keyword" for more details), then it is applied to the input from the file specified after it. Figure 1.6 presents the supported directive keywords that can be specified within a `TFR`.

Input is loaded from any uncommented line that contains one of the keywords defined in 1–10 in figure 1.6. The input frequency depends on the model feature, but typically is in line with the MODFLOW stress period (for example, the fifth uncommented row containing the keyword in 1–10 serves as input to stress period five). For example, several of the input options in the Farm Process, such as precipitation arrays, can be read by stress period or by time step (the input frequency. The keyword **`SFAC`** may appear multiple times, one per uncommented line, and is only applied to the subsequent input specified by items 1through 10 (that is, it modifies the stress-period input only for that stress period). Figure 1.7 is a simple example `TFR` that loads nine stress periods of input and includes an **`SFAC`** that is applied in the first stress period.

This concludes the top-down overview to the *List-Array Input*; what follows begins the bottom-up development of the utilities that make it up. The abbreviation `LAI` refers to the *List-Array Input* utility input options, and the letters S, T, A, L stand for **`STATIC`**, **`TRANSIENT`**, **`ARRAY`**, and **`LIST`**, respectively. These four options for specification of the data input used for the Farm Process (FMP) block input are explained in detail in the "Farm Process Input Updates" (appendix 6).

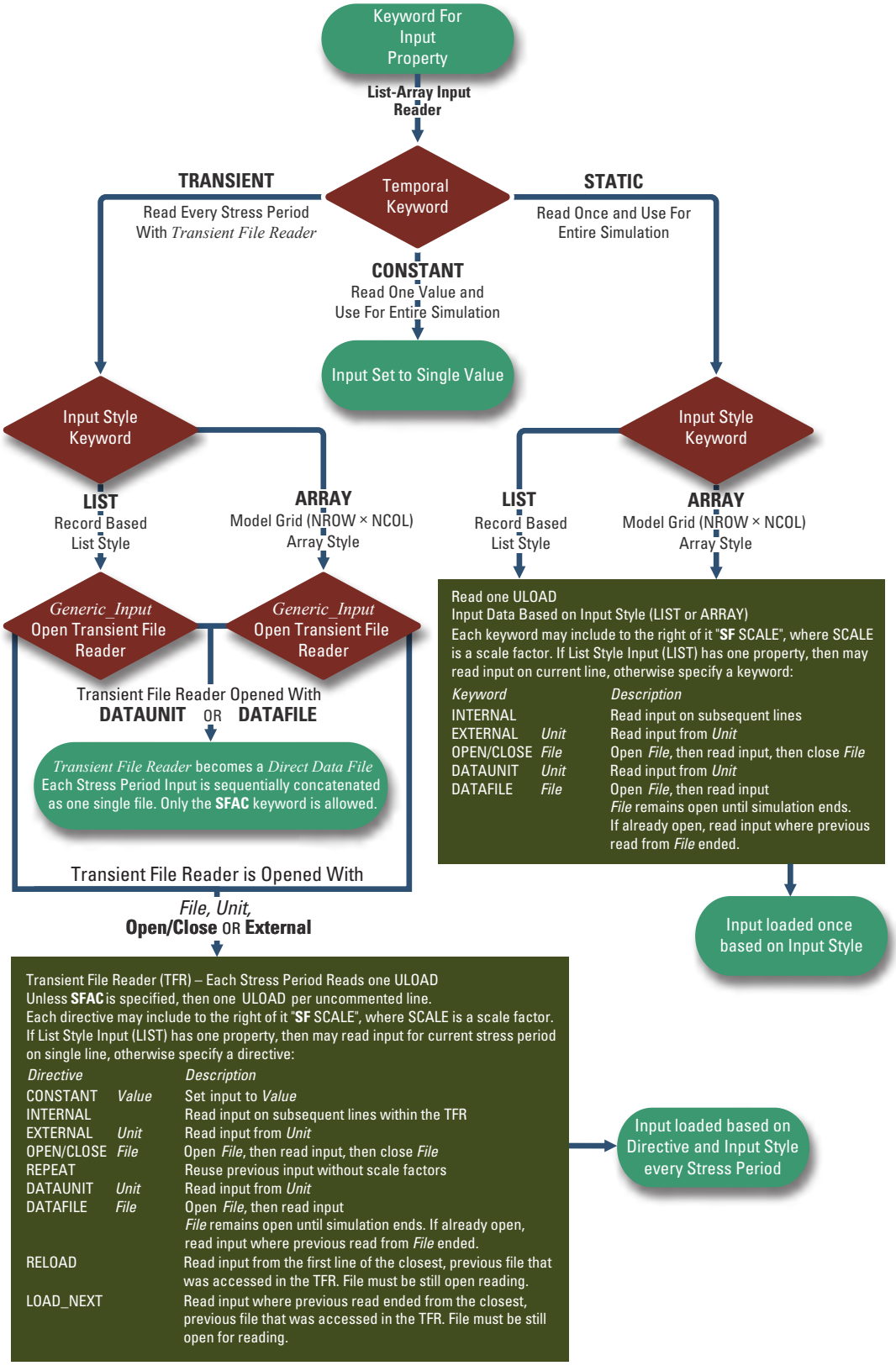


Figure 1.4. Flow chart showing the keyword-based control of new MF-OWHM2 input utilities.

```
1) KEYWORD  CONSTANT   VALUE
2) KEYWORD  STATIC     LIST   ULOAD
3) KEYWORD  STATIC     ARRAY  ULOAD
4) KEYWORD  TRANSIENT  LIST   TFR                  → Transient File Reader
5) KEYWORD  TRANSIENT  ARRAY  TFR                  → Transient File Reader
6) KEYWORD  TRANSIENT  LIST   DATAFILE FILE        → Direct Data File
7) KEYWORD  TRANSIENT  LIST   DATAUNIT UNIT        → Direct Data File
```

where

KEYWORD	is a package input keyword (PIK) that identifies the model input feature being set with the *List-Array Input* utility. The keyword depends on the package and the feature being loaded.
CONSTANT VALUE	indicates a single value, VALUE, is used for the entire simulation.
STATIC	indicates that the input is read once and used for the entire simulation.
TRANSIENT	indicates that the input is read for every time interval (stress period).
LIST	indicates that input uses *List Style*.
ARRAY	indicates that input uses *Array Style*.
ULOAD	is the *Universal Loader* that reads input information.
TFR	is a *Transient File Reader* that is opened with *Generic_Input_OptKey*. The TFR cannot be opened with **INTERNAL**, **DATAFILE**, or **DATAUNIT**.
DATAFILE FILE	indicates that the transient input bypasses the TFR and instead reads the data directly within the FILE. FILE is then the location of a *Direct Data File* (DDF).
DATAFILE UNIT	indicates that the transient input bypasses the TFR and instead reads the data directly within the UNIT number declared in the NAME file as " DATA UNIT FILE " or as " DATA(BINARY) UNIT FILE " where FILE is then the location of the DDF.
Transient File Reader	is a file that contains a directive keyword for each stress period that indicates where the input is located, then uses ULOAD to read the stress period's input.
Direct Data File	is a single file containing all the input for all stress periods. A DDF may only contain the actual input and, optionally, **SFAC** keywords. It is analogous to a TFR with only **INTERNAL** directives, but the DDF equivalent does not include the keyword **INTERNAL**.

Figure 1.5. Possible keyword combinations for an input that uses the *List-Array Input* utility and a brief explanation about them.

0) **SFAC**	[**DIMKEY**]	ULOAD	Optional, advanced scale factor features. Repeat as needed, using one **SFAC** per line of text. The **SFAC**s are multipliers for item 1–10. ULOAD reads a single scale factor, unless a **DIMKEY** keyword is included, which defines the number of scale factors read by ULOAD and how they are applied.
1) **CONSTANT**	VALUE		Set all input to VALUE.
2) **INTERNAL**		[**SF** SCALE]	Input on subsequent lines. **SF** SCALE is optional scale factor, do not include [ ], where SCALE is a number that multiplies with the input. **SF** SCALE may be repeated as needed on the same line
3) **OPEN/CLOSE**	FILE	[**SF** SCALE]	Input within FILE. First open file, then read input from first line, then close file after input is loaded.
4) **EXTERNAL**	UNIT	[**SF** SCALE]	Input is in file UNIT, which defined in the NAME file. First read of UNIT during a simulation loads from the first line. Subsequent references to UNIT read from current line. File UNIT remains open until simulation ends.
5) **DATAUNIT**	UNIT	[**SF** SCALE]	Same as **EXTERNAL**.
6) **DATAFILE**	FILE	[**SF** SCALE]	Check if FILE has been opened previously, either as a UNIT in the NAME file or use of **DATAFILE** FILE in the simulation. If FILE is open, then read input from current line. If not previously opened, then open FILE and read input from first line. FILE remains open until simulation ends and subsequent references to FILE read from current line.
7) **REPEAT**		[**SF** SCALE]	Reuse previously loaded, unscaled input. That is, the input before SCALE or **SFAC** is applied.
8) **RELOAD**		[**SF** SCALE]	Move to the first line, then read input from a file that was, in the TFR, used in the closest, previous **EXTERNAL**, **DATAUNIT**, or **DATAFILE** directive.
9) **LOAD_NEXT**		[**SF** SCALE]	Read input from the current line from a file that was, in the TFR, used in the closest, previous **EXTERNAL**, **DATAUNIT**, or **DATAFILE** directive.
10) **SKIP**			Set input to **0.0** or **0** or an empty string.

Figure 1.6. *Transient File Reader* (TFR) directive keywords that direct how input is loaded for each stress period. Each stress period must contain only one directive defined in items 1 through 10. **SFAC** (item 0) is optional and is only applied to the next directive keyword (items 1 through 10). **SF** SCALE is enclosed in brackets to indicate it is optional and is only applied to the directive with which it appears on the same line. **SFAC** and **SF** SCALE scaling only remain in effect for the directive they are applied to. If a new directive is specified, such as **REPEAT**, the scale factors are not carried forward, such that a **REPEAT** directive will repeat using only the previous, unscaled input and then apply any new scale factors associated with the new directive. [Note that these keywords also work with the *Universal Loader* (ULOAD), but have limited functionality outside of the scope of a TFR.]

```
# TFR Example for reading 9 Stress Periods (SP)
#
# The scale factor keyword, SFAC, can be specified before, after, or
# both before and after the TFR input directive (such as INTERNAL, OPEN/CLOSE, or REPEAT)

SFAC  1.25              # SP1 input is multiplied by 1.25
SFAC  0.80              # SP1 input is multiplied by 0.8
INTERNAL                # SP1 input is read on subsequent lines
   1.0   2.0   3.0
   8.0  16.0  24.0            # Blank lines are allowed

INTERNAL                # SP2 input is read on subsequent lines
SFAC  1.1               # SP2 input is multiplied (scaled) by 1.1
   2.0   4.0   6.0
   8.0  16.0  24.0

REPEAT                  # SP3 input uses SP2 unscaled input (ignores the "SFAC  1.1")

SFAC  1.25              # SP4 input is multiplied by 1.25
REPEAT                  # SP4 input uses SP2 input without SP2's scale factor
                        #
OPEN/CLOSE ./INP.txt    # SP5 input is read from "INP.txt". File is closed after input read.

DATAFILE   ./DAT.txt    # SP6 input is read from "DAT.txt".
                        # DAT.txt is opened and input is read from first line
                        # DAT.txt remains open until simulation ends

DATAFILE   ./DAT.txt    # SP7 input is read from "DAT.txt", which is already open.
                        # DAT.txt reads input from where the SP6 read ended

REPEAT     SF 1.25      # SP8 input uses SP7 input and multiplies it by 1.25

RELOAD                  # SP9 input is read from the start of "DAT.txt", because
                        # DAT.txt is still open and was the previously used file.
```

Figure 1.7. Example *Transient File Reader* (TFR) file that specifies the location of nine stress periods' input. The input is assumed to be *Array Style* that consists of 2 rows and 3 columns. [Note that these keywords also work with the *Universal Loader* (ULOAD), but have limited functionality outside of the scope of a TFR.]

Generic Input and Generic Output Files

A generalized file input and output Fortran module was written for MF-OWHM2 to standardize how files are opened and maintained. This is currently only implemented in the new MF-OWHM2 features, but may be incorporated to the older packages in subsequent upgrades and code releases. In this report, any documented input sections using the keywords *Generic_Input* or *Generic_Output* refer to this input. As the names indicate, *Generic_Input* refers to files that are "read only" and are meant to be loaded as input; *Generic_Output* refers to files that are "write only" and are opened for writing MF-OWHM2 output and results.

The sections that follow discuss how to set up a *Generic_Input* or *Generic_Output* file. It begins with some of the beneficial features that include buffering of files with the **BUFFER** option and the ability to split output files into multiple parts. An in-depth discussion follows about the difference between a text (ASCII or Unicode UTF-8) file and a MF-OWHM2 binary file. For binary files, methods of loading the data with Python scripts are discussed. Lastly, the formal input to the *Generic_Input* or *Generic_Output* file is presented with examples.

Buffering of Files

One important feature that *Generic_Input* and *Generic_Output* files have is the ability to specify a buffered value in kilobytes (KB)—using the post-keyword "**BUFFER** BUF_SIZE_KB"—that reserves additional random access memory (RAM) for file operations. Each *Generic_Input* and *Generic_Output* file has its own buffer that results in reduced input and output operations (I/O), leading to faster simulation runtimes at the expense of increasing total RAM usage. For *Generic_Input*, the buffer serves as space where the file preloads for faster input. For the *Generic_Output*, the buffer serves as space where output is written until the buffer is full; then the entire buffer is written to the file, and the buffer begins to refill again. This minimizes file writing to the hard drive for the *Generic_Output*, but the output file is only updated after the buffer is full. For example, if the LIST file has a 1024 kilobyte buffer (post-keyword "**BUFFER** 1024"), then it only updates the actual file in 1024 KB chunks. That is, after MF-OWHM2 writes a total of 1024 KB of text to the LIST file, the user is only then able to see the actual file updates. Empirical tests have shown the fastest performance is for buffer sizes between 32 KB and 1024 KB. By default, all files that are opened in MF-OWHM2 have their buffer set to 32 KB. If it is desired to have immediate writing of output files or no preloading of input files, then the buffer should be set to zero ("**BUFFER** BUF_SIZE_KB" is set to "**BUFFER** 0"). If a model has a design flaw or input structure that does not trigger a clean error message, that is a runtime Fortran stack error is raised, then it is recommended to zero the buffer for the LIST file to ensure that all its contents are written before the simulation halts.

Splitting Generic Output Files into Parts of the Same Size

For long simulations, output files can grow substantially in size causing issues for post-processors or even viewing the results. To overcome this issue, MF-OWHM2 has the option to split a *Generic_Output* file into a new file once the original reaches a user specified size in megabytes (MB), using the post-keyword "**SPLIT** SPLIT_SIZE_MB". The file size is checked at the end of each stress period to determine if it exceeds the SPLIT_SIZE_MB and needs to be split into a new file. When a file is split, the newly created file has the same file name, but with a number appended to the end of the filename root. The header in the original file is also included in each of the split files created, and when the simulation is restarted, all split files are removed. For example, if the original output file was "MyFile.txt" and the keyword used for it was "**SPLIT** 900", then each time the file size exceeds 900MB, a new file is created with the following naming sequence: MyFile.txt, then MyFile01.txt, and then MyFile02.txt. This is most useful for the LIST file, which can become very large.

Text and Binary Format of Generic Input and Generic Output

The *Generic_Input* and *Generic_Output* both support text (ASCII/Unicode UTF-8) format and binary format. The text format is the default and is a human readable format when opened with any text editor. Most text files are stored in the UTF-8 Unicode format, which is backward compatible with ASCII. All text files read by MF-OWHM2 must use the ASCII or UTF-8 Unicode character encodings. It should be noted that the Microsoft Windows program Notepad.exe saves text files using the UTF-8-BOM format that adds a Byte Order Mark (BOM) binary header to a UTF-8 text file. Fortran does not support reading the UTF-8-BOM encoding format, so this text file format should not be used with any program written in Fortran. MF-OWHM2 does check for a Microsoft Windows style BOM and if found will position the start of the file just past the BOM and process the file using UTF-8 character encoding.

In MF-OWHM2, the binary format uses standard FORTRAN 2003 Unformatted Stream I/O, which may have limited portability among computers. The limited portability is not operating system based (for example, Windows, Linux, or Unix), but is the result of the central processing unit (CPU) "endianness" making the binary CPU dependent. There are two types of binary endianness, referred to as big-endian or little-endian format. Most microprocessors for servers and desktops (x86, x86-64, x64, and AMD64 instruction) use the little-endian, making a binary file portable among them.

Binary files can be read by separate Fortran programs that open the file with the ACCESS="STREAM" option. The file can also be loaded using Python structures or Python's Numpy Module with the "numpy.fromfile" method. The file must be loaded with the same variable size for each variable position. For example, if the first value in the binary file is a double precision number and the second variable is an integer, then it must be loaded with a double precision variable followed by an integer variable to load correctly. For Fortran, this method is direct and uses the same naming, but for Python, the user has to specify the "numpy.dtype". When a binary file output is requested in MF-OWHM2, its record structure is written to the LIST file, so the user can reconstruct its output. The records use keywords, described in figure 1.8, to indicate the type of storage.

In addition to the keyword, there could be a dimension to the variable written. This is to accommodate arrays that are written to the binary file. To identify arrays, brackets are used with the dimension enclosed within them (for example, [10] or [5,15]). Within the brackets, the words NROW, NCOL, and NLAY represent the model's number of rows, columns, and layers, respectively. Figure 1.9 presents an example of output that is written to the LIST file to indicate the binary file's record structure.

This example indicates that each record written to the binary file writes the DATE_START as 19 characters, PER as an integer, STP as an integer, DYEAR as a double precision, and DATA as a double precision array of size NCOL by NROW. Note that most MODFLOW arrays are stored with the model grid's "column" on the first array dimension rather than the second. That is, computer arrays are specified as [Dimension 1, Dimension 2, Dimension 3, …], so "DATA[NCOL,NROW]" indicates that the first array dimension is of size NCOL (the number of model columns) and the second array dimension is of size NROW (the number of model rows). Figure 1.10 and 1.11 are code examples in Fortran and Python, respectively, necessary to load one record from the figure 1.9 binary-stream output. Fortran can use its native variables, if they are the same type and the file is opened using the options ACCESS="STREAM" and FORM="UNFORMATTED" options. Conversely, Python 3 requires using the NumPy module; defining the binary structure using "numpy.dtype" and loading the binary data with the "numpy.fromfile" method.

Keyword	Storage	Fortran Type	Python Numpy Type
(double) →	8 bytes,	Double Precision,	numpy.dtype('float64')
(int) →	4 bytes,	Integer,	numpy.dtype('int32')
(X char) →	X × 1 bytes,	Character(X),	numpy.dtype('SX')
(sngl) →	4 bytes,	Real,	numpy.dtype('float32')

Figure 1.8. Definition of keywords used by MF-OWHM2 to define a stream-binary file's memory storage for different variable types and the type of Fortran and Python variables that can read them. [X is used as a place holder for an integer number, such as Character(19) or numpy.dtype('S19'). NumPy is a library for the Python programming language. Fortran assumes that REAL and DOUBLE PRECISION are not modified by compiler options. To make Fortran compiler option independent, it is recommended to use the parameters INT32, REAL32, and REAL64 from the intrinsic module ISO_FORTRAN_ENV and declare INTEGER variables as INTEGER(INT32), REAL variables as REAL(REAL32), and DOUBLE PRECISION variables as REAL(REAL64).]

```
DATE_START (19char), PER (int), STP (int), DYEAR (double), DATA[NCOL,NROW] (double)
```

Figure 1.9. Example output, written to the LIST file, that indicates the structure of a MF-OWHM2 binary-stream output file. The text not enclosed in parenthesis is the variable name and the part in the parenthesis indicates the binary-stream format used. [Note that MODFLOW stores arrays as NCOL by NROW. 19char indicates the record is a Fortran character of length 19, int indicates the record is a 4-byte Fortran integer (INT32), double indicates the record is an 8-byte Fortran floating point number (REAL64), which is commonly called double precision.]

```
PROGRAM READ_BINARY
USE, INTRINSIC:: ISO_FORTRAN_ENV, ONLY: REAL32, REAL64

! REAL32 is single precision -> REAL(REAL32) ⇔ REAL              (4 bytes)
! REAL64 is double precision -> REAL(REAL64) ⇔ DOUBLE PRECISION  (8 bytes)

INTEGER, PARAMETER:: NROW = 5, NCOL = 10  ! Dimension of Model Grid

CHARACTER(len=19):: DATE_START
INTEGER:: PER, STP
REAL(REAL64):: DYEAR
REAL(REAL64), DIMENSION(NCOL, NROW):: DATA  ! MODFLOW Stores Arrays as NCOL x NROW

INTEGER:: IU  ! Variable holds Fortran unit number associated with binary file

IU = 0        ! Initialize the variable, will be set by OPEN with Fortran unit number

OPEN(NEWUNIT=IU, FILE="myfile.bin", STATUS="OLD", ACCESS="STREAM", FORM="UNFORMATTED")

! Read one binary record from file unit IU

READ(IU) DATE_START, PER, STP, DYEAR, DATA

END PROGRAM
```

Figure 1.10. Fortran code example that can read a stream-binary file "myfile.bin" that contains binary record composed of 19 characters, 2 integers, a double-precision real number, and then a double precision array of size that is 5 model rows and 10 model columns. [Note that MODFLOW stores arrays as NCOL by NROW, where NROW is 5 and NCOL is 10. DATE_START is the calendar date at the start of the time step, PER is the stress period number, STP is the time step number, DYEAR is a decimal year representation of DATE_START, DATA is the NCOL by NROW array that is read.]

```
import numpy

NROW = 5
NCOL = 10

# Note that MODFLOW stores binary arrays as NCOL x NROW

# But Python stores arrays by row, while Fortran stores array by column,
#   so unlike a Fortran read, Python reads NROW x NCOL
#   and then must transpose the result to get the original shape (NCOL, NROW)

dt = numpy.dtype([
                 ('DATE_START', numpy.dtype('S19')                  ),
                 ('PER',        numpy.dtype('int32')                ),
                 ('STP',        numpy.dtype('int32')                ),
                 ('DYEAR',      numpy.dtype('float64')              ),
                 ('DATA',       numpy.dtype('float64'), (NROW, NCOL) ),
                 ])

# Set "F" to binary file that will be read from

F = open('myfile.bin','rb')

# Read single record defined with dt from file "F"

REC = numpy.fromfile(F, dtype=dt, count=1)

# REC holds each record as an array of length 1,
#   the following extracts the binary contents to individual variables

DATE_START = REC['DATE_START'][0]        # Get stored starting date
PER        = REC['PER'       ][0]        # Get stress period number
STP        = REC['STP'       ][0]        # Get time step number
DYEAR      = REC['DYEAR'     ][0]        # Get starting date decimal year
DATA       = REC['DATA'][0].transpose()  # Get DATA as NCOL x NROW array
```

Figure 1.11. Python 3 code example that can read a stream-binary file "myfile.bin" that contains binary record composed of 19 characters, 2 integers, a double precision real number, and then a double precision array of size that is 5 model rows and 10 model columns. [Note that MODFLOW stores arrays as NCOL by NROW, where NROW is 5 and NCOL is 10. Python reads arrays using the right most dimension first, whereas Fortran reads the left most dimension first. Because of this, the DATA array, which was written with Fortran as NCOL by NROW, is read by Python with an array dimensioned as NROW by NCOL, then must be transposed to get the original MODFLOW array. DATE_START is the calendar date at the start of the time step, PER is the stress period number, STP is the time step number, DYEAR is a decimal year representation of DATE_START, DATA is the NROW by NCOL array that is read, REC holds the first record in myfile.bin read by numpy.fromfile.]

Input Structure

The *Generic_Input* and *Generic_Output* module systematically looks for keywords to indicate how to access or open a file and what options to include with it. When keywords are optional (fig. 1.12*A*, items 1 and 7), then the "_OptKey" is added to *Generic_Input* and *Generic_Output*, changing it to *Generic_Input_OptKey* or *Generic_Output_OptKey.* Because of the potential for ambiguity of input options, it is recommended to use the keywords even when they are optional.

The order in which the *Generic_Input_OptKey* and *Generic_Output_OptKey* module detects and opens a specified file is to first check to see if it can load a single integer; if so, then the integer indicates that it is a unit number that is defined in NAME file, which is associated with a DATA or DATA(BINARY) file (fig. 1.12*A*, item 1). If it fails to read an integer, then it checks to see if there are keywords for the unit number; keywords to open a file; or lastly, if the line just contains a file name to open (fig. 1.12*A*, items 2 through 7). Once the file has been identified through its unit number or file name, there are a set of optional post-keywords and scale factors available (fig. 1.12*A*, items A through H), that override default options or allow advanced file operations. Any comments to the right and on the same line must be preceded by a "**#**" symbol (commonly called a number sign, pound sign, or hash symbol). Preceding a comment by "**#**" symbol is necessary to indicate to MF-OWHM that the text to the right is a comment and not a post-keyword. Figure 1.12*A* contains the decision order for how the file is detected, opened, and what post-keywords can be applied, and a description of the keywords is in figure 1.12*B* and figure 1.12*A*, *C*. Because of the potential for ambiguity of input, it is recommended to use the keywords **`EXTERNAL`** and **`OPEN/CLOSE`** (fig. 1.12) rather than directly loading the `UNIT` or `FILE` name for the *Generic_Input_OptKey* and *Generic_Output_OptKey* versions.

The two most powerful optional keywords for the *Generic_Input* are the **`SF`** `SCALE` and **`REWIND`**. When **`SF`** `SCALE` is loaded, it is multiplied by any data loaded from the *Generic_Input*. Multiple scale factors are allowed to provide clarity for the input. For example, two scale factors could be used to separate a unit conversion factor from a calibration factor. Continuing this example, an input line that reads from the file MyText.txt could be "**`OPEN/CLOSE`** `MyFile.txt` **`SF`** `0.3048` **`SF`** `1.05`", where `0.3048` is a unit conversion factor and `1.05` is a calibration adjustment factor that increases the input by 5 percent. The **`REWIND`** option resets a file that is already opened (excluding **`INTERNAL`**) to the first line. This is advantageous if a file only has a certain number of input records that are repeated after a set number of stress periods. For example, if an input was repeated every 12 stress periods (that is, one input for each month), then it only requires 12 lines to represent the 12 months in one file that is opened with **`DATAFILE`**, **`DATAUNIT`**, or **`EXTERNAL`**. Once the file is accessed and read 12 times for the 12 stress periods, the file utility then uses the keyword **`REWIND`** to move to the start of the file to cycle through another 12 stress periods.

The following is a set of examples for different ways to access or open a file using *Generic_Input* or *Generic_Input_OptKey*:

```
INTERNAL    # READ input on subsequent lines
```

```
# READ input on subsequent lines, multiply input loaded by 2.5 and 1.25
INTERNAL  SF 2.5   SF 1.25
```

```
# READ input on subsequent lines, multiply input loaded by 2.5 and 1.25
INTERNAL  2.5  1.25
```

```
# READ input on subsequent lines, multiply input loaded by 2.5, "#" excludes 1.25
INTERNAL  2.5  # 1.25
```

```
OPEN/CLOSE MyFile.txt # Open MyFile.txt, load its contents, then close file
```

```
# Open MyFile.txt, load its contents and multiply it by 2.5 and 1.25, then close file
OPEN/CLOSE MyFile.txt    SF 2.5  SF 1.25
```

```
# Open MyFile.txt, load its contents and multiply it by 2.5 and 1.25, then close file
OPEN/CLOSE MyFile.txt        2.5      1.25
```

```
EXTERNAL 55    # Load contents from Unit 55 as specified in the NAME file
```

```
# Load contents from Unit 55 as specified in the NAME file, multiply it by 2.5 and 1.25
EXTERNAL 55   SF 2.5  SF 1.25
```

```
# Load contents from Unit 55 as specified in the NAME file, multiply it by 2.5 and 1.25
EXTERNAL 55      2.5     1.25
```

The following are additional examples of different ways to access or open input with *Generic_Input_OptKey*:

```
MyFile.txt          # Open MyFile.txt, load its contents, then close file
```

```
# Open MyFile.txt, load its contents and multiply it by 2.5 and 1.25, then close file
MyFile.txt   SF 2.5  SF 1.25
```

```
# Open MyFile.txt, load its contents and multiply it by 2.5 and 1.25, then close file
MyFile.txt      2.5     1.25
```

```
55                  # Load contents from Unit 55 as specified in the NAME file
```

```
# Load contents from Unit 55 as specified in the NAME file, multiply it by 2.5 and 1.25
55   SF 2.5  SF 1.25
```

```
# Load contents from Unit 55 as specified in the NAME file, multiply it by 2.5 and 1.25
55      2.5     1.25
```

The following is a set of examples for different ways to specify keywords for *Generic_Output* or *Generic_Output_OptKey*:

```
INTERNAL                # Write output to LIST file
```

```
LIST                    # Write output to LIST file, same as INTERNAL
```

```
OPEN/CLOSE MyFile.txt # Open MyFile.txt and write output to it
```

```
EXTERNAL 55    # Write output to Unit 55 as specified in the NAME file
```

The following are additional examples for different ways of setting up the output for *Generic_Output_OptKey*:

```
MyFile.txt      # Open MyFile.txt and write output to it
```

```
55                  # Write output to Unit 55 as specified in the NAME file
```

The following are examples for present the usage of post-keywords for *Generic_Output and Generic_Output_OptKey*:

```
OPEN/CLOSE MyFile.txt        SPLIT 500  #MB
```

```
# Write output to Unit 55 and split file every 500MB. Note if file is binary,
# then it should be declared as DATA(BINARY) in NAME file
55 SPLIT 500
```

The second to last example would split the *Generic_Output* file "MyFile.txt" if its size exceeded 500 megabytes and begin writing output to a new file called "MyFile01.txt". If this new file size exceeded 500 megabytes, then it would be split, and output would be written to the file "MyFile02.txt". This splitting of files continues until the MF-OWHM2 simulation is completed. The following is an example of buffering a file with 512 kilobytes of memory:

```
# This works for Generic_Input too
OPEN/CLOSE MyFile.txt        BUFFER  512  #KB buffer for file
```

Both **BUFFER** and **SPLIT** can be used simultaneously, and the order does not matter. This allows for a file to be buffered and split into multiple files. The following is an example of using both keywords, which would turn off buffering of a binary file, but still split the file whenever it was greater than 500 megabytes in size:

```
# Open MyFile.txt as binary, with no buffer and split to new file every 500MB
MyFile.txt  BINARY  BUFFER 0  SPLIT 500
```

or

```
OPEN/CLOSE MyFile.txt  SPLIT 500  BINARY   BUFFER 0
```

Finally, the keyword **BINARY** may be placed at the start of a *Generic_Input* or *Generic_Output* input section:

```
BINARY OPEN/CLOSE MyFile.txt  SPLIT 500  BUFFER 0
```

A

- One of the following items must be present accessing or opening a file with *Generic_Input*, *Generic_Output*, *Generic_Input_OptKey*, or *Generic_Output_OptKey*:

  1) `UNIT` → *Generic_Input_OptKey* and *Generic_Output_OptKey*
  2) **`INTERNAL`**
  3) **`EXTERNAL`** `UNIT`
  4) **`DATAUNIT`** `UNIT`
  5) **`OPEN/CLOSE`** `FILE`
  6) **`DATAFILE`** `FILE`
  7) `FILE` → *Generic_Input_OptKey* and *Generic_Output_OptKey*
  8) **`NOPRINT`** → *Generic_Output* and *Generic_Output_OptKey*

- The following are optional, post-keywords that are checked for after items 1–7.
  The order of the post-keywords does not matter nor do any have to be specified.
  Any option supported by *Generic_Input* is supported by *Generic_Input_OptKey*.
  Any option supported by *Generic_Output* is supported by *Generic_Output_OptKey*.

  A) **`BINARY`** → *Generic_Input* and *Generic_Output*
  B) **`BUFFER`** `BUF_SIZE_KB` → *Generic_Input* and *Generic_Output*
  C) **`SPLIT`** `SPLIT_SIZE_MB` → *Generic_Output*
  D) **`NOPRINT`** → *Generic_Output*
  E) **`REWIND`** → *Generic_Input*
  F) **`DIM`** `DIM_SIZE` → *Generic_Input*
  G) **`SF`** `SCALE` → *Generic_Input*
  H) `SCALE` → *Generic_Input*

Figure 1.12. Syntax for accessing or opening files with the utilities: *Generic_Input*, *Generic_Output*, *Generic_Input_OptKey*, and *Generic_Output_OptKey*. *A*, The keyword decision order for how the file is detected, then either accessed or opened (items 1 through 8), and what post-keywords can be applied (items A through H). *B*, Explanation of items 1 through 8 in part *A*. *C*, Explanation of the post-keywords in part *A*. [The symbol, → indicates that the option is only available to a specific file utility.]

B

INTERNAL	indicates that *Generic_Input* files are read on subsequent lines, and *Generic_Output* files are written to the LIST file.
NOPRINT	indicates that *Generic_Output* output will not be written.
UNIT	is the unit number of a file opened in the NAME file with DATA or DATA(BINARY).
EXTERNAL	indicates that *Generic_Input* files are read from a file associated with UNIT, and *Generic_Output* files are written to a file associated with UNIT.
DATAUNIT	This option is identical to **EXTERNAL**, except if the expected file to be opened is a *Transient File Reader* (TFR), then **DATAUNIT** indicates that the TFR is bypassed the UNIT is a *Direct Data File* (DDF). A TFR and DDF are part of the *List-Array Input* (LAI).
BINARY	is a keyword that indicates FILE should open as a **BINARY** file. The keyword may be placed at either the beginning or ending of the *Generic_Input* and *Generic_Output* input.
FILE	is the file name and location (file path) that will be opened for reading or writing to. Note that item 7 is equivalent to specifying **OPEN/CLOSE** FILE.
OPEN/CLOSE	indicates that FILE is to be opened and read or written to at the start of the file. FILE is closed when it is no longer required. *Generic_Input* file is closed after reading input. Note, a TFR reads input until the simulation ends. *Generic_Output* is closed when either the simulation ends or it is no longer used for writing output.
DATAFILE	indicates that if FILE is not open, then open it; The file remains open until simulation ends. Any subsequent **DATAFILE** references to the same FILE continue reading or writing at the file's previous position. This is analogous to **EXTERNAL** UNIT, but uses the file name, FILE, instead of UNIT, and it is not required to define FILE in the NAME file. If the expected file to be opened is a TFR, then **DATAFILE** indicates that the TFR is bypassed and FILE is a DDF.

Figure 1.12. —Continued

C

BUFFER	indicates that the input or output file should be buffered in RAM to improve speed. The size of the buffer is in kilobytes (KB). When not specified the default buffer is 32KB. *Generic_Input* preloads the file into RAM for reading. *Generic_Output* writes output first to RAM, then writes to the actual file in `BUF_SIZE_KB` chunks.
`BUF_SIZE_KB`	Size of the buffer in KB. If the buffer is set to `0`, then *Generic_Input* disable preloading of the file before reading. *Generic_Output* results in immediate writing of output.
SPLIT	indicates that a *Generic_Output* file should split into a new file after a specified size in megabytes (MB). The new file uses the original files name with a sequential number appended to it to not overwrite the original.
`SPLIT_SIZE_MB`	is the minimum size, in MB, of the output before it is split.
REWIND	indicates that the file's position is reset to the start of the file before the file is read from or written to This is done by default for newly opened files. This option is ignored when used with the keyword **`INTERNAL`**.
DIM	sets user-specified dimension, `DIM_SIZE`, for the input file. The use of `DIM` depends on what the input file is used for. If the input file does not support `DIM`, then a warning is raised. One notable use is that if `DIM_SIZE` is specified when opening a *Transient File Reader* (`TFR`) and *Direct Data File* (`DDF`), then it specifies the largest line size that is read as input and overrides the default size of 700 characters.
`DIM_SIZE`	is an integer number that is specified after **`DIM`**.
SF	a scale factor, `SCALE`, is read and multiplied with the input data. "**`SF`** `SCALE`" may be repeated multiple times. Scale factors are ignored for input that is expected as an integer (`int`).
`SCALE`	is a scale factor that is read and multiplied with the input data. The use of the keyword **`SF`** is optional but recommended for clarity. "`SCALE`" may be repeated multiple times.
NOPRINT	suppresses the creation and writing of an output file.

Figure 1.12. —Continued

ULOAD Input Utility and SFAC Keyword—Universal Loader and Scale Factors

The following section introduces the *Universal Loader* (ULOAD) input utility and optional keyword **SFAC**, which provides supplemental scale factors for modifying input. ULOAD is an array reading utility that is capable of loading text, integers, and floating-point numbers that are structured as *List Style* input or *Array Style* input. *List Style* is a record-based input that reads one record per row; a single record starts with a record identifier (ID, an integer) and is followed by one or more columns of input data. *Array Style* loads a two-dimensional array of input data without a record ID (identifier). *List Style* may have its record ID associated with a specific input property or may be linked to a spatial identification array that is loaded with the *Array Style*. Typically, *Array Style* is used for loading spatial data that conform to the MF-OWHM2 model grid (NROW by NCOL). Ultimately, the key difference between the *List Style* and *Array Style* is whether record IDs are read or not.

Note that MODFLOW, and consequently MF-OWHM2, reads (or writes) arrays from (or to) text files with the dimension NROW by NCOL (traditional matrix structure), but reads (or writes) arrays from (or to) **BINARY** files with the dimension NCOL by NROW. This is the reason why figures 9 and 10 dimension the array as (NCOL, NROW), which is an array that is read from a binary file.

ULOAD—A Universal Array and List Load Utility

The MF-OWHM2 ULOAD input utility is capable of loading either *List Style* or *Array Style* input data. The utility automatically skips blank lines and ignores any commented text, which must be preceded with "#" symbol. The data loaded depend on the input keywords specifying what the dataset comprises, which can be either integer, floating point (single/double precision), or character data (for example, ASCII text). The specific data type is defined by the input package that is relying on ULOAD to load the data.

List Style reads a row at a time that starts with a record ID (as an integer) then multiple records—properties—to the right of it. The *Array Style* loads a two-dimensional array (typically, NROW by NCOL) and must be formatted in the structure of the array (for example, it must have NCOL inputs for each row and must have NROW rows). The input data may be space separated, comma separated, or tab separated, and comments are allowed between rows and after the last column of input. Figure 1.13 presents two *List Style* input examples and one and *Array Style* example.

Unless otherwise stated in the input section of the utility that is calling ULOAD, the row record ID is the only placeholder not used by MF-OWHM2. It is instead required to provide an ID to help the user identify the input ID (or row of input). In addition, unless otherwise stated by the input section, the rows must be well ordered, irrelevant of the ID, such that the first row applies to record 1, and the second row applies to record 2, and so forth. The number of rows that are read is specified by each input section.

ULOAD is capable of loading binary files. This is an advanced input and is discussed here only to provide a complete description of ULOAD. ULOAD imposes several limitations to the structure of the binary file. First, the *Generic_Input* must include the post-keyword **BINARY** or be associated with a unit that was opened with DATA(BINARY) in the NAME file. Second, the *List Style* input cannot have the record ID in the binary file. The binary structure using *List Style* input is such that the first property (from the first record to the last record) is written first, then the second property, and so forth. Third, the *Array Style* binary file array structure will be defined by the input utilizing it. If there is no description of the input structure defined, the *Array Style* input assumes that the array is written in binary format such that the first column is first, then the second column, and so forth. This assumption coincides with the way MODFLOW stores arrays as (NCOL, NROW). The advantage of binary files compared to text files is that they load faster and are typically smaller in size. Because text files are easier to maintain and more portable, however, they are the recommended input format.

A special case is ULOAD_NoID, which follows the same rules, but does not read the row record ID (only the data are read in). In fact, *Array Style* uses ULOAD_NoID to read a two-dimensional array. In addition to this, ULOAD_NoID is how a single value (for example, a single integer, scalar or floating-point number, or single word) is loaded. ULOAD_NoID can be thought of as a generalization of the standard MODFLOW input utilities *U1DREL*, *U2DREL* and *U2DINT*, because it performs the same operation without requiring format codes and automatically adjusts for single precision and integer input (and for some special inputs can read text input, such as a name).

The ULOAD input data type (integer, floating point, text) and input style is specified by the calling package, and within the model input dataset being loaded. ULOAD first attempts to identify a keyword (fig. 1.14*A*, items 1 through 4). If ULOAD does not identify a keyword, then the input is assumed to be specified as an *Implied_Internal* and located along the current line. If an *Implied_Internal* is not allowed, then an error is raised, and the simulation stopped.

The following is a set of examples that use ULOAD to read in three values (Val1, Val2, Val3) that define one input property (for example, root depth for NCROP = 3 Crops). The first example is for using a constant value:

```
CONSTANT VALUE          # Sets Val1 Val2 Val3 Equal to VALUE
```

The **INTERNAL** keyword requires a record ID:

```
INTERNAL
ID1  Val1
ID2  Val2
ID3  Val3
```

If no keyword is present and an *Implied_Internal* is allowed, then the input is loaded along the same line, without the List Style record ID. The following is an example *Implied_Internal*:

```
Val1 Val2 Val3   # Load Val1 Val2 Val3 to Record ID 1, 2, 3, respectively
```

The input can be specified in a separate file. One method of loading input is to use the keyword **OPEN/CLOSE**:

```
OPEN/CLOSE MyFile.txt
```

The file "MyFile.txt" contains the following:

```
ID1  Val1
ID2  Val2
ID3  Val3
```

The final example uses **EXTERNAL**:

```
EXTERNAL 55
```

The entry in the NAME file (NAM) is as follows:

```
DATA 55 ./MyFile.txt   READ BUFFER 16
```

The file "MyFile.txt" contains the following:

```
ID1  Val1
ID2  Val2
ID3  Val3
```

For these examples, values of ID1, ID2, and ID3 would be 1, 2, and 3 to serve as the place holder, and Val1, Val2, and Val3 would be the input property (for example, 3.14, 2.718, and 1.618).

As described before, ULOAD is capable of loading multiple properties per record. In this case, ULOAD does not allow for an *Implied_Internal* because of the additional column dimensions. The following is an example of a ULOAD with three records that load property A and B.

```
INTERNAL
ID1  Val1_A  Val1_B
ID2  Val2_A  Val2_B
ID3  Val3_A  Val3_B
```

Because ULOAD uses *Generic_Input* to check for the keywords **INTERNAL**, **EXTERNAL**, and **OPEN/CLOSE**, it also supports all the post-keywords (for example, **BUFFER**, **BINARY**, **SF** SCALE). To illustrate the application of **SF** SCALE, the following example loads the values in bold and then multiplies them by SCALE1 and SCALE2, which represent multiple instances of **SF** SCALE.

```
INTERNAL
ID1  Val1_A  Val1_B
ID2  Val2_A  Val2_B
ID3  Val3_A  Val3_B
```

or

```
INTERNAL
ID1  Val1_A  Val1_B
ID2  Val2_A  Val2_B
ID3  Val3_A  Val3_B
```

```
INTERNAL  SF SCALE1  SF SCALE2
ID1  Val1_A  Val1_B
ID2  Val2_A  Val2_B
ID3  Val3_A  Val3_B
```

A

```
# List Style Input; 3 IDs and 1 Properties
# ID  Prop1
  1  Val       # Comment
  2  Val
  3  Val
```

B

```
# List Style Input; 3 IDs and 2 Properties
# ID  Prop1  Prop2
  1  Val  Val # Comment
  2  Val  Val
  3  Val  Val
```

C

```
# Array Style Input; 4 rows and 5 columns
 Val  Val  Val  Val  Val # Comment
 Val  Val  Val  Val  Val
 Val  Val  Val  Val  Val
 Val  Val  Val  Val  Val
```

Figure 1.13. Example input structures expected by *Universal Loader* (ULOAD). *A*, *List Style* input that reads three records that define one property. *B*, *List Style* input that reads three records that each define two properties. *C*, *Array Style* input that reads a 4-by-5 array. [Val is a placeholder that represents a single value of the format expected by the input using ULOAD.]

A

0) **SFAC** [**DIMKEY**] ULOAD	Optional and may be repeated with one **SFAC** per line. Defines advanced scale factors that are applied to input indicated by items 2–5. **SFAC** scale factors are read with a separate instance of ULOAD. If present, then item 2, 3, 4, or 5 is expected on the next uncommented line.
1) **SKIP**	
2) **REPEAT** [**SF** SCALE]	Only allowed when the same input was previously loaded with ULOAD. Typically, this only occurs within a *Transient File Reader*.
3) **CONSTANT** VALUE	
4) *Generic_Input* [**SF** SCALE]	Specify the location of the input with **INTERNAL**, **EXTERNAL**, **OPEN/CLOSE**, **DATAUNIT**, or **DATAFILE**. At the start of the input within the *Generic_Input* "**SFAC** [**DIMKEY**] ULOAD" can optionally be included.
5) *Implied_Internal*	Input is assumed to be on current line if and only if the expected input is a single VALUE, or *List Style* input with only one property, or *Array Style* input that is a vector (one-dimensional array).

Figure 1.14. *Universal Loader* (ULOAD) supported keywords. *A*, The syntax of ULOAD keywords, ordered by how ULOAD checks for keywords; for example, **REPEAT** is checked for before **CONSTANT**. *B*, Explanation of items 0 through 5 in part *A*. [**SF** SCALE is enclosed in brackets to indicate it is optional and is only applied to the directive with which it appears on the same line. **SFAC** and **SF** scaling only remain in effect for the directive they are applied to. If a new directive is specified, such as **REPEAT**, the scale factors are not carried forward, such that a **REPEAT** directive will by default repeat only the unscaled input.]

B

SFAC [**DIMKEY**] ULOAD

SFAC indicates that advanced scale factors are read with a separate instance of ULOAD and applied to the input read by items 2–5.

SFAC may be repeated as necessary, but only one is allowed per line.

DIMKEY is an optional keyword that indicates the number of scale factors read and how they are applied to the input.

If **DIMKEY** is not present or not supported, then **SFAC** reads a single scale factor that is applied to all the input.

If **DIMKEY** is supported, then the accepted keywords and how they are applied to the input are defined by the input that supports it.

SKIP specifies that values input properites are set to zero.

REPEAT specifies input that was previously loaded should be reused.

Repeated input does not carry forward any **SFAC**, **SF**, or SCALE scale factors from the previous load; thus it requires the scale factors to be repeated to reproduce the same input.

REPEAT is only allowed if the dataset has been previously loaded by ULOAD (that is, the input from the previous stress period).

SF SCALE specifies an optional scale factor, SCALE, that is applied to the **REPEAT** input or the input read from the *Generic_Input*. Multiple **SF** SCALE are allowed but must be on the same line. SCALE, without the keyword **SF**, is accepted in the place of "**SF** SCALE".

CONSTANT specifies that VALUE should be used as to the input. VALUE must match the data type that the input utility expects, which can be integer, floating point, or text.

Generic_Input uses the keywords **INTERNAL**, **EXTERNAL**, **OPEN/CLOSE**, **DATAUNIT**, and **DATAFILE** to specify the location of the input.

All the *Generic_Input* post-keywords are supported, including **SF** SCALE.

SFAC [**DIMKEY**] ULOAD can be used within the *Generic_Input* on the line (or lines) before the expected input data begins.

Implied_Internal assumes that the input data is located along the current line because items 1–4 are not identified.

An error is raised if input is not one of the following cases:

- A scalar—that is a single input value.
- *List Style* input that only has one property (record ID and one column)
- *Array Style* input that expects a vector (one-dimensional array)

A single property *List Style* input in an *Implied_Internal* does not include the record ID and assumes that the first input has a record ID of 1, and the second has a record ID of 2, and so forth.

Figure 1.14. —Continued

SFAC—Scale Factor Keyword

MF-OWHM2 input sections may specify that they support **SFAC** as a keyword. **SFAC** allows loading and applying scale factors to a set of input data. **SFAC** is automatically multiplied with any loaded *Generic_Input* "SCALE" values to derive a composite scale factor. **SFAC** scale factors rely on a *List Style* ULOAD (see previous section) to read either a single scale factor or a set of scale factors. If **SFAC** reads a set of scale factors, then it is required to include the keyword **DIMKEY** to specify the number of scale factors and how they should be applied. Figure 1.15 presents the general input structure when using **SFAC**.

Because **SFAC** relies on ULOAD with *List Style* input to read the scale factors, it must include a record ID. If the scale factors are read with an *Implied_Internal*, then the record ID is not read. For simplicity of input, it is recommended to use the *Implied_Internal* for scale factors.

The following are examples of ways to load scale factors for an input utility that accepts a **DIMKEY** of "**ByWBS**" and has three water balance subregions (WBS or farms with NWBS = 3). Note that **SFAC** always accepts the **DIMKEY** keyword **ALL**, but it is optional because **SFAC** automatically reads a single scale factor if **DIMKEY** is not present.

The following examples load a single scale factor, SVAL, that is multiplied by whatever input it is associated with. The first two text boxes use a ULOAD *Implied_Internal* to load SVAL, and the second two boxes use ULOAD with a *Generic_Input* keyword to specify where to find SVAL—note ULOAD requires the record ID:

```
SFAC SVAL
```

or

```
SFAC ALL SVAL
```

or

```
SFAC INTERNAL
ID1   SVAL
```

or

```
SFAC OPEN/CLOSE MyText.txt
```

MyText.txt contains the following:

```
ID1   SVAL
```

The following illustrate using ULOAD with an *Implied_Internal* and two *Generic_Input* keyword examples that load the scale factors SVAL1, SVAL2, SVAL3 to be applied where WBS 1, 2, and 3 are located, respectively:

```
SFAC ByWBS  SVAL1 SVAL2 SVAL3
```

```
SFAC ByWBS INTERNAL
ID1  SVAL1
ID2  SVAL2
ID3  SVAL3
```

```
SFAC ByWBS OPEN/CLOSE MyText.txt
```

MyText.txt contains the following:

```
ID1  SVAL1
ID2  SVAL2
ID3  SVAL3
```

Because a *List Style* ULOAD is used to load the **SFAC** scale factors, this method also supports loading additional SCALE factors by using the *Generic_Input* post-keyword. This is useful for keeping conversion factors or correction factors separated from calibration scale factors.

The following is an example that includes a conversion factor of inches to meters and a correction factor of 0.99 that are multiplied by all the scale factors in MyText.txt. The product is then multiplied by the input data (the following example uses real numbers to clarify the multiplication):

```
SFAC ByWBS OPEN/CLOSE MyText.txt  0.0254  0.99
```

MyText.txt contains the following:

```
1  1.5
2  2.5
3  3.5
```

In this example in the input, 0.0377 (that is, 0.0254 × 0.99 × 1.5), is multiplied by the associated input property everywhere WBS 1 resides, 0.0629 is the multiplier everywhere WBS 2 resides, and 0.088 is the multiplier everywhere WBS 3 resides.

SFAC allows for a layered approach to scale factors that adds flexibility for calibration and distinguishes conversion factors from parameter-estimation factors. For example, the file MyText.txt could be a calibration template file that optimization software uses to modify the MF-OWHM2 input, but the other scale factors do not change because they just transform the input to the proper units.

SFAC [**DIMKEY**] ULOAD

where

SFAC	is the keyword that initiates the advanced scale factor routine. Only one **SFAC** is allowed per line
DIMKEY	is an optional keyword that indicates the number of scale factors to be read in and how they are applied. If **DIMKEY** is not present or is set to "**ALL**", then a single scale factor is read and applied to the input data. An example **DIMKEY**, from FMP, is "**ByWBS**", which indicates **NWBS** scale factors are read, and the first scale factor is applied to areas where WBS 1 is located, and the second scale factor is applied where WBS 2 is located, and so on.
ULOAD	is *Universal Loader* utility that reads the scale factors with *List Style* input. ULOAD can use an *Implied_Internal* to read the scale factors, which is recommended method for reading **SFAC** scale factors. It is not recommended to use keyword **INTERNAL** to load scale factors.

Figure 1.15. Advanced scale factor input structure and explanation of keywords.

ULOAD That Contains SFAC

Any input utility file that is loaded with a call to ULOAD also supports the **SFAC** keyword. This can be confusing because **SFAC** relies on a separate instance of ULOAD to load the scale factors. If there is a single **SFAC** in a ULOAD, the routine ULOAD is called twice; the first time loads the scale factor data, and the second time loads the input data and applies the scale factors to it.

ULOAD supports multiple calls of **SFAC**, but the keyword **SFAC** must be located before the input data and only have one **SFAC** keyword for each line. The exception to this is when reading *List Style* with *Implied_Internal*, ULOAD will check for a single **SFAC** keyword after the list records read on the same line. If an input record read by ULOAD uses a *Generic_Input* keyword (for example, **INTERNAL**, **EXTERNAL**, **OPEN/CLOSE**, **DATAUNIT**, and **DATAFILE**), then it also checks in the *Generic_Input* file for the **SFAC** keyword before loading the input data. This allows for scale factors to be applied before checking for the *Generic_Input* flags and in the *Generic_Input* file itself. In addition to this, if the *Generic_Input* is followed by a SCALE value that is written to the right of it, it is also included as a scale factor for the data.

The following examples use **SFAC** with ULOAD to load three values called Val1, Val2, and Val3 and two scale factors called SF1 and SF2. The general form, figure 1.16, of this load has one or two "**SFAC** ULOAD" to read a single scale factor and then reads one ULOAD that loads Val1, Val2, and Val3.

This example loads the scale factor SF1, then uses an *Implied_Internal* on the next line to load the three values:

```
SFAC SF1          ← Implied_Internal read of SF1
Val1 Val2 Val3    ← Implied_Internal read of Val1, Val2, and Val3
```

This causes the final input for the three values to be Val1 × SF1, Val2 × SF1, and Val3 × SF1. The next example includes two scale factors that are applied to the values read using an *Implied_Internal*:

```
SFAC SF1
SFAC SF2
Val1 Val2 Val3
```

This causes the final input for the three values to be Val1 × SF1 × SF2, Val2 × SF1 × SF2, and Val3 × SF1 × SF2. Since the input uses an *Implied_Internal*—that is, no *Generic_Input* keyword was found so the *List Style* input is read along the current line—the previous input box can be reformatted to take advantage that an *Implied_Internal* allows specifying a single **SFAC** to the right of its input. The following illustrates this:

```
SFAC SF1
Val1 Val2 Val3  SFAC SF2
```

Note that the line of text that uses an *Implied_Internal* is preloaded into memory. By default, the maximum preloaded line size is set to 700 characters of text. Assuming there are no keywords on the line, 700 characters is approximately 35 numbers that are each 20 characters long (assuming the space that separates the numbers is part of the 20 characters). Because of this preloaded line character limit, it is not recommended to use an *Implied_Internal* when input exceeds 25 numbers. If the input must use an *Implied_Internal* with a text line that exceeds 700 characters, then the post-keyword "**DIM** DIM_SIZE" (fig. 1.12 item F) can be used to set the preloaded line to a length of DIM_SIZE characters.

When the input exceeds 25 numbers, the use of *Implied_Internal* is not recommended and instead the input should explicitly supply the *Generic_Input* keyword. The following example uses the **INTERNAL** keyword to load the three variables—the same Val1, Val2, and Val3 as presented before. As presented in figure 1.17, scale factors are always loaded before the input data, but **SFAC** may be located before or after the **INTERNAL** keyword.

If the input data reside in an external file, then the **SFAC** keyword can appear before the **EXTERNAL** or **OPEN/CLOSE** keywords or in the file that is opened. In both cases, **SFAC** must appear before the actual input data.

```
SFAC SF1
OPEN/CLOSE MyFile.txt
```

MyFile.txt is as follows:

```
ID1  Val1
ID2  Val2
ID3  Val3
```

or

```
OPEN/CLOSE MyFile.txt
```

MyFile.txt is as follows:

```
SFAC SF2
ID1  Val1
ID2  Val2
ID3  Val3
```

or

```
SFAC SF1
OPEN/CLOSE MyFile.txt
```

MyFile.txt is as follows:

```
SFAC SF2
ID1  Val1
ID2  Val2
ID3  Val3
```

Because MyFile.txt is opened with *Generic_Input*, it supports the optional **SF** SCALE value. The following includes a third scale factor:

```
SFAC SF1
OPEN/CLOSE MyFile.txt SF3
```

MyFile.txt is as follows:

```
SFAC SF2
ID1  Val1
ID2  Val2
ID3  Val3
```

This would cause the final data input to be Val1 × SF1 × SF2 × SF3, Val2 × SF1 × SF2 × SF3, and Val3 × SF1 × SF2 × SF3.

The previous **SFAC** examples relied on the ULOAD *Implied_Internal* to load the scale factors, but they can also be in an external file opened with *Generic_Input*. The following are examples that have the scale factors in a separate file:

```
SFAC OPEN/CLOSE SFAC1.txt
OPEN/CLOSE MyFile.txt
```

SFAC1.txt contains the following:

```
ID1  SF1
```

MyFile.txt contains the following:

```
ID1  Val1
ID2  Val2
ID3  Val3
```

The next example illustrates the use of **DIMKEY** to load more than one scale factor. **DIMKEY** is defined by the data set that supports **SFAC** and determines how the scale factors are applied when **DIMKEY** is used. An example **DIMKEY** from FMP is **ByWBS**, which indicates there are **NWBS** scale factors, and the first scale factor is applied to any input data item that is in WBS 1, and the second scale factor is applied where WBS 2 is located, and so on. For simplicity, the following example uses the **DIMKEY** "**ByVal**" and loads three scale factors that are applied to the three input values, respectively:

```
SFAC ByVal OPEN/CLOSE SFAC3.txt
INTERNAL
ID1  Val1
ID2  Val2
ID3  Val3
```

SFAC3.txt contains the following:

```
ID1  SF1
ID2  SF2
ID3  SF3
```

This causes the final data input to be Val1 × SF1, Val2 × SF2, and Val3 × SF3. The original ULOAD can contain the unscaled data and reference a separate file (SFAC3.txt) that contains a set of scale factors. This is typically used to build a parameter estimation template by keeping the unscaled input separate from the scale factor file—such as MyFile.txt, SFAC1.txt, or SFAC3.txt—that is adjusted by the parameter estimation software.

Because **SFAC** relies on ULOAD to load the scale factors, they could technically have scale factors of scale factors of scale factors, without a limit. This is not recommended, but is presented to illustrate how the two routines are interrelated:

```
SFAC OPEN/CLOSE SFAC1.txt
INTERNAL
ID1  Val1
ID2  Val2
ID3  Val3
```

SFAC1.txt contains the following:

```
SFAC OPEN/CLOSE SFAC2.txt
ID1  SF1
```

SFAC2.txt contains the following:

```
SFAC OPEN/CLOSE SFAC3.txt
ID1  SF2
```

SFAC3.txt contains the following:

```
ID1  SF3
```

This causes the final data input to be Val1 × SF1× SF2 × SF3, Val2× SF1× SF2 × SF3, and Val3× SF1× SF2 × SF3. The following is the recommended way to obtain the identical final input (note that the inclusion of **SF** is optional):

```
INTERNAL SF SF1 SF SF2 SF SF3
ID1  Val1
ID2  Val2
ID3  Val3
```

```
SFAC ULOAD   ← Load a scale factor with ULOAD
ULOAD        ← Load input data with ULOAD
```

Figure 1.16. General structure of a ULOAD input that includes an advanced scale factor, **SFAC**. **SFAC** uses ULOAD to read the scale factors and the ULOAD on the second line reads the input to which the scale factors are applied. [ULOAD is an abbreviation for the *Universal Loader*.]

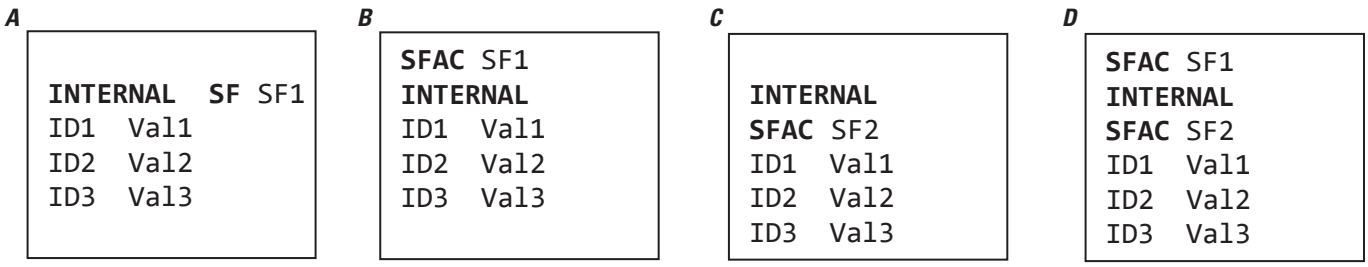


Figure 1.17. Possible locations that scale factors (**SF**) and advance scale factors (**SFAC**) can be placed within input read using ULOAD with the **INTERNAL** keyword. *A*, Input that uses the *Generic_Input* scale factor **SF** SCALE to apply SF1 to input data. *B*, **SFAC** is placed before the **INTERNAL** keyword. *C*, **SFAC** is placed after the **INTERNAL** keyword, but before the input data. *D*, One **SFAC** is placed before the **INTERNAL** keyword and another **SFAC** is placed after the **INTERNAL** keyword, but before the input data. In this case the resulting scale factor is the product of SF1 and SF2. [*Universal Loader* (ULOAD) examples use *List Style* input that reads three records identified with ID1, ID2, and ID3 being integer values of the record ID.SF1, SF2, Val1, Val2, Val3 represent actual numbers that would be used as input.]

List-Array Input Structure—Spatial-Temporal Input

This section describes the *List-Array Input* (LAI). This new MF-OWHM2 input structure facilitates initial set up as well as incorporating future updates. It also includes scale factors for calibration. LAI relies on keywords that specify the frequency of reading input and its spatial style (fig. 1.18), then reads the input data with ULOAD. The input frequency is either to load the input once and reuse it for the entire simulation, keyword **STATIC**, or to load the input every stress period, keyword **TRANSIENT**. The input style is either *List Style* (keyword **LIST**) or *Array Style* (**ARRAY**). It also offers an advanced input structure called *IXJ Style* (keyword **IXJ**), in which input depends on the package input **KEYWORD** that uses it.

All *List-Array Input* structures support specification of the **TEMPORAL_KEY**, **INPUT_STYLE**, and **INPUT** keywords (fig. 1.18). If a specific input **KEYWORD** only allows for one **TEMPORAL_KEY** or **INPUT_STYLE**, however, specifying it is optional. For example, the FMP keyword, **PRECIPITATION**—which specifies the precipitation rate over the model grid—only supports the *Array Style* input.

```
PRECIPITATION   STATIC     ARRAY OPEN/CLOSE  ./Precip.txt
PRECIPITATION   STATIC           OPEN/CLOSE  ./Precip.txt
PRECIPITATION   TRANSIENT  ARRAY OPEN/CLOSE  ./Precip_TFR.txt
PRECIPITATION   TRANSIENT        OPEN/CLOSE  ./Precip_TFR.txt
```

The **STATIC** keyword indicates that the input is read once and resides in the Precip.txt text file. This is an example of *Array Style* input data for a 3-by-5 (NROW by NCOL) model grid:

```
# File Precip.txt
# Precipitation rate in length translated to 3 by 5 Model Grid
0.50  0.68  0.81  0.75  0.72
0.55  0.72  0.82  0.77  0.82
0.52  0.73  0.83  0.79  0.92
```

In LAI, the **TRANSIENT** keyword indicates that Precip_TFR.txt is the *Transient File Reader* (TFR) file. The section "*Transient File Reader* and Direct Data Files" describes in detail the structure of a TFR.

If the input keyword only allows for one possible **TEMPORAL_KEY** and one **INPUT_STYLE**, then all the keywords are optional, and the input may be loaded with ULOAD directly. For example, the FMP keyword, **SURFACE_ELEVATION**—which specifies the initial land-surface elevation of the model grid—can only be loaded once with *Array Style*. In this case, any of the following works for loading the input from external file, DEM.txt:

```
SURFACE_ELEVATION STATIC    ARRAY OPEN/CLOSE  ./DEM.txt
SURFACE_ELEVATION STATIC          OPEN/CLOSE  ./DEM.txt
SURFACE_ELEVATION             ARRAY OPEN/CLOSE  ./DEM.txt
SURFACE_ELEVATION                 OPEN/CLOSE  ./DEM.txt
```

An example DEM.txt is as follows:

```
# DEM.txt, Land surface elevation of 3 by 5 Model Grid
0.50  0.50  0.50  0.50  0.50
0.50  0.50  0.50  0.50  0.50
0.50  0.50  0.50  0.50  0.50
```

The last line, "**SURFACE_ELEVATION OPEN/CLOSE** ./DEM.txt", directly calls ULOAD because both the **TEMPORAL_KEY** and the **INPUT_STYLE** are optional.

A special **TEMPORAL_KEY** is **CONSTANT** VALUE. The contents of VALUE must be consistent with the expected input data type (that is, integer, floating point, or text). This single value is then applied to the input keyword. **CONSTANT** automatically picks the **INPUT_STYLE** that uses the least amount of memory that is allowed. That is, if the **INPUT_STYLE** supports both **ARRAY** and **LIST**, then **CONSTANT** applies the single value as if **LIST** were selected. Conversely, if the **INPUT_STYLE** only supports the **ARRAY** keyword, then **CONSTANT** applies the single value to the array. Because **CONSTANT** is parsed by ULOAD, which uses *Generic_Input*, the use of **CONSTANT** can include the post-keyword **SF** SCALE to multiply SCALE with VALUE. The following *List-Array Input* example is equivalent to the previous example (that loaded uniform values for land surface elevation from DEM.txt) and illustrates that **CONSTANT** can produce input with spatial as well as temporal uniformity:

```
SURFACE_ELEVATION STATIC    ARRAY OPEN/CLOSE ./DEM.txt
SURFACE_ELEVATION CONSTANT 0.50
```

```
KEYWORD  TEMPORAL_KEY  INPUT_STYLE  INPUT
```

where

KEYWORD is a package input keyword (PIK) that initiates the *List-Array Input* utility.

TEMPORAL_KEY is the temporal input keyword that is set to

- **STATIC** to indicate that input data is read once with ULOAD.
- **TRANSIENT** to indicate that input data is read every stress period with either a *Transient File Reader* (TFR) or a *Direct Data File* (DDF).
- **CONSTANT** VALUE to indicate that input is a single value, VALUE, and not changed for the duration of the simulation. When using the keyword **CONSTANT**, it is optional to specify **INPUT_STYLE** and VALUE is considered the **INPUT**.

INPUT_STYLE is the spatial input keyword that is set to

- **LIST** to use *List Style* input.
- **ARRAY** to use *Array Style* input.
- **IXJ** to use *IXJ Style* input.

INPUT is the actual input data that is loaded by ULOAD, TFR, or DDF.

The input read frequency is defined by the **TEMPORAL_KEY** keyword and the spatial input style used by the **INPUT_STYLE** keyword.

Figure 1.18. *List-Array Input* (LAI) general input structure and explanation of input.

LAI[S,T,A,L] Input Format Meaning

To condense writing the options available for the *List-Array Input* format, a special short-hand notation is used (fig. 1.19). In most circumstances, the **ARRAY** keyword (A) always reads an array in the same dimension as the model grid (NROW×NCOL), and the **LIST** (L) keyword only reads one property (two columns, one for the ID and one for the property). If the **LIST** keyword loads more than one property (L-K, with K≥2), then it rarely supports the **ARRAY** keyword, because it cannot represent multiple properties across a single model-grid array. If the **ARRAY** is not the same as the model grid, then its dimension is defined by (I,J), where I is the number of rows, and J is the number of columns.

LAI[S, T, A, L] or LAI[S, T, A, L-K] or LAI[S, T, A(I,J), L]

where

LAI		indicates that input uses the *List-Array Input* structure. The contents within the brackets, [ ], indicate which keywords are supported.
S	**STATIC**	keyword is supported by the LAI input.
T	**TRANSIENT**	keyword is supported by the LAI input.
A	**ARRAY**	keyword is supported by the LAI input. *Array Style* input uses the default size, which is the model grid size (NROW, NCOL).
L	**LIST**	keyword is supported by the LAI input. *List Style* input expects a record ID and one input property. The length of the list is defined by the input that is using LAI.
L-K	**LIST**	keyword is supported by the LAI input. K is set to an integer, such as "L-4", which represents the number of properties. *List Style* input expects a record ID and K input properties. The length of the list is defined by the input that is using LAI.
A(I,J)	**ARRAY**	keyword is supported by the LAI input. I and J are the number of rows and columns, respectively, that is read using *Array Style* input.

Figure 1.19. Explanation of the *List-Array Input* (LAI) structure variants and special short-hand notation. This notation is used with model input keywords to indicate the LAI features supported by each.

The Keyword **STATIC**

The keyword **STATIC** makes use of just a single ULOAD call to read the input information and then use it for the entire simulation. For example, the following example shows how to load data for the FMP keyword **ROOT_DEPTH**—which specifies NCROP crop-root depths—with the **STATIC** keyword, and ULOAD points to the input location of root depths:

```
ROOT_DEPTH STATIC ARRAY ULOAD #Array Style Input
ROOT_DEPTH STATIC LIST  ULOAD #List Style  Input
```

For the *Array Style* input, an array the same size as the model grid is loaded by ULOAD and the crop that grows in each grid cell has the root depth specified for that row and column location. Conversely, the *List Style* would read **NCROP** rows of input with the crop ID as the record ID and the next column as the root depth of the crop. Then any grid cell where the crop ID array has crop 1 receives the root depth specified in record 1, and where the crop ID array has crop 2 receives the root depth specified in record 2, until NCROP root depths have been applied.

The following is an example for a three-by-five (NROW by NCOL) model grid with three crops (NCROP = 3) that have the root depth specified in feet, but converted to meters (0.3048 m/ft), to match the model units. It also makes use of the crop ID array, which is an integer array that specifies the locations of the crops. Note that a real simulation only allows specifying the **ROOT_DEPTH** keyword once; the two versions are represented here to illustrate the difference between **ARRAY** and **LIST**.

```
# Crop ID (LOCATION) - Array Style Input of 3 by 5 Model Grid
LOCATION STATIC ARRAY INTERNAL
   1   1   2   2   2
   1   1   2   2   3
   3   3   2   3   3
#Array Style Input 3 by 5 Model Grid
ROOT_DEPTH STATIC ARRAY INTERNAL 0.3048
   1.50  1.50  0.81  0.79  0.79
   1.50  1.50  0.81  0.81  0.50
   0.50  0.50  0.81  0.50  0.50

#List Style Input
ROOT_DEPTH STATIC LIST  INTERNAL  0.3048
   1   1.50
   2   0.81
   3   0.50
```

In this example, the *Array Style* allows for crop 2 to have two different root depths (see **green bold** numbers). The root depths read as an array are mapped cell by cell using the crop ID array allowing for crop 2 to have root depths set to 0.81 and 0.79. The *List Style* input allows for more compact input but requires crop 2 to have the same root depth for the entire model array. The final root depths applied to the model grid with *List Style* are then as follows:

```
1.50  1.50  0.81  0.81  0.81
1.50  1.50  0.81  0.81  0.50
0.50  0.50  0.81  0.50  0.50
```

This would become the following after the conversion factor (0.3048) is multiplied by the input data:

```
 0.45720   0.45720   0.24689   0.24689   0.24689
 0.45720   0.45720   0.24689   0.24689   0.15240
 0.15240   0.15240   0.24689   0.15240   0.15240
```

Note that ULOAD supports **SFAC**, by which one could specify the scale factor either before the keyword or in the external file that is opened. The following examples show three methods that can be used to apply the feet-to-meters conversion factor for the *List Style* input. Example (1), using post-keyword **SF** SCALE is this:

```
# Using Generic_Input post-keyword SF SCALE
ROOT_DEPTH STATIC LIST OPEN/CLOSE ./Root1.txt 0.3048
```

Example (2), using keyword **SFAC** outside the loaded text file is this:

```
# Using SFAC before Keyword
SFAC 0.3048
ROOT_DEPTH STATIC LIST OPEN/CLOSE ./Root1.txt
```

Where Root1.txt is as follows:

```
# Root Depth – List Style
   1   1.50
   2   0.81
   3   0.50
```

Example (3), using **SFAC** within the loaded file Root2.txt is this:

```
# Using SFAC after Keyword
ROOT_DEPTH STATIC LIST OPEN/CLOSE ./Root2.txt
```

Where Root2.txt is as follows:

```
# Scale feet to meters then
# Root Depth – List Style
SFAC 0.3048
   1   1.50
   2   0.81
   3   0.50
```

Also, additional comments can be placed anywhere:

```
# Scale feet to meters
SFAC 0.3048                          # Comment Here
# Root Depth – List Style
# Crop 1
  1    1.50     # A Comment Here
  #
  # Crop 2
  #
  2    0.81        # A Comment Here
  # Crop 3
  3    0.50        # A Comment Here
```

Although the **STATIC** keyword relies on a single ULOAD, the **TRANSIENT** keyword relies on the *Transient File Reader* to load the input for each stress period. The next section discusses this in detail.

Transient File Reader and Direct Data Files

The *Transient File Reader* (TFR) and *Direct Data File* (DDF) file types are used by the *List-Array Input* (LAI) style to load temporally varying (**TRANSIENT**) input data. The TFR can be thought of as a special spatial-temporal input format (for example, reading precipitation arrays every stress period). The TFR and DDF are both opened in the LAI structure at the **INPUT** keyword with *Generic_Input_OptKey* and must be specified as a separate file from the one that contains the LAI keywords. This precludes opening a TFR and DDF with the keywords **INTERNAL** or using an *Implied_Internal*.

The *Generic_Input_OptKey* file-opening keywords indicate if the temporal file is a TFR or DDF. Specifically, if **DATAFILE** or **DATAUNIT** open the file, then it is a DDF, whereas the rest of the keywords open a TFR.

Input from a TFR or DDF and the files that they open are preloaded, one line at a time, into memory and parsed for keywords, *List style* input, and an *Implied_Internal* input line. For most situations, this feature can be ignored by most users—since most input is loaded with a *Generic_Input* keyword—such as, **INTERNAL**, **EXTERNAL**, **OPEN/CLOSE**, **DATAUNIT**, and **DATAFILE**, which do not exceed the preloaded line size. The memory reserved for the preloaded line is 700 characters by default, which is roughly 35 numbers that are each 20 characters long. All input from a TFR or DDF—except for reading an *Array Style* array—are loaded this way. If the input fails to load a number on a line, then an error is raised signifying either not enough numbers on the line or the preloaded line size is not large enough. This can likely occur when reading *List Style* input as an *Implied_Internal*; this is because the records are all written on one line rather than one record per line. To change the maximum size of the preloaded line, the TFR or DDF that is opened with *Generic_Input_OptKey* support the post-keyword option **DIM** DIM_SIZE, which makes the preloaded line size equal to DIM_SIZE. Although it is uncommon to exceed 700 characters in a line of a file, this post-keyword is a useful add-in response to an error being raised because the input exceeds the default DIM_SIZE. Another benefit of defining DIM_SIZE is that if the input is *Array Style* or is *List Style* that never uses *Implied_Internal*, then the preloaded line size can be reduced to the largest line—excluding lines that contain the actual array—to save memory.

Transient File Reader

The *Transient File Reader* (TFR) is a pointer file that contains a ULOAD on each row that reads input per time interval it applies to. The time interval depends on the keyword that is using the ULOAD, which is typically once per stress period. Because the TFR uses ULOAD to read each temporal input, it supports all features of ULOAD, such as comments, multiple scale factors, and *Implied_Internal* (for *List Style* input with one property). Figure 1.20 presents the general format of a *Transient File Reader* file.

The next set of examples uses the previous example for root depth (FMP keyword **ROOT_DEPTH**) and changes its **TEMPORAL_KEY** to **TRANSIENT**. The set of keywords then becomes this:

```
# Generic_Input_OptKey opens TFR.
# A TFR cannot be opened with INTERNAL, DATAFILE or DATAUNIT
ROOT_DEPTH TRANSIENT ARRAY Generic_Input_OptKey # TFR reads Array Style input
ROOT_DEPTH TRANSIENT LIST  Generic_Input_OptKey # TFR reads List Style input
```

The following examples are TFR files that specify and load the input every stress period (SP) for 10 stress periods using different ULOAD keywords that open the files defined as Root1.txt and Root2.txt in the previous section (fig. 1.21). Note that although this example uses *List Style* input, the *Array Style* works in the same manner and only differs in the input data structure. Comments are included in the sample TFR file to explain how the data are loaded.

Because the file remains open after reading with the keywords **DATAFILE**, **EXTERNAL**, and **DATAUNIT**, multiple sets of input can be specified in the same file. This allows the user to work with fewer files and have the TFR control the input order and, optionally, the scale factors. Figure 1.22 is an example of a TFR that uses **DATAFILE** to cycle through three stress periods of input.

By combining **DATAFILE**, **EXTERNAL**, and **DATAUNIT** with **REWIND**, a single file could contain a repeated block of stress-period input that is cycled through. For example, root depths could vary through the year, but are typically the same depth at the same time of the year. If a model contained stress periods that align with the months of the year, then a file Root12.txt (fig. 1.23*A*) would contain 12 sets of input for each month's root depth. The structure of the input depends on the input style, where *List Style* input expects a root depth for each crop and array style reads a model grid array that is the root depth for each model row and column. Figure 1.23*A* presents the root depth raw input using *List Style* for three crops. This raw input is then cycled through using the **DATAFILE** directive (fig. 1.23*B*). Another method of cycling through a file is using the **LOAD_NEXT** and **RELOAD** directives (fig. 1.23*C*).

Using an annual file that is cycled through each year provides an easy method for maintaining input datasets in a compact manner that is easy to understand. For more information on cycling through a TFR file, please see the "*Transient File Reader* – Spatial-Temporal Input" in appendix 2.

If one file is cycled through for the entire simulation by using only **DATAFILE**, **DATAUNIT**, or **EXTERNAL** (that is, no **REWIND**), then the *Transient File Reader* can be bypassed and loaded as a *Direct Data File* (DDF)—discussed in the next section.

```
ULOAD   # First     Load of Input
ULOAD   # Second    Load of Input
ULOAD   # Third     Load of Input
  ⋮
ULOAD   # Nth       Load of Input
```

Figure 1.20. General input structure of a *Transient File Reader* (TFR) file. If the TFR input occurs once per stress period, then this example represents reading the input for the first N stress periods. Each uncommented, non-blank line in the file is expected to load the input needed for that stress period. Input for each stress period is read with the *Universal Loader* (ULOAD). That is, the first ULOAD reads input for stress period 1, and the second ULOAD reads input for stress period 2, and so forth. [Comments are any text that are written to the write of a "#" symbol.]

```
# Transient File Reader Example that loads 10 Stress Periods (SP)
INTERNAL 0.3048 # SP1
   1   1.50
   2   0.81
   3   0.50

REPEAT  SF 0.3048    # SP2: Reuse previous input, multiply it by 0.3048
                     # Note the keyword "SF" is optional, so "REPEAT  0.3048" works.
#
# SP3: Implied Internal - SFAC must be used instead of "SF SCALE" or "SCALE"
#
SFAC 0.3048
1.50  0.81  0.50
#
# SP4:  Open and then close Root2.txt and multiply its input by 0.3048
OPEN/CLOSE ./Root2.txt    SF 0.3048
#
EXTERNAL 55    # SP5: Unit 55 is Root2.txt that is open in the NAME file
#
# Note that keywords DATAFILE and DATAUNIT work within the TFR
# Root2.txt remains open for the rest of the simulation and is buffered into 64 kb of RAM
#
DATAFILE ./Root2.txt  SF 0.3048 BUFFER 64 # SP6: Load from first line of Root2.txt
#
DATAFILE ./Root2.txt  SF 0.3048  # SP7:  Load input from current line of Root2.txt
                                 #       The current line is the next text file line
                                 #       from where the SP6 input load ended.

# SP8:  Move to first line of Root2.txt, load input, and apply SCALE (multiply by 0.3048)
RELOAD  0.3048        # Note the use of "SF" is optional

# Unit 55 is Root2.txt, but it was read already once
# so a second call to the unit with EXTERNAL or DATAUNIT
# would cause an end of file error, so the open file must be rewound back to the first line.
#
EXTERNAL  55 REWIND # SP9 move to line 1 of file, then load input
DATAUNIT  55 REWIND # SP10 Same effect as EXTERNAL 55
```

Figure 1.21. Example *Transient File Reader* (TFR) file that loads input for 10 stress periods (SP) using various TFR directives. Input is assumed to be *List Style* that reads 3 records. Comments are used to explain what each directive is doing and what is being read. [Comments are any text that are written to the right of a "#" symbol.]

A

```
# Transient File Reader file that reads from Root2.txt input for three stress periods
# Each stress period's input is scaled by 0.3048
# (Note that "SF 0.3048" could be used in the place of "0.3048")
#
DATAFILE ./Root2.txt 0.3048 # SP1: open Root2.txt, load input from first line
DATAFILE ./Root2.txt 0.3048 # SP2: Root2.txt already open, load at current file location
DATAFILE ./Root2.txt 0.3048 # SP3: continue loading input from Root2.txt
```

B

```
# Root2.txt
   1   1.50  # SP 1 Root Depth – List Style
   2   0.81
   3   0.50
# SP 2        -- Note that comment does not have to be on separate line
   1   1.50
   2   0.81
   3   0.50
# SP 3
   1   1.50
   2   0.81
   3   0.50
```

Figure 1.22. Example *Transient File Reader* (TFR) file that loads input for three stress periods (SP) using the DATAFILE directive and *List Style* input that expects 3 records. Comments are used to explain what each directive is doing and what is being read. *A*, The TFR file that opens and reads input from Root2.txt once per SP. *B*, The input file Root2.txt that is read from. In the file, the comments are optional and used to separate, visually, each stress period's input. [Comments are any text that are written to the right of a "#" symbol.]

A

```
# Root12.txt
#  ID    Root_Depth
   1    0.20   #SP -- January
   2    0.21
   3    0.10
   1    0.60   #SP -- February
   2    0.71
   3    0.30
   1    1.50   #SP -- March
   2    0.81
   3    0.50

   1    1.50   #SP -- November
   2    0.81
   3    0.50
   1    0.10   #SP -- December
   2    0.15
   3    0.05
```

B

```
DATAFILE  ./Root12.txt  0.3048           # SP1   January
DATAFILE  ./Root12.txt  0.3048           # SP2   February
DATAFILE  ./Root12.txt  0.3048           # SP3   March
DATAFILE  ./Root12.txt  0.3048           # SP4   April
DATAFILE  ./Root12.txt  0.3048           # SP5   May
DATAFILE  ./Root12.txt  0.3048           # SP6   June
DATAFILE  ./Root12.txt  0.3048           # SP7   July
DATAFILE  ./Root12.txt  0.3048           # SP8   August
DATAFILE  ./Root12.txt  0.3048           # SP9   September
DATAFILE  ./Root12.txt  0.3048           # SP10  October
DATAFILE  ./Root12.txt  0.3048           # SP11  November
DATAFILE  ./Root12.txt  0.3048           # SP12  December
DATAFILE  ./Root12.txt  0.3048  REWIND   # SP13  January - Rewind to top of file
DATAFILE  ./Root12.txt  0.3048           # SP14  February
DATAFILE  ./Root12.txt  0.3048           # SP15  March
```

Figure 1.23. Example *Transient File Reader* (TFR) file that cycles through a file that contains 12 months of root depth input for three crops (*List Style* input with three records). In this example, the TFR time interval is a Stress Period (SP) defined equivalently with the months of the year. For example, SP 1 and 13 represent January and SP 2 and 14 represent February. *A*, The raw input file, Root12.txt, which contains the root depth for three crops for each month of the year. If a model only used between 1 and 12 Stress Periods, then Root12.txt could also be used opened as a *Direct Data File* (DDF). *B*, An example TFR file that uses the **DATAFILE** directive to open and read input from Root12.txt for each SP. *C*, An example TFR file that uses the **DATAFILE** directive to open and read input from Root12.txt initially and then uses the **LOAD_NEXT** and **RELOAD** directives to cycle through the file for each SP. [Comments are any text that are written to the right of a "#" symbol. The TFR examples use a scale factor, SCALE = 0.3048 that multiplies with the root depth to convert the depths from feet to meters.]

C

```
DATAFILE  ./Root12.txt  0.3048          # SP1   January
LOAD_NEXT   0.3048                      # SP2   February
LOAD_NEXT   0.3048                      # SP3   March
LOAD_NEXT   0.3048                      # SP4   April
LOAD_NEXT   0.3048                      # SP5   May
LOAD_NEXT   0.3048                      # SP6   June
LOAD_NEXT   0.3048                      # SP7   July
LOAD_NEXT   0.3048                      # SP8   August
LOAD_NEXT   0.3048                      # SP9   September
LOAD_NEXT   0.3048                      # SP10  October
LOAD_NEXT   0.3048                      # SP11  November
LOAD_NEXT   0.3048                      # SP12  December
RELOAD      0.3048                      # SP13  January - Rewind to top of file
LOAD_NEXT   0.3048                      # SP14  February
LOAD_NEXT   0.3048                      # SP15  March
```

Figure 1.23. —Continued

Direct Data File

The *Direct Data File* (DDF) is opened when the LAI uses the *Generic_Input* keyword **DATAFILE** or **DATAUNIT**. Functionally, the DDF acts identically to a *Transient File Reader* (TFR) file that only contains **DATAFILE**, **DATAUNIT**, or **EXTERNAL** references with no **SFAC** or post-keywords (for example, **SF SCALE**, **REWIND**). The DDF allows a shortcut for accessing an input file without having to double-specify keywords. A DDF is the closest analogue to how input was loaded by previous versions of FMP and can help with transforming previous FMP inputs to the current input design. To translate such an older structure to a LAI structure, the same **EXTERNAL** file could be opened with **DATAUNIT** as with the LAI **INPUT** keyword.

The following example presents a TFR and its equivalent DDF using the previously discussed root depth examples with *List Style* input for three crops and for four stress periods; the base text file Root3.txt is shown first:

```
# Root3.txt – Stress period input is defined as contiguous blocks – no keywords
   1   1.0       # SP 1 Root Depth – List Style
   2   0.8
   3   0.5
   1   1.1       # SP  2
   2   0.9
   3   0.6
   1   1.2       # SP 3
   2   0.95
   3   0.7
   1   1.3       # SP 4
   2   0.99
   3   0.8
```

Next, the FMP input keyword **ROOT_DEPTH** uses keywords to indicate *List-Style* input lines will be loaded using the *Transient File Reader* (TFR) file, Root_TFR.txt, opened as is shown:

```
# Generic_Input_OptKey opens TFR.
# A TFR cannot be opened with INTERNAL, DATAFILE or DATAUNIT
ROOT_DEPTH TRANSIENT LIST OPEN/CLOSE ./Root_TFR.txt # Reads List Style input
```

```
# Root_TFR.txt
DATAFILE ./Root3.txt 0.3048    # SP1, Open Root3.txt, read from first line
DATAFILE ./Root3.txt 0.3048    # SP2, continue reading from Root3.txt
DATAFILE ./Root3.txt 0.3048    # SP3, continue reading from Root3.txt
DATAFILE ./Root3.txt 0.3048    # SP4, continue reading from Root3.txt
```

Alternatively, Root_TFR.txt could use **EXTERNAL** by declaring Root3.txt in the NAME file:

```
# Root_TFR.txt, Unit 56 is Root3.txt opened in the NAME file
EXTERNAL 56  0.3048    # SP1, read from unit 56 (Root3.txt)
EXTERNAL 56  0.3048    # SP2, read from unit 56 (Root3.txt)
EXTERNAL 56  0.3048    # SP3, read from unit 56 (Root3.txt)
EXTERNAL 56  0.3048    # SP4, read from unit 56 (Root3.txt)
```

This LAI structure using a TFR could be translated into one that opens a DDF by using the keyword **DATAFILE** or **DATAUNIT** as part of the LAI **INPUT**. The following example directly accesses Root4.txt and bypasses the TFR:

```
# Direct Data File:  Is only opened with DATAFILE or DATAUNIT
#
ROOT_DEPTH TRANSIENT LIST DATAFILE ./Root3.txt 0.3048 # Open as DDF
#
# or if Root3.txt is opened in the NAME file on Unit 56
#
ROOT_DEPTH TRANSIENT LIST DATAUNIT 56 0.3048
```

The limitation of the DDF is it does not allow for easy specification of time-varying scale factors. The only way a scale factor can vary in a DDF is by specifying it with the **SFAC** keyword before each stress-period input. The following example file illustrates the only method for applying a temporally varying scale factor when input is opened as a DDF. This example reads the input for four stress periods and applies the scale factors 1.1, 1.2, 1.3, and 1.4, to stress periods 1, 2, 3, and 4, respectively.

```
SFAC 1.1          # SFAC applied to first read of DDF
   1   1.0        # SP 1 Root Depth – List Style
   2   0.8
   3   0.5
SFAC 1.2          # SFAC applied to second read of DDF
   1   1.1        # SP 2
   2   0.9
   3   0.6
SFAC 1.3
   1   1.2        # SP 3
   2   0.95
   3   0.7
SFAC 1.4
   1   1.3        # SP 4
   2   0.99
   3   0.8
```

IXJ Style Input—Advanced Structured Input

The *IXJ Style* input is an advanced input that serves as a surrogate to the *List* and *Array Style* inputs. The *IXJ Style* has been intentionally left out of the previous descriptions to prevent confusion with the recommended input structures (**LIST** and **ARRAY**). It is not necessary to use the *IXJ Style* input structure as part of the standard *List-Array Input*. It is documented here for completeness and to provide examples of the benefits of using it. *IXJ Style* input is loaded using ULOAD, so the *IXJ Style* supports all of the ULOAD keywords and scale factors (**SFAC**). The name " IXJ " is a reference to the loading of an arbitrary number of lines of input in which each line has a prespecified set of integers (I), followed by a set of floating-point numbers (X), and finally by a second set of integers (J). The number of integers (I and J) and floating-point numbers (X) that are read on each line depends on the input keyword that loads them. The loading of *IXJ Style* input continues until it either reaches the end of the file or encounters the keyword **STOP IXJ**.

The dimensions I, X, and J for *IXJ Style* input are either explicitly defined or specified with a shorthand notation. The shorthand notation has the structure of IXJ[DIM_I, DIM_X, DIM_J], where DIM_I, DIM_X, DIM_J are set to a number that represents the number of integers read for I, floating-point real numbers read for X, and integers read for J, respectively. For example, IXJ[3, 1, 0] indicates that the *IXJ Style* input expects to read on each row three integers (I), one floating-point (X), and zero integers (J). The shorthand can optionally exclude DIM_J, which indicates its value is zero. Using this option would shorten the previous example to IXJ[3, 1].

A common use of the *IXJ Style* is to specify *Array Style* using a compressed coordinate structure in which each line contains the row number, the column number, and the floating-point value to assign at that row and column location. Any row and column location that is not defined in the *IXJ Style* is set to zero (in fact, the array is initialized to zero, and then the *IXJ Style* input overwrites each specified row and column location with the assigned value). Using *IXJ Style*, the compressed coordinate structure would read two values for `I`, one value for `X`, and zero values for `J`.

As an example of input using the *IXJ Style* as surrogate for *Array Style*, reconsider the previously used **`ARRAY`** input for **`PRECIPITATION`** (shown in the example following fig. 1.18). The following is an example use of `LAI` for precipitation input using *Array Style* where Precip.txt (fig. 1.24*E*) contains the *Array Style* precipitation data:

```
PRECIPITATION    STATIC     ARRAY OPEN/CLOSE   ./Precip.txt
```

The same data can be loaded as *IXJ Style* input by translating the precipitation array to *IXJ Style* (fig. 1.24*B*) and changing the input style keyword from **`ARRAY`** to **`IXJ`**. This would change the previous example as follows:

```
PRECIPITATION    STATIC     IXJ OPEN/CLOSE   ./PrecipIXJ.txt
```

Figure 1.24 presents a precipitation *List Array Input* example that uses the **`TRANSIENT`** keyword and *IXJ Style* input (fig. 1.24*A*). Figure 1.24*B* is the `TFR` that reads two stress periods; the first stress period uses **`INTERNAL`** directive to load five lines of *IXJ Style* input, and the second stress period uses **`OPEN/CLOSE`** directive to read the *IXJ Style* input from the file PrecipIXJ.txt (fig. 1.24*C*). If a row and column is not specified, then *IXJ Style* automatically assumes a value of zero for precipitation. The final precipitation arrays for the two stress periods are presented in figure 1.24*D* and *E*.

The *IXJ Style* is suited best for input of the land-use area fractions if there are multiple land-use types allowed in one model cell. This advanced feature in the FMP allows more than one land use type for each model cell. The input keyword that specifies crop-area fractions is **`LAND_USE_AREA_FRACTION`**, and it expects *Array Style* input that reads multiple `NROW` by `NCOL` arrays (one per Crop) of fractions of the cell area. Figure 1.25*A* is an example that loads such arrays for a three-by-five (`NROW` by `NCOL`) model grid's crop-area fractions for three crops (`NCROP = 3`). The equivalent *IXJ Style* input (fig. 1.25*B*) reads three integers and one floating-point number. The three integers are crop ID, row, and column, and the floating-point number is the fractional cell area applied to the specified crop ID at that model grid row and column. The advantage of using *IXJ Style* is that only the non-zero fractions must be specified. In this example, the difference between the *IXJ* and *Array Styles* is trivial, but the advantage can be quite substantial for large scale models.

A

```
PRECIPITATION   TRANSIENT IXJ OPEN/CLOSE ./PrecipIXJ_TFR.txt
```

B

```
# PrecipIXJ_TFR.txt
INTERNAL                        # SP1
#ROW  COLUMN  PRECIP
 1     3       0.81
 2     1       0.55
 2     2       0.72
 2     4       0.77
 3     4       0.79
STOP IXJ                        # Keyword that ends loading IXJ input for stress period
#
OPEN/CLOSE PrecipIXJ.txt # SP2 loads from file PrecipIXJ.txt
```

C

```
# File: PrecipIXJ.txt
#ROW  COLUMN  PRECIP
 1     1       0.50
 1     2       0.68
 1     3       0.81
 1     4       0.75
 1     5       0.72
 2     1       0.55
 2     2       0.72
 2     3       0.82
 2     4       0.77
 2     5       0.82
 3     1       0.52
 3     2       0.73
 3     3       0.83
 3     4       0.79
 3     5       0.92
#
# End of file indicates termination of input;
# or could use keyword
# STOP IXJ
```

Figure 1.24. *A*, Example package input keyword, **PRECIPITATION**, that uses *List-Array Input* with IXJ Style to specify for two stress periods the precipitation rate for a 3 by 5 model grid. *B*, Example *Transient File Reader* file, PrecipIXJ_TFR.txt. The first stress period is loaded with the **INTERNAL** directive and reads on each uncommented, non-blank line, the row and column number of the model grid and the precipitation rate that is assigned to it. *IXJ Style* input continues to read lines until encountering the keyword "STOP IXJ". The second stress period loads the *IXJ Style* input from the file PrecipIXJ.txt. *C*, The file PrecipIXJ.txt, that contains *IXJ Style* input, is read until the end of file is reached. *D*, The resulting array that MF-OWHM2 uses after reading the *IXJ Style* input specified by the **INTERNAL**. *E*, The resulting array that MF-OWHM2 uses after reading PrecipIXJ.txt with *IXJ Style* input. [Comments are any text that are written to the right of a "#" symbol.]

D

```
# Resulting Precipitation array from IXJ Style read with INTERNAL
# Precipitation rate in length translated to 3 by 5 Model Grid
0.00  0.00  0.81  0.00  0.00
0.55  0.72  0.00  0.77  0.00
0.00  0.00  0.00  0.79  0.00
```

E

```
# File Precip.txt
# Resulting Precipitation array from IXJ Style read of PrecipIXJ.txt
# Precipitation rate in length translated to 3 by 5 Model Grid
0.50  0.68  0.81  0.75  0.72
0.55  0.72  0.82  0.77  0.82
0.52  0.73  0.83  0.79  0.92
```

Figure 1.24. —Continued

A

```
#Array Style Input 3 by 5 Model Grid and NCROP=3
LAND_USE_AREA_FRACTION STATIC ARRAY INTERNAL
#                                  CROP 1
0.5  0.0  0.0  0.0  0.0
0.2  0.0  0.0  0.0  0.2
0.0  0.0  0.3  0.0  0.0
#                                  CROP 2
0.0  0.8  0.0  0.0  0.0
0.8  0.8  0.0  0.0  0.0
0.0  0.0  0.3  0.0  0.0
#                                  CROP 3
0.0  0.0  0.0  0.0  0.0
0.0  0.0  0.0  0.5  0.5
0.0  0.0  0.3  0.5  1.0
```

B

```
#IXJ Style Input 3 by 5 Model Grid and NCROP=3
LAND_USE_AREA_FRACTION STATIC IXJ INTERNAL
#Crop  Row  Col    Fraction
  1     1    1     0.5
  1     2    1     0.2
  1     2    5     0.2
  1     3    3     0.3
  2     1    2     0.8
  2     2    1     0.8
  2     2    2     0.8
  2     3    3     0.3
  3     2    4     0.5
  3     2    5     0.5
  3     3    3     0.3
  3     3    4     0.5
  3     3    5     1.0
STOP IXJ
```

Figure 1.25. Examples of *List-Array Input* for the FMP Land_Use block using input keyword **LAND_USE_AREA_FRACTION** to *A*, load an *Array Style* input for 3 crops and a model grid that is 3 rows by 5 columns. This particular input expects to read an array that is 9 rows (3 sets of 3) by 5 columns, where the first 3 by 5 array represents the fraction of each cell that crop 1 is planted, and the second 3 by 5 array is for crop 2, and so forth; and *B*, loads an *IXJ Style* input equivalent to the *List-Array Input* of part *A*. [Comments are any text that are written to the right of a "#" symbol. Comments are optional and included to help organize the example.]

Lookup Table Input Structure

The Lookup Table input structure reads *Lookup Style* input; that is, it uses lookup tables that are composed of pairs of data—the lookup value and its associated return value. In all lookup tables, the lookup values should be sorted in ascending order, but if they are not, then MF-OWHM2 sorts the lookup table rows to ensure ascending order. An individual lookup table is read once (for example, a single stream stage-discharge table), but may be used for different lookup values or during different times of a simulation (such as evaluating the discharge rate given a stream stage at the start of each stress period). The *Lookup Style* input within a `LAI` currently only supports the **`STATIC`** **`TEMPORAL_KEY`** keyword—that is, `LAI` reads all the lookup tables once and reuses them for the entire simulation.

In MF-OWHM2, a lookup table is optimized for fast searches within small tables—less than 1,000 rows of data—with basic interpolation methods. There is no limit to the size of a lookup table, but it is recommended to resample the table to make it smaller if it contains redundant or unnecessary information.

A lookup value may be specified as a date (using any of the date formats described in appendix 2), but MF-OWHM2 internally converts each value in the lookup column to an equivalent decimal year that is treated as a regular floating-point number, and the specified lookup value also is converted likewise for comparison. For example, if the lookup value is specified as 1979-4-23, 1979-4, or APR-23, these are converted to `1979.307`, `1979.247`, and `0.307`, respectively. Note that if the day of month is not specified it is assumed to be 1. If the year is not specified, then it is assumed to be zero. Figure 1.26 is an example lookup table composed of nine rows.

There are four methods available for determining the return value for a given search value to seek among the lookup values. The method is set at the time of loading the lookup table and cannot change during a simulation. The first method, signified by **`METHOD`** keyword **`NEAREST`**, searches for closest lookup value and returns the paired return value. If the search value is equidistant from two lookup values, then the larger lookup value is selected. The second method, signified by **`METHOD`** keyword **`STEP_FUNCTION`**, is a step function that searches for the closest lookup value that is less than or equal to the search value. For the **`STEP_FUNCTION`** method, if the search value is less than the first lookup value then the first return value is used. The third option, signified by **`METHOD`** keyword **`INTERPOLATE`**, linearly interpolates a return value based on the lookup values bracketing the search value. If the search value is less than the smallest lookup value in the table then the linear interpolation uses the first two rows of values to extrapolate a return value. Similarly, if the search value exceeds the table's largest lookup value, then the last two rows are used to extrapolate. The fourth method, signified by **`METHOD`** keyword **`CONSTANT`**, is a special case that forces the table to always return the same value. Figure 1.27 presents an example Lookup Table and the results from three different methods for given search values.

A set of lookup tables can be loaded with `ULOAD`, which allows lookup tables to be loaded with `LAI`; however, limitations exist—only the `LAI` keyword **`LIST`** is supported for lookup tables. Specifically, MF-OWHM2 only supports *List Style* input that reads one lookup table for each record. The Lookup Table input structure (*Lookup Style* input, fig. 1.28) is composed of three parts, the first being the **`METHOD`** keyword that indicates how the lookup table returns a value—**`NEAREST`**, **`STEP_FUNCTION`**, **`INTERPOLATE`**, or **`CONSTANT`**. The second part, `NTERM`, is the number of rows in the lookup table, and the last part is a *Generic_Input* file identifier that points to where the lookup table is. A scale factor, **`SF`** `SCALE`, maybe optionally specified after the *Generic_Input*. At the start of the *Generic_Input* file the keyword **`SFAC`** is supported, but it does not support any **`DIMKEY`** keywords—that is, it only supports loading a single scale factor. If a scale factor is read, then it is only applied to the return value of the lookup table (the second column). Figure 1.28 is a formal description of *Lookup Style* input.

Figure 1.29 is an example package input **`KEYWORD`** that uses *List Style* input to read a list of three record ID's that use *Lookup Style* input. The lookup table associated with record ID 1 is loaded from the file TAB.txt and specifies that it will find return values using the **`NEAREST`** method. Because `NTERM` is set to zero, the number of rows in the table is automatically determined during reading of TAB.txt. The table associated with record ID 2 will use the **`INTERPOLATION`** lookup method and contains 4 rows. The **`INTERNAL`** keyword indicates that the lookup table is loaded on the subsequent lines. The lookup table associated with record ID 3 has the lookup method set to **`CONSTANT`**, so it will always return 0.5 for all search values.

```
# LookUp  ReturnValue
  1.0      19.0
  2.0      66.0
  5.0     -91.0
 10.0     -82.0
 12.0      17.0
 12.4     -57.0
100.0     -75.0
123.0      23.0
956.0      91.0
```

Figure 1.26. Example lookup table used by the Lookup Table input structure (*Lookup Style*). The first column of numbers are the lookup values and the second column of numbers are the associated return values.

A

```
# LookUp  ReturnValue
   10.0      0.0
   20.0      2.0
   30.0      6.0
   40.0     14.0
   50.0     16.0
```

B

	Value Returned by **METHOD**		
Search Value	**NEAREST**	**STEP_FUNCTION**	**INTERPOLATE**
5	0	0	–1.0
14	0	0	0.8
15	2	0	1.0
16	2	0	1.2
29	6	2	5.6
30	6	6	6.0
31	6	6	6.8
55	16	16	17.0

Figure 1.27. Summary of Lookup Table input (*Lookup Style*) examples for various search values given three of the available lookup methods: *A*, example lookup table; *B*, return values obtained by each method for the given set of search values; graphs showing lookup table's returned values using the *C*, **NEAREST** method, *D*, **STEP_FUNCTION** method, and *E*, **INTERPOLATE** method.

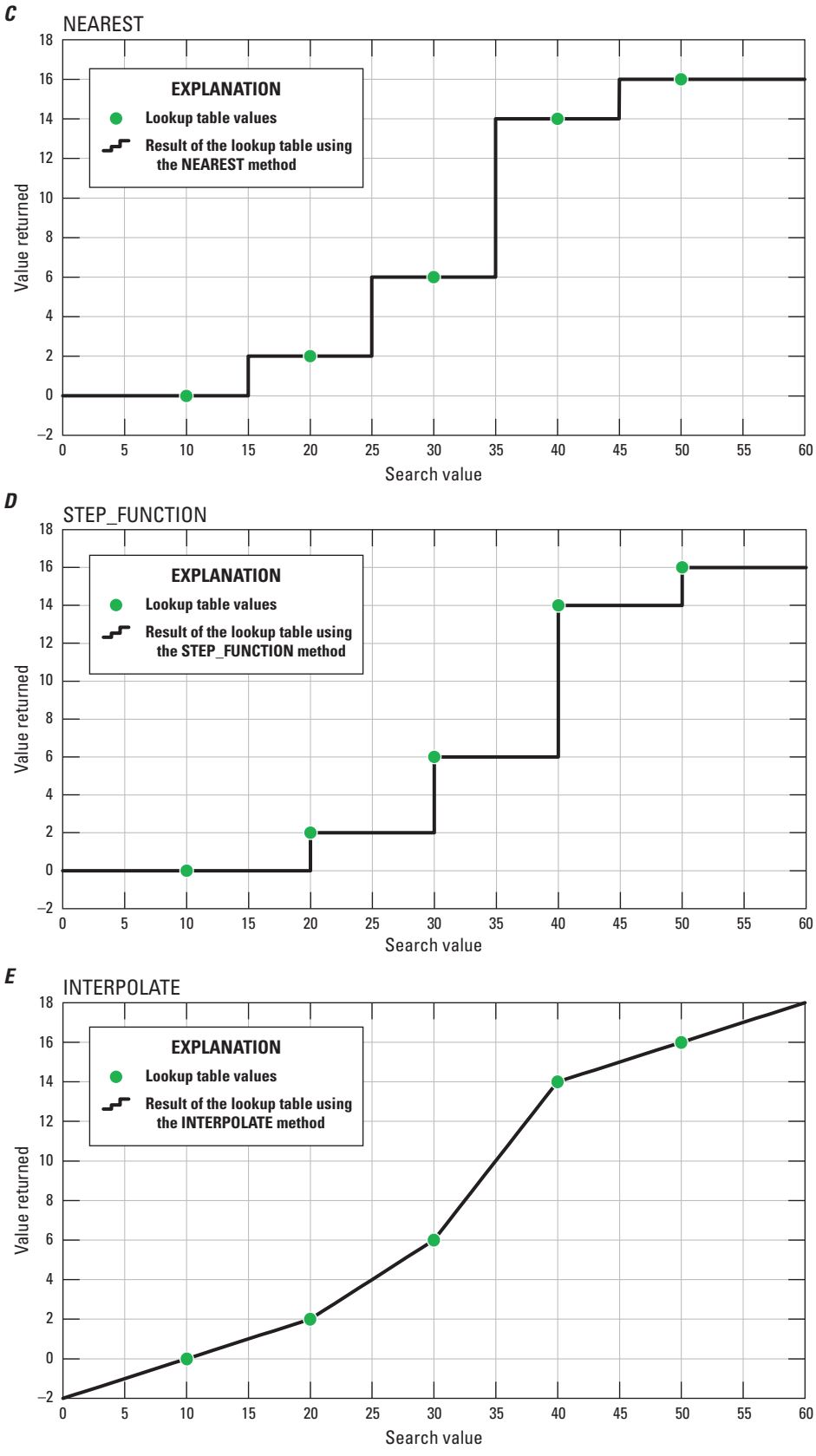


Figure 1.27. —Continued

METHOD NTERM *Generic_Input_OptKey*

where

METHOD defines the method that is used for determining the return value based on the relationship between the search value and the lookup value. It must be set to one of the following keywords:

- **NEAREST** the nearest lookup value determines the return value.
- **STEP_FUNCTION** the nearest smaller or equal lookup value determines the return value.
- **INTERPOLATE** linearly interpolates between the lookup values to the search value to determine the return value.
- **CONSTANT** VALUE declares that lookup table is composed of a single number that is returned for all lookup values. VALUE must be specified after the keyword **CONSTANT** and is the number that will be returned. Because they are not needed, NTERM and *Generic_Input* are not read.

NTERM is the number of rows in the lookup table. If set to a negative value or zero, then the lookup table must reside in a separate file and its size is automatically determined. The length is determined based on the number of successfully loaded, uncommented, rows in the table (namely the bottom of the file is reached or there is a non-commented line that fails to load).

Generic_Input_OptKey is the input file that contains the lookup table.

If there is a scale factor, SCALE, specified then it is only applied to the return values (second column).

Figure 1.28. General input structure for *Lookup Style* input, which loads a single lookup table and specifies the lookup method by which return values will be associated with a search value.

A

```
#
KEYWORD STATIC LIST INTERNAL
#ID Lookup-Style
#ID METHOD   NTERM GENERIC_INPUT
1   NEAREST      0      TAB.txt   # Load table in TAB.txt; auto-count rows
2   INTERPOLATE  4     INTERNAL   # Table with 4 rows on subsequent lines
                      10    0
                      20    2
                      30    8
                      40   14
3   CONSTANT     0.5              # Table always returns 0.5
```

B

```
# File: TAB.txt
# LookUp   ReturnValue
  5.0        25.0     # First Value
  8.0        50.0
 10.0        75.0
        # Comments can be anywhere
 15.0        80.0
 50.0        98.0
# End of file determines table size is 5 = NTERM
```

Figure 1.29. Example using *Lookup Style* input. The shorthand notation is **KEYWORD** LAI[S, L] using *Lookup Style*, where **KEYWORD** represents the package input keyword that supports lookup tables. The *List Style* input reads three lookup tables (three records) that each define the lookup method and lookup table location. *A*, The input for **KEYWORD** LAI[S, L] using *Lookup Style*. *B*, The lookup table specified in part *A* as the TAB.txt file.

References Cited

Harbaugh, A.W., 2005, MODFLOW-2005—The U.S. Geological Survey modular ground-water model—The ground-water flow process: U.S. Geological Survey Techniques and Methods 6–A16, variously paginated, https://pubs.usgs.gov/tm/2005/tm6A16/.

Appendix 2. Separation of Spatial and Temporal Input Options

One of the concepts used in MODFLOW One-Water Hydrologic Flow Model (MF-OWHM2) is the separation of spatial input from temporal input. This concept was introduced with TabFiles, which are time-series-like input linked to a spatial location—for example a General Head Boundary (GHB) cell can have its boundary head (`BHEAD`) set to values specified in a TabFile (time-tabulated input). With this release of MF-OWHM2, the TabFile functionality has been expanded; a new type of time-series file input is introduced, called a Time Series File (`TSF`); and a new alternative input format called LineFeed is included. Another new feature is the *Transient File Reader* (`TFR`), which is described in detail in appendix 1. The advantages of using the `TFR` during calibration are discussed in this appendix.

TabFiles have been further improved by including a new set of keywords that alter their function. The most notable addition is that the "Expression Parser" was linked to the TabFile, allowing the time-series value to be passed to a custom function that translates it. An example TabFile expression is "`5*TAB + LOG(TIM)`", where `TAB` and `TIM` are automatically set to the TabFile's value and current time step time, respectively. The Expression Parser is described in detail in Hanson and others (2014). This makes it easier to have one TabFile represent multiple features that have a nonlinear relationship to each other in a "one-to-many" association—the one being the TabFile and the many being the multiple model features it is connected too. An example of this is a TabFile containing sea-level gage information that is then translated to its freshwater-level equivalent for a series of GHB cells representing an ocean boundary. The new TabFile features are discussed in the "Tabfiles—Time-Series Input" section of this appendix. This function allows fewer TabFiles to be required to describe a set of model features—for example, one GHB sea-level gage spread across multiple GHB cells.

Time Series Files (`TSF`) are like TabFiles in that they are single files that contains a time stamp and associated data input. In contrast to a TabFile, a `TSF` is optimized to be applied to a single model feature—for example, streamflow to a single streamflow-routing (SFR) segment. A `TSF` sets the single model feature according to a requested time, or time interval. Unless otherwise stated, the requested time is the date at the end of the current time step, and the time interval is the starting and ending dates of the current time step. A `TSF` includes options for how to extract the time-series data in it. Some of the available options are interpolating the data to a single date, using a step function, or elapsed time-weight-averaging of the data. A TSF may specify dates as a month and day—such that it only specifies 1 year of input—and it automatically cycles through the file appending the current time step's year. This allows the TSF to specify 1 year of data that are reused throughout a simulation.

The LineFeed input separates spatial and temporal input by defining all spatial locations of a model feature (for example, GHB cell location and conductance) at the start of the simulation. Once all the features are defined, then each stress period defines the features that are in use and their temporal component (for example, GHB locations currently active and their `BHEAD`). This separation makes the input easier to maintain through the linkages to websites, a database or spreadsheet software, and build automation for temporal data updates through scripting. Further, previous model inputs can easily be translated into this new input structure because it mimics the "step-function" stress-period style input. LineFeed has been implemented in the GHB, Well (WEL), Multi-Node Well (MNW2), and the Drain Return (DRT) Packages and the FMP farm-well input. It also is available in partial form to the Streamflow-Routing Package (SFR). The LineFeed input is discussed in detail in the section "LineFeed—Alternative Temporal Input."

The last time-series input structure discussed in this appendix is the *Transient File Reader* (`TFR`). The `TFR` feature provides an easy way to maintain and update spatial-temporal input data to the model, such as climate arrays of precipitation or potential evapotranspiration. The `TFR` is a part of the *List-Array* input that is described in detail in appendix 1. The discussion in this appendix is about its utility for specifying spatial-temporal varying inputs and applications to calibration.

TabFiles—Time-Series Input

TabFiles are a time-series-like input that sets a model property based on the time step's simulation time. The TabFile feature allows for convenient construction of an input dataset that is independent of the stress period and time step number. TabFiles also apply the input property by time step instead of stress period. TabFiles were first implemented in the revised SFR Package in MF-NWT (Niswonger and others, 2011) and subsequently added to the Surface-Water Routing (SWR) process to specify stream inflows and diversions by model simulation time. MF-OWHM2 extended TabFiles in four ways: they work with additional packages; support date-time as decimal years and calendar dates; can apply properties by stress period or time step; and can pass the TabFile result to a custom expression. This extension broke the compatibility with the MF-NWT style of TabFiles. To maintain backward compatibility, if the keyword **`TABFILES`** is specified in SFR or SWR, then the input uses the MF-NWT style TabFile feature and input structure. In contrast, the keyword **`TABFILE`**—without the **`S`**— is used for the MF-OWHM2 style TabFiles for SFR, GHB, WEL, MNW2, and DRT packages (note that the new MF-OWHM2 style TabFiles were not included in SWR).

The advantage of the new TabFile feature is that each individual TabFile has a unique name to enable it to link to multiple features within MF-OWHM2 packages (for example, multiple WEL pumping wells can have their rate set to a TabFile, or multiple GHB cells can define their `BHEAD` to the same TabFile). Another advantage is that a tab scale factor is included for each of the features that are linked to a TabFile. The scale factor is applied to the numerical value stored in the TabFile, making its value unique to each time series specified.

The TabFile structure is a list of two columns of data. The first column contains simulation times, and the second column contains numerical values used in the model. An example TabFile that could be used for a model having 30-day stress periods (SP) and 5-day time steps follows:

```
10., 50.        #End of Time Step 2, SP 1
20., 55.
24., 58.
25., 60.
30., 54.        #End of SP 1
60., 52.        #End of SP 2
```

The data in the TabFile are interpreted in one of three ways depending on the availability of data in the current time step (that is, `TOTIM-DELT < DATA <= TOTIM`, where `TOTIM` is the elapsed simulated time and `DELT` is the time-step length). If the current simulation time is before the first TabFile time, after the last TabFile time, or has a single value in the time step, then the appropriate single value in the TabFile is applied. For instance, in this example, the first time step (0 to 5 days) uses a value of 50, and any time step after 60 days has a value of 52. If there are no values in a time step, then a value is linearly interpolated to `TOTIM`. In this example, the time step from 35 to 40 days linearly interpolates to 40 days using the values (30, 54) and (60, 52). If there are multiple TabFile values in a time step, the time-weighted average is used. Here, the time from 20 to 25 day time averages (20, 55) and (24, 58). A note of caution is that the data are applied on a time-step basis, not by stress period. An example using a 30-day stress period (SP) and 10-day time steps (TS) is shown here:

```
10.0, 52.     # SP 1, TS 1 (52 is applied to interval (0,  10] d)
20.0, 54.     # SP 1, TS 2 (54 is applied to interval (10, 20] d)
30.0, 56.     # SP 1, TS 3 (56 is applied to interval (20, 30] d)
60.0, 99.     # SP 2  linear interpolations between 56 and 99 are
              #       applied to time steps between 30 and 60 d;
              #       and for TS 3, 99 is applied to interval (50, 60] d
```

With this release of MF-OWHM2, the TabFile input has been revised to allow processing of the time-series input either on the time-step level (original method) or on the stress-period level (new method), or to ignore the TabFile time value and assume that each row of the TabFile is loaded every time step or stress period (similar to a LineFeed file). If the package that uses the TabFiles supports an options block, then the input may be specified there or with the keyword **`TABFILE`** at the start of the input. TabFiles may also now be linked to the Expression Parser to evaluate the time-series value in a custom expression. The Expression Parser link allows for a single TabFile to be extended to multiple model features that may have a non-linear relationship. This simplifies the input structure and makes it easier to maintain. This TabFile can then be passed to the Expression Parser to translate the ocean level to freshwater equivalent—to account for the salt-water density differences—for each GHB ocean boundary cell.

The TabFile input was modified to incorporate new features. The new features are enabled through the use of optional keywords specified on the line that includes the **`TABFILE`** keyword. This allows the TabFile input to be backward compatible with previous versions. The TabFile input begins with specifying the **`TABFILE`** keyword after—if they are present—the **`PARAMETER`** keyword and any block input. Figure 2.1 presents the TabFile syntax and formal description. Figure 2.2 presents the general input structure used for mapping a TabFile declared with the **`TABFILE`** keyword (fig. 2.1) to a specific package feature input (such as a WEL package well).

```
TABFILE NTAB FILEIO TIMEOPTION [SPBASIS] [TABEQN]
TABNAM TABLOCATION          →   Read NTAB times NTAB > 0
```

where

TABFILE is the keyword that triggers reading subsequent TabFile information.

NTAB is the number of TabFiles that are defined on the subsequent lines.

FILEIO is a flag that determines how TabFiles are handled with regard to random access memory (RAM) usage relative to file input and output (I/O). The flag is set to either a `0` or a `1` with the following meanings:

- `0` indicates that the entire TabFile is loaded into memory (fast, but large RAM requirement).
- `1` Recommended option; indicates that only the portion of the TabFile that pertains to the current time step is loaded into memory (negligible RAM usage).

TIMEOPTION is the spatial input keyword that is set to

- **SIMTIME** specifies that TabFile times use the model simulated time with time units specified by the DIS and a starting point of `0`.
- **REALTIME** specifies that TabFile times use decimal years, or dates. If specified, then the BAS package must include the **STARTDATE** option to enable calendar dates as part of the simulation.
- **IGNORE_TIME** specifies that TabFile should ignore the time values and instead expect a one to one relationship with the time steps and each TabFile row.

SPBASIS is an optional keyword that indicates TabFile times should be parsed by the stress period rather than the time step.

If the **IGNORE_TIME** option is used, then a single row of the TabFile is loaded for each stress period.

TABEQN is an optional keyword that, when present, indicates that where each TabFile is applied to an input that the TabFile should check for an optional expression, TAB_EQN, and if present apply it to the TabFile value.

TABNAM is a unique name (20 character maximum) that identifies the TabFile.

TABLOCATION is the location of the Tabfile specified with *Generic_Input_OptKey*.

Neither the keyword **INTERNAL** nor *Implied_Internal* is allowed for reading the location of a TabFile.

Figure 2.1. TabFile input structure expected at the start of the one of the TabFile supported packages. [As of 2019, supported packages are GHB, WEL, MNW2, and SFR.]

ABC [xyz] [TABNAM TSFAC [TAB_EQN]]

where

ABC is the specific package feature's input that is being associated with a TabFile. This is typically the spatial location as layer, row, and column and also contains the model input that is being set by the TabFile.

For example, the GHB package TabFiles update BHEAD with a TabFile, so the expected ABC is

"Layer Row Column BHEAD Cond"

such that the full line input is

"Layer Row Column Bhead Cond [xyz] [TABNAM TSFAC [TAB_EQN]]".

TABNAM is a unique name that identifies the TabFile.
It must match one of the TABNAMs specified in the **TABFILE** input.
If not specified, then model feature assumed to not be linked to a TabFile.

TSFAC is a scale factor that is multiplied to the final TabFile value.

If **TAB_EQN** is specified in the **TABFILE** input, then TSFAC is applied to the TabFile value after it is evaluated by the ExpressParser.

Set to "1.0" to skip the scale factor.

TAB_EQN is an equation read if the **TABEQN** keyword is specified.

This equation must be enclosed in single quotes and can be used with any of the operations defined by the Expression Parser.

There are three reserved variable names that have a meaning to the Expression Parser when used by a TabFile:

TAB If the variable name **TAB** is present in the TAB_EQN, then it is replaced with the TabFile value for the current step.

For example, '5*TAB + TAB^2' would take the TabFile value and multiply it by five and then add its square to that value.

SIM If the variable name **SIM** is present in the TAB_EQN, then it is replaced with the elapsed simulated time (TOTIM) at the end of the time step.

REL If the variable name **REL** is present in the TAB_EQN, then it is replaced with the date, as a decimal year, at the end of the time step.

Figure 2.2. Expected input to link a TabFile to a specific package input feature.

Time Series Files (TSF)

A Time Series File (TSF, fig. 2.3) is an alternative input that is a single file containing a time stamp and associated data input. The use of a TSF requires the simulation to specify a starting calendar date to keep track of each time step's starting and ending date. To specify a starting calendar date, the BAS package options block must include the option **STARTDATE** DATE, where DATE is the simulation's starting calendar date. A TSF is like a TabFile in that both contain a time stamp (as a decimal year or calendar date or calendar date with a 24-hour clock time) and the time stamp's associated datum. A TSF differs from a TabFile because it is only associated with one feature, but offers a more robust set of options to parse and interpret the time series. Examples of a single feature are one SFR stream segment's inflow or a single WEL package well. It should be noted that the same TSF file may be opened multiple times at once by referencing the same file multiple times, allowing reuse of one file's content as input for multiple features. For example, the same file can be associated with multiple WEL package wells. However, the TSF are not optimized to the extent that TabFiles are for a "one-to-many" connection (for example, one TabFile could easily connect to 1,000 or more GHB model cells). A TSF is also optimized for time series of any length, where there is a minimal runtime and RAM penalty for longer time-series files.

The use of a TSF depends on what package or property is using it, but the location of the time-series file is specified with the *Generic_Input_OptKey* (appendix 1), and an optional keyword (**TSF_OPTION**) may be specified to indicate how the time-series data are parsed and interpreted. A TSF is used to look up and return a single value for a given date or a date range. Typically, if a TSF is parsed using a single date, then the TSF uses the current time step's ending calendar date to return a data value. Additionally, if a TSF is parsed using a date range, then the TSF uses the current time step's starting and ending calendar date to parse and return a data value. For example, a single date (such as the end of the time step) can be parsed out of a TSF and if it is not found, then the two closest dates are linearly interpoated (or extrapolated) to it to find the return value (which occurs when **TSF_OPTION** = **INTERPOLATE**).

A TSF has a simple input structure that consists of two columns of information. The first column contains a date, and the second column contains the numerical value that is recorded or associated for that specified date. A datum point in the TSF is any row in the text file that contains a date and value. Comments that are preceded by a "#" are allowed before the first data row, after the last data row, and to the right of the specified data on any row (fig. 2.3). It is required that the data rows in a time-series file be sorted in chronological order, and elapsed simulated time (TOTIM) is not supported as a date option. The format of acceptable date options is described in tables 2.1 and 2.2. In general, the different date formats may be intermixed in the same time-series file, with one exception—if the year is omitted from one date, then it must be omitted for all the dates in the time-series file (fig. 2.4).

```
# Comments only allowed to the right of the time series data or
# before and after the time series dataset
# First column is the time stamp, second column data value
# Date              Data
  10/20/1982   4.9     # A data point that occurs on October 20th, 1982
  10/25/1982   3       # Comments are allowed to the right of data
  10/26/1982   3.4
  10/27/1982   1.5
  10/28/1982   3
  10/29/1982   3.9
  10/30/1982   5
  10/31/1982   7.5
# Comment allowed after time series data
```

Figure 2.3. Example Time Series File (TSF) that is parsed by a single date or date range and returns the appropriate input data. If a 24-hour clock time is not provided as part of the date, then 00:00:00 (midnight) is assumed. Comments are preceded by a # and may only appear before or after the time-series dataset or to the right of individual time-series data records.

Table 2.1. Definitions for calendar date symbology.

Symbol	Representation
mmm	Three letter month or full name—Jan or January.
mm	Two-digit month number—01 or 1 with valid domain from 1 to 12.
dd	Two-digit day of month, numeric—01 or 1 with valid values from 1 to 31.
yyyy	Four-digit year, numeric—must be four digits—1979 with valid values from 0000 to 9999.
T	Separator to indicate that 24-hour clock is included with calendar date.
hh	Two-digit hour of day, 24 hour format—01 or 1 with range from 00 to 23.
mm	Two-digit minute of hour—01 or 1 with range from 00 to 59.
ss	Two-digit second of minute—01 or 1 with range from 00 to 59.

Table 2.2. Time Series File acceptable date formats.

[ISO standard refers to the International Organization for Standardization 8601 format (https://www.iso.org/iso-8601-date-and-time-format.html).]

Date format	Comment
mm/dd/yyyy	American style date.
mmm/dd/yyyy	American style date with month as a word.
mm/yyyy	Automatically sets day to 1.
yyyy-mm-dd	ISO standard.
mm/dd/yyyy**T**hh:mm:ss	**T** separates calendar date from 24-hour clock time.
yyyy-mm-dd**T**hh:mm:ss	**T** separates calendar date from 24-hour clock time.
DYear	Decimal year—1979.307, indicates 4/23/1979.
mm\dd	Auto-append year based on time step's year.
mmm-dd	Auto-append year based on time step's year.
mmm	Auto-append year based on time step's year and automatically sets day of month to 1.

If the TSF date does not contain the 24-hour clock time, then the clock time is assumed to be at the start of the day (hh:mm:ss = 00:00:00). Leap years are automatically accounted for by using 365-day years for non-leap years and 366 day years for leap years; this ensures proper application of 02/29/yyyy, where yyyy is a leap year. Improper dates, such as "2/**29**/1961", "11/**31**/2001", or "**13**/15/2001", raise an error during the processing of time-series file (the example bad dates have the bad input in bold).

A cyclic annual time-series file (CAT) is special case of a TSF that occurs when the TSF's dates do not include the year—that is the dates are specified only with the month or month and day. When a CAT is parsed using a single date or date range, it uses the associated year in the single date or date range to determine the year that the CAT's dates have. A CAT may include February 29 (or 2\29), but it is not recommended because it creates ambiguity during non-leap years with the first of March (3\1) with both having the same day of the year. Figure 2.4 presents examples of time-series files that do not include the year.

When a CAT is parsed using a date range that results in a year change the CAT cycles from the bottom of the file back to the top to start a new year. For example, if figure 2.4*A* is parsed using the date range October 1, 1995, to February 1, 1996, then it would use the data specified on the rows with `OCT`, `DEC`, and `JAN`. In this example, the date `OCT` is converted to `10/1/2015`, the date `DEC` is converted to `12/1/2015`, and the date `JAN` is converted to `1/1/2016`. If the date range spans multiple years, then the CAT file cycles through itself until it reaches the final year. For example, if figure 2.4*A* is parsed using the date range October 1, 1995, to February 1, 1997, then the CAT is parsed using the data specified on the rows with `OCT` and `DEC` for the year 1995, then the entire file using the year 1996, and then finally `JAN` for the year 1997.

A

```
# Date does not include year
# Day of month assumed 1,
# because it is not specified

# Date     Data
  JAN     4.9
  FEB     4.2
  MAR     3
  APR     3.1
  MAY     3.4
  JUL     1.5
  OCT     3
  DEC     5
```

B

```
# Date does not include year

# Date        Data
  JAN-15     4.9
  JAN-30     4.6
  FEB-10     4.2
  MAR-5      3
  MAY-08     3.4
  JUL-18     1.5
  OCT-31     3
  DEC-12     5
```

C

```
# Date does not include year

# Date      Data
  1\15     4.9
  1\30     4.6
  2\10     4.2
  3\5      3
  5\08     3.4
  7\18     1.5
  10\31    3
  12\12    5
```

Figure 2.4. Example Time Series File (TSF) that does not include the year (yyyy) making the TSF a cyclic annual time series (CAT). A CAT assumes that the file contains the input for a single year and can be cycled through to represent time-series data across multiple years. When a CAT is parsed by a date, the year for the date is applied appropriately to the CAT's month and day. If a date range is specified and extends beyond the end of the file, then the file cycles back to the start and updates the year. *A*, Example CAT-TSF that specifies a three-letter month name (mmm). This case automatically sets the day of the month to one. *B*, Example CAT-TSF that specifies a three-letter month name and two-digit day number (mmm-dd). *C*, Example CAT-TSF that specifies a two-digit month number and two-digit day number (mm\dd). The backslash is used to indicate that the date is only the month and day. [Comments are preceded by a # and may only appear before or after the time-series data or to the right of the time-series data.]

A TSF format allows for specifying a keyword option that defines how to interpret the input data rows for different types of time stamps. The interpretation is based on the starting and ending date of the time step. Table 2.3 describes the options available for interpreting the time-series data. Depending on the option, it results in the use of either the calendar date at the end of the time step (one-point) or the start and end calendar dates of the time step (two-point). An example of a one-point option is **`INTERPOLATE`**, which selects the calendar dates that are immediately before and after a time step's end date in the TSF and interpolates between them to the end date of the time step. The set of one-point options are **`INTERPOLATE`**, **`NEAREST`**, **`STEP_FUNCTION`**, and **`NEXT_VALUE`**. For all the one-point options, if the time step end date is within 5 seconds of the date-time stamp for one of the TSF data rows, then the value from that TSF row is automatically used. For example, a TSF with the option keyword **`STEP_FUNCTION`** and a time step ending date of 4/23/1979T00:00:00 uses the data on the date of 4/23/1979T00:00:02 instead of 4/22/1979T23:00:00, even though the **`STEP_FUNCTION`** option normally selects the closest date before the end of the time step. The two-point options are **`TIME_MEAN`**, **`MAX`**, **`MIN`**, **`SUM`**, and **`DAY_SUM`**. Each of these options are applied to any TSF data points between the start and end dates of the time step. If the time-step end date is before the first date in the TSF, then the first value is returned regardless of the TSF option keyword. Also, if there are no data points between the time-step start and ending dates, then the values returned are from the closest TSF data point before the end of the time step (that is, it switches to the **`STEP_FUNCTION`** option if there are no data points in the time step).

It is recommended to use the **`TIME_MEAN`** option for most cases. This option multiplies the time-series data by elapsed time in the time step for each data point and then divides the sum of these time-weighted data values by the time-step length. The **`TIME_MEAN`** option is best suited for features such as streamflow where mass should be conserved—that is, the streamflow at each time entry is converted to a volume, then summed and divided by the total time to obtain an equivalent flow rate. If data are sparse in the time-series file, then the **`STEP_FUNCTION`** option works well for features that need to hold the value constant until a new value is available. This is advantageous for pumpage records that may only have a monthly or seasonal pumping rate specified. This option causes the specified rate to be used until the end of the time step.

Figure 2.5 presents an example TSF (fig. 2.5*A*) and how the TSF data are interpreted for different options. Figure 2.5*B* through figure 2.5*G* define the x-axis as the date to search for in the TSF, which could represent the end of the time step's date. Using the x-axis date, figure 2.5*B*, *C*, *D*, and *E* specify on the y-axis the interpreted result for the **TSF_OPTION**'s **NEAREST**, **STEP_FUNCTION**, **NEXT_VALUE**, and **INTERPOLATE**, respectively. The **TIME_MEAN** option requires two points, which are typically the starting and ending dates of the time step. Figure 2.5*F* illustrates the **TIME_MEAN** by setting the starting date to 12/1/2009 (first point) and using the x-axis value for the ending date (second value). Consequently, figure 2.5*F* represents a cumulative moving average with respect to 12/1/2009 of the time-series data specified in figure 2.5*A*. Figure 2.5*G* illustrates the **TIME_MEAN** by setting the starting date to 90 days less than the ending date and uses the x-axis value for the ending date (second value). Because of this, figure 2.5*G* represents a 90 day-moving average of the time-series data specified in figure 2.5*A*. Figure 2.5*H* provides specific results for the **TIME_MEAN** option for different starting and ending dates.

Table 2.3. List of Time Series File data interpretation options.

[One-point indicates that the method only uses the calendar date at the end of the time step or single date input (one time value). Two-point indicates that the method uses the calendar date at the start and end of the time step or requires two date inputs. No-point indicates the keyword is independent of calendar dates. **Abbreviation**: TSF, time-series file]

TSF option	Description
CONSTANT	Disables the TSF and only uses a single constant value for all time. The option must be followed by a single number that represents the constant value (no-point).
INTERPOLATE	Interpolates for all "time-series" data points closest to matching the ending date of the time step. Except if end of time step is within 5 seconds of a time-series date, then use that TSF data point—no interpolation (one-point).
NEAREST	Uses the closest time-series value to the end of the time step (one-point).
STEP_FUNCTION	Uses the closest time-series value before the end of the time step (one-point).
NEXT_VALUE	Uses the closest time-series value after the time-step end (one-point).
TIME_MEAN	Elapsed time weighted average of all data points that lie within the time step's start and ending dates (two-point).
MAX	Maximum value that lies within the time step's start and ending dates (two-point).
MIN	Minimum value that lies within the time step's start and ending dates (two-point).
SUM	The sum of all the values that lie within the time step's start and ending dates (two-point).
DAY_SUM	The sum of values multiplied by their elapsed time within the time step starting and ending. If this sum was divided by the time-step length, it would yield the "time mean" value (two-point).
SKIP	Disables the TSF and sets the return value to 0 (no-point).

A

```
# Time Series File:  TSF.txt
# Date          Data     Days to next date
 1/1/2010      20.       # 31
 2/1/2010      10.       # 28
 3/1/2010       5.       # 31
 4/1/2010      20.       # 30
 5/1/2010      30.       # 31
 6/1/2010       0.       # 92
 9/1/2010      30.       # 30
10/1/2010      20.
```

Figure 2.5. Summary of Time Series File (TSF) examples for various dates given different data interpretation options (**TSF_OPTION**): *A*, example Time Series File, "TSF.txt"; *B*, **NEAREST** option; *C*, **STEP_FUNCTION** option; *D*, **NEXT_VALUE** option; *E*, **INTERPOLATE** option; *F*, **TIME_MEAN** option that is relative to the date 12/1/2009, that is, the first point is 12/1/2009 and the second point is the x-axis; *G*, **TIME_MEAN** option that is a 90 day moving average, that is, the second point is the x-axis date and the first point is 90 days before that date; and *H*, **TIME_MEAN** option for select date ranges.

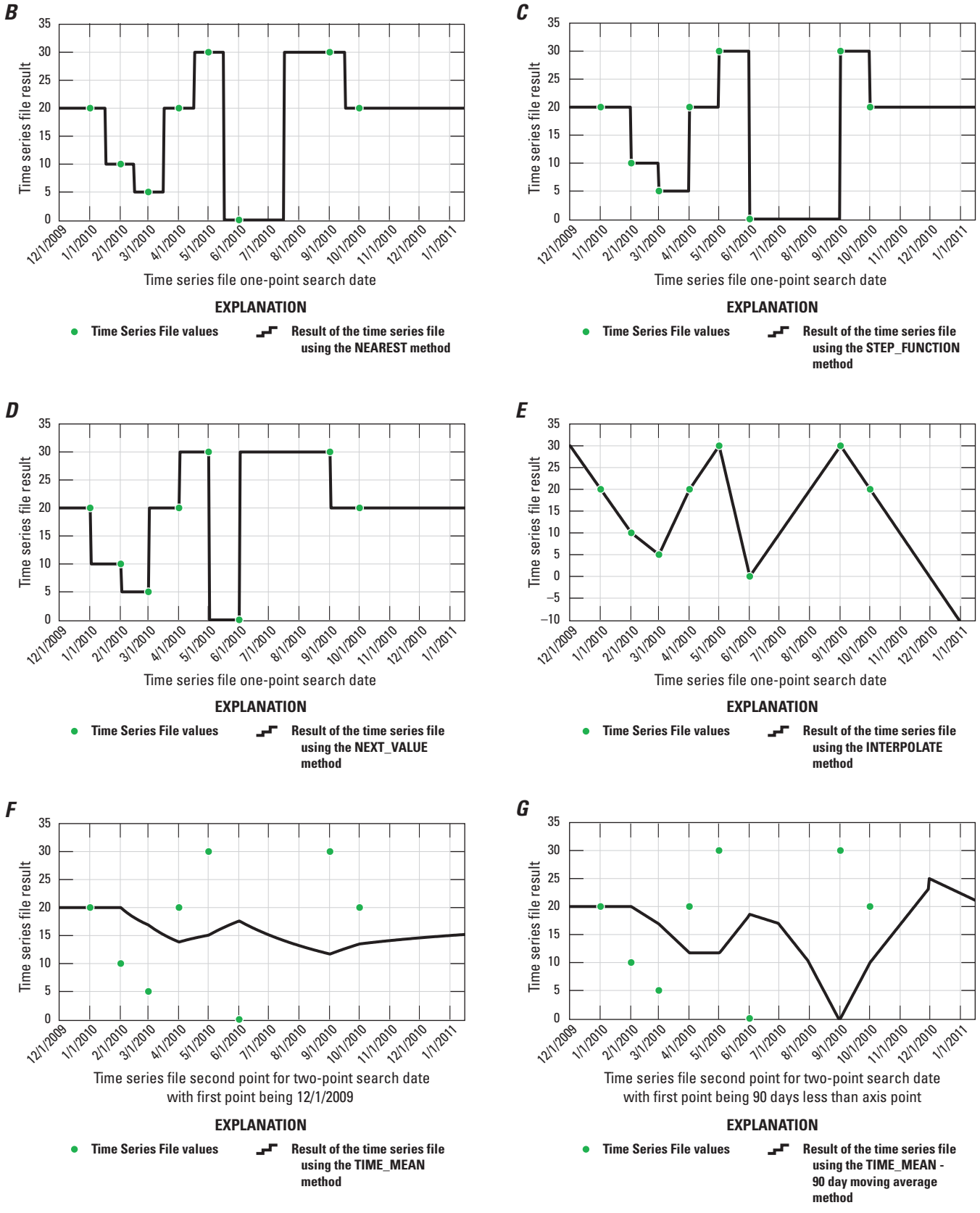


Figure 2.5. —Continued

H

First date (first point)	Second date (second point)	TIME_MEAN
1/01/2010	2/15/2010	16.88889
1/01/2010	3/01/2010	15.25424
3/15/2010	4/15/2010	11.77419
3/01/2010	8/01/2010	11.01307
1/01/2010	1/01/2011	14.58904

Figure 2.5. —Continued

Figure 2.6*A* contains the same dates and data as figure 2.5*A* but does not include the calendar year. This results in the TSF in figure 2.6*A* being interpreted as a CAT, which allows for the TSF to span multiple years without explicitly defining the dates for each year. This is advantageous when the time-series data are cyclic on an annual time frame. Note that figure 2.5*A* can span multiple years but must directly specify in the TSF all dates and data that should be included. For example, the TSF figure 2.5*A* must specify values (data) with dates in 2009, 2010, 2011, and 2012 to generate the same charts as presented in figure 2.6*B* through 2.6*G*.

Figure 2.6*B* through figure 2.6*G* define the x-axis as the date to search for in the CAT, which could represent the end of the time step's date. The CAT's dates have the year automatically determined based on the calendar year in the x-axis's dates. Using the x-axis date, figure 2.6*B*, *C*, *D*, and *E* specify on the y-axis the interpreted result for the **TSF_OPTION**'s **NEAREST**, **STEP_FUNCTION**, **NEXT_VALUE**, and **INTERPOLATE**, respectively. Figure 2.6*F* illustrates the **TIME_MEAN** by setting the starting date to 12/1/2009 (first point) and uses the x-axis value for the ending date (second value). Figure 2.6*F* is analogous to a cumulative moving average with respect to 12/1/2009 of the cyclic time-series data specified in figure 2.6*A*. Figure 2.6*G* illustrates the **TIME_MEAN** by setting the starting date to 90 days less than the ending date and uses the x-axis value for the ending date (second value). This results in figure 2.6*G* representing a 90 day-moving average of the time-series data specified in figure 2.6*A*. Figure 2.6*H* provides specific results for the **TIME_MEAN** option for different starting and ending dates. Note that the results of figure 2.6*H* are identical to figure 2.5*H*. This occurs because the CAT in figure 2.6*A* is parsed using the year 2010, which converts all its dates to 2010 making it identical to figure 2.5*A*. The tables would not match if the date ranges included years other than 2010. This occurs because CAT repeats annual times series while the TSF only uses the values at its end members (1/1/2010 and 10/1/2010) just use its end members.

A

```
# Cyclic Annual Time Series File:  CAT.txt
# This is a special type of Time Series File,
# which does not include in the date
# the calendar year.
# The dates and data represent
# an annual time series that is repeated.
# Date          Data
 Jan-1        20.
 Feb-1        10.
 Mar-1         5.
 Apr-1        20.
 May-1        30.
 Jun-1         0.
 Sep-1        30.
 Oct-1        20.
```

Figure 2.6. Example of a Time Series File (TSF) that does not include the calendar year in any of the dates, which is identified as a cyclic annual time series (CAT). Summary of CAT examples for various dates given different data interpretation options (**TSF_OPTION**): *A*, example TSF that is a CAT, "CAT.txt"; *B*, **NEAREST** option; *C*, **STEP_FUNCTION** option; *D*, **NEXT_VALUE** option; *E*, **INTERPOLATE** option; *F*, **TIME_MEAN** option that is relative to the date 12/1/2009, that is, the first point is 12/1/2009 and the second point is the x-axis; *G*, **TIME_MEAN** option that is a 90-day moving average, that is, the second point is the x-axis date and the first point is 90 days before that date; and *H*, **TIME_MEAN** option for select date ranges.

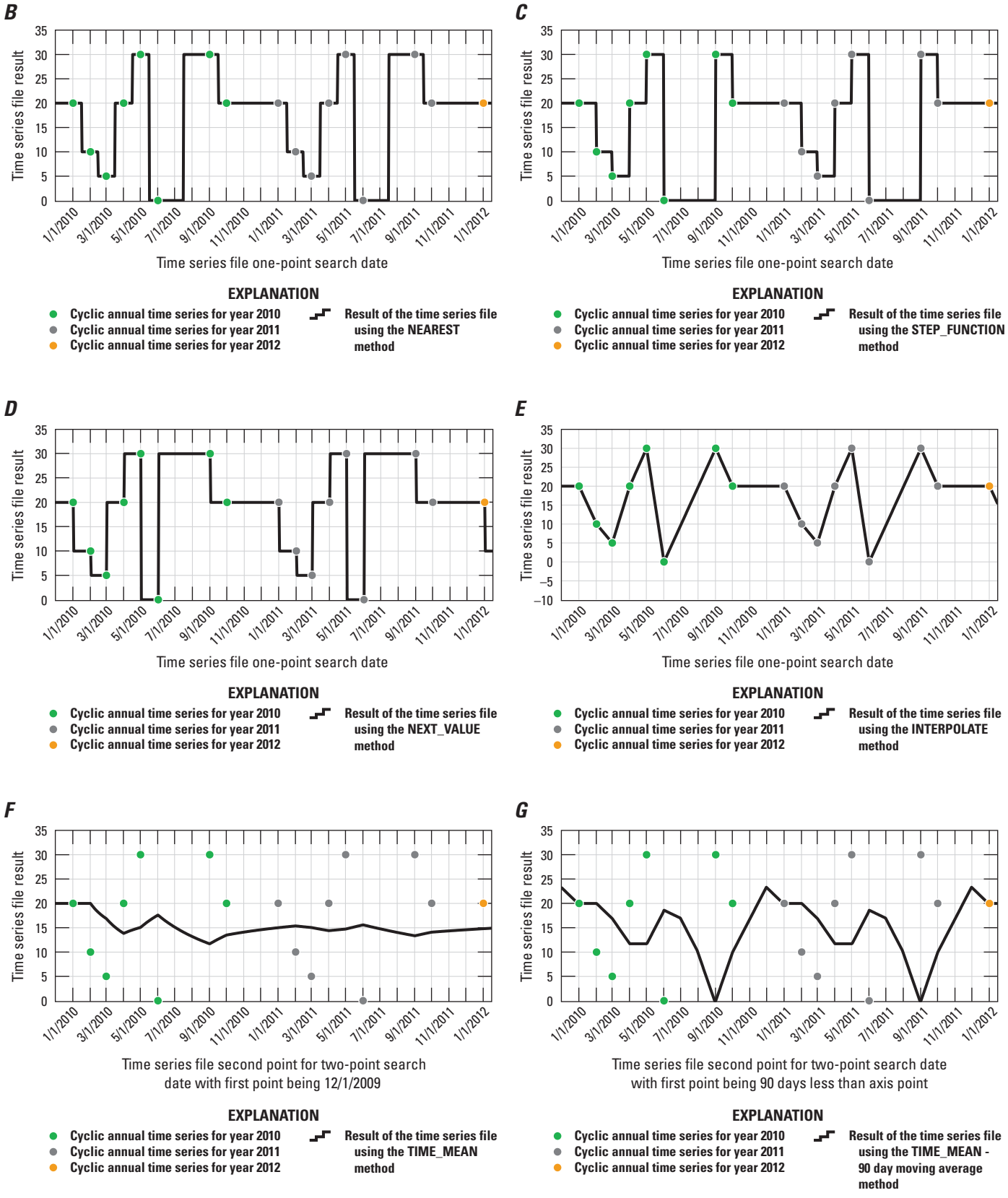


Figure 2.6. —Continued

H

First date (first point)	Second date (second point)	TIME_MEAN
1/01/2010	2/15/2010	16.88889
1/01/2010	3/01/2010	15.25424
3/15/2010	4/15/2010	11.77419
3/01/2010	8/01/2010	11.01307
1/01/2010	1/01/2011	14.58904

Figure 2.6. —Continued

Time Series File and ULOAD

The *Universal Loader* (ULOAD) described in appendix 1 can load a set of Time Series Files (TSF) using the *List Style* input structure. This input structure is referred to as *TSF Style* input, and each *List Style* record must be formatted as described in figure 2.5.

The *TSF Style* input (fig. 2.7) specifies the TSF location with *Generic_Input_OptKey*, which includes a set of optional post-keywords. If the scale-factor post-keyword, **SF** *SCALE*, is specified, then it is only applied to the data (second) column in the TSF. The post-keyword **BUFFER** may improve the TSF input and output performance, and it is recommended to set the buffer slightly more than the size of the TSF—note that by default the buffer is set to 32 kilobytes. It should be noted that *TSF Style* input record number, ID (fig. 2.7), must be in numerical order starting 1 and increases sequentially to the number of expected TSFs.

Figure 2.8 is an example *TSF Style* input that loads three time-series files—that is, it uses ULOAD with *List Style* to read three TSFs. In this example, ULOAD uses the keyword **INTERNAL** to indicate the *List Style* input is on the subsequent lines. The first TSF, ID=1, is specified as Time_Series_File1.txt. This file uses the option **STEP_FUNCTION**; has its second data column multiplied by 0.3048; and is pre-buffered into 96 kilobytes of RAM. The second and third TSF use the options **INTERPOLATE** and **MAX**, respectively. Note that the third TSF does not include the post-keyword option **BUFFER**, so it is automatically pre-buffered into 32 kilobytes of RAM.

Time-Series File Block Style Input

Time-series files (TSF) may be loaded as a group of files that each are applied to a different model feature (such as inflow to different SFR segments). *Block Style* input (appendix 1) can load an arbitrary number of TSFs and relies on an identifier, TSFNAME, that maps the TSF to the specific model feature. How TSFNAME is defined depends on the package or process that is utilizing the TSF *Block Style* input. Figure 2.9 presents TSF *Block Style* input general input structure. An example that uses *Block Style* input to read three TSFs for the SFR package is presented in figure 2.10. In SFR, the TSFNAME is the SFR segment that is assigned an inflow or diversion amount based on the associated TSF data. See appendix 3 for more information on how TSFs are applied to various packages.

TSF Style reads a set of Time Series Files (TSFs) with ULOAD using *List Style* input.
Each *List Style* input record must include

ID **TSF_OPTION** *Generic_Input_OptKey*

where

ID is the *List Style* input record ID number.

TSF_OPTION a keyword that indicates how the time-series data are parsed and interpreted for a given search date or search date range.

Accepted keywords and their description are

CONSTANT *VALUE*	Disables the TSF and only uses a single constant value for all time. The option must be followed by a single number, *VALUE*, that represents the constant value (no-point). *Generic_Input_OptKey* is not read.
INTERPOLATE	Interpolates using the two "time-series" data points that are closest to the requested search date (one-point). If end of time step is within 5 seconds of a time-series date, then use that TSF data point.
NEAREST	Uses the closest time-series date and value to the search date (one-point).
STEP_FUNCTION	Uses the closest time-series date and value located before the search date (one-point).
NEXT_VALUE	Uses the closest time-series date and value located after the search date (one-point).
TIME_MEAN	Elapsed time weighted average of all data points that lie within a search date range (two-point).
MAX	Maximum value that lies within a search date range (two-point).
MIN	Minimum value that lies within a search date range (two-point).
SUM	The sum of all the values that lie within a search date range (two-point).
DAY_SUM	The sum of values multiplied by their elapsed time within the search date range (two-point).

Generic_Input_OptKey is the Time Series File (TSF) location.
INTERNAL cannot specify the location of the TSF.
If there is a scale factor, *SCALE*, specified, then it is only applied to the return values (second column).

Figure 2.7. *TSF Style* input general input structure and explanation of input.

```
INTERNAL
1  STEP_FUNCTION   Time_Series_File1.txt   BUFFER 96   SF 0.3048
2  INTERPOLATE     Time_Series_File2.txt   BUFFER 64
3  MAX             Time_Series_File3.txt
```

Figure 2.8. Example *TSF Style* input that loads three time-series files with the **INTERNAL** keyword.

A

```
BEGIN TIME_SERIES_FILES
  #
  TSFNAME   TSF_OPTION   Generic_Input_OptKey   # Repeat as needed
  #
END
```

Figure 2.9. Time series file (TSF) *Block Style* input *A*, general input structure and *B*, explanation of input.

B

TSFNAME	is a unique identifier that maps the TSF to a model property. The structure of TSFNAM and type depends on the input using the TSF. For example, the SFR package defines TSFNAM as the SFR segment number that the TSF is applied to as an inflow or diversion amount.
TSF_OPTION	a keyword that indicates how the time-series data are parsed and interpreted for a given search date or search date range. Accepted keywords and their description are
CONSTANT VALUE	Disables the TSF and only uses a single constant value for all time. The option must be followed by a single number, VALUE, that represents the constant value (no-point). *Generic_Input_OptKey* is not read.
INTERPOLATE	Interpolates using the two "time-series" data points that are closest to the requested search date (one-point). If end of time step is within 5 seconds of a time-series date, then use that TSF data point.
NEAREST	Uses the closest time-series date and value to the search date (one-point).
STEP_FUNCTION	Uses the closest time-series date and value located before the search date (one-point).
NEXT_VALUE	Uses the closest time-series date and value after the search date (one-point).
TIME_MEAN	Elapsed time weighted average of all data points that lie within a search date range (two-point).
MAX	Maximum value that lies within a search date range (two-point).
MIN	Minimum value that lies within a search date range (two-point).
SUM	The sum of all the values that lie within a search date range (two-point).
DAY_SUM	The sum of values multiplied by their elapsed time within the search date range (two-point).
Generic_Input_OptKey	is the Time Series File (TSF) location. **INTERNAL** cannot specify the location of the TSF. If there is a scale factor, SCALE, specified, then it is only applied to the return values (second column).

Figure 2.9. —Continued

```
# SFR time series input
# TSFNAME = SFR Segment Number
#
BEGIN TIME_SERIES_INPUT
#
# TSFNAME  TSF_OPTION       Generic_Input_OptKey
       22  STEP_FUNCTION  Time_Series_File1.txt  BUFFER 32  SF 0.3048
       15  INTERPOLATE    Time_Series_File2.txt  BUFFER 64
       8   MAX            Time_Series_File3.txt
END
```

Figure 2.10. Example Time series file (TSF) *Block Style* input that specifies the inflow or diversion for three SFR segments. [Comments are any text that are written to the right of a # symbol.]

LineFeed—Alternative Temporal Input

The LineFeed alternative input format (LineFeed) is introduced in this release of MF-OWHM2. This input format allows packages to separate the spatial input from temporal input. The main reason to do this is to simplify package input so it is easier to maintain. It also facilitates linkages to websites, databases, or spreadsheets. This input also allows users of older, stress-period based models to easily translate their input to MF-OWHM2 and then simplify future updates. This corresponds to the "self-updating" concepts presented in the main body of this report. The name LineFeed refers to process used by the original line printer to advance paper one line at a time.

LineFeed simplifies management of datasets that have frequently changing input or model features that "come and go" from the overall suite of inputs. For example, LineFeed is advantageous for the well packages (WEL and MNW2) because wells frequently change their pumping rates and during a simulation old wells can be destroyed (removed from the simulation) or drilled (added it the simulation). Currently, the LineFeed alternative input format is available in the GHB, WEL, MNW2, DRT, and FMP packages. It also is available in the SFR package for specifying inflows for stream segments in the SFR network (it is similar in implementation to SFR TabFiles). If LineFeed is used, then the regular input of the package is optional or may be filled out and the LineFeed input is added to the regular stress-period input. Any package input that is defined by LineFeed cannot use TabFiles. TabFiles may be used in the regular input structure, allowing for a mix of LineFeed model input with the regular input structure associated with TabFiles. For example, SFR can define inflow for segments 2, 4, and 6 with TabFiles, and segments 1, 3, and 8 with LineFeed—but you cannot specify segment 2 with a TabFile and LineFeed.

For MF-OWHM2, LineFeed format is composed of a two-part feed file that splits the spatial input and temporal input. Each feed file is processed at the same time, allowing the data for a model input to be distributed across multiple files or specified entirely in one file. A feed file also supports inclusion of comments anywhere preceded by a "#" symbol (which is called a pound, hash, or number symbol). Figure 2.11 illustrates an example feed file that defines for three MNW2 wells the desired pumping rate (`Qdes`) and uses MF-OWH input comments—text preceded by a "#" symbol—to explain each part of the LineFeed feed file.

At the beginning of a feed file, all spatial input for all model features in the simulation is defined—this is similar to a MODFLOW Parameter, which requires the user to pre-specify the parameter input and then later activate the parameter. The number of spatial inputs is automatically counted as the number of uncommented lines from the start of the file to the termination keyword **`TEMPORAL INPUT`**. That is, each uncommented line contains a single spatial input—such as layer, row, and column—until the keyword **`TEMPORAL INPUT`** is read on a line. The second, or temporal-input, part of the feed file is specified after the keyword **`TEMPORAL INPUT`**. Each subsequent line after the **`TEMPORAL INPUT`** keyword specifies the input for all the previously defined spatial locations that is applied for a single "time interval." In this first release of LineFeed, the only "time interval" available is one stress period—that is, one stress period input per uncommented line. In the rest of this appendix, the words "stress period" are used synonymously with the words "time interval" because a stress period is the fundamental time interval used by MODFLOW to receive and hold constant user-specified inflows and outflows.

```
# MNW2 LineFeed Example
#
# The MNW2 well structure (such as screen interval) must be defined in
# the MNW2 input and referenced in the LineFeed file by its associated WELLID.
#
# First define the MNW2 WELLIDs that have their pumping rate set by LineFeed
#
# Spatial Section
#
WEL1         # 1st  column in TEMPORAL INPUT corresponds with Qdes for WEL1
WEL2         # 2nd  column in TEMPORAL INPUT corresponds with Qdes for WEL2
WEL3         # 3rd  column in TEMPORAL INPUT corresponds with Qdes for WEL3
#
TEMPORAL INPUT
#
# Temporal Section
#  WEL1     WEL2      WEL3       SP = Stress Period
 -10.0      NaN       0.0     # SP 1 - WEL2 does not exist; WEL3 exists and has Qdes = 0
 -20.0      NaN       0.0     # SP 2
 -30.0   -50.0        NaN     # SP 3 - WEL2 now exists; WEL3 no longer exists (destroyed)
```

Figure 2.11. Example MNW2 "feed file" that read by the LineFeed alternative input format (LineFeed). The feed file specifies three MNW2 wells whose pumping rate is defined for three stress periods. [Comments are any text that are written to the right of a # symbol.]

The implicit tie between the spatial framework and temporal parts of a feed file is in the ordering of inputs; that is, the ordered list of spatial input corresponds to the order of the temporal input in each row. Specifically, the first specified spatial input corresponds with the first value specified on a temporal LineFeed row; the second specified spatial input corresponds with the second value specified on a temporal LineFeed row, and so forth. This structure requires that each defined spatial input has a corresponding column in the temporal input. Not all the spatial inputs are active for the entire simulation period, so a null keyword is used to disable a spatial location that corresponds to a LineFeed row for a stress period. The null value, NaN, serves as a place holder for a column that is not used. The value of zero may also be used in some cases, but it has a different meaning than NaN for packages like MNW2, for which 0.0 means no pumping rate, but the well still exists and could be contributing to vertical interlayer flow through wellbore flow.

Figure 2.12 is the general input for a feed file that loads a number (N) of model features (for example, N wells for WEL package) and reads NPER temporal inputs (for example, WEL package pumping rates for the NPER model stress periods). The number of spatial features in use for one stress period depends on the number of columns specified across all the feed files on the same temporal-input row that do not have a value of NaN specified. This number is automatically added to the package's input for the stress period (sometimes referred to as `ITMP` in the package input description).

This column-based temporal input makes it easy to build the dataset with a spreadsheet software (for example, Microsoft Excel®). Figure 2.13 illustrates the general format of the spreadsheet. Figure 2.13*A* defines each of the spatial inputs, which are specified on each row of the spreadsheet. By transposing the spatial locations, the temporal input is defined in figure 2.13*B*. In this example, the commented name is included as a header for the temporal input. Each row in figure 2.13*B* represents a single stress period input. The spreadsheets in figure 2.13 can easily be copy and pasted to a feed file that includes the keyword **`TEMPORAL INPUT`** separating the two tables. Subsequent updates to the model would only need to add rows to the temporal spreadsheet.

```
# Spatial Input Section
SPATIAL_1     # Comment
SPATIAL_2
   ⋮
SPATIAL_N
#
TEMPORAL INPUT  # Keyword to initiate Temporal Input Section
#
# Spatial_1      Spatial_2     … Spatial_N
  TEMPORAL_1  TEMPORAL_2 … TEMPORAL_N   # First      Time Period Temporal Input
  TEMPORAL_1  TEMPORAL_2 … TEMPORAL_N   # Second   Time Period Temporal Input
  ⋮
  TEMPORAL_1  TEMPORAL_2 … TEMPORAL_N   # NPER^th   Time Period Temporal Input
```

where

SPATIAL_N	is the Nth spatial input. The expected input structure is defined by the package that uses LineFeed. For example, the WEL package expects the spatial input to be the model grid's Layer, Row, and Column.
TEMPORAL INPUT	is the keyword that ends the auto-counting of the spatial inputs and begins the temporal LineFeed input that loads one row for each time interval (stress period).
TEMPORAL_N	is the Nth temporal input that is applied to SPATIAL_N. The temporal input is defined by the package that uses LineFeed. If TEMPORAL_N is set to NaN, then SPATIAL_N is disabled. For example, the WEL package defines TEMPORAL_N as being the SPATIAL_N well's pumping rate Q. If TEMPORAL_N is set to NaN, then the corresponding SPATIAL_N well is not simulated for that time period—that is, the well does not exist. In contrast, if TEMPORAL_N set to 0.0, then the well is simulated with a pumping rate of 0, such that Q = 0.

Figure 2.12. General input structure of a LineFeed alternative input's "feed file." Each feed file contains a spatial input section (`Spatial_N`) and temporal input section (`Temporal_N`). The package that uses LineFeed defines the input structure for `Spatial_N` and `Temporal_N`. A package may include an arbitrary number of feed files, which are evaluated at the same time. That is, the first period input from each feed file is applied to the first period. [Comments are any text that are written to the right of a # symbol. ⋮ is a place holder for the third through the N-1 feed file input. NPER is the number of stress periods in a simulation.]

A

Spatial_1	# Name 1
Spatial_2	# Name 2
⋮	# …
Spatial_N	# Name N

B

# Name 1	Name 2	…	Name N	
Temporal_1	Temporal_2	…	Temporal_N	# SP 1
Temporal_1	Temporal_2	…	Temporal_N	# SP 2
⋮	⋮	⋮		⋮
Temporal_1	Temporal_2	…	Temporal_N	# SP NPER

Figure 2.13. Using a spreadsheet software to build a LineFeed alternative input "feed file." *A*, Template that defines N spatial inputs (such as model layer, row, and column for the WEL package). *B*, Template that defines NPER temporal inputs for the N spatial locations (such as pumping rate, Q, for the WEL package). [Comments are any text that are written to the right of a # symbol and all commented text is optional. ⋮ is a place holder for the third through the N-1 feed file input. SP is shorthand notation for stress period. NPER is the total number of stress periods simulated. `Spatial_N` is the Nth spatial input location. `Temporal_N` is the Nth temporal input for spreadsheet row (Stress Period) it resides in.]

LineFeed is enabled for a package by including at the start of the package input a *Block Style* input (see appendix 1) with the block **`NAME`** set to **`LINEFEED`**. If a package supports multiple *Block Style* inputs, then the order of the input blocks does not matter (for example, an **`OPTIONS`** block may appear before or after the **`LINEFEED`** block). Within the **`LINEFEED`** block are the locations of the feed files (called `FeedFile`), which are specified on each uncommented line using *Generic_Input_OptKey* (appendix 1) to specify the feed file location. Since a `FeedFile` must be a separate file, the *Generic_Input_OptKey* keyword **`INTERNAL`** is not allowed for specifying the feed file location. Figure 2.14 defines the LineFeed block input structure. Each `FeedFile` specified first reads the spatial section at the start of the simulation (figure 2.13*A*), and then one temporal row is loaded at the start of each stress period (figure 2.13*B*). The sections that follow discuss feed file format usages unique to specific packages.

LineFeed—Wel Package Input

The WEL package input with LineFeed can replace the input or supplement an already existing input structure. In MF-OWHM2, there are two WEL packages (WEL and WEL1) to maintain backward compatibility with the older **`TABFILE`** input and provide support for LineFeed. The original well package (declared in the Name file as WEL1) does not support LineFeed files and its input structure for TabFiles is documented in Hanson and others (2014). The input for WEL1, otherwise, is identical to that of the MF-OWHM2 WEL package. Please see appendix 3 for more information on the two WEL packages and how their TabFile input differs.

LineFeed is only supported in the MF-OWHM2 WEL package (declared in the Name file as WEL) and not the older WEL1 package. To use LineFeed in the WEL package it must include the **`LINEFEED`** input block at the start of the input file. If the input contains the optional **`PARAMETER`** keyword, then the LineFeed block may be specified before or after it. It is recommended to specify all *Block Style* input before any of the regular package input (such as **`PARAMETER`** or `MXACTW`).

A

```
BEGIN LINEFEED  [ FeedOpt ]
  #
  FeedFile  # May be repeated, as needed with one FeedFile per uncommented line
  #             FeedFile location specified with Generic_Input_OptKey
END
```

B

BEGIN LINEFEED initiates the *Block Style* input that reads a set of LineFeed feed files.

END terminates reading the LineFeed feed files.

FeedOpt is an optional set of global keywords that, when present, affect how the FeedFiles are interpreted. Multiple options can be specified, and are space separated after the block name LIENFEED. The options are

- **XY** Instead of specifying ROW and COLUMN, the x-coordinate and y-coordinate are specified in that order. These coordinates use the DIS package coordinate system to look up the ROW and COLUMN.
- **XYZ** Same is **XY** but replaces LAYER with a z-coordinate that is translated to a layer number using the DIS package's layer elevations (BOTM).
- **PRINT** Writes the information loaded from FeedFile for each stress period to the LIST file.

FeedFile is the LineFeed "feed file" location specified with *Generic_Input_OptKey*. Multiple feed files are allowed, and can be declared, with one file

INTERNAL cannot specify the location of the feed file.

If there is a scale factor, SCALE, specified, then it is only applied to the feed file's temporal input.

In the block input, FeedFile, can be repeated to define multiple feed files specified with *Generic_Input_OptKey* on each uncommented line.

Each separate feed file is processed at the same time, such that the first temporal input row of each file applies to the first stress period.

Figure 2.14. *A*, General input structure for LineFeed alternative input format *Block Style* input, and *B*, explanation of input. [Comments are any text that are written to the right of a # symbol. Optional input is enclosed in brackets, such as [FeedOpt].]

Figure 2.15 presents the WEL package feed file spatial input section. Figure 2.15*A* provides the general input format that identifies each well that is part of the feed file and figure 2.15*B* provides an explanation of each of the spatial inputs. The feed file's temporal input section (fig 2.13*B*) contains the pumping rate for each stress period on each row of text and each well's rate specified as each column on a row. If a well was not used for that stress period, then its pumping rate is set to NaN. Note that the well package pumping rate of zero results in the same effect as NaN; however, this slows simulation speed because the well is included in the simulation with a pumping rate of zero. Each column of the temporal input contains each well's pumping rate, Q, in units of L^3/T. A positive value indicates recharge, and a negative value indicates discharge (pumping).

Figure 2.16 is an example of WEL package input that uses LineFeed to define all its input. Figure 2.16*A* specifies that LineFeed will read two feed files, WEL_FEED1.txt and WEL_FEED2.txt, that define a total of five WEL package wells. In the first stress period, three wells pump groundwater, and one well injects water (figure 2.16*B* and figure 2.16*C*). The well commented as "WEL_2" with the NaN in stress period one is not included. It also includes the **BUDGET_GROUP** feature, which is described in the appendix 3. This feature allows the budget information printed to the LIST file and cell-by-cell file to be refined into custom groups rather than aggregating into one value for the entire package. If the **BEGIN BUDGET_GROUPS** is not present or is empty, the group name, **BGROUP**, is not read.

LineFeed also can work with the regular WEL package input file, instead of replacing it entirely as was done in Figure 2.16. The example presented in figure 2.17 includes both the regular WEL package input and LineFeed. Figure 2.17 does this by moving the previously discussed second feed file (WEL_FEED2.txt, figure 2.16*C*) to the regular WEL input structure (figure 2.17*A*). The feed file that is specified, WEL_FEED_1.txt (figure 2.17*B*), automatically adds the number of wells specified in it to the maximum number of wells in use during any stress period (MXACTW) and automatically includes the rates for each stress period. Note that in this version, **BUDGET_GROUPS** is not in use, so the package information uses the default single group name of "WELLS" in the LIST volumetric budget and cell-by-cell file.

A

```
#
Layer Row Column [BGROUP] [xyz]   # Repeat input as needed,
#                                   one per uncommented line
#                                   until "TEMPORAL INPUT" keyword
```

B

Layer	is the layer number of the model cell that contains the well perforations.
ROW	is the row number of the model cell that contains the well.
COLUMN	is the column number of the model cell that contains the well.
BGROUP	is the budget group name, up to 16 characters long. It is only specified if the **BUDGET_GROUPS** block has been declared and must be one of the defined BGROUP names. See appendix 3 for more details.
xyz	represents any auxiliary variables for a well that has been defined in the options block.

Figure 2.15. *A*, Well (WEL) package spatial input structure for LineFeed alternative input format, and *B*, explanation of input. [Comments are any text that are written to the right of a # symbol. Optional input is enclosed in brackets, such as [BGROUP].]

A

```
# Well Package Input
#
BEGIN BUDGET_GROUPS   # Optional block that allows well pumpage to be aggregated into
  #                     volumetric budget groups other than the default name "WELLS"
  #                     Each group contains its own IN and OUT volumetric budget
  #                     See appendix 3, "Budget Groups," for more details
  WEL_GRP1
  WEL_GRP2
END
#
BEGIN LINEFEED
  ./WEL_FEED_1.txt  BUFFER 64            # Optional Buffering from Generic_Input
  ./WEL_FEED_2.txt  SF 0.3048 SF 86400   # SF converts cfs to cmd
END
#
BEGIN OPTIONS          # WEL Options block (optional to include)
  WELL_CBC  44         # Sets IWELCBC input (CBC output unit)
END
# No more input necessary, but a mixture of input is allowed.
# Could optionally specify "ITMP NP", the regular stress period input for the WEL package.
```

B

```
# Feed File: WEL_FEED1.txt
# Spatial Section
1  4  4  WEL_GRP1 # WEL_1
1  3  5  WEL_GRP1 # WEL_2
3  2  8  WEL_GRP2 # WEL_3
#
TEMPORAL INPUT
#
#  WEL_1  WEL_2   WEL_3
  -55.0    NaN  -100.0  # SP 1
  -35.0    NaN  -150.0  # SP 2
  -25.0  -75.0  -200.0  # SP 3
```

C

```
# Feed File: WEL_FEED2.txt
#
# Spatial Section
3  4  4  WEL_GRP2 # WEL_4
3  3  5  WEL_GRP2 # WEL_5
#
TEMPORAL INPUT
#
#  WEL_4   WEL_5
 -155.0   100.0    # SP 1
 -135.0     0.0    # SP 2
    NaN     NaN    # SP 3
```

Figure 2.16. Example WEL package input that uses the LineFeed alternative input format (LineFeed) and associated "feed files" that define five wells for three stress periods (SP). *A*, The WEL package input. *B*, The feed file WEL_FEED1.txt that is read as part of the WEL package input. *C*, The feed file WEL_FEED2.txt that is read as part of the WEL package input. [Comments are any text that are written to the right of a # symbol. This example defines the entire WEL package with the block input (including LineFeed), but a mixture of the original input and LineFeed input is allowed.]

A

```
# Well Package Input                                    – Note input not using Budget Groups
#                                                         "WELLS" budget group is the default
BEGIN LINEFEED
  ./WEL_FEED_1_NO_BGROUP.txt
END
#
BEGIN OPTIONS          # WEL Options block (optional to include)
  NOPRINT
END
#
2  44 # MXACTW  IWELCB  –  The LineFeed Wells are auto-added to MXACTW
2  0                  # ITMP  NP  SP 1  –  Auto-Applies WEL_FEED_1_NO_BGROUP.txt Wells
    3  4  4  -155.0 # WEL_4  (Layer, Row, Column, Q)
    3  3  5   100.0 # WEL_5
2  0                  # ITMP  NP  SP 2  –  Auto-Applies WEL_FEED_1_NO_BGROUP.txt Wells
    3  4  4  -135.0 # WEL_4
    3  3  5     0.0 # WEL_5
0  0                  # ITMP  NP  SP 3  –  Auto-Applies WEL_FEED_1_NO_BGROUP.txt Wells
```

B

```
# Feed File: WEL_FEED_1_NO_BGROUP.txt
# Spatial Section
1  4  4  WEL_GRP1 # WEL_1
1  3  5  WEL_GRP1 # WEL_2
3  2  8  WEL_GRP2 # WEL_3
#
TEMPORAL INPUT
#
#  WEL_1  WEL_2   WEL_3
  -55.0    NaN  -100.0  # SP 1
  -35.0    NaN  -150.0  # SP 2
  -25.0  -75.0  -200.0  # SP 3
```

Figure 2.17. Example WEL package input for three stress periods (SP) that uses regular input structure to define two wells and the LineFeed alternative input format (LineFeed) to define three wells (SP). *A*, The WEL package input. *B*, The feed file WEL_FEED1.txt that is read as part of the WEL package input. [Comments are any text that are written to the right of a # symbol.]

LineFeed—GHB Package Input

The GHB package input with LineFeed is nearly identical to the WEL package LineFeed, presented in the previous section, but includes boundary conductance (Cond) as part of its spatial input and the temporal input specifies the boundary head (BHEAD). In cases where the conductance does not change with time, the GHB LineFeed can completely replace the spatial and temporal input or supplement an already existing input structure. Similar to the WEL package, if GHB uses LineFeed, then it must include **BEGIN LINEFEED** at the start of the package input and use *Block Style* input to read all the feed files. The order of any other *Block Style* input in the GHB does not matter, but all the blocks should appear before any of the regular GHB input (such as MXACTB and IGHBCB).

Figure 2.18 is the general input format of the GHB feed file spatial input section. The temporal input section of the feed file contains the boundary head (Bhead) for each stress period. If a boundary cell is not used for that stress period, then its Bhead is set to NaN. This is important because substituting a value of zero assigns to the boundary cell an elevation of 0.0 model units.

Figure 2.19 is an example of GHB package input that uses the *Block Style* input to define all the package's input and options. Figure 2.19*A* includes the LineFeed input block that loads two feed files (fig. 2.19*B* and *C*) that define a total of five general head boundary cells. The general head boundary cell GHB_2 (fig. 2.19*B*, SP 1 input row) is has its Bhead set to NaN, so it is not simulated in stress period 1. In this example, stress period 1 (SP 1) includes in the simulation 4 GHB cells, and stress period 2 (SP 2) and 3 (SP 3) both include in the simulation 3 GHB cells (same total count but at different locations). Figure 2.19 also includes the **BUDGET_GROUP** feature described in the Budget Group section in appendix 3. This feature allows the budget information that is printed to the LIST file and cell-by-cell file to be refined into custom groups rather than aggregating into one value for the entire package. If the **BEGIN BUDGET_GROUPS** is not present or is empty, the group name, BGROUP, is ignored.

A

```
#
Layer Row Column Cond [BGROUP] [xyz]  # Repeat input as needed,
#                                       one per uncommented line
#                                       until "TEMPORAL INPUT" keyword
```

B

Layer	is the layer number of the cell affected by the head-dependent boundary.
ROW	is the row number of the cell affected by the head-dependent boundary.
COLUMN	is the column number of the cell affected by the head-dependent boundary.
Cond	is the hydraulic conductance of the interface between the aquifer cell and the boundary condition.
BGROUP	is the budget group name, up to 16 characters long. It is only specified if the **BUDGET_GROUPS** block has been declared and must be one of the defined BGROUP names. Please see appendix 3 for more details.
xyz	represents any auxiliary variables for a well that has been defined in the options block.

Figure 2.18. *A*, General Head Boundary (GHB) package spatial input structure for LineFeed alternative input format, and *B*, explanation of input. [Comments are any text that are written to the right of a # symbol. Optional input is enclosed in brackets, such as [BGROUP].]

A

```
# GHB Package Input
#
BEGIN BUDGET_GROUPS   # Optional block that allows GHB boundary flow budgets to be
  #                     aggregated into volumetric budget groups other
  #                     than the default name of "HEAD DEP BOUNDS"
  #                     Each group contains its own IN and OUT volumetric budget
  #                     See appendix 3, "Budget Groups," for more details
  GHB_SEA
  GHB_LAND
END
#
BEGIN LINEFEED
  ./GHB_FEED_1.txt  BUFFER 64          # Optional Buffering from Generic_Input
  ./GHB_FEED_2.txt  SF 0.3048          # SF converts feet to meter
END
#
BEGIN OPTIONS          # GHB Options block (optional to include)
  GHB_CBC  44          # Sets IGHBCBC input (CBC output unit)
END
# No more input necessary, but a mixture of input is allowed.
# Could optionally specify "ITMP NP", the regular stress period input for the GHB package.
```

B

```
# Feed File: GHB_FEED1.txt
# Spatial Section
1  4  4 100. GHB_SEA # GHB_1
1  3  5 100. GHB_SEA # GHB_2
1  2  8 100. GHB_SEA # GHB_3
#
TEMPORAL INPUT
#
#  GHB_1  GHB_2   GHB_3
    5.5     NaN   -0.10   # SP 1
    5.8     NaN   -0.10   # SP 2
    5.9    1.05   -0.02   # SP 3
```

C

```
# Feed File: GHB_FEED2.txt
#
# Spatial Section
3  4  4 250. GHB_LAND # GHB_4
3  3  5 250. GHB_LAND # GHB_5
#
TEMPORAL INPUT
#
#  GHB_4   GHB_5
   55.0    50.0     # SP 1
   65.0     NaN     # SP 2
    NaN     NaN     # SP 3
```

Figure 2.19. Example GHB package input that uses the LineFeed alternative input format (LineFeed) and associated "feed files" that define five boundary condition cells for three stress periods (SP). *A*, The GHB package input. *B*, The feed file GHB_FEED1.txt that is read as part of the GHB package input. *C*, The feed file GHB_FEED2.txt that is read as part of the GHB package input. [Comments are any text that are written to the right of a # symbol. This example defines the entire GHB package with the block input (including LineFeed), but a mixture of the original input and LineFeed input is allowed.]

Figure 2.20 presents an example GHB input that includes a mixture of the regular package input and LineFeed. In this example, the feed file used in figure 2.19*C* (GHB_FEED2.txt) is translated to the regular GHB input format (fig. 2.20). The feed file that is specified, GHB_SEA_FEED.txt, automatically adds the number of boundary cells specified in it to the maximum number of general head boundary cells in use for any stress period (MXACTB) and automatically includes the rates for each stress period. For example, the number of GHB cells in use for stress period 1 is four, with two being defined in the GHB standard input and two from the feed file. It should be noted that in figure 2.20*A*, MXACTB is specified as 2 to indicate the maximum number of boundary cells defined within the regular GHB input. After the LineFeed feed files are loaded, MXACTB is increased to 5 to account for the three additional boundary cells defined in the feed files. Note that in this example, **BUDGET_GROUPS** is not in use, so the feed files do not include the input BGROUP and the GHB package uses the default group name of "HEAD DEP BOUNDS" in the LIST volumetric budget and cell-by-cell file.

A

```
# GHB Package Input                    – Note input not using Budget Groups
#                                        "HEAD DEP BOUNDS" budget group is the default
BEGIN LINEFEED
  ./GHB_SEA_FEED.txt
END
#
BEGIN OPTIONS          # GHB Options block (optional to include)
  NOPRINT
END
#
2  44 # MXACTB  IGHBCBC  –  The LineFeed GHB's are auto-added to MXACTB
2  0                          # ITMP  NP  SP 1  –  Auto-Applies GHB_SEA_FEED.txt GHB's
    3  4  4  55.0  250.0      # GHB_4  (Layer, Row, Column, BHead, Cond)
    3  3  5  50.0  250.0      # GHB_5
1  0                          # ITMP  NP  SP 2  –  Auto-Applies GHB_SEA_FEED.txt GHB's
    3  4  4  65.0  250.0      # GHB_4
0  0                          # ITMP  NP  SP 3  –  Auto-Applies GHB_SEA_FEED.txt GHB's
```

B

```
# Feed File: GHB_SEA_FEED.txt
# Spatial Section
1  4  4 100.   # GHB_1
1  3  5 100.   # GHB_2
1  2  8 100.   # GHB_3
#
TEMPORAL INPUT
#
#  GHB_1  GHB_2   GHB_3
    5.5     NaN   -0.10   # SP 1
    5.8     NaN   -0.10   # SP 2
    5.9    1.05   -0.02   # SP 3
```

Figure 2.20. Example GHB package input for three stress periods (SP) that uses regular input structure to define two GHB cells and the LineFeed alternative input format (LineFeed) to define three GHB cells. *A*, The GHB package input. *B*, The feed file GHB_SEA_FEED.txt that is read as part of the GHB package input. [Comments are any text that are written to the right of a # symbol.]

LineFeed—MNW2 Package Input

The MNW2 package input with LineFeed differs from other packages in that the MNW2 input is not entirely defined in the feed files. Specifically, the well construction must be specified as part of the normal MNW2 input file (the part before `ITMP`). As of 2020, LineFeed did not allow wells that have time-varying hydraulic head boundaries (`Qlimit` set to a value less than 0). LineFeed also has limited support for specifying a discharge based on the pump's lift capacity (`PUMPCAP` set to value greater than 0). If `PUMPCAP` is greater than zero, then the multiplier for implementing head-capacity relations (`CapMult`) is always set to 1.0 (no modification of the original table).

The LineFeed block-style input specifications are at the beginning of the MNW2 input file and can be in any order relative to the other block-style inputs. For MNW2, the feed file spatial-input section specifies the name of one MNW2 well (`WELLID`) for each row; well names also are defined in the main MNW2 input file that specifies the well structure. The temporal-input section then defines the desired pumping rate (`Qdes`) for the stress period. MNW2 simulates groundwater flow through the well between model layers (intraborehole flow), even when `Qdes` is zero, so the null input value, NaN, should always be used if a well does not exist. It is recommended to always specify NaN for stress periods before the well is drilled and after it is destroyed and to set `Qdes` to zero if the well is not in use.

Figure 2.21 is an example of MNW2 input that uses the LineFeed input format to specify the well pumping rate for three stress periods. This example simulates four wells that have their screen interval and well construction defined in the MNW2 main input (fig. 2.21*A*) and their pumping rates defined in the LineFeed file (fig. 2.21*B*). The MNW2 wells `MNW2WELL_1` and `MNW2WELL_4` are simulated with pumping rates for all three stress periods. In the MNW2 feed file (fig. 2.21*B*), `MNW2WELL_3` specifies `Qdes` for the three stress periods as NaN, 0.0, and –200.0, respectively. This indicates that `MNW2WELL_3` is not included in the simulation during stress period 1; is included in the simulation during stress period 2 with no pumping, but may include intraborehole flow; `and` is included in the simulation during stress period 3 with a desired rate of –200 L^3/T (negative indicates extraction pumping), which may also include intraborehole flow.

The user can have wells defined in the normal MNW2 stress period input (`ITMP`) in addition to the LineFeed file as well. Mixing the MNW2 regular stress period input with LineFeed is not recommended, but may be necessary if LineFeed is incorporated to existing input files to include new wells. To illustrate a mixture of the two inputs, figure 2.22 uses the previous LineFeed file example (fig. 2.21) but shifts the LineFeed `Qdes` for wells `MNW2WELL_1` and `MNW2WELL_3` to the regular MNW2 input.

A

```
# MNW2 Package Input
BEGIN LINEFEED
  ./MNW_FEED.txt
END
# MNW Construction Part
4 0 2                    # MNWMAX   IWL2CB   MNWPRNT
MNW2WELL_1  4            # WELLID   NNODES
  SKIN  0 1 0 0          # LOSSTYPE   PUMPLOC   Qlimit   PPFLAG   PUMPCAP
  0.1  0.35  1.6683      # Rw   Rskin   Kskin
     1  6   2            # LAY   ROW   COL
     3  6   2
     5  6   2
     7  6   2
    205.  0              # Hlim QCUT
MNW2WELL_2  2
  SKIN  0 1 0 0
  0.1  0.35  1.6683
     1  8   2
     3  8   2
    205.  0
MNW2WELL_3  2
  SKIN  0 1 0 0
  0.1  0.35  1.6683
     1  10  2
     3  10  2
    205.  0
MNW2WELL_4  1
  SKIN  0 0 0 0
  0.1  0.35  1.1122
     3  3   19
# ITMP - No more input necessary.
# A mixture of input with using ITMP > 0 is allowed, but not recommended
```

Figure 2.21. Example MNW2 package input for three stress periods (SP) that uses regular input to define the well screen and structure for four wells and uses the LineFeed alternative input format (LineFeed) to specify MNW2 desired pumping rate (Qdes). *A*, The MNW2 package input. *B*, The feed file MNW_FEED.txt that is read as part of the MNW2 package input. [Comments are any text that are written to the right of a # symbol.]

B

```
# Feed File: MNW_FEED.txt
#
# Spatial Section        – Define by referencing a WELLID name from the main input
#
MNW2WELL_1
MNW2WELL_2
MNW2WELL_3
MNW2WELL_4
#
TEMPORAL INPUT       #   – Specify each MNW2 well's Qdes
#
#  MNW2WELL_1   MNW2WELL_2   MNW2WELL_3   MNW2WELL_4
         -55.0          NaN          NaN        -50.0   # SP 1
         -35.0          NaN          0.0        -50.0   # SP 2
         -25.0          0.0       -200.0       -150.0   # SP 3
```

Figure 2.21. —Continued

A

```
# MNW2 Package Input
BEGIN LINEFEED
  ./MNW_FEED2.txt
END
# MNW Construction Part
4 0 2                    # MNWMAX   IWL2CB   MNWPRNT
MNW2WELL_1  4            # WELLID   NNODES
  SKIN  0 1 0 0          # LOSSTYPE   PUMPLOC   Qlimit   PPFLAG   PUMPCAP
  0.1  0.35  1.6683      # Rw   Rskin   Kskin
     1  6   2            # LAY   ROW   COL
     3  6   2
     5  6   2
     7  6   2
    205.  0              # Hlim QCUT
MNW2WELL_2  2
  SKIN  0 1 0 0
  0.1  0.35  1.6683
     1  8   2
     3  8   2
    205.  0
MNW2WELL_3  2
  SKIN  0 1 0 0
  0.1  0.35  1.6683
     1  10  2
     3  10  2
    205.  0
MNW2WELL_4  1
  SKIN  0 0 0 0
  0.1  0.35  1.1122
     3  3   19
1                    # ITMP          SP 1  –  Auto-Applies MNW_FEED2.txt Wells
MNW2WELL_1  -55.0    # WELLID   Qdes
2                    # ITMP          SP 2  –  Auto-Applies MNW_FEED2.txt Wells
MNW2WELL_1  -35.0
MNW2WELL_3    0.0
2                    # ITMP          SP 3  –  Auto-Applies MNW_FEED2.txt Wells
MNW2WELL_1  -25.0
MNW2WELL_3 -200.0
```

Figure 2.22. Example MNW2 package input for three stress periods (SP) that uses regular input to define the well screen and structure for four wells, uses the regular input to define the stress period input for two wells, and uses the LineFeed alternative input format (LineFeed) to specify MNW2 desired pumping rate (Qdes) for two wells. *A*, The MNW2 package input. *B*, The feed file MNW_FEED2.txt that is read as part of the MNW2 package input.

B

```
# Feed File: MNW_FEED2.txt
#
# Spatial Section    – Define by referencing a WELLID name from the main input
#
MNW2WELL_2
MNW2WELL_4
#
TEMPORAL INPUT       #   – Specify each MNW2 well's Qdes
#
#  MNW2WELL_2   MNW2WELL_4
           NaN         -50.0   # SP 1
           NaN         -50.0   # SP 2
           0.0        -150.0   # SP 3
```

Figure 2.22. —Continued

LineFeed—SFR Package Input

The SFR package input cannot be replaced entirely with LineFeed; instead LineFeed will only overwrite the inflow or diversion rate (`FLOW`) of specified segments. Functionally, the SFR LineFeed is identical to TabFiles, except it reads the SFR flows every stress period from the feed files. The LineFeed *Block Style* input (**`BEGIN LINEFEED`**) is specified at the start of the SFR input file. The order of any other *Block Style* input in the SFR does not matter, but all the blocks should appear before any of the regular SFR input—which is `NSTRM`, `NSS`, `NSFRPAR`, `NPARSEG`, `CONST`, `DLEAK`, `ISTCB1`, and `ISTCB2`. For the SFR, the feed file spatial-input section specifies the segment number (`ISEG`) for each row; segments correspond to the columns of the temporal section that specify the inflow rates for the segments. Since LineFeed only modifies the existing SFR input, the null value, NaN, is converted to zero before it overwrites `FLOW`. Figure 2.23 is an example SFR input file that includes only the *Block Style* inputs and a feed file that overwrites the inflow for segments 22 and 41.

A

```
# SFR Package Input      – Note example does not include main SFR input
#
# Various block inputs   – Block order does not matter
#
BEGIN LINEFEED
  ./SFR_FEED.txt
END
#
BEGIN OPTIONS
              NOPRINT
              FIX_STREAM_BOTTOM
END
#
# Remaining regular input for SFR starting with
# NSTRM   NSS   NSFRPAR   NPARSEG   CONST   DLEAK   ISTCB1   ISTCB2
```

B

```
# Feed File: SFR_FEED.txt
#
# Spatial Section          – Define ISEG that has FLOW value overwritten
#
22   # SFR Segment 22
41   # SFR Segment 41
#
TEMPORAL INPUT
#
#  ISEG_22     ISEG_41
    100.0       125.0   # SP 1
      0.0        55.0   # SP 2
      0.0        75.0   # SP 3
```

Figure 2.23. Partial example SFR package input that uses LineFeed alternative input format (LineFeed) to specify two segment's FLOW input. *A*, The partial SFR package input, which only includes the LineFeed and Options blocks. *B*, The feed file, SFR_FEED.txt, that is read as part of the SFR package input. [Comments are any text that are written to the right of a # symbol.]

Transient File Reader—Spatial-Temporal Input

Separation of spatial data by time is most useful for compiling and managing input that require large data streams of spatially distributed, temporally varying, input data (such as climate and land-use data). This is facilitated with the *Transient File Reader* (TFR, appendix 1), which analogous to a pointer file for a single input property that directs how each stress period's input is handled. The TFR allows spatial and temporal data to be separate, which, for example, makes it easier to build and manage climate data derived from downscaled Global Climate Model (GCM)-simulated data or from common land-use models. Each input that is read by a TFR can optionally include a set of advanced scale factors, **SFAC** and **SF** (appendix 1), that help parameterize input, provide unit conversion, or adjust idealized properties to non-idealized conditions. Often, input properties—such as reference evapotranspiration or crop coefficients—are reported for idealized conditions and require adjustment related to regional effects, climate variability, or improving the fit of observations.

A TFR reads each stress period input with the *Universal Loader* (ULOAD, appendix 1). ULOAD checks for a directive keyword, such as **RELOAD** or **LOAD_NEXT**, or identifies the input location with *Generic_Input*, such as **OPEN/CLOSE** or **EXTERNAL**. The structure of TFR allows the temporal series of input to be split among multiple files rather than interwoven as different property inputs (as in traditional MODFLOW). For an in-depth description of the *Transient File Reader* and its input structure, please refer to appendix 1.

There are two major reasons for the development of the TFR. The first is to simplify temporal input by breaking down each property into its own file. The second is to enable the use of scale factors that can be linked to a calibration software. The simplified input allows spatial arrays to be easily specified as either a set of text files or one large contiguous binary file that is cycled through for each stress period.

Figure 2.24 is an example TFR input for the Farm Process (FMP), which includes the keywords that indicate the location of the TFR (fig. 2.24*A* and *B*). Figure 2.24*C* is an example TFR file that reads precipitation arrays for 15 stress periods—the precipitation arrays are stored in separate files named by the year and month they represent. The units of precipitation are converted from inches per day to meters per day with the post-keyword "**SF** `0.0254`". The TFR file itself (Precip_TFR.txt) is opened with *Generic_Input_OptKey*, which supports scale factors (**SF** SCALE, appendix 1) and supports reading scale factors. If the scale factor is applied to the TFR, it is globally applied to all input loaded by the TFR. This makes it advantageous for evaluating scenarios. For example, the effect of a 20 percent reduction in precipitation from normal can be simulated by simply adding a scale factor to the TFR file. Figure 2.24*B* illustrates how to reduce precipitation by 20 percent by scaling the TFR input with the post-keyword "**SF** `0.8`".

Each time the TFR loads data, it uses ULOAD (appendix 1). Because ULOAD allows for multiple scale factors (which come from *Generic_Input*), the user can easily modify the input to evaluate different precipitation scenarios. For example, the TFR in figure 2.24 evaluates a scenario where the winter months (January–March) have twice the precipitation and the summer months (July–September) have half by adding appropriate scale factors.

A MF-OWHM2 model can be calibrated using separate model-independent parameter estimation (or uncertainty analysis) software that uses a "template file" to construct model input files. The template file is a copy of the model input file, but contains tagged keywords for the software's optimizer to replace as it builds the input file. The TFR makes an excellent template file because it has a simple structure.

For example, the Farm Process package (FMP) supports specifying crop coefficients (Kc) spatially with the keyword **ARRAY** (*List-Array Input*, see appendix 1). The spatial input of crop coefficients could be linked to remotely sensed information to determine the spatial variation of crop evapotranspiration. The dataset could require a correction, however, or minor adjustment in calibration to account for differences in the simulation software; to compensate for bias in the original dataset; or to adjust for geographic location (for example, a coastal crop's actual Kc may differ from the same crop located inland of the ocean).

Figure 2.26 is an example FMP input that illustrates how to use advanced scale factors. Figure 2.26*A* is a partial input example from FMP that contains the keywords that define the crop location and crop coefficient. The crop coefficients are specified with a TFR (fig. 2.26*B*) that reads input for two stress periods. The second stress period input includes the **SFAC** keyword (advanced scale factors, appendix 1). **SFAC** makes use of the **DIMKEY** keyword "**ByCrop**" to indicate that NCROP scale factors are applied wherever the respective crops are located. This example loads the crop-location array and uses it for the entire simulation (fig. 2.26*C*). In this example, corn is the crop grown in the upper left corner of the model, onions are grown in the lower right, and native vegetation everywhere else. This crop-ID location array (fig. 2.26*C*) is used for identifying which crop coefficient (fig. 2.26*D*), specified in the NROW by NCOL array, corresponds with which crop. It also serves as a masking array for any scale factors that use the **SFAC** **DIMKEY** "**ByCrop**", which would load NCROP scale factors and apply them by CID. For example, "**SFAC ByCrop** `2. 1. 1.`" multiplies everywhere corn (crop 1) grows by 2 and multiplies onions and native vegetation by 1. Figure 2.26*E* is the final result of combining the scale factor "**SFAC ByCrop** `1.1 1.2 1.0`" with the input read from file "2000_06_Kc.txt" (fig. 2.26*C*).

A

```
# Farm Process (FMP) input keywords that specify a Transient File Reader (Precip_TFR.txt)
# This file specifies the landscape precipitation for each stress period.
#
PRECIPITATION  TRANSIENT  ARRAY ./Precip_TFR.txt
```

B

```
# Farm Process (FMP) input keywords that specify a Transient File Reader (Precip_TFR.txt)
# This file specifies the landscape precipitation for each stress period.
# The Generic_Input post-keyword SF indicates each array that is read is scaled by 0.8
#
PRECIPITATION  TRANSIENT  ARRAY ./Precip_TFR.txt  SF 0.8
```

C

```
# Transient File Reader: Precip_TFR.txt
# This file specifies the precipitation rate (L/T) for each stress period. Model Units are L/T  = m/d.
#
# Each OPEN/CLOSE file contains precipitation (in/day) as a NROW by NCOL array
#
OPEN/CLOSE ./Precip/2000_01.txt  SF 0.0254     #  JAN-2000   SP  1
OPEN/CLOSE ./Precip/2000_02.txt  SF 0.0254     #  FEB-2000   SP  2
OPEN/CLOSE ./Precip/2000_03.txt  SF 0.0254     #  MAR-2000   SP  3
OPEN/CLOSE ./Precip/2000_04.txt  SF 0.0254     #  APR-2000   SP  4
OPEN/CLOSE ./Precip/2000_05.txt  SF 0.0254     #  MAY-2000   SP  5
OPEN/CLOSE ./Precip/2000_06.txt  SF 0.0254     #  JUN-2000   SP  6
OPEN/CLOSE ./Precip/2000_07.txt  SF 0.0254     #  JUL-2000   SP  7
OPEN/CLOSE ./Precip/2000_08.txt  SF 0.0254     #  AUG-2000   SP  8
OPEN/CLOSE ./Precip/2000_09.txt  SF 0.0254     #  SEP-2000   SP  9
OPEN/CLOSE ./Precip/2000_10.txt  SF 0.0254     #  OCT-2000   SP 10
OPEN/CLOSE ./Precip/2000_11.txt  SF 0.0254     #  NOV-2000   SP 11
OPEN/CLOSE ./Precip/2000_12.txt  SF 0.0254     #  DEC-2000   SP 12
OPEN/CLOSE ./Precip/2001_01.txt  SF 0.0254     #  JAN-2001   SP 13
OPEN/CLOSE ./Precip/2001_02.txt  SF 0.0254     #  FEB-2001   SP 14
OPEN/CLOSE ./Precip/2001_03.txt  SF 0.0254     #  MAR-2001   SP 15
```

Figure 2.24. *Transient File Reader* (TFR) use examples for reading Farm Process (FMP) precipitation. *A*, FMP precipitation input keywords that specify the location of a TFR. *B*, FMP precipitation input keywords that specify the location of a TFR and scales by 0.8 any input read. *C*, TFR that reads precipitation input for 16 stress periods (SP) and scales each input by 0.0254 as a unit conversion. [Comments are any text that are written to the right of a # symbol. Note that if *B* opens *C*, then the resulting scale factor is "0.8 × 0.0254"]

```
# Transient File Reader: Precip_TFR_ScaleFactor.txt
# This file specifies the precipitation rate (L/T) for each stress period. Model Units are L/T  = m/d.
#
# Each OPEN/CLOSE file contains precipitation (in/day) as a NROW by NCOL array
#
OPEN/CLOSE ./Precip/2000_01.txt  SF 0.0254  SF 2.0   #  JAN-2000  SP  1
OPEN/CLOSE ./Precip/2000_02.txt  SF 0.0254  SF 2.0   #  FEB-2000  SP  2
OPEN/CLOSE ./Precip/2000_03.txt  SF 0.0254  SF 2.0   #  MAR-2000  SP  3
OPEN/CLOSE ./Precip/2000_04.txt  SF 0.0254          #  APR-2000  SP  4
OPEN/CLOSE ./Precip/2000_05.txt  SF 0.0254          #  MAY-2000  SP  5
OPEN/CLOSE ./Precip/2000_06.txt  SF 0.0254          #  JUN-2000  SP  6
OPEN/CLOSE ./Precip/2000_07.txt  SF 0.0254  SF 0.5   #  JUL-2000  SP  7
OPEN/CLOSE ./Precip/2000_08.txt  SF 0.0254  SF 0.5   #  AUG-2000  SP  8
OPEN/CLOSE ./Precip/2000_09.txt  SF 0.0254  SF 0.5   #  SEP-2000  SP  9
OPEN/CLOSE ./Precip/2000_10.txt  SF 0.0254          #  OCT-2000  SP 10
OPEN/CLOSE ./Precip/2000_11.txt  SF 0.0254          #  NOV-2000  SP 11
OPEN/CLOSE ./Precip/2000_12.txt  SF 0.0254          #  DEC-2000  SP 12
OPEN/CLOSE ./Precip/2001_01.txt  SF 0.0254  SF 2.0   #  JAN-2001  SP 13
OPEN/CLOSE ./Precip/2001_02.txt  SF 0.0254  SF 2.0   #  FEB-2001  SP 14
OPEN/CLOSE ./Precip/2001_03.txt  SF 0.0254  SF 2.0   #  MAR-2001  SP 15
```

Figure 2.25. *Transient File Reader* (TFR) example for reading Farm Process (FMP) precipitation with two scale factors. This example doubles the precipitation from January to March and halves the precipitation from July to September. [Comments are any text that are written to the right of a # symbol.]

In practice, **SFAC** is used as a tool in the TFR for calibration. Figure 2.27 is an example TFR that is set up as a calibration template file that makes use of **SFAC** scaling and opens crop-coefficient arrays, such as 2000_06_Kc.txt in figure 2.26*D*. This example uses the "@" symbol (indicated by "jtf @") to delineate parameter names that are replaced by the calibration software with a number. This set up allows for the calibration file to rely on the TFR as the template and not the raw input data, which should not require any future changes. In this example, figure 2.27, the calibration software scales the crop coefficient for crops 1 and 2 and does not scale crop 3.

Scale factors can be specified in a separate file. This has the advantage of keeping the calibration template file separate from the raw data input. The **SFAC** keyword can be combined with the keywords **EXTERNAL**, **DATAUNIT**, or **DATAFILE** (appendix 1) to read scale factors from a separate file. Figure 2.28 is an example that separates scale factors loaded from a separate file. Figure 2.28*A* is another example TFR file, Kc_TFR2.txt, but rather having the TFR be the template, this Kc_TFR2.txt contains a reference to a file (KC_SF.txt) that is made from a template (KC_SF.tpl). In figure 2.28, only the scale factors themselves are required to be in a template; the TFR is a master file that specifies the raw crop-coefficient data and the file that is made from the **SFAC** template. This allows for separating the spatial raw input, the temporal discretization, and the calibration scale factors into three sets of files.

Often properties are assigned on an annual basis and are repeated through a simulation. This cycling can be extended to include a single annual file that is cycled through according to the appropriate commands in the TFR. Often the annual file is made unique to the year simulated by additional scale factors that act as corrections to non-idealized conditions. For example, crop coefficients often have a correction factor based on wet or dry precipitation years (for more details, see the "Consumptive-Use Stress Factor" section in appendix 4) and another correction based on a longer climate cycle (for example, El Niño Southern Oscillation or Pacific Decadal Oscillation).

Figure 2.29 is an example that mimics figure 2.26, except the simulation is extended to 13 stress periods and uses a *List Style* input (appendix 1) to read the 3 crop coefficients. The first 12 stress periods are assumed to have a wet climate cycle to a dry cycle starting on stress period 13. *List Style* input reads a set of records (fig. 2.29*C*) that are matched to an integer masking array. For the case of crop coefficients with three crops, the first record (ID = 1) is placed everywhere there is a 1 in the crop-ID location array (for example, fig. 2.26*C*), and the second record where there is a 2, and the third record where there is a 3.

A

```
# Partial Farm Process (FMP) input for the LAND_USE input block
#
# NPER   = 2 stress periods  –          Total number of stress periods
# NROW = 3 rows              – Model row dimension
# NCOL   = 5 columns         – Model column dimension
# NCROP = 3 crops            – Number of FMP simulated crops identified as 1, 2, and 3.
#                              If a crop is identified as 0, then it is treated fallowed land
#
# Crop ID Location Array     –  NROW by NCOL array specifies integers 0, 1 , 2, and 3
#
# STATIC indicates to read ARRAY once with the Universal Loader (ULOAD)
#
LOCATION   STATIC   ARRAY   OPEN/CLOSE ./CID.txt
#
# Crop Coefficient (Kc)      –  NROW by NCOL array that is Kc for each model location
#
# TRANSIENT indicates that input opens a Transient File Reader (TFR) that
#    for each stress period directs how to read the ARRAY input
#
CROP_COEFICIENT   TRANSIENT   ARRAY   OPEN/CLOSE    Kc_TFR.txt
```

B

```
# File: Kc_TFR.txt           – Transient File Reader (TFR)
#
# TFR directs how to loads Crop Coefficients (Kc) and optionally apply scale factors (SFAC)
#
# Open file 2000_06_Kc.txt, read contents and apply to model
#
OPEN/CLOSE   2000_06_Kc.txt                  # SP 1
#
# SFAC is the keyword for advanced scale factors that are applied to the next stress period
#    The second keyword “ByCrop” indicates that NCROP (3) scale factors are read
#    and that the first number is applied to Crop 1, the second to Crop 2 and third to Crop 3.
#
SFAC ByCrop   1.1   1.2   1.0               # Scale next input
OPEN/CLOSE   2000_06_Kc.txt                  # SP 2
```

Figure 2.26. Sample input from the Farm Process (FMP) that uses a *Transient File Reader* (TFR) with advanced scale factors (SFAC) to scale a model grid array of crop coefficients (Kc). *A*, Partial FMP input from the LAND_USE input block that defines the crop locations and Kc. *B*, TFR file specifies for two stress periods (SP) the Kc input. *C*, The crop location array. *D*, The Kc array. *E*, Input used in the simulation by FMP for SP 2. [Comments are any text that are written to the right of a # symbol.]

C

```
# File: CID.txt  – Crop ID Location Array
#
# Crop:
#       1  =  Corn
#       2  =  Onion
#       3  =  Native Vegetation
#
1  1  3  3  3
1  1  2  2  2
3  3  2  2  2
```

D

```
# File: 2000_6_Kc.txt
#
# Crop Coefficient (Kc)
#
# for the model grid (3 by 5)
# for June-2000
#
1.13  1.14  1.36  1.36  1.35
1.15  1.15  0.85  0.80  0.75
1.40  1.37  0.84  0.82  0.79
```

E

```
# Final input after combining the advanced scale factor “SFAC ByCrop  1.1  1.2  1.0”
# with file “CID.txt” and with “2000_6_Kc.txt
#
# 1.1 multiplies Kc values coincident with Crop 1
# 1.2 multiplies Kc values coincident with Crop 2
# 1.0 multiplies Kc values coincident with Crop 3
#
1.24  1.25  1.36  1.36  1.35
1.27  1.27  1.02  0.96  0.90
1.40  1.37  1.01  0.98  0.95
```

Figure 2.26. —Continued

```
jtf  @
#       ^= declares which character is used as a delimiter in template
#
# File: Kc_TFR.tpl        – Template Transient File Reader for Crop Coefficients
#
#
# NPER   = 6 stress periods     – Total number of stress periods
# NCROP = 3 crops               – Number of FMP simulated crops identified as 1, 2, and 3.
#
# Crop Based Scale Factor:
#                     Crop 1 – Corn:    @KC_SF_CRP1@
#                     Crop 2 – Onion:   @KC_SF_CRP2@
#                     Crop 3 – Native Vegetation: Not Adjusted, by multiplying by 1.0
#
SFAC ByCrop @KC_SF_CRP1@  @KC_SF_CRP2@  1.0
OPEN/CLOSE  2000_01_Kc.txt                      # JAN-2000     SP 1
#
SFAC ByCrop @KC_SF_CRP1@  @KC_SF_CRP2@  1.0
OPEN/CLOSE  2000_02_Kc.txt                      # FEB-2000     SP 2
#
SFAC ByCrop @KC_SF_CRP1@  @KC_SF_CRP2@  1.0
OPEN/CLOSE  2000_03_Kc.txt                      # MAR-2000    SP 3
#
SFAC ByCrop @KC_SF_CRP1@  @KC_SF_CRP2@  1.0
OPEN/CLOSE  2000_04_Kc.txt                      # APR-2000     SP 4
#
SFAC ByCrop @KC_SF_CRP1@  @KC_SF_CRP2@  1.0
OPEN/CLOSE  2000_05_Kc.txt                      # MAY-2000    SP 5
#
SFAC ByCrop @KC_SF_CRP1@  @KC_SF_CRP2@  1.0
OPEN/CLOSE  2000_06_Kc.txt                      # JUN-2000     SP 6
#
```

Figure 2.27. Example *Transient File Reader* (TFR) that is set up as a calibration template file. [Comments are any text that are written to the right of a # symbol.]

A

```
#
# File: Kc_TFR2.txt      – Transient File Reader for Crop Coefficients
#
# NPER   = 6 stress periods     – Total number of stress periods
# NCROP = 3 crops               – Number of FMP simulated crops identified as 1, 2, and 3.
#
# The scale factors are specified using List Style input
# in the external file KC_SF.txt
#
# Each reference to "DATAFILE  KC_SF.txt" reads the next set of 3 scale factors in KC_SF.txt
#
SFAC ByCrop DATAFILE KC_SF.txt
OPEN/CLOSE  2000_01_Kc.txt                  # JAN-2000  SP  1
#
SFAC ByCrop DATAFILE KC_SF.txt
OPEN/CLOSE  2000_02_Kc.txt                  # FEB-2000  SP  2
#
SFAC ByCrop DATAFILE KC_SF.txt
OPEN/CLOSE  2000_03_Kc.txt                  # MAR-2000  SP  3
#
SFAC ByCrop DATAFILE KC_SF.txt
OPEN/CLOSE  2000_04_Kc.txt                  # APR-2000  SP  4
#
SFAC ByCrop DATAFILE KC_SF.txt
OPEN/CLOSE  2000_05_Kc.txt                  # MAY-2000  SP  5
#
SFAC ByCrop DATAFILE KC_SF.txt
OPEN/CLOSE  2000_06_Kc.txt                  # JUN-2000  SP  6
```

Figure 2.28. Example *Transient File Reader* (TFR) that refers to a separate scale factor file that set up as a calibration template file. *A*, The TFR file. *B*, The scale factor template file. [Comments are any text that are written to the right of a # symbol.]

B

```
jtf  @
#       ^= declares which character is used as a delimiter in template
#
# File: Kc_SF.tpl          – Template scale factor file that becomes KC_SF.txt
#
# Calibration software creates the file KC_SF.txt from this template
#     KC_SF.txt is the same as Kc_SF.tpl, except every word
#     enclosed by the @ delimiter is replaced with a number.
#
# Crop Based Scale Factor:
#                       Crop 1 – Corn:     @KC_SF_CRP1@
#                       Crop 2 – Onion:    @KC_SF_CRP2@
#                       Crop 3 – Native Vegetation: Not Adjusted, by multiplying by 1.0
#  ID      ScaleFactor
   1     @KC_SF_CRP1@     #  JAN-2000   SP  1
   2     @KC_SF_CRP2@
   3     1.0
   1     @KC_SF_CRP1@     #  FEB-2000   SP  2
   2     @KC_SF_CRP2@
   3     1.0
   1     @KC_SF_CRP1@     #  MAR-2000  SP  3
   2     @KC_SF_CRP2@
   3     1.0
   1     @KC_SF_CRP1@     #  APR-2000   SP  4
   2     @KC_SF_CRP2@
   3     1.0
   1     @KC_SF_CRP1@     #  MAY-2000  SP  5
   2     @KC_SF_CRP2@
   3     1.0
   1     @KC_SF_CRP1@     #  JUN-2000   SP  6
   2     @KC_SF_CRP2@
   3     1.0
```

Figure 2.28. —Continued

In figure 2.29, the TFR file (fig. 2.29*B*) acts as a master file that controls the application of scale factors and crop-coefficient data. Specifically, it cycles through a file that contains 12 months of crop coefficient data (Kc_Annual.txt, fig. 2.29*C*) and then reads another file that contains the climate scale factor (WET_DRY_YEAR_SF.txt). The file "WET_DRY_YEAR_SF.tpl" (fig. 2.29*D*) is a template file that is translated by a calibration software make the file WET_DRY_YEAR_SF.txt.

At the 13th stress period, the TFR reloads Kc_Annual.txt, moving the load point to the first line of the file and then loading again the January part of the annual file. Figure 2.29*D* is an example scale-factor file that applies a wet or dry year factor to each of the stress periods. Note that the integer ID is only used to keep track of the stress period (because it must be loaded as part of ULOAD *List Style* input, but does not serve any purpose). Figure 2.29*D* is the only file used by the calibration software as a template file; thus, the TFR and crop-coefficient files are kept independent of the calibration.

A

```
# Partial Farm Process (FMP) input for the LAND_USE input block
#
# NPER   = 13 stress periods    –  Total number of stress periods
# NCROP = 3 crops               –  Number of FMP simulated crops identified as 1, 2, and 3.
#
# Crop Coefficient (Kc)       –  List Style input that reads NCROP Kc records
#
# TRANSIENT indicates that input opens a Transient File Reader (TFR) that
#    for each stress period directs how to read the List Style input with NCROP records.
#
CROP_COEFICIENT   TRANSIENT   LIST   OPEN/CLOSE ./Kc_TFR3.txt
#
```

Figure 2.29. Sample input from the Farm Process (FMP) that uses a *Transient File Reader* (TFR) with advanced scale factors (SFAC) to scale all of crop coefficients (Kc) with a climate-based scale factor. *A*, Partial FMP input from the LAND_USE input block that defines Kc. List Style input (LIST) indicates input reads three Kc's (NCROP=3) and applies the first Kc everywhere that crop 1 resides, the second Kc everywhere crop 2 resides, and the third Kc everywhere crop 3 resides. *B*, TFR file that specifies 13 stress periods (SP) of the Kc input. *C*, List Style input of Kc for 12 stress periods. This file is cycled back to the start (RELOAD) on the thirteenth stress period. *D*, Climate-based scale factor template file for 13 stress periods. [Comments are any text that are written to the right of a # symbol.]

B

```
# File: Kc_TFR3.txt        – Transient File Reader for Crop Coefficients (Kc)
#
# NPER    = 13 stress periods     – Total number of stress periods
# NCROP  = 3 crops                – Number of FMP simulated crops identified as 1, 2, and 3.
#
# The scale factors        are using List Style input in the external file WET_DRY_YEAR_SF.txt
# The crop coefficients   are using List Style input in the external file Kc_Annual.txt
# SFAC does not include the optional DIMKEY (such as "SFAC ByCrop DATAFILE  CropScales.txt"),
#       so only 1 scale factor is read and applied to all input.
#       SFAC DATAFILE  WET_DRY_YEAR_SF.txt either opens and reads the first scale factor
#       from WET_DRY_YEAR_SF.txt or reads the next scale factor in the file
#
                                        # JAN   SP 1
SFAC DATAFILE WET_DRY_YEAR_SF.txt       #   <– Open and load first scale factor
DATAFILE  Kc_Annual.txt                 #   <– Open and load first set of Kc
                                        # FEB   SP 2
SFAC DATAFILE WET_DRY_YEAR_SF.txt       #   <– Load next scale factor
LOAD_NEXT                               #   <– Load next set of Kc in "Kc_Annual.txt"
SFAC DATAFILE WET_DRY_YEAR_SF.txt       # MAR  SP 3
LOAD_NEXT
SFAC DATAFILE WET_DRY_YEAR_SF.txt       # APR   SP 4
LOAD_NEXT
SFAC DATAFILE WET_DRY_YEAR_SF.txt       # MAY  SP 5
LOAD_NEXT
SFAC DATAFILE WET_DRY_YEAR_SF.txt       # JUN   SP 6
LOAD_NEXT
SFAC DATAFILE WET_DRY_YEAR_SF.txt       # JUL   SP 7
LOAD_NEXT
SFAC DATAFILE WET_DRY_YEAR_SF.txt       # AUG  SP 8
LOAD_NEXT
SFAC DATAFILE WET_DRY_YEAR_SF.txt       # SEP   SP 9
LOAD_NEXT
SFAC DATAFILE WET_DRY_YEAR_SF.txt       # OCT   SP 10
LOAD_NEXT
SFAC DATAFILE WET_DRY_YEAR_SF.txt       # NOV  SP 11
LOAD_NEXT
SFAC DATAFILE WET_DRY_YEAR_SF.txt       # DEC   SP 12
LOAD_NEXT
SFAC DATAFILE WET_DRY_YEAR_SF.txt       # JAN   SP 13
RELOAD                                  #   <–  Move to first line of "Kc_Annual.txt"
                                        #         then load first set of Kc
```

Figure 2.29. —Continued

C

```
# File: Kc_Annual.txt      –  Annual Crop Coefficient (Kc) file
# Crop  ID
#       1  =  Corn
#       2  =  Onion
#       3  =  Native Vegetation
#   ID    Kc              Crop          Month
    1     0.06      #     Corn          Jan
    2     0.06      #     Onion
    3     0.06      #     Native
    1     0.06      #     Corn          Feb
    2     0.06      #     Onion
    3     0.06      #     Native
    1     0.19      #     Corn          Mar
    2     0.3       #     Onion
    3     0.9       #     Native
    1     0.19      #     Corn          Apr
    2     0.3       #     Onion
    3     0.9       #     Native
    1     1.17      #     Corn          May
    2     1.14      #     Onion
    3     0.9       #     Native
    1     1.17      #     Corn          Jun
    2     1.14      #     Onion
    3     0.9       #     Native
    1     0.4       #     Corn          Jul
    2     0.63      #     Onion
    3     0.9       #     Native
    1     0.4       #     Corn          Aug
    2     0.63      #     Onion
    3     0.9       #     Native
    1     0.4       #     Corn          Sep
    2     0.63      #     Onion
    3     0.9       #     Native
    1     0.4       #     Corn          Oct
    2     0.63      #     Onion
    3     0.9       #     Native
    1     0.06      #     Corn          Nov
    2     0.06      #     Onion
    3     0.06      #     Native
    1     0.06      #     Corn          Dec
    2     0.06      #     Onion
    3     0.06      #     Native
```

Figure 2.29. —Continued

D

```
jtf  @
#       ^= declares which character is used as a delimiter in template
#
# File: WET_DRY_YEAR_SF.tpl         – Template scale factor file that becomes WET_DRY_YEAR_SF.txt
#
# Calibration software creates the file KC_SF.txt from this template
#     WET_DRY_YEAR_SF.txt is the same as WET_DRY_YEAR_SF.tpl, except every word
#     enclosed by the @ delimiter is replaced with a number.
#
# Even though only one scale factor is read, the List Style record ID is still required.
# For List Style with only a single record, the Record IDs are just a place holder,
#     rather than just repeating 1 at the start of each stress period,
#     instead it is set to the stress period number to make the file easier to read
#
# Climate Based Scale Factor:
#                       @WET_YR@    –  Scale factor for years classified as a wet  climate
#                       @DRY_YR@    –  Scale factor for years classified as a dry  climate
#   SP     ScaleFactor
     1    @WET_YR@
     2    @WET_YR@
     3    @WET_YR@
     4    @WET_YR@
     5    @WET_YR@
     6    @WET_YR@
     7    @WET_YR@
     8    @WET_YR@
     9    @WET_YR@
    10    @WET_YR@
    11    @WET_YR@
    12    @WET_YR@
    13    @DRY_YR@
```

Figure 2.29. —Continued

These examples illustrate the power of the *Transient File Reader* and use of scale factors to temporally adjust spatial information. It further illustrates how to cycle through a set of 12 datasets (annual, monthly input) and apply the sets to multiple years. These examples only include a single **SFAC** per stress period. A single input may include multiple **SFAC**s, which can break complicated scaling into a series of simpler products. For example, figure 2.30 presents a simple TFR that reads one stress period of crop coefficient input, but includes two **SFAC**s. The first **SFAC** incorporates a crop-based scale factor, and the second includes a global climate scale factor.

One final note, although the **SFAC** makes use of the **DATAFILE** command to keep loading through a single file, it does not support the use of the keywords **LOAD_NEXT**, **RELOAD**, or **REPEAT**. For example, "**SFAC LOAD_NEXT**" or "**SFAC RELOAD**" are not allowed because **SFAC** does not keep track of the previous file it was associated with. If they are used, then an error is raised by MF-OWHM2, and the execution is stopped. A workaround for the **RELOAD** option is to use the command **DATAFILE** along with the *Generic_Input* post-keyword **REWIND**, which moves the file to the first line before loading the data. For example, the TFR command to reload the scale-factor file in this example would be "**SFAC DATAFILE** WET_DRY_YEAR_SF.txt **REWIND**". This then enables the use of a scale-factor file of a similar structure to the one presented as in the annual crop-coefficient file.

```
jtf  @
#       ^= declares which character is used as a delimiter in template
#
# File: Kc_TFR4.tpl      – Template Transient File Reader for Crop Coefficients
#
# NPER    = 1 stress period      – Total number of stress periods
# NCROP = 3 crops                – Number of FMP simulated crops identified as 1, 2, and 3.
#
# Crop Based Scale Factor:
#                       Crop 1 – Corn:     @KC_SF_CRP1@
#                       Crop 2 – Onion:   @KC_SF_CRP2@
#                       Crop 3 – Native Vegetation: Not Adjusted, by multiplying by 1.0
#
# Climate Based Scale Factor:
#                       @WET_YR@    –   Scale factor for years classified as a wet  climate
#                       @DRY_YR@    –   Scale factor for years classified as a dry   climate


#
SFAC ByCrop @KC_SF_CRP1@  @KC_SF_CRP2@  1.0
SFAC @WET_YR@
OPEN/CLOSE  2000_01_Kc.txt                          # JAN-2000     SP 1
#
```

Figure 2.30. Example *Transient File Reader* (TFR) that is set up as a calibration template file that specifies two sets of advanced scale factors (SFAC). [Comments are any text that are written to the right of a # symbol.]

References Cited

Hanson, R.T., Boyce, S.E., Schmid, W., Hughes, J.D., Mehl, S.M., Leake, S.A., Maddock, T., III, and Niswonger, R.G., 2014, One-Water Hydrologic Flow Model (MODFLOW-OWHM): U.S. Geological Survey Techniques and Methods 6–A51, 120 p., http://dx.doi.org/10.3133/tm6A51.

Niswonger, R.G., Panday, S., and Ibaraki, M., 2011, MODFLOW-NWT, A Newton formulation for MODFLOW-2005: U.S. Geological Survey Techniques and Methods 6–A37, 44 p., https://doi.org/10.3133/tm6A37.

Appendix 3. Modflow Upgrades and Updates

This appendix covers the set of upgrades and revisions to the MODFLOW-2005 base code incorporating new features that expand on the existing packages (fig. 3.1). It begins by discussing some global changes to the MODFLOW base code and how the "Options" section of input to certain packages has been changed to a block-style input (appendix 1). It then discusses expansions to the Discretization (DIS) and Basic (BAS) packages, which now have calendar dates included as part of the simulations to support additional, related options. The next section covers the new Budget Group feature, which allows some of the packages to divide their budgets into multiple groups for printing to the Listing File or separate Budget File (BAS package option **BUDGETDB**). The appendix continues with a discussion of the new Well (WEL) package and continued availability of the original package, which is now accessed with the keyword **WEL1** in the Name file. Subsequent sections present modifications to the General Head Boundary (GHB) package that includes variable conductance; improvements to the Newton-Raphson numerical solver (NWT), Stream Flow Routing (SFR), Parameter Value (PVAL), Multiplier Array (MULT), and Zone Array (ZONE) packages; the LIST and Name files; and a summary of the new Warning Package (WARN), which holds all warnings from packages. This appendix concludes with the new time-step based output available for the Observation (OBS) process and a new stress-period based input of convergence criteria for the NWT and Precondition Conjugate Gradient (PCG) solvers.

Package	Full Name and Description
Name	Name file that describes all packages in use for a simulation and location of their input files.
LIST	Listing file that contain a transcript of the simulation input and output.
WARN	Warning file that contain a transcript of the simulation errors and warnings.
BAS	Basic package that contains global options and initial starting heads.
DIS	Discretization package that specifies the model's spatial and temporal discretization.
PCG	Precondition Conjugate Gradient numerical solver for the groundwater equations.
PCGN	Precondition Conjugate Gradient Nonlinear numerical solver for the groundwater equations.
NWT	Newton-Raphson numerical solver for the groundwater equations.
SFR	Stream Flow Routing package that simulates 1-D river and stream flow.
GHB	General Head Boundary package that defines a head dependent boundary condition.
WEL	Well package that extracts or injects groundwater.
PVAL	Parameter Value package that overrides each package parameter values.
MULT	Multiplier Array package that specifies a set of arrays that can be used with parameters.
ZONE	Zone Array package that specifies a set of arrays that can be used with parameters.
OBS	Observation process that reports observations for various packages. (Packages part of the process: HOB, DROB, DRTOB, GBOB, CHOB, RVOB.)

Figure 3.1. MF-OWHM list of modified of MODFLOW-2005 base-packages. For MF-OWHM2, these specific packages are either completely rewritten or had substantial code updates to add new features. [**Abbreviation**: 1-D, one-dimensional.]

Free Format Input Files Are Now Default

The input file structure used by MODFLOW-2005, by default, had a fixed-format input. This input structure relied on the number of columns of text to define the input. For example, if an input file required two integer variables to be read in a fixed-format, the first integer must reside within the first 10 columns, and the second integer must reside within the second 10 columns. If the integers were outside the correct columns, then MODFLOW would "crash." Figure 3.2 is an example of input formatted for a fixed-format read of the numbers `28` and `123`.

If the user wanted to use free format, then it was required to specify the **`OPTION`** keyword **`FREE`** in BAS. Free format indicates that the input data may be placed anywhere on a line and multiple values are separated by either one or more spaces or tabs or by one comma. This release of MF-OWHM2 presumes free format by default, as if the BAS option **`FREE`** is always specified. Under rare circumstances, the fixed-format input is not compatible with **`FREE`**. Figure 3.3 shows two examples of acceptable fixed formats that fail to be read correctly using a free-format option. When read as fixed 10-column fields of formatted input, the first loads the numbers `25` and `35` (fig. 3.3*A*), and the second loads the numbers `1551313035` and `9203456138` (fig. 3.3*B*).

To maintain compatibility with fixed-format input files that fail to load correctly under the default free format, the BAS package options include the new keyword **`NOFREE`**. This option enables the legacy fixed-format input structure for the base MODFLOW packages. If the option **`FREE`** is specified for a BAS package, then it is ignored because it is now the default option for all simulations.

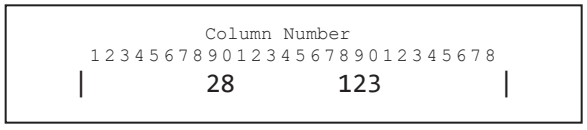


Figure 3.2. Fixed-format input example that supplies 2 integer inputs that each occupy 10 columns of a row of text input. The software reads an integer from columns 1 to 10 and another integer from columns 11 to 20.

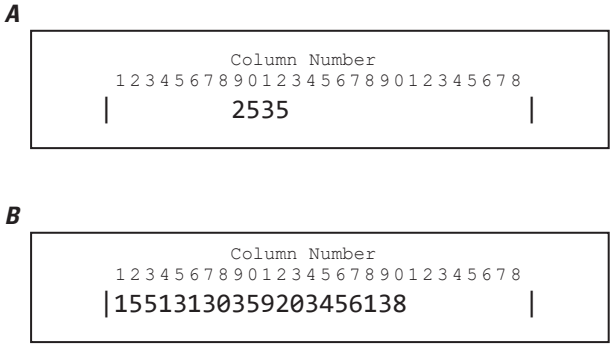


Figure 3.3. Examples of acceptable fixed formats that fail to be read correctly using a free-format option. If coded for reading fixed-format input, the software reads an integer from columns 1 to 10 and another integer from columns 11 to 20. *A*, Example that results in a fixed-format read to load 25 and 35, whereas a free-format read loads 2,535 and raises an error for failing to load a second number. *B*, Example that results in a fixed-format read to load 1551313035 and 9203456138, whereas a free-format read loads 15,513,130,359,203,456,138 and raises an error for failing to load a second number.

Double Precision Number Simulation

MODFLOW-2005 stores floating point numbers as a mix of single and double precision. A single precision number contains seven significant digits—for example `1.234567`—before rounding and can have scientific notation to `1E±38`. Double precision has 15 significant digits—for example `1.23456789012345`—and can have scientific notation to `1E±308`. The disadvantage of double precision numbers is that the random-access memory (RAM) required to store the number (8 byte) is twice that of single precision numbers (4 byte). Given that current computer systems are not memory limited, as was the case in 2005, all real numbers in MF-OWHM2 are now stored using double precision. This improves the accuracy of the simulation and may reduce the simulation time as a result of faster solver convergence.

To maintain backward compatibility with post-processing programs, such as ZoneBudget, the MODFLOW-2005 standard output, binary files—those written with *ULASAV*, *UBDSVA*, *UBDSVB*, *UBUDSV*, *UBDSV1*, *UBDSV2*, *UBDSV3*, and *UBDSV4*—write the double precision numbers as single precision. The most common output file that uses these write utilities is the MODFLOW Cell-By-Cell (CBC) file. The BAS package offers a new option, **`DOUBLE_PRECISION_CBC`**, that instructs MF-OWHM2 to maintain the double precision numbers when writing to the CBC. It should be noted that another advantage of using single precision variables to write to the CBC is that the resulting binary output file uses approximately half the hard drive space as a double precision binary file.

Additional Convergence Metric

The groundwater-flow finite difference equations can be compactly written as Ah = RHS, where A is a matrix, h is a vector containing every model cell's head, and RHS represents the right-hand side of the equations (non-head-dependent components). A solver iteration, called an "outer iteration," requires each package—using the current set of heads in h—to modify the RHS and the A matrix, and then solves for a new h vector. Please see Harbaugh (2005) for a full definition of how the A matrix is assembled and h is solved for.

MODFLOW solvers typically have two convergence criteria. The first is a head tolerance—called `HCLOSE`—that represents the change between the previous solved head vector and the newly solved head vector, h, that is aggregated to a single number using the L_2 or L_∞ norm. The second criterion is a flow residual tolerance—called `RCLOSE`—that represents the discrepancy between the matrix-vector multiplication of A and the updated h vector to the RHS. Specifically, it solves for Ah – RHS = r. The vector, r, has model units of L^3/T and represents the flow residual error. This second solver's convergence criterion is met when the L_2 or L_∞ norm of r is less than `RCLOSE`; the type of norm depends on the solver used. If the equation is solved "perfectly," then the flow residual vector is zero (that is, $r = 0$).

MF-OWHM2 includes a third criterion, called the Relative Volume Error, before convergence is allowed. The Relative Volume Error is the flow residual error vector, r, normalized by the corresponding model cell volume vector, v, which mathematically is r/v and has model units of 1/T. MF-OWHM2 requires that the L_∞ norm of the Relative Volume Error be less than `0.025` before convergence is allowed. Specifically, it is required that $\left\| r/v \right\|_\infty \leq 0.025$ must be true before convergence is allowed. To override the default value, BAS has the option **`MAX_RELATIVE_VOLUME_ERROR`**, which is followed by a new maximum allowed relative-volume error.

Package Options Moved to Block-Style Input

The Basic (BAS), Unsaturated-Zone Flow (UZF), Streamflow Routing (SFR), General-Head Boundary (GHB), and Well (WEL) packages were modified to have their keyword **OPTIONS** loaded using a block style input. The `Options` line of the BAS package, located on Data Set 1 (see the "New Basic Package Options" section), is replaced with **BEGIN OPTIONS** and then has one option per line; the block is terminated by the keyword **END**. The original BAS input `Options` line—that is, the original input structure—is still supported to maintain backward compatibility, but the `Options` line does not support the input of any of the new keywords discussed in this appendix. If the **OPTIONS** block is used, then it must be located at the beginning of the BAS input file along with any other *Block Style* input. If the package supports additional block inputs, then the order of the block inputs does not matter—for example, the **BUDGET_GROUPS** and **OPTIONS** block may be specified in any order. Additionally, if the package uses the optional keyword **PARAMETER**, then it may appear before or after the block input.

For backward compatibility with MODFLOW-2005, the `Options` line is still checked for the options that it supports, but all MF-OWHM2 specific options must now be in the new block style. For models that still use the `Options` line, an innocuous warning is raised during runtime indicating that the input should be changed to block style. If only the original options are used, then the program continues after the warning, but if one of the new options of MF-OWHM2 is used, then it raises an error and stops the program.

In the release of MODFLOW-NWT version 1.1 (Niswonger, and others, 2011), the `Options` and keyword input sections for the SFR and UZF packages were changed to support a *Block Style* input but are backwardly compatible and still support specifying all options on a single line. To maintain backward compatibility in this version of MODFLOW-NWT, the SFR and UZF options only support *Block Style* input. MF-OWHM2 does check for the original `Options` input location, and if it finds the keywords, an error is raised indicating that the keyword must be moved to an **OPTIONS** block. Figure 3.4 shows an example BAS **OPTIONS** block (using MF-OWHM2 options) that must be placed at the start of the input file.

```
BEGIN OPTIONS
      START_DATE  DATE
      NOFREE
      CBC_EVERY_TIMESTEP
      NO_FAILED_CONVERGENCE_STOP
      PERCENTERROR PDIFFPRT
END
```

Figure 3.4. Example of **OPTIONS** block (using MF-OWHM2 options) that must be placed at the start of the BAS package file. [For using during a simulation, DATE is replaced by the starting calendar date of the model; PDIFFPRT is replaced by the mass error percentage that results in raising a percent error warning.]

Calendar Dates

A new Fortran module containing code composed of a calendar datetime object is included in MF-OWHM2. The calendar datetime object allows the use of calendar dates as part of the simulation input and for MF-OWHM2 to keep track of the starting and ending date of each time step. The calendar dates work with all the MF-OWHM2 time units (DIS variable `ITMUNI`), but it is not recommended to use a time unit of years (`ITMUNI = 5`) because of the ambiguity between leap years and non-leap years. The time unit of undefined (`ITMUNI = 0`) is treated as if a time unit of days (`ITMUNI = 4`) had been specified.

If calendar dates are used by any of the MF-OWHM2 features, then a starting calendar date must be specified with the BAS package option **`START_DATE`**, which is followed by the starting calendar date (`DATE`). The **`START_DATE`** keyword replaces the use of the DIS package keyword **`STARTTIME`**, which was used in MF-OWHM to define a starting date using decimal years. **`STARTTIME`** is still supported but is not recommended because it does not account for leap years nor does it allow calendar dates for MF-OWHM2 input. Figure 3.5*A* defines the symbology used, and figure 3.5*B* lists the accepted formats for `DATE`. It is recommended to use either "American" style date format, mm/dd/yyyy, or to use the International Organization for Standardization (ISO) standard, yyyy-mm-dd, for specifying calendar dates.

If the **`START_DATE`**'s "`DATE`" is specified as a decimal year, DYear (fig. 3.5*B*), then it is automatically converted to a calendar date. MF-OWHM2 converts a decimal year to a calendar date by first splitting the integer part from the fractional part (such as `1979.307` to `1979` and `0.307`). The integer part is the calendar year that is either a "non-leap" or "leap" year—the latter of which includes February 29th. The fractional part is then converted to a Julian day of the year (DoY) by multiplying by `365` or `366` days—depending on if the year is a "non-leap" or "leap" year, respectively. The DoY is then converted to the appropriate month and day. This procedure is different when using the DIS package "**`STARTTIME`** DYear" option, which did not distinguish between "non-leap" and "leap" years and assumed `365.2425` days in all years. Figure 3.5*C* presents the day of the year for "non-leap" and "leap" years with the corresponding decimal fraction of a year for select dates. The fraction of the year is determined by subtracting one from the day of the year and dividing it by the total number of days in the year. This results in January 1st always having a fraction of the year equaling 0.0, but December 31st is never 1.0—instead, it is approximately 0.99726 of a year. The reason the fraction is set up this way is to ensure that the date always starts at the start of the day, which is 00:00:00. An example conversion of `1979.307` yields the calendar date `4/23/1979` with a 24-hour clock time of 1:19:12 (hour:minute:second, respectively).

If a starting calendar date is specified, then MF-OWHM2 keeps track of each stress-period date and adds the calendar date to the volumetric budget in the Listing File and relevant package output options. Specifically, when calendar dates are specified in the HOB package, the HOB output file includes both the decimal year and the calendar date with each observation. When importing this date format into spread sheet programs (for example, Microsoft Excel®), the delimiter "T" may need to be removed for the date-time value to be recognized correctly. This can be done by doing a search and replace for the letter "T" with a blank space. (Alternatively, the Excel® "Convert Text to Columns Wizard" allows you to specify any character as a delimiter. By specifying "T" as a delimiter, Excel® reads the date and time as separate input data values.)

A

Symbol	Representation
mmm	Three letter month or full name - Jan or January
mm	Two-digit month number - 01 or 1, with valid values being 1 to 12
dd	Two-digit day of month number - 01 or 1, with valid values being 1 to 31
yyyy	Four-digit year number - must be four digits - 1979, with valid values being 0000 to 9999
T	Separator to indicate that a time of day is included after calendar date
hh	Two-digit hour, 24-hour format - 01 or 1, with range from 00 to 23
mm	Two-digit minute of hour - 01 or 1, with range from 00 to 59
ss	Two-digit second of minute - 01 or 1, with range from 00 to 59

B

Date Format	Comment
mm/dd/yyyy	American style date
mmm/dd/yyyy	American style date in which month is represented by 3 letters
mm/yyyy	Automatically sets day to 1
yyyy-mm-dd	ISO Standard
mm/dd/yyyy**T**hh:mm:ss	T separates calendar date from time
yyyy-mm-dd**T**hh:mm:ss	T separates calendar date from time
DYear	Decimal year - 1979.307, implies 4/23/1979

Figure 3.5. Input for Calendar Dates module in MF-OWHM2: *A*, calendar date symbology; *B*, calendar date formats accepted; and *C*, day of the year and fraction of a year for select dates during a non-leap or leap year. [The fraction of the year is determined by subtracting one from the day of the year and dividing it by the total number of days within the year.]

C

Date (mmm-dd)	Day of Year		Fraction of Year	
	Non-Leap	Leap	Non-Leap	Leap
Jan-01	1	1	0.000000	0.000000
Jan-15	15	15	0.038356	0.038251
Jan-31	31	31	0.082192	0.081967
Feb-01	32	32	0.084932	0.084699
Feb-28	59	59	0.158904	0.158470
Feb-29	—	60	—	0.161202
Mar-01	60	61	0.161644	0.163934
Mar-15	74	75	0.200000	0.202186
Mar-31	90	91	0.243836	0.245902
Apr-01	91	92	0.246575	0.248634
Apr-15	105	106	0.284932	0.286885
Apr-23	113	114	0.306849	0.308743
Apr-30	120	121	0.326027	0.327869
May-01	121	122	0.328767	0.330601
May-15	135	136	0.367123	0.368852
May-31	151	152	0.410959	0.412568
Jun-01	152	153	0.413699	0.415301
Jun-15	166	167	0.452055	0.453552
Jun-30	181	182	0.493151	0.494536
Jul-01	182	183	0.495890	0.497268
Jul-15	196	197	0.534247	0.535519
Jul-31	212	213	0.578082	0.579235
Aug-01	213	214	0.580822	0.581967
Aug-15	227	228	0.619178	0.620219
Aug-31	243	244	0.663014	0.663934
Sep-01	244	245	0.665753	0.666667
Sep-15	258	259	0.704110	0.704918
Sep-30	273	274	0.745205	0.745902
Oct-01	274	275	0.747945	0.748634
Oct-15	288	289	0.786301	0.786885
Oct-31	304	305	0.830137	0.830601
Nov-01	305	306	0.832877	0.833333
Nov-15	319	320	0.871233	0.871585
Nov-30	334	335	0.912329	0.912568
Dec-01	335	336	0.915068	0.915301
Dec-15	349	350	0.953425	0.953552
Dec-31	365	366	0.997260	0.997268

Figure 3.5. —Continued

Discretization Package (DIS) Improvements

Several modifications were made to the DIS package to make it more representative of real-world time keeping. Specifically, it now has the ability to keep track of leap years, supports calendar dates, and can specify time-step lengths directly (rather than as a multiplier).

Variable Time-Step Length

The DIS package was modified to allow the user to specify the exact time-step length. The time-step lengths are loaded on the same line as the stress-period information (`PERLEN NSTP TSMULT SS/TR`). This feature is initiated when the time-step count (`NSTP`) is specified as a negative number and the multiplier is set to `1`. Then, the absolute value of the time-step count represents the number of time-step lengths read to the right of the stress-period type (`SS/TR`), and the sum is the stress-period length (overwrites `PERLEN`). This allows the user to customize time-step lengths to match observation times or to create an acceleration factor that uses a more compact set of time steps (for example, `1`, `2`, `7`, `10`, `80` to accelerate to a total of `100` days). The compact numbering can be used to prevent simulation times with decimal parts by requesting time-step lengths to be whole numbers. This is particularly advantageous when the stress periods mimic calendar months, and the month can be broken into different counts of days, such as `31` days, and four time steps could have lengths of `7`, `8`, `8`, and `8` days.

Figure 3.6 is an example of DIS input that uses the variable time-step lengths, `TS_LEN`, for the odd-numbered stress periods. It only shows the stress period specification part of a larger DIS package that specified a time unit of days (`ITMUNI = 4`).

Specifying Land or Ground-Surface Elevation

The DIS was modified to optionally load a land-surface elevation (LSE, sometimes called ground-surface elevation, GSE) array that represents the model-grid's surface elevations (a digital elevation model, or DEM). This allows a global surface-elevation grid to be accessed by other packages. To have the DIS load the surface elevation in model units, [length, L], the keyword **`SURFACE`** must be placed on the line after reading `DELC` and before `Top` (see the "Full DIS Input Instructions" section). If the keyword is not present, then global surface elevations are ignored and `Top` (Dataset 5b) is loaded instead. If loaded, the LSE array is read with the *Universal Loader* (`ULOAD`) described in appendix 1. Figure 3.7 shows a part of the DIS input for loading an LSE array. As of this report's publication, if global surface elevations are specified in the DIS package, the only package that uses them is the Farm Process (FMP). If it is specified in the DIS, then the ground-surface elevation array is not required to be specified as part of the FMP input; it automatically uses the DIS package array.

```
#  Example Basic DIS Package Input that illustrates using a Variable Time Step Length
#     Input assumes that BAS package includes the option "START_DATE  4/23/1979"
#
  2   3   4   3   4   2  # NLAY  NROW  NCOL  NPER  ITMUNI  LENUNI
  0   0          #  LAYCBD(NLAY)
  CONSTANT 100.  #  DELR(NCOL)
  CONSTANT 100.  #  DELC(NROWS)
  CONSTANT 500.  #  TOP(NCOL, NROW)
  CONSTANT 450.  #  LAY 1 BOTM(NCOL, NROW)
  CONSTANT 300.  #  LAY 2 BOTM(NCOL, NROW)
#  PERLEN  NSTP  TSMULT  ss/tr  [TS_LEN]
  8.000 -2  1     ss  3. 5.        # SP1  4/23/1979  to  5/1/1979
  31.00  2  1     tr               # SP2  5/1/1979  to  6/1/1979
  30.00 -4  1     tr  7. 7. 8. 8.  # SP3  6/1/1979  to  7/1/1979
  31.00  2  1     tr               # SP4  7/1/1979  to  8/1/1979
  31.00 -2  1     tr 14. 16.       # SP5  8/1/1979  to  9/1/1979
```

Figure 3.6. Example of DIS input that uses the variable time-step lengths, TS_LEN, for the odd-numbered stress periods. [The DIS input assumes that the BAS package includes "START_DATE 4/23/1979" in the option block.]

```
4.    DELC(NROW) - U1DREL

5a.   [SURFACE   ULOAD(NCOL,NROW)]

5b.   Top(NCOL,NROW) - U2DREL
```

Figure 3.7. Part of DIS input and read utilities for loading a land-surface elevation (LSE) array that represents the model-grid's surface elevations. The numbers 4., 5a., and 5b. represent the MODFLOW-2005 DIS package input item numbers.

LAYCBD Keyword to Disable for All Layers

The DIS package requires loading a flag that indicates if a layer contains a quasi-three-dimensional (3D) confining bed beneath it. The input variable that defines this is called `LAYCBD`, and the user is required to specify its value as either `0` or `1` for each layer (`NLAY` layers) to indicate if it overlies the confining bed. Because the quasi-3D confining bed is a legacy feature of MODFLOW, and its use is limited—and not recommended—a new keyword is available that automatically sets `LAYCBD` to `0` for all layers. If the keyword **`NO_LAYCBD`** is placed where the DIS package expects to find `NLAY` integer flags for `LAYCBD`, then its value is automatically set to zero for all layers. Figure 3.8 is an example of the DIS input that shows the appropriate place to use the **`NO_LAYCBD`** keyword. Note that input items enclosed in brackets, [ ], are optional.

ITMUNI (TIME) and LENUNI (LENGTH) Support Keywords

The DIS package uses two integer variables to define the simulation-model time and space units. The first variable is the flag `ITMUNI`, which defines the time unit, and the next is `LENUNI`, which defines the length unit. MODFLOW requires all input to be consistent with the specified values of these two variables. Most input packages use the letter T to indicate the time unit defined by `ITMUNI` and the letter L to indicate the length unit defined by `LENUNI`. The DIS package defines these two variables with an integer that represents the time and space units. The input has been extended to also check for use of a keyword instead of the integers. Figure 3.9 provides two lists of supported keywords that may be used in the place of the integer flags to indicate time and distance units of measure.

Full DIS Input Instructions

The previous sections have described modifications to the DIS package. Each section presented the specific location of the input for the user to add that specific feature that changed. The full input is described here, with the modifications included, to provide a user reference for all of DIS input in one location. Comments are allowed on any line of input when they are preceded by a "`#`" character, except between lines of data loaded with *U1DREL* and *U2DREL* read utilities. Figure 3.10 provides examples of DIS input data items by line. Note that input items enclosed in brackets, [ ], are optional.

```
1. NLAY NROW NCOL NPER ITMUNI LENUNI [XFIRSTCORD YFIRSTCORD GRIDROTATION [COORD_OPTIONS]

2. LAYCBD(NLAY) or NO_LAYCBD

3. DELR(NCOL) - U1DREL
```

Figure 3.8. Example of DIS input that shows the appropriate place to use the **NO_LAYCBD** keyword. The numbers, 1., 2., and 3. represent the MODFLOW-2005 DIS package input item numbers.

A

ITMUNI	Keyword Equivalent	
	Singular	Plural
1	**SECOND**	**SECONDS**
2	**MINUTE**	**MINUTES**
3	**HOUR**	**HOURS**
4	**DAY**	**DAYS**
5	**YEAR**	**YEARS**

B

LENUNI	Keyword Equivalent	
	Singular	Plural
1	**FOOT**	**FEET**
2	**METER**	**METERS**
3	**CENTIMETER**	**CENTIMETERS**

Figure 3.9. Keywords in the DIS package that specify *A*, model time unit [T] ITMUNI; and *B*, model length unit [L] LENUNI. The keyword, either the singular or plural version, can be used in the place of the original DIS package ITMUNI and LENUNI integer flags. [DIS, Discretization package; [T], time unit used for all input and output during a simulation; [L], length unit used for all input and output during a simulation.]

A

1. NLAY NROW NCOL NPER ITMUNI LENUNI [XFIRSTCORD YFIRSTCORD GRIDROTATION [**COORD_OPTIONS**]]
2. LAYCBD(NLAY)
   The keyword **NO_LAYCBD**, may be used on line 2 instead of the integer values to set LAYCBD to zero for all layers
3. DELR(NCOL) - *U1DREL*
4. DELC(NROW) - *U1DREL*

5a. [**SURFACE** ULOAD(NCOL,NROW)]

5b. Top(NCOL,NROW) - *U2DREL*

6. BOTM(NCOL,NROW) - *U2DREL*
   Item 6 is repeated for each model layer and Quasi-3D confining bed in the grid.
   These bottom-of-layer variables are read in sequence going down from the top of the system.
   Thus, the number of BOTM arrays must be NLAY plus the number of Quasi-3D confining beds.
7. PERLEN NSTP TSMULT SS/TR [TS_LEN]
   Item 7 is repeated for each stress period (NPER).

Figure 3.10. Discretization (DIS) package input structure and explanation: *A*, DIS input data items by line; and *B*, explanation of the input variable names.

B

NLAY is the number of layers in the model grid.

NROW is the number of rows in the model grid.

NCOL is the number of columns in the model grid.

NPER is the number of stress periods in the simulation.

ITMUNI indicates the time unit of model data, which must be consistent for all data values that involve time. All inputs that use the term "T" will use this as the unit for time. This can be either an integer flag or keyword. The keyword is the same as the name of the time unit. Using "Undefined" units is not recomended.

Flag	Unit and Keyword
0	Undefined – Not a supported keyword
1	**SECOND**
2	**MINUTE**
3	**HOUR**
4	**DAY**
5	**YEAR**

LENUNI indicates the length unit of model data, which must be consistent for all data values that involve length. All inputs that use the term "L" (or "L^m") will use this as the base unit for length [L], area [L^2], and volume [L^3]. This can be either an integer flag or keyword. The keyword is
the same as the name of the distance unit. Using "Undefined" units is not recomended.

Flag	Unit and Keyword
0	Undefined – Not a supported keyword
1	**FEET**
2	**METER**
3	**CENTIMETER**

XFIRSTCORD is the X Cartesian coordinate of the model cell center at Row 1, Column 1

YFIRSTCORD is the Y Cartesian coordinate of the model cell center at Row 1, Column 1

GRIDROTATION is the Polar angle of the model grid

COORD_OPTIONS the following are accepted keyword options for the coordinate system:

LLCOORDINATE is an optional keyword that indicates that XFIRSTCORD and YFIRSTCORD refer to the cell center of row NROW and column 1 (that is, the lower left corner).

CORNERCOORD is an optional keyword that indicates that XFIRSTCORD and YFIRSTCORD refer to the cell's outermost corner instead of the cell center.

PRINTCOORD is an optional keyword that prints the coordinate arrays to the list file.

LAYCBD is either a single keyword, **NO_LAYCBD**, or a list of NLAY integer flags that indicate if the layer overlies a Quasi-3D confining bed.
0 indicates no confining bed, and a non-zero value indicates a confining bed.
LAYCBD for the bottom layer must be 0.
The keyword **NO_LAYCBD** automatically sets LAYCBD to zero for all layers.

Figure 3.10. —Continued

***B** (continued)*

DELR	is the cell width along rows. Input one value for each of the NCOL columns. This is a multi-value one-dimensional variable with one value for each model column.
DELC	is the cell width along columns. Input one value for each of the NROW rows. This is a multi-value one-dimensional variable with one value for each model row.
SURFACE	is a keyword that indicates the land surface elevation (LSE) is loaded with ULOAD and reads a NROW by NCOL two-dimensional array. The LSE must be greater than or equal to the elevations specified in Top.
Top	is the top elevation of layer 1. If layer 1 represents a water-table aquifer, then it may be reasonable to set Top equal to the land-surface elevation. That is, it may be the same array read by **SURFACE**.
BOTM	is the bottom elevation of a model layer or a Quasi-3D confining bed.
PERLEN	is the length of a stress period.
NSTP	is the number of time steps in a stress period. If NSTP < 0, then the time step length is specified and the absolute value, |NSTP|, represents the number of time step lengths that are defined as TS_LEN.
TSMULT	is the multiplier for the length of successive time steps. If NSTP < 0, then TSMULT is read but not used. If TSMULT is equal to 1, then the time step evenly divides PERLEN by NSTP. If TSMULT is negative, then its value is automatically set to 1. If TSMULT is not equal to 1, then the length of a time step is calculated by multiplying the length of the previous time step, that is, Δt_i = TSMULT $\times \Delta t_{i-1}$. The length of the first time step, Δt_1, is calculated as $\Delta t_1 = \text{PERLEN}\left(\frac{\text{TSMULT}-1}{\text{TSMULT}^{\text{NSTP}}-1}\right)$.
TS_LEN	is only read if NSTP < 0 and represents a user specified time step length. if NSTP < 0, then |NSTP| time step lengths are read and PERLEN is set to their sum.
SS/TR	is a character variable that indicates if the stress period is transient or steady state. The only allowed options are "SS" and "TR" but these are case insensitive

Figure 3.10. —Continued

New Basic Package (BAS) Options

The basic package was modified to include a set of new options in the basic package **OPTIONS** block that alter the approach to model runs in MF-OWHM2. Some of the new options are **START_DATE**, **FASTFORWARD**, **INPUT_CHECK**, **BUDGETDB**, **NOCBC**, **NOCBCPACK**, **CBC_EVERY_TIMESTEP**, **PRINT_CONVERGENCE**, **PRINT_FLOW_RESIDUAL**, **PRINT_RELATIVE_VOLUME_ERROR**, **NO_DIM_CHECK**, **DEALLOCATE_MULT**, **TIME_INFO**, and **NO_FAILED_CONVERGENCE_STOP**. The primary new options are listed with short descriptions in figure 3.11 and are discussed in the sections that follow.

A

Keyword	Description
CBC_EVERY_TIMESTEP	Indicates that the CBC is written to for every time step by all packages that have a non-zero cell-by-cell file specified.
CBC_LAST_TIMESTEP	Indicates that the CBC is written to at the last time step for each stress period by all packages that have a non-zero cell-by-cell file specified.
CHTOCH	Flow between adjacent constant-head cells should be calculated.
DEALLOCATE_MULT	Deallocate MULT package arrays once they are no longer required.
DOUBLE_PRECISION_CBC	Write CBC using double precision numbers instead of single precision.
FREE	Free format input for MODFLOW packages (option enabled by default).
INPUT_CHECK	Run simulation without the solver to check input and output files.
NO_DIM_CHECK	Skip length, width, thickness, and volume checks in the BAS and DIS. This improves speed for model grids that are known to be error free.
NO_FAILED_CONVERGENCE_STOP	Simulation continues, even if convergence fails. Same as the BAS option "**STOPERROR** 1E30"
NOCBC	Disables writing to the cell-by-cell file for all packages.
NOCBCPACK	Disables writing to the cell-by-cell file for all packages except for the main flow package.
NOFREE	Fixed-format input for supported MODFLOW packages. This is necessary for legacy MODFLOW models that rely on fixed-format input, which in no longer the default option.
PAUSE	Requires the user to press enter at the end the simulation.
XSECTION	Indicates the model is a 1-row cross section for which STRT and IBOUND should each be read as single two-dimensional variables with dimensions of NCOL and NLAY.

Figure 3.11. Complete list and descriptions of supported basic (BAS) package options: *A*, Options that do not have any arguments; *B*, Options that require arguments; and *C*, Options that are associated with writing output information. [The keyword for the desired option must be specified in the BAS OPTIONS block, one per line, and if there is an additional, required, argument, then it is specified in the argument column. Options and arguments must be specified on the same line in the BAS OPTIONS block.]

B

Keyword	Argument	Description
DAMPEN_START	ITER DMP	Apply a dampening factor (DMP) to the first ITER outer-solver iterations of the simulation. Solver convergence is disabled during the first ITER outer-solver iterations. This is useful if the initial conditions are unstable and result in a floating point overflow during the first time step of the simulation. ITER is specified as an **INT** and DMP as a **FLOAT**. Recommended values of DMP are 0.1–0.5.
FASTFORWARD	SPSTART [SPSTOP]	Run simulation from stress period SPSTART to SPSTOP, by reading each stress period input until SPSTART. The initial conditions defined in the BAS package are used as the initial condition at SPSTART. If SPSTOP is not specified, then it is automatically set to the end of the simulation. SPSTART and SPSTOP are specified either as an integer, to indicate the stress period number, or as a date that is contained by one of the stress periods. If set to a date, then the stress period that contains the date is the starting or ending stress period.
HEAD_DISTANCE_ABOVE_LSE_LIMIT	ABOVE_LSE_LIM	Imposes an upper limit to the head solution generated from the solver. The limit is a distance above the user specified land surface elevation (LSE). If LSE is not specified, then the TOP of the first layer is used instead. If the solver calculates a head value greater than the distance above the LSE, then the head is set to the elevation that distance represents. ABOVE_LSE_LIM, specified as a **FLOAT**, is the max distance above LSE or TOP that the solver calculated head solution can be. This is useful for models that have floating point overflow errors.

Figure 3.11. —Continued

***B** (continued)*

Keyword	Argument	Description
MAX_RELATIVE_VOLUME_ERROR	MAX_RVOL_ERROR	MAX_RVOL_ERROR is the maximum allowed relative volume error specified as a **FLOAT**. If not specified, then the default is 0.025 Set to large value to disable additional convergence check.
MAXBUDGET	MXBUD	Specify the maximum number of budget groups allowed. If not specified, then MXBUD is set to 100. Increase MXBUD to a greater value number when an error is raised because there is not enough memory to hold all budget groups. To save on memory you may reduce MXBUD to the number of packages in use plus 10.
MAXPARAM	MXPAR MXCLST MXINST	Specify maximum number of parameters (MXPAR), parameter clusters (MXCLST), and parameter instances (MXINST). Default is MXPAR=2000, MXCLST=2000000, MXINST=50000
MIN_SOLVER_ITERATION	MIN_SOLV_ITER	Require that each time step solve at least MIN_SOLV_ITER outer-solver iterations. This is helpful if solver converges too soon, resulting in mass balance errors.
PERCENTERROR	PDIFFPRT	Specifies the minimum rate percent error at the end of each time step before a warning is raised and the volumetric budget is printed. If PDIFFPRT is set to 0 then a warning is raised every time step. If not specified, then defaults to 5 percent.

Figure 3.11. —Continued

***B** (continued)*

Keyword	Argument	Description

SHIFT_STRT	ULOAD	Shift the initial conditions (STRT) for each layer.
		Input uses the *Universal Loader* (ULOAD) with *List Style* input to read NLAY **FLOAT** numbers that are added to STRT.
		For a three-layer model (NLAY=3), the following example
		Adds 9 to the layer 1's STRT array,
		Adds –7 to the layer 2's STRT array,
		Adds 0 to the layer 3's STRT array.
		`SHIFT_STRT INTERNAL`
		`1 9.0`
		`2 -7.0`
		`3 0.0`

START_DATE	SIM_START_DATE	Specifies the starting date of the simulation.
		This keyword is required if any input uses calendar dates.
		This date is a reference for all calendar date input and is used to determine each time step's starting and ending date and time.
		SIM_START_DATE is specified using any of the MF-OWHM2 accepted date formats that include.
		Examples that specify April 23rd, 1979, as the starting date:
		`START_DATE 4/23/1979`
		`START_DATE 1979-4-23`
		Examples that include the 24hr clock time for 11:15 PM:
		`START_DATE 4/23/1979T23:15`
		`START_DATE 1979-4-23T23:15`
		Examples that set the seconds to 30:
		`START_DATE 4/23/1979T23:15:30`
		`START_DATE 1979-4-23T23:15:30`

STOPERROR	STOPER	STOPER is a percent error that is compared to the budget percent discrepancy if the solver convergence criteria are not met. If the percent error is less than STOPER, then the simulation does not stop. If not specified, then STOPER is set to 0.0 percent. Note that the option **NO_FAILED_CONVERGENCE_STOP** is equivalent to "**STOPERROR** 1E30"

Figure 3.11. —Continued

C

Keyword	Argument	Description
BUDGETDB	*Generic_Output_OptKey*	Write every time step's groundwater-flow budget in a spreadsheet (database) friendly format. Output contains same information as in the LIST file volumetric rate budget.
CUMULATIVE_RESIDUAL_ERROR_ARRAY	*Generic_Output_OptKey*	Write at the end of the simulation the cumulative residual error for every model cell.
HEAD_DISTANCE_ABOVE_LSE_PRINT	ABOVE_LSE_PRNT *Generic_Output_OptKey*	Write to a file any model cell that has a head elevation greater than ABOVE_LSE_PRNT plus the user-specified land surface elevation (LSE). If LSE is not specified, then the TOP of the first layer is used instead. ABOVE_LSE_PRNT is specified as a **FLOAT**.
ITERATION_INFO	*Generic_Output_OptKey*	Write every time step's the final solver information.
PRINT_CONVERGENCE	NTERM OUTER_START *Generic_Output_OptKey*	After the OUTER_START solver iterations, write the NTERM cells that had the worst solver change in head to a file. Irrelevant of OUTER_START, the last solver iteration is always printed. If OUTER_START = 0, then only print the last solver iteration. If OUTER_START < 0, then print the last MXITER − OUTER_START iterations.
PRINT_FLOW_RESIDUAL	NTERM OUTER_START *Generic_Output_OptKey*	After the OUTER_START solver iterations, write the NTERM cells that had the worst flow residual to a file. Irrelevant of OUTER_START, the last solver iteration is always printed. If OUTER_START = 0, then only print the last solver iteration. If OUTER_START < 0, then print the last MXITER − OUTER_START iterations.

Figure 3.11. —Continued

***C** (continued)*

Keyword	Argument	Description
PRINT_RELATIVE_VOLUME_ERROR	NTERM OUTER_START *Generic_Output_OptKey*	After the OUTER_START solver iterations, write the NTERM cells that had the worst relative volume error to a file. Irrelevant of OUTER_START, the last solver iteration is always printed. If OUTER_START = 0, then only print the last solver iteration. If OUTER_START < 0, then print the last MXITER − OUTER_START iterations.
RESIDUAL_ERROR_ARRAY	*Generic_Output_OptKey*	Write at the end of each time step the residual error for every model cell.
RESIDUAL_ERROR_ARRAY_THRESHOLD	PRNT_RES_LIM	If specified, then **RESIDUAL_ERROR_ARRAY** only writes residual errors for time steps that have a rate percent error greater than PRNT_RES_LIM rate percent error.
TIME_INFO	*Generic_Output_OptKey*	Write each model step's time step length, its starting calendar date, starting decimal year and simulation time to a file. This is useful for post-processing tools.

Figure 3.11. —Continued

FASTFORWARD—Simulation Time-Frame Adjustments (BAS)

The keyword **FASTFORWARD** is available with the BAS **OPTIONS** block to specify a simulation time window that is a subset of the current stress-period set up. An example of a simulation time-frame adjustment is for a model that is developed with 10 stress periods, but only the simulation of stress-periods 5 through 7 are of interest. The **FASTFORWARD** option would cycle through the input files until reaching stress-period 5, then run the simulation until stress-period 7 completes. This allows analysis of shortened time frames without having to rebuild all the input datasets. Another advantage of this feature is to allow calibration of different simulation-time windows without having multiple copies of the model.

To initiate a **FASTFORWARD**, the keyword must be located in the BAS **OPTIONS** block followed by the starting and ending stress period. If a **START_DATE** is specified as a BAS option ("Calendar Dates" section), then a starting and ending calendar date may be specified for **FASTFORWARD**, which is then translated to the appropriate starting and ending stress periods.

If the **FASTFORWARD** feature is enabled, the simulation rolls forward until the starting stress period is found. The initial conditions specified in the simulation (for example, the BAS STRT variable) are propagated forward and used as the initial conditions at the requested starting stress period. This requires the initial heads to be set to the initial condition for the selected **FASTFORWARD** starting stress period. Figure 3.12 shows the **OPTIONS** block of BAS with the **FASTFORWARD** option. When SPSTART and SPSTOP are set as calendar dates, MF-OWHM2 searches for the first stress period that contains that date. Figure 3.13 presents the search algorithm used by MF-OWHM2 to discern the starting and ending stress periods when they are specified as calendar dates.

Figure 3.14 provides an example of using the **FASTFORWARD** feature for a simulation model. The example presumes that input to the BAS package **START_DATE** option specifies the starting date of 1/1/2000 and there are 12 monthly stress periods (that is, each stress-period length is equal to the length of the respective month; for example, January has 31 days or 744 hours, or 44,640 minutes, or 2,678,400 seconds).

A

```
BEGIN  OPTIONS
          #
          FASTFORWARD SPSTART  [SPSTOP]
          #
END
```

B

FASTFORWARD	is a BAS option to specify a simulation-time frame.
SPSTART	is the simulation starting point that is specified either as a calendar date or a stress-period number.
SPSTOP	optional; is the simulation stopping point that is specified either as a calendar date or stress-period number. If not specified, then SPSTOP is the total number of stress periods (NPER).

Figure 3.12. BAS package **OPTIONS** block with **FASTFORWARD** keyword: *A*, example of structure; and *B*, input description.

A

```
IF        DATE < STARTDATE:
                              SP = 1
ELSE IF   DATE > SPDATE(NPER):
                              SP = NPER
ELSE Search for SPDATE(I) <= DATE < SPDATE(I+1):
                              SP=I
```

B

DATE	is SPSTART or SPSTOP in the format of a calendar date.
STARTDATE	Starting date of simulation defined by BAS option **START_DATE**.
SPDATE(I)	Date at the start of the I^{th} stress period.
NPER	Total number of stress periods (as specified in the DIS package).
SP	Selected stress period number.

Figure 3.13. Algorithm for finding first and last stress periods used by the **FASTFORWARD** feature: *A*, the search algorithm used when the **FASTFORWARD** simulation period (values of SPSTART and SPSTOP) is specified using calendar dates; and *B*, explanation of variables used in the search algorithm.

```
#  START_DATE = 1/1/2000
#  NPER = 12 stress periods; each stress period represents 1 month.
#  Stress period length is equal to the number of days in the month it represents
#  Start on stress period 3 and end the simulation at NPER (which is stress period 12)
FASTFORWARD   3

# Start on stress period 3 and end the simulation at NPER (which is stress period 12)
# This is because stress period 3 is from 3/1/2000 through 3/31/2000
FASTFORWARD   3/1/2000

# Start on stress period 3 and end on stress period 5
FASTFORWARD   3    5

# Start on stress period 3 and end on stress period 5
FASTFORWARD   3/1/2000    5/1/2000
```

Figure 3.14. Examples of using the **FASTFORWARD** feature for a simulation model. [Examples assume that the simulation starting date is 1/1/2000 and there are 12 monthly stress periods.]

Input_Check—Cycling Through All Input Files, BAS Option

Similar to the **FASTFORWARD** option, the BAS package now includes the keyword **INPUT_CHECK**. It results in a simulation cycling through all input files. Essentially, this provides a fast method to check if there are any input errors by allowing a simulation run to cycle through the input files without solving the groundwater flow or any process equations. This is especially useful for long simulations that might otherwise take a long time to reach an input section that has an error.

BUDGETDB—Budget Information Written to Separate Database Friendly File

Previously, the budget information was written to the LIST file (or WBGT file) when requested by the Output Control (OC) package. To obtain budget information independent of the OC package, the BAS option, **BUDGETDB**, has been added. When this option is invoked, the budget information is written to a *Generic_Output* file (appendix 1). The *Generic_Output* file can be either formatted text or binary format.

The **BUDGETDB** file formatted as text includes the following fields on the header for the output from each time step: DATE_START, PER, STP, DELT, SIMTIME, STORAGE_IN, STORAGE_OUT, CHD_IN, CHD_OUT, and so on. The header continues the listing of each package's name with _IN and _OUT appended to its abbreviation. The DATE_START is the calendar date at the start of the time step in the form of yyyy-mm-ddThh:mm:ss (for example, 1979-4-23T21:21:00). The T is used as a separator between the calendar date and 24-hour clock time. The PER and STP are the stress-period and time-step numbers, respectively. The budget output in text format facilitates further analysis with external software such as a database system or spreadsheet.

If the **BUDGETDB** is written to a binary file, then it has a slightly different format. The format is printed in the LIST file and as the first record in the binary file. The first record is a single Fortran Integer, which is a "count" of the number of columns in the database. Then there are "count" text strings of 16 characters length that define the column names. Similar to the text version, the first set of column names are DATE_START, PER, STP, DELT, and SIMTIME. After SIMTIME, the remaining column names depend on which packages are used for the simulation. The binary file does not include the headers for total input and output (IN_OUT) or percent error (PERCENT_ERROR), which are in the text version. After the "count" lines of text strings, the remaining records, one for each time step, contain only the information for the actual budget record defined by the column names. DATE_START is 19 characters long and uses the same format as the text version; PER and STP are integer variables, and the remaining numerical values are Fortran single precision. The numerical values are changed to double precision if the BAS option **DOUBLE_PRECISION_CBC** is included.

NOCBC and NOCBCPACK—Turn Off Cell-By-Cell Writing (CBC)

Most packages offer a Fortran unit number to write CBC budget information. This file is then used by post processing programs, such as zonebudget, to analyze simulation results. During calibration, the CBC output is often not required. To deactivate CBC output when using versions previous to MF-OWHM2, the input files all had to be modified to set each package's CBC value to zero. This was done to reduce simulation run time and avoid the input and output (I/O) for the excessively large CBC file. There is an increased potential for user-input error when setting all the CBC values to zero and then subsequently resetting them back to the correct unit number. To prevent this, the user can leave the CBC Fortran file unit number in all the packages and add the BAS option keyword **NOCBC**, which automatically suppresses all writing to the CBC file. If the user wishes to only have the flow packages (for example, LPF, UPW, HUF) write to the CBC, then the BAS option keyword **NOCBCPACK** can be used.

CBC_EVERY_TIMESTEP—Turn On Cell-By-Cell Writing (CBC)

The converse of the BAS option keyword **NOCBC** is the keyword **CBC_EVERY_TIMESTEP**. This keyword forces writing to the CBC at the end of every time step for all packages that have a non-zero Fortran unit specified for their CBC output. The advantage of this is it is not necessary to specify the keyword **SAVE BUDGET** in the OC for every model time step. Alternatively, the BAS keyword **CBC_LAST_TIMESTEP** is available to write to the CBC for the last time step of each stress period.

Obtaining Solver Information to External File

For nonlinear models, it is advantageous to see how the solver proceeds to convergence to identify model cells that are highly nonlinear and oscillate. For details on the convergence criteria, please review the "Additional Convergence Metric" section, which gives an overview of the different solver convergence metrics. The keywords **PRINT_CONVERGENCE**, **PRINT_FLOW_RESIDUAL**, **PRINT_RELATIVE_VOLUME_ERROR**, **ITERATION_INFO**, **RESIDUAL_ERROR_ARRAY**, and **CUMULATIVE_RESIDUAL_ERROR_ARRAY** allow writing specific convergence information to separate output files. The keywords determine the convergence criteria that are to be written. **PRINT_CONVERGENCE** writes information on how the head changes for each solver iteration. **PRINT_FLOW_RESIDUAL** writes the solver flow residual error (which is compared against RCLOSE). **PRINT_RELATIVE_VOLUME_ERROR** writes the relative-volume error changes for each solver iteration. **ITERATION_INFO** writes the number of outer iterations required by the solver for each time step and the rate error, volume error, and percent error at the end of the time step. **RESIDUAL_ERROR_ARRAY** writes the flow residual error for each model layer at the end of each time step. **CUMULATIVE_RESIDUAL_ERROR_ARRAY** writes the cumulative flow residual error for each model layer at the end of the simulation.

The advantage of these output options is the user can investigate the model cells with large changes for potential input errors or questionable conceptual development for time steps that fail to converge. Common causes of a time step failure to converge are a model cell that is too thin (small vertical thickness), SFR segments with conductance values that are too large, or cells that incur the so-called MODFLOW "Wet-Dry" problem within a MF-OWHM2 simulation. Knowing which cells cause the failed convergence gives the user a point of reference to investigate. Figure 3.15 shows the input format to include the additional convergence output files. Note, all the options are presented for completeness, but one or any combination may be used.

Figure 3.16 lists and defines the header items included in the output files. Figure 3.16*A* defines formatted text headers for **PRINT_CONVERGENCE**. Figure 3.16*B* shows the structure of each binary record if the **PRINT_CONVERGENCE** is accompanied by specifications to write a binary file. Figure 3.16*C* lists and defines the header items included in the output file that results from specifying **PRINT_FLOW_RESIDUAL**. Figure 3.16*D* lists and defines the header items included in the output file that results from specifying **PRINT_RELATIVE_VOLUME_ERROR**.

A

```
BEGIN  OPTIONS
   #
   PRINT_CONVERGENCE                NTERM OUTER_START  Generic_Output
   #
   PRINT_FLOW_RESIDUAL              NTERM OUTER_START  Generic_Output
   #
   PRINT_RELATIVE_VOLUME_ERROR      NTERM OUTER_START  Generic_Output
   #
   ITERATION_INFO                   Generic_Output
   #
   RESIDUAL_ERROR_ARRAY             Generic_Output
   #
   CUMULATIVE_RESIDUAL_ERROR_ARRAY  Generic_Output
END
```

B

NTERM	is the number of cells to print for each outer iteration. Each outer iteration then prints NTERM model cells that had the largest change since the previous iteration.
OUTER_START	is the solver iteration to begin printing the NTERM cells. If set to zero, then it only prints the final-converged or last-solver iteration. If set to a negative number, then it prints either final-converged iteration or iterations after the maximum solver iterations minus the OUTER_START plus one.
Generic_Output	is the file written to.

Figure 3.15. BAS package options for convergence related output file: *A*, example of **OPTIONS** block with keywords; and *B*, explanation of input variables used by the keywords.

A

SP	is the stress-period number.
TS	is the time-step number.
ITER	is the outer iteration number
LAY	is the layer of the head value.
ROW	is the row of the head value.
COL	is the column of the head value.
HEAD	is the solver calculated head for the outer iteration, ITER.
CHNG_HEAD	is the change in head between the current solver iteration, ITER, and the previous iteration, ITER – 1. That is, CHNG_HEAD = HEAD(ITER) - HEAD(ITER-1)
DATE	is the month and year of the time step.
CELL_ID	is the model cell's unique identifier (ID). The cell ID is useful for re-sorting the file to look at sequential changes in data for a specific model cell. The ID is determined by the following formula: $CELL\_ID = COL + NCOL * (ROW - 1) + NCOL * NROW * (LAY - 1)$

B

STP	INTEGER
TS	INTEGER
ITER	INTEGER
LAY	INTEGER
ROW	INTEGER
COL	INTEGER
HEAD	DOUBLE
CHNG_HEAD	DOUBLE
DATE	CHARACTER(8), starting date of time step as "mmm-yyyy"
ID	INTEGER

Figure 3.16. Lists and descriptions of header items included in the output files that result from specifying *A*, **PRINT_CONVERGENCE** with formatted text; *B*, **PRINT_CONVERGENCE** with binary formatting; *C*, **PRINT_FLOW_RESIDUAL**; and *D*, **PRINT_RELATIVE_VOLUME_ERROR**.

C

SP	is the stress-period number.
TS	is the time-step number.
ITER	is the outer iteration number
LAY	is the layer of the head value.
ROW	is the row of the head value.
COL	is the column of the head value.
HEAD	is the solver calculated head for the current solver iteration.
FLOW_RESIDUAL	is the flow residual error for the current solver iteration.
VOL_RESIDUAL	is the flow residual error times the time-step length to yield the volume residual for the current solver iteration.
CELL_VOLUME	is volume of the model cell at LAY, ROW, COL.
DATE	is the month and year of the time step.
CELL_ID	is the model cell's unique identifier (ID). The cell ID is useful for re-sorting the file to look at sequential changes in data for a specific model cell. The ID is determined by the following formula: $CELL\_ID = COL + NCOL * (ROW - 1) + NCOL * NROW * (LAY - 1)$

D

SP	is the stress-period number.
TS	is the time-step number.
ITER	is the outer iteration number
LAY	is the layer of the head value.
ROW	is the row of the head value.
COL	is the column of the head value.
HEAD	is the solver calculated head for the current solver iteration.
REL_VOL_ERR	is the relative volume error.
VOL_RESIDUAL	is the flow residual error multiplied by the time-step length to yield the volume residual for the current solver iteration.
FLOW_RESIDUAL	is the flow residual error for the current solver iteration.
DATE	is the month and year of the time step.
CELL_ID	is the model cell's unique identifier (ID). The cell ID is useful for re-sorting the file to look at sequential changes in data for a specific model cell. The ID is determined by the following formula: $CELL\_ID = COL + NCOL * (ROW - 1) + NCOL * NROW * (LAY - 1)$

Figure 3.16. —Continued

NO_DIM_CHECK—Bypass Warning for Thin Model Cells

The keyword **NO_DIM_CHECK** causes the BAS package to not check for thin or narrow cells in a model. A thin cell is one that has a thickness five orders of magnitude smaller than the thickest cell of the same model layer. The "dimensions check" further looks at DELR and DELC—width and length of cells—for the same five order of magnitude difference in those cell dimensions in the same column or row. If the **NO_DIM_CHECK** keyword is present in the **OPTIONS** block, this check is bypassed, but MF-OWHM2 still checks for negative thicknesses, widths, and lengths of model cells and stops the simulation if it finds a negative value.

DEALLOCATE_MULT—Reduce MULT Package Memory

The MULT package allocates a large number of arrays that are only in use when constructing a MODFLOW Parameter (or Instance). Once the MULT arrays have been used for their initial purpose, they are no longer used for the rest of the simulation. To reduce the MF-OWHM2 memory (RAM) foot print, the MULT arrays can be released from memory after they have been used. This is done with the BAS package keyword **DEALLOCATE_MULT**. This option is only useful if the MULT package is in use and defines more than 100 multiplier arrays, which results in substantial memory savings.

TIME_INFO—External File of All Time Step Times

The legacy MODFLOW-2005 output files only include the stress period and time step and may additionally include the time-step length and total simulated time. MF-OWHM2 simulations that specify a starting date in the DIS package include both a decimal year and calendar date as part of the time step time. For post processing convenience, the model time-step's ending simulation time, date, and time-step length can all be printed to an external file. This file can serve both to check that the time discretization is set up correctly and for post-processors to have a look up table of stress period and time step to a corresponding date and decimal year. If a starting date is not specified, only the simulated time is printed.

Figure 3.17 shows the input format to print the time information and describes the new variables. Figure 3.18 lists and describes the header items included in the output file that results from specifying **TIME_INFO**.

```
BEGIN  OPTIONS
              TIME_INFO   Generic_Output
END
```

Figure 3.17. Structure of BAS package input used to print the time information to an external file. [*Generic_Output* specifies the location where the output is written.]

STEP	is a sequential number that represents the total number of model time steps solved.
SP	is the stress-period number.
TS	is the time-step number.
DELT	is the time-step length.
SIMTIM	is the total simulated time at the end of the time step.
DYEAR	is the decimal year at the end of the time step.
DATE	is the date at the end of the time step formatted as yyyy-mm-ddThh:mm:ss.

Figure 3.18. BAS package option TIME_INFO output file header description.

Budget_Groups—Splitting a Package Budget Information into Subgroups

The MODFLOW-2005 "Volumetric Budget" output (and cell-by-cell file) for each package includes the total volumetric flow rate in and out between the package and the groundwater flow equation. Only allowing for a single volumetric budget per package can be limiting if a package serves multiple purposes, such as simulating coastal and inland boundary conditions. Several packages were modified to allow the user to define Budget Groups that allow for multiple volumetric budgets per package. The groups contain a unique name that takes the place of the package name in the List file's Volumetric Budget (and in the CBC file), allowing for a more detailed accounting of how the flows for each of the packages interact with groundwater flow. The Budget Group feature is currently available for the MNW2, RIP, WEL, GHB, DRN, DRT, and RIV packages.

The Budget Group input requires two modifications to the standard input file. The first is to declare a **BUDGET_GROUPS** block using *Block Style* input (appendix 1). The **BUDGET_GROUPS** defines all the Budget Group names (BGROUPs). Each BGROUP defined in the block is then printed in the Volumetric Budget. The second modification is to associate all package features with one of the BGROUPs defined in the **BUDGET_GROUPS** block.

The **BUDGET_GROUPS** block is defined at the beginning of the package input file following the optional **PARAMETER** keyword. If there are other input blocks—such as **OPTIONS** or **LINEFEED**—placed after **PARAMETER**, the block order does not matter. In the **BUDGET_GROUPS** block, all budget group names, BGROUP, must be defined using one name per line. If the block is empty, then MF-OWHM2 ignores the **BUDGET_GROUPS** block and proceeds with the default MODFLOW budget name. Figure 3.19 is a list of the packages that **BUDGET_GROUPS** are available for and the location of the **BUDGET_GROUPS** block in the package.

The general input structure for the **BUDGET_GROUPS** block is shown in figure 3.20. The budget group names, BGROUPs, can be any alpha-numeric phrase, up to 16 characters long. This can be repeated for all the potential budget group names. For ease of handling output files, it is recommended, but not required, to keep the budget groups names shorter than 12 characters long. If there are numerous BGROUPs, then the block may specify a single *Generic_Input* (appendix 1) that points to a file containing a list of all the BGROUPs.

Package	Located after	Located before
DRN	**PARAMETER**	MXACTD
DRT	# Text – optional comment header	MXADRT
GHB	**PARAMETER**	**TABFILE**
MNW2	# Text – optional comment header	**TABFILE**
RIP	# Text – optional comment header	MAXRIP
RIV	**PARAMETER**	MXACTR
WEL	**PARAMETER**	**TABFILE**

Figure 3.19. List of the packages for which **BUDGET_GROUPS** are available and the location of the **BUDGET_GROUPS** block in the package.

```
BEGIN BUDGET_GROUPS
    # List Budget Names   -- Comments are allowed within the block
    BGROUP_1      # First      budget group name
    BGROUP_2      # Second  budget group name
      ⋮
    BGROUP_N      # Repeat as needed
END
```

Figure 3.20. General input structure for the **BUDGET_GROUPS** block. The word BGROUP_X, where X is replaced by a number from 1 to N, represents unique budget group name (BGROUP) that is up to 16 characters long. [⋮ is a place holder for the third through the N – 1 budget group names.]

If Budget Groups are in use (that is, the **BUDGET_GROUPS** block exists and contains at least one BGROUP), then every package feature must be associated with a BGROUP. For example, if the General Head Boundary (GHB) package contains the **BUDGET_GROUPS** block with at least one BGROUP, then every GHB cell must be associated with a BGROUP. The location for specifying the package feature BGROUP association depends on the package. Figure 3.21 presents the input location for BGROUP for each of the supported packages. Figure 3.22 is an example GHB package input—with and without budget groups—and associated example LIST file volumetric budget.

A

```
2a.    WELLID  NNODE  BGROUP
```

B

```
7.     Layer Row Column NPOLY BGROUP
```

C

```
4b.    Layer Row Column Qfact BGROUP [xyz] [TABNAM TSFAC [TABEXP]]
6.     Layer Row Column Q     BGROUP [xyz] [TABNAM TSFAC [TABEXP]]
```

D

```
4b.  Layer Row Column Bhead CondFact BGROUP [xyz] [TABNAM TSFAC [TABEXP]]
6.   Layer Row Column Bhead Cond     BGROUP [xyz] [TABNAM TSFAC [TABEXP]]
```

E

```
3b. Layer Row Column Elevation CondFact BGROUP [xyz]
5.  Layer Row Column Elevation Cond     BGROUP [xyz]
```

F

```
3b. Layer Row Column Elevation CondFact [LayR RowR ColR Rfprop] BGROUP [xyz]
5.  Layer Row Column Elevation Cond     [LayR RowR ColR Rfprop] BGROUP [xyz]
```

G

```
4b. Layer Row Column Stage CondFact Rbot BGROUP [xyz] [TABNAM TSFAC [TABEXP]]
6.  Layer Row Column Stage Cond     Rbot BGROUP [xyz] [TABNAM TSFAC [TABEXP]]
```

Figure 3.21. Packages that support budget groups and the associated input location that identifies the specific input feature with a budget group name (BGROUP). The supported packages are *A*, MNW2 package; *B*, RIP package; *C*, WEL package; *D*, GHB package; *E*, DRN package; *F*, DRT package; *G*, RIV package. [The number at the start of each input line—such as "2a." or "7."—is the MODFLOW-2005 input data item number.]

A

```
# General-Head Boundary package (GHB) for model with NLAY, NROW, NCOL = 1, 5, 2, respectively.
#
# MODFLOW-2005 default budget name is "HEAD DEP BOUNDS"
4     0     # MXACTC  IGHBCB
4     0     # ITMP  NP                        -> Stress period 1
 1   1   1    -5.0   100.0                    # LAY  ROW  COL  Bhead  Cond  BGROUP
 1   1   2    -5.0   100.0
 1   5   1   -10.0   100.0
 1   5   2   -10.0   100.0
```

B

```
# General-Head Boundary package (GHB) for model with NLAY, NROW, NCOL = 1, 5, 2, respectively.
#
# Declare Budget Groups – If not declared, MODFLOW-2005 default budget name is "HEAD DEP BOUNDS"
BEGIN BUDGET_GROUPS
                    GHB_OCEAN     # BGROUP_1
                    GHB_INLAND    # BGROUP_2
END
4     0     # MXACTC  IGHBCB
4     0     # ITMP  NP                        -> Stress period 1
 1   1   1    -5.0   100.0  GHB_OCEAN        # LAY  ROW  COL  Bhead  Cond  BGROUP
 1   1   2    -5.0   100.0  GHB_OCEAN
 1   5   1   -10.0   100.0  GHB_INLAND
 1   5   2   -10.0   100.0  GHB_INLAND
```

C

```
-----------------------------------------------------------------------------
 VOLUMETRIC BUDGET FOR ENTIRE MODEL AT END OF TIME STEP 1 IN STRESS PERIOD 1
-----------------------------------------------------------------------------

    CUMULATIVE VOLUMES      L**3       RATES FOR THIS TIME STEP      L**3/T
    ------------------                 ------------------------

          IN:                                IN:
          ---                                ---
            STORAGE =     57236.2616             STORAGE =      1846.3310
      CONSTANT HEAD =         0.0000       CONSTANT HEAD =         0.0000
    HEAD DEP BOUNDS =     20717.0354     HEAD DEP BOUNDS =       668.2915
           TOTAL IN =     77953.2970            TOTAL IN =      2514.6225

         OUT:                               OUT:
         ----                               ----
            STORAGE =     35599.0869             STORAGE =      1148.3576
      CONSTANT HEAD =         0.0000       CONSTANT HEAD =         0.0000
    HEAD DEP BOUNDS =     42354.2138     HEAD DEP BOUNDS =      1366.2650

          TOTAL OUT =     77953.3007            TOTAL OUT =      2514.6226
```

Figure 3.22. Example General Head Boundary (GHB) package input without and with budget groups: *A*, Example GHB input without **BUDGET_GROUPS**; *B*, Example GHB input that defines two budget group names; *C*, Example LIST file Volumetric Budget output when not using budget groups; and *D*, Example LIST file Volumetric Budget output with two GHB budget groups. [Bold lettering is used to indicate important parts of the figure. Note that if GHB does not use budget groups, then the default MODFLOW Volumetric Budget name for GHB is "HEAD DEP BOUNDS"]

D

```
------------------------------------------------------------------------------
 VOLUMETRIC BUDGET FOR ENTIRE MODEL AT END OF TIME STEP  1 IN STRESS PERIOD  1
------------------------------------------------------------------------------

    CUMULATIVE VOLUMES      L**3       RATES FOR THIS TIME STEP      L**3/T
    ------------------                 ------------------------

          IN:                                IN:
          ---                                ---
            STORAGE =     57236.2616           STORAGE =      1846.3310
      CONSTANT HEAD =         0.0000     CONSTANT HEAD =         0.0000
          GHB_OCEAN =     20717.0354         GHB_OCEAN =       668.2915
        GHB_INLAND =          0.0000       GHB_INLAND =         0.0000

           TOTAL IN =     77953.2970          TOTAL IN =      2514.6225

          OUT:                               OUT:
          ----                               ----
            STORAGE =     35599.0869           STORAGE =      1148.3576
      CONSTANT HEAD =         0.0000     CONSTANT HEAD =         0.0000
          GHB_OCEAN =          0.0000         GHB_OCEAN =         0.0000
        GHB_INLAND =     42354.2138       GHB_INLAND =      1366.2650

           TOTAL OUT =     77953.3007          TOTAL OUT =      2514.6226
```

Figure 3.22. —Continued

Two WEL Packages

The well package (WEL) source code was rewritten to mimic the input style of the GHB package with regard to TabFiles and includes the new LineFeed input format and additional options. The original WEL package is still retained in MF-OWHM2 to maintain backward compatibility with respect to the old TabFile inputs. The original WEL package is declared in the Name File by the package name "WEL1", and the new WEL package is declared by the package name "WEL". This allows for two sets of WEL packages to be defined; the only difference between the two is how **`TABFILE`** is specified, LineFeed as an input option, and the MF-OWHM2 **`OPTION`** block keywords.

A new option for the WEL package is provided by the keyword **`SMOOTHING`**, which activates MODFLOW-NWT (Niswonger and others, 2011) style smoothing of pumpage when the head approaches the bottom of a model cell for all flow packages (note that this smoothing works with all the solver packages). This helps alleviate the "Wet-Dry" problem where a high pumping rate in a well causes oscillations in the head calculation for a cell. The new input instructions are presented in figure 3.23.

A

FOR EACH SIMULATION

0. [#Text]
   Item 0 is optional—"#" must be in column 1. Item 0 can be repeated multiple times.

1a. [**PARAMETER** NPWEL MXL]
   This optional item must start with the keyword "PARAMETER".

2a. [**BEGIN BUDGET_GROUPS**] BLOCK STYLE INPUT*
   [BGROUP] REPEAT AS NEEDED, ONE PER LINE
   [**END BUDGET_GROUPS**]

2b. [**BEGIN LINEFEED** [FEEDOPT]]** BLOCK STYLE INPUT*
   [FEED_FILE] specified using *Generic_Input,*
   REPEAT AS NEEDED, ONE PER LINE
   [**END LINEFEED**]

2c. [**BEGIN OPTIONS**] BLOCK STYLE INPUT*
   [OPT] REPEAT AS NEEDED, ONE PER LINE
   [**END OPTIONS**]

   * Note that the order of the block style input does not matter.
   The order of items 2a, 2b, and 2c is recommended, but not required.
   In each block, comments are preceded with a "#" and blank lines are ignored.

   ** LINEFEED input structure is described in appendix 2.

3. [**TABFILE** NTAB FILEIO TIMEOPTION [**SPBASIS**] [**TABEQN**]]***
   [TABNAM TABLOCATION] READ NTAB TIMES IF NTAB>0,
   ONE TABNAM PER LINE

   *** Note that item 3 may also be specified in the OPTIONS block.
   If it is in the options block, then item 3 cannot be specified
   (double specifying TABFILE will raise an error).

4. MXACTW IWELCB

5. [PARNAM PARTYP Parval NLST [INSTANCES NUMINST]]

6a. [INSTNAM]

6b. [Layer Row Column Qfact [BGROUP] [xyz] [TABNAM TSFAC [TAB_EQN]]

B

FOR EACH STRESS PERIOD

7. ITMP NP

8. Layer Row Column Q [BGROUP] [xyz] [TABNAM TSFAC [TAB_EQN]
   –Repeat Item 8 ITMP times

9. [PNAME [INAME]]
   –Repeat Item 9 NP times

Figure 3.23. Structure of WEL package input format: *A*, for each simulation; *B*, for each stress period; *C*, explanation of input variables used by the keywords.

C

Text	is an optional character variable (699 characters) that is written to LIST file. Text may be specified on multiple lines, but a "#" must begin each line's Text.
NPWEL	is the number of WEL parameters defined in items 5 and 6. Note, a WEL parameter must be defined in items 5 and 6 and then made active using items 7 and 8 to have an effect in the simulation. If not specified, then NPWEL is set to 0.
MXL	is the maximum number of WEL cells to be defined using parameters.
MXACTW	is the maximum number of wells in use during any stress period. MXACTW includes wells defined using parameters and defined without using parameters. Wells defined in LINEFEED are added to MXACTW, so if only LINEFEED wells are used, then the initial value of MXACTW is set to 0. If LINEFEED is in use and IWELCB is defined in the OPTIONS block, then MXACTW is optional.
IWELCB	is a flag and a file unit number. It may be specified in the OPTIONS block or in Data Item 4. If IWELCB > 0, it is the unit number to which cell-by-cell flow terms will be written when "SAVE BUDGET" or a non-zero value for ICBCFL is specified in Output Control. If IWELCB = 0, cell-by-cell flow terms will not be written. If IWELCB < 0, well recharge for each well will be written to the LIST file if "SAVE BUDGET" or a non-zero value for ICBCFL is specified in Output Control.
BGROUP	is the budget group name, up to 16 characters long. If the BUDGET_GROUPS block is present and contains at least one BGROUP, then BGROUP is required in items 6b and 8 and is one of the names specified in the block.
FEED_FILE	is a separate file that represents the LineFeed alternate input (appendix 2). If FEED_FILEs are specified, then this entire input structure is optional (except for the LINEFEED block), and the wells in use are determined by the FEED_FILEs.

Figure 3.23. —Continued

***C** (continued)*

OPT is an optional keyword. Multiple options are allowed, but only one per line within the block. The following are the available options:

"**WEL_CBC** IWELCB" specifies the cell-by-cell unit number, IWELCB.
If this OPT is in the **OPTIONS** block, then IWELCB in Data Item 4 is not required.

"**NOPRINT**" specifies that WEL input lists are not to be written to the listing file.

"**SMOOTHING BY_FRACTION** PHIRAMP [Print_Smooth]" or
"**SMOOTHING BY_LENGTH** PHIRAMP [Print_Smooth]"

specifies that WEL pumping rate is smoothed towards zero when the head of the cell it pumps approaches the cell bottom. This option is available for all flow packages and applies to both confined and convertible layers.

PHIRAMP is the threshold fraction or length of the cell thickness that represents the saturated thickness when smoothing is activated. If the layer is declared as "confined", then the saturated thickness is the cell head minus the cell's bottom elevation.

BY_FRACTION indicates that PHIRAMP is a fraction between 0.00001 and 1.0, which is multiplied by the cell thickness to get the smoothing saturated thickness.

BY_LENGTH indicates that PHIRAMP is the smoothing saturated thickness.

Print_Smooth is an optional *Generic_Output* file that contains a transcript of the smoothing that is applied to wells.

"**AUXILIARY** abc" or "**AUX** abc" defines an auxiliary variable by the keyword abc.
For each auxiliary variable, abc, a corresponding integer, xyz, is read for each WEL cell in items 6b and 8. The order of input values for each auxiliary abc must be the same order in which values for the xyz are loaded.

"**TABPRINT**" specifies that WEL TabFiles write additional information to the LIST file.

Figure 3.23. —Continued

***C** (continued)*

TABFILE is a keyword that triggers reading subsequent TabFile information.

NTAB is the number of tabfiles that will be read.

FILEIO is a keyword that determines how TabFiles are handled in terms of memory usage and file input and output. The accepted values are

0 to indicate the entire TabFile is loaded into memory and processed.

1 to indicate only the part of the TabFile that pertains to the current time step is loaded into memory and processed (recommended option).

TIMEOPTION is a required keyword that must be specified by one of the two following keywords.

SIMTIME specifies that TabFile times use the model simulated time with time units specified by the DIS package input and a starting point of 0. This is the default operation if **TIMEOPTION** is not specified.

REALTIME specifies that TabFile times use decimal years that begin with the date specified in either the BAS package option **START_DATE** (recommended) or the DIS package keyword **STARTTIME** (legacy option).

IGNORE_TIME specifies that the time values specified be ignored when writing to the TabFile and instead an ordered one-to-one relationship between the time steps and TabFile entries (that is, a row in the TabFile) is assumed.

SPBASIS is an optional keyword; it indicates that TabFile times should be parsed based on the stress period rather than the time step. If the **IGNORE_TIME** option is used, then one row of the TabFile is loaded for each stress period.

TABEQN is an optional keyword; it indicates that TAB_EQN is an equation that is loaded and that the TabFile value is passed to it. Please see appendix 2 for more details.

TAB_EQN is an equation that is read when the **TABEQN** keyword is specified. This equation must be enclosed in single quotes and can perform any of the operations defined by the "ExpressionParser" (Hanson and others, 2014). The following are reserved names that are replaced with values:

TAB is replaced in TAB_EQN with the TabFile value for the current step. For example, '5*TAB + TAB^2' multiplies the TabFile value by 5 and then adds its square to that value.

SIM is replaced in TAB_EQN with the simulated time at the end of the time step.

REL is replaced in TAB_EQN with the decimal year at the end of the time step.

TABNAM is a unique name (maximum of 20 characters) that identifies the TabFile.

TSFAC is a scale factor multiplied with the final TabFile value. If TAB_EQN is specified, then it is done first and TSFAC is applied to the result.

TABLOCATION is either the location (relative or absolute) of the TabFile or the keyword **EXTERNAL** followed by a unit number that is defined in the Name file. The TabFile is self-counted, stored, and associated with the unique ID TABNAM.

Figure 3.23. —Continued

***C** (continued)*

PARNAM	is the name of a parameter up to 10 characters long and is not case sensitive.
PARTYP	is the type of parameter. For the WEL Package, the only allowed parameter type is **Q**, which defines values of the volumetric recharge rate.
Parval	is the parameter value. Values defined in the Parameter Value File supercede Parval.
NLST	is the number of wells that are included in a non-time-varying parameter or in each instance of a time-varying parameter.
LAYER	is the layer number of the model cell that contains the well.
ROW	is the row number of the model cell that contains the well.
COLUMN	is the column number of the model cell that contains the well.
Qfact	is the factor used to calculate well recharge rate from the parameter value. The recharge rate is the product of Qfact and the parameter value.
xyz	represents any auxiliary variables for a well defined in the **OPTIONS** block.
ITMP	is a flag and counter. It is optional if LineFeed is used and NPWEL = 0 If ITMP < 0, then non-parameter well data is reused If ITMP ≥ 0, then ITMP is the number of non-parameter well data to load in Item 8.
NP	is the number of parameters in use in the current stress period, where 0 ≤NP ≤ NPWEL. ITMP must be specified if NP is specified. Specifying NP is optional if NPWEL = 0.
Q	is the volumetric net-recharge rate. A positive value indicates recharge, and a negative value indicates discharge (pumping).
PNAME	is the parameter name in use for the stress period. NP parameter names are read.

Figure 3.23. —Continued

General Head Boundary (GHB) Flow Package Linkage and Other Updates

GHB Options in Block-Style Input

The GHB options originally were to the right of the first input line after the optional **PARAMETER** keyword. This structure caused problems if there were a large number of options or the user made a mistake with one of the options, which resulted in the rest being ignored. To overcome this limitation, the option part of the input was moved to a *Block-Style* input just after the **PARAMETER** keyword (see item 2b of the "GHB Input Structure" section for more details). If it is desired, the **PARAMETER** keyword may be placed in the **OPTIONS** block (either way it is parsed correctly).

The *Block-Style* input starts with the words "**BEGIN OPTIONS**", has one option per line, and is terminated with the words "**END OPTIONS**". In the block, any blank lines are automatically skipped, and all comments must be preceded by a "#" symbol. Figure 3.24 shows an example set of options.

```
BEGIN OPTIONS
   # COMMENT
   NOPRINT                          # SUPRESS GHB LIST WRITING
                                    # SKIPPED LINE
   AUX FLOW_PACK_COND               # AUXILIARY OPTION - Flow Package Linkage
   AUX VARIABLE_CONDUCTANCE         # AUXILIARY OPTION - Variable Conductance
END OPTIONS
```

Figure 3.24. Structure of the GHB package **OPTIONS** block with example options.

GHB Flow Package Linkage

The MODFLOW 2005 General Head Boundary Package (GHB) input required the user to specify a "conductance" that represented the porous medium that the boundary condition flow passed through to the aquifer. To provide a direct linkage to the MF-OWHM2 Flow package, the GHB package is now configured to optionally construct this conductance from the flow package horizontal hydraulic conductivity. The GHB can obtain its conductance from the flow package only when using the Layer Property Flow (LPF) and Upstream Weighting Flow (UPW) packages. To construct the conductance, the boundary condition (the $GHB_{i,j,k}$ cell, fig. 3.25) is assumed to be a model cell adjacent to the model cell it is associated with and has identical aquifer properties. The identical properties are hydraulic conductivity, row width (Δr), column length (Δc), and vertical height (Δv). If the model cell is convertible, then the GHB's vertical height is set equal to the cell's saturated thickness at the start of the time step.

The advantages of this linkage are that the boundary cell mimics the aquifer properties, making the input easier for the user, and that calibration is more constrained because the GHB conductance varies with the model cell conductance. Further, if the GHB cell is in a model layer defined as "convertible," where the conductance varies with the saturated thickness, then the cell's GHB calculated conductance also varies with saturated thickness. When this occurs, the saturated thickness is calculated using both the GHB boundary head (`BHead`) and the model cell's head (`HNEW`). For GHB cells using the vertical hydraulic conductivity, only use the saturated thickness and do not apply any correction factors for water-table conditions.

When the GHB package is linked to the flow package, the GHB input conductance (`Cond`) and the **`PARAMETER`** conductance factor (`CondFact`) are still part of the package input. Instead of being applied directly as the conductance, however, they act as a multiplier that is applied to the calculated conductance. This allows the user to scale the calculated GHB conductance to a larger or smaller value, if required for calibration or to improve the boundary flows.

To link the GHB package to the flow package, an auxiliary keyword called **`FLOW_PACK_COND`** must be specified in the GHB **`OPTIONS`** block. Note that MODFLOW-2005 auxiliary keywords are preceded by the word aux, so "**`AUX FLOW_PACK_COND`**" is the proper input. If the keyword is present, then each GHB cell's auxiliary value indicates which hydraulic conductivity direction to use to build the conductance (row, column, or vertical). Figure 3.26 lists the accepted auxiliary values, which indicate the GHB flow direction and if it varies with saturated thickness. The negative options (`-2` and `-1`, fig. 3.26) indicate that the GHB conductance does not vary with saturated thickness. The positive options `2` and `1` (fig. 3.26) indicate that if the model layer is declared as convertible, then the GHB conductance varies with saturated thickness.

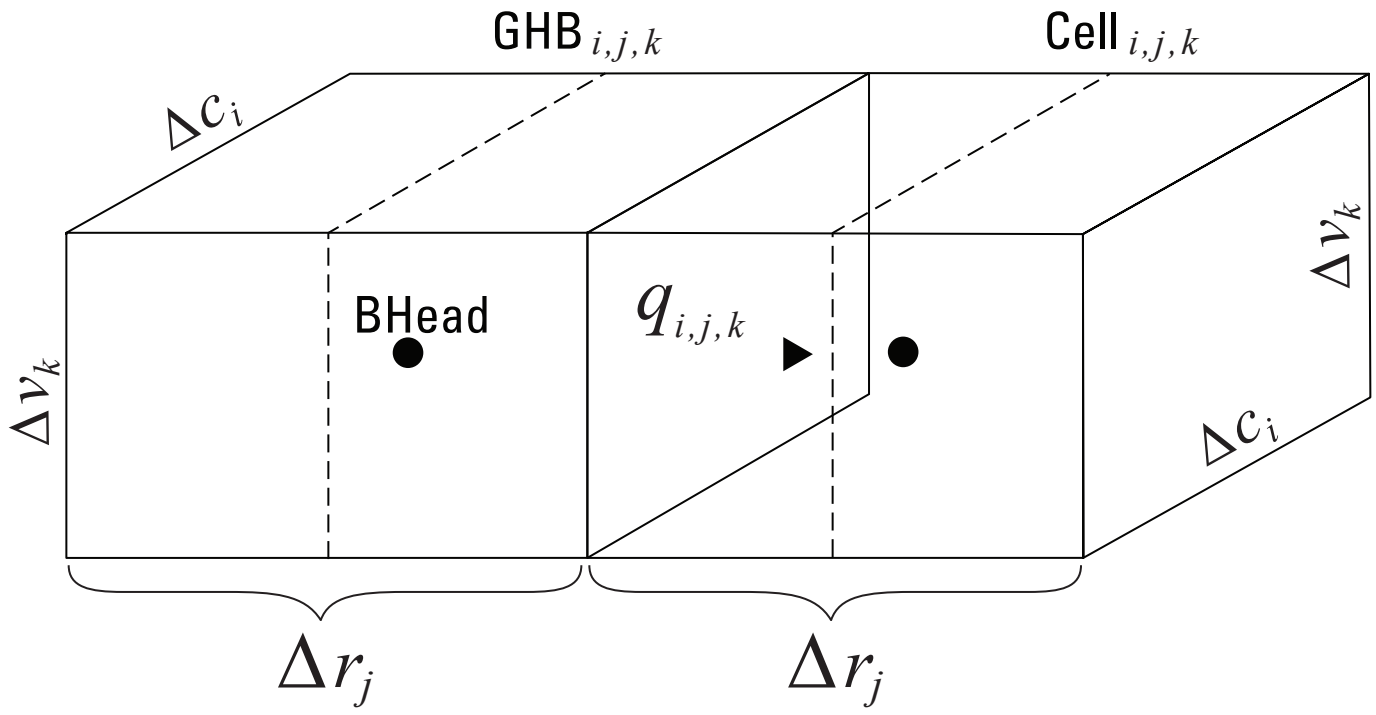


Figure 3.25. Example GHB (general head boundary) cell that flows in the row direction into model cell *i, j, k*. The BHead is the user-specified GHB boundary head value, $q_{i,j,k}$ is the flow rate, and $\text{Cell}_{i,j,k}$ is the model cell connected to the boundary condition.

```
-2 := GHB Flow in Column Direction,  conductance never varies with saturated thickness.
-1 := GHB Flow in Row Direction,     conductance never varies with saturated thickness.
 0 := GHB uses conductance specified in input, no flow package linkage.
 1 := GHB Flow in Row Direction,     conductance varies with saturated thickness, if convertible.
 2 := GHB Flow in Column Direction,  conductance varies with saturated thickness, if convertible.
 3 := GHB Flow in Vertical Direction, conductance never varies with saturated thickness.
```

Figure 3.26. Auxiliary values used to indicate which hydraulic conductivity direction to use to build the conductance.

GHB Variable Conductance

One of the limitations of the GHB package had been that the conductance does not vary with saturated thickness. The GHB package is now configured to vary the user-specified conductance linearly with saturated thickness regardless of the layer type definition, convertible or confined (**VARIABLE_CONDUCTANCE**). To invoke this feature, an auxiliary variable is required and declared as "**AUX VARIABLE_CONDUCTANCE**" in the Options section of the input structure. If the keyword is present, then each GHB cell's auxiliary value is set to one to indicate the conductance varies with saturated thickness or to zero to not vary. This auxiliary variable is different from "**AUX FLOW_PACK_COND**" because it will vary the conductance for both confined and convertible model cells.

The saturated thickness is calculated as a fraction (SAT_FRAC) of cell saturation either using the boundary head or the model cell head, whichever is larger. The saturated fraction is then multiplied by the user-specified conductance for calculating the boundary flows. This is similar to how UPW handles saturated thickness between two model cells. Figure 3.27 presents the decision tree for determining SAT_FRAC.

A GHB cell that has enabled **VARIABLE_CONDUCTANCE** always varies the conductance (fig. 3.27). Specifically, if the same GHB cell has the flow package linkage is in use (**FLOW_PACK_COND**), then the conductance always varies with saturated thickness for the auxiliary options -2, -1, 0, 1, and 2 (fig. 3.26). This is different compared to only the **FLOW_PACK_COND** option, which does not vary conductance for -2 and -1 and only varies for convertible layers for 1 and 2.

```
SAT_HEAD = MAX( BHead, HEAD )

IF     ( SAT_HEAD > TOP ) THEN
                        SAT_FRAC = 1
ELSEIF ( SAT_HEAD < BOT ) THEN
                        SAT_FRAC = 0
ELSE
                        SAT_FRAC = (SAT_HEAD – BOT) / (TOP – BOT)
```

where

BHead	is user specified boundary head
HEAD	is the model cell's head
TOP	is the model cell's top elevation
BOT	is the model cell's bottom elevation
SAT_FRAC	is the resulting saturated fraction

Figure 3.27. General Head Boundary (GHB) package decision algorithm used to calculate a model cell's saturated fraction (SAT_FRAC), which is multiplied with the GHB conductance (Cond) to determine boundary flows.

GHB Database Friendly Output

An optional database friendly output file is now available for the GHB package. This provides the user with the ability to check each time step's GHB input and easily load the results into data-column based software. This is important with the new "**AUX FLOW_PACK_COND**" and "**AUX VARIABLE_CONDUCTANCE**" because they calculate the conductance on the basis of aquifer properties and these can vary by time step. It also provides direct access to the resulting boundary flows for post-analysis. To activate the database output option, the keyword **DBFILE** followed by a *Generic_Output* file (appendix 1) name must be added to the **OPTIONS** Block (fig. 3.28). The *Generic_Output* file may be either a text file or binary. Figure 3.29 explains the meaning of each header.

If the GHB uses **BUDGET_GROUPS** (section "Budget_Groups—Splitting a Package Budget Information into Subgroups"), then it can print a database-friendly output file that only contains the model cells for a specific budget group. This functions identically to DBFILE option, but only writes a subset of the general head boundary cells. To activate the database output option, the keyword **BUDGET_GROUP_OUTPUT** is added to the **OPTIONS** block and is followed by the a BGROUP defined in the **BUDGET_GROUPS** block, then followed by a *Generic_Output* file name. Figure 3.30 is an example **OPTIONS** block that makes the database file "`./output/GHB_Inland_Flow.txt`", which only contains the GHB cells that are part of the GHB_LAND group.

```
BEGIN  OPTIONS
               DBFILE   Generic_Output
END
```

Figure 3.28. Structure of the General Head Boundary (GHB) package **OPTIONS** block input with the **DBFILE** keyword. This option writes GHB flows to a format to print the time information to an external file. [Generic_Output is the location to write the output file to.]

A

DATE_START	is the calendar date at the start of the time step (yyyy-mm-ddThh:mm:ss)	
PER	is the stress period number	
STP	is the time step number	
DELT	is the time step length	[T]
SIMTIME	is the total simulated time at the end of the time step in model units	[T]
LAY	is layer of the GHB cell	
ROW	is row of the GHB cell	
COL	is column of the GHB cell	
GHB_CONDUCTANCE	is the conductance of the GHB cell	[L^2/T]
GHB_HEAD	is the boundary head value of the GHB cell	[L]
HEAD	is the groundwater head for the model cell at LAY, ROW, COL	[L]
GHB_FLOW_RATE	is the flow rate from the boundary condition. Positive is inflow to groundwater	[L^3/T]

B

DATE_START	CHARACTER(19), starting date formatted as 'yyyy-mm-ddThh:mm:ss'
PER	INTEGER
STP	INTEGER
DELT	DOUBLE
SIMTIME	DOUBLE
LAY	INTEGER
ROW	INTEGER
COL	INTEGER
GHB_CONDUCTANCE	DOUBLE
GHB_HEAD	DOUBLE
HEAD	DOUBLE
GHB_FLOW_RATE	DOUBLE

Figure 3.29. General Head Boundary (GHB) package **OPTIONS** block keyword **DBFILE** text file header explanation and binary record structure: *A*, text-header explanation; and *B*, binary-record structure. [[T], unit of time in model units; [L], length in model units; [L^2/T], area per time in model units; [L^3/T], volumetric rate in model units; CHARACTER(19) indicates a record is 19 characters long (19 bytes); INTEGER is a 4-byte integer record; DOUBLE is a 8-byte floating-point number record.]

```
BEGIN  BUDGET_GROUPS

     # Assumes that GHB input is appropriately tagged with each BGROUP
     GHB_SEA      # BGROUP 1
     GHB_LAND     # BGROUP 2
END BLOCK

BEGIN OPTIONS
     # Print Database file of GHB cells associated with GHB_LAND budget group
     #                          BGROUP     Generic_Output
     BUDGET_GROUP_OUTPUT  GHB_LAND  ./output/GHB_Inland_Flow.txt
END OPTIONS
```

Figure 3.30. Example of General Head Boundary (GHB) package **BUDGET_GROUPS** and **OPTIONS** block input to illustrate the use of the **BUDGET_GROUP_OUTPUT** option.

GHB Input Structure

Figure 3.31 is the input directions for the GHB package. The **BUDGET_GROUPS** block is defined in "Budget_Groups—Splitting a Package Budget Information into Subgroups" section and the **LINEFEED** block is defined in appendix 2. If a **LINEFEED** block is used, then the entire input structure beyond that block is optional—except for the **LINEFEED** block—and the FeedFile's input is applied for each stress period.

A

FOR EACH SIMULATION

0. [#Text]
   Item 0 is optional—"#" must be in column 1. Item 0 can be repeated multiple times.

1a. [**PARAMETER** NPGHB MXL]
   This optional item must start with the keyword "PARAMETER".

2a. [**BEGIN BUDGET_GROUPS**] BLOCK STYLE INPUT*
   [BGROUP] REPEAT AS NEEDED, ONE PER LINE
   [**END BUDGET_GROUPS**]

2b. [**BEGIN LINEFEED** [FEEDOPT]]** BLOCK STYLE INPUT*
   [FEED_FILE] specified using *Generic_Input,*
   REPEAT AS NEEDED, ONE PER LINE
   [**END LINEFEED**]

2c. [**BEGIN OPTIONS**] BLOCK STYLE INPUT*
   [OPT] REPEAT AS NEEDED, ONE PER LINE
   [**END OPTIONS**]

   * Note that the order of the block style input does not matter.
   The order of items 2a, 2b, and 2c is recommended, but not required.
   In each block, comments are preceded with a "#" and blank lines are ignored.

   ** LINEFEED input structure is described in appendix 2.

3. [**TABFILE** NTAB FILEIO TIMEOPTION [**SPBASIS**] [**TABEQN**]]***
   [TABNAM TABLOCATION] READ NTAB TIMES IF NTAB>0,
   ONE TABNAM PER LINE

   *** Note that item 3 may also be specified in the OPTIONS block.
   If it is in the options block, then item 3 cannot be specified
   (double specifying TABFILE will raise an error).

4. MXACTB IGHBCB

5. [PARNAM PARTYP Parval NLST [INSTANCES NUMINST]]

6a. [INSTNAM]

6b. [Layer Row Column Bhead CondFact [BGROUP] [xyz] [TABNAM TSFAC [TAB_EQN]]

B

FOR EACH STRESS PERIOD

7. ITMP NP

8. Layer Row Column Bhead Cond [BGROUP] [xyz] [TABNAM TSFAC [TAB_EQN]]
   –Repeat Item 8 ITMP times

9. [PNAME [INAME]]
   –Repeat Item 9 NP times

Figure 3.31. Structure of General Head Boundary (GHB) package input format: *A*, for each simulation; *B*, for each stress period; and *C*, explanation of input variables used by the keywords.

C

Text	is an optional character variable (699 characters) that is written to LIST file. Text may be specified on multiple lines, but each line's Text must begin with a "#".
NPGHB	is the number of GHB parameters defined in items 5 and 6. Note, a GHB parameter must be defined in items 5 and 6 and then made active using items 7 and 8 to have an effect in the simulation. If not specified, then NPGHB is set to 0.
MXL	is the maximum number of GHB cells to be defined using parameters.
MXACTB	is the maximum number of GHB cells in use during any stress period. MXACTB includes GHB cells defined using parameters and defined without using parameters. GHB cells defined in LINEFEED are added to MXACTB, so if only LINEFEED GHB cells are used, then the initial value of MXACTB is set to 0. If LINEFEED is in use and IGHBCB is defined in the OPTIONS block, then MXACTB is optional.
IGHBCB	is a flag and a file unit number. It may be specified in the OPTIONS block or in Data Item 4. If IGHBCB > 0, it is the unit number to which cell-by-cell flow terms will be written when "SAVE BUDGET" or a non-zero value for ICBCFL is specified in Output Control. If IGHBCB = 0, cell-by-cell flow terms will not be written. If IGHBCB < 0, boundary flow for each GHB cell will be written to the LIST file if "SAVE BUDGET" or a non-zero value for ICBCFL is specified in Output Control.
BGROUP	is the budget group name, up to 16 characters long. If the BUDGET_GROUPS block is present and contains at least one BGROUP, then BGROUP is required in items 6b and 8 and is one of the names specified in the block.
FEED_FILE	is a separate file that represents the LineFeed alternate input (appendix 2). If FEED_FILEs are specified, then this entire input structure is optional (except for the LINEFEED block), and the GHB cells in use are set by the FEED_FILEs.

Figure 3.31. —Continued

***C** (continued)*

OPT is an optional keyword. Multiple options are allowed, but only one per line within the block. The following are the available options:

"**GHB_CBC** IGHBCB" specifies the cell-by-cell unit number, IGHBCB.
If this OPT is in the **OPTIONS** block, then IGHBCB in Data Item 4 is not required.

"**DBFILE** *Generic_Output*" writes GHB boundary flow to an external file.

"**BUDGET_GROUP_OUTPUT** BGROUP *Generic_Output*" writes for the specified GHB BGROUP the boundary flow to an external file.

"**NOPRINT**" specifies that GHB input lists are not to be written to the listing file.

"**AUXILIARY** abc" or "**AUX** abc" defines an auxiliary variable by the keyword abc. For each auxiliary variable, abc, a corresponding integer, xyz, is read for each GHB cell in items 6b and 8. The order of input values for each auxiliary abc must be the same order in which values for the xyz are loaded. The following are some of the supported auxiliary options.

"**AUX FLOW_PACK_COND**" specifies that xyz is a set of integers (flag values) that indicates how the GHB conductance is assembled.

"**AUX VARIABLE_CONDUCTANCE**" specifies that xyz is a set of integers (flag values) that indicates if saturated thickness is used to scale the GHB conductance.

"**TABPRINT**" specifies that GHB TabFiles write additional information to the LIST file.

Figure 3.31. —Continued

***C** (continued)*

TABFILE	is a keyword that triggers reading subsequent TabFile information.
NTAB	is the number of TabFiles that will be read.
FILEIO	is a keyword that determines how TabFiles are handled in terms of memory usage and file input and output. The accepted values are 0 to indicate the entire TabFile is loaded into memory and processed. 1 to indicate only the part of the TabFile that pertains to the current time step is loaded into memory and processed (recommended option).
TIMEOPTION	is a required keyword that must be specified by one of the two following keywords. **SIMTIME** specifies that TabFile times use the model simulated time with time units specified by the DIS package input and a starting point of 0. This is the default operation if **TIMEOPTION** is not specified. **REALTIME** specifies that TabFile times use decimal years that begin with the date specified in either the BAS package option **START_DATE** (recommended) or the DIS package keyword **STARTTIME** (legacy option). **IGNORE_TIME** specifies that the time values specified be ignored when writing to the TabFile and instead an ordered one-to-one relationship between the time steps and TabFile entries (that is, a row in the TabFile) is assumed.
SPBASIS	is an optional keyword; it indicates that TabFile times should be parsed based on the stress period rather than the time step. If the **IGNORE_TIME** option is used, then one row of the TabFile is loaded for each stress period.
TABEQN	is an optional keyword; it indicates that TAB_EQN is an equation that is loaded and that the TabFile value is passed to it. Please see appendix 2 for more details.
TAB_EQN	is an equation that is read when the **TABEQN** keyword is specified. This equation must be enclosed in single quotes and can perform any of the operations defined by the "ExpressionParser" (Hanson and others, 2014). The following are reserved names that are replaced with values: **TAB** is replaced in TAB_EQN with the TabFile value for the current step. For example, 'TAB/3 + TAB^2' divides the TabFile value by 3 and then adds its square to that value. **SIM** is replaced in TAB_EQN with the simulated time at the end of the time step. **REL** is replaced in TAB_EQN with the decimal year at the end of the time step.
TABNAM	is a unique name (maximum of 20 characters) that identifies the TabFile.
TSFAC	is a scale factor multiplied with the final TabFile value. If TAB_EQN is specified, then it is done first and TSFAC is applied to the result.
TABLOCATION	is either the location (relative or absolute) of the TabFile or the keyword **EXTERNAL** followed by a unit number that is defined in the Name file. The TabFile is self-counted, stored, and associated with the unique ID TABNAM.

Figure 3.31. —Continued

***C** (continued)*

PARNAM	is the name of a parameter; up to 10 characters long and is not case sensitive.
PARTYP	is the type of parameter. For the GHB Package, the only allowed parameter type is **GHB**, which defines values of the general-head boundary hydraulic conductance.
Parval	is the parameter value. Values defined in the Parameter Value File supersede Parval.
NLST	is the number of GHB cells that are included in a non-time-varying parameter or in each instance of a time-varying parameter.
LAYER	is the model layer of the cell affected by the head-dependent boundary.
ROW	is the model row of the cell affected by the head-dependent boundary.
COLUMN	is the model column of the cell affected by the head-dependent boundary.
Bhead	is the boundary head. This value can be over written by a TabFile.
CondFact	is the factor used to calculate hydraulic conductance from the parameter value. The calculated conductance is the product of CondFact and the parameter value. If the "**AUX FLOW_PACK_COND**" option is in use and the corresponding xyz is nonzero, then the conductance is calculated by the flow package and then multiplied by the CondFact and the parameter value.
xyz	represents any GHB auxiliary variables defined in the **OPTIONS** block. The values of auxiliary variables must be present in each repetition of items 6b and 8 if they are defined in item 2b. The values must be specified in the order used to define the auxiliary variables in item 2b.
ITMP	is a flag and counter. It is optional if LineFeed is used and NPGHB = 0 If ITMP < 0, then non-parameter GHB data is reused If ITMP ≥0, then ITMP is the number of non-parameter GHB cell to load in Item 8.
NP	is the number of parameters in use in the current stress period, where 0 ≤ NP ≤ NPGHB. ITMP must be specified if NP is specified. Specifying NP is optional if NPGHB = 0.
Cond	is the hydraulic conductance of the interface between the aquifer cell and the boundary. If the "**AUX FLOW_PACK_COND**" option is in use and the corresponding xyz is nonzero, then the conductance is calculated by the flow package and then multiplied by the Cond.
PNAME	is the parameter name in use for the stress period. NP parameter names are read.

Figure 3.31. —Continued

Figure 3.32 is an example GHB input that makes use of **BUDGET_GROUPS**, **TABFILES**, and includes the **FLOW_PACK_COND** auxiliary flag. The package input is split into two budget groups, GHB_OCEAN and GHB_LAND. Each group is associated with two GHB model cells to separate out an ocean boundary condition for an inland one in the volumetric budget. BHead for the GHB cells associated with an ocean boundary condition uses the **TABEQN** to translate the sea-level gauge values (−5.0) to a freshwater equivalent (−3.875). This takes the values in the TabFile ("File: GHB_Ocean_Level.txt", fig. 3.32*B*), then passes it through the specified function (for example, **'1.025*TAB + 1.25'**). Conversely, the inland boundary condition does not specify an equation, so it sets BHead to the TabFile value of −5.0 ("File: GHB_Inlnd_Level.txt", fig. 3.32*B*). All the GHB cells have set the xyz auxiliary flag to 2, which indicates that the conductance is calculated using the flow package's hydraulic conductivity in the column direction (fig. 3.26).

A

```
# General-Head Boundary package (GHB) Main Input
#        Model has NLAY, NROW, NCOL = 1, 5, 2, respectively. BAS Package START_DATE = 1/1/2000
#
# TabEqn to convert seawater water head (h)to freshwater equivalent head (hf) is
#   hf = 1.025*h - 0.025*Zp       where Zp is the cell center elevation
#                                 For this example, Zp = –50, so "–0.025*Zp" = + 1.25, which makes
#   hf = 1.025*h + 1.25
#
BEGIN BUDGET_GROUPS
                    GHB_OCEAN     # BGROUP_1
                    GHB_INLAND    # BGROUP_2
END
#
BEGIN OPTIONS
                    NOPRINT
                    #         Generic_Output
                    DBFILE  ./output/GHB_Flow.txt
                    #                      BGROUP       Generic_Output
                    BUDGET_GROUP_OUTPUT GHB_INLAND ./output/GHB_Inland_Flow.txt
                    #
                    AUX FLOW_PACK_COND   # indicates auxiliary flag “xyz” is specified
END
# if TABNAM is specified, then Bhead replaced by Tabfile result
#          NTAB  FILEIO  TIMEOPTION   [SPBASIS]  [TABEQN]
TABFILE   2     1      REALTIME             TABEQN
  #
  OCEAN_TAB  ./Tabfiles/GHB_Ocean_Level.txt   # TABNAM   TABLOCATION
  INLND_TAB  ./Tabfiles/GHB_Inlnd_Level.txt
  #
4     0      # MXACTC  IGHBCB
4     0      # ITMP  NP                          -> Stress period 1
# L  R  C  BHead  Cond  BGROUP       xyz  TABNAM      TSFAC   TAB_EQN
  1  1  1  NaN    1.0   GHB_OCEAN     2   OCEAN_TAB   1.0     ‘1.025*TAB + 1.25’
  1  1  2  NaN    1.0   GHB_OCEAN     2   OCEAN_TAB   1.0     ‘1.025*TAB + 1.25’
  1  5  1  NaN    1.0   GHB_INLAND    2   INLND_TAB   1.0
  1  5  2  NaN    1.0   GHB_INLAND    2   INLND_TAB   1.0
```

Figure 3.32. Example General Head Boundary (GHB) package input: *A*, GHB package main input file; *B*, TabFiles that are read by the GHB input; and *C*, example output that is produced from the GHB input.

B

File: GHB_Ocean_Level.txt

```
# Ocean sea level gauge
# Note the Tabfile has only two entries with the same value, so BHead is always set to -5
#
1/1/2000  -5.0
2/1/2000  -5.0
```

File: GHB_Inlnd_Level.txt

```
# Inland boundary condition level
# Note the Tabfile has only two entries with the same value, so BHead is always set to -10
#
1/1/2000  -10.0
2/1/2000  -10.0
```

C

File: GHB_Flow.txt

DATE_START	PER	STP	DELT	SIMTIME	LAY	ROW	COL	GHB_CONDUCTANCE	GHB_HEAD	HEAD	GHB_FLOW_RATE	GHB_BUD_GROUP
2000-01-01T00:00:00	1	1	31.0	31.0	1	1	1	959.72155	-3.875	-4.180	292.49909	GHB_OCEAN
2000-01-01T00:00:00	1	1	31.0	31.0	1	1	2	960.35909	-3.875	-4.053	166.99563	GHB_OCEAN
2000-01-01T00:00:00	1	1	31.0	31.0	1	5	1	920.90466	-10.0	-5.719	-3937.45729	GHB_INLAND
2000-01-01T00:00:00	1	1	31.0	31.0	1	5	2	921.38166	-10.0	-5.619	-3994.56188	GHB_INLAND

File: GHB_Inland_Flow.txt

DATE_START	PER	STP	DELT	SIMTIME	LAY	ROW	COL	GHB_CONDUCTANCE	GHB_HEAD	HEAD	GHB_FLOW_RATE
2000-01-01T00:00:00	1	1	31.0	31.0	1	5	1	920.90466	-10.0	-5.719	-3937.45729
2000-01-01T00:00:00	1	1	31.0	31.0	1	5	2	921.38166	-10.0	-5.619	-3994.56188

Figure 3.32. —Continued

Streamflow Routing (SFR) Upgrades

Time-Series Files and Line Feed

The Stream Flow Routing (SFR) Package is linked both to **LINEFEED** and Time Series Files to define the inflow at specified SFR segments. The **LINEFEED** and Time Series Files are described in detail in appendix 2. The SFR **LINEFEED**, described in appendix 2, is mentioned here to emphasize that it is a new feature of MF-OWHM2 that adds flexibility to the SFR input. To use **LINEFEED** with SFR, the **BEGIN LINEFEED** block must be specified at the start of the input file and closed with the keyword **END**. Time Series Files have a similar input structure as **LINEFEED** files because they are loaded as a Time Series File Group (appendix 2) in the block **BEGIN TIME_SERIES_INPUT**, which contains the Time Series File Group that loads the integer ID, which represents the SFR segment to which the Time Series File applies. Figure 3.33 shows the general input structure for loading time-series stream-inflow data from files using the **TIME_SERIES_INPUT** block within the SFR package.

A

```
BEGIN TIME_SERIES_INPUT
      #
      ISEG  Option  Generic_Input  # Repeat as needed
      #
END
```

B

ISEG	is the SFR segment identifier that has its inflow set by the Time Series File (TSF).
OPTION	is the Time Series File interpretation option. For a complete description, please see Appendix 2. The following are select options presented in order of recommended use:
	TIME_MEAN uses an elapsed time weighted average of all data points that lie within the time step start and ending dates.
	STEP_FUNCTION uses the closest data point located before the end of the time step.
	NEXT_VALUE uses the closest data point located after the end of the time step.
	NEAREST uses the closest data point located near the end of the time step.
	CONSTANT VALUE disables the TSF and only uses VALUE for all time.
	INTERPOLATE using the two data points that are closest to the end of the time step.
Generic_Input	is location of the time series file.

Figure 3.33. Inputting the time-series stream-inflow data using the **TIME_SERIES_INPUT** block within the SFR package: *A*, SFR TSF input structure; and *B*, explanation.

Separate Flow Output File

The Stream Flow Routing (SFR) Package now supports an optional database friendly output. This provides the user with the ability to check each time step's SFR output in a data-column based software for post-analysis.

To activate the database output file, the user must add to the SFR input's **OPTIONS** block the keyword **DBFILE** followed by a *Generic_Output* file name (fig. 3.34*A*). Please see the section on *Generic_Output* (appendix 1) for further details and options for *Generic_Output* files. Figure 3.34*B* is an example **OPTIONS** block that writes the database file to "./output/SFR_DB.txt". If a calendar date is provided (BAS package option **START_DATE**), then the date is included in the output; otherwise, "NaN" is placed in the DATE_START column. The output header is a listing of column names corresponding to contents (fig. 3.35).

A

```
BEGIN  OPTIONS
               DBFILE   Generic_Output
END
```

B

```
BEGIN  OPTIONS
               DBFILE   ./output/SFR_DB.txt
END
```

Figure 3.34. Structure of the Streamflow Routing (SFR) package **OPTIONS** block input with the **DBFILE** keyword. This option writes SFR flows to a format to print the time information to an external file: *A*, general input structure; and *B*, example that specifies a file location. [Generic_Output is the location to write the output file to.]

A

DATE_START	is the calendar date at the start of the time step (yyyy-mm-ddThh:mm:ss).	
PER	is the stress period number.	
STP	is the time step number.	
DELT	is the time step length.	[T]
SIMTIME	is the total simulated time at the end of the time step in model units.	[T]
SEG	is the stream segment identifier.	
RCH	is the stream reach identifier.	
FLOW_IN	is the streamflow in to the reach.	[L^3/T]
FLOW_SEEPAGE	is the seepage across the streambed. A positive value is flow out of stream reach to groundwater.	[L^3/T]
FLOW_OUT	is the streamflow out of the reach.	[L^3/T]
RUNOFF	is the overland runoff into stream reach.	[L^3/T]
PRECIP	is the precipitation that falls on the stream reach.	[L^3/T]
STREAM_ET	is the evapotranspiration rate out of the stream reach.	[L^3/T]
HEAD_STREAM	is the stream stage expressed as hydraulic head	[L]
HEAD_AQUIFER	is the head of the uppermost active model cell that the stream reach is superimposed on. That is, the first active model cell beneath it.	[L]
DEPTH_STREAM	is the stream reach's maximum streambed water depth above channel bed.	[L]
LENGTH_STREAM	is stream reach's total length.	[L]
HEAD_GRADIENT	is the head gradient from the streambed to the aquifer.	[L]
COND_STREAM	is the vertical hydraulic streambed conductance.	[L^2/T]
ELEV_UP_STREAM	is the streambed elevation at the upper part of the reach.	[L]
FLOW_WT	is the recharge from the unsaturated zone beneath stream reach. This is only included when the UZF package is specified in the Name file.	[L^3/T]
CHNG_UNSAT_STOR	is the change in unsaturated-zone storage beneath stream reach. This is only included when the UZF package is specified in the Name file.	[L^3/T]

Figure 3.35. Streamflow routing (SFR) package **OPTIONS** block keyword **DBFILE** text file header explanation and binary record structure: *A*, text-header explanation; and *B*, binary-record structure. [[T], unit of time in model units; [L], length in model units; [L^2/T], area per time in model units; [L^3/T], volumetric rate in model units; CHARACTER(19), indicates a record is 19 characters long (19 bytes); INTEGER, is a 4-byte integer record; DOUBLE, is a 8-byte floating-point number record.]

B

DATE_START	CHARACTER(19), starting date formatted as 'yyyy-mm-ddThh:mm:ss'
PER	INTEGER
STP	INTEGER
DELT	DOUBLE
SIMTIME	DOUBLE
SEG	INTEGER
RCH	INTEGER
FLOW_IN	DOUBLE
FLOW_SEEPAGE	DOUBLE
FLOW_OUT	DOUBLE
RUNOFF	DOUBLE
PRECIP	DOUBLE
STREAM_ET	DOUBLE
HEAD_STREAM	DOUBLE
HEAD_AQUIFER	DOUBLE
DEPTH_STREAM	DOUBLE
LENGTH_STREAM	DOUBLE
HEAD_GRADIENT	DOUBLE
COND_STREAM	DOUBLE
ELEV_UP_STREAM	DOUBLE
FLOW_WT	DOUBLE; always included; set to is 0.0 if UZF package is not specified in the Name file.
CHNG_UNSAT_STOR	DOUBLE; always included; set to is 0.0 if UZF package is not specified in the Name file.

Figure 3.35. —Continued

New SFR Options

The SFR package was modified to include a set of new options available in the **OPTIONS** block. The following subsections introduce briefly each of those options. The new options are **DBFILE**, **PRINT_GW_FLOW_RESIDUAL**, **AUTOMATIC_NEGATIVE_ITMP**, and **NOPRINT**. The option **DBFILE** is discussed in the "Separate Flow Output File," so it is not discussed here.

PRINT_GW_FLOW_RESIDUAL—Solver Information to External File

SFR option **PRINT_GW_FLOW_RESIDUAL** writes the convergence information for model cells that contain an SFR segment and reach to separate output files (fig. 3.36). This option is like the BAS package open **PRINT_FLOW_RESIDUAL** ("Obtaining Solver Information to External File" section), except that it only writes the solver flow residual error for model cells that contain an SFR reach.

A

```
BEGIN  OPTIONS
   #
   PRINT_GW_FLOW_RESIDUAL NTERM OUTER_START Generic_Output
   #
END
```

B

NTERM	is the number of cells, that are underneath SFR reaches, to print for each outer iteration. Each outer iteration then prints NTERM model cells that had the largest change since the previous iteration. NTERM cannot exceed the total number of SFR reaches.
OUTER_START	is the solver iteration to begin printing the NTERM cells. If set to zero, then it only prints the final-converged or last-solver iteration. If set to a negative number, then it prints either final-converged iteration or iterations after the maximum solver iterations minus the OUTER_START plus one.
Generic_Output	is the file written to.

C

SP	is the stress-period number.
TS	is the time-step number.
ITER	is the outer iteration number
LAY	is the layer of the head value.
ROW	is the row of the head value.
COL	is the column of the head value.
SEG	is the stream segment identifier.
RCH	is the stream reach identifier.
HEAD	is the solver calculated head for the current solver iteration.
FLOW_RESIDUAL	is the flow residual error for the current solver iteration.
VOL_RESIDUAL	is the flow residual error times the time-step length to yield the volume residual for the current solver iteration.
CELL_VOLUME	is volume of the model cell at LAY, ROW, COL.
DATE	is the month and year of the time step.
CELL_ID	is the model cell's unique identifier (ID). The cell ID is useful for re-sorting the file to look at sequential changes in data for a specific model cell. The ID is determined by the following formula: $CELL\_ID = COL + NCOL * (ROW - 1) + NCOL * NROW * (LAY - 1)$

Figure 3.36. SFR package option **PRINT_GW_FLOW_RESIDUAL**. This option outputs to an external file the convergence criteria for model cells that contain an SFR reach: *A*, expected input in the SFR **OPTIONS** block; *B*, explanation of input variables used; and *C*, descriptions of the header items included in the external file.

AUTOMATIC_NEGATIVE_ITMP—Only Define SFR Network Once

The standard SFR input requires that the entire stream network (segment and reaches) be specified for every stress period. This is necessary if the if there are changes in the network configuration, specified inflows, or specified diversions. If all the SFR stream inflow and diversions are defined by **LINEFEED**, **TABFILE**, or Time Series Files and the stream network does not change, then it is not necessary to define the network more than once. The SFR option **AUTOMATIC_NEGATIVE_ITMP** indicates that SFR should read the first stress period's input, then for each additional stress period reuse that input (which is equivalent to setting `ITMP = -1`). Specified stream inflow and diversions are then updated through the **LINEFEED**, **TABFILE**, or Time Series Files, but the network remains the same. This has the benefit of reducing the SFR input file size and improves the simulation execution speed.

NOPRINT Option—Reduce LIST File Writing

The SFR package writes a large amount of information to the **LIST** file. Most of this information just repeats the stream-network input. This information is valuable when initially developing the stream network. Once the network has been established, the amount of information written to the **LIST** file can greatly increase the file size and result in longer simulation run time. A new SFR option, **NOPRINT**, that suppresses most of the SFR output to the **LIST** file is now part of MF-OWHM2. To use **NOPRINT**, it must be placed in the SFR **OPTIONS** block.

PVAL, MULT, and ZONE Automatic Counting

PVAL, MULT, and ZONE packages may have comments, preceded by a "#" character, placed anywhere in their respective input structures. These packages offer automatic counting of the number of parameters, multiplier arrays, or zone arrays, respectively, by setting the count to –1. If the automatic counting is used, then any part of the input file that does not pertain to the actual input must be preceded by a "#" symbol (that is, commented). Otherwise there is a chance that the auto-count routine might include the comment in the count.

WARN Package

Traditionally, MODFLOW wrote all errors and warnings to the Listing file. Because of the length of the Listing file, it was difficult to identify important warnings that various packages may have raised. The Warning Package (WARN) is an optional output package that contains a copy of all warnings in one location, regardless of the package of origin. The warnings are still written to the Listing file, but the WARN package provides a convenient common location for all warnings and errors. WARN is declared just like any other package with the package name, WARN, followed by the optional unit number, then by the file name, then the optional post-keywords. If using Local Grid Refinement (LGR), the WARN package may only be specified for the parent grid, and all warnings from the child grids are written to that file as well.

LIST File Improvements

During a MODFLOW and MF-OWHM2 simulation run, a transcript of all operations is written to the Listing file (LIST). Previously, LIST always had to be enabled and declared at the start of the Name file. To remove this limitation, MF-OWHM2 does not require the LIST file for a simulation to continue and the LIST file may be specified anywhere in the Name file. The LIST file is now opened by *Generic_Output_OptKey* (appendix 2), which supports the post-keywords **SPLIT** and **BUFFER**. These two allow for the LIST file to be broken into multiple, smaller, files and to write the LIST to a buffered portion of RAM to improve speed.

LIST File Is Optional

For large simulation models, the LIST file can become quite large. The large file size can affect hard-drive performance, slowing down the overall simulation run time. This is particularly important during calibration, when multiple copies of the Listing file can occupy a large amount of hard-drive space. If run time and hard-drive space is an issue, the LIST file is now optional for a MF-OWHM2 simulation. If it is not specified in the Name file, then it is not included in the simulation. LIST suppression was included in the MF-OWHM by using the `LSTLVL` feature, but this feature has been removed from MF-OWHM2 version 2 now that the Listing file is optional.

Splitting the List File into Smaller Parts

Often the LIST file size can be problematic if it exceeds the allowable size to be opened in a text editor. That is, the file size exceeds the computers available RAM or is greater than 2 gigabytes for a 32-bit editor. To overcome this obstacle, the LIST file can now be split into multiple, smaller files. This allows for the LIST file to be created as a series of files that are approximately the same size (typically small enough to open in a text editor). The LIST file is split with the optional post-keyword **SPLIT** followed by the split size in megabytes (MB). After each time step, the size of the LIST file is checked; if it exceeds the split size, then that LIST file is closed, and output listing continues into a newly created "split file." When the LIST file is split, the newly created file has the same file name with a number appended to the end of it. The header in the original file is also included in each of the split files. Note that when the simulation is restarted, all split files are removed. For example, the following could be specified in the Name file: "LIST 55 ./List.txt **SPLIT** 900" to indicate that the LIST file is split when the file size exceeds 900 MB (note that 55 is the LIST file's unit number). The naming sequence used for each of the split LIST files is List.txt, then List01.txt, List02.txt, and so forth until the simulation ends.

Buffering the List File

Another performance improvement to the LIST file is the ability to reserve a buffer of RAM where LIST output can be pre-written. RAM is a type of the computer system memory that can read and write data faster than the hard drive. By default, in MF-OWHM2, the LIST file reserves a 32-kilobyte (KB) buffer in RAM; once 32 KB of text has been written to the RAM, it is transferred to the hard drive—to the actual LIST file. Although this increases the speed of a simulation by minimizing the frequency of writing to the hard drive, it updates the LIST file in 32-KB chunks (that is, when new text appears in the file). Another limitation of the buffer appears if power is interrupted to the computer; anything stored in the 32-KB buffered RAM is lost before writing to the LIST file. To change the size of the RAM buffer, the keyword **BUFFER** followed by the buffer size in kilobytes, may be included after the LIST file name in the Name file. Specify "**BUFFER 0**" to disable buffering, which results in the immediate writing of the LIST output to the hard drive. From empirical tests of buffering performance, using the **BUFFER** option for the LIST file in MF-OWHM2 resulted in a 5-percent reduction in simulation runtime by changing the buffer size from 256 to 1024 KB. "LIST 55 ./List.txt **SPLIT** 900" is an example that buffers the LIST file in 1024 KB of RAM.

Name File Updates

The Name file's read utilities were updated to allow unit numbers to be optional for packages and to support a set of new keywords that improve file operations. The sections that follow will provide details of each improved feature.

New Keywords: Buffer, Read, Write

All Name file packages, DATA files, and DATA(BINARY) files support a new set of keywords that are specified to the right of the file name. Although they are optional, the effects of these keywords improve the input and output performance of MF-OWHM2, and they may be specified in any order. As with the use of LIST file buffering (see "Buffering the List File"), the keyword **BUFFER** followed by a buffer size in kilobytes is available to all packages, DATA files, and DATA(BINARY) files. By default, all Name file's files are buffered to 32 KB of RAM; to disable this, the keyword "**BUFFER 0**" must be added for each package in the Name file. The post-keyword **READ** is like the old keyword **OLD** and forces the file to be opened for 'read-only' access. This allows the same file to be accessed simultaneously by different programs or packages. For example, if two MF-OWHM2 simulations were running, they could access the same DATA file if it was opened with read only access. For **READ** files, the value of **BUFFER** should not exceed the size of the input file (rounded upward to a power of 2); otherwise, RAM is wasted because only as much as the entire file can be pre-loaded into the buffer. The opposite of the **READ** flag is the keyword **WRITE** (similar to the old keyword **REPLACE**), which indicates that the file is an output file that only allows writing access to it. For **WRITE** files, the buffer size should be set after considering both how often the user wants to see the output in the disk file during runtime and the amount of data that will be written to it. Typically, larger output files, such as the cell-by-cell type, should have a larger buffer than smaller ones, such as the HOB package's output. Empirical tests showed that buffer sizes greater than 1024 KB do not improve write performance. Note that Package files are input files, so only the keyword **READ** applies; if the keyword **WRITE** is used, it is ignored. Figure 3.37 indicates the keywords available to LIST, Packages, DATA, or DATA(BINARY) file-opening operations. Note that the keyword **SPLIT** is described in the "Splitting the List File into Smaller Parts" section. Figure 3.38 is an example Name file that makes use of the post-keywords.

	Supported Post-Keywords					
Keyword	BUFFER	READ	WRITE	REPLACE	OLD	SPLIT
LIST	X	—	X	X	—	X
Packages	X	X	—	—	X	—
DATA	X	X	X	X	X	—
DATA(BINARY)	X	X	X	X	X	—

Figure 3.37. Post-keywords available in the Name file that can be specified after the file name. [Supported post-keywords are marked with an X. Packages refer to all the MODFLOW and MF-OWHM2 packages. Abbreviation: —, not supported; X, is supported.]

```
LIST     50 ./LISTING_FILE.txt  SPLIT 500  BUFFER 1024
BAS6     51 ./BAS_FILE.txt                 BUFFER 64
DIS      52 ./DIS_FILE.txt      READ       BUFFER 64
LPF      53 ./LPF_FILE.txt      READ
DATA     54 ./CBC_FILE.txt      WRITE      BUFFER 512
DATA     55 ./INPUT_FILE.txt    READ       BUFFER 128
DATA   1979 ./SCOTT_FILE.txt    READ       BUFFER 16
DATA   1982 ./NO_BUFFER.txt     WRITE      BUFFER 0
```

Figure 3.38. Example Name file to illustrate the use of the post-keywords SPLIT, BUFFER, READ, and WRITE.

Associate a Variable Name with Text

The Name file now supports assigning a variable name to set of text. The variable name may then be used as part of any file name in the Name file. A variable is declared by enclosing a word within percentage symbols (%) at the start of any line and is associated with the first space-delimited text to the right of it. If it is necessary to associate the variable with text that contains spaces, then it must be enclosed within matching single or double quotes. The variable name, enclosed in quotes, then may be used as any part of the file directory paths and is replaced by the text with which it is associated. This option allows for easily changing output file locations or pointing to different package groups to run different scenarios.

Figure 3.39 is an example Name file that assigns three variables. The first variable declares a common output location for all the data files, the second declares a common input location, and the third illustrates that an entire package path can be stored.

Package Version Numbers Optional

In MODFLOW-2005, if a package had been updated to a new version, its name in the Name file included the version that was used. Because MODFLOW-2005 did not support, nor include, the previous version of the package, the version number became a potential source of input error. MF-OWHM2 loads both the package with the version number and without if there is only one unique version of the package, so this does not apply to the two WEL packages described in this appendix nor the MNW packages (MNW1 and MNW2). Figure 3.40 lists the packages that can have their names simplified.

```
#
# Variable name set via % % where package is declared.
# The declared variables are %OUTPUT%, %DATA_IN%, and %DIS_PACK%
#
# %VarName%  VarString
%OUTPUT%    ./Output/
#
%DATA_IN%   ./Input/Files/
#
# Quotes are required if the text assigned to the variable contains spaces
#
%DIS_PACK%   "./Scenario A/DIS.txt"
#
# Variable text is substituted where MF-OWHM2 finds %variable%
#
# MAIN PACKAGES ****************************
#
# Package          Unit    File Path
DIS                50    %DIS_PACK%
#
BAS                51    %DATA_IN%BAS.bas
LPF                52    %DATA_IN%LPF.lpf
GHB                53    %DATA_IN%GHB.chd
#
PCGN               54    ./SolverInput/PCGN.pcgn
#
# Output Files ***************************************************
# Package,         Unit,   File Path               Post-Keywords
LIST               60    %OUTPUT%LIST.lst  BUFFER 1024  SPLIT 900
WARN               61    %OUTPUT%Warn.txt  BUFFER 0
#
DATA(BINARY)   69    %OUTPUT%CBC.cbc       WRITE BUFFER 1024
DATA           96    %OUTPUT%HeadOut.fhd   WRITE
#
# Input Files – Necessary for EXTERNAL Unit references ********************
#
DATA           85    %DATA_IN%IBOUND.txt   READ BUFFER 64
```

Figure 3.39. Example Name file to illustrate how to associate a variable name with text. The variable name is initialized by surrounding a keyword with percent signs (%). [BUFFER X indicates the file should be first written to X kilobytes of RAM; SPLIT Y indicates that when the file size exceeds Y megabytes that it should start a new one; WRITE indicates the file is write only; READ indicates the file is read only.]

Package Name	Alternative Name
BCF6	BCF
LMT6	LMT
HFB6	HFB
HUF2	HUF
BFH2	BFH
SWI2	SWI

Figure 3.40. Packages that support using their name without the version number at the end.

Name File Unit Numbers Are Optional for Packages

The normal set up for the Name file is to have the package name followed by an integer number that represents its Fortran file-unit number. This caused problems for users because each package had to have a unique unit number or there could be strange effects while running a simulation. The unit numbers are now optional, and when not specified, they are auto-assigned a value. Unit numbers are still required for `DATA` and `DATA(BINARY)` input types because they are referenced by unit number (`UNIT`) in other some package inputs (for example, **`EXTERNAL`** `UNIT`). Figure 3.41 is an example Name file that does not specify a unit number for each of the packages.

```
#
# Example illustrates that Package Unit numbers are optional.
#
# MAIN PACKAGES ****************************
#
# Package          [Unit]   File Path
DIS                       "./Scenario A/DIS.txt"
#
BAS                       ./Input/BAS.bas
LPF                       ./Input/LPF.lpf
GHB                       ./Input/GHB.chd
#
PCGN                      ./SolverInput/PCGN.pcgn
#
# Output Files ****************************************************
# Package,         [Unit],  File Path              Post-Keywords
LIST                      ./Output/LIST.lst BUFFER 1024  SPLIT 900
WARN                      ./Output/Warn.txt  BUFFER 0
#
# Unit numbers are still necessary for DATA and DATA(BINARY) files.
# Package,         Unit,    File Path              Post-Keywords

#
DATA(BINARY)   69   ./Output/CBC.cbc      WRITE BUFFER 1024
DATA           96   ./Output/HeadOut.fhd  WRITE
#
# Input Files – Necessary for EXTERNAL Unit references *********************
#
DATA           85   ./Input/IBOUND.txt  READ BUFFER 64
```

Figure 3.41. Example Name file to illustrate that specifying a unit number is optional for packages. [BUFFER X indicates the file should be first written to X kilobytes of RAM; SPLIT Y indicates that when the file size exceeds Y megabytes that it should start a new one; WRITE indicates the file is write only; READ indicates the file is read only.]

Observation Process (HOB, DROB, GBOB, RVOB) New Features

Write Observations at End of Each Time Step

The Observation Process packages—Head Observation (HOB), Drain Observation (DROB), General Head Observation (GBOB), and River Observation (RVOB)—were modified to allow printing to a separate file at the end of every time step. Previously, this was only available by setting values to the input variables `IUHOBSV`, `IUDROBSV`, `IUGBOBSV`, and `IURVOBSV`, such that observations were written to a separate file only if the simulation completed successfully. The problem with this is that during model calibration, a calibration software runs multiple simulation models with different parameter sets and if one of the evaluated parameter sets results in a simulation failure, then the calibration software stops because of a failure to load the Observation Process output files.

This issue was resolved by allowing the observation process packages to write to an external file at every time step. If the time step is before an observation's time, then the observation value written is either set to HDRY (HOB) or zero (DROB, GBOB, RVOB). This additional output file is initiated by specifying at the start of the input file—that is, after any comments and before specifying Data Set 1—the keyword **`TIME_STEP_PRINT_ALL`** followed by the location of where to write the observations (specified with *Generic_Output*). Formally this is defined as "**`TIME_STEP_PRINT_ALL`** *Generic_Output*", where *Generic_Output* (appendix 1) specifies the file to which the observation package writes its observations at every time step. If it is desired to use the same unit number as the one specified by `IUHOBSV`, `IUDROBSV`, `IUGBOBSV`, and `IURVOBSV`, then the keywords **`EXTERNAL`** or **`DATAUNIT`** should be used with the same unit number as that specified for the corresponding `IUxxOBSV` variable.

Observations Include Calendar Dates Implicitly

The HOB was modified so that if the simulate specifies a starting date (Calendar Dates section), then it calculates the calendar date of each observation and prints it as part of the output. This is an automatic feature that is included whenever HOB is in use and BAS contains the keyword **`START_DATE`**. Figure 3.42 shows how the new HOB header when the simulation includes calendar dates.

Here, the two new header columns (fig. 3.42) are `DATE` and `DECIMAL_YEAR`. The DATE column contains the calculated date of the observation in the following format: yyyy-mm-dd, where mm is the month, dd is the day of the month, and yyyy is the Gregorian year (note the 24-hour clock time is not included). The `DECIMAL_YEAR` column contains the decimal year equivalent of the observation date and takes into account leap years (365- or 366-day years). Figure 3.43 is an example of the HOB output for one observation when the BAS contains the keyword **`START_DATE`**.

```
"SIMULATED EQUIVALENT" "OBSERVED VALUE" "OBSERVATION NAME” DATE DECIMAL_YEAR
```

Figure 3.42. Head Observation Process (HOB) output file header when the BAS package includes the option **`START_DATE`**. If the BAS option is not included, then DATE and DECIMAL_YEAR are not printed.

```
"SIMULATED EQUIVALENT" "OBSERVED VALUE" "OBSERVATION NAME” DATE      DECIMAL_YEAR
 7.96124649048E+01   4.03400001526E+01   SEB_0038_1        1979-4-23 1979.3082192
```

Figure 3.43. Head Observation Process (HOB) output file example when the BAS package includes the option **`START_DATE`**. If the BAS option is not included, then DATE and DECIMAL_YEAR are not printed.

NWT Solver Upgrades

The Newton (NWT) solver previously only worked when using the Upstream Stream Weighting (UPW) package. The UPW package input is nearly identical to the Layer Property Flow (LPF) package, except for one integer flag, `IPHDRY`, that specifies if dry model cells should be set to the MODFLOW HDRY value or retain the head value returned from the solver. The BAS package was modified to allow the use of the NWT Solver with the LPF flow package and to allow for the UPW package to work with the other solvers—for example, PCG, PCGN, and GMG. When the NWT Solver and LPF are used together, dry cell values are retained (`IPHDRY=0`), upstream weighting is still used, and the following LPF options are not supported: **`STORAGECOEFFICIENT`**, **`CONSTANTCV`**, **`THICKSTRT`**, **`NOCVCORRECTION`**, **`NOVFC`**. In short, only the **`NOPARCHECK`** option is supported. If the other options are present, they are read, but not applied. Conversely, if the UPW package is not used with the NWT solver, it does not apply the upstream weighting, and it functions identically to LPF. This feature is provided for the convenience of not having to rebuild the LPF file to test a simulation using the NWT solver nor to rebuild the UPW when evaluating one of the other MODFLOW solvers.

The NWT solver performs a thin-cell check at the start of the simulation. This check automatically removes model cells—by setting their `IBOUND` value to zero—that have a vertical thickness considered too thin. The problem with this check is it can automatically remove a significant part of the model grid that is intended to be a part of the simulation. For example, a three-layer model that has a very "thin" second layer, representing an aquitard, could have most of the model cells removed from the simulation by changing their `IBOUND` values to zero. This would disable any vertical flow between the first and third layers, changing the aquitard to an aquifuge, without the user realizing it. To prevent this from happening NWT, by default, no longer performs the thin-cell check. If this feature is still desired, then the NWT input in the **`OPTIONS`** part of DATASET 1 now supports the keyword **`THIN_CELL_CHECK`**. The keyword **`THIN_CELL_CHECK`** may be placed either before or after the keyword **`CONTINUE`**, if it is present.

A

FOR FIRST STRESS PERIOD OR ALL STRESS PERIODS

0. [#Text]
   Item 0 is optional—"#" must be in column 1. Item 0 can be repeated multiple times.
1. MXITER ITER1 NPCOND IHCOFADD
2. HCLOSE RCLOSE RELAX NBPOL IPRPCG MUTPCG DAMPPCG [DAMPPCGT]

B

OPTIONAL FOR EACH SUBSEQUENT STRESS PERIOD

3. MXITER ITER1 NPCOND IHCOFADD
4. HCLOSE RCLOSE RELAX NBPOL IPRPCG MUTPCG DAMPPCG [DAMPPCGT]

Figure 3.44. Preconditioned Conjugate-Gradient (PCG) solver package input format: *A*, for each simulation; *B*, for each stress period (optional); and *C*, explanation of input variables used by the keywords.

C

Text	is an optional character variable (699 characters) that is written to LIST file.
MXITER	is the maximum number of outer iterations. For a linear problem, MXITER should be 1 unless more than 50 inner iterations are required, when MXITER could be as large as 10. For nonlinear problems, a larger number is required. If MXITER in Data Item 3 is set to -1, then the previous stress period's convergence criteria are reused, and the rest of Data Item 3 variables and Data Item 4 are not read.
ITER1	is the number of inner iterations. Typically, a value between 30 and 100 is sufficient.
NPCOND	is the flag used to select the matrix conditioning method. Accepted values are 1 is for modified incomplete Cholesky (for use on scalar computers); 2 is for polynomial (for use on vector computers or to conserve computer memory).
IHCOFADD	is a flag that determines what happens to an active cell surrounded by dry cells. Accepted values are 0 – Cell converts to dry regardless of the HCOF value. (Former default before option). 1 – Cell converts to dry only if the HCOF is 0 (no head-dependent stresses or storage terms).
HCLOSE	is head-change criterion for convergence, in model units of length [L]. If the maximum absolute value of head change from all nodes during an iteration is less than or equal to HCLOSE and RCLOSE is also satisfied, the iteration stops
RCLOSE	is residual criterion for convergence, in model units of cubic length per time [L^3/T]. If the maximum absolute value of the residual at all nodes during an iteration is less than or equal to RCLOSE and HCLOSE is also satisfied, the iteration stops
RELAX	is relaxation parameter when NPCOND = 1. RELAX is not used if NPCOND is not 1. Usually, RELAX = 1.0, but sometimes values of 0.99, 0.98, or 0.97 may improve speed.
NBPOL	is read, but not used if NPCOND ≠ 2. If NPCOND = 2, then NBPOL indicates whether the estimate of the upper bound on the maximum eigenvalue is 2.0 (set NBPOL = 2), or whether the estimate is to be calculated (set NBPOL ≠ 2). Convergence is generally insensitive to this parameter.
IPRPCG	is the printout interval for PCG. If IPRPCG is equal to 0, it is changed to 999. The maximum head change (positive or negative) and residual change are printed for each iteration of a time step whenever the time step is an even multiple of IPRPCG. This prints out also at the end of each stress period regardless of the value of IPRPCG.
MUTPCG	is a flag that controls printing of convergence information from the solver to the LIST file. Accepted values are 0 – to print tables of maximum head change and residual each iteration. 1 – to print only the total number of iterations. 2 – to not print convergence information. 3 – to print only when convergence fails level 0 information.
DAMPPCG	is the damping factor. Set to 1 to indicate no damping or set to >0 – to specify the damping factor applied to steady-state and transient stress periods. <0 – to only apply dampening, as -1×DAMPPCG, to steady-state stress periods.
DAMPPCGT	is only read if DAMPPCG<0 and is the damping factor for only transient stress periods.

Figure 3.44. —Continued

PCG and NWT Solver Loading of Convergence Criteria by Stress Period

The PCG and NWT solver packages were modified to allow specification of the convergence criteria by stress period. This feature allows strict tolerances for stress periods that require high accuracy, whereas stress periods that are not as important for obtaining good mass balances can have relaxed tolerances. This customization of solver tolerances by stress period can improve the speed of the simulation.

Input for the PCG and NWT solvers remains the same, except the solver checks whether there are additional convergence criteria specified on the line after the normal input. If the criteria are present, then they are read in at the beginning of each stress period, except the first period (the first stress period is specified by the normal solver-input structure). If any of the input variables fail to be read in the subsequent stress-period information (for example, a blank line or text is found where a number is expected), then the solver writes a warning to the Listing File, stops reading further stress-period-specific solver information, and uses the previously accepted solver information for the remainder of the simulation. This feature has a similar input structure to that of the HFB2 package described in Hanson and others (2014). Input instructions for a modified PCG solver is shown in figure 3.44: *A* is an example of the standard format for the PCG input files, and *B* is read for each additional stress period for which different convergence criteria are read. Items written in blue text can change with each stress period, whereas items written in red text must match what was specified in original input (for example, you cannot change the PCG preconditioner NPCOND in a single simulation run). Figure 3.45 is NWT input structure that includes the new stress period input.

A

FOR FIRST STRESS PERIOD OR ALL STRESS PERIODS

0. [#Text]

Item 0 is optional—"#" must be in column 1. Item 0 can be repeated multiple times.

1. HEADTOL FLUXTOL MAXITEROUT THICKFACT LINMETH IPRNWT IBOTAV **OPTIONS**
[DBDTHETA DBDKAPPA DBDGAMMA MOMFACT BACKFLAG [MAXBACKITER BACKTOL BACKREDUCE]]

If LINMETH = 1 and **OPTIONS** contains "**SPECIFIED**" then read Item 2 .

2. [MAXITINNER ILUMETHOD LEVFILL STOPTOL MSDR]

If LINMETH = 2 and **OPTIONS** contains "**SPECIFIED**" then read Item 3 .

3. [IACL NORDER LEVEL NORTH IREDSYS RRCTOLS IDROPTOL EPSRN HCLOSEXMD MXITERXMD]

B

OPTIONAL FOR EACH SUBSEQUENT STRESS PERIOD

4. HEADTOL FLUXTOL MAXITEROUT THICKFACT LINMETH IPRNWT IBOTAV **OPTIONS**
[DBDTHETA DBDKAPPA DBDGAMMA MOMFACT BACKFLAG [MAXBACKITER BACKTOL BACKREDUCE]]

Figure 3.45. Newton (NWT) solver package input format: *A*, for each simulation; *B*, for each stress period (optional); and *C*, explanation of input variables used by the keywords. [Note that each data item should be specified on a single line of text in the input file.]

C

Text	is an optional character variable (699 characters) that is written to LIST file.
HEADTOL	is head-change criterion for convergence, in model units of length [L].
FLUXTOL	the residual criterion for convergence, in model units of cubic length per time [L^3/T].
MAXITEROUT	is the maximum number of outer iterations allowed.
THICKFACT	is threshold, specified as a fraction of the cell thickness, that switches from varying conductance linearly with respect to saturated thickness to varying it parabolically. For example, a 15-meter-thick cell with a THICKFACT = 0.1 indicates that when the saturated thickness is less than 1.5 meters, the conductance varies parabolically. The THICKFACT fraction must be between 1×10^{-6} and 1.
LINEMETH	is the flag used to select the matrix conditioning method. Accepted values are 1 – GMRES matrix solver is used. 2 – χMD matrix solver is used. χMD cannot be used with Local Grid Refinement (LGR).
IPRNWT	is a flag to print to the LIST file convergence output. Set to 0 to not print and >0 to print.
IBOTAV	is a flag that indicates whether corrections are made to groundwater head relative to the cell-bottom elevation if the cell is surrounded by dewatered cells. A value of 1 indicates that a correction be made, and a value of 0 indicates no correction is to be made. This influences the convergence speed and is problem dependent.
OPTION	is a set keywords (required and option) that set up solver parameters. One, and only one, of the following options must be selected A rule of thumb is to first try using **MODERATE** and then test **COMPLEX** or **SIMPLE**. Often **SIMPLE** results in faster running models, but with larger mass-balance errors or convergence problems later in the simulation.
SPECIFIED	indicates that the items enclosed in brackets, [ ], in data item 1 are specified. Also, depending on LINEMETH, data items 2 and 3 are specified. Note that Data Items 2 and 3 are only read once.
SIMPLE	sets up the NWT parameters that work well for nearly linear models. The option results in faster model runs than the other options but can yield erroneous results and poor estimates of mass balances for highly nonlinear models. This is best for models that are confined or consist of a single unconfined layer that is thick enough to contain the water table within a single layer.
MODERATE	sets up the NWT parameters that work well for moderately nonlinear models. This is used for models that include nonlinear stress packages and models that consist of one or more unconfined layers.
COMPLEX	sets up the NWT parameters that work well for nonlinear models. This is used for models that include nonlinear stress packages and models that consist of one or more unconfined layers representing complex geology and surface-water and groundwater interaction.
	The following options are optional, and one or all may be specified.
CONTINUE	indicates that simulation continues, even when convergence fails. This is identical to BAS option **NO_FAILED_CONVERGENCE_STOP**.
THIN_CELL_CHECK	indicates that NWT will remove model cells within the same layer that have a thickness less than 1 percent of the largest cells thickness. This was the default NWT behavior before this option was added. Since, it was removed, the option was added to enable to legacy feature.

Figure 3.45. —Continued

***C** (continued)*

The following variables in data item 1 are read only if **OPTION** includes the keyword **SPECIFIED**.

DBDTHETA	is a coefficient used to reduce the weight applied to the head change between nonlinear iterations. DBDTHETA is used to control oscillations in head. Values range between 0.0 and 1.0, and larger values increase the weight (decrease under-relaxation) applied to the head change.
DBDKAPPA	is a coefficient used to increase the weight applied to the head change between nonlinear iterations. DBDKAPPA is used to control oscillations in head. Values range between 0.0 and 1.0, and larger values increase the weight applied to the head change.
DBDGAMMA	is a factor used to weight the head change for iterations n – 1 and n. Values range between 0.0 and 1.0, and greater values apply more weight to the head change calculated during iteration n.
MOMFACT	is the momentum coefficient "m" and ranges between 0.0 and 1.0. Greater values apply more weight to the head change for iteration n.
BACKFLAG	is a flag used to specify whether residual control is to be used. A value of 1 indicates that residual control is active, and a value of 0 indicates residual control is inactive.
MAXBACKITER	is the maximum number of reductions (backtracks) in the head change between nonlinear iterations. A value between 10 and 50 works well. MAXBACKITER is only read if BACKFLAG > 0.
BACKTOL	is the proportional decrease in the root-mean-squared error of the groundwater-flow equation used to determine if residual control is required at the end of a nonlinear iteration. BACKTOL is only read if BACKFLAG > 0.
BACKREDUCE	is a reduction factor used for residual control that reduces the head change between nonlinear iterations. Values can be between 0.0 and 1.0, and smaller values result in smaller head-change values. BACKREDUCE is only read if BACKFLAG > 0.

Data Sets 2 and 3 are only read once. Please see Niswonger and others (2011) for the exact meaning.

Figure 3.45. —Continued

References Cited

Hanson, R.T., Boyce, S.E., Schmid, W., Hughes, J.D., Mehl, S.M., Leake, S.A., Maddock, T., III, and Niswonger, R.G., 2014, One-water hydrologic flow model (MODFLOW-OWHM): U.S. Geological Survey Techniques and Methods 6–A51, 120 p., https://doi.org/10.3133/tm6A51.

Harbaugh, A.W., 2005, MODFLOW-2005—The U.S. Geological Survey modular ground-water model—The ground-water flow process: U.S. Geological Survey Techniques and Methods 6–A16, variously paginated, https://pubs.usgs.gov/tm/2005/tm6A16/.

Niswonger, R.G., Panday, S., and Ibaraki, M., 2011, MODFLOW-NWT—A Newton formulation for MODFLOW-2005: U.S. Geological Survey Techniques and Methods 6–A37, 44 p., https://pubs.usgs.gov/tm/tm6a37/pdf/tm6a37.pdf.

Appendix 4. Consumptive Use and Evapotranspiration in the Farm Process

One of the fundamental parts of the Farm Process (FMP) is the calculation of consumptive water use according to land use. For MF-OWHM2, the consumptive use is defined as the consumption of water from any source to meet a land's use (or crop's) water consumption (evaporation and transpiration). The word "land use" in this report is synonymous with the word "crop" and may represent any simulated element that has a consumptive use. The term "crop" may represent a plant functional group that is either an aggregate of multiple crops or splits the same crop type into multiple plant functional groups, "crops," because the properties differ substantially. An example of aggregating is to lump different sets of berries, such as blackberries, blueberries, and raspberries, together as one crop and using one set of properties to define them. Plant functional groups may be location specific, however. For example, it may be necessary to split an FMP "crop"—such as strawberries—into two FMP "crops" because of spatial variability in the crop's properties such that coastal strawberries differ from inland strawberries.

The sections that follow describe how FMP calculates consumptive use, which becomes the demand component, and then attempts to satisfy it within water supply constraints. At a minimum, FMP requires specification for each crop includes a consumptive-use value, root length, soil capillary fringe, fraction of transpiration, and fraction of inefficient losses to surface water that comes from precipitation. If there are crops that receive applied water (irrigation), then irrigated crops additionally require an irrigation type flag, irrigation efficiency, fraction of evaporation from irrigation, and fraction of inefficient losses to surface water from irrigation. If the consumptive use is defined by a crop coefficient, then the user is required to specify a reference evapotranspiration. Published crop coefficients should cite the description of its associated reference evapotranspiration—typically, it is the evapotranspiration from well-watered grass of uniform height, actively growing, and completely shading the ground. For consistency in a simulation, it is required that all inputted crop coefficients refer to the same definition of reference evapotranspiration.

There are three methods available in the FMP to define how a land use's "root zone" interacts directly with groundwater and if the crop's potential consumptive use is reduced. The root-zone interaction with groundwater includes crop consumption of groundwater (called root groundwater uptake), crop anoxia reduction, and non-idealized soil stress reduction. The first method disables root groundwater update and consumptive-use reductions; the second is a linear root groundwater uptake and anoxia-limitation concept that takes into account the overlap between the groundwater capillary fringe and the root depth (*linear root response*); and the third is an analytical pseudo-steady state "reduced consumptive use" soil-moisture, soil-stress concept (*analytical root response*). If the *analytical root response* is used, then FMP requires the user to specify four stress-response root pressures, PSI, and the soil type in which the crop is grown—note PSI, or ψ, is in units of hydraulic head. Negative ranges of pressure heads reflect unsaturated conditions, whereas positive hydrostatic pressures reflect saturated conditions that act on the active roots zone and determine the extent to which the crop is under anoxic conditions, consumes groundwater, or wilts.

A fundamental assumption in FMP is that crops are grown in a unit-cropped area, which is partitioned into transpiratory-cropped area (leaf matter covered area) and evaporative-cropped area (exposed landscape). For most FMP simulations, the unit-cropped area is analogous to a single model cell's top-surface area. This assumption allows for the crop's consumptive use to be split into transpiration and evaporation components. These two components are further subdivided on the basis of consumption of water from groundwater, precipitation, and irrigation (applied water). This results in the consumptive use being split into six parts: transpiration from root groundwater uptake (T_{uptake}), transpiration from precipitation (T_p), transpiration from irrigation (T_{irr}), evaporation from irrigation (E_{irr}), evaporation from precipitation (E_p), and evaporation from groundwater (E_{gw}). The sum of these six terms is the final consumptive use (CU). Consumption of water is always calculated first, then any remaining water either becomes surface-water runoff or percolates through the root zone to groundwater as deep percolation. The priority order for satisfying consumption is always as follows:

1. Transpiration from root-groundwater uptake.
2. Transpiration from precipitation.
3. Transpiration from irrigation.
4. Evaporation from irrigation.
5. Evaporation from precipitation.
6. Evaporation from groundwater.

Although this appendix may seem complicated, with numerous variable definitions, the FMP minimum input is rather small. For a Crop that is not irrigated, FMP requires specification of the following for each unit-cropped area:

1. Spatial location of the crop.
2. Potential Consumptive Use specified as
   - the crop potential evapotranspiration or
   - a crop coefficient and reference evapotranspiration.
3. Root depth below land surface of the crop.
4. Fraction of transpiration (FTR, leaf covered area divided by unit-cropped area).
5. Precipitation runoff coefficient called fraction of inefficient losses from precipitation to surface water (FIESWP).

Although not directly related to the Crop, the following must be specified:

6. Soil capillary fringe length.
7. Precipitation rate (can be zero).

If the crop is irrigated, then the following, additionally, must be specified:

8. Sources of irrigation
   - Imported Water (Non-Routed Delivery, NRD).
   - Surface Water (Semi-Routed Delivery, SRD).
   - Groundwater well (FWEL).
9. Irrigation efficiency (OFE).
10. Irrigation runoff coefficient called fraction of inefficient losses from irrigation to surface water (FIESWI).
11. Fraction of Evaporation from Irrigation, which is the part of the unit-cropped area that is not covered by leaf matter but receives irrigation (FEI).

Equation Variable Definitions

This section provides a listing of variables and keywords used in this appendix and are presented in alphabetical order. The following name list is provided for convenience, and the names are formally defined in this appendix.

Area	is the unit-cropped area [L^2].
Anoxia	is level of anoxia, $0 \leq$ Anoxia [L^3/T].
BareArea	is the “bare soil” or fallow area—area not defined with an FMP land use type [L^2].
CIR	is the crop irrigation requirement assuming perfect efficiency [L^3/T].
CIR_{MAX}	is maximum quantity of irrigation with perfect irrigation efficiency that can be applied to a crop [L^3/T].
CU_{Bfinal}	is the final consumptive use for BareArea [L^3/T].
CU_{final}	is the final consumptive use for the unit-cropped area [L^3/T].
CU_{ini}	is the crop potential consumptive use [L/T].
$D_{irrigation}$	is the irrigation demand to satisfy the potential transpiration, $\tilde{T}_{pot}$ [L^3/T].
DP	is water that infiltrates to groundwater, deep percolation [L^3/T].
E_{act}	is the actual evaporation over the unit-cropped area [L^3/T].
E_{Bgw}	is the evaporation from groundwater for bare soil [L^3/T].
E_{Bgw_pot}	is the potential evaporation from groundwater [L^3/T].

E_{Bp} is the evaporation from precipitation on bare soil; it is always equal to E_{Bpot} [L^3/T].

E_{Bpot} is the potential evaporation from bare-soil evaporation or reference evapotranspiration [L^3/T].

E_{CIR} is the potential consumption of irrigation water necessary to fully satisfy evaporation [L^3/T].

E_{GW} is the amount of evaporation from groundwater [L^3/T].

E_{GWpot} is the potential evaporation from groundwater [L^3/T].

$E_{irrigation}$ is the actual consumption of irrigation water as evaporation, which may be limited by T_{CIR} [L^3/T].

E_p is the quantity of precipitation that is consumed by evaporation [L^3/T].

E_{pot} is the potential evaporation from groundwater and precipitation in the unit-cropped area [L^3/T].

ET_c is the crop potential evapotranspiration [L/T].

ET_{CIR} is potential consumption of irrigation water necessary to fully satisfy evapotranspiration [L^3/T].

ET_{ref} is the reference evapotranspiration [L/T].

FEI is the fraction of the unit-cropped area that has irrigated water applied to the exposed soil part of the unit-cropped area (that is, the area not covered by the crop's leaf matter), $0 \le FEI \le 1 - FTR$ [-].

FIESWI is the fraction of inefficient losses from irrigation to surface water, $0 \le FIESWI \le 1$ [-].

FIESWP is the fraction of inefficient losses from precipitation to surface water, $0 \le FIESWP \le 1$ [-].

FTR is the fraction of transpiration, $0 \le FTR = K_{cb}/K_c \le 1$ [-].

FWEL FMP supply well, a groundwater well supply source.

IRR is the total applied irrigation to a unit-cropped area [L^3/T].

IRR_{Ex} is the excess irrigation that becomes either runoff or deep percolation, $IRR_{Ex} = IRR - IRR \cdot OFE$ [L^3/T].

K_c is the crop coefficient, $0 < K_c$ [-].

K_{cb} is the basal (transpiration) crop coefficient, $0 < K_{cb} \le K_c$ [-].

NRD Non-Routed Delivery, an imported water supply (conveyance not simulated).

OFE is the irrigation efficiency, called on-farm efficiency, $0 < OFE \le 1$ [-].

P is precipitation that falls over the unit-cropped area [L/T].

P_{eff} is effective precipitation in the unit-cropped area [L/T].

P_{Bare} is precipitation that falls over the BareArea [L/T].

P_{BEpot} is the potential evaporation of precipitation from bare-soil area [L^3/T].

P_{Epot} is the potential evaporation from precipitation from unit-cropped area [L^3/T]; if $E_{pot} > P_{Epot}$, then $P_{Epot} = E_{pot}$.

P_{Tpot} is the potential transpiration from precipitation [L^3/T]; if $T_{pot} > P_{Tpot}$, then $P_{Tpot} = T_{pot}$.

$Precip_{Ex}$ is the excess precipitation that becomes either runoff or deep percolation [L^3/T].

SRD Semi-Routed Delivery, a surface-water supply source (SFR stream reach).

SW_R is surface-water runoff from excess precipitation and irrigation [L^3/T].

T_{act} is the actual transpiration in the unit-cropped area [L^3/T].

T_{CIR} is the potential consumption of irrigation water necessary to fully satisfy transpiration [L^3/T].

$T_{irrigation}$ is the actual consumption of irrigation water as transpiration, which may be limited by available irrigation supplies [L^3/T].

T_p is the quantity of precipitation that is consumed by transpiration [L^3/T].

T_{pot} is the potential transpiration in the unit-cropped area [L^3/T].

$\tilde{T}_{pot}$ is potential transpiration after any anoxia reduction [L^3/T].

T_{surf} is potential transpiration demand to consume precipitation and irrigation [L^3/T].

T_{uptake} is the transpiration satisfied from root groundwater uptake [L^3/T].

Consumptive Use, Crop Coefficients, and Crop Fractions

MF-OWHM2 attempts to satisfy the potential consumptive use (CU_{ini}) with water supplies that originate from root groundwater uptake, precipitation, or applied water (irrigation) from imported water, surface-water, and groundwater sources. An imported water source represents water available for consumption, but its conveyance and delivery are not directly simulated. Surface-water sources are simulated as a diversion for a stream network. Groundwater sources originate from pumping wells that extract water from the underlying aquifers. Applied water supplies are subject to efficiency losses that reduce the water supply available for consumption by the land use. The initial demand (DMD_{ini}) of the land use is the total water required by CU_{ini} plus any inefficient losses incurred during consumption. If the land use is a crop, then it may have its CU_{ini} lessened because of anoxia from a high water-table in the root zone. If the water supply, less inefficient losses, cannot meet the CU_{ini}, less by anoxia, then the final consumptive use (CU) is reduced to equal the water supply. The excess water from applied water efficiency losses and precipitation that is not consumed by a land use becomes either surface runoff or percolates through the root zone to groundwater (deep percolation).

Consumptive use in the Farm Process (FMP) is either a directly specified potential consumptive-use ($CU_{specified}$) or calculated from a crop coefficient (K_c) and reference evapotranspiration (ET_{ref}). A crop coefficient is a published crop property that can be multiplied by a reference evapotranspiration to obtain a potential evapotranspiration of the crop (ET_c). Reference evapotranspiration represents the evapotranspiration from a standardized vegetated surface. The specific vegetated surface is defined by the crop coefficient, which is commonly an extensive surface of well-watered grass of uniform height, actively growing, and completely shading the ground—from this definition, the grass used to define ET_{ref} has a $K_c = 1$. In FMP, ET_{ref} is specified spatially, aligned with the surface model grid, whereas crop coefficients are defined for each simulated crop.

Estimates of reference evapotranspiration can be derived from direct measurements, physically based equations, or empirically based equations. An example of a direct measurement system is the California Irrigation Management Information System (CIMIS) station system, developed in 1982. The most detailed approximation of reference evapotranspiration is the Penman-Monteith equation (Snyder and Eching, 2002), but requires the largest amount of input information and field observations, including daily mean temperature, wind speed, relative humidity and solar radiation. Samani (2000) determined that temperature and radiation explain at least 80 percent of reference evapotranspiration. The Priestley-Taylor equation was developed as a simplified substitute to the Penman-Monteith equation that calculates reference evapotranspiration using only solar radiation (irradiance) observations and replaces the Penman-Monteith aerodynamic term with a dimensionless empirical factor. Another method is the Hargreaves-Samani equation (Hargreaves and Samani, 1982, 1985; Hargreaves and others, 1985; Hargreaves and Allen, 2003), which estimates reference evapotranspiration using temperature and solar radiation data. Climate simulation models, such as the Basin Characterization Model (Flint and Flint, 2014; Flint and others, 2015), may also provide the spatial approximations of reference evapotranspiration that are necessary for crop coefficients.

Obtaining values of crop coefficients involves both researching published values and communicating with local water districts and agriculture organizations for their measured coefficients. The crop coefficient should not be confused with a basal crop coefficient (K_{cb}), which is multiplied by a reference evapotranspiration to obtain potential transpiration for the crop (T_{pot}) rather than the crop's potential evapotranspiration (ET_c). For FMP, the potential evapotranspiration is the crop's initial-potential consumptive use. The calculation of the consumptive use is as follows:

$$CU_{ini} = ET_c = K_c \cdot ET_{ref} \tag{4.1}$$

where

CU_{ini} is the crop potential consumptive use [L/T];
ET_c is the crop potential evapotranspiration [L/T];
K_c is the crop coefficient [-], $0 < K_c$; and
ET_{ref} is the reference evapotranspiration [L/T].

The FMP input defines a crop's initial-potential consumptive use either as a value, with the keyword **CONSUMPTIVE_USE**, or as a crop coefficient, with the keyword **CROP_COEFFICIENT**. If both are supplied, then their sum is used by FMP—that is, $CU_{ini} = CU_{user\text{-}specified} + K_c \cdot ET_{ref}$. To make the FMP input's CU_{ini} clearer, it is recommended to only specify for each crop either a consumptive use or a crop coefficient—that is, not both at the same time for the same crop. The option to include both allows for a mixing of input or to separate crop consumption into two input parts, but its use is not recommended. Demands that are not consumed by the crop may be specified as an added demand for irrigated water (keyword **ADDED_CROP_DEMAND**).

The FMP has multiple methods for calculating the actual consumption of water that depend on different combinations of crop properties and available water sources. In all cases, the final supply must equal the demanded water by either lowering excess supply to meet demand or curtailing demand to meet the insufficient supply. The curtailing of demand is contradictory to economic descriptions of demand, but remains in this report to be consistent with past FMP publications that are focused on water balances, such that final water supply is always equal to final water demand. The order of consumption of different water supplies is always root groundwater uptake, precipitation, and then applied water. The order of consumption of applied water is always imported water sources, surface-water sources, and then groundwater sources. In FMP, imported water sources are called Non-Routed Deliveries (NRD), surface-water sources are called Semi-Routed Deliveries (SRD), and groundwater sources are called FMP supply wells (FWEL).

Root groundwater uptake and anoxia reduction are calculated either using the *linear root response* or the *analytical root response*. Both methods determine consumption of groundwater by a crop based on the relative distances between the water table and the capillary fringe with the land surface elevation and root zone. The water table is calculated by the MF-OWHM2 flow package and the capillary fringe is specified as an FMP soil property keyword **`CAPILLARY_FRINGE`**. The root zone is the vertical space from the land surface elevation to a user-supplied root depth. The land surface elevation and root depth are defined as part of the FMP input with the keywords **`SURFACE_ELEVATION`** and **`ROOT_DEPTH`**, respectively.

In the *linear root response*, the crop's consumption of groundwater, through its roots, increases linearly with increasing overlap between the crop's root zone and the water table plus the capillary fringe. If there is no overlap, there is no uptake. When the water table reaches the bottom of the root depth, the full consumptive use is fulfilled by root groundwater uptake (there is maximum overlap between the capillary fringe and root zone). After that point, increases in the water table result in anoxic conditions and the crop's consumptive use decreases as the water table increases its overlap with the root zone. The *analytical root response* calculates the pressure head acting on the crop's root zone (ψ_{root}) using the water-table elevation, capillary fringe, and an analytical function (appendix 5). FMP combines ψ_{root} with a set of four user-supplied stress-response root zone pressures (ψ_1, ψ_2, ψ_3, ψ_4, keyword **`ROOT_PRESSURE`**) to quantify a crop's anoxia, groundwater consumption, and wilting. A soil saturation is anoxic to the crop's roots when ψ_{root} is greater than ψ_2 and is lethal to the crop when ψ_{root} is greater than ψ_1. A crop can consume groundwater directly when ψ_{root} is greater than ψ_4 and the consumption of groundwater is optimal (that is, consumptive use is fully satisfied with groundwater) when ψ_2 is greater than ψ_{root} is greater than ψ_3. Wilting (no consumption of groundwater from the roots) occurs when ψ_4 is greater than ψ_{root}. The four pressures can be split into three zones of stress response, which are anoxic conditions (ψ_{root} is greater than ψ_2), groundwater consumption (ψ_{root} is greater than ψ_4), and wilting (ψ_4 is greater than ψ_{root}).

FMP's calculation of actual consumption starts by splitting the initial consumptive use into a transpiration component and evaporation component (fig. 4.1). This split is determined by a user-supplied crop property called the fraction of transpiration (FTR, keyword **`TRANSPIRATION_FRACTION`**). The fraction of transpiration is the crop-covered area (area of transpiration) per unit-cropped area and the remaining area is non-vegetated "exposed soil" (area of evaporation). The exposed soil area is assumed to only have evaporation of water from groundwater, precipitation, or excess irrigation that is applied to the exposed soil. The unit-cropped area is the surface area of a model cell or user-specified fraction of the model cell's surface area (multi-crop per cell option). The formal definition of FTR is the ratio the basal crop coefficient to the crop coefficient:

$$FTR = \frac{K_{cb}}{K_c} \tag{4.2}$$

where

FTR is the fraction of transpiration [-], $0 \leq FTR \leq 1$;
K_c is the crop coefficient [-], $0 < K_c$; and
K_{cb} is the basal (transpiration) crop coefficient [-], $0 < K_{cb} \leq K_c$.

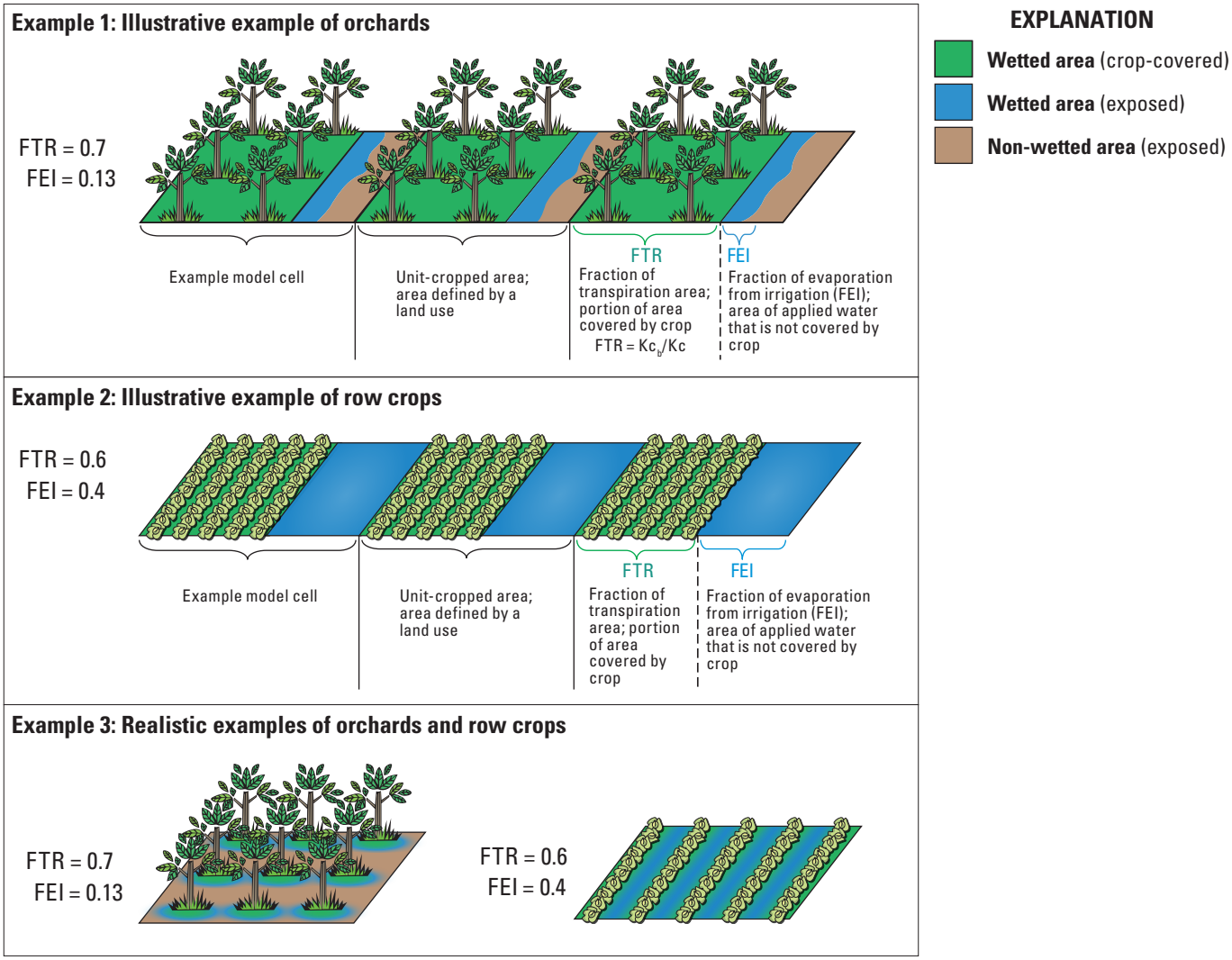


Figure 4.1. Spatial layout of the fraction of transpiration and the fraction of evaporation from irrigation for orchards and row crops (modified from Schmid and others, 2006).

The unit-cropped area is divided by the FTR to account for the crop transpiration of groundwater, precipitation, and applied water (irrigation), and the remaining unit-cropped area (1 – FTR) is assumed to contribute evaporation from groundwater and precipitation.

$$T_{pot} = CU_{ini} \cdot FTR \cdot Area \geq T_{act} \tag{4.3}$$

$$\begin{aligned} E_{pot} &= CU_{ini} \cdot (1 - FTR) \cdot Area + (T_{pot} - T_{act}) \\ &= CU_{ini} \cdot Area - T_{act} \end{aligned} \tag{4.4}$$

where

Area is the unit-cropped area [L^2],
T_{pot} is the potential transpiration in the unit-cropped area [L^3/T],
T_{act} is the actual transpiration in the unit-cropped area [L^3/T], and
E_{pot} is the potential evaporation from groundwater and precipitation in the unit-cropped area [L^3/T].

Determining potential and actual transpiration is discussed in detail in the "Satisfying the Potential Transpiration Component," in the "Evaporation in a Unit-Cropped Area," and in the "Evaporation in a Bare Soil Area" sections later in this appendix.

Precipitation is assumed to be distributed equally throughout the cropped area. The FTR provides a way of dividing rainfall available for potential consumption by the crop from rainfall that falls on the exposed soil (1 − FTR) and is available for potential evaporation. The following defines these relationships:

$$P_{Tpot} = P \cdot FTR \cdot Area \tag{4.5}$$

$$P_{Epot} = P \cdot (1 - FTR) \cdot Area \tag{4.6}$$

where

P is precipitation that falls over the unit-cropped area [L/T];
P_{Tpot} is the potential transpiration from precipitation [L^3/T], if $T_{pot} > P_{Tpot}$, then $P_{Tpot} = T_{pot}$; and
P_{Epot} is the potential evaporation from precipitation [L^3/T], if $E_{pot} > P_{Epot}$, then $P_{Epot} = E_{pot}$.

The potential transpiration and evaporation from precipitation cannot exceed the respective total potentials defined in equation 3 and 4, respectively.

Irrigation water is primarily applied to the transpiratory-cropped area (crop covered, FTR), and some excess is applied to the evaporative-cropped area (exposed soil, 1 − FTR). To account for this, the fraction of evaporation from irrigation (FEI, keyword **EVAPORATION_IRRIGATION_FRACTION**) must be defined (fig. 4.1). FEI is the fraction of the unit-cropped area that is exposed soil and has irrigated water applied to it. FEI values depend on both the style of irrigation and the cultural agricultural practices. For example, drip irrigation has a small FEI, whereas flood irrigation has a large value (such as FEI = 1 − FTR). The limit of FEI is that its sum with FTR cannot be greater than one.

$$FTR + FEI \leq 1 \tag{4.7}$$

where

FEI is the fraction of the unit-cropped area that has irrigated water applied to the exposed soil part of the unit-cropped area (that is, the area not covered by the crop's leaf matter) [-], $0 \leq FEI \leq 1 - FTR$.

It should be noted that irrigation is the amount that can satisfy available potential transpiration and not the potential consumptive use (CU_{ini}). The available potential transpiration is the remaining portion of the potential transpiration after transpiratory consumption from groundwater and precipitation. If an irrigated crop does not consume any groundwater or precipitation (that is, $E_p = T_p = E_{GW} = T_{GW} = 0$), then the maximum quantity of irrigation with perfect irrigation efficiency (OFE = 1) is equal to equation 8.

$$CIR_{MAX} = CU_{ini} \cdot (FTR + FEI) \cdot Area \geq CIR \tag{4.8}$$

where

CIR_{MAX} is maximum quantity of irrigation with perfect irrigation efficiency that can be applied to a crop [L^3/T], and
CIR is the crop irrigation requirement assuming perfect efficiency [L^3/T].

The specific values of K_c and FTR depend on the type of crop grown and its growth stage, whereas FEI depends on irrigation practices. Crop coefficients may vary with time to represent different periods in the crop's growth cycle throughout the year. If there are multiple crop yields within a year, then the annual time series of crop coefficients has a sawtooth shape. Typically, increases in K_c are also correlated with increasing K_{cb}, such that their ratio (FTR, eq. 4.2) stays relatively constant throughout the year. Annual variation in FTR should be much less than the variation in K_c. The FEI tends not to change unless the irrigation practices change or there is a violation of equation 7; that is, if the FTR plus FEI are greater than one, then the FTR and FEI are scaled so the sum equals one.

The largest area that a unit-cropped area can occupy is a single model cell's surface area. By default, crops are simulated on a cell-by-cell basis using each model cell's surface area as the unit-cropped area. If a unit-crop area is less than a model cell's surface area, then the remaining, non-cropped area is designated as "bare soil." If a model cell contains multiple crops (keyword **MULTIPLE_LAND_USE_PER_CELL**), then the "bare soil" area is equal to the model cell's surface area less the sum of the cell's crops' unit-crop areas. For example, if a model cell has a surface area of 100 m^2 and contained two crops that had unit-crop areas of 30 m^2 and 60 m^2, then the bare soil area is 10 m^2. Bare soil area only simulates evaporation from groundwater and precipitation, which is described in the "Evaporation in a Bare Soil Area" section. Bare soil calculations are included in case of incomplete datasets or to represent fallowed land, but it is recommended to fully described all crops such that there is no simulated bare soil. If a simulation contains bare soil (or multiple crops per model cell), then it is required to specify a potential bare-soil evaporation rate or the reference evapotranspiration (ET_{ref}) for FMP to calculate the bare soil evaporation from groundwater and precipitation. If both are specified, then the bare-soil evaporation rate is used. If the reference evapotranspiration is used to determine the potential bare soil evaporation, then it is assumed that the potential bare evaporation is half that of the reference evapotranspiration, which is the same as assuming a $K_c = 0.5$ (Allen and others, 1998). Instead of relying on bare soil calculations, it is recommended to define a "fallow crop" or "bare crop" type that contains a near-zero FTR and associated consumptive use.

Water from precipitation or irrigation not consumed by the crop either becomes surface runoff or infiltrates to groundwater (deep percolation). Excess precipitation is that precipitation not fully evaporated or consumed through transpiration. The irrigation efficiency (OFE) determines the excess irrigation required to meet demand after efficiency losses, which are referred to as inefficient losses. The fraction of inefficient losses from precipitation to surface water (FIESWP) and fraction of inefficient losses from irrigation to surface water (FIESWP) are used to determine the proportion of excess water that becomes surface-water runoff as follows:

$$IRR_{Ex} = IRR - IRR \cdot OFE \tag{4.9}$$

$$Precip_{Ex} = P - T_p - E_p \tag{4.10}$$

$$SW_R = Precip_{Ex} \cdot FIESWP + Irr_{Ex} \cdot FIESWI \tag{4.11}$$

$$DP = Precip_{Ex} \cdot (1 - FIESWP) + Irr_{Ex} \cdot (1 - FIESWI) \tag{4.12}$$

where

IRR_{Ex} is excess irrigation that becomes either runoff or deep percolation [L^3/T];
IRR is the total applied irrigation to a unit-cropped area [L^3/T];
T_p is the quantity of precipitation that is consumed by transpiration [L^3/T];
E_p is the quantity of precipitation that is consumed by evaporation [L^3/T],
$Precip_{Ex}$ is the excess precipitation that becomes either runoff or deep percolation [L^3/T];
OFE is the irrigation efficiency, called on farm efficiency, $0 < OFE \leq 1$ [-];
SW_R is surface-water runoff from excess precipitation and irrigation [L^3/T];
FIESWP is the fraction of inefficient losses from precipitation to surface water [-], $0 \leq FIESWP \leq 1$;
DP is water that infiltrates to groundwater, deep percolation [L^3/T];
Irr_{Ex} is the excess irrigation not consumed by the crop [L^3/T]; and
FIESWI is the fraction of inefficient losses from irrigation to surface water [-], $0 \leq FIESWI \leq 1$.

These two fractions are limited between 0 and 1 because they represent a split in the excess water between surface-water runoff and deep percolation.

For complex cropping, it may be necessary to aggregate multiple crop coefficients to make a composite coefficient that represents multiple crops within a time frame. Figure 4.2 presents the crop coefficient and fraction of transpiration used to describe deciduous orchard trees and raspberries, blackberries, and blueberries (Hanson and others, 2014d).

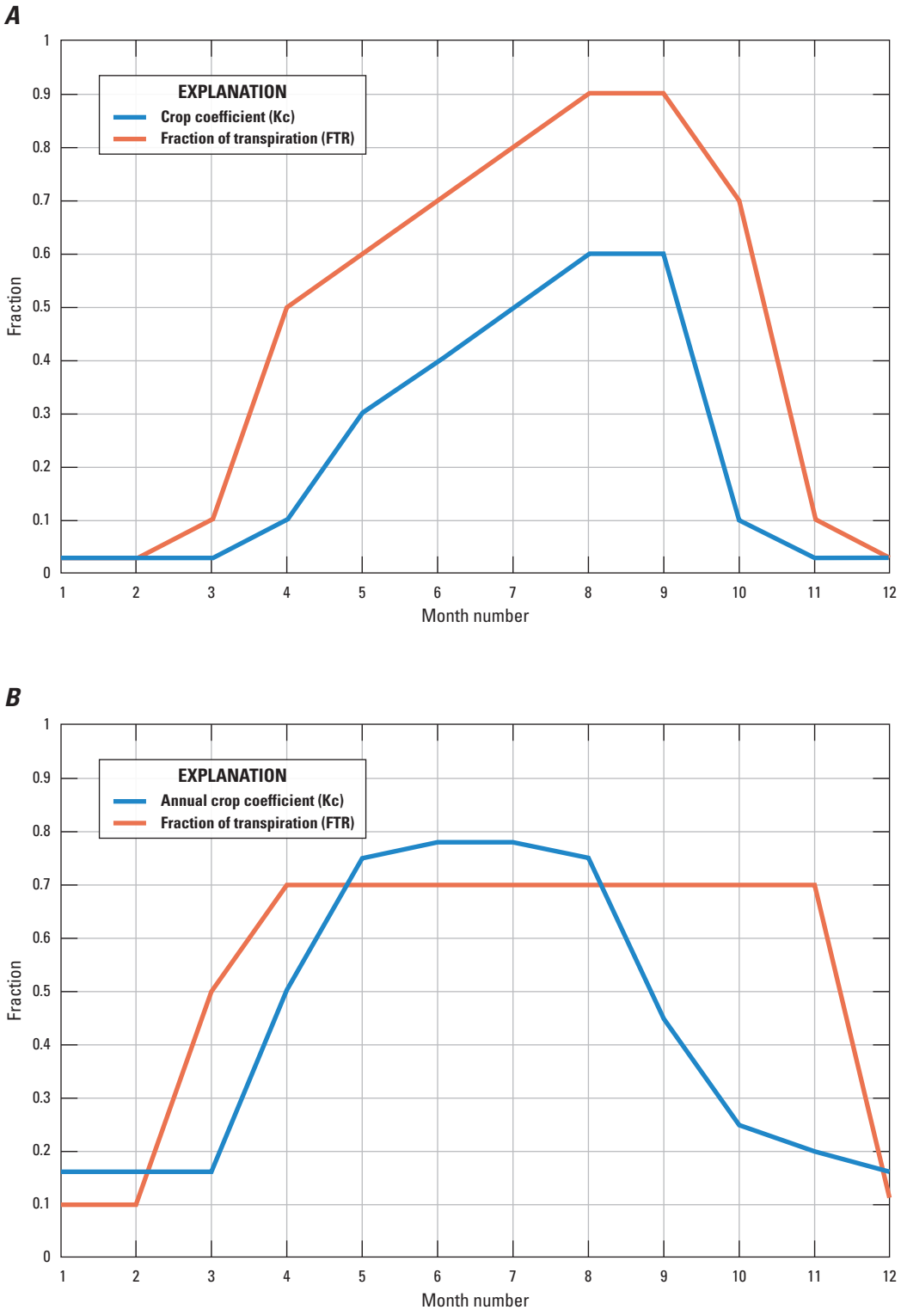


Figure 4.2. Monthly crop coefficients (K_c) and fractions of transpiration (FTR) used to represent the following crop types (modified from Hanson and others, 2014d): *A*, deciduous orchard trees; *B*, raspberries, blackberries, and blueberries.

Capillary Fringe, Root Depth, and Ponding

The FMP (farm process) determines transpiration and evaporation from crops and groundwater based on the overlap between the capillary fringe and crop root depth. The capillary fringe is the zone above the water table that contains water pulled upward by the porous medium's capillary forces (in FMP, this zone is a length added to the top of the water table). This is a property of the soil, and typical values span from 0.5 to 3 meters above the water table. Finer grained sediments typically have larger capillary fringe lengths compared to coarse material. Groundwater is assumed to evaporate when the water-table elevation plus the user-specified capillary fringe length is at or above the land surface. The root depth is a crop property that may change through time, depending on the growth stage of the crop. If the elevation represented by the water table added to the capillary-fringe length is greater than the land surface minus the root depth, then this crop takes up groundwater as part of its consumptive use. If the water-table elevation added to the capillary fringe overlaps too much of the root zone, the soil becomes anoxic (the crop starts to drown). If the water table reaches the land surface, plants may die. Some riparian vegetation and crops, such as rice, can sustain evapotranspiration under flooded conditions.

Three methods are available in FMP to prevent plant death and anoxia. The first is to remove root groundwater uptake and anoxia completely and require full consumptive use from precipitation and applied water. The second is to specify a ponding level for the *linear root response* groundwater uptake and anoxia concept. This alters the slope of the linear anoxia level to cause plant death to coincide with the land-surface elevation added to the ponding depth of the water. The third is to specify positive root zone pressures, PSI, for the *analytical root response* concept to indicate the water-table is above the cropped land surface. For crops that are semiaquatic, disconnected from the groundwater source, or are constantly ponded, it is recommended to remove root groundwater uptake and anoxia completely (by setting **GROUNDWATER_ROOT_INTERACTION** to 1 for the crop). An example crop that this works best for is rice, which grows in flooded areas or paddy fields.

Consumptive-Use Stress Factor

Typically, consumptive-use estimates and crop coefficients are based on idealized, unstressed conditions. MF-OWHM2 has two methods for accounting for stress on a crop. The first method uses the analytical pseudo-steady state soil-moisture, "reduced consumptive use" (*analytical root response*) concept for crops. This was developed as part of the original release of the FMP (Schmid and others, 2006). It requires the user to input a set of values for four stress-response root zone pressures, called PSI values, that indicate pressure ranges over which the crop optimally uptakes groundwater, wilts, or suffers from anoxia. The second input requirement is the soil type in which it grows, which can be one of four types: silt, silty clay, sandy loam, or sand. This feature results in a minor reduction in the consumptive use because the soil is not an ideal matrix. The FMP also may further reduce consumptive use as a result of anoxia (drowning of the roots due to high water table). It is not necessary to define all crops by PSI root pressures, but if one crop is defined with it, then it is required to define the soil type for the entire model. Any crops not defined with PSI values do not have consumptive use reduced because of non-idealized soil, but may have it reduced because of anoxia (note that setting all the PSI values to zero is the same as not specifying them).

Another method to account for stress is to include a climatic scale factor. A common practice for the U.S. Geological Survey MF-OWHM2 models is to use the cumulative departure from the mean (CDM) of precipitation to identify wet and dry periods of record (Faunt and others, 2009, 2015; Hanson and others, 2014a–c). The CDM takes a precipitation record (monthly or annual) and compares the mean precipitation ($\bar{P}$) for the entire record to a cumulative summation over the *m* individual departure records. If the change in the CDM is negative or follows a downward trend, the period is considered dry. Conversely, a positive change or upward trend is considered wet. The formal steps for calculating the CDM are as follows:

$$\begin{aligned} \bar{P} &= \frac{1}{m}\sum_{i=1}^{m} P_i \\ CDM_1 &= \left(P_1 - \bar{P}\right) \\ CDM_i &= \left(P_i - \bar{P}\right) + CDM_{i-1} \qquad \forall i = 2:m \end{aligned} \tag{4.13}$$

An example from the Pajaro Valley that has an annual period of record from 1880 to 2014 is presented in figure 4.3. At the start of the record, the change from the first year, 1880, to the second, 1881, is negative, so 1881 is considered a dry period. Conversely, the tenth data point, 1889, has a positive change to the next year, 1890, so the year of 1890 is considered a wet year.

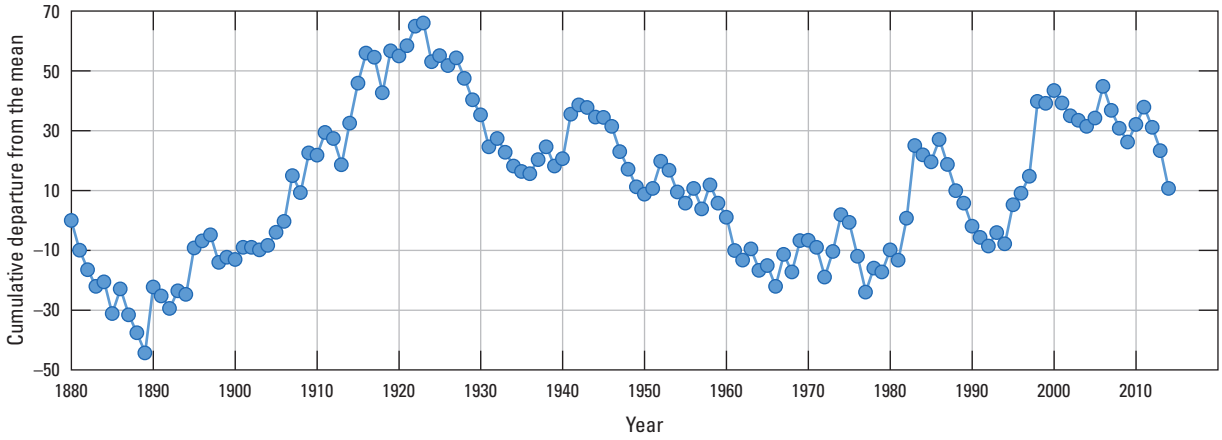


Figure 4.3. Example cumulative departure from the mean from an annual precipitation record for 1880 to 2014 from the integrated hydrologic model of Pajaro Valley (Hanson and others, 2014a).

The scale factors that incorporate climate stress are described by Allen and others (1998, 2005) as an additional stress factor that is used to scale the unstressed crop coefficients. They are commonly broken into a wet-seasonal group and dry-seasonal group of stress factors. This results in eight seasonal wet- and dry-year climatic scale factors: wet-fall, wet-winter, wet-spring, wet-summer, dry-fall, dry-winter, dry-spring, and dry-summer. These eight scale factors are applied to the consumptive use and the crop coefficients. To minimize the number of parameters, it is recommended to use one scale factor that multiplies all the different crop coefficients. For example, if there are NCROPs and the stress period is simulating January during a dry year, then the NCROP crop coefficients are all multiplied by the dry-winter scale factor. Previous publications (Hanson and others, 2014a–d) have scale factors greater than one (increase) for dry years and less than one (decrease) for wet years.

If the crops receive applied or irrigation water (non-zero irrigation flag), then there is an associated irrigation efficiency to account for efficiency losses. Irrigation efficiency is also multiplied by its own set of wet or dry seasonal scale factors. During dry periods, most agricultural practices tend to become more efficient as a result of the scarcity of water, so dry-year scale factors tend to be greater than one. Conversely, during wet periods, a common practice is to specify a lower efficiency, or wet-year scale factors less than one, because water is not scarce.

Satisfying the Potential Transpiration Component

The potential transpiration component (T_{pot}) is the maximum possible rate at which a crop may transpire for its given consumptive use. It is calculated from the fraction of transpiration (FTR), which is assumed to be independent of whether the potential transpiration component is satisfied by root groundwater uptake, precipitation, or applied water. This results in the following formulation:

$$T_{pot} = K_c \cdot ET_{ref} \cdot FTR \cdot Area = CU_{ini} \cdot FTR \cdot Area \geq T_{act} \tag{4.14}$$

The FMP attempts to satisfy the potential transpiration with the consumption of water from root groundwater uptake, precipitation, and applied water (irrigation). Based on the user-specified root-groundwater interaction level, anoxia and root groundwater uptake are calculated. If anoxia is included, it is determined by the water-table elevation and adjusts the potential transpiration as follows:

$$\tilde{T}_{pot} = T_{pot} - Anoxia \tag{4.15}$$

where

$\tilde{T}_{pot}$ is potential transpiration after any anoxia reduction [L^3/T], and

Anoxia is level of anoxia [L^3/T], $0 \leq Anoxia$.

Once the anoxia level is calculated, the root groundwater uptake, T_{uptake}, is determined by the root-groundwater interaction level.

The FMP is capable of simulating five root-groundwater interaction levels, which are outlined in figure 4.4. The different levels determine if a crop root is capable of consuming groundwater (root groundwater uptake = yes), if its consumptive use is reduced from anoxia (anoxia reduction = yes), and if its consumptive use is reduced as a result of soil-water-related stresses (soil stress reduction = yes). The default FMP approach, and the only option in previous releases, is level 5.

The first root-groundwater interaction level is to not have any interaction between the groundwater and the crop roots. With this option, defining the crop-root depth is optional and may be set to zero. The consumptive use is not lowered as a result of anoxia or stress conditions. It may be lowered, however, if there is not enough water supply to satisfy the transpiration demand, resulting in wilting conditions. This method is best for crops that are always disconnected from groundwater or do not suffer from anoxia, such as rice. The next level allows anoxia to reduce transpiration of the crop and consequently the consumptive use, but does not allow any root groundwater uptake, thus requiring all consumption of water to be from precipitation and irrigation. The third level is the opposite of the second, allowing for root groundwater uptake, but not reductions from anoxia or wilting. This may be best for native vegetation areas that do not suffer from anoxia and rely on root groundwater uptake as their main water source. Lastly, the fifth level is full interaction between the groundwater and the crop roots. Using the fifth level, if anoxia exists in the root zone, then the crop's reduced transpiration demand only consumes groundwater by root uptake. In previous releases of FMP, the fifth level was the only option, and with this release, it becomes the default if the root groundwater-interaction flag is not specified. If the water-table elevation plus the capillary fringe is lower than the land surface minus the root depth, then there is no root groundwater interaction, and all water consumption must come from surface sources (precipitation and irrigation).

If there is root-groundwater interaction in the form of groundwater uptake or anoxia, two concepts determine the amount of uptake and anoxia. The first is the *linear root response,* which is described in detail in appendix 5 as an "implicit stress assumption" or "non-reduced consumptive use" concept that relies on a piecewise linear approximation of root groundwater uptake and anoxia. This assumes that there is a linear increase in root groundwater uptake when the water table added to the capillary fringe intersects the crop's roots, and the full potential transpiration is taken up until the water table reaches the bottom of the roots. If the water table intersects the root depth, then the crop's transpiration decreases as a result of anoxia; transpiration progressively decreases as the water-table rise continues until the water table reaches the land surface (plus a ponding depth if specified). Once the water table is at the land surface, the crop is assumed to have drowned and no longer transpires water.

The second is the *analytical root response,* which is described in detail in appendix 5 as an analytical pseudo-steady state soil-moisture, "reduced consumptive use" soil-stress concept or so-called "explicit stress response." First, the user must supply the soil type in which the crop is grown. This can be defined as *silt, silty clay, sandy loam,* or *sand*; these soil types are used to determine any water-stress related reduction in transpiration. The user must also supply four root-zone pressures, PSI, that represent the upper pressure head limit at which the root uptake becomes zero because of anoxia, then two pressure heads that represent the range of optimal root uptake, and then finally a lower pressure head limit that results in wilting or zero root uptake.

Level	Root groundwater uptake	Anoxia reduction	Water-stress reduction
0	—	—	—
1	No	No	No
2	No	Yes	Yes
3	Yes	No	No
4	Yes	No	Yes
5	Yes	Yes	Yes

Figure 4.4. Farm Process (FMP) crop root-groundwater interaction level's effect on the calculation of potential evapotranspiration (ET_c). Level zero sets the crop's potential evapotranspiration to zero. [— indicates the crop evapotranspiration process is not simulated, so it is neither yes or no. Root groundwater uptake is the process that a crop's roots can consume water directly from saturated groundwater. Anoxia reduction can reduce ET_c because the water table overlaps with the crop's roots. Water-stress reduction can reduce ET_c resulting from non-idealized soil effects.]

Using an iterative, empirical formulation, the soil-water pressure in the root zone is calculated according to the soil type, PSI root pressures, and the water-table elevation plus capillary fringe (see appendix 5). This root-zone pressure then determines water-stress and anoxia transpiration reduction and root groundwater uptake.

After the level of anoxia and transpiratory direct uptake from groundwater, $T_{uptake,}$ is determined, any remaining transpiration demand is supplied by surface supplies (precipitation and irrigation). The following is the demand for surface-supplied transpiration:

$$T_{surf} = \tilde{T}_{pot} - T_{uptake} \tag{4.16}$$

where

T_{surf} is potential transpiration demand to consume precipitation and irrigation [L^3/T].

The demand for surface-supplied transpiration first uses any available precipitation for transpiration, P_{Tpot}, to yield the final contribution of precipitation to transpiration, T_p. If any unmet transpiration demand remains, then it must be satisfied with irrigation water.

$$T_{CIR} = \tilde{T}_{pot} - T_{uptake} - T_p \geq T_{irrigation} \tag{4.17}$$

where

T_{CIR} is the potential consumption of irrigation water necessary to fully satisfy transpiration [L^3/T], and

$T_{irrigation}$ is the actual consumption of irrigation water as transpiration, which may be limited by available irrigation supplies [L^3/T].

If the crop is not irrigated ($T_{irrigation} = 0$), then its transpiration is reduced as a result of wilting conditions; consequently, its consumptive use is reduced. If the crop is irrigated, then FMP attempts to fulfil T_{CIR} using imported water, surface water, and groundwater pumping.

If the irrigation supplies are enough to fulfill the potential transpiration demand from irrigation, then $T_{irrigation} = T_{CIR}$. The total irrigation water required to satisfy the potential transpiration (T_{CIR}) and evaporation demand (E_{CIR}) for irrigation is the crop irrigation requirement (CIR). FMP assumes that evaporation from irrigation varies linearly with the transpiration from irrigation. With this assumption, the total evapotranspiration from irrigation and CIR can be calculated as follows:

$$\frac{E_{CIR}}{T_{CIR}} = \frac{FEI}{FTR} = \frac{E_{irrigation}}{T_{irrigation}} \tag{4.18}$$

$$CIR = ET_{CIR} = \left(FEI / FTR\right) \cdot T_{CIR} + T_{CIR} \tag{4.19}$$

$$D_{irrigation} = CIR / OFE \tag{4.20}$$

where

T_{CIR} the potential consumption of irrigation water necessary to fully satisfy transpiration [L^3/T];

E_{CIR} is the potential consumption of irrigation water necessary to fully satisfy evaporation [L^3/T];

$E_{irrigation}$ is the actual consumption of irrigation water as evaporation, which may be limited by T_{CIR} [L^3/T];

ET_{CIR} is potential consumption of irrigation water necessary to fully satisfy evapotranspiration [L^3/T];

CIR is the crop irrigation requirement assuming perfect efficiency [L^3/T];

OFE is the irrigation efficiency, called on-farm efficiency, $0 < OFE \leq 1$ [-]; and

$D_{irrigation}$ is the irrigation demand to satisfy the potential transpiration, $\tilde{T}_{pot}$ [L^3/T].

To meet the irrigation demand ($D_{irrigation}$, eq. 4.20), imported water is applied first, then surface water, then any remaining irrigation required is supplied by groundwater pumping up to the maximum capacity of the wells.

If groundwater pumping is not enough to satisfy the $D_{irrigation}$, there are two potential scenarios. With the first, "deficit irrigation," the crop suffers from wilting from a lack of water and its actual transpiration is decreased to meet the water supplies. To calculate the actual transpiration from irrigation under a deficit irrigation scenario, equations 19 and 20 can be reformulated as follows:

$$T_{irrigation} = \frac{IRR \cdot OFE}{1 + FEI / FTR} \tag{4.21}$$

where

IRR is the total applied irrigation to a unit-cropped area [L^3/T].

The second scenario is called the "external water scenario." Under this situation, it is assumed that the crop receives additional water from unknown sources to meet the demanded irrigation, $D_{irrigation}$. If the external water is applied, then the transpiration from irrigation is fully met ($T_{irrigation} = T_{CIR}$).

The final transpiration is the sum of its three transpiration components:

$$T_{act} = T_{uptake} + T_p + T_{irrigation} \tag{4.22}$$

Evaporation in a Unit-Cropped Area

The total evaporation from a unit-cropped area is subdivided into three sources: irrigation, precipitation, and groundwater. FMP first calculates the evaporation from irrigation ($E_{irrigation}$), then evaporation from precipitation (E_p), then evaporation from groundwater (E_{GW}).

Evaporation from Irrigation

The evaporation from irrigation depends on the amount of water applied to the crop and is correlated to the transpiration from irrigation. Conversely, precipitation and groundwater evaporation have a maximum rate equal to the potential evaporation rate from groundwater and precipitation, E_{pot}, which is determined by the fraction of transpiration (FTR) with equation 4.

The evaporation from irrigation is assumed to vary linearly with transpiration from irrigation and follows the relationship defined in equation 18. This relationship necessitates first evaluating the transpiration demanded from irrigation and the corresponding amount of applied water. For a given amount of applied water, irrigation efficiency, and fractions of transpiration and evaporation supplied by irrigation, the final evaporation loss from irrigation water is calculated as follows:

$$E_{irrigation} = \left(FEI / FTR\right) \cdot T_{irrigation} \tag{4.23}$$

Evaporation from Precipitation

After the evaporation from irrigation is established, then evaporation from precipitation is calculated. By default, FMP assumes that evaporative consumption of precipitation (E_p) is equal to the calculated potential evaporation rate such that $E_p = P_{Epot}$ (eq. 4.6). This assumption works well for arid and semi-arid simulation domains, but can lead to an overconsumption of precipitation if there is frequent high intensity, short duration rainfall a simulated time step. To prevent overconsumption, FMP supports specifying an upper limit for consumption of precipitation, which is discussed in the "Limiting Precipitation Consumption" section.

Evaporation from Groundwater

This changes the potential evaporation for groundwater as follows:

$$E_{GWpot} = E_{pot} - E_p - E_{irrigation} \tag{4.24}$$

where

E_{GWpot} is the potential evaporation from groundwater [L^3/T].

Groundwater only evaporates when the water-table elevation added to the user-specified capillary fringe depth is above the land-surface elevation. When this occurs, there is a linear increase in evaporation from groundwater as the water table rises to reach its maximum, E_{GWpot}, when the water table is above the land surface.

The actual evaporation over the total cropped area (crop covered area and exposed area) is then the sum of the three evaporative components:

$$E_{act} = E_p + E_{GW} + E_{irrigation} \tag{4.25}$$

where

E_{act} is the actual evaporation over the unit-cropped area [L^3/T],
E_p is the quantity of precipitation that is consumed by evaporation [L^3/T], and
E_{GW} is the amount of evaporation from groundwater [L^3/T].

Evaporation in a Bare Soil Area

The bare-soil evaporation represents any area that is not specified with a crop (undefined area). Bare soil is not the same as "exposed soil," which is the area not covered by leaf matter (1 − FTR). Bare soil does not include transpiration (FTR = 0) and follows the same calculation process described the "Evaporation in a Unit-Cropped Area" section (E_{pot}), except the potential evaporation (E_{Bpot}) is different and there is no irrigated evaporative component. If there are bare soil cells in the model, then the user is required to specify either a potential bare-soil evaporation rate (E_{Bpot}) or the reference evapotranspiration rate (ET_{ref}), which represents the upper limit of bare soil evaporation. If both rates are specified, then the bare-soil evaporation rate is used. As with satisfying evaporation from crops, precipitation evaporates first, and its rate is calculated as follows:

$$P_{BEpot} = P_{Bare} \cdot BareArea \tag{4.26}$$

where

P_{BEpot} is the potential evaporation of precipitation [L^3/T],
BareArea is the bare-soil area [L^2], and
P_{Bare} is precipitation that falls over the BareArea [L/T].

This allows for the potential evaporation from groundwater to be this:

$$E_{Bgw\_pot} = E_{Bpot} - P_{BEpot} \tag{4.27}$$

where

E_{Bgw_pot} is the potential evaporation from groundwater [L^3/T], and
E_{Bpot} is the potential evaporation from bare-soil evaporation or reference evapotranspiration [L^3/T].

Groundwater under bare soil only evaporates when the water-table elevation plus the user-specified capillary fringe depth is greater than the land-surface elevation. When this occurs, there is a linear increase in evaporation from groundwater under bare soil, E_{Bgw}, as the water table rises, and it reaches its maximum, E_{Bgw_pot}, when the water table is above the land surface.

Limiting Precipitation Consumption

In FMP, precipitation that falls over a landscape is either consumed as evapotranspiration (as part of CU_{final}), flows away over the land surface as runoff, or percolates below the root zone of the Crop (deep percolation). In FMP, effective precipitation (P_{eff}) is the portion of precipitation that is potentially consumable by the crop. The FMP assumes that the difference between precipitation and effective precipitation ($P - P_{eff}$) represents the quantity of water that always becomes runoff. It should be noted that if a Crop does not consume all of P_{eff}, then the unconsumed portion ($P_{eff} - CU_{final}$, given that P_{eff} is greater than CU_{final}) becomes deep percolation and additional runoff. Factors that influence P_{eff} include the climate, soil texture, soil structure, and the depth of the root zone.

By default, FMP assumes that the potential consumption of precipitation is limited to a Crop's consumptive use less any groundwater consumption, which is analogous to $P = P_{eff}$. This assumption works well for arid and semi-arid simulation domains. If a simulation's stress period contains frequent, high-intensity, short duration rainfall, then the equivalent stress period precipitation can result in an overconsumption of precipitation. To prevent overconsumption, the FMP allows specifying effective precipitation, which serves as an upper limit for consumption of precipitation. If effective precipitation is specified as part of the simulation, then FMP replaces equations 6, 7, and 11 with equations 28, 29, and 30.

$$\begin{aligned} &\text{if } P > P_{eff} \\ &\qquad P_{Tpot} = P_{eff} \cdot FTR \cdot Area \\ &\text{else} \\ &\qquad P_{Tpot} = P \cdot FTR \cdot Area \end{aligned} \tag{4.28}$$

$$\begin{aligned} &\text{if } P > P_{eff} \\ &\qquad P_{Epot} = P_{eff} \cdot (1 - FTR) \cdot Area \\ &\text{else} \\ &\qquad P_{Epot} = P \cdot (1 - FTR) \cdot Area \end{aligned} \tag{4.29}$$

$$\begin{aligned} &\text{if } P > P_{eff} \\ &\qquad SW_R = Precip_{Ex} \cdot FIESWP + Irr_{Ex} \cdot FIESWI + (P - P_{eff}) \\ &\text{else} \\ &\qquad SW_R = Precip_{Ex} \cdot FIESWP + Irr_{Ex} \cdot FIESWI \end{aligned} \tag{4.30}$$

where

P_{eff} is effective precipitation in the unit-cropped area [L/T].

Final Consumptive Use

The final consumptive use is the summation of the transpiration from the crop, evaporation from the crop, and evaporation from bare soil. Typically, a model cell has been defined as to crop or is simulated as bare soil. This results in a total summation as follows:

$$\begin{aligned} CU_{final} &= T_{uptake} + T_p + T_{irrigation} + E_p + E_{GW} + E_{irrigation} \\ CU_{Bfinal} &= E_{Bp} + E_{Bgw} \end{aligned} \tag{4.31}$$

where

CU_{final} is the final consumptive use for the unit-cropped area [L^3/T];
CU_{Bfinal} is the final consumptive use for the BareArea (L^3/T);
T_{uptake} is the transpiration satisfied from root groundwater uptake [L^3/T];
T_p is the quantity of precipitation that is consumed by transpiration [L^3/T];
$T_{irrigation}$ is the transpiration satisfied from irrigation [L^3/T];
E_p is the quantity of precipitation that is consumed by evaporation [L^3/T];
E_{GW} is the evaporation from groundwater [L^3/T];
$E_{irrigation}$ is the evaporation from applied water (irrigation) [L^3/T];
E_{Bp} is the evaporation from precipitation on bare soil, which is always equal to E_{Bpot} [L^3/T]; and
E_{Bgw} is the evaporation from groundwater for bare soil [L^3/T].

The final consumptive use is in units of volume per time, even though the initial input was in units of length per time. To express the final consumptive use in units of length, it must be divided by the cropped area (Area) or bare-soil area (BareArea).

Flow Chart of Evapotranspiration Calculation and Data Requirement Options

The diagrams that follow provide an overview of the final consumptive-use calculation and the data required. The consumptive-use estimation assumes that a crop may be irrigated, so irrigation options are included as required input. Figure 4.5 shows that the workflow path of consumptive-use estimation for an individual crop requires specification of a sequence of attributes that are conditioned by the set of options specified by the user, starting with the consumptive-use concept and then branching through the respective options for the two concepts.

The other component of consumption is bare-soil evaporation. Figure 4.6 shows the estimation sequence for the evaporation component (E) of evapotranspiration for the remainder of the area in a model cell (exposed soil) that is not contributing to consumptive use through the canopy and transpiration.

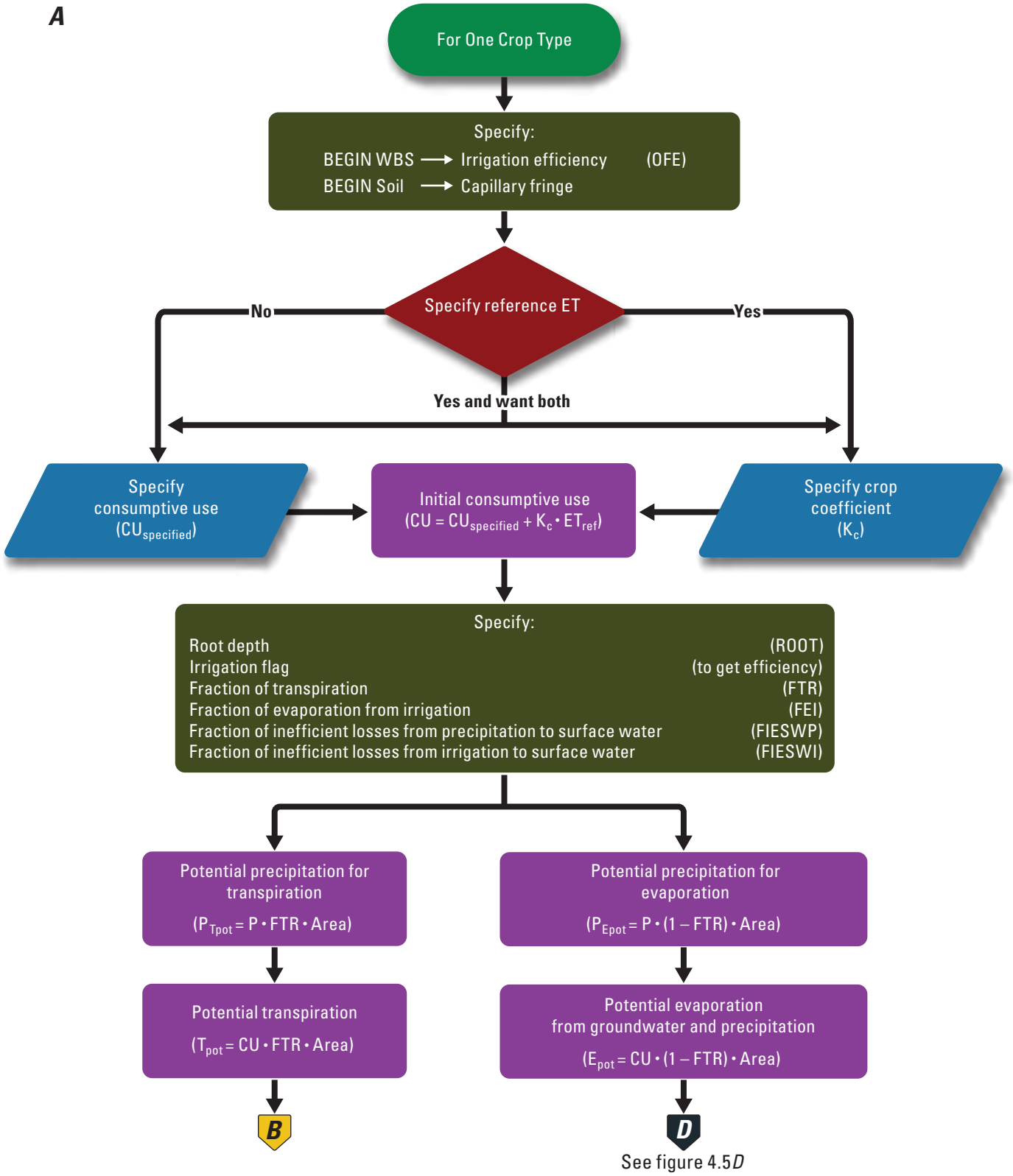


Figure 4.5. The FMP (farm process) evapotranspiration calculation and data requirements.

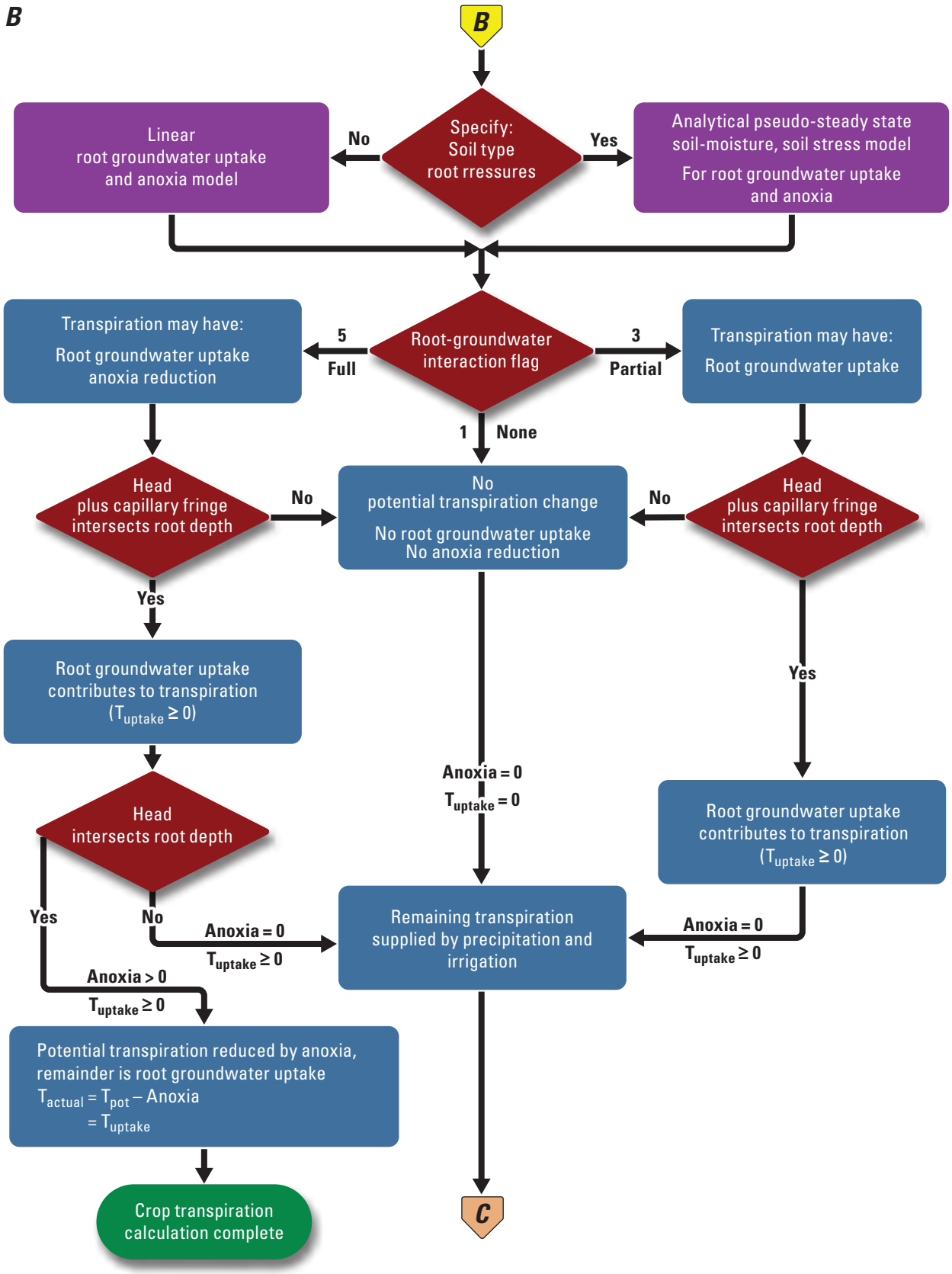


Figure 4.5. —Continued

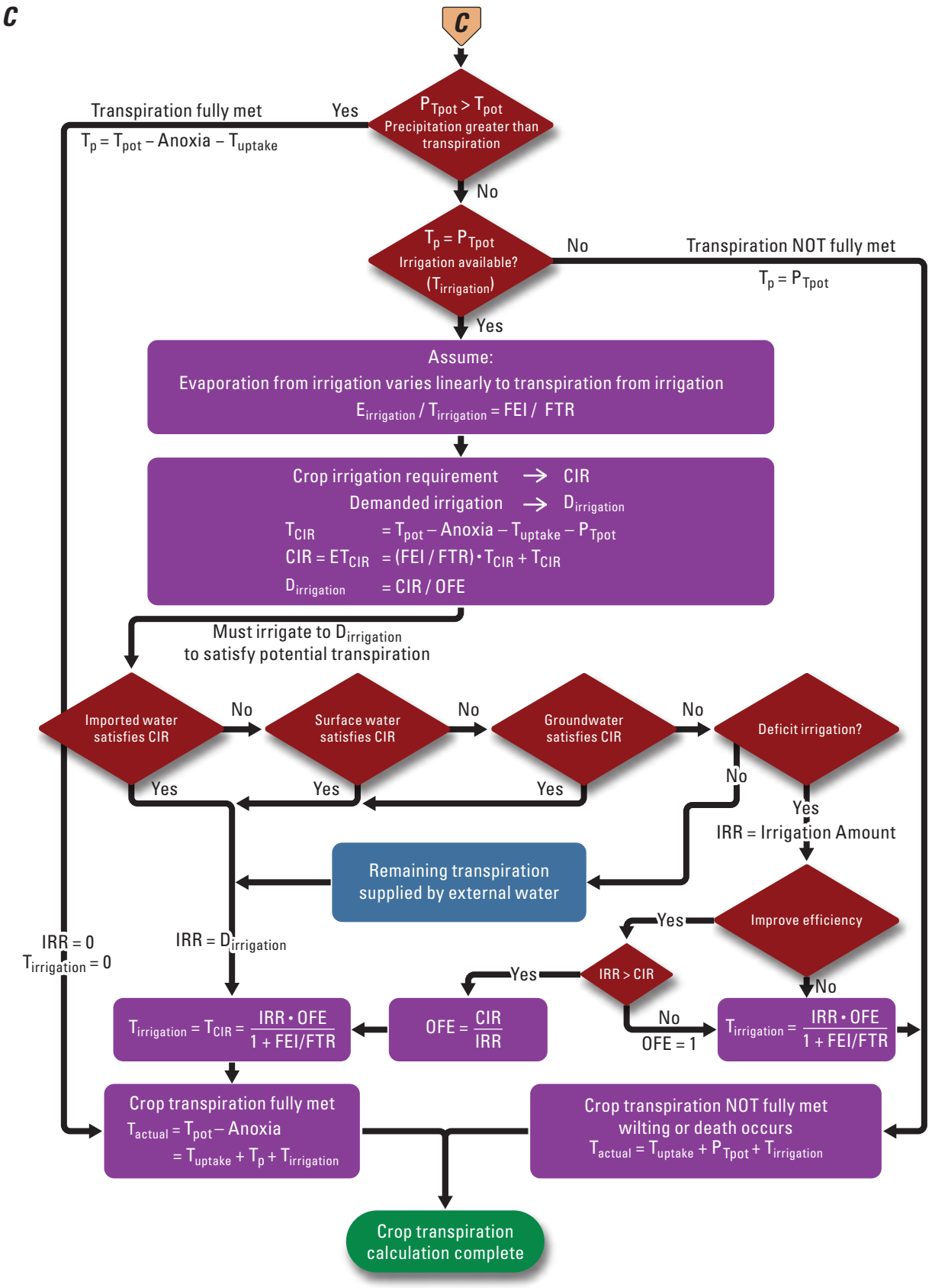


Figure 4.5. —Continued

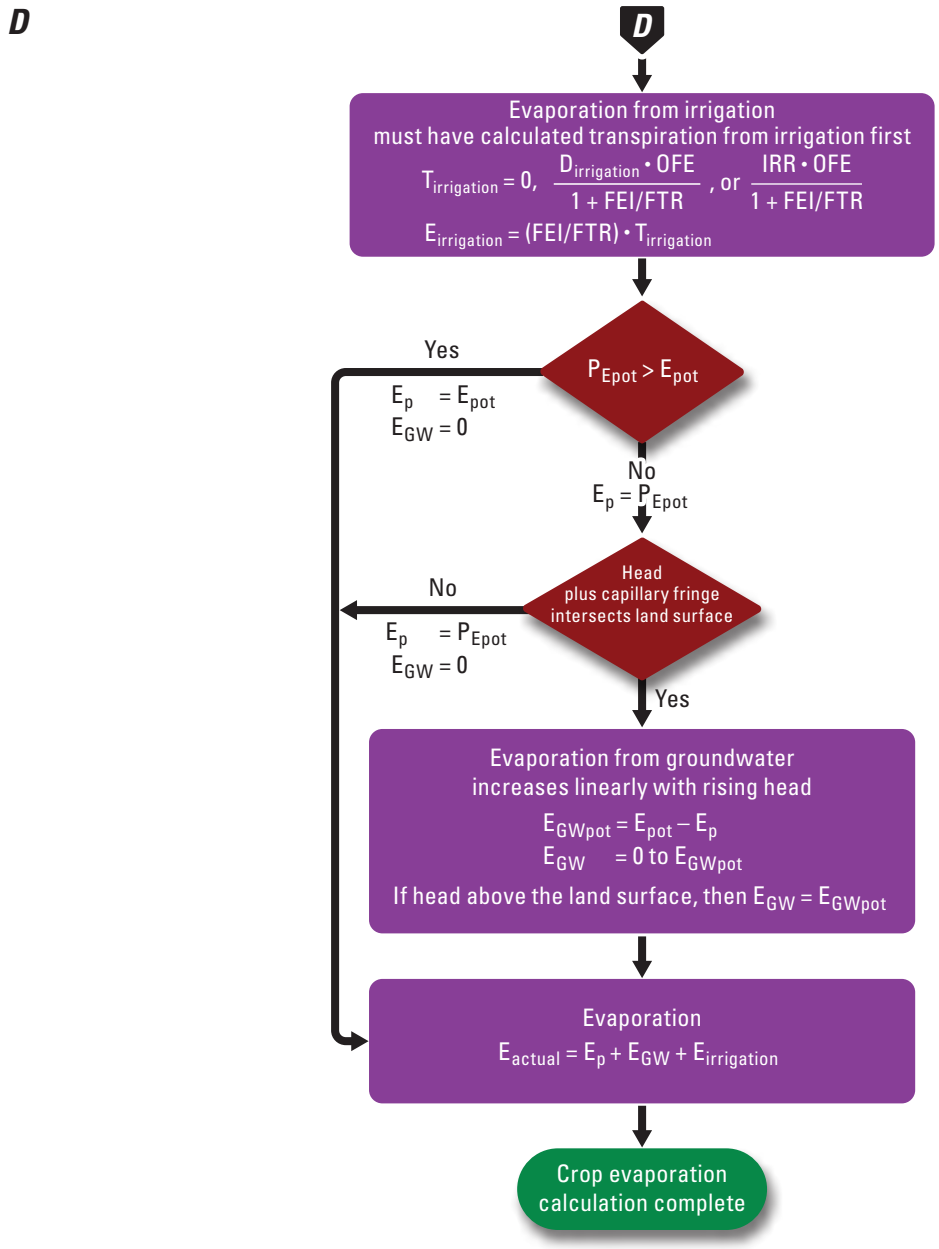


Figure 4.5. —Continued

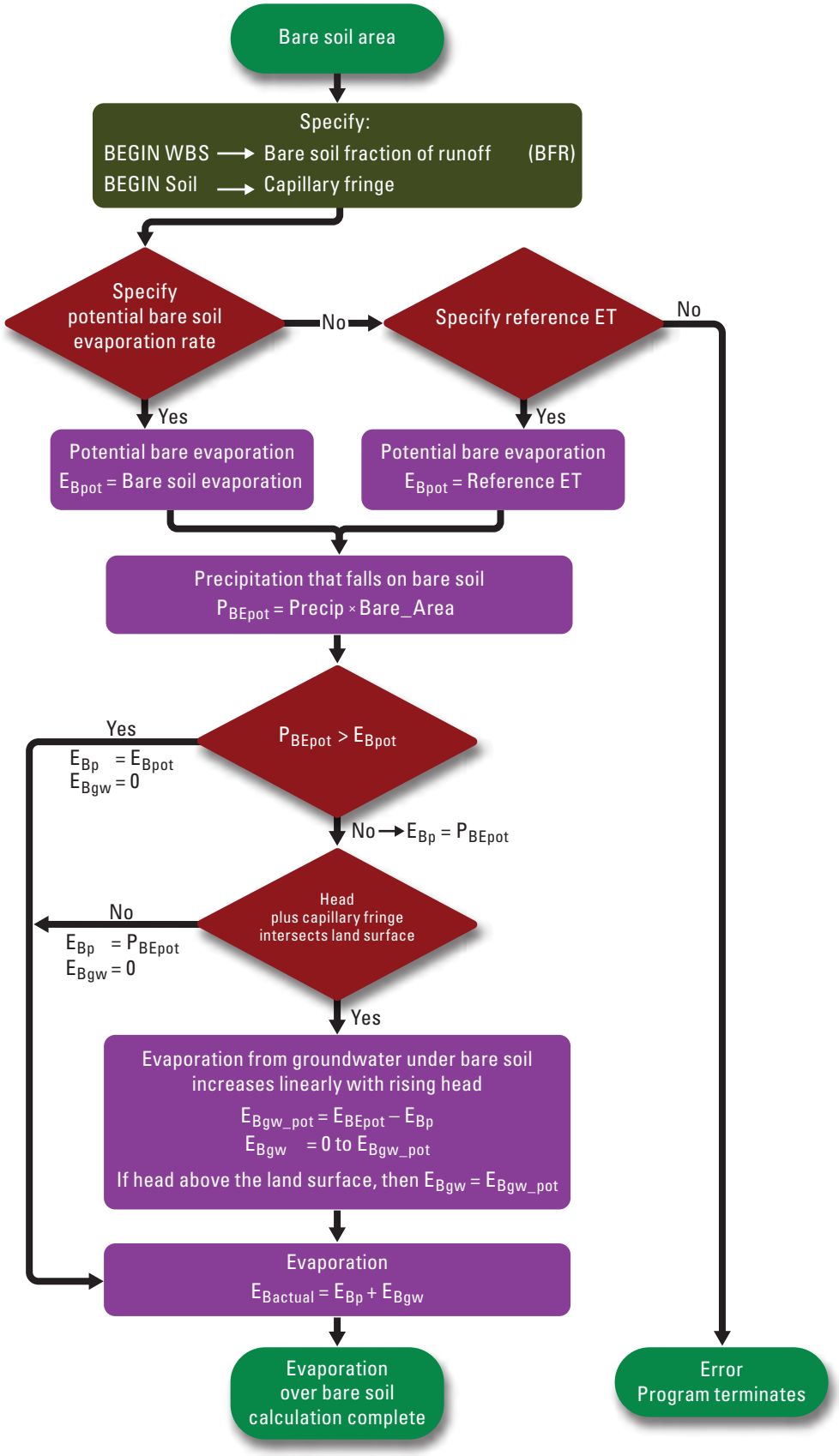


Figure 4.6. Overview of bare-soil evaporation component calculation for consumptive use. [Bare Soil represents any area that is not defined by an FMP land use (Crop).]

References Cited

Allen, R.G., Pereira, L.S., Raes, D., and Smith, M., 1998, Crop evapotranspiration—Guidelines for computing crop water requirements: Rome, Italy, Food and Agriculture Organization of the United Nations, Irrigation and Drainage Paper 56, 300 p., http://www.fao.org/docrep/X0490E/X0490E00.htm.

Allen, R.G., Clemmens, A.J., Burt, C.M., Solomon, K., and O'Halloran, T., 2005, Prediction accuracy for projectwide evapotranspiration using crop coefficients and reference evapotranspiration: American Society of Civil Engineers, Journal of Irrigation and Drainage Engineering, v. 131, no. 1, p. 24–36, https://ascelibrary.org/doi/abs/10.1061/%28ASCE%290733-9437%282005%29131%3A1%2824%29.

Faunt, C.C., Hanson, R.T., Belitz, K., and Rogers, L., 2009, California's Central Valley groundwater study—A powerful new tool to assess water resources in California's Central Valley: U.S. Geological Survey Fact Sheet 2009–3057, 4 p., http://pubs.usgs.gov/fs/2009/3057/.

Faunt, C.C., Stamos, C.L., Flint, L.E., Wright, M.T., Burgess, M.K., Sneed, M., Brandt, J., Coes, A.L., and Martin, P., 2015, Hydrogeology, hydrologic effects of development, and simulation of groundwater flow in the Borrego Valley, San Diego County, California: U.S. Geological Survey Scientific Investigations Report 2015–5150, 154 p., https://pubs.er.usgs.gov/publication/sir20155150.

Flint, L.E., and Flint, A.L., 2014, California basin characterization model—A dataset of historical and future hydrologic response to climate change: U.S. Geological Survey Data Release, https://doi.org/10.5066/F76T0JPB.

Hanson, R.T., Lockwood, B., and Schmid, W., 2014b, Analysis of projected water availability with current basin management plan, Pajaro Valley, California: Journal of Hydrology, v. 519, p. 131–147, https://ca.water.usgs.gov/pubs/2014/HansonLockwoodSchmid2014.pdf.

Hanson, R.T., Schmid, W., Faunt, C.C., Lear, J., and Lockwood, B., 2014a, Integrated hydrologic model of Pajaro Valley, Santa Cruz and Monterey Counties, California: U.S. Geological Survey Scientific Investigations Report 2014–5111, 166 p., https://doi.org/10.3133/sir20145111.

Hanson, R.T., Flint, L.E., Faunt, C.C., Gibbs, D.R., and Schmid, W., 2014c, Hydrologic models and analysis of water availability in Cuyama Valley, California: U.S. Geological Survey Scientific Investigations Report 2014–5150, 150 p., https://pubs.usgs.gov/sir/2014/5150/pdf/sir2014-5150.pdf.

Hanson, R.T., Boyce, S.E., Schmid, W., Hughes, J.D., Mehl, S.M., Leake, S.A., Maddock, T., III, and Niswonger, R.G., 2014d, One-water hydrologic flow model (MODFLOW-OWHM): U.S. Geological Survey Techniques and Methods 6–A51, 120 p., https://doi.org/10.3133/tm6A51.

Hargreaves, G.H., and Allen, R.G., 2003, History and evaluation of the Hargreaves evapotranspiration equation: Journal of Irrigation and Drainage Engineering, v. 129, no. 1, p. 53–63, https://doi.org/10.1061/(ASCE)0733-9437(2003)129:1(53).

Hargreaves, G.H., and Samani, Z.A., 1982, Estimating potential evapotranspiration: Journal of the Irrigation and Drainage Division, American Society of Civil Engineers, v. 108, no. 3, p. 225–230.

Hargreaves, G.H., and Samani, Z.A., 1985, Reference crop evapotranspiration from temperature: Applied Engineering in Agriculture, v. 1, no. 2, p. 96–99.

Samani, Z., 2000, Estimating solar radiation and evapotranspiration using minimum climatological data: Journal of Irrigation and Drainage Engineering, v. 126, no. 4, p. 265–267, https://doi.org/10.1061/(ASCE)0733-9437(2000)126:4(265).

Schmid, W., Hanson, R.T., and Maddock, T., III, 2006, Overview and advancements of the farm process for MODFLOW-2000: MODFLOW and More—Managing Ground-Water Systems Conference: Golden Colorado, Colorado School of Mines, International Ground-Water Modeling Center, May 21–24, 2006, oral presentation.

Snyder, R.L., and Eching, S., 2002, Penman-Monteith daily (24-hour) reference evapotranspiration equations for estimating ET_o, ET_r and HS ET_o with daily data, 5 p., http://biomet.ucdavis.edu/Evapotranspiration/PMdayXLS/PMdayDoc.pdf.

Appendix 5. Landscape and Root-Zone Processes and Water Demand and Supply

This appendix provides a conceptual overview of consumptive use and how it is calculated. The concepts are presented first, with few equations. Then, the description of the mathematics used in the farm process (FMP) and how each of its components is calculated are presented. For the readers who are interested in the mathematics of the concepts, the equation numbers have been provided in the concept sections.

Concepts of Landscape and Root-Zone Processes

The landscape and root-zone processes in the farm process (FMP) package consider two types of water budgeting for the control volume, which is defined by the horizontally delineated land-surface areas and the vadose zone that extends to the water table (fig. 5.1). Initially, in the MF-FMP, these areas were called Farms (Schmid and others, 2006a; Schmid and Hanson, 2009a, 2009b), but subsequent applications of the farm process have advanced this definition to Water-Balance Subregions (WBS) that include regions other than agricultural farms (Hanson and others, 2014a). These WBS can include irrigated and non-irrigated farms, natural vegetation, and urban areas. In some subregions, consumptive use can be zero or non-vegetative. Examples of non-vegetative use include percolation requirements for managed aquifer recharge (MAR) systems or urban demand. Three types of budgeting are associated with these WBS:

- Mass balance between physical inflow and outflow components to and from the control volume (FMP output block keyword FARM_BUDGET to produce FB_DETAILS.out).
- Operational balance between the irrigation water demand and the water supply from different surface or groundwater components to meet this demand (FMP output block keyword FARM_DEMAND_SUPPLY_SUMMARY to produce FDS.out).
- Detailed information on land-use water supply, demand, consumption, runoff, and deep percolation (FMP crop block keyword PRINT BYFARM_BYCROP).

In the real world, the physical water balance is always achieved (that is, mass is not created or lost), but the "economic balance" between supply and demand may not be maintained. For instance, farmers may apply more water than the true crop irrigation requirements, an unforeseen drought may limit irrigation, or non-irrigated lands that depend solely on precipitation may not get enough water in drought seasons and get too much water in wet seasons.

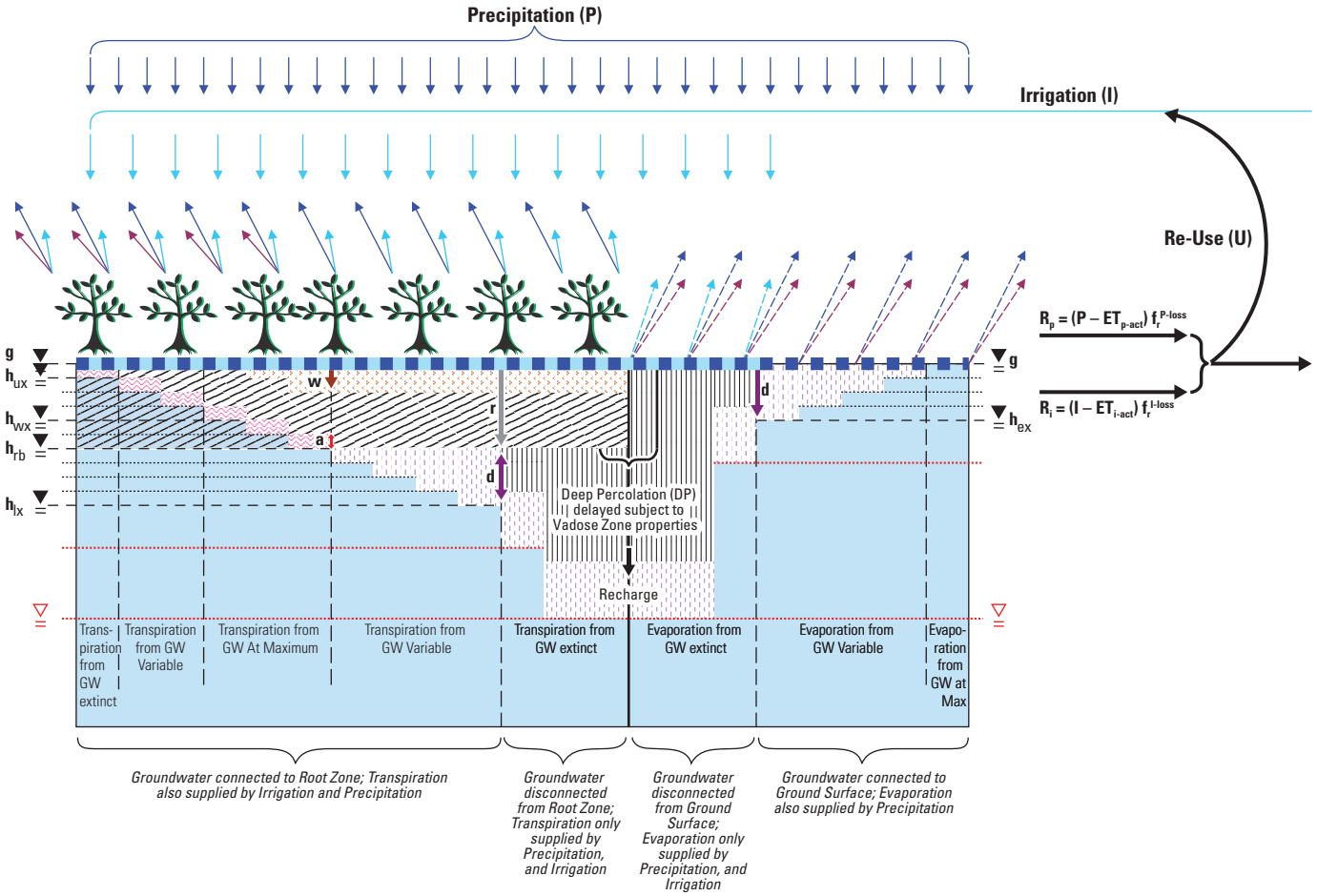
A simplified conceptual schematic of the root zone and landscape processes simulated in FMP is shown in figure 5.1. The general mass-balance equation for the root zone with changes in soil water from time step to time step is given in equation 5.1:

$$P^{t+1} + I^{t+1} - ET_{gw-act}^{t+1} - ET_{c-act}^{t+1} - R^{t+1} - DP^{t+1} = \frac{\theta^{t+1} - \theta^{t}}{\Delta t} \qquad (5.1)$$

$$\text{where } R^{t+1} = R_p^{t+1} + R_i^{t+1}$$

where

P	is precipitation [LT^{-1}],
I	is irrigation water [LT^{-1}],
ET_{gw-act}	is groundwater uptake by roots [LT^{-1}],
ET_{c-act}	is the total actual crop evapotranspiration [LT^{-1}],
R	is the runoff from precipitation and irrigation [LT^{-1}],
R_p	is the surface runoff from precipitation [LT^{-1}],
R_i	is the surface return flow of irrigation water [LT^{-1}],
DP	is the deep percolation that leaves the root zone as the moisture moves downward [LT^{-1}],
θ^{t+1}	is the soil moisture at the end of a time step [L^3/L^2],
θ^{t}	is the soil moisture at the beginning of a time step [L^3/L^2],
Δt	is the time-step length [T], and
t	is the time-step index (-).



EXPLANATION

	P	Precipitation
	I	Irrigation
	U	Re-use of irrigation water
	DP	Deep percolation
	R_p	Returnflow related to precipitation
	R_i	Returnflow related to irrigation
	$f_r^{P\text{-}loss}$	Fraction of inefficient losses to surface-water runoff related to precipitation
	$f_r^{I\text{-}loss}$	Fraction of inefficient losses to surface-water runoff related to irrigation
$T_{gw\text{-}act}$	$T_{gw\text{-}act}$	Portion of actual transpiration fed by groundwater*
$T_{p\text{-}act}$	$T_{p\text{-}act}$	Portion of actual transpiration fed by precipitation*
$T_{i\text{-}act}$	$T_{i\text{-}act}$	Portion of actual transpiration fed by irrigation*
$E_{gw\text{-}act}$	$E_{gw\text{-}act}$	Portion of actual evaporation fed by groundwater*
$E_{p\text{-}act}$	$E_{p\text{-}act}$	Portion of actual evaporation fed by precipitation*
$E_{i\text{-}act}$	$E_{i\text{-}act}$	Portion of actual evaporation fed by irrigation*
	$ET_{p\text{-}act}$	Portion of actual evapotranspiration fed by precipitation*
	$ET_{i\text{-}act}$	Portion of actual evapotranspiration fed by irrigation*
	g	Ground-surface elevation (L)
	h	Groundwater head elevation (L)
	h_{rb}	Head elevation at which head is at the bottom of the root zone (L)
	h_{ux}	Head elevation at which top of anoxia fringe reaches ground-surface elevation with rising head (elevation of upper transpiration extinction) (L)
	h_{wx}	Head elevation at which bottom of the wilting zone reaches ground-surface elevation with rising head (elevation of wilting zone extinction) (L)
	h_{lx}	Head elevation at which top of capillary fringe is at bottom of root zone, hrb (elevation of lower transpiration extinction) (L)

(* assuming no changes in soil-water storage for the length of the time-steps)

a — Depth of anoxia zone (L)
w — Depth of wilting zone (L)
r — Depth of root zone (L)
d — Depth of capillary fringe (L)
Vadose Zone
Groundwater
Canopy receiving irrigation and precipitation
Exposed and wetted by irrigation and precipitation
Exposed and wetted only by precipitation
Head elevations used in equations 8, 9, 10, and 14
Groundwater head disconnected from Root Zone or Ground Surface

Figure 5.1. Schematic representation of root zone and land-use processes simulated by FMP (modified from Dogrul and others, 2011).

The mass-balance equation used for the root zone within a time step in the FMP assumes steady-state soil water and includes iteratively changing flow terms dependent on the groundwater head (h) of the previous iteration:

$$P^{t+1} + I^{k+1}\left(h^{k}\right) + ET_{gw-act}^{k+1}\left(h^{k}\right) - ET_{c-act}^{k+1}\left(h^{k}\right) - R^{k+1}\left(h^{k}\right) - DP^{k+1}\left(h^{k}\right) = 0 \quad (5.2)$$

The current version of the FMP does not consider changes in soil-water storage in the root zone (that is, right hand side of equation 5.2 = 0). The FMP does simulate changes in storage in the deeper vadose zone below the root zone through the optional linkage to the unsaturated-zone flow package (UZF; Niswonger and others, 2006) by treating deep percolation out of the root zone as quasi-infiltration to the deeper vadose zone.

In the FMP, equation 1 is solved for each cell at each iteration (eq. 5.2). Because many of the terms depend directly or indirectly on h, $ET_{gw\text{-}act}$ and $ET_{c\text{-}act}$ vary with groundwater head where the water table is shallow enough for evaporation or transpiration to cause losses of groundwater.

The sections that follow describe how the ET terms in equation 5.2 are computed. Many of these terms depend on the groundwater head and other previously calculated ET flow terms. Therefore, the sequence of the description of these terms is aligned with the order of terms in equation 5.1 and is arranged accordingly in the sequence of calculation. The equations are summarized in a the "Mathematical Representation of Consumptive-Use Components" section. For simplicity, indices for time step (t) and iteration (k) are omitted in the expressions that follow. Variable names relative to those in previous user guides (Schmid and others, 2006a; Schmid and Hanson, 2009b) have been simplified for use in this document.

Applied irrigation (I) and return flows from excess irrigation (R, runoff, and DP, deep percolation) in equations 5.1 and 5.2 depend on partly head-dependent ET flow terms as part of the irrigation requirement calculation (see the "Irrigation Water," "Runoff," and "Deep Percolation" sections). The irrigation demand (total farm delivery requirement, TFDR) depends on the head-dependent component of the crop-irrigation requirement (CIR). If the water supply is sufficient, the surface- or groundwater supply is driven by the irrigation demand. Therefore, the irrigation water supplied by surface water or by groundwater pumping depends indirectly on the head-dependent CIR. This head and flow dependency of the FMP is characteristic of dynamic head-, flow-, and deformation-dependent linkages in MF-OWHM2 (Hanson and others, 2014a; Schmid and others, 2014).

Consumptive Use and Evapotranspiration

Consumptive use is commonly defined as actual evapotranspiration (ET), which includes both plant transpiration and evaporation in an agricultural area (Colorado's Decision Support Systems, 1995; Gelt and others, 1999; European Environment Agency, 2004). The primary objective of estimating the consumptive use in the FMP is to estimate evapotranspiration and its separate transpiration and evaporation components (fig. 5.1), that is, crop water use and bare-soil evaporation of water. At a steady state of soil moisture, ET is composed of six components from three sources: groundwater, precipitation, and irrigation ($T_{gw\text{-}act}$, $E_{gw\text{-}act}$, T_p, E_p, T_i, E_i,). By calculating the natural components of groundwater uptake and precipitation first, the FMP can back out the crop-irrigation requirement, CIR, composed of the sum of the irrigation requirement necessary to fulfil the crop water demand for transpiration and the related evaporation losses from irrigation. Because the CIR varies with changing groundwater levels in each model cell, it has been formulated to be head dependent. For calculating the irrigation water demand, the FMP uses the computed actual evapotranspiration, $ET_{c\text{-}act}$, as the target crop consumptive use to be met. For this computation, the FMP offers options reflecting different concepts where stresses to water uptake, such as wilting and anoxic conditions in the unsaturated root zone, are either specified through the input data or explicitly simulated by the FMP.

The transpiration and evaporation components are derived from the crop consumptive use or potential crop ET and the transpiration and evaporative crop coefficients. Potential crop ET, $ET_{c\text{-}pot}$, can be specified for each crop or calculated internally as the product of specified reference ET, ET_r, and crop coefficients, K_c. Using a specified fraction of transpiration, K_t, $ET_{c\text{-}pot}$ is separated into potential crop transpiration, $T_{c\text{-}pot} = K_t\, ET_{c\text{-}pot}$, and potential crop evaporation, $E_{c\text{-}pot} = (1-K_t)\, ET_{c\text{-}pot}$. Separating E and T losses in data input is in line with multi-component ET models (Shuttleworth and Wallace, 1985; Kustas and Norman, 1997; Guan and Wilson, 2009), some variably saturated flow models (for example, HYDRUS, Simunek and others, 1999; or SWAP, Kroes and van Dam, 2003), or with the use of transpiration (K_{cb}) and evaporation (K_e) crop coefficients (Allen and others, 1998). The FMP differs from the last one by not computing K_c with separate K_{cb} and K_e coefficients, but by making use of literature data on K_c and K_{cb} to preprocess fractions of transpiration as ratios of K_c and K_{cb}. Preprocessing or estimating K_t fractions, however, is required from the user in advance rather than being calculated by the FMP.

Change in Evapotranspiration with Varying Water Levels

The primary objective of the FMP's evapotranspiration and root zone concepts is to evaluate crop-irrigation requirements for time steps as they exist in most groundwater models as opposed to irrigation scheduling software. For typical conditions of groundwater flow, suitable time steps are on the order of weeks or months, rather than days. For such time intervals, in the FMP, a convergence between inflows into and outflows out of the root zone is assumed. In the absence of precipitation and irrigation, this means that the flux from groundwater across the bottom of the root zone equals the transpiration flux into the atmosphere across the ground surface (fig. 5.2). In the presence of precipitation and irrigation, the transpiration flux equals the sum of its component fluxes from groundwater, precipitation, and irrigation (fig. 5.3). These simple mass balances of the root-zone control volume are only possible in the absence of any change in soil-water storage in the root zone. Contrary to irrigation scheduling, which requires a consideration of daily changes in soil-water storage, in the FMP it is assumed that changes in soil-water storage can be neglected for time steps commonly used in groundwater modeling (steady-state soil-moisture assumption). The FMP is not designed to simulate any situation where changes in soil-water storage have to be considered, such as short time steps or root-zone processes in deep root zones (on the order of several meters) that have high soil-water storage potential. Typically, for most agricultural settings under irrigation, soil moisture is well managed. Conversely, natural vegetation or dry-land farming may have more variable soil moisture in some settings and soil types.

The steady-state soil-moisture assumption in the FMP was evaluated using HYDRUS2D soil-column models (fig. 5.2 and 5.3). As figure 5.3 shows, however, the uptake from all three sources (groundwater, precipitation, and irrigation) does not reach the potential transpiration as specified in HYDRUS2D. The HYDRUS2D simulations demonstrated how the resulting fluxes of the actual transpiration from groundwater as the only source merge with fluxes across the water table when approaching a pseudo steady state (fig. 5.4). In addition, the variation of maximum possible actual transpiration from groundwater and irrigation concomitant with changing depths to water was determined (fig. 5.4) along with its linear approximation (figs. 5.5 and 5.6). Each diagram is split into two parts: transpiration on the left and evaporation on the right. The source of water for transpiration and evaporation can vary depending on the head. The source of water for transpiration can be precipitation (T_p), irrigation (T_i), or groundwater (T_{GW}). Transpiration from groundwater can be further subdivided into transpiration from the saturated zone ($T_{gw\text{-}sat}$) and transpiration from the unsaturated zone ($T_{gw\text{-}unsat}$). Similarly, the source of water for evaporation can be precipitation (E_p), irrigation (E_i), or groundwater (E_{GW}). It is assumed that irrigation is applied at a rate just sufficient to keep transpiration at its maximum rate.

The section that follows describes two different FMP concepts about how consumptive-use components (split into separate evaporation and transpiration flux components from precipitation, irrigation, and groundwater uptake) and crop irrigation requirement vary concomitant with changing water levels (fig. 5.6). Both concepts are piecewise linear approximations of these six flux terms and assume soil water to be at steady state. The two concepts only differ by changes in transpiration for water levels ranging between ground surface and the bottom of the root zone, but not by changes in transpiration for water levels below the root zone or for changes in evaporation concomitant with varying water levels.

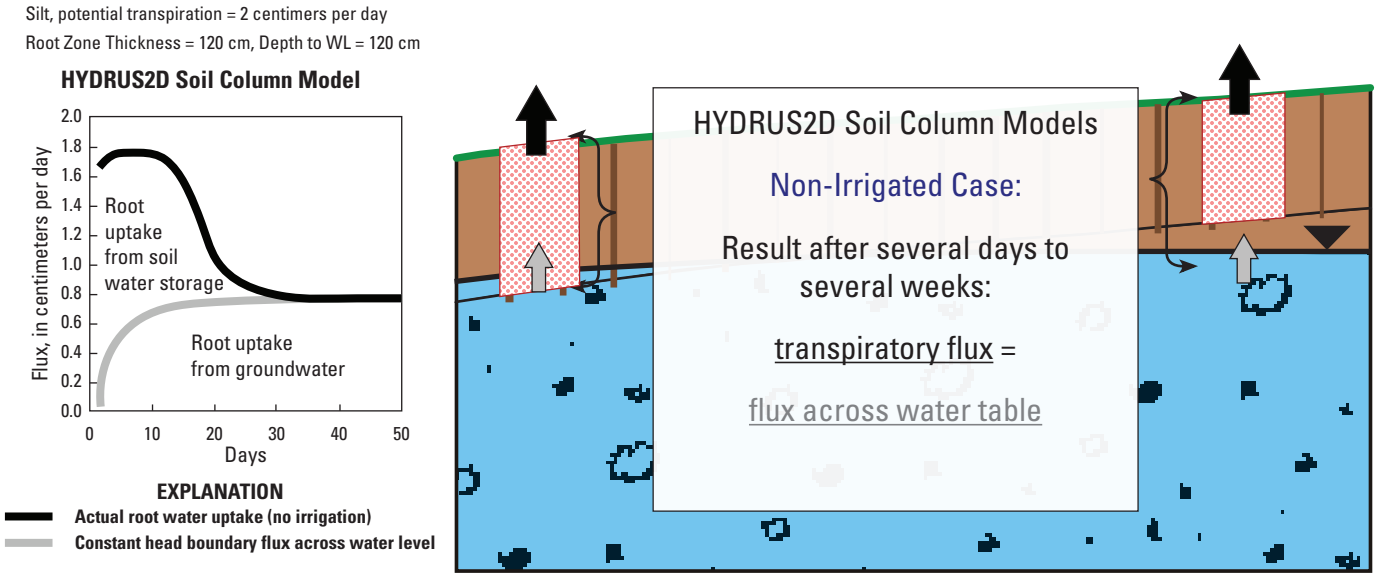


Figure 5.2. Benchmarking of steady-state assumption between transpiration from groundwater and flux across the water table for a non-irrigated case (modified from Schmid and others, 2006b).

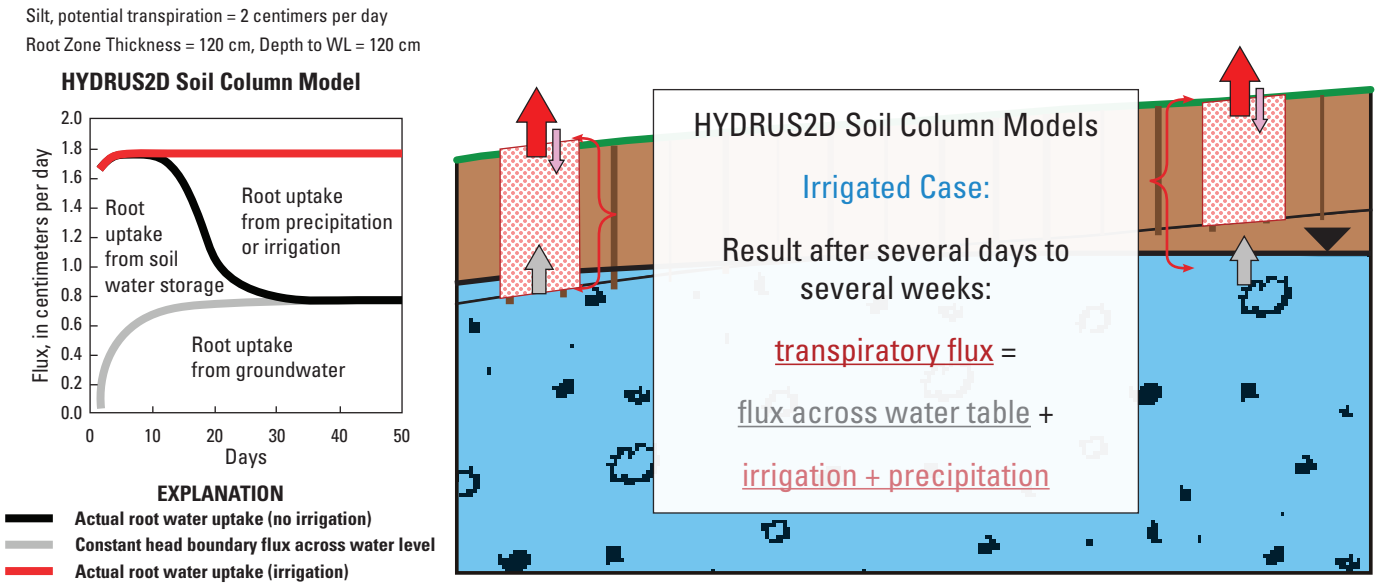


Figure 5.3. Benchmarking of steady-state assumption between transpiration from groundwater and irrigation and flux across the water table for an irrigated case (modified from Schmid and others, 2006b).

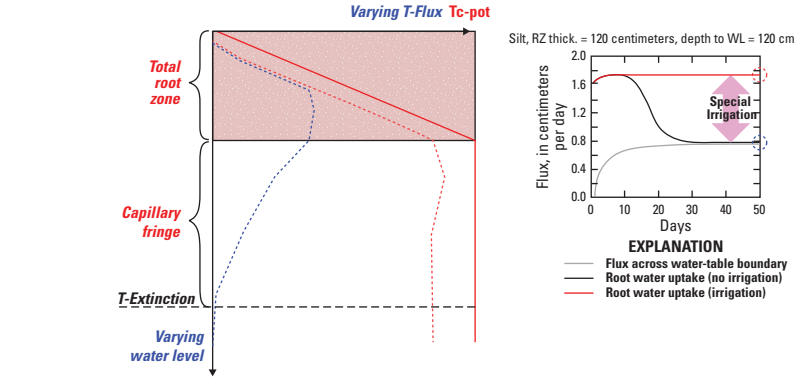


Figure 5.4. Maximum possible actual transpiration based on the assumptions of full uptake under unsaturated conditions and no uptake under saturated conditions pseudo steady-state results from HYDRUS2D for maximum possible transpiration and transpiration from groundwater only (modified from Schmid and others, 2006a, and Schmid and Hanson, 2009c). [RZ, root zone; T, transpiration; $T_{c\text{-}pot}$, maximum potential transpiration; WL, water level]

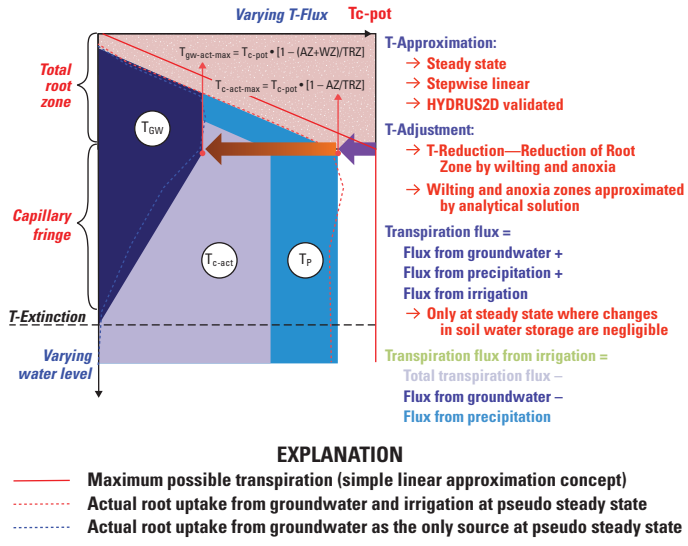


Figure 5.5. Linear approximation of pseudo steady-state groundwater uptake by roots; results from HYDRUS2D (modified from Schmid and Hanson, 2009c). [AZ, anoxia zone; WZ, wilting zone; and TRZ, transpiration root zone]

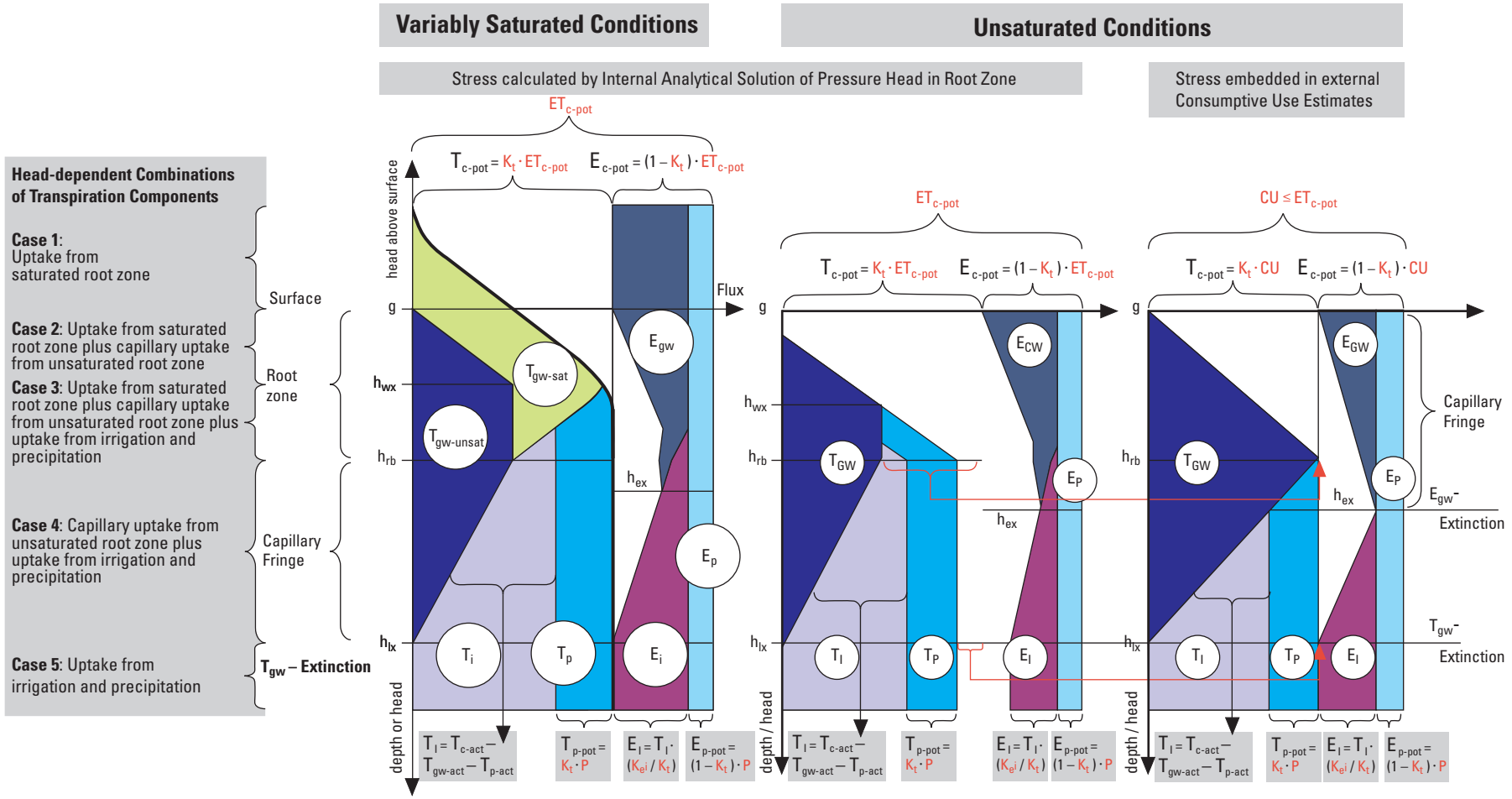


Figure 5.6. Conceptual approximations to change of crop consumptive-use components with varying head: left, reduced consumptive-use concept under variably saturated conditions; middle, reduced consumptive-use concept under unsaturated conditions; right, non-reduced consumptive-use concept under unsaturated conditions (modified from Schmid and others, 2006a; Schmid and Hanson, 2009a). The fraction of the potential evaporation or transpiration from each source as a function of head is shown by the colored regions in each diagram.

Transpiration for Water Levels Above the Base of Root Zone

Groundwater-Root Zone Interaction with Implicit Stress Assumption (Non-reduced Consumptive Use)

Assumptions for the non-reduced consumptive-use conceptual model are as follows:

- The "non-reduced consumptive-use" concept represents a piecewise linear approximation of transpiration when groundwater levels are between the bottom of the root zone and the ground surface without further reduction of maximum possible uptake (which is assumed to equal a specified consumptive use).
- When groundwater levels are between the bottom of the root zone and the ground surface, there is a linear decrease in transpiration as the groundwater level rises (red solid line in fig. 5.4). The active root zone is reduced proportionally to the depth of groundwater in the root zone and is inactive for anoxic conditions caused by saturation with groundwater. For example, if the water level rises to half the depth of the root zone, the potential transpiration would also be reduced by half. Wilting conditions are ignored and not assumed to reduce the active root zone.
- When water levels are at or below the bottom of the root zone, the entire root zone is available for uptake, and the actual transpiration can, at maximum, be equal to the potential transpiration.
- The potential transpiration is distributed equally over the root zone.

The appropriate use of the "implicit stress assumptions" are described as follows. If consumptive-use data already account for some in situ factors and stresses that constrain the actual ET, that is, if $ET_{c\text{-}pot}$ is derived from field measurements with "non-ideal" partially stressed conditions or from Moderate Resolution Imaging Spectroradiometer (MODIS) data, which may reflect a priori stressed ET. In such cases, the user would not want to further reduce the estimated transpiration to a smaller fraction of maximum uptake, unless the root zone became fully saturated. In other words, water-stress response is at optimum for an entirely unsaturated root zone and zero for saturated conditions. Lacking any better data needed to use the "reduced consumptive-use" conceptual model, such as a crop-specific water stress response function, however, the user may also default to the "non-reduced consumptive-use" concept knowing that it overestimates the transpiration from groundwater and, hence, underestimates the crop-irrigation requirement.

Groundwater and Root-Zone Interaction with Explicit Stress-Response Calculation (Reduced Consumptive Use)

Assumptions for the "reduced consumptive use" conceptual model are the following:

- The "reduced consumptive use" concept represents a step-wise linear approximation of transpiration when groundwater levels are between the bottom of the root zone and the ground surface, where maximum uptake is reduced by conditions of wilting and anaerobiosis.
- The potential crop transpiration is reduced to the actual crop transpiration in proportion to the reduction of the total root zone volume to an active root zone by wilting or anoxia (figs. 5.4 and 5.5) for negative ranges of pressure heads (unsaturated conditions) or by hydrostatic pressure for positive ranges of pressure heads (saturated conditions).
- The response of crops to stresses of wilting or anoxia is specified as crop-specific pressure heads at which uptake is either zero, indicating wilting or anaerobiosis points (Feddes and others, 1976), or at maximum (Prasad, 1988; Mathur and Rao, 1999; Simunek and others, 1999).

The appropriate use of the "explicit stress assumptions" or "reduced consumptive use" follows the assumption of a potential transpiration under unstressed conditions, as used in the HYDRUS2D soil column models, as an atmospheric boundary condition. Commonly, $ET_{c\text{-}pot}$ input data or related crop coefficients ($ET_{ref}/ET_{c\text{-}pot}$) are derived under "unstressed conditions" as, for instance, described by Allen and others (1998) for ET_c. Under stress, even at conditions of maximum uptake (such as when the water level is at the bottom of the root zone), the potential transpiration cannot be reached. Stresses that impair uptake are conditions of wilting and anoxia. This concept uses an analytical solution of the HYDRUS2D results from the vertical pressure head distribution for varying potential transpiration, soil types, root-zone depths, and crop-specific stress responses to mimic these stresses. Matching crop-specific critical pressure heads, between which uptake is possible, with an analytical solution of vertical pressure-head distribution allows the approximation of an active root zone reduced by zones of anoxia and wilting. This allows the FMP to decrease transpiration proportionately to the reduction in the active root zone volume. Using $ET_{c\text{-}act}$ as input data for this option would erroneously double-account for simulated stresses already inherent in the measurement.

The "reduced consumptive-use" concept was initially developed to allow root uptake of groundwater only for crops from unsaturated root zones, that is, for stress-response functions at critical negative pressure heads (Schmid and others, 2006a).

Certain applications, however, such as riparian evapotranspiration (for example, willow trees) and certain rice irrigation procedures, required an expansion of the concept to allow for possible uptake from saturated portions of the root zone (phreatic zones), where there are positive ranges of pressure heads (Schmid and Hanson, 2009b). Moreover, the concept accounts for the eventual reduction of uptake as positive pressure heads increase in the saturated root zone or, for ponding conditions, up to a user-specified limit of water level above the ground-surface elevation. This concept allows the simulation of water uptake and irrigation requirements for natural vegetation or crops rooting in soils that are fully or partially saturated. Full or partial saturation is achieved by the groundwater level rising into the root zone or even above the ground surface (for example, in alluvial valleys). Under such conditions, irrigation is required only for vegetation specified as irrigated crops where uptake from groundwater does not fully satisfy the potential transpiration.

Stress-Response Functions

Varying hydraulic pressure heads in a root zone imposes different levels of stress on a crop, resulting in water uptake ranging between a maximum and zero. The relationship between dimensionless water uptake, α ($0 \leq \alpha \leq 1$), and pressure head, ψ, is called a 'stress response function' (fig. 5.7). Such a crop-specific stress response function can be defined by four critical pressure heads at which water uptake is at its maximum (ψ_2, ψ_3) or at which water uptake ceases either as a result of anoxia or of wilting (ψ_1, ψ_4). Note that if all four pressures are set to zero, then the stress-response function is not used.

Among several sources in the literature, Taylor and Ashcroft (1972) and Wesseling (1991) provide the most detailed databases for stress-response pressure heads for numerous crops. If data are lacking for aerobic crops, $\psi 1$ (anoxia) may be approximated as the air entry pressure head, ψ_a, in the water-retention function, where the water content, θ, approaches the porosity, n. Common bounds for the field capacity, if known, may provide an approximation of the value for ψ_2 (normally between –0.06 and –0.3 bar or –60 and –300 centimeter pressure head). A maximum allowable depletion (in percent) describes the reduction of the water content at field capacity. The minimum allowable water content, below which transpiration is reduced, can be related to a pressure head to approximate the value of ψ_3. The permanent wilting point for most crops, ψ_w, is at about −15 bar or −15,000-centimeter pressure head and can be used as an indication of the ψ_4 value. The approximation of ψ_1 though ψ_4 from the air-entry pressure head, a range of field-capacity pressure heads, the maximum allowable depletion, and the permanent wilting point can be difficult for some crop types, however. Although the air-entry pressure head and the field capacity vary by soil type, the maximum allowable depletion and the permanent wilting point vary by crop type. Because the FMP requires stress-response pressure heads to be crop-type specific attributes, the user is encouraged to search the literature for databases of strictly crop-type related stress-response pressure heads.

The FMP simplifies the stress-response function to a step function, where water uptake is considered at maximum between the averages of ψ_1 and ψ_2 and of ψ_3 and ψ_4 (fig. 5.7). Zones in the root zone where conditions of wilting or anoxia eliminate water uptake (in the FMP, wilting or anoxia zones) are found by matching these averages of crop-specific anoxia- or wilting-related pressure heads to a vertical steady-state pressure-head distribution.

Analytical Solution of Vertical Pressure-Head Distribution

Pressure-head distributions across the depth of a root zone can be approximated by various models of the vertical pressure-head configuration across the root zone (fig. 5.7). One approach is to solve for vertical transient pressure-head distributions using Richard's equation-based variable-saturation flow models. These require soil-water constitutive input parameters and may be computationally expensive when linked to regional groundwater models, however. Instead, the FMP uses analytical solutions of vertical steady-state pressure-head distributions derived from transient, Richard's equation-based variably saturated soil-column models upon convergence of actual root uptake and upward fluxes across the water table after time intervals of several days to weeks (fig. 5.4). These soil-column models were developed using HYDRUS2D (Simunek and others, 1999) for various soil-specific soil-water constitutive parameters, crop-specific stress-response functions, root-zone depths, depths to groundwater, and rates of potential transpiration where groundwater is the only source of water for root uptake (Schmid, 2004). For groundwater rising above the root-zone base, a wilting zone in the upper part of the root zone decreased linearly, and an anoxic fringe above the water table remained constant until its top reached ground surface. For other HYDRUS2D simulations, infiltration (for example, from precipitation or irrigation) was added as an additional source of water for root uptake. The actual transpiration, $T_{c\text{-}act,}$ did not reach $T_{c\text{-}pot,}$ however, because infiltration wetting-fronts also can have pressure heads at which the crop's response to anoxia reduces transpiration (Drew, 1997). Hence, even for root zones not influenced by groundwater, $T_{c\text{-}act}$ cannot exceed an anoxia-constrained maximum possible $T_{c\text{-}act\text{-}max}$. Adding infiltration in excess of $T_{c\text{-}act\text{-}max}$ resulted in transpiration-inefficient losses. $T_{c\text{-}act\text{-}max}$ may be diminished further if pressure heads of a wetting front are higher than those of an anoxia fringe above a water table or if drainage in lower parts of the root zone causes wilting.

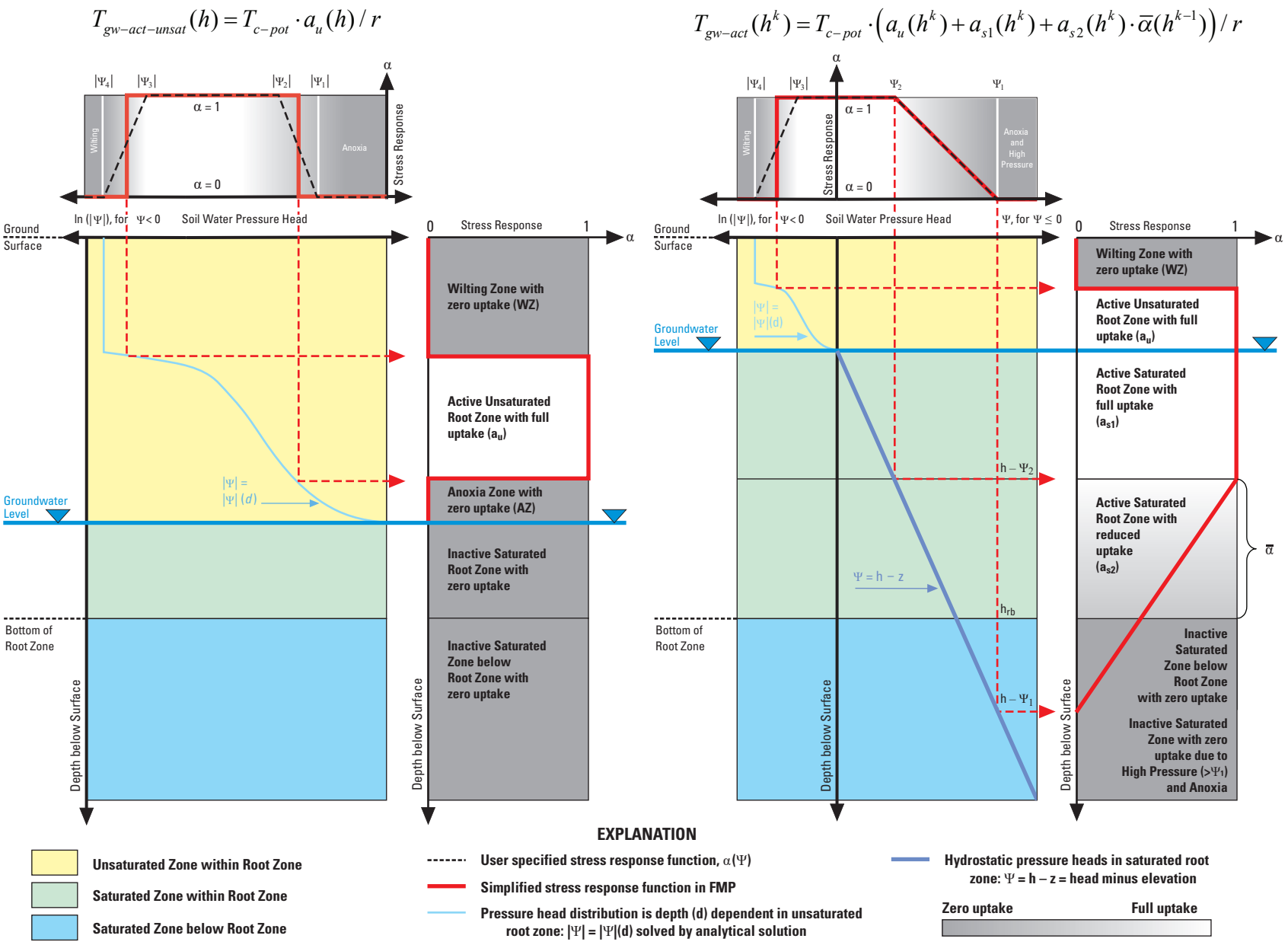


Figure 5.7. Root-zone pressure head relative to groundwater level for unsaturated conditions (left) and variably saturated conditions (right), where the top shows a crop-specific stress-response function and the bottom shows analytical function fitting the vertical pressure-head distribution by depth (modified from Schmid and Hanson, 2009a).

Matching Crop Stress Response to the Root-Zone Pressure Head

Root Uptake Under Unsaturated Conditions

Regions of the root zone with pressure heads less than the average of ψ_4 and ψ_3 or greater than the average of ψ_2 and ψ_1 are considered inactive wilting and anoxia zones, respectively (WZ, AZ). For a water level at the base of the root zone (h_{rb}), the residual active unsaturated root zone (a_u) is equal to the total root zone minus WZ and AZ (eq. 5.5 for $h_{wx} \geq h > h_{rb}$). As the groundwater level rises, the vertical pressure-head distribution is shifted upward. The active root zone and the AZ remain constant, but the WZ at the top end of the pressure-head distribution is gradually eliminated until the water level reaches a point where the depth of the WZ is zero (water level at that point = h_{wx}). For water levels rising beyond this point, the active unsaturated root zone is reduced linearly (eq. 5.5) for $h_{ux} > h > h_{wx}$ until the top of the anoxia fringe above the water level reaches the ground-surface elevation (fig. 5.5). At this position of the water level, transpiration reaches extinction (water level at that point = h_{ux}). The vertical range over which pressure heads are less than the wilting-point pressure head is found by assuming groundwater to be the only source for transpiration.

Root Uptake Under Variably Saturated Conditions

For deep root zones and for groundwater levels in the root zone, roots can take up water under unsaturated as well as saturated conditions. Above the groundwater level, uptake can be zero or full under unsaturated conditions in the WZ and the active unsaturated root zone (AURZ), respectively. For crops characterized by positive critical pressure heads ψ_1 and ψ_2, the active unsaturated root zone (a_u) is not restricted by anoxia (AZ = 0; fig. 5.7). Below the groundwater level, uptake can be full or reduced in the active saturated root zone (fig. 5.7):

- Uptake is full under saturated conditions for a region of positive pressure heads in the root zone that range from zero at the groundwater level to the user-specified pressure head ψ_2. This region of the root zone is defined as the active saturated root zone 1 (a_{s1}). In this zone, the stress response to water uptake, α, is equal to one, indicating that full uptake is possible. For water levels rising above the ground-surface elevation (GSE), the a_{s1} extends from the GSE to where the critical pressure head ψ_2 is found.

- Uptake is reduced under saturated conditions for a region of positive pressure heads that range from ψ_2 (full uptake) to ψ_1 (zero uptake) or the pressure head at the base of the root zone, whichever is least. We define this region of the root zone as active saturated root zone 2 (a_{s2}). In this zone, the stress response to water uptake, α, is taken to be equal to the average of stress responses, $\bar{\alpha}$, owing to pressure heads that are found in the root zone between ψ_2 and ψ_1 or the pressure head at the base of the root zone. Where ψ_1 is found below the base of the root zone (fig. 7), a_{s2} is not bound by ψ_1, but by a nonzero pressure head at the base of the root zone.

For water levels at and above the base of the root zone, the uptake under saturated conditions over distance r from the surface to the groundwater elevation (r) is formulated in eqs. 5.3 and 5.6. Noticeably, a_{s1}, a_{s2}, and $\bar{\alpha}$ depend on the vertical location of the hydrostatic pressure heads ψ_1 and ψ_2 (head elevations h–ψ_1 and h–ψ_2) because ψ_1 and ψ_2 move vertically up or down as the water level rises or falls. The terms a_{s1}, a_{s2}, and α depend on the simulated groundwater level and, therefore, are head-dependent terms. To avoid the term $a_{s2} \cdot \bar{\alpha}$ becoming nonlinear in head, we evaluate $\bar{\alpha}$ on the basis of the head of the previous iteration (k–1), whereas a_{s2} is related to the head of the current iteration (k) in equation 5.3, which calculates actual transpiration from groundwater under variably saturated conditions:

$$T_{gw-act}\left(h^k\right) = T_{c-pot}\left(h^k\right)\left(\frac{a_u\left(h^k\right) + a_{s1}\left(h^k\right) + a_{s2}\left(h^k\right)\bar{\alpha}\left(h^{k-1}\right)}{r}\right) \tag{5.3}$$

Although figure 5.7 demonstrates a situation for a specific water-level elevation, figure 5.6 illustrates the conceptual approximation to the change of all transpiration and evaporation components concomitant with varying groundwater level. Depending on where the water level is positioned (above, in, or below the root zone), five different cases of combinations of up to four transpiration components are possible. These components are fed by capillary rise from groundwater (unsaturated root zone), by direct uptake from groundwater (saturated root zone), by irrigation, or by precipitation. For instance, for case 3 (fig. 5.6), the water level rises only slightly above the base of the root zone and wilting still might occur in the drying top soil. Transpiration is fed by groundwater uptake from the unsaturated and saturated part of the root zone. The deficit between the transpiration from groundwater and the maximum possible transpiration may be supplemented by precipitation or irrigation.

If the water level continues to rise (case 2, fig. 5.6), however, all possible transpiration is from groundwater uptake from the unsaturated and saturated root zone. Finally, when the water level rises above the ground surface and ponds, uptake is only from the saturated root zone (case 1, fig. 5.6).

Examples 1, 2, and 3 (fig. 5.8) show how the total transpiration uptake from the saturated root zone (light-green curve) is composed of the uptake from the fully active and partially active parts of the saturated root zone. The uptake from the fully active root zone (light-blue curve) is a piecewise linear approximation (eq. 5.6 in the "Mathematical Representation of Consumptive-Use Components" section). The uptake from the partially active root zone (purple curve) depends on the product of two head-dependent terms: the depth of this zone and the average stress response, $\bar{\alpha}$, in this zone. Therefore, as shown in figure 5.8, this part of the uptake is nonlinear as head changes (eq. 5.6). For select positive ψ_1 and ψ_2 values, the range of positive pressure heads with reduced uptake ($\psi_1 - \psi_2$) can be less than the thickness of the total root zone. In this case, the "partial uptake zone," a_{s2}, and the average stress response, $\bar{\alpha}$, in that zone can remain constant as water level changes if the elevation where ψ_2 is found (head – ψ_2) is less than the ground-surface elevation, and the elevation where ψ_1 is found (head – ψ_2) is greater than the base of the root zone.

Transpiration for Water Levels between the Root Zone and the Extinction Depth

Transpiration is simulated for groundwater levels between the base of the root zone and the extinction depth in the same way for the crop consumptive-use concepts (fig. 5.6) and is assumed to decrease linearly with depth. As the groundwater level drops below the root zone, the actual transpiration from groundwater $T_{gw\text{-}act}$ is assumed to decrease linearly from the respective maximum actual transpiration from groundwater, $T_{gw\text{-}act\text{-}max}$, at the base of the root zone to zero at a transpiration-extinction depth (defined to be equal to the height of the capillary fringe) below the root zone (eqs. 5.5 and 5.6 for $h_{rb} \geq h > h_{lx}$ in the "Mathematical Representation of Consumptive-Use Components" section). For a groundwater level above the base of the root zone, the total actual transpiration $T_{c\text{-}act}$ is assumed to remain constant at the maximum actual transpiration $T_{c\text{-}act\text{-}max}$ (fig. 5.6 and eq. 5.4 for $h \geq h_{rb}$ in the "Mathematical Representation of Consumptive-Use Components" section).

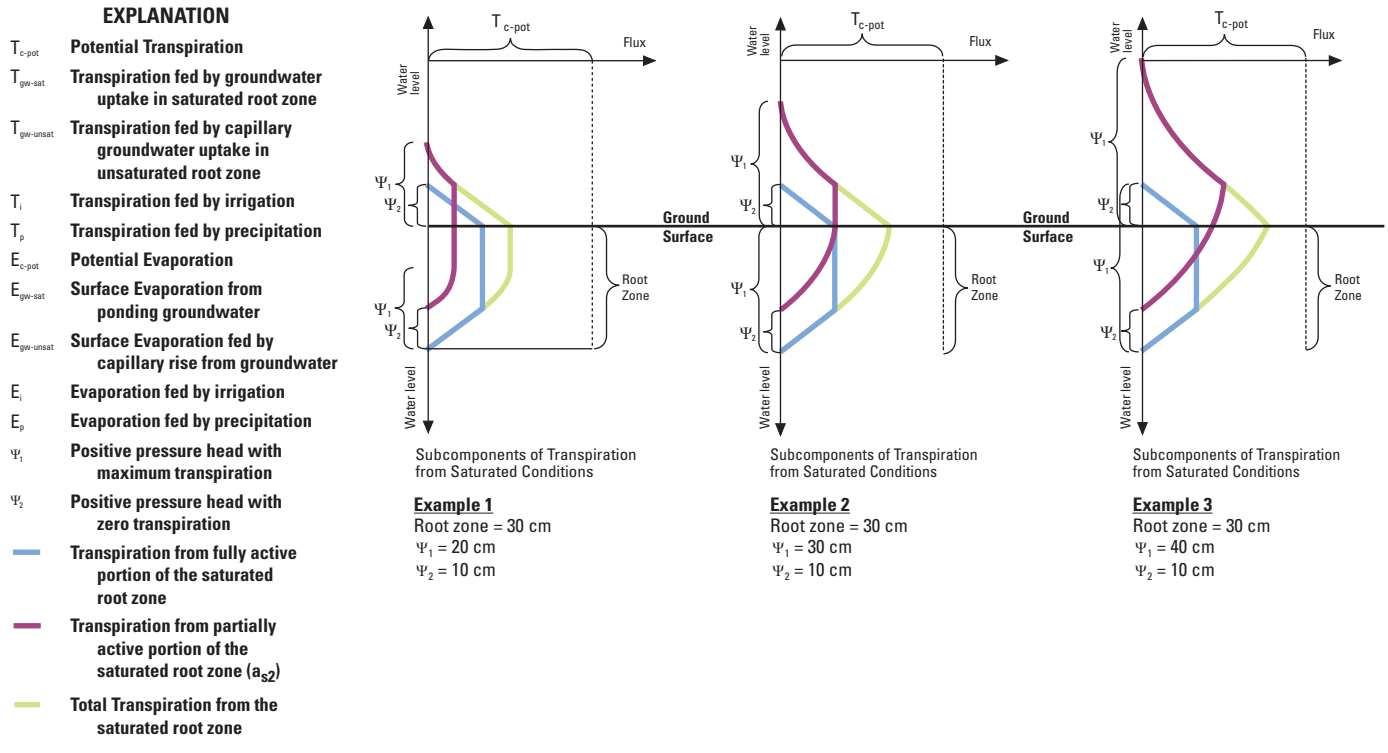


Figure 5.8. Change in water uptake for transpiration from fully and partially active parts of the saturated root zone as water levels vary (three examples with different ψ_1 values; modified from Schmid and Hanson, 2009a). [cm, centimeter]

Transpiration from Precipitation

As transpiration from groundwater increases with rising groundwater level, the amount of actual transpiration provided by precipitation, $T_{p\text{-}act}$, decreases as long as it is less than the available potential transpiration from precipitation that is needed. Hence, in this range of water levels, $T_{p\text{-}act}$ is head-dependent (fig. 5.6 and eq. 5.7 for $T_{p\text{-}pot} > T_{c\text{-}act} - T_{gw\text{-}act}$ in the "Mathematical Representation of Consumptive-Use Components" section).

Crop-Irrigation Requirement

The irrigation requirement for transpiration in each model cell (T_i; fig. 5.6) is simulated by subtracting the actual transpiration from groundwater $T_{gw\text{-}act}$ and the actual transpiration from precipitation $T_{p\text{-}act}$ from the total potential transpiration. After adding the irrigation needs for evaporation (E_i; eq. 5.9 in the "Mathematical Representation of Consumptive-Use Components" section) to compensate surface-evaporation losses from irrigation, this yields the estimate of the crop-irrigation requirement, CIR (eq. 5.8 in the "Mathematical Representation of Consumptive-Use Components" section). Because CIR depends on head-dependent terms, it is also head-dependent.

Evaporation for Water Levels Between Ground Surface and the Extinction Depth

Evaporation from groundwater is simulated for groundwater levels between the ground surface and the evaporation extinction depth, h_{lx} (fig. 5.6). The maximum actual evaporation from groundwater, $E_{gw\text{-}act\text{-}max}$, is assumed to equal the proportion of the saturation water-vapor pressure deficit over exposed no-crop areas (potential evaporation $E_{c\text{-}pot}$) that is not compensated by evaporation of precipitation (E_p), where $E_{gw\text{-}act\text{-}max} = E_{c\text{-}pot} - E_p$. For the crop consumptive-use concepts, the extinction of evaporation from groundwater is likely when the highest point of the capillary fringe is below the ground surface (Robinson, 1958). Thus, evaporation from groundwater, $E_{gw\text{-}act}$, is assumed to decrease linearly with the groundwater level from the maximum actual evaporation from groundwater, $E_{gw\text{-}act\text{-}max}$, at the ground surface to no evaporation at the evaporation extinction depth (defined to be equal to the height of the capillary fringe) below ground surface (eq. 5.10 in the "Mathematical Representation of Consumptive-Use Components" section).

Mathematical Representation of Consumptive-Use Components

The FMP calculates a maximum actual transpiration ($T_{c\text{-}act}$, eq. 5.4) and proportions of transpiration fed by uptake from groundwater ($T_{gw\text{-}act}$, eq. 5.5 for unsaturated and eq. 5.6 for variably saturated conditions), precipitation ($T_{p\text{-}act}$, eq. 5.7), and supplemental irrigation ($T_{i\text{-}act}$, eq. 5.8), assuming no changes in soil-water storage between time steps and equal spatial distribution of roots and potential transpiration ($T_{c\text{-}pot}$) in the root zone. In summary, the estimate of actual from potential transpiration in the FMP is formulated as three components distinguished by water source: groundwater, precipitation, and irrigation.

Actual crop transpiration is estimated as follows:

$$T_{c-act} = \begin{Bmatrix} 0 \\ T_{c-pot}\,\dfrac{h_{ux} - h}{r} \\ T_{c-pot}\left(1 - \dfrac{a}{r}\right) = T_{gw-act-\max} \end{Bmatrix} \quad h_{ux} = g - a \quad \begin{matrix} h \geq h_{ux} \\ h_{ux} > h > h_{rb} \\ h \leq h_{rb} \end{matrix} \tag{5.4}$$

Actual transpiration from groundwater under unsaturated conditions is shown here:

$$T_{gw-act} = \begin{Bmatrix} 0 \\ T_{c-pot}\left(\frac{a_u}{r}\right) = T_{gw-act-\max} \\ T_{gw-act-\max}\left(1 - \frac{h_{rb} - h}{d}\right) \end{Bmatrix} \quad \begin{Bmatrix} a_u = g - a - h \\ a_u = r - a - w \end{Bmatrix} \quad \begin{matrix} h \geq h_{ux} \\ \begin{Bmatrix} h_{ux} > h > h_{wx} \\ h_{wx} \geq h > h_{rb} \end{Bmatrix} \\ h_{rb} \geq h > h_{lx} \\ h \leq h_{lx} \end{matrix} \tag{5.5}$$

Actual transpiration from groundwater under variably saturated conditions is estimated as follows:

$$T_{gw-act} \quad h \geq g + \varphi_1$$

$$= \begin{Bmatrix} 0 \\ T_{c-pot}\left(a_u + a_{s1} + a_{s2}\,\bar{\alpha}\right) \\ T_{gw-act-\max}\left(1 - \frac{h_{rb} - h}{d}\right) \\ 0 \end{Bmatrix} \begin{Bmatrix} a_u = 0 \\ a_u = g - h \\ a_u = r - w \\ a_{s1} = 0 \\ a_{s1} = r \\ a_{s1} = g - (h - \varphi_2) \\ a_{s1} = \varphi_2 \\ a_{s1} = h - h_{rb} \\ a_{s2} = 0 \\ a_{s2} = r \\ a_{s2} = (g - (h - \varphi_2))a \\ a_{s2} = (\varphi_1 - \varphi_2)a \\ a_{s2} = (h - \varphi_2 - h_{rb})a \end{Bmatrix} \begin{matrix} h_{rb} \geq h > h_{lx} \\ h \leq h_{lx} \end{matrix}$$

$h > g$	$\vee$			$a = 0$
$g > h > h_{wx}$		$h > h_{rb}$		$a = 0$
$h_{wx} \geq h > h_{rb}$				
$h < h_{rb}$	$\vee$			$a_u = 0$
$h > g$	$\wedge$	$h - \varphi_2 > g$		$a_u = 0$
$h > g$	$\wedge$	$h - \varphi_2 < h_{rb}$		$a = 0$
$h < g$	$\wedge$	$h - \varphi_2 < h_{rb}$		$a = 0$
$h < g$	$\wedge$	$h - \varphi_2 > h_{rb}$	$\bar{\alpha} = (a_g + a_{rb})/2$	
$h - \varphi_2 < h_{rb}$	$\vee$	$h - \varphi_1 > g$		$a_u, a_{s1} = 0$
$h - \varphi_2 > g$	$\wedge$	$h - \varphi_1 < h_{rb}$	$\bar{\alpha} = a_g/2$	$a_u, a_{s1} = 0$
$h - \varphi_2 > g$	$\wedge$	$h - \varphi_1 > h_{rb}$	$\bar{\alpha} = 1/2$	$a = 0$
$h - \varphi_2 < g$	$\wedge$	$h - \varphi_1 > h_{rb}$		$a = 0$
$h - \varphi_2 < g$	$\wedge$	$h - \varphi_1 < h_{rb}$	$\bar{\alpha} = (1 + a_{rb})/2$	

(5.6)

Actual transpiration from precipitation is estimated in this way:

$$T_{p-act} = \begin{Bmatrix} 0 \\ T_{c-act} - T_{gw-act} \\ T_{p-pot} \end{Bmatrix} \quad \begin{matrix} h \geq h_{wx} & with & h_{wx} = g - r + w \\ h < h_{wx} & \wedge & T_{p-pot} > T_{c-act} - T_{gw-act} \\ h < h_{wx} & \wedge & T_{p-pot} \leq T_{c-act} - T_{gw-act} \end{matrix} \tag{5.7}$$

Actual transpiration from irrigation is estimated as follows:

$$T_{i-act} = T_{c-act} - T_{gw-act} - T_{p-act} \tag{5.8}$$

Actual evaporation from irrigation is estimated as follows:

$$E_{i-act} = T_{i-act}\left(\frac{K_e^i}{K_t}\right) \tag{5.9}$$

Actual evaporation from groundwater is estimated in this way:

$$E_{gw-act} = \begin{Bmatrix} E_{c-pot} - E_{p-act} \\ \left(T_{c-act} - T_{gw-act}\right)\left(1 - \frac{g+h}{d}\right) \\ 0 \end{Bmatrix} \quad \begin{matrix} h \geq g \\ g < h < h_{ex} \quad \text{with} \quad h_{ex} = g - d \\ h \leq h_{ex} \end{matrix} \tag{5.10}$$

where

a is the depth of the anoxia fringe [L];
w is the depth of wilting zone [L];
r is the total depth of root zone [L];
d is the depth of capillary fringe [L];
g is the ground-surface elevation [L];
h is the groundwater-head elevation [L];
a_u is the active unsaturated root zone;
a_{s1} is the active saturated root zone at maximum uptake;
a_{s2} is the active saturated root zone at reduced uptake;
$\bar{\alpha}$ is the average of stress responses found in a_{s2};
φ_1 is the positive pressure head at which water uptake ceases as a result hydrostatic pressure;
φ_2 is the pressure head at which water uptake is at maximum;
h_{rb} is the groundwater-head elevation at the base of the root zone [L];
h_{ux} is the head elevation at which the top of anoxia fringe (a) above the water level reaches ground-surface elevation (g) with rising head, which is called the elevation of upper transpiration extinction [L];
h_{wx} is the head elevation at which the bottom of the wilting zone, w, reaches ground-surface elevation (g) with rising head, which is called the elevation of wilting zone extinction [L];
h_{lx} is the head elevation at which the top of capillary fringe (d) is at base of root zone (h_{rb}) which is the elevation of lower transpiration extinction [L];
h_{ex} is the head elevation at which top of capillary fringe (d) reaches ground-surface elevation (g) which is the elevation of evaporation extinction [L];
K_t is the transpiration fraction of $ET_{c\text{-}pot}$; and
K_e^i is the evaporation fraction of $ET_{c\text{-}pot}$ related to irrigation.

In the "reduced consumptive-use" concept in the unsaturated root zone, $T_{c\text{-}act}$ varies linearly in equation 5.4 between the elevation of upper transpiration extinction, h_{ux}, and the elevation of the root-zone base, h_{rb}. For heads below the root-zone base, $T_{c\text{-}act}$ is constant and reduced by the ratio between the thicknesses of the anoxia fringe, a, and the total root zone, r. In equation 5.5, $T_{gw\text{-}act}$ varies linearly between the elevation of upper transpiration extinction, h_{ux}, and the elevation of wilting zone extinction, h_{wx}. For heads between h_{wx} and root-zone base, $T_{gw\text{-}act}$ is constant and reduced from $T_{c\text{-}pot}$ to a maximum actual transpiration from groundwater, $T_{gw\text{-}act\text{-}max}$, by the ratio of the sum of thicknesses of the anoxia and wilting zones, a + w, to the total root zone, r. $T_{gw\text{-}act}$ also varies linearly between the head elevations of the root-zone base and the lower transpiration extinction, h_{lx}.

For crops able to take up water from a variably saturated root zone, $T_{gw\text{-}act}$ is supplied from the active unsaturated root zone, a_u, the active saturated root zone with maximum uptake, a_{s1}, and the active saturated root zone with reduced uptake, a_{s2} (eq. 5.6).

The calculation of a_u is as explained previously, and the only difference is that the anoxic zone, a, is absent. The uptake from the fully active root zone, $T_{gw\text{-}act\text{-}as1}$, is a piecewise linear approximation that varies linearly between the elevations of ponding water level above the ground surface and the elevation in the saturated root zone at which the positive hydrostatic pressure head equals ψ_2, if it is above the base of the root zone (head – ψ_2). If the water level is below the surface and ψ_2 is still above the base of the root zone, $T_{gw\text{-}act\text{-}as1}$ is constant and reaches ψ_2. If the water drops below the surface and ψ_2 is below the base of the root zone, then $T_{gw\text{-}act\text{-}as1}$ is reduced linearly to zero as the water level reaches the base of the root zone. The uptake from the partially active root zone depends on the product of two head-dependent terms: the depth of this zone and the average stress response, $\bar{\alpha}$, in this zone. Therefore, this part of the uptake varies nonlinearly with changing head. For select positive ψ_1 and ψ_2 values, the range of positive pressure heads at which uptake is reduced ($\psi_1 - \psi_2$) may be less than the thickness of the total root zone. In this case, the "partial uptake zone," a_{s2}, and the average stress response, $\bar{\alpha}$, in that zone can remain constant with a moving water level, as long as the elevation of ψ_2 (head – ψ_2) is less than the ground-surface elevation and as long as the elevation of ψ_1 (head – ψ_1) is greater than the base of the root zone. In equation 5.7, $T_{p\text{-}act}$ is equal to $T_{p\text{-}pot}$, except when limited to the remainder of $T_{c\text{-}act}$ that is not yet satisfied by transpiration from $T_{gw\text{-}act}$.

For the non-reduced consumptive use concept, wilting and anoxia above the water level are not simulated (a = 0, w = 0 in eq. 5.4 and 5.5), but $T_{c\text{-}pot}$ is still linearly reduced to $T_{c\text{-}act}$ (eq. 5.4) or $T_{gw\text{-}act}$ (eq. 5.5) as the active root zone is reduced by a rising water level. $T_{c\text{-}act}$ equals $T_{c\text{-}pot}$ for water levels below the root-zone base, and $T_{gw\text{-}act}$ reaches $T_{c\text{-}pot}$ for water levels at the root-zone base.

The actual evaporation from precipitation, $E_{p\text{-}act}$, is equal to the potential evaporation from precipitation, $E_{p\text{-}pot}$, if precipitation in open areas exceeds $E_{p\text{-}pot}$ and is equal to precipitation in open areas if $E_{p\text{-}pot}$ exceeds the precipitation. The potential evaporation from irrigation, $E_{i\text{-}pot}$, can be reduced in open and exposed areas if not fully wetted. Evaporation fractions of $ET_{c\text{-}pot}$ related to irrigation, K_e^i, therefore, can be smaller than ($1–K_t$). If ET input data reflect local wetting patterns of irrigation methods and a reduction in evaporation is implicitly accounted for, then the user should keep $K_e^i = (1–K_t)$. In equation 5.9, the actual evaporation from irrigation, $E_{i\text{-}act}$, accounts for losses of irrigation to evaporation and varies proportionally to the irrigation requirement for transpiration by a ratio of K_e^i and K_t.

The remaining saturation water-vapor pressure deficit for the exposed areas that is not yet satisfied by $E_{p\text{-}act}$ or $E_{i\text{-}act}$ is assumed to be met by evaporation of capillary groundwater as long as the groundwater level in a cell keeps the capillary fringe partially above the extinction depth. The evaporation from groundwater, $E_{gw\text{-}act}$, varies linearly concomitant with the groundwater level (eq. 5.10) between zero for groundwater heads below the elevation of evaporation extinction, h_{ex} (which equals surface elevation, g, minus capillary fringe, c), and a maximum for heads rising to or above ground surface, g.

Irrigation Water

Irrigation water, *I* (in root-zone mass-balance eqs. 5.1 and 5.2), is equal either to irrigation demand or to irrigation supply. Irrigation water is equal to demand-driven-by-agricultural, urban, or natural vegetation consumptive use (that is, demand not met by precipitation or uptake from groundwater) if it can be met by irrigation supply components. Irrigation water is equal to supply if supply is restricted by any shortage or constrained by controls on stream diversions, pumping, and imported water. In short, the FMP follows a demand-driven, but supply-constrained system. This section discusses the computation of unmet water demand, which in cases of supply sufficiency equals the quantity of irrigation water, *I*. Situations where *I* is different from the unmet water demand are discussed in the "Balance Between Water Supply and Demand" section.

In the FMP, the crop-irrigation requirement, CIR, is equal to the sum of actual transpiration from irrigation and evaporation from irrigation, $ET_{i\text{-}act}$ (eqs. 5.8, 5.9, and 5.11). It is computed for each model cell at each transient time step. It assumes a pseudo-steady state between all flows into and out of the root zone at the end of time intervals, typical for MODFLOW. The FMP calculates an irrigation-delivery requirement for a specific time step iteratively on the basis of a dynamically updated groundwater head-dependent evaporative crop-irrigation requirement, $CIR = ET_{i\text{-}act}$, and CIR yields *I* if increased by inefficient losses at that time step (eq. 5.12). The CIR is computed only for cells that have land use defined as either an irrigated urban landscape or an irrigated agricultural crop. It is zero for cells with non-irrigated land use. The total irrigation water demand for each WBS (TFDR) is computed as cell delivery requirements for all cells in the unit (eq. 5.13).

Crop irrigation requirement equals the actual evapotranspiration from irrigation at each cell as follows:

$$CIR^{t,k}\left(h^{t,k-1}\right) = ET_{i-act}^{t,k}\left(h^{t,k-1}\right) = T_{i-act}^{t,k}\left(h^{t,k-1}\right) + E_{i-act}^{t,k}\left(h^{t,k-1}\right) \tag{5.11}$$

where

CIR is the crop irrigation requirement for each cell [LT^{-1}];
$T_{i\text{-}act}$ is the proportion of the actual transpiration supplied by irrigation [LT^{-1}];
$E_{i\text{-}act}$ is the actual evaporation loss from irrigation [LT^{-1}] proportional to $T_{i\text{-}act}$; and
$h_{t,k-1}$ is the head at the previous iteration, k–1, for a particular time step, t.

Irrigation delivery requirement at each cell, c, is adjusted as follows:

$$I^{t,k}\left(h^{t,k-1}\right) = \frac{ET_{i-act}^{t,k}\left(h^{t,k-1}\right)}{e^{t}} \tag{5.12}$$

where

I is the irrigation delivery requirement [LT^{-1}], and
e_t is the on-farm efficiency of the delivery system (dimensionless).

Total delivery requirement at each farm, f, or WBS is the summation over nc cells as follows:

$$TFDR_{f}^{t,k}\left(h^{t,k-1}\right) = \sum_{c=1}^{nc} I_{c}^{t,k}\left(h_{c}^{t,k-1}\right) \tag{5.13}$$

where

$TFDR$ is the total farm delivery requirement for a specific farm [LT^{-1}].

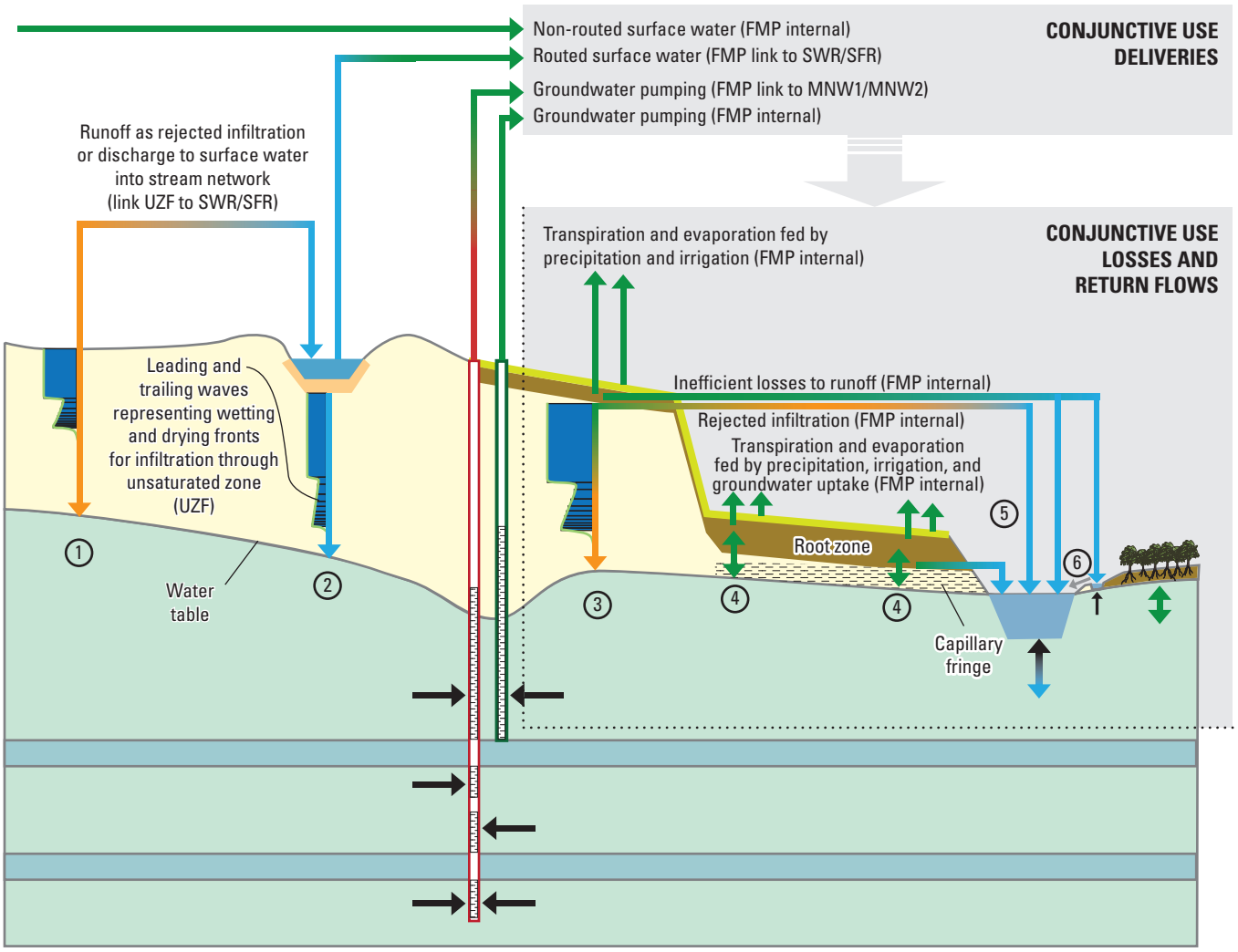
The on-farm efficiency is defined as the fraction of the total irrigation water that is used beneficially on the farm.

Runoff

In general, overland runoff can be composed of several flow components, such as (1) direct runoff, (2) interflow from excess precipitation and irrigation, (3) runoff generated by infiltration in excess of the saturated hydraulic conductivity of the unsaturated zone beneath the root zone, and (4) runoff from groundwater discharge and from rejected infiltration in areas of high groundwater levels. The FMP captures many, but not all these components. The FMP was initially developed to assess flood and basin-level irrigation along the Rio Grande of New Mexico (Schmid and others, 2009) or the Sacramento and San Joaquin Valleys of California (Schmid and Hanson, 2009a), where slopes are gradual and direct runoff is negligible, but interflow runoff can matter in different intensities for irrigation and precipitation. Hence, FMP simulates the second runoff component (fig. 5.9). The last two runoff components are available in the FMP by a linkage to the UZF (unsaturated-zone flow) package (Schmid and Hanson, 2009b) for vadose zones that extend below the root zone (fig. 5.9).

The FMP computes runoff, R (in root-zone mass-balance eqs. 5.1 and 5.2), as the proportion of crop-inefficient losses from precipitation or irrigation that contributes to runoff. Runoff related to precipitation, R_p, is calculated as follows:

$$R_{p} = \left(P - ET_{p-act}\right) f_{r}^{P-loss} \tag{5.14}$$



(Modified from Schmid and Hanson, 2009)

1. Vertical unsaturated flow through deep unsaturated zones equals delayed recharge (UZF internal)
2. Vertical unsaturated flow beneath streams (SFR internal) and canals (SWR)
3. Inefficient losses to percolation equals infiltration into deeper unsaturated zones and simulation of delayed recharge (Link to UZF package)
4. Inefficient losses to percolation equals instant recharge (FMP internal)
5. Runoff (by FMP or UZF) discharge into stream network (by linking FMP to SFR or UZF to SFR)
6. Drain return flows (from DRT) link from discharge into SFR from FMP or directly to SWR

Flows formulated by—

- Farm Process (FMP)
- Multi-Node Well Package (MNW1/MNW2)
- Streamflow Routing Package (SFR)
  Surface-water Routing Process (SWR)
- Unsaturated-Zone Flow Package (UZF)
- Groundwater Flow Process (GWP)
- Drain/Drain Return Flow Packages (DRN/DRT)

Figure 5.9. Interdependencies of flows in a hydrologic system simulated by the MF-OWHM (modified from Schmid and Hanson, 2009b).

Runoff related to irrigation, R_i, is calculated as follows:

$$R_i = \left(I - ET_{i-act}\right) f_r^{I-loss} \tag{5.15}$$

where

ET_{p-act} and ET_{i-act} are the parts of actual ET, ET_{c-act}, supplied by precipitation or irrigation [LT^{-1}], respectively, and

f_r^{P-loss} and f_r^{I-loss} are fractions of the respective crop's inefficient losses from precipitation or irrigation that go to runoff, given as time-series data.

Losses from precipitation or irrigation that do not contribute to runoff are assumed to become deep percolation.

The FMP assumes that all precipitation or irrigation is initially available for crop evapotranspiration before runoff in the form of crop inefficient losses. That is, runoff is generated as part of these inefficient losses only after ET_{c-act} is calculated. Fractions of the inefficient losses to surface-water runoff are specified for each virtual crop type for each stress period. Surface-water runoff is assumed to depend on irrigation method, which, in turn, may depend, in part, on the crop type. Because rainfall intensity and irrigation application methods further influence runoff, the FMP requires input of two separate fractions for inefficient losses to surface-water runoff: one related to precipitation, f_r^{P-loss}, and another one related to irrigation, f_r^{I-loss}, which may be omitted or set to placeholder zero values for non-irrigated crop types, such as natural vegetation. Instead of specifying f_r^{P-loss} and f_r^{I-loss} manually, the FMP also provides an alternative option to calculate these fractions on the basis of local (cell-by-cell) slope of the surface.

In the FMP, irrigation return flow is routed to any user-specified stream reach (called semi-routed return flow) or, alternatively, the FMP can search for a stream reach nearest to lowest elevation of the farm, where return flow is assumed to gather (called fully routed return flow). The stream network is simulated by a linkage between the FMP and the SFR (streamflow routing package of MODFLOW). Re-use of irrigation return flow is not explicitly modeled in the FMP. The user has the option to return the entire runoff from precipitation and irrigation losses to points of diversion either to the farm, from which the runoff originates, or to a downstream farm. In this way, runoff becomes available for diversions and can be re-used.

Deep Percolation

The FMP computes deep percolation, DP (eqs. 5.1 and 5.2), as the sum of deep percolation of precipitation and irrigation to below the root zone. It is the user-specified proportion of losses of precipitation and irrigation that are not consumptively used by plants and not lost to surface-water runoff (as explained in the "Runoff" section). Deep percolation is calculated as follows:

$$DP = \left(P - ET_{p-act}\right)\left(1 - f_r^{P-loss}\right) + \left(I - ET_{i-act}\right)\left(1 - f_r^{I-loss}\right) \tag{5.16}$$

This approach assumes no delay between percolation past the base of the root zone and recharge to the uppermost active aquifer (fig. 5.9, item 4).

For deep vadose zones that extend far below the root zone, the FMP also offers the calculation of a delay of DP by using a linkage between the FMP and the unsaturated-zone flow (UZF) package (Niswonger and others, 2006), passing the FMP-generated DP below the root zone to the UZF package as quasi-"applied infiltration" to the vadose zone below the root zone (fig. 5.9, item 3). This linkage allows the coupling of some farms between the FMP and UZF, but still retains the option of "stand-alone" UZF infiltration areas. The FMP-calculated percolation is passed on and partitioned by the linked UZF package to different components, including various runoff components, actual infiltration into the deeper vadose zone under farms, unsaturated-zone storage under farms, and recharge. In the UZF package, vertically downward flow through the unsaturated zone is simulated by a kinematic wave approximation to Richards' equation, which, in turn, is solved by the method of characteristics (Smith, 1993; fig. 5.9, items 1, 2, and 3). The approach assumes that unsaturated flow responds to gravity potential gradients only and ignores negative potential gradients; the approach further assumes uniform hydraulic properties in the unsaturated zone for each vertical column of model cells. The Brooks-Corey function is used to define the relation between unsaturated hydraulic conductivity and water content (Brooks and Corey, 1966).

Water Demand and Supply

The FMP calculates the irrigation water demand and attempts to match it with available supply components, which may be constrained by natural, legal or policy, or structural constraints (fig. 5.10). This is the core of the FMP's demand-driven and supply-constrained system. Computed or estimated water demand and available water supplies do not always balance, however. In dry years, actual deliveries or water rights allocations may not match water demands. Conversely, in wet years, actual deliveries may exceed what is needed for irrigation to protect surface-water rights, sustain flushing of saline soils, or to increase deep percolation and associated groundwater storage. Non-irrigated areas with natural vegetation rely solely on precipitation, which can be more or less than the actual plant evapotranspiration requirement. The FMP is designed to address (1) most of the issues regarding the computation of water demand, (2) the configuration of different sources of water supply to meet this demand, and (3) the computation of the hydrologic effects of unbalanced demand and supply. The next section discusses features representing total water demand, water-supply components, and the balance between supply-and-demand components.

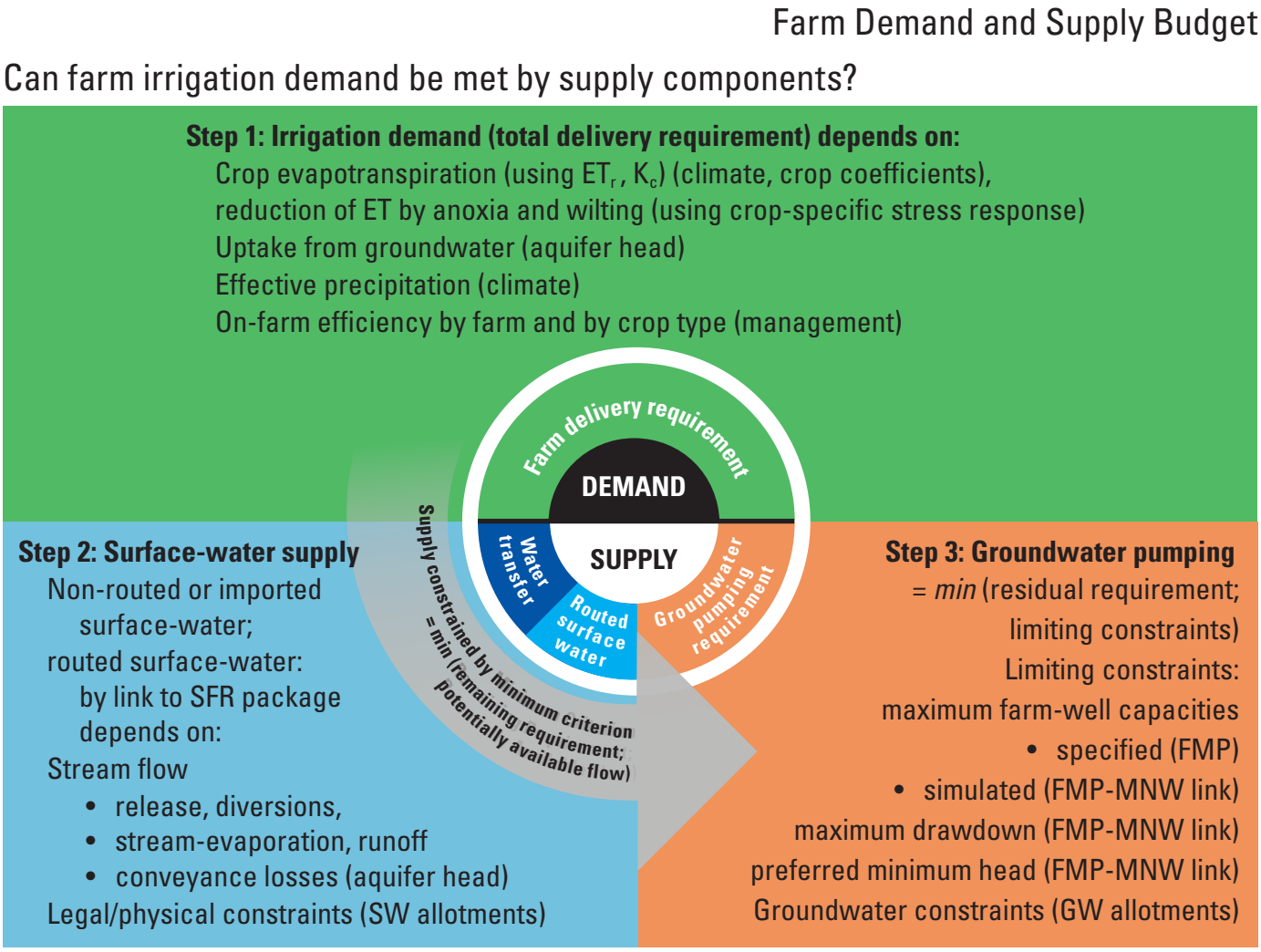


Figure 5.10. Demand-driven and supply-constrained system of the FMP (farm process) water demand and supply components (modified from Schmid and others, 2006b). [min, minimize; NRD, non-routed deliveries; SFR, streamflow routing package; SRD, semi-routed deliveries; RD, fully routed deliveries; GW, groundwater]

Total Water Demand

In addition to irrigation water demand, as discussed in the "Irrigation Water" section, the FMP also allows non-irrigation demand, such as urban, municipal, and industrial, to contribute to the total requested demand to be met with surface-water and groundwater supply components. In the FMP, other non-crop urban demand can be factored into the data input for non-routed deliveries. Inputs to the FMP for non-routed deliveries are computed by subtracting municipal and industrial demand needs from non-routed external water transfers (assuming that they are known). These demands may exceed the water transfers available, resulting in a negative non-routed delivery. This indicates a shortage that must be satisfied, along with water demand for urban irrigated landscapes, by routed surface water and pumped groundwater.

Another non-irrigation demand can target percolation rates for ponds or well injection rates for managed aquifer recharge (MAR) and Aquifer Storage and Recovery (ASR). This demand can be simulated as a "design" irrigation demand of a "virtual zero-transpiration crop" that is based on the known maximum infiltration rate of the ASR pond or injection wells (Hanson and others, 2008, 2014b, and 2015). These and other non-routed deliveries are accounted for separately for each farm.

Water-Supply Components

In the FMP, the initial sources of water to meet the total water demand come from precipitation and root uptake of groundwater. Because of the steady-state assumption, supply does not come from changes in soil moisture stored in the root zone. Any unmet demand is satisfied, in sequence of priority, by imported water, stream diversions, and groundwater pumping (fig. 5.10). Imported water from outside the model domain is simulated as non-routed deliveries (NRD). Multiple types of the NRDs can be specified (for example, for interstate water transfers, water from wastewater-treatment plants, or well fields delivering stored groundwater through ASR operations), which are linked to the water-balance subregions they serve. The NRDs must include information about maximum volumes, sequence of ranking in which each type is used to meet irrigation demand, and whether to route potential excess from the NRDs to the stream network or to injection wells.

Any demand not met by the NRDs is served by deliveries that originate from stream diversions in the model domain. These are simulated as semi- or fully-routed deliveries (SRD or RD; fig. 5.10). Locations in the stream network where the SRDs are withdrawn are specified by the user along modeled stream reaches. The RDs are automatically diverted to a farm from the uppermost stream reach, either from segments that are used for diversion only or from any type of river segment that is in the domain of the respective farm. Natural, legal, or structural constraints can pose limitations on surface water. The SRDs or RDs are limited by the available stream flow or by legal constraints such as equal or prior appropriation allotments (fig. 5.10). Specification of diversion rates for a streamflow diversion from a main-stem river to a diversion segment are possible through data input to the SFR package. These "river-to-canal" diversions can be specified along a segment near or farther upstream from the segment that contains the SRDs or RDs as "canal-to-farm" diversions. Subject to any canal water losses or gains between the "river-to-canal" and "canal-to-farm" diversion points, this mechanism can be used to construct a demand-driven and supply-constrained surface-water delivery system that is implicitly linked to the potential amount of water simulated to be conveyed in the stream to the point of diversion and delivery.

Any residual delivery requirement not met by NRDs, SRDs, nor RDs is supplied by the fourth source of water, groundwater pumping from farm wells at user-specified cells. Wells are associated with the WBS they serve through a unit-identification code; therefore, each well can be either inside or outside each WBS. The groundwater pumping in each WBS equals the sum of either the residual delivery requirement or the cumulative maximum pumping capacity, whichever is less (fig. 5.10). For single-aquifer wells (fig. 5.9), the maximum pumping capacity is specified, but for the FMP multi-aquifer wells linked to the MNW (multi-node well) package (fig. 9, Halford and Hanson, 2002; Konikow and others, 2009), a maximum capacity is simulated. Multi-node wells can represent non-uniform wellbore inflow from vertical, fully or partially penetrating, multi-aquifer wells.

The inflow is both head- and transmissivity-dependent. This allows for additional wellbore flow between model layers or aquifers, typical of large irrigation-supply wells. The wellbore flow can occur during both periods of pumping and no pumping. The MNW2 improvements from MNW1 include partial penetration, multiple MNW wells in one model cell, and better identification of FMP-MNW linked wells (Hanson and others, 2014a). The FMP-MNW linkage also allows for additional constraints on farm-well pumpage using the head and drawdown features of the MNW package (fig. 5.10), which simulate the loss in pressure as water flows through the aquifer toward the well. Pumpage rates will depend on the radius of each multi-node farm well, the aquifer properties. and the hydrologic head. Therefore, they may be less than the user specified flow rates.

The WELLFIELD option in the FMP allows for a redistribution of stored groundwater from recovery wells or well fields used for MAR and ASR to receiving farms; the redistribution amounts are related to the cumulative demand of these farms. This pumpage is, in the case of the recovery wells of an ASR, recovered and reused water that originally was diverted from the stream network and percolated to groundwater by the ASR pond. The pumpage of any well field is distributed as simulated NRDs to receiving farms and given priority over local farm-well pumpage. Farms can receive simulated NRDs from any number of well fields, with fields sequentially supplying demand according to a user-specified priority designated in the input data (Schmid and Hanson, 2009b). Whenever one well field's pumpage is limited by rate, head, or drawdown constraints, the well field next in priority then contributes to the simulated demand of the NRDs. These ASR and multi-aquifer farm-well features are unique to the FMP and provide more potential linkages to the use and reuse of water resources in the framework of a supply-constrained and demand-driven water balance (Hanson and others, 2008).

Balance between Water Supply and Demand

The FMP does not simulate changes in soil-moisture storage; therefore, no depletion in soil moisture can contribute to satisfying the crop water demand. It is assumed that for most modeling applications and typical managed irrigation practices, this distinction has minor consequences because most irrigation is provided on a regular basis during the growing season. Hence, an imbalance between irrigation demand and irrigation supply components is not buffered by a soil-water reservoir. This becomes apparent at the first iteration of an FMP time step. In case of supply deficit, the FMP requires that at each time step, a solution to a deficit problem must be found according to the user's choice. The user has the choice to assume that (1) the necessary water supply must be guaranteed and that the deficiency is made up by alternative sources external to the model domain; (2) the available supply is used, but that after improving the efficiency and minimizing inefficient losses, the actual evapotranspiration is reduced, indicating that crop yields are negatively affected by the deficit irrigation; or (3) profitability of a particular cropping pattern on a farm must be guaranteed by optimizing the profit, subject to crop market benefits and water costs associated with a particular water source. The latter option may lead to a reduction of each cell's cropped area. Once the FMP detects a deficiency at the first iteration of a time step, the user-selected response to the deficit problem is dynamically applied in the succeeding iterations of the same time step. These features of the FMP provide a broad context for responding to deficits, considering all supply and demand components and spanning all the farms in a watershed or groundwater basin.

In the FMP, the total water supply is made available to meet crop-irrigation requirements and to account for inefficient losses. Water supply in excess of the crop water demand is converted to irrigation return flow and deep percolation using equations 5.15 and 5.16, respectively. Water supply can only exceed the total demand for excess imported water (NRDs) by user specification to either discharge the excess back to the conveyance network or to injection wells.

References Cited

Allen, R.G., Pereira, L.S., Raes, D., and Smith, M., 1998, Crop evapotranspiration—Guidelines for computing crop water requirements: Rome, Italy, Food and Agriculture Organization of the United Nations, Irrigation and Drainage Paper 56, 300 p., http://www.fao.org/docrep/X0490E/X0490E00.htm.

Brooks, R.H., and Corey, A.T., 1966, Properties of porous media affecting fluid flow: American Society of Civil Engineers, Journal of Irrigation and Drainage, v. 101, p. 85–92.

Colorado's Decision Support Systems, 1995, Colorado's Decision Support Systems homepage: Colorado's Decision Support Systems Web page, https://www.colorado.gov/pacific/cdss.

Dogrul, E.C., Schmid, W., Hanson, R.T., Kadir, T.N., and Chung, F.I., 2011, Integrated water flow model and Modflow-Farm Process—A comparison of theory, approaches, and features of two integrated hydrologic models: California Department of Water Resources Technical Information Record, TIR–1, 70 p.

Drew, M.C., 1997, Oxygen deficiency and root metabolism—Injury and acclimation under hypoxia and anoxia: Annual Review of Plant Physiology and Plant Molecular Biology, v. 48, p. 223–250.

European Environment Agency, 2004, European Environment Agency multilingual environmental glossary: European Environment Agency Web page, https://www.eea.europa.eu/help/glossary/eea-glossary.

Feddes, R.A., Bresler, E., and Neuman, S.P., 1974, Field test of a modified numerical model for water uptake by root systems: Water Resources Research, v. 10, no. 6, p. 1199–1206.

Gelt, J., Henderson, J., Seesholes, K., Tellman, B., Woodard, G., Carpenter, G., Hudson, C., and Sherif, S., 1999, Water in the Tucson area—Seeking sustainability: Tucson, Ariz., Water Resources Research Center, College of Agriculture and Life Sciences, The University of Arizona, no. 20, p. 1–55, https://wrrc.arizona.edu/publications/water-tucson-area-seeking-sustainability.

Guan, H., and Wilson, J.L., 2009, A hybrid dual-source model for potential evaporation and transpiration partitioning: Journal of Hydrology, v. 377, nos. 3–4, p. 405–416, https://doi.org/10.1016/j.jhydrol.2009.08.037.

Halford, K.J., and Hanson, R.T., 2002, User guide for the drawdown-limited, multi-node well (MNW) package for the U.S. Geological Survey's modular three-dimensional finite-difference ground-water flow model, versions MODFLOW-96 and MODFLOW-2000: U.S. Geological Survey Open-File Report 2002–293, 33 p., https://doi.org/10.3133/ofr02293.

Hanson, R.T., Schmid, W., and Leake, S.A., 2008, Assessment of conjunctive use water-supply components using linked packages and processes in MODFLOW: Golden, Colo., Modflow and More—Ground Water and Public Policy, May 18–21, 2008, p. 5.

Hanson, R.T., Schmid, W., Faunt, C.C., Lear, J., and Lockwood, B., 2014a, Integrated hydrologic model of Pajaro Valley, Santa Cruz and Monterey Counties, California: U.S. Geological Survey Scientific Investigations Report 2014–5111, 166 p., https://doi.org/10.3133/sir20145111.

Hanson, R.T., Lockwood, B., and Schmid, W., 2014b, Analysis of projected water availability with current basin management plan, Pajaro Valley, California: Journal of Hydrology, v. 519, p. 131–147, https://ca.water.usgs.gov/pubs/2014/HansonLockwoodSchmid2014.pdf.

Hanson, R.T., Flint, L.E., Faunt, C.C., Gibbs, D.R., and Schmid, W., 2015, Hydrologic models and analysis of water availability in Cuyama Valley, California (ver. 1.1, May 2015): U.S. Geological Survey Scientific Investigations Report 2014–5150, 150 p., https://pubs.usgs.gov/sir/2014/5150/pdf/sir2014-5150.pdf.

Konikow, L.F., Hornberger, G.Z., Halford, K.J., and Hanson, R.T., 2009, Revised multi-node well (MNW2) package for MODFLOW ground-water flow model: U.S. Geological Survey Techniques and Methods 6–A30, 67 p., https://pubs.usgs.gov/tm/tm6a30/.

Kroes, J.G., and van Dam, J.C., eds., 2003, Reference manual SWAP version 3.0.4: Wageningen, The Netherlands, Wageningen, Alterra, Green World Research, Alterra-Report 773, 211 p.

Kustas, W.P., and Norman, J.M., 1997, A two-source approach for estimating turbulent fluxes using multiple angle thermal infrared observations: Water Resources Research, v. 33, no. 6, p. 1495–1508.

Mathur, S., and Rao, S., 1999, Modeling water uptake by plant roots: Journal of Irrigation and Drainage Engineering, v. 125, no. 3, p. 159–165, https://doi.org/10.1061/(ASCE)0733-9437(1999)125:3(159).

Niswonger, R.G., Prudic, D.E., and Regan, R.S., 2006, Documentation of the unsaturated-zone flow (UZF1) package for modeling unsaturated flow between the land surface and the water table with MODFLOW-2005: U.S. Geological Survey Techniques and Methods 6–A19, 62 p., https://pubs.usgs.gov/tm/2006/tm6a19/pdf/tm6a19.pdf.

Prasad, R., 1988, A linear root water uptake model: Journal of Hydrology, v. 99, nos. 3–4, p. 297–306, https://www.researchgate.net/publication/245098790_A_linear_root_water_uptake_model.

Robinson, T.W., 1958, Phreatophytes: U.S. Geological Survey Water-Supply Paper 1423, 84 p., https://pubs.usgs.gov/wsp/1423/report.pdf.

Schmid, W., 2004, A farm package for MODFLOW-2000—Simulation of irrigation demand and conjunctively managed surface-water and ground-water supply: Tucson, Ariz., University of Arizona, Department of Hydrology and Water Resources, Ph.D. dissertation, 278 p.

Schmid, W., and Hanson, R.T., 2009a, Appendix 1: Supplemental information—Modifications to MODFLOW-2000 packages and processes *in* Faunt, C.C., ed., Groundwater availability of the Central Valley aquifer of California: U.S. Geological Survey Professional Paper 1766, 225 p., https://pubs.usgs.gov/pp/1766/.

Schmid, W., and Hanson, R.T., 2009b, The farm process version 2 (FMP2) for MODFLOW-2005—Modifications and upgrades to FMP1: U.S. Geological Survey Techniques and Methods 6–A32, 102 p., https://pubs.er.usgs.gov/publication/tm6A32.

Schmid, W., and Hanson, R.T., 2009c, Current concepts of crop evapotranspiration and crop irrigation requirements in the farm process for MODFLOW (MF-FMP) and future concept expansions: University of California, Davis, Calif., Department of Land, Air, and Water Resources Presentation, May 28, 2009.

Schmid, W., Hanson, R.T., Maddock, T., III, and Leake, S.A., 2006a, User guide for the farm process (FMP1) for the U.S. Geological Survey's modular three-dimensional finite-difference ground-water flow model, MODFLOW-2000: U.S. Geological Survey Techniques and Methods 6–A17, 127 p., https://water.usgs.gov/nrp/gwsoftware/mf2005_fmp/tm6A17.pdf.

Schmid, W., Hanson, R.T., and Maddock, T., III, 2006b, Overview and advancements of the farm process for MODFLOW-2000: Golden, Colo., MODFLOW and More—Managing Ground-Water Systems Conference, Colorado School of Mines, International Ground-Water Modeling Center, p. 23–27, May 21–24, 2006, oral presentation.

Schmid, W., King, J.P., and Maddock, T.M., III, 2009, Conjunctive surface-water/ground-water model in the southern Rincon Valley using MODFLOW-2005 with the farm process: Las Cruces, N. Mex., New Mexico Water Resources Research Institute Completion Report no. 350.

Schmid, W., Hanson, R.T., Leake, S.A., Hughes, J.D., and Niswonger, R.G., 2014, Feedback of land subsidence on the movement and conjunctive use of water resources: Environmental Modelling and Software, v. 62, p. 253–270.

Shuttleworth, W.J., and Wallace, J.S., 1985, Evaporation from sparse crops—An energy combination theory: Quarterly Journal of the Royal Meteorological Society, v. 111, no. 469, p. 839–855, https://rmets.onlinelibrary.wiley.com/doi/abs/10.1002/qj.49711146910.

Simunek, J., Sejna, M., and van Genuchten, M.T., 1999, HYDRUS-2D/MESHGEN-2D software for simulating water flow and solute transport in two-dimensional variably saturated media, version 2.0. IGWMC–TPS–53C: Golden, Colo., Colorado School of Mines, International Ground Water Modeling Center, 227 p.

Smith, M., 1993, Neural networks for statistical modeling: New York, Van Nostrand Reinhold, 235 p.

Taylor, S.A., and Ashcroft, G.L., 1972, Physical edaphology—The physics of irrigated and nonirrigated soils: San Francisco, Calif., W.H. Freeman and Co., 533 p.

Wesseling, J.G., 1991, Meerjarige simulaties van grondwateronttrekking voor verschillende bodemprofielen, gondwatertrappen en gewassen met het model SWTRE: Wageningen, The Netherlands, Winand Staring Centre, Rep. 152, 63 p.

Appendix 6. Farm Process Version 4 (FMP)

The Farm Process (FMP) underwent substantial changes with this release of MF-OWHM2. Most of the base Fortran code was modernized to FORTRAN 2003 and 2008 to simplify developing land-use simulations, allow for easier development of future input updates, and remove FMP features that were not used. This is the fourth major release of the FMP and is referenced as FMP4 to distinguish it from previous releases; however, in this appendix the acronyms FMP and FMP4 are used interchangeably. The description of numerous changes to FMP4 input initially may seem overwhelming, but this is because all FMP4 input options are presented in this appendix.

Fundamentally, the FMP has four main input parts. The first and most important part, defines the Water Balance Subregions (WBS) that are spatially defined groups of surface model cells over which supply and demand calculations are aggregated. The WBS also acts as a surface IBOUND array—the boundary variable array defined in the Basic Package (BAS)—where the FMP will only perform its calculations in areas defined by a WBS identification (ID) that is specified within the range of one to the maximum number of WBS IDs (**NWBS**). The second part defines the landscape potential consumption from evaporation and transpiration and how excess water is handled on the landscape—where the excess becomes either surface runoff or deep percolation. An overview of this potential consumption, called consumptive use (CU), is described in appendix 4. The third part describes the available supplies to meet the potential consumption—for example, groundwater supply wells or surface-water diversions—and describes the case of irrigation and associated irrigation efficiency. The fourth part applies when climate or soil data are available; it describes how precipitation, reference evapotranspiration, and soil capillary fringe depths can optionally be defined.

This appendix first discusses the block-style input used for all the FMP input. The "Block-Style Input" section provides an overview of the new input structure, an explanation of the bare minimum requirements to run a FMP4 model, and a simple example input. The handling of the FMP4 input is adaptive by utilizing the amount of information given to determine the appropriate assumptions for landscape simulation. Options not specified are set to default values for the users' convenience. If there is a combination of options that are invalid, then an error is raised explaining why the FMP cannot continue. At the end of this appendix are several example input sets of input keywords based on common FMP model setups.

Features Removed

The features that were removed from the original input structure and replaced with a simpler block input are the prior appropriation scheme for ranked appropriation by farms that represented a water-rights hierarchy of preferred deliveries; the deficit irrigation options for acreage optimization, conservation pool, water stacking; and the LGR-FMP (Local Grid Refinement-FMP) "P" flag that would initiate an automatic transfer of farm and crop properties from the LGR parent to child model. The fully routed delivery option has been removed, and it is recommended to use semi-routed deliveries (SRDs) in its place. A fully routed delivery provided an automatic search for the uppermost streamflow routing (SFR) diversion segment to remove water from, which can be directly specified as an SRD point as described in the "SURFACE_WATER Block" section. Additional features removed are the farm well parameters, the daily crop coefficient, and root-depth time series for the entire simulation—formerly set as `ICUFL=3` and `IRTFL=3`, the efficiency behavior flag (`IEBFL`) options (efficiency is now held constant until a new efficiency value is loaded), and the option to declare a Non-Irrigation season (`IROTFL>0`). Lastly, the fraction of evaporation (FEP) from precipitation input is no longer required; instead, it is automatically calculated as 1 minus the fraction of transpiration (FTR), so that FEP = 1 – FTR.

Block-Style Input

The FMP uses the new block-style input structure, defined in appendix 1, to group common input properties that are set using keywords. This input redesign is not backward compatible with the previous versions of the FMP (denoted as FMP1, FMP2, FMP3), which relied on a set structure of integer flags that dictated what features were in use, what land-use concepts to employ, and the frequency of data input. In contrast, the FMP version 4 automatically determines features in use and concepts to simulate on the basis of the amount of information provided or keywords supplied. After the keywords, specific input is loaded using the *List Array Input* (`LAI`) structure, *Universal Loader* (`ULOAD`), *Generic_Input* files, and *Generic_Output* files, which are defined in appendix 1.

Keywords that are not specified—that is, not written in the input block—will not activate the specific feature it represents, be assigned a default value, or FMP will raise an error message (if the keyword is a required input). If the keyword is required for the simulation to continue—such as **REFERENCE_ET** (ET_{ref}), which is required if **CROP_COEFFICIENT** (Kc) is specified (Kc × ET_{ref}; appendix 4)—then an error message is raised explaining why the simulation stopped and how to rectify the error. If an optional keyword is not specified, then it is either ignored or set to a default value. For example, if **PRECIPITATION** is not included, then it is assumed that the model has zero precipitation for the entire simulation. If the keyword pertains to a specific simulation concept, then the appropriate concept is selected depending on the level of input data provided. For example, if crop stress response **ROOT_PRESSURE** is not included in the input, then the FMP will use the simpler anoxia- or wilting-point groundwater-root concept, called the *linear root response* (appendixes 4 and 5). Conversely, if the **ROOT_PRESSURE** keyword is specified—indicating that stress-response root pressures are defined—then the crop model automatically uses the *analytical root response* option (appendixes 4 and 5). Previously this was done in MF-OWHM's FMP3 either by setting the input variable `ICCFL` to `1` or `3` to indicate stress-response root pressures are loaded to simulate the *Analytical root response* option or the `ICCFL` was set to `2` or `4` to indicate that the *Implicit Stress Assumption* was in use.

For additional flexibility, FMP4 allows for the mixture of some properties. For example, it is possible to have specified crops use the *Implicit Stress Assumption* and the remainder use the a*nalytical root response* option. This is done by setting to zero the four main inputs (that is, the root pressures) for crops that use the *implicit stress assumption* and by specifying a single non-zero root pressure causes the crop to use the *analytical root response.*

The block input structure supports the block specific keyword **BLOCK_INCLUDE**. Statements using this keyword function identically to C language `#include "file"` and Fortran `INCLUDE File` statements allowing the user to specify a portion of a block in a separate file that is inserted at the **BLOCK_INCLUDE** location. Consider a simple example of **BLOCK_INCLUDE** that loads five keywords (fig. 6.1). At runtime, before any keywords are processed, FMP4 inserts all **BLOCK_INCLUDE** files, strips out any comments, and adjusts all text to be left justified (fig. 6.1*C*).

The FMP4 blocks and keywords in the blocks may appear in any order. If a block is not present, or is empty, then it will automatically have all its values set to a default value or disabled. For example, if the **CLIMATE** block is not present in the input, then precipitation is set to zero and reference evapotranspiration is flagged internally as not being available for use within FMP. If a feature defined in another block relies on something in the **CLIMATE** block, then an error is raised explaining what input is necessary. If the block contains a conflicting set of options, either internally or from another block, then an error is raised explaining why the program stopped.

A

```
BEGIN GLOBAL DIMENSION
   #
   NWBS       5
   NCROP      3
   #
   BLOCK_INCLUDE ./Global_NVAR.txt
   #
   NRD_TYPES 0
END
```

B

```
# Contents of
# Global_NVAR.txt
# that is loaded with BLOCK_INCLUDE

NSOIL      1

NIRRIGATE 0

# note to self - remember this note about...
```

C

```
BEGIN GLOBAL DIMENSION
NWBS       5
NCROP      3
NSOIL      1
NIRRIGATE 0
NRD_TYPES 0
END
```

Figure 6.1. Block-style input example that uses BLOCK_INCLUDE to insert text located in an external file. *A*, Example GLOBAL DIMENSION block input that contains BLOCK_INCLUDE to load the contents of the file Global_NVAR.txt. *B*, The file Global_NVAR.txt that is inserted into the GLOBAL DIMENSION block input. *C*, The input that is read by MF-OWHM2.

Although there is no requirement for the order of the blocks, it is recommended to follow the figure 6.2 listed order in an FMP input file. The block names were selected to indicate the type of input properties that are nested in them and the common dimension the input properties have. For example, all input that reads a list of numbers in the **LAND_USE** block always reads **NCROP** numbers, where **NCROP** is the number of land-use types.

Input Style Options and Explanation of Shorthand Notation

In the FMP4, blocks are the keywords that enable FMP features or load specific input. Keywords that load additional input either specify it as a *Generic_Input*, ULOAD, LAI, load a single integer (**INT**), or floating-point number (**FLOAT**). See appendix 1 for a detailed and general description of these input utilities. An example of an **INT** is any number that does not contain a decimal place (-1, 0, 1, or 2), and a **FLOAT** is any number specified with a decimal place or in scientific and engineering notation (-1., 0.0, 1.5, 2E0, or 3.14E1). Please note that scientific and engineering notation implies that the number contains a decimal place, even when it is not specified, so the following are not **INT** numbers because they contain a decimal place: 10.0, 10., 1.E1, and 1E1. If the input expects a **FLOAT** and reads an **INT** number, then it converts it to a **FLOAT**—for example 2, 4, and 6 are read as 2.0, 4.0, and 6.0, respectively. If a keyword expects one of a potential set of additional keywords, then the additional keywords are enclosed in curly braces { } to indicate that one of the keywords within the curly braces must be selected. For example, the keyword **PRORATE_DEFICIENCY** may only be specified once and must be followed by either the word **ByDEMAND** or **ByAVERAGE**. The shorthand notation would then indicate the input as **PRORATE_DEFICIENCY** { **ByDEMAND**, **ByAVERAGE** } to indicate the input option may only be **PRORATE_DEFICIENCY ByDEMAND** or **PRORATE_DEFICIENCY ByAVERAGE**. Unless otherwise specified any input that loads a two-dimensional array with ULOAD expects the array to be the same dimension as the model grid (NROW × NCOL). If the input is loaded using LAI, then it will include the shorthand notation [S, T, A, L] to indicate the keywords it supports. These are described in detail in the "LAI [S, T, A, L] Input Format Meaning" section in appendix 1. The description is briefly reprised here in the context of the FMP with the option keywords (letters) representing the following:

S → **STATIC**	for single load of input with ULOAD
T → **TRANSIENT**	for stress period input with the *Transient File Reader* (TFR)
A → **ARRAY**	for *Array Style* input that loads a two-dimensional array
L → **LIST**	for *List Style* input that loads a record ID and one value per row of input

Block Name	Description
1) **GLOBAL DIMENSION**	Global properties used by other FMP blocks.
2) **WATER_BALANCE_SUBREGION**	Properties that pertain to defining WBS.
3) **OPTIONS**	Global modifier options (legacy FMP features).
4) OUTPUT	Ancillary output files (for example, cell-by-cell unit number).
5) **SOIL**	Soil specific properties.
6) **CLIMATE**	Climate-related properties.
7) **SURFACE_WATER**	Surface-water deliveries and runoff properties.
8) **SUPPLY_WELL**	Groundwater supply well properties.
9) **ALLOTMENT**	Apply a limit on different water supplies.
10) **LAND_USE**	Land-use (Crop) specific properties.
11) **SALINITY_FLUSH_IRRIGATION**	Additional irrigation demand for salinity leaching.

Figure 6.2. Farm Process version 4 (FMP) supported Block Style input block names and basic description.

All LAI inputs loaded with LAI will have at least one **TEMPORAL_KEY** (S and T) and at least one **INPUT_STYLE** (A and L). If a keyword does not support a specific LAI option, then it is not included in the brackets [ ]. For example, **REFERENCE_ET** may only be loaded with *Array Style* input, so the input statement is as follows:

```
REFERENCE_ET LAI[S, T, A]
```

This would indicate that the following are potential keyword options when specifying reference ET:

```
REFERENCE_ET STATIC    ARRAY   ULOAD
REFERENCE_ET TRANSIENT ARRAY   Generic_Input_OptKey
REFERENCE_ET STATIC            ULOAD
REFERENCE_ET TRANSIENT         Generic_Input_OptKey
REFERENCE_ET CONSTANT          VALUE
```

The *Generic_Input_OptKey* specifies the TFR, which would specify what input is loaded for each stress period. A TFR file must be a separate input file and cannot be specified with the **INTERNAL** keyword. The FMP interpreter assumes a **TEMPORAL_KEY** if there is only one available. For example, because the keyword **PRECIPITATION** only supports *Array Style* input the use of the keyword **ARRAY** is optional. The keyword **CONSTANT** is syntactic sugar to the specifying **REFERENCE_ET STATIC ARRAY CONSTANT** VALUE (which is one of the ULOAD keywords).

Multi-Column List Style Input

Some of the FMP4 input requires a multi-column *List Style* input structure. For example, the keyword **ROOT_PRESSURE** loads a crop ID and four root pressures. Under this situation the L part of the abbreviation is modified to LAI[S, T, A, L-K], where K indicates the number of columns of data that are loaded in addition to the record ID (integer in column 1). If a keyword is associated with multi-column *List Style* input, then it will NOT support *Array Style*. For example, **ROOT_PRESSURE** may only be loaded with *List Style* input, so the input statement is as follows:

```
ROOT_PRESSURE LAI[S, T, L-4]
```

This indicates that the following are potential keyword options when specifying root pressures:

```
ROOT_PRESSURE STATIC    LIST   ULOAD
ROOT_PRESSURE TRANSIENT LIST   Generic_Input_OptKey
ROOT_PRESSURE STATIC           ULOAD
ROOT_PRESSURE TRANSIENT        Generic_Input_OptKey
ROOT_PRESSURE CONSTANT         VALUE
```

The keyword **CONSTANT** is syntactic sugar to the specifying **ROOT_PRESSURE STATIC LIST CONSTANT** VALUE. The L-4 indicates that the root-pressure input expects four columns of data in addition to the record ID. Figure 6.3 is example input that would load root pressures for five land-use types (**NCROP** = 5). Because the keyword **ROOT_PRESSURE** only supports *List Style* input the use of the keyword **LIST** is optional. Further, the read utilities interpret the keyword **INTERNAL** to only work with **STATIC**, so specifying **STATIC** with **INTERNAL** is optional (fig. 6.3*B*).

A

```
# Expected input: ROOT_PRESSURE LAI[S, T, L-4]
# TEMPORAL_KEY  keywords are TRANSIENT and STATIC
# INPUT_STYLE       keyword is LIST
# Generic_Input keyword is "INTERNAL" to specify input location.
#
ROOT_PRESSURE STATIC  LIST  INTERNAL
# ID     PSI1    PSI2      PSI3    PSI4
  1   -0.15   -0.3    -5.45    -80    # Strawberries
  2   -0.15   -0.3    -7.25    -80    # Vineyards
  3   -0.15   -0.3   -52.1    -160    # Potatoes
  4   10.5     1.13  -1.25    -50    # Phreatophytes
  5   -0.075  -0.2    -8.25   -115    # Native classes
```

B

```
# Since input only supports "LIST" INPUT_STYLE, the keyword is optional.
# INTERNAL keyword implies STATIC since it is read only once.
#
ROOT_PRESSURE INTERNAL
# ID     PSI1    PSI2      PSI3    PSI4
  1   -0.15   -0.3    -5.45    -80    # Strawberries
  2   -0.15   -0.3    -7.25    -80    # Vineyards
  3   -0.15   -0.3   -52.1    -160    # Potatoes
  4   10.5     1.13  -1.25    -50    # Phreatophytes
  5   -0.075  -0.2    -8.25   -115    # Native classes
```

Figure 6.3. Example use of the FMP keyword ROOT_PRESSURE that uses List-Array input. *A*, Example that explicitly specifies the TEMPORAL_KEY as "STATIC" and INPUT_STYLE as "LIST"; *B*, Example that infers the TEMPORAL_KEY from the Generic_Input keyword "INTERNAL"; the INPUT_STYLE is inferred from the FMP keyword because only LIST input is supported. [LAI, list-array input with the following letters representing the supported keywords: S for STATIC, T for TRANSIENT, A for ARRAY, and L for LIST]

IXJ Style Input as Surrogate to Array-Style Input

Several of the FMP4 keywords support the *IXJ Style* input option described in appendix 1. This is an advanced input style—intentionally not included as part of the `LAI`[S, T, A, L] shorthand notation—that offers an alternative to the *Array Style* input. The *IXJ Style* input is similar to coordinate compressed array storage and is advantageous when *Array Style* input contains mostly zero values. The *IXJ Style* input also is best suited for models that specify multiple crops per model cell and have a large spatial grid—that is, NROW + NCOL >> 1—and large number of land-use types (**NCROP** >> `1`).

The *IXJ Style* input reads a set of integers (`I`), followed by a set of floating-point numbers (`X`), and finally by another set of integers (`J`). The number of integers, floating point numbers, and subsequent integers are specified in the input, which may include zero as the count. For example, there could only be the first set of integers read (`I`) and no floating-point numbers nor subsequent integers—that is, no `X` or `J` read.

If an input block has a keyword that supports the *IXJ Style* input, then an additional section is included that describes the expected *IXJ Style* input structure for the supported keywords. At the time of publication all the keywords that support the *IXJ Style* input read two or three integers followed by a single floating-point number and no subsequent integers—that is, two or three `I` inputs, one `X` input, and zero `J` input. If the keyword expects to read two `I` inputs, then the first integer represents the model row and the second integer is the model column that will be assigned the one `X` floating point number loaded—that is, one input record for the **`IXJ`** is row, column, and non-zero model property. If the keyword expects to read three `I` inputs, then the first integer represents a global property, such as land use/crop ID, and the second and third integers are the model row and column that will be assigned the one `X` floating point number loaded. Figure 6.4 contains two examples that use FMP keywords with the *Generic_Input* keyword **`INTERNAL`** to load the *IXJ Style* input.

```
# Example that loads 2 I and 1 X
# This input keyword loads REFERENCE_ET   LAI[S, T, A]
# This example loads the model row,  model column and then reference ET
#
REFERENCE_ET STATIC IXJ INTERNAL
# ROW  COL   RefET
  2    3    1.1
  5    7    0.8
  1    1    1.2
  4    1    0.9
  STOP IXJ  # Stop loading IXJ input

# Example that loads 3 I and 1 X
# This input keyword, LAND_USE_AREA_FRACTION LAI[S, T, A],
# is used when specifying the fraction of the model cell
# (ROW, COL) area that contains the land use/crop covers.
#
# When there are multiple crops per model cell allowed the array input
# must specify the area fraction as a NROW by NCOL array for each crop.
#
# IXJ input loads only the Crop ID, the model row, and model column,
# then the area fraction for the non-zero crop area fractions.
# Any crop ID, row, and column not specified is assumed to be zero
#
LAND_USE_AREA_FRACTION STATIC IXJ INTERNAL
# CropID  ROW   COL  Fraction
   1      2     3    0.90
   1      3     2    0.25
   2      2     3    0.10
   2      4     2    0.50
  STOP IXJ  # Stop loading IXJ input
```

Figure 6.4. Example IXJ Style input for the FMP Land_Use block keywords REFERENCE_ET and LAND_USE_AREA_FRACTION. [Comments are preceded with the # symbol. Both example keywords use the *Generic_Input* keyword INTERNAL to indicate that the IXJ style input is loaded on the subsequent lines.]

Loading Array Style with "INTERNAL" in the Block—Not Recommended

In this appendix, most examples make use of the keyword **INTERNAL**. This use of this keyword is to minimize the number of figures or text boxes necessary to illustrate how to load the input. When the FMP input file is loaded, MF-OWHM2 determines the longest line length in the file—after all the **BLOCK_INCLUDE** files have been inserted—and pre-allocates enough memory to parse the *Block Style* input. For models with many columns—that is, NCOL >> 1—loading *Array Style* input with **INTERNAL** can result in excessive memory allocation due the line length of the input. As a result, it is not recommended to use in a block the keyword **ARRAY** with **INTERNAL** to improve simulation speed. This pre-allocation of memory only applies to input specified directly within a block and does not include any input from external files loaded with **EXTERNAL**, **OPEN/CLOSE**, **DATAFILE**, **DATAUNIT**, or anything loaded from a TFR.

Figures 6.5, 6.6, 6.7, and 6.8 are examples of **CLIMATE** blocks that load **PRECIPITATION** `LAI[S, T, A]` to specify the amount of precipitation that falls on each model cell. The assumed model grid has NROW equal to 3 and NCOL equal to 10,000. Figure 6.5 and 6.6 are examples of what is not recommended. Because a **BLOCK_INCLUDE** directly includes the file in the block, figure 6.6*A* produces—within the context of MF-OWHM2—an identical input as figure 6.5.

Figure 6.7 is an example for loading a **STATIC ARRAY** within a block that does not require a larger memory allocation to read the input. By referencing a separate file, the *Block Style* input utility only loads the line "**PRECIPITATION STATIC ARRAY OPEN/CLOSE** `./Precip.txt`", into memory, which requires 50 characters (or 50 bytes of memory). After the *Block Style* input has parsed the keywords, then the contents of the file "`./Precip.txt`" are efficiently read.

Because a TFR is always read from an external file, it is never part of *Block Style* input's pre-allocation line. For example, figure 6.8 has the secondary keywords **TRANSIENT ARRAY** to indicate input is read by stress period with the TFR file "`./Precip_TFR.txt`". By referencing a TFR, the *Block Style* input utility only loads the line "**PRECIPITATION TRANSIENT ARRAY OPEN/CLOSE** `./Precip_TFR.txt`" into memory, which requires 57 characters (or 57 bytes of memory). After the *Block Style* input has parsed the keywords, then for each stress period the appropriate array is efficiently read from the file "`./Precip_TFR.txt`".

```
BEGIN CLIMATE
  #  Within a Block style input, ARRAY style with INTERNAL is not recommended.
  #
  #  Block style input must allocate enough memory to hold the longest line within the block.
  #
  #  The precipitation array is specified within the block by using the INTERNAL keyword,
  #  so it has to pre-allocate enough space to the line with 10,000 numbers.
  #
  #  Keyword PRECIPITATION specifies the precipitation flux over model domain
  PRECIPITATION STATIC ARRAY INTERNAL # Load 3×10000 array
  0.1  0.1  0.2  …  …  …  0.20   0.10
  0.1  0.1  0.2  …  …  …  0.20   0.10
  0.1  0.1  0.2  …  …  …  0.20   0.10
  #
END CLIMATE
```

Figure 6.5. Example loading a 3-by-10000 array with INTERNAL in the FMP Climate block keyword PRECIPITATION. The text within the Block Style input is preloaded into computer memory (RAM), so it is not recommended to specify large arrays with INTERNAL within an input block. An alternative is to change INTERNAL to OPEN/CLOSE and move the 3-by-10000 array to a separate file. [… is a place holder for the input numbers.]

A

```
BEGIN CLIMATE
  #  Within a Block style input, ARRAY  style with INTERNAL  is not recommended.
  #
  #  Block style input must allocate enough memory to hold the longest line within the block.
  #
  #  Since BLOCK_INCLUDE inserts text from Precip_Include.txt into the block and
  #  the precipitation array is specified using the INTERNAL keyword,
  #  the read utility has to pre-allocate enough space to the line with 10,000 numbers.
  #
  #  Keyword PRECIPITATION specifies the precipitation flux over model domain
  #  Array is within block indirectly through BLOCK_INCLUDE.
  #
  #  "BLOCK_INCLUDE ./Precip_Include.txt" is replaced by the
  #  text stored in the file Precip_Include.txt
  #
  BLOCK_INCLUDE ./Precip_Include.txt
  #
END CLIMATE
```

B

```
  # File: Precip_Include.txt
  # Precipitation flux for a 3 by 10000 grid
  #
  PRECIPITATION STATIC ARRAY INTERNAL
  0.1  0.1  0.2  …  …  …  0.20   0.10
  0.1  0.1  0.2  …  …  …  0.20   0.10
  0.1  0.1  0.2  …  …  …  0.20   0.10
```

Figure 6.6. Example loading a 3-by-10000 array with INTERNAL in the FMP Climate block keyword PRECIPITATION. The array is indirectly inserted into the Climate block by the BLOCK_INCLUDE keyword. The text within the Block Style input is preloaded into computer memory (RAM), so it is not recommended to specify large arrays with INTERNAL within an input block. An alternative is to change INTERNAL to OPEN/CLOSE and move the 3-by-10000 array to a separate file. *A*, FMP Climate block that contains a BLOCK_INCLUDE. *B*, The file, Precip_Include.txt, that is inserted into the Climate block. [… is a place holder for the input numbers.]

A

```
BEGIN CLIMATE
  #  Block style input must allocate enough memory to hold the longest line within the block.
  #
  #  This example only has to pre-allocate enough memory to store the text
  #  "PRECIPITATION STATIC ARRAY OPEN/CLOSE ./Precip.txt"
  #   which is 50 characters long (or 50 bytes)
  #
  #  Keyword PRECIPITATION specifies the precipitation flux over model domain
  #
  PRECIPITATION STATIC ARRAY OPEN/CLOSE ./Precip.txt
  #
END CLIMATE
```

B

```
  # File: Precip.txt for
  # Precipitation flux for a 3 by 10000 grid
  0.1  0.1  0.2  …  …  …  0.20   0.10
  0.1  0.1  0.2  …  …  …  0.20   0.10
  0.1  0.1  0.2  …  …  …  0.20   0.10
```

Figure 6.7. Example loading a 3-by-10000 array with OPEN/CLOSE specified in the FMP Climate block keyword PRECIPITATION. Only 1 line is read by the Block Style input utility, which is 50 characters (50 bytes) long. The precipitation array is stored in a separate file that is read after the Block Style input utility has processed all the keywords. *A*, FMP Climate block. *B*, The file, Precip.txt, that contains the 3-by-1000 precipitation array. [… is a place holder for the input numbers.]

A

```
BEGIN CLIMATE
  #  Block style input must allocate enough memory to hold the longest line within the block.
  #
  #  This example only has to pre-allocate enough memory to store the text
  #  "PRECIPITATION TRANSIENT ARRAY OPEN/CLOSE ./Precip_TFR.txt"
  #   which is 57 characters long (or 57 bytes)
  #
  #  The word "TRANSIENT" indicates that Precip_TFR.txt is a Transient File Reader
  #  and specifes the precipitation flux for each stress period.
  #
  #  Keyword PRECIPITATION specifies the precipitation flux over model domain
  #
  PRECIPITATION TRANSIENT ARRAY OPEN/CLOSE ./Precip_TFR.txt
  #
END CLIMATE
```

B

```
  # File: Precip_TFR.txt
  #
  # File is a Transient File Reader (TFR) that reads a set of precipitation
  # fluxes for a 3 by 10000 model grid for each stress period (SP).
  #
  INTERNAL # SP 1
  0.1   0.1   0.2   …   …   …   0.20   0.10
  0.1   0.1   0.2   …   …   …   0.20   0.10
  0.1   0.1   0.2   …   …   …   0.20   0.10
  INTERNAL # SP 2
  0.1   0.1   0.2   …   …   …   0.20   0.10
  0.1   0.1   0.2   …   …   …   0.20   0.10
  0.1   0.1   0.2   …   …   …   0.20   0.10
  INTERNAL # SP 3
  0.1   0.1   0.2   …   …   …   0.20   0.10
  0.1   0.1   0.2   …   …   …   0.20   0.10
  0.1   0.1   0.2   …   …   …   0.20   0.10
```

Figure 6.8. Example loading a 3-by-10000 array with a *Transient File Reader* (TFR) specified in the FMP Climate block keyword PRECIPITATION. Only 1 line is read by the Block Style input utility, which is 57 characters (57 bytes) long. The precipitation arrays are stored the TFR and read with the keyword INTERNAL. *A*, FMP Climate block. *B*, A TFR file, Precip_TFR.txt, that specifies the precipitation flux for three stress periods. [… is a place holder for the input numbers.]

Advanced Scale Factors and SFAC

FLOAT input that loads with ULOAD and LAI supports scale factors as described in appendix 1. The scale factors can be used as part of calibration, to convert units, or for scenario evaluations. At a minimum all ULOAD and LAI **FLOAT** input can apply *Generic_Input* scale factors, using **SF** SCALE syntax. If the *Generic_Input* is a TFR, then scale factors can be applied to each input loaded in the TFR. In the "Multi-Column List Style Input" section's **ROOT_PRESSURE** example (fig. 6.3*B*), the data were loaded using the *Generic_Input* keyword **INTERNAL** to load **FLOAT** data. Figure 6.9*A* recasts this example to use the *Generic_Input* post-keyword "**SF** SCALE" to multiply the **ROOT_PRESSURE**s by **0.3048**. This would be the equivalent to the input presented in figure 6.9*C*.

When ULOAD loads **FLOAT** input, it also optionally supports the **SFAC** keyword (appendix 1). All FMP **FLOAT** input keywords support rescaling using a single scale factor, and FMP applies it to all the input—note the similarity to the *Generic_Input* scale factor **SF** SCALE. Figure 6.9*B* shows the same **0.3048** scaling as in figure 6.9*A* but uses the keyword **SFAC** instead.

Advanced **SFAC** scaling can include dimensional keywords (**DIMKEY**s) that indicate the number of scale factors read and how they are applied. The **SFAC** **DIMKEY**s supported in FMP and their associated dimension (the dimensions are defined in the "Global Dimension Block" section) are presented in figure 6.10.

A

```
  #  FMP LAND_USE Block keyword that specifies with List-Array Input (LAI)
  #  four root pressures (PSI1, PSI2, PSI3, PSI4) that represent
  #  the upper limit pressure head at which the root uptake becomes zero
  #     as a result of anoxia, then the two pressure heads that represent
  #  the range of optimal root uptake, and then
  #  a lower limit pressure head that results in wilting (zero root uptake), respectively.
  #
  #  LAI uses GENERIC_INPUT to specify the input location of the input.
  #  The GENERIC_INPUT keyword used is INTERNAL to indicate input is on
  #     subsequent lines. GENERIC_INPUT, also supports the post keyword SF
  #     to indicate a global scale factor is applied.
  #
  #  Note that the use of INTERNAL with LAI implies the STATIC keyword and
  #  List-Style input is the only input style supported, so the LIST keyword is optional.
  #
  ROOT_PRESSURE INTERNAL SF 0.3048   # Convert PSI from feet to meters
# ID    PSI1    PSI2      PSI3    PSI4
  1   -0.15   -0.3    -5.45    -80    # Strawberries
  2   -0.15   -0.3    -7.25    -80    # Vineyards
  3   -0.15   -0.3   -52.1    -160    # Potatoes
  4   10.5     1.13   -1.25    -50    # Phreatophytes
  5   -0.075  -0.2    -8.25    -115   # Native classes
```

Figure 6.9. Examples of *A*, use of the Generic_Input post-keyword "SF SCALE" to apply a scale factor for the FMP keyword ROOT_PRESSURE. *B*, Equivalent ROOT_PRESSURE input that uses the advanced scale factor keyword **SFAC**. *C*, Equivalent ROOT_PRESSURE input without a scale factor.

B

```
  #  FMP keywords that load float input check for the SFAC keyword
  #      to load advanced scale factors. SFAC is defined before loading the actual input.
  #
  #  This example uses SFAC to indicate the input is multiplied by 0.3048
  #      to convert the input from feet to meters.
  #
  ROOT_PRESSURE INTERNAL
  SFAC 0.3048                          # Convert PSI from feet to meters
# ID     PSI1    PSI2       PSI3     PSI4
  1   -0.15   -0.3    -5.45    -80    # Strawberries
  2   -0.15   -0.3    -7.25    -80    # Vineyards
  3   -0.15   -0.3   -52.1    -160    # Potatoes
  4   10.5     1.13   -1.25    -50    # Phreatophytes
  5   -0.075  -0.2    -8.25    -115   # Native classes
```

C

```
  ROOT_PRESSURE INTERNAL
# ID     PSI1        PSI2         PSI3        PSI4
  1   -0.046   -0.091    -1.661   -24.384    # Strawberries
  2   -0.046   -0.091    -2.210   -24.384    # Vineyards
  3   -0.046   -0.091   -15.880   -48.768    # Potatoes
  4    3.200    0.344    -0.381   -15.240    # Phreatophytes
  5   -0.023   -0.061    -2.515   -35.052    # Native classes
```

Figure 6.9. —Continued

DIMKEY	Dimension
ByWBS	NWBS
ByCrop	NCROP
ByIrrigate	NIRRIGATE
BySoil	NSOIL
BySource	4
ByNRD	NRD_TYPES

Figure 6.10. FMP supported secondary dimension keyword, **DIMKEY**, for advanced scale factors, **SFAC**, that indicate the number of scale factors that are read and how they are applied. The dimension column references FMP Global Dimension block keywords. [NWBS is the number of water balance subregions; NCROP is the number of land use types simulated; NIRRIGATE is the number of irrigation types; NSOIL is the number of soil types; NRD_TYPES are the number of non-routed delivery types that are defined.]

The keywords in the figure 6.10 dimension column are defined in the **GLOBAL DIMENSION** block and indicate the number of scale factors loaded. For example, **SFAC ByCrop** would load **NCROP** scale factors with ULOAD. The first of the loaded scale factors would be applied to Crop 1, the second to Crop 2, and so forth until **NCROP**. The **BySoil DIMKEY** is only available in the **SOIL** block keyword **CAPILLARY_FRINGE** and is applied on the basis of the **SOIL_ID**, which is from 1 to **NSOIL**. The **ByNRD DIMKEY** is only available in the **SURFACE_WATER** block keyword **NON_ROUTED_DELIVERY** and is only applied to the volume of water delivered for each of the **NRD_TYPES**. The **BySource DIMKEY** is only used in the **SALINITY_FLUSH_IRRIGATION** block keyword **WBS_SUPPLY_SALT_CONCENTRATION**, which loads the salt concentration of non-routed deliveries, SRDs, groundwater pumping, and external water (EXT) sources available for irrigation, which are where the four scale factors are applied, respectively.

Figure 6.11 reuses the previous **ROOT_PRESSURE** example (fig. 6.9), applying **SFAC** with a **DIMKEY**. Figure 6.11*A* uses the **DIMKEY** "**ByCrop**" to read **NCROP** scale factors and apply them to each crop. That is, for crops 1, 3, and 5, root-pressure inputs are multiplied by 3.1, and those for crops 2 and 4 are multiplied by 1.0. Figure 6.11*B* shows scaling based on the location of the WBS by using the **DIMKEY** "**ByWBS**" to read **NWBS** scale factors—**NWBS** is the maximum number of the WBS in a simulation. In this example, any land use that is in the WBS 1 has its **ROOT_PRESSURE**s multiplied by 2.0; those in the WBS 2 are multiplied by 4.0; and lastly, those in the WBS 3 are multiplied by 6.0. In this appendix, there are a set of subsections titled "Supported **SFAC DIMKEY**" that indicate the supported **DIMKEY**s for specific FMP input keywords.

A

```
  #  Example use of the advanced scale factor SFAC that reads the dimension key (DIMKEY) "ByCrop"
  #
  #  ByCrop indicates that NCROP scale factors are read and applied such that
  #     the first scale factor is applied to the Crop 1,
  #     the second scale factor is applied to the Crop 2, and so forth until Crop NCROP.
  #
  #  This example assumes NCROP = 5 and SFAC uses an Implied Internal to read the 5 scale factors.
  #  Root pressures of  Crops 1, 3, and 5 are multiplied by 3.1 and Crops 2 and 4 are multiplied by 1.
  #
  ROOT_PRESSURE INTERNAL
  SFAC ByCrop  3.1  1.0  3.1  1.0  3.1
# ID     PSI1    PSI2      PSI3    PSI4
  1   -0.15   -0.3    -5.45    -80    # Strawberries
  2   -0.15   -0.3    -7.25    -80    # Vineyards
  3   -0.15   -0.3   -52.1    -160    # Potatoes
  4   10.5     1.13   -1.25    -50    # Phreatophytes
  5   -0.075  -0.2    -8.25    -115   # Native classes
```

B

```
  #  NWBS is the total number of water balance subregions (WBSs)
  #
  #  ByWBS indicates that NWBS scale factors are read and applied such that
  #     the first scale factor is applied to crops that are in WBS 1,
  #     the second scale factor is applied to crops that are in WBS 2, and so forth until WBS NWBS.
  #
  #  This example assumes NWBS = 3 and SFAC uses an Implied Internal to read the 3 scale factors.
  #
  ROOT_PRESSURE INTERNAL
  SFAC ByWBS  2.0  4.0  6.0
# ID     PSI1    PSI2      PSI3    PSI4
  1   -0.15   -0.3    -5.45    -80    # Strawberries
  2   -0.15   -0.3    -7.25    -80    # Vineyards
  3   -0.15   -0.3   -52.1    -160    # Potatoes
  4   10.5     1.13   -1.25    -50    # Phreatophytes
  5   -0.075  -0.2    -8.25    -115   # Native classes
```

Figure 6.11. Examples using advanced scale factors (**SFAC**) with a dimension keyword (**DIMKEY**). *A*, Example using the ByCrop **DIMKEY** to read five scale factors and apply them by crop type. *B*, Example using the ByWBS **DIMKEY** to read three scale factors and apply them by Water Balance Subregion (WBS).

Global Dimension Block

The **GLOBAL DIMENSION** block is always required as part of the FMP input. As the block name implies, it specifies the properties necessary to set dimensions for all the remaining features in the FMP. The only required inputs for the **GLOBAL DIMENSION** are the number of WBSs, land-use types (Crops), soil types, irrigation types, and Non-Routed Deliveries (NRDs) types. Figure 6.12 describes the required keywords for the **GLOBAL DIMENSION** block. These dimensions specify the number of values read during a *List Style* input. For example, the **WATER_BALANCE_SUBREGION** block loads **NWBS** records (rows) of input for all keywords that use list style. The dimensions defined here represent a maximum number of input records but not all of them have to be in use at any particular stress period. For example, **NCROP** could be set to 3, indicating that all **LAND_USE** *List Style* input statements load three records–such as three root depths or three crop coefficients, but it is not required to simulate all crops all the time—for example one stress period could simulate crops 1, 2, and not 3, whereas another stress period could simulate crops 2, 3, and not 1.

The number of soil types, **NSOIL**, is only used to specify the number of capillary fringe lengths and soil coefficients to read. Soil coefficients are only necessary if it is desired to use the analytical pseudo steady-state soil moisture, soil-stress concept called *analytical root response* (appendix 5). It is recommended to specify **NSOIL** as one or more soil types for all FMP simulations and to specify a capillary fringe length—this allows for the calculation of evaporation of groundwater and the potential of groundwater-root uptake from a land use.

Keyword	Input	Description
NWBS	**INT**	Number of Water Balance Subregions. Alternative keyword—**NFARM**
NCROP	**INT**	Number of land-use types (or crops)
NSOIL	**INT**	Number of soil types. Only used in **SOIL** block
NIRRIGATE	**INT**	Number of irrigation types
NRD_TYPES	**INT**	Number of non-routed deliveries. Only used in **SURFACE_WATER** block
NSFR_DELIV	**INT**	Maximum number of SFR surface-water delivery points that are defined. Only used in **SURFACE_WATER** block
NSFR_RETURN	**INT**	Maximum number of SFR surface water return-flow points that are defined. Note—return-flow points can be automatically defined with fully routed return flow. Only used in **SURFACE_WATER** block

Figure 6.12. Required keywords for the Global Dimension Block. [After each keyword, a single integer, **INT**, is expected; SFR, streamflow routing package.]

The keyword **NIRRIGATE** specifies the number of irrigation efficiencies (OFE in appendix 4), which are specified in the **WATER_BALANCE_SUBREGION** block with the keyword **EFFICIENCY** and loaded with LAI[S, T, A, L-**NIRRIGATE**]. When **EFFICIENCY** uses *List Style* input, then it is loaded with **NWBS** rows and **NIRRIGATE** columns of efficiency values—note the actual number of columns is **NIRRIGATE**+1 to include the *List Style* record ID. If a crop is not irrigated, then in the **LAND_USE** block the keyword **IRRIGATION** LAI[S, T, A, L] has an irrigation flag set to 0 for the non-irrigated crop. Conversely, if it is irrigated, then its irrigation flag must be between 1 and **NIRRIGATE** when it is specified with "L-**NIRRIGATE**" to indicate which column to use in the OFE (**EFFICIENCY**). For more details and examples using the keyword **EFFICIENCY**, see the "Water-Balance Subregion (WBS) Block" section.

The keywords **NRD_TYPES**, **NSFR_DELIV**, and **NSFR_RETURN** define the maximum dimensions for properties defined in the **SURFACE_WATER** block, and if they are not specified, are set to zero. The **NRD_TYPES** is the number of **NON_ROUTED_DELIVERY** triplets that are specified per WBS. Each triplet defines a volume of water that is delivered to a WBS, its priority ranking among the other **NRD_TYPES**, and what to do with the unused portion of the water. **NSFR_DELIV** and **NSFR_RETURN** define the number of records that are read in for the **SURFACE_WATER** block keywords **SEMI_ROUTED_DELIVERY** and **SEMI_ROUTED_RETURN**, respectively (see the "**SURFACE_WATER** Block" section).

If the land-surface elevation is not specified in the Discretization package (DIS; appendix 3), then it must be specified as part of the **GLOBAL DIMENSION** block. Figure 6.13 describes the **SURFACE_ELEVATION** keyword that specifies the land-surface elevation grid that is loaded as NROW × NCOL array with ULOAD.

The remaining FMP keywords are optional and alter the behavior of the FMP globally. If the Unsaturated-Zone Flow (UZF) package is part of the simulation, then it may optionally be linked to the FMP. This simulates delayed recharge by having water that leaves the root zone, as deep percolation, behave as infiltration to UZF. This linkage is only necessary when there is field evidence that the transient time of deep percolation to the water table is much greater than the time step length. To initiate the FMP-UZF link, the keyword **UZF_LINK** must be specified in the **GLOBAL DIMENSION** block. If the **UZF_LINK** is enabled, it is important that the UZF package's ET option is disabled to prevent double accounting from FMP and UZF. All consumptive-use concepts require, at a minimum, specifying the soil capillary fringe in the **SOIL** block. The linear root uptake of groundwater and anoxia concept (formally designated concept 2 with ICCFL = 2 or 4) requires specifying the root depth (ROOT), fraction of transpiration (FTR), fraction of evaporation from irrigation (FEI), and fractions of delivery losses to surface water from precipitation and irrigation in the **LAND_USE** block for each crop.

Part of the FMP input requires specifying the SFR package segment and reach (or just the segment). A unique name can be assigned to the SFR segment and reaches, which may be used as a surrogate input in the FMP input that expects a SFR segment and reach. The FMP **GLOBAL DIMENSION** block may optionally specify the keyword **SFR_NAMES**, which loads with ULOAD a list of unique SFR stream names and their assigned segment and reach. As many names as necessary may be defined as long as they follow the ULOAD *List Style* input format. Each row loaded with ULOAD must specify the record ID, the unique name (SFRNAME), the SFR segment identifier (ISEG), and the SFR reach identifier (IRCH). The SFRNAME has a maximum size of 20 characters and the ID, ISEG, and IRCH are loaded as **INT**. Figure 6.14 is an example that defines three **SFR_NAMES**, which then allows the use of the words FARM_1_Deliv_Pnt, RES1_RELASE, and MainRiv in the place of specifying the SFR segment and reach.

Keyword	Input	Description
SURFACE_ELEVATION	ULOAD	Land-surface elevation of the model. Required if not specified in DIS. Loaded as **FLOAT** Array [L].

Figure 6.13. Keyword that specifies the land-surface elevation. Keyword is not required if land-surface elevation is already specified in the Discretization package (DIS). [FLOAT, floating-point number; [L], length in model units; ULOAD, universal loader]

```
#  Example use of the GLOBAL DIMENSION keyword SFR_NAMES.
#  Input uses ULOAD List Style input and continues to load input lines
#    if it successfully reads a List Style record ID
#
#  The name, SFRNAME, is linked to the specified SFR segment and reach
#
#  FMP input that expects an SFR segment and reach identity
#     may use the SFRNAME as a surrogate for the integer identifiers.
#
SFR_NAMES INTERNAL
# ID     SFRNAME           ISEG  IRCH
  1 FARM_1_Deliv_Pnt     22     1
  2 RES1_RELASE           2     3
  3 MainRiv               5     1
```

Figure 6.14. Example use of the FMP Global Dimension Keyword SFR_NAMES. This keyword associates a unique name with a stream flow routing (SFR) segment and reach.

By default, use of the **TRANSIENT** keyword in the **WATER_BALANCE_SUBREGION**, **CLIMATE**, and **LAND_USE** blocks specifies a TFR that loads input every stress period. If it is desired for the TFR to load information every time step, then the optional keyword **BY_TIMESTEP** followed by the keyword **WATER_BALANCE_SUBREGION**, **CLIMATE**, or **LAND_USE** will alter the TFR operation to load by time step for the respective FMP blocks. The following is an example of such usage for all three keywords:

```
BY_TIMESTEP WATER_BALANCE_SUBREGION
#
BY_TIMESTEP CLIMATE
#
BY_TIMESTEP LAND_USE
```

By default, all land-use types are assumed to have a surface elevation based on the land-surface elevation defined in the DIS package or with the keyword **SURFACE_ELEVATION**. If a specific crop has a surface elevation different from the default surface elevation, then, if used, the keyword **NCROP_SPECIFIED_ELEVATIONS** **INT** specifies the number of crops for which elevations are defined with the **LAND_USE** block. This is an advanced feature, and its use is not recommended, unless it is important to represent a land use (crop type) that has a different elevation than the model cell land-surface elevation. Figure 6.15 presents the **GLOBAL DIMENSION** block with all its supported keywords, their expected input structure, and a commented explanation.

```
BEGIN GLOBAL DIMENSION
  # Required Keywords ----------------------------------------
  NWBS        INT   # Number of water balance subregions (FARMS)
  NCROP       INT   # Number of land use types (CROPS)
  NSOIL       INT   # Number of soil types that are defined (SOIL)
  NIRRIGATE   INT   # Number of irrigation types that are defined (EFFICIENCY)
  NRD_TYPES   INT   # Number of non-routed delivery types defined (NRD)
  #
  # Required if not specified in DIS package -----------------------------
  #
  SURFACE_ELEVATION   ULOAD   # Read NROW by NCOL array of surface elevations
  #
  # The rest are optional keywords ---------------------------------
  #
  UZF_LINK     # Deep percolation becomes delayed recharge with the UZF Package
  #
  # Define custom unique names for SFR segments and reaches
  SFR_NAMES   ULOAD[ID, SFRNAME, ISEG, IRCH]   # List Style, as many records as necessary
  #
  # TFR for specified block is loaded every time step instead of by stress period
  BY_TIMESTEP WATER_BALANCE_SUBREGION
  BY_TIMESTEP LAND_USE
  BY_TIMESTEP CLIMATE
  #
  NCROP_SPECIFIED_ELEVATIONS INT   # Number of crops that have their elevation specified
  using LAND_USE block
  #
END GLOBAL DIMENSION
```

Figure 6.15. GLOBAL DIMENSION block with all supported keywords and their input format. [FLOAT, floating-point number; INT, integer; WBS, water-balance subregion; ULOAD is the universal loader; LAI, list-array input with the following letters representing the supported keywords: S for STATIC, T for TRANSIENT, A for ARRAY, and L for LIST]

Water Balance Subregion (WBS) Block

The **WATER_BALANCE_SUBREGION** (WBS) block specifies all properties that pertain to defining the supply and demand framework. All *List Style* input in the block expects **NWBS** records. In FMP, the block name can be declared as **WATER BALANCE SUBREGION** or **WATER_BALANCE_SUBREGION**, but the later version, with underscores, is recommended for clarity. Figure 6.16 describes keywords that may be required depending on the FMP setup.

LOCATION LAI[S, T, A], which defines the spatial location of each WBS, is required for all FMP simulations. The **LOCATION** keyword's LAI input is a two-dimensional **INT** array, where **INT** represents the WBS number. If the **LOCATION** input for any model cell is specified with **0**, then it is not included in supply and demand calculations and is ignored by the FMP. For example, if a model grid has three rows and four columns (NROW=3 and NCOL=4) and "**NWBS** 5" is in the **GLOBAL DIMENSION** block, then figure 6.17 illustrates acceptable **LOCATION** arrays. The spatial location of each WBS ID identifies the model cells that are included in the FMP—that is, if its WBS ID is greater than zero—and what sets of model cells are aggregated for supply and demand calculations—grouped by the WBS ID. An FMP subprocess—such as crop evapotranspiration, precipitation, evaporation—in a WBS location is aggregated as part of the FMP output—total precipitation over the WBS, total CU over the WBS.

If the land use (crop) receives applied-irrigated water, then there is an associated irrigation efficiency to account for; the excess irrigation required to meet demand after efficient losses becomes either surface runoff or deep percolation to groundwater. The **GLOBAL DIMENSION** block keyword **NIRRIGATE** defines the number of irrigation efficiencies that are specified. The irrigation efficiency (OFE, in appendix 4) is defined with the keyword **EFFICIENCY** and is loaded with LAI[S, T, A, L-**NIRRIGATE**]. It is recommended to use *List Style* input, which would then load **NWBS** records with **NIRRIGATE** columns of input. The OFE values can vary on the basis of agricultural practices and the type and quality of irrigation equipment used. It is recommended to research the efficiency on the basis of the simulated region, but typical values of OFE are **0.8** to **0.95** for drip irrigation, **0.7** to **0.85** for sprinkler irrigation, and **0.55** to **0.7** for flood irrigation. Figure 6.18 contains two examples of how to load the irrigation efficiencies when "**NIRRIGATE** 3" is in the **GLOBAL DIMENSION** block.

EFFICIENCY can be thought of as a lookup table for irrigated crops. If a crop is not irrigated, then its irrigation flag (see the "**LAND_USE** Block" section) is set to **0**. If the crop is irrigated and its **EFFICIENCY** is defined with *List Style* input, then the irrigation flag indicates the column in **EFFICIENCY** input to use, and the WBS that the crop resides in indicates the row. For example, given figure 6.18*A* and a crop has an irrigation flag of 2 and grows in WBS 3, then it would use the sprinkler efficiency for WBS ID 3, which is **0.77**. If **EFFICIENCY** is defined as an array, then a crop is irrigated if its irrigation flag is non-zero and uses the efficiency value at the same row and column as the crop's location.

Keyword	Input	Description
LOCATION	LAI[S, T, A]	**INT** array that specifies the spatial location of the WBS (always required).
EFFICIENCY	LAI[S, T, A, L-**NIRRIGATE**]	**FLOAT**, Irrigation efficiency (OFE) with 0 < OFE ≤ 1 Required if **NIRRIGATE** is greater than **0**.

Figure 6.16. WATER_BALANCE_SUBREGION block main keywords. [FLOAT, floating-point number; INT, integer; WBS, water-balance subregion; LAI, list-array input with the following letters representing the supported keywords: S for STATIC, T for TRANSIENT, A for ARRAY, and L for LIST]

A

```
#  Example use of the WATER_BALANCE_SUBREGION block keyword LOCATION
#
#  LOCATION defines the spatial location of each Water Balance Subregion (WBS).
#
#  Example assumes NROW = 3, NCOL = 4, NWBS = 5; Input type is LAI[S, T, A]
#
#  Input is an INT array of the model grid (NROW by NCOL).
#  INT is between 1 and NWBS, max number of WBS, to indicate the location of each WBS.
#
#  If a location is <1 or >NWBS, such as 0, then FMP ignores the model cell.
#      That is, it's not part of any WBS, so there is no supply and demand calculation.
##
# This following only specifies WBS 1 and 5, so WBS 2, 3, and 4 are not simulated.
#
# Row 3, Column 1 is zero, so it's not associated with a WBS nor simulated in FMP.
LOCATION  STATIC ARRAY INTERNAL
1  1  5  5
1  1  5  5
0  5  5  5
```

B

```
#  Example specifies WBS 1, 2, and 3, so WBS 4 and 5 are not simulated
LOCATION  INTERNAL
2  2  3  1
2  2  3  1
3  1  3  2
```

C

```
#  Example specifies WBS 1, 2, 3, 4 and 5, so all WBS are simulated
LOCATION  INTERNAL
5  4  3  2
1  5  4  3
2  1  5  4
```

Figure 6.17. Input examples for FMP WATER_BALANCE_SUBREGION block keyword LOCATION. Each example assumes that the maximum number of WBS is five. *A*, Example that specifies the location of WBS 1 and 5. *B*, Example that specifies the location of WBS 1, 2, and 5. *C*, Example that specifies the location of WBS 1, 2, 3, 4, and 5. [FLOAT, floating-point number; INT, integer; WBS, water-balance subregion; LAI, list-array input with the following letters representing the supported keywords: S for STATIC, T for TRANSIENT, A for ARRAY, and L for LIST]

A

```
#  Example use of the WATER_BALANCE_SUBREGION block keyword EFFICIENCY
#
#  EFFICIENCY defines the irrigation efficiency for each irrigation type.
#      – List Style input reads NWBS records
#
#  Example assumes NROW = 3, NCOL = 4, NWBS = 5; Input is LAI[S, T, A, L-NIRRIGATE]
#
#  Input is a FLOAT between 0 and 1.
#
# Chosen NIRRIGATE Types:
# 1 = Drip;     2 = Sprinkler;     3 = Flood
#
EFFICIENCY  STATIC LIST INTERNAL
#       Drip   Sprinkler   Flood
# ID    IRR1       IRR2    IRR3
  1    0.9     0.70    0.6
  2    0.9     0.70    0.6
  3    0.9     0.77    0.6
  4    0.9     0.70    0.6
  5    0.9     0.70    0.6
```

B

```
#  Example assumes NROW = 3, NCOL = 4, NWBS = 5; Input is LAI[S, T, A, L-NIRRIGATE]
#
EFFICIENCY  STATIC ARRAY INTERNAL
  0.9  0.9  0.7  0.7
  0.9  0.9  0.7  0.7
  0.6  0.6  0.6  0.6
```

Figure 6.18. Examples of input for FMP WATER_BALANCE_SUBREGION block keyword EFFICIENCY. *A*, List style input example. *B*, Array style input example. [FLOAT, floating-point number; INT, integer; WBS, water-balance subregion; LAI, list-array input with the following letters representing the supported keywords: S for STATIC, T for TRANSIENT, A for ARRAY, and L for LIST]

Recommended Water-Balance (WBS) Keywords

Keywords may be required depending on the FMP setup and are recommended as part of the normal input. Figure 6.19 lists the keywords, input format, and a description of the functions.

The first recommended keyword (fig. 6.19) is **`DEFICIENCY_SCENARIO`** `LAI[S, T, L]`, which defines if a WBS has the capability to obtain external water supplies for irrigated crops when its supply does not meet its demand. A value of `0` indicates that the WBS has access to external water (EXT) sources outside of the simulation domain, called the Zero-Scenario, that ensure demand is always met. A value of `1` indicates that the WBS only uses its available water supplies, called the Deficit Irrigation Scenario. If **`DEFICIENCY_SCENARIO`** is not specified, then all WBSs are automatically set to the Deficit Irrigation Scenario and will only apply the water supplies specified in the FMP input—that is, from non-routed deliveries, surface-water deliveries, and groundwater pumping. The following are examples of specifying **`DEFICIENCY_SCENARIO`** for five WBSs.

Keyword	Input	Description
DEFICIENCY_SCENARIO	LAI[S, T, L]	**INT.** Defines whether a WBS has Deficit Irrigation or has access to external water sources to meet a supply shortfall. If not specified, then Deficit Irrigation is the default option.
BARE_RUNOFF_FRACTION	LAI[S, T, A]	**FLOAT.** Defines the fraction of precipitation that becomes runoff when it falls on "bare soil", for surface area that is not defined with a Land Use (Crop). Bare soil is any model cell that is associated with a WBS LOCATION between 1 and NWBS and has either: a Crop LOCATION number not between 1 and NCROP or the remaining area in a model cell if LAND_USE_AREA_FRACTION is specified and does not sum to 1. If not specified, then the default fraction is set to 0.75
WATERSOURCE	LAI[S, T, L-3]	**INT.** Specifies sources of water, 0 to indicate it is not available/disabled and 1 to indicate it is available. The sources are groundwater pumping, surface water deliveries, and non-routed deliveries. If a source is disabled, then it automatically disables the source, overriding any contrary indication specified in another input. If enabled, then the sources of water are defined by their respective input sections. If not specified, then it is automatically set to a default value, 1, for all sources—all sources are available.

Figure 6.19. WATER_BALANCE_SUBREGION block recommended keywords. [FLOAT, floating-point number; INT, integer; LAI, list-array input; NCROP, number of land-use types; NWBS, maximum number of water-balance subregions; WBS, water-balance subregion; LAI, list-array input with the following letters representing the supported keywords: S for STATIC, T for TRANSIENT, A for ARRAY, and L for LIST]

```
# NWBS = 5
#
# Recommended Setup, All WBS Deficit Irrigate.
# (This is the also the default when not specified)
#
DEFICIENCY_SCENARIO CONSTANT 1
#
# WBS 2 and 3 can obtain external water supplies
DEFICIENCY_SCENARIO STATIC LIST
INTERNAL
1  1
2  0
3  0
4  1
5  1
```

Another important keyword is **BARE_RUNOFF_FRACTION** LAI[S, T, A, L], which accounts for the fraction of effective precipitation that falls on the bare soil area that becomes runoff compared to deep percolation. It should be a value between 0 and 1 and represents the fraction of excess precipitation that becomes runoff compared to infiltration, for example, 0.75 indicates that 75 percent of the precipitation that falls on bare soil becomes surface runoff. The following syntax examples show **BARE_RUNOFF_FRACTION** when input with (1) *List Style*, specified by WBS, and (2) *Array Style*, specified by model Row and Column input:

```
# NROW = 3, NCOL = 4, NWBS = 5
#
BARE_RUNOFF_FRACTION STATIC LIST INTERNAL
# ID   Fraction of effective precipitation to runoff
  1   0.0   # WBS 1 has all become deep
  percolation
  2   0.9
  3   1.0   # WBS 3 has all become runoff
  4   0.25
  5   0.50  # WBS 5 has half become runoff
#
# This example shows the array style input
BARE_RUNOFF_FRACTION STATIC ARRAY
INTERNAL
0.9  0.9  0.25  0.7
0.9  0.9  0.25  0.7
0.5  0.5  0.50  0.7
```

The **WATERSOURCE** LAI[S, T, L-3] keyword declares the available sources of irrigation water to each WBS. These sources are only used when groundwater-root uptake and precipitation are not enough to meet a WBS's demand. There are three available sources of water: (1) groundwater pumping from the FMP wells and Multi-Node Well Package Version 2 (MNW2), (2) surface-water deliveries, as semi-routed deliveries (SRDs) from SFR, and (3) non-routed deliveries (NRDs). The keyword signals a *List Style* input composed of three columns of values set to 0 or 1. A value of 0 indicates that the source is not available, and 1 indicates it is available. The order of the columns, groundwater pumping, surface-water delivery, and non-routed delivery, are fixed and represent the sources of water. The following are examples of how to set the **WATERSOURCE** keyword:

```
# NROW = 3, NCOL = 4, NWBS = 5
#
WATERSOURCE  STATIC LIST INTERNAL
# ID  GW   SW  NRD
  1    1    0    0  # WBS 1 only has access to groundwater pumping
  2    0    1    0  # WBS 2 only has access to semi-routed deliveries
  3    1    1    1  # WBS 3 has access to all sources of water
  4    0    0    0  # WBS 4 has no sources of water other than natural ones
  5    0    0    1  # WBS 5 only has access to non-routed deliveries
```

If a source of water is declared as not being available through the **WATERSOURCE** keyword, then this supersedes any contradictory input. However, it also is recommended to ensure that if a WBS does not have access to a specific supply defined it also is designated as not having it as a water source. For example, if a WBS does not have an NRD, its delivery volume should be set to zero, and if another WBS does not have a surface-water delivery, then its SRD segment and reach should be set to zero (ISEG=0 and IRCH=0). The **WATERSOURCE** keyword can be thought of as a simulation speed improvement as it triggers an automatic bypass of the algorithms that check the water sources. For example, without **WATERSOURCE**, the FMP would still check for the total supply of non-routed deliveries even though their total may amount to zero.

Additional Water-Balance (WBS) Options

Additional keywords can be used depending on the FMP setup for situations such as additional demand, prorating available supplies during irrigation deficiency, and efficiency improvement. Figure 6.20 lists the keywords, input format, and a description of the functions.

If the user wishes to specify additional demand to a WBS, such as pre-wetting a field, the keywords **ADDED_DEMAND_RUNOFF_SPLIT** and **ADDED_CROP_DEMAND** allow for this. The **ADDED_DEMAND_RUNOFF_SPLIT** specifies the fraction of the additional demand that becomes runoff and the remainder becomes deep percolation. It is specified for each WBS and Irrigation type (**NIRRIGATE**). If not specified, the **ADDED_DEMAND_RUNOFF_SPLIT** defaults to **0.1** to indicate **10** percent becomes runoff and **90** percent is deep percolation. It also may be specified using *Array Style*, which would apply the split to any additional demanded water that is applied to the specific row and column location. The keyword **ADDED_CROP_DEMAND** specifies any additional demands that are applied during a stress period. The second keyword associated with **ADDED_CROP_DEMAND** indicates the units in which the additional demand is specified. Both **ADDED_CROP_DEMAND FLUX** and **ADDED_CROP_DEMAND RATE** may be specified in the **WATER_BALANCE_SUBREGION** block, but **FLUX** and **RATE** may only be specified once (fig. 6.20). The keyword **FLUX** indicates that the additional demand is specified as a length per time (L/T) or flux (L^3/L^2-T), and the keyword **RATE** indicates that the additional demand is specified as a volume per time or volumetric rate (L^3/T). The additional demand is specified for each WBS and for each crop. If the crop has an additional demand, but is not grown within the WBS, then its additional demand is ignored. In the following example input block, **NCROP** is equal to 2:

```
# NROW = 3, NCOL = 4, NWBS = 5, NCROP = 2,  NIRRIGATE = 3
# Model length units are in meters, and time unit is day
#
ADDED_DEMAND_RUNOFF_SPLIT   STATIC LIST INTERNAL
#      Drip   Sprinkler   Flood
#     IRR1       IRR2     IRR3
  1    0.1       0.2     0.1
  2    0.1       0.2     0.1
  3    0.0       0.1     0.05
  4    0.1       0.2     0.1
  5    0.1       0.2     0.1
#
ADDED_CROP_DEMAND FLUX   STATIC  LIST  INTERNAL
#       CROP1    CROP2
  1    0.0254     0.0   # WBS 1, Crop 1 has 1 inch/day  of additional demand
  2    0.01       0.01  # WBS 2, Crop 1 and 2 have 1 cm/day  of additional demand
  3    0.0        0.0   # WBSs 3-5 have no additional demand
  4    0.0        0.0
  5    0.0        0.0
#
ADDED_CROP_DEMAND RATE   STATIC LIST INTERNAL
#       CROP1  CROP2
  1    100.0    0.0   # WBS 1, Crop 1 has 100 m3/day of additional demand
  2    1E3      1E3   # WBS 2, Crops 1 and 2 have 1000 m3/day of additional demand
  3    0.0      0.0   # WBSs 3-5 have no additional demand
  4    0.0      0.0
  5    0.0      0.0
```

Note that in this example the additional demand specified from **ADDED_CROP_DEMAND FLUX** and **ADDED_CROP_DEMAND RATE** is cumulative—that is, the sum of all additional demands applied to a land use is the final amount specified for the respective WBS.

For the WBSs that have Deficit Irrigation (by setting the **DEFICIENCY_SCENARIO** to 1), two options are provided to prorate the available supplies to the individual demands within the WBS. This option is set with the keyword **PRORATE_DEFICIENCY** {**ByDEMAND**, **ByAVERAGE**}, where **ByDEMAND** indicates that the available supply is applied to each individual demand on the basis of the ratio of its demand to the total WBS demand, and **ByAVERAGE** indicates that the supply is applied on the basis of an average demand across the WBS. The options **ByDemand** and **ByAverage** are described in detail in the "New Implementation of Deficit Irrigation" section in the main body of this report. If neither **PRORATE_DEFICIENCY** option is specified, then the default is to use **ByDemand**.

For the WBSs that have Deficit Irrigation (by setting the **DEFICIENCY_SCENARIO** to 1), two options determine how the irrigation efficiencies that are defined with **EFFICIENCY** are affected when under deficit irrigation. The two options are to improve irrigation efficiency when under deficit irrigation—a supply shortfall—or hold irrigation efficiencies constant. This selection is set with the keyword **EFFICIENCY_IMPROVEMENT** LAI[S, T, A, L-**NIRRIGATE**], where a **0** indicates that irrigation efficiency remains constant, and a 1 indicates that the irrigation efficiency improves when the WBS is under deficit irrigation. When **EFFICIENCY_IMPROVEMENT** is not specified, then a value of **0** is set by default and irrigation efficiency remains constant.

Keyword	Input	Description
ADDED_DEMAND_RUNOFF_SPLIT	LAI[S, T, A, L-**NIRRIGATE**]	**FLOAT**, Fraction of any specified additional demand that becomes runoff as opposed to deep percolation. If not specified, the default is **0.1**
ADDED_CROP_DEMAND FLUX	LAI[S, T, L-**NCROP**]	**FLOAT**, Additional demand added to a land use (CROP). Input specified as length [L/T] or flux [L^3/L^2-T]
ADDED_CROP_DEMAND RATE	LAI[S, T, L-**NCROP**]	**FLOAT**, Additional demand added to a land use (CROP). Input specified as volumetric rate [L^3/T]
PRORATE_DEFICIENCY	{ **ByDEMAND**, **ByAVERAGE** }	Defines how applied water is spread among different demands when total supply does not meet total demand. If not specified, then the default is **ByDEMAND**.
EFFICIENCY_IMPROVEMENT	LAI[S, T, A, L-**NIRRIGATE**]	**INT**, If Deficit Irrigation occurs allows irrigation efficiency (OFE) to improve in response to supply shortfall. Set to **0** to hold OFE constant and **1** to indicate OFE improves under deficit irrigation. If not specified, the default is **0**.
WBS_NAME	ULOAD	Specify a name for each WBS. Input expects **NWBS** records. Max size is 20 characters.

Figure 6.20. WATER_BALANCE_SUBREGION block optional, advanced keywords. [FLOAT, floating-point number; INT, integer number; [L/T], length per time in model units; [L^3/L^2-T], flux in model units; [L^3/T], volumetric rate in model units; NWBS, maximum number of water-balance subregions; ULOAD, universal loader; WBS, water-balance subregion; LAI, list-array input with the following letters representing the supported post-keywords: S for STATIC, T for TRANSIENT, A for ARRAY, and L for LIST]

Supported SFAC DIMKEY

The WATER_BALANCE_SUBREGION block keywords that support SFAC DIMKEYs are presented in figure 6.21. Only the keywords that support SFAC DIMKEY are specified, and an X is placed under the respective DIMKEY if it is supported by the corresponding keyword. See section "Advanced Scale Factors and SFAC" for descriptions of the usage of the SFAC keyword and its supported DIMKEYs. Figure 6.22 presents the WATER_BALANCE_SUBREGION block with keyword usages that are relevant to most simulations; the basic definition of each keyword is included as comments.

Keyword	DIMKEY		
	ByWBS	ByCrop	ByIrrigate
EFFICIENCY	X	X	X
BARE_RUNOFF_FRACTION	X		
ADDED_DEMAND_RUNOFF_SPLIT	X	X	X
ADDED_CROP_DEMAND FLUX	X	X	
ADDED_CROP_DEMAND RATE	X	X	

Figure 6.21. WATER_BALANCE_SUBREGION block keywords that support SFAC DIMKEY. The supported keywords that can be used for DIMKEY are ByWBS, ByCrop, and ByIrrigate. [Supported keyword and DIMKEY combinations are marked with an X.]

```
BEGIN WATER_BALANCE_SUBREGION
 #
 #  Required Keywords ----------------------------------------
 #
 LOCATION     LAI[S, T, A]                  # Specify spatial location of the WBS
 #
 EFFICIENCY  LAI[S, T, A, L-NIRRIGATE]  # Specify Irrigation Efficiency (OFE) if NIRRIGATE > 0
 #
 #  Recommended Keywords -------------------------------------
 #
 DEFICIENCY_SCENARIO  LAI[S, T, L]  # Set to: 1 for Deficit Irrigation (default); 0 for Zero-Scenario
 #
 #  Fraction of precipitation on bare soil area that becomes runoff. Default is 0.75, if not specified.
 BARE_RUNOFF_FRACTION LAI[S, T, A, L]
 #
 #  Additional WBS Options -----------------------------------
 #
 WBS_NAME    ULOAD  # WBS name used in output. Default name is "WBS_X" where X is the WBS ID.
 #
 #  Fraction of additionally demanded irrigation water that becomes runoff. Default is 0.1, if not
 specified.
 ADDED_DEMAND_RUNOFF_SPLIT  LAI[S, T, A, L-NIRRIGATE]
 #
 #  Specify additional demand applied to each crop
 #
 ADDED_CROP_DEMAND  FLUX    LAI[S, T, A, L-NCROP]  # Specified in units of length [L/T]
 #
 ADDED_CROP_DEMAND  RATE    LAI[S, T, A, L-NCROP]  # Specified in units of volume [L^3/T]
 #
 #  Specify how supply is distributed when deficit irrigation occurs.
 #  If keyword is not present, then the default is ByDEMAND
 PRORATE_DEFICIENCY  {ByDEMAND, ByAVERAGE}
 #
 #  Specify if a WBS can improve irrigation efficiency when under deficit irrigation.
 #  Set to 0, no improvement, or to 1, improve efficiency. Default is 0
 EFFICIENCY_IMPROVEMENT  LAI[S, T, A, L-NIRRIGATE]
 #
 #  Specify whether WBS has available sources of irrigation or supplemental water for delivery.
 #  L-3 signifies that values (0 or 1) input as 3 columns refer to each of 3 sources—GW (pumping), SW
 (semi-routed), and NRD (non-routed), respectively.
 #  Set to 0 to disable source or 1 to indicate source is available. Default is all sources available.
 WATERSOURCE LAI[S, T, L-3]
 #
END WATER_BALANCE_SUBREGION
```

Figure 6.22. WATER_BALANCE_SUBREGION block with most of the supported keywords and their input format. [NIRRIGATE, number of irrigation types; NCROP, number of land use types; GW, groundwater; SW, surface water; LAI, list-array input with the following letters representing the supported keywords: S for STATIC, T for TRANSIENT, A for ARRAY, and L for LIST]

Options Block and Associated Keyword List

The **OPTIONS** block is optional and offers keywords that enact legacy FMP features that did not fit into any other input block. An FMP simulation can run without including the **OPTIONS** block or with specifying the block with nothing in it. It is recommended not to use these features unless necessary to mimic a feature of a previous FMP version. Figure 6.23 presents the **OPTIONS** block supported keywords. Figure 6.24 presents the **OPTIONS** block with the expected input structure and a commented explanation.

Keyword	Description
NOPRINT	Suppress some of the farm well output information from being printed to the LIST file.
WELLFIELD	Allows non-routed deliveries (NRD) to have the well field option, which simulates aquifer storage and recovery.
RECOMP_Q_BD	Forces the FMP to re-run its land-use calculations after model time step has completed. This keyword use is not recommended.
MNWCLOSE QCLOSE HPCT RPCT	Additional multi-node well package (MNW2) closure criteria (3 × **FLOAT**). QCLOSE is pumping criterion [L^3/T] HPCT is head criterion [L] RPCT is residual criterion [L^3/T] Use of this keyword is not recommended; the FMP automatically handles this.

Figure 6.23. Options block keywords list. [FLOAT, floating-point number; [L], length in model units; [L^3/T], volumetric rate in model units; MNW2, multi-node well package version 2]

```
BEGIN OPTIONS
  # All Keywords are Optional
  NOPRINT
  WELLFIELD
  RECOMP_Q_BD
  MNWCLOSE FLOAT FLOAT FLOAT # Reads 3 floating-point numbers: QCLOSE
  HPCT RPCT
  #
END OPTIONS
```

Figure 6.24. Options block with all supported keywords and their input format. [FLOAT, floating-point number.]

Output Block and Additional Output Keywords

The **OUTPUT** block is optional and contains keywords to output files that are not specifically related to any of the other blocks. Figure 6.25 presents the four keywords that are recommended for specifying output for all the FMP simulations.

FARM_WELL_CBC and **FARM_NET_RECHARGE_CBC** specify the binary cell-by-cell (CBC) file unit number that all MODFLOW packages write to. Each specified file unit number should match a unit number defined in the MF-OWHM2 NAME file. If they are not specified, then the CBC file unit number is set to `0` to indicate that CBC flows are not written to an external binary file. This CBC output file is then used by post-processing programs, like ZoneBudget, for analysis of all flows in and out of the groundwater system.

The keyword **FARM_DEMAND_SUPPLY_SUMMARY** may be, optionally, followed by a *Generic_Output_OptKey* to specify the file. If *Generic_Output_OptKey* is not present, then the file created is named "FDS.out". For every time step the output file contains each WBSs demand for water and corresponding supplies that were used to meet those demands. The output may be text or binary format. If the output file is text format, then it contains a single header line (fig. 6.26*A*) followed by the output written for each time step. If the output file is binary format, then there is no header and each binary record is formatted as described in figure 6.26*B*.

Keyword	Input	Description
FARM_WELL_CBC	**INT**	CBC unit number to write to for farm wells not linked to MNW2. **INT** is a unit number that is specified in the NAME file. The unit must be declared with the keyword "DATA(BINARY)".
FARM_NET_RECHARGE_CBC	**INT**	CBC unit number to write to for the Farm (WBS) Net-Recharge = Deep Percolation – ET_{gw}
FARM_DEMAND_SUPPLY_SUMMARY	[*Generic_Output_OptKey*]	Supply and Demand Summary on a WBS basis. Optional to specify file name and location with *Generic_Output_OptKey*, otherwise the default name is FDS.out
FARM_BUDGET	[*Generic_Output_OptKey*]	Landscape budget; all flows are relative to the landscape—IN to the landscape or OUT of the landscape. Optional to specify file name and location with *Generic_Output_OptKey*, otherwise the default name is "FB.DETAILS.OUT"

Figure 6.25. Output block recommended keyword list. These should be included in all Farm Process (FMP) simulations. [CBC, cell-by-cell unit; INT, integer input; MNW2, multi-node well package version 2; WBS, water-balance subregion; ET_{gw}, evapotranspiration from groundwater]

A

PER		is the stress period
STP		is the time step
TIMEUNIT		is elapsed simulation time. Note that the word TIMEUNIT is set to the model time unit T: "SECONDS", "MINUTES", "HOURS", "DAYS", or "YEARS"
FID		is the WBS (Farm) ID number
OFE		is total irrigation efficiency of the WBS
TFDR-INI	[L^3/T]	is the WBS initial demand for water
NR-SWD-INI	[L^3/T]	is the WBS initial demand for non-routed deliveries
R-SWD-INI	[L^3/T]	is the WBS initial demand for semi-routed deliveries.
QREQ-INI	[L^3/T]	is the WBS initial demand for groundwater pumping
TFDR-FIN	[L^3/T]	is the WBS amount of demand satisfied with supply
NR-SWD-FIN	[L^3/T]	is the portion of demand satisfied with non-routed deliveries
R-SWD-FIN	[L^3/T]	is the portion of demand satisfied with semi-routed deliveries.
QREQ-FIN	[L^3/T]	is the demanded groundwater pumping
Q-FIN	[L^3/T]	is the portion of demand satisfied with groundwater pumping
DEF-FLAG		is the WBS **DEFICIENCY_SCENARIO**
DATE_START		is the starting calendar date of the time step in the form: yyyy-mm-ddThh:mm:ss
ACTIVE		is set to if the WBS is in use for the stress period and 0 if it is not

B

DATE_START		CHARACTER(19), starting date formatted as 'yyyy-mm-ddThh:mm:ss'
ACTIVE		INTEGER
PER		INTEGER
STP		INTEGER
DELT	[T]	DOUBLE, time step length
TOTIM	[T]	DOUBLE, simulated time at end of time step
FID		INTEGER
DEF-FLAG	[L^3/T]	INTEGER
TFDR-INI	[L^3/T]	DOUBLE
NR-SWD-INI	[L^3/T]	DOUBLE
R-SWD-INI	[L^3/T]	DOUBLE
QREQ-INI	[L^3/T]	DOUBLE
TFDR-FIN	[L^3/T]	DOUBLE
NR-SWD-FIN	[L^3/T]	DOUBLE
R-SWD-FIN	[L^3/T]	DOUBLE
QREQ-FIN	[L^3/T]	DOUBLE
Q-FIN	[L^3/T]	DOUBLE

Figure 6.26. FARM_DEMAND_SUPPLY_SUMMARY (FDS.out) text file header explanation and binary record structure. *A*, Text-header explanation. *B*, Binary-record structure. [[T], unit of time in model units; [L^3/T], volumetric rate in model units; CHARACTER(19) indicates record is 19 characters long (19 bytes); INTEGER is a 4-byte integer record; DOUBLE is a 8-byte floating-point number record.]

The keyword **FARM_BUDGET** may be optionally followed by a *Generic_Output_OptKey* to specify the file. If *Generic_Output_OptKey* is not present, then the file created is named "FB_DETAILS.out". The **FARM_BUDGET** output file may be text or binary format. If the output file is text format, then it contains a single header line (fig. 6.27*A*) followed by the output written for each time step. If the output file is binary format, then there is no header and each binary record is formatted as described in figure 6.27*B*. The output file contains for every time step the flow rates in and out of each WBS landscape, where the flow directions are relative to the landscape. That is, an "`-in`" flow is a flow that enters the landscape—for example precipitation enters the landscape—and an "`-out`" is a flow that leaves the landscape—the precipitation that enters the landscape may leave as evaporation, transpiration, runoff, or deep percolation. The **FARM_BUDGET** output file does contain several "`-in`" headers that are always equal to their corresponding "`-out`" headers because the flow pass through the landscape—for example, groundwater evaporation passes through the landscape to the atmosphere, so `Q-egw-in` = `Q-egw-out`.

Figure 6.28 describes keywords that create output files available in previous versions of the FMP. For exact details on their output see Hanson and others (2014) and Schmid and others (2006, 2009). Each keyword maybe followed with a *Generic_Output_OptKey* to specify the file name. If *Generic_Output_OptKey* is not specified, then the default filename is used. These output files do not support **BINARY** output.

A

PER	is the stress period	
STP	is the time step	
TIMEUNIT	is elapsed simulation time. Note that the word TIMEUNIT is set to the model time unit T: "SECONDS", "MINUTES", "HOURS", "DAYS", or "YEARS"	
FID	is the WBS (Farm) ID number	
Q-p-in	is precipitation that enters the WBS	[L^3/T]
Q-nrd-in	is the rate that non-routed delivery flow enters the WBS	[L^3/T]
Q-srd-in	is the rate that semi-routed delivery flow enters the WBS	[L^3/T]
Q-wells-in	is the rate that groundwater well water enters the WBS	[L^3/T]
Q-egw-in	is evaporation from groundwater	[L^3/T]
Q-tgw-in	is transpiration from groundwater	[L^3/T]
Q-mar-in	is flood water applied to the WBS as managed aquifer recharge.	[L^3/T]
Q-drch-in	is **DIRECT_RECHARGE** applied to WBS—similar to Recharge Package.	[L^3/T]
Q-ext-in	is the Zero-Scenario external water required to meet demand.	[L^3/T]
Q-tot-in	is total flows into the WBS	[L^3/T]
Q-ei-out	is evaporation from irrigation out of the landscape	[L^3/T]
Q-ep-out	is evaporation from precipitation out of the landscape	[L^3/T]
Q-egw-out	equals Q-egw-in because it passes through the landscape to the atmosphere	[L^3/T]
Q-ti-out	is transpiration from irrigation out of the landscape	[L^3/T]
Q-tp-out	is transpiration from precipitation out of the landscape	[L^3/T]
Q-tgw-out	equals Q-tgw-in because it passes through the landscape to the atmosphere	[L^3/T]
Q-run-out	is total overland runoff out of the WBS	[L^3/T]
Q-dp-out	is deep percolation that leaves the root zone—this includes drch and mar.	[L^3/T]
Q-nrd-out	is the rate that non-routed delivery flow leaves the WBS	[L^3/T]
Q-srd-out	is the rate that semi-routed delivery flow leaves the WBS from excess NRD	[L^3/T]
Q-rd-out	is the rate that fully-routed flow leaves the WBS from excess NRD	[L^3/T]
Q-wells-out	is the rate that groundwater well water is injected from excess NRD	[L^3/T]
Q-tot-out	is the total outflow	[L^3/T]
Q-in-out	is the difference between Q-tot-in and Q-tot-out	[L^3/T]
Q-Discrepancy[%]	is the percent error between Q-tot-in and Q-tot-out	[%]
DATE_START	is the starting calendar date of the time step in the form: yyyy-mm-ddThh:mm:ss	
ACTIVE	is set to 1 if the WBS is in use for the stress period and 0 if it is not	

Figure 6.27. FARM_BUDGET (FB_DETAILS.out) text file header explanation and binary record structure. *A*, Text-header explanation. *B*, Binary-record structure. [[T], unit of time in model units; [L^3/T], volumetric rate in model units; CHARACTER(19) indicates record is 19 characters long (19 bytes); INTEGER is a 4-byte integer record; DOUBLE is an 8-byte floating-point number record; %, percent.]

B

DATE_START	CHARACTER(19), starting date formatted as 'yyyy-mm-ddThh:mm:ss'	
ACTIVE	INTEGER	
PER	INTEGER	
STP	INTEGER	
DELT	DOUBLE, time step length	[T]
TOTIM	DOUBLE, simulated time at end of time step	[T]
FID	INTEGER	
Q-p-in	DOUBLE	
Q-nrd-in	DOUBLE	
Q-srd-in	DOUBLE	
Q-wells-in	DOUBLE	
Q-egw-in	DOUBLE	
Q-tgw-in	DOUBLE	
Q-mar-in	DOUBLE	
Q-drch-in	DOUBLE	
Q-ext-in	DOUBLE	
Q-ei-out	DOUBLE	
Q-ep-out	DOUBLE	
Q-egw-out	DOUBLE	
Q-ti-out	DOUBLE	
Q-tp-out	DOUBLE	
Q-tgw-out	DOUBLE	
Q-run-out	DOUBLE	
Q-dp-out	DOUBLE	
Q-nrd-out	DOUBLE	
Q-srd-out	DOUBLE	
Q-rd-out	DOUBLE	
Q-wells-out	DOUBLE	

Figure 6.27. —Continued

A

Keyword	Default Filename	Description
FARM_BUDGET_COMPACT	FB_COMPACT.OUT	Same as FARM_BUDGET, but aggregates water by common type [L^3/T].
FARM_NET_RECHARGE_ARRAY	FNRCH_ARRAY.OUT	Write cell-by-cell WBS (Farm) net recharge (FNR) as an array to a text file [L^3/T]. FNR is the difference between deep percolation and evaporation from groundwater.
FARM_NET_RECHARGE_LIST	FNRCH_LIST.OUT	Write WBS (Farm) net recharge (FNR) for each WBS to a text file [L^3/T]. FNR is the difference between deep percolation and evaporation from groundwater.
FARM_WELL_SUMMARY	FWELLS.OUT	Print out layer, row, column, and final rate for all WBS (Farm) wells [L^3/T].

B

Keyword	Default Filename	Description
EVAPOTRANSPIRATION_SUMMARY SUM	ET_ARRAY.OUT	Write cell-by-cell evapotranspiration as an array to a text file [L^3/T].
EVAPOTRANSPIRATION_SUMMARY SEPARATE	E_n_T_ARRAY.OUT	Write cell-by-cell evaporation and transpiration as two separate arrays in a text file [L^3/T].
EVAPOTRANSPIRATION_LIST	ET_LIST.OUT	Write to a text file the evaporation, transpiration, and evapotranspiration for each WBS [L^3/T].

Figure 6.28. Output block additional keywords that provide specific output information. The resulting file created has the default file name or, optionally, the output file name and location can be specified with *Generic_Output_OptKey*. *A*, WBS (farm) specific output options. *B*, Evapotranspiration specific output options. *C*, Runoff and deep percolation output options. [[L^3/T], volumetric rate in model units; NCOL and NROW, number of columns and rows, respectively, in the model grid; WBS, water-balance subregion]

C

Keyword	Default Filename	Description
LANDSCAPE_RUNOFF	RUNOFF.out	Write the cell-by-cell surface runoff [L^3/T]. An optional keyword, COMPACT, can be placed after the keyword to indicate that output should only write model cells whose runoff is greater than zero as a compact list of row, column, runoff. If not specified, then output is a NROW by NCOL array.
DEEP_PERCOLATION	DPERC.out	Write the cell-by-cell deep percolation [L^3/T]. An optional keyword, COMPACT, can be placed after the keyword to indicate that output should only write model cells whose runoff is greater than zero as a compact list of row, column, runoff. If not specified, then output is a NROW by NCOL array.

Figure 6.28. —Continued

Climate Block

The **CLIMATE** block is conditionally optional and specifies reference evapotranspiration, potential evaporation, precipitation, potential consumption of precipitation, and any additional recharge passed as deep percolation—this is like the Recharge Package (RCH). Figure 6.29 presents all the keywords that the **CLIMATE** block can load.

The **CLIMATE** block becomes required if a land use/crop is defined using crop coefficients or if a model cell that is in a WBS does not have a defined land use. If crop coefficients are defined in the **LAND_USE** block, then it is necessary to specify **REFERENCE_ET** (fig. 6.29*A*) to calculate the land use potential consumptive use (CU). The second situation that requires the **CLIMATE** block is a model cell that resides in a WBS but has no defined land use, which is assumed to be bare soil. This situation requires a user to specify a **POTENTIAL_EVAPORATION_BARE** (fig. 6.29*B*) to use as a potential evaporation rate for groundwater and precipitation. If **REFERENCE_ET** is specified, but the **POTENTIAL_EVAPORATION_BARE** is not specified, the FMP will automatically use the **REFERENCE_ET** as the potential evaporation rate over bare soil. Any remaining precipitation after evaporation becomes either runoff or deep percolation based on the **WATER_BALANCE_SUBREGION** block keyword **BARE_RUNOFF_FRACTION**.

The **PRECIPITATION** keyword (fig. 6.29*A*) is optional and specifies the precipitation rate (L/T) that falls over the WBSs. It is specified as an array that has the same dimension as the model grid (NROW × NCOL). The scale of the precipitation must be an equivalent rate for the entire stress period—or equivalent for the time step if the **WATER_BALANCE_SUBREGION** block has the keyword **BY_TIMESTEP CLIMATE**. Precipitation is automatically set to zero if it is specified on a model cell that is not associated with a WBS—specifically, any model cells with a **WATER_BALANCE_SUBREGION LOCATION** that is less than 1 or greater than **NWBS** have no defined WBS and thus are assigned no precipitation. If **PRECIPITATION** is not specified, the FMP assumes there is no precipitation for the simulation.

The **PRECIPITATION_POTENTIAL_CONSUMPTION** keyword (fig. 6.29*C*) is optional and limits how much precipitation can be consumed as evaporation and transpiration (consumptive use). The use of the **PRECIPITATION_POTENTIAL_CONSUMPTION** keyword is not recommended unless FMP simulation results indicate an overconsumption of precipitation. This could occur when a stress period has a long-time span compared to the total time precipitation occurred during the stress period. Instead of using this option, it may be better to specify precipitation by time step to get a more accurate representation of precipitation events—this is done with the keyword **BY_TIMESTEP CLIMATE** in the **GLOBAL DIMENSION** block. Using **PRECIPITATION_POTENTIAL_CONSUMPTION** keyword is only advantageous when a stress period has high intensity, short-duration precipitation events relative to the stress period length or as a correction factor for sporadic precipitation events during a stress period.

The limit on precipitation consumption can be viewed as defining a potential effective precipitation, which represents the quantity of rainfall that is not runoff nor deep percolation. The reason it is considered a potential effective precipitation is because there can be additional runoff and deep percolation if the landscape's CU is less than the potential consumption of precipitation—that is, the simulated actual evapotranspiration does not consume all the available precipitation. If not specified, then the potential consumption of precipitation is the full amount of precipitation. The limit can be defined as a length—**PRECIPITATION_POTENTIAL_CONSUMPTION BY_LENGTH**—or as a fraction—**PRECIPITATION_POTENTIAL_CONSUMPTION BY_FRACTION**. If defined as a length, then only precipitation up to the specified length (L/T) is available for consumption. If defined as a fraction, then it is multiplied by the precipitation to get the effective precipitation length. To disable limiting the effective precipitation, set the length to 1E30 or any value larger than the actual precipitation.

Note that the keyword **EFFECTIVE_PRECIPITATION_TABLE** in the **SOIL** block functions identically to **PRECIPITATION_POTENTIAL_CONSUMPTION**. The difference is that the **SOIL** block specifies the potential consumption of precipitation as a lookup table of precipitation to effective precipitation by soil type. See the "Soil Block" section for more details and examples of transforming precipitation to effective precipitation.

A

Keyword	Description
REFERENCE_ET	Reference Evapotranspiration (ET_{ref}), as a length or flux, that occurs over model cell. Required if land use defined with crop coefficients (K_c). Zero is an acceptable value for ET_{ref}, but this will disable all evapotranspiration. Loaded as **FLOAT** Array [L/T].
PRECIPITATION	Precipitation, as a length or flux, that falls over model cell. Precipitation that is for model cells that are not associated with a WBS is ignored—that is a **WATER_BALANCE_SUBREGION LOCATION** that is less than 1 or greater than **NWBS**. Loaded as **FLOAT** Array [L/T].

Figure 6.29. CLIMATE block keyword list. All keywords are followed by input loaded with LAI[S, T, A]. Secondary keywords enclosed in curly braces, { }, indicate that only one of the keywords may be selected and used during a simulation. *A*, Recommended CLIMATE block keywords. *B*, Optional CLIMATE block keywords. *C*, Advanced CLIMATE block keywords. [FLOAT, floating-point number; L/T, length per time in model units; L^3/T, volumetric rate in model units; NWBS, number of water-balance subregions; UZF, unsaturated-zone flow package; WBS, water-balance subregion; LAI, list-array input with the following letters representing the supported keywords: S for STATIC, T for TRANSIENT, A for ARRAY]

B

Keyword	Description
DIRECT_RECHARGE {FLUX, RATE}	Recharge that is added to deep percolation and not available for consumption as evaporation or transpiration. The model cell that has DIRECT_RECHARGE must be associated with a WBS. Mimics the Recharge package (RCH), except if UZF_LINK is enabled then deep percolation is received as infiltration to UZF. DIRECT_RECHARGE must be followed by either **FLUX** or **RATE**. **FLUX** keyword indicates input is read as [L/T] and the **RATE** keyword indicates input is [L^3/T] Read as a **FLOAT** array.
POTENTIAL_EVAPORATION_BARE	Potential Evaporation rate of groundwater and precipitation over any model cell located in a WBS that does not have a land use defined (Bare Soil). If not specified and REFERENCE_ET is, then it is set to half of REFERENCE_ET. It is required, either specified directly or via REFERENCE_ET, if there are model cells within a WBS that do not have a defined land use (Bare Soil). Loaded as **FLOAT** Array [L/T].

C

Keyword	PRECIPITATION_POTENTIAL_CONSUMPTION {BY_LENGTH, BY_FRACTION}
Description	Places a limitation on the consumption of precipitation by the landscape as evaporation and transpiration. Precipitation that is beyond limit becomes runoff or deep percolation. If not specified, then there is no limit and actual evapotranspiration can potentially consume all the precipitation. The **BY_LENGTH** keyword indicates that input is in units of length per time [L/T] and any precipitation in excess of this length is not available for consumption. The **BY_FRACTION** keyword indicates that input is a fraction between 0 and 1 to indicate the fraction of the precipitation that is available for consumption. Loaded as **FLOAT** Array.

Figure 6.29. —Continued

DIRECT_RECHARGE (fig. 6.29*B*) is a source of water not available for consumption by a land use/crop. **DIRECT_RECHARGE** is applied directly to deep percolation that either recharges the water table or becomes an inflow to the unsaturated zone—through the **UZF_LINK**—for delayed recharge. It can be viewed as a surrogate for the Recharge (RCH) package but is included in the FMP landscape budgets and optionally linked to UZF.

Supported **SFAC DIMKEY**

The **CLIMATE** block keywords that support **SFAC DIMKEY**s are presented in Figure 6.30. Only the keywords that support **SFAC DIMKEY** are specified and an "X" placed under the respective **DIMKEY**. The only **SFAC DIMKEY** that is supported by the **CLIMATE** block is **ByWBS**.

IXJ Style Input Support

At the time of this report's publication, all the supported keywords for the **CLIMATE** block allow for the use of the *IXJ Style* input that serves as an alternative input to the *Array Style* input. If *IXJ Style* input is used, then the FMP expects to load two integers (I) and one floating-point number (X) and does not read any subsequent integers (J). The floating-point number is the keyword property—such as the precipitation rate—and the two integers are the model row and column that the property is applied to. Figure 6.31 illustrates the use of *IXJ Style* input in the **CLIMATE** block.

Keyword List

Figure 6.32 presents the **CLIMATE** block with all its supported keywords, their expected input structure, and a commented explanation.

Keyword	DIMKEY		
	ByWBS	**ByCrop**	**ByIrrigate**
REFERENCE_ET	X		
POTENTIAL_EVAPORATION_BARE	X		
PRECIPITATION	X		
DIRECT_RECHARGE	X		

Figure 6.30. **CLIMATE** block keywords that support **SFAC DIMKEY**. The only supported word for **DIMKEY** is **ByWBS**. [Supported keyword and **DIMKEY** combinations are marked with an X.]

```
#  Example CLIMATE block for a model grid with NROW = 3, NCOL = 4
#  [L]   Model length units are in meters
#  [T]   Model time units are in days
#
BEGIN CLIMATE
  #
  #  PRECIPITATION keyword with INTERNAL automatically implies STATIC
  #      and, because the only INPUT_STYLE supported is Array style (ARRAY or IXJ),
  #      so if not specified implies that the input follows ARRAY as an input.
  #      To use IXJ style, then the keyword IXJ must be specified.
  #
  PRECIPITATION INTERNAL
  0.50  0.68  0.81  0.75
  0.55  0.72  0.82  0.77
  0.52  0.73  0.83  0.79
  #
  #
  #  DIRECT_RECHARGE keyword uses IXJ Style input to specify recharge
  #      as deep percolation at three model cell locations.
  #
  #  Note that IXJ Style is a surrogate for Array Style input.
  #      That is, the array is specified by row and column reference rather than as a full array.
  #
  DIRECT_RECHARGE STATIC IXJ INTERNAL
  # ROW   COL    Recharge Rate
    2     4      1.E3
    3     4      50.5
    3     3      75.0
  STOP IXJ  # Stop loading IXJ input
  #
END CLIMATE
```

Figure 6.31. CLIMATE block example with Array Style and IXJ Style inputs for a 3-by-4 model grid. [NROW is the number of model rows; NCOL is the number of model columns; COL is shorthand for model column.]

```
BEGIN CLIMATE
 #
 # Recommended Keyword ----------------------------------------------
 #
 PRECIPITATION   LAI[S, T, A]   # Specify precipitation flux over model domain
 #
 # Conditionally Required Keywords ----------------------------------
 #
 REFERENCE_ET   LAI[S, T, A] # Specify reference evapotranspiration flux over model domain
 #
 #  Specify potential evaporation flux for model cells with undefined land use (bare soil area)
 #    If not specified, then set to REFERENCE_ET × REFERENCE_ET_TO_BARE
 POTENTIAL_EVAPORATION_BARE   LAI[S, T, A]
 #
 # Advanced Keywords ------------------------------------------------
 #
 DIRECT_RECHARGE {FLUX, RATE} LAI[S, T, A, L] # Additional recharge added to deep percolation
 #
 # Multiplier to convert REFERENCE_ET to POTENTIAL_EVAPORATION_BARE, if not specified default is 0.5
 REFERENCE_ET_TO_BARE FLOAT
 #
 # Specify potential consumption limit for precipitation (converts precip to effective precip).
 PRECIPITATION_POTENTIAL_CONSUMPTION {BY_LENGTH, BY_FRACTION} LAI[S, T, A]
 #
END CLIMATE
```

Figure 6.32. CLIMATE block with all supported keywords and their input format. [FLOAT, floating-point number; LAI, list-array input with the following letters representing the supported keywords: S for STATIC, T for TRANSIENT, and A for ARRAY]

Soil Block

The **SOIL** block is conditionally optional and specifies the Soil ID location, soil capillary fringe, and soil coefficients. The number of soil types is defined external to the **SOIL** block by the Global Dimension block variable, **NSOIL**, and is the number of records read with *List Style*. Figure 6.33 presents the four keywords that are available in the **SOIL** block.

The **SOIL_ID** array is only required if **NSOIL** is greater than 1. If **SOIL_ID** is not specified and **NSOIL** is equal to 0 or 1, then it is automatically set to 1 for the entire model grid. The **CAPILLARY_FRINGE** specifies the distance above the water table that the capillary fringe extends to. The water table elevation plus the capillary fringe defines a threshold elevation used to determine whether there is (1) the potential for groundwater evaporation when the threshold is above the **SURFACE_ELEVATION** and (2) the potential for a land use to have groundwater-root uptake—groundwater consumption—when the threshold is above the bottom of the root zone. Appendixes 4 and 5 describe in detail the determination of groundwater evaporation when the water table is between ground surface and the extinction depth and of crop root uptake under variably saturated conditions. The **CAPILLARY_FRINGE** may be specified with *Array Style*—in which case it does not require nor use **SOIL_ID**—to specify the capillary fringe for the surface layer of the model grid or FMP may use *List Style* to read **NSOIL** records that apply the capillary fringe on the basis of the **SOIL_ID** array—for example, record 1 is applied to every cell containing a 1 in the **SOIL_ID** array. Figure 6.34 illustrates the relationship between **SOIL_ID** and **CAPILLARY_FRINGE** when using *List Style* input and *Array Style* input.

A

Keyword	Input	Description
SOIL_ID	ULOAD	**INT** Array whose value is between 0 and **NSOIL** to indicate the location of each soil type. A Soil ID of 0 is ignored. If **NSOIL**=1, then **SOIL_ID** is optional; if not specified, then the entire model grid is automatically set to a **SOIL_ID** of 1.
CAPILLARY_FRINGE	LAI[S, A, L]	**FLOAT** Array or List (length = **NSOIL**) that specifies the capillary fringe length above the water table [L]. The capillary fringe in the FMP is the length above saturated groundwater in which water is held by surface tension and can evaporate if near the land surface or be transpired if near a crop's roots. For groundwater evaporation to occur, the groundwater head must be greater than the land-surface elevation minus the capillary length. For groundwater transpiration—direct uptake of groundwater by a crop— to occur, the groundwater head must be greater than the land-surface elevation minus the crop's root length minus the capillary length. *Array Style* specifies each model cell's capillary fringe and *List Style* uses the **SOIL_ID** array to assign the **NSOIL** capillary fringes.

Figure 6.33. SOIL block keyword list. Keywords enclosed in curly braces, {}, indicate that only one of the keywords may be used during a simulation. *A*, Required keywords. *B*, Advanced keywords. [Input that is not separated by an or is specified on the same line. Abbreviations: FMP, farm process; ID, water balance subregion identification number; WBS, water-balance subregion; INT, integer; [L], length in model units; [L/T], length per time in model units.]

B

Keyword	Input	Description
COEFFICIENT	LAI[S, L] or LAI[S, L-5]	Soil coefficients used in the analytical, pseudo steady-state soil-moisture, soil-stress concept (*analytical root response*) land-use calculation. Required if **LAND_USE** block specifies **ROOT_PRESSURE**. *List Style* reads for each record one of the following secondary keywords: **SILT**, **SILTYCLAY**, **SANDYLOAM**, and **SAND** or five **FLOAT** numbers that represent the coefficients A, B, C, D, and E defined in appendix 6.
EFFECTIVE_PRECIPITATION_TABLE	{**BY_LENGTH**, **BY_FRACTION**} LAI[S, L]	Reads **NSOIL** Lookup Tables that specify the relationship between **PRECIPITATION** and effective precipitation. This is applied as a limitation on the potential consumption of precipitation by the landscape to satisfy evaporation and transpiration. Precipitation that is beyond limit becomes runoff or deep percolation. If not specified, then there is no limit and actual evapotranspiration can potentially consume all the precipitation. Each table is loaded with *Lookup*-Style Input. The "lookup values" are precipitation rates in [L/T]. Effective precipitation is the "return value" of the table. The **BY_LENGTH** keyword indicates that the return value is [L/T] and any precipitation in excess of this is not available for consumption. The **BY_FRACTION** keyword indicates that the return value is a fraction between 0 and 1 and is the fraction of the precipitation that is effective.

Figure 6.33. —Continued

A

```
#  Example use of the SOIL block keyword SOIL_ID
#
#  NROW = 3, NCOL = 4, NSOIL = 3; [L] has model length units in meters
#
#  SOIL_ID defines the spatial location of each soil type.
#  SOIL_ID is read with ULOAD to read a 3 by 4 array, INTERNAL indicates it is on subsequent lines.
SOIL_ID  INTERNAL
1  1  2  2
1  1  2  2
1  1  3  3
```

B

```
#  CAPILLARY_FRINGE defines the capillary fringe length above the water table [L]. Input is LAI[S, A, L]
#  The keyword STATIC is optional because it is the only TERMPORAL_KEY
#  ARRAY or LIST must be specified to indicate the input structure.
#
#  This example illustrates the use of LIST-style input to load NSOIL=3 records
#
CAPILLARY_FRINGE  STATIC LIST INTERNAL
# ID    Capillary Fringe Length                    ↓ Resulting Model Grid ↓
  1    1.2                                  #  1.2   1.2   1.8   1.8
  2    1.8                                  #  1.2   1.2   1.8   1.8
  3    1.5                                  #  1.2   1.2   1.5   1.5
#
```

C

```
#  This example illustrates the use of ARRAY-style input to load a 3 by 4 array of capillary fringe
lengths
#
#  This example also includes the advanced scale factor, SFAC BySoil, to scale the array by soil type
#    SFAC BySoil indicates that soil 1 is scaled by 1.1, soil 2 by 1.2, soil 3 by 1.3
#    Note that if SFAC BySoil was not present, the capillary fringe is identical to the LIST version in (B)
#
CAPILLARY_FRINGE  STATIC ARRAY INTERNAL
SFAC BySOIL  1.1  1.2  1.3                  #↓        SFAC result        ↓
1.2  1.2  1.8  1.8                         #  1.32   1.32   2.16   2.16
1.2  1.2  1.8  1.8                         #  1.32   1.32   2.16   2.16
1.2  1.2  1.5  1.5                         #  1.32   1.32   1.95   1.95
```

Figure 6.34. The relationship between the SOIL block keywords SOIL_ID and CAPILLARY_FRINGE. *A*, Define the soil type location. *B*, Define the capillary fringe with List Style input. *C*, Define the capillary fringe with Array Style input with an advanced scale factor, **SFAC**, using the BySoil **DIMKEY**. [**DIMKEY**, Dimensional keyword; BySoil, indicates NSOIL scale factors are read; NSOIL is the number of soil types; NROW is the number of model rows; NCOL is the number of model columns; LAI, list-array input with the following letters representing the supported keywords: S for STATIC, T for TRANSIENT, A for ARRAY, and L for LIST]

The **EFFECTIVE_PRECIPITATION_TABLE** keyword is optional and limits how much precipitation can be consumed as evaporation and transpiration (Consumptive Use). The only supported input is *List Style*, which reads **NSOIL** record IDs followed by *Lookup* Style input. The return value for **EFFECTIVE_PRECIPITATION_TABLE** is analogous to effective precipitation, which represents the quantity of rainfall that becomes neither runoff nor deep percolation. It is different from effective precipitation because there can be additional runoff and deep percolation if the landscape's consumptive use is less than the potential consumption of precipitation—that is, the simulated actual evapotranspiration does not consume all the available precipitation. If not specified, then the potential effective precipitation—potential consumption—is the same as the precipitation. The actual consumption of precipitation is either the effective precipitation or the actual evapotranspiration less any groundwater-root uptake for transpiration, whichever is smaller. The limit can be defined as a length—**EFFECTIVE_PRECIPITATION_TABLE BY_LENGTH**—or as a fraction—**EFFECTIVE_PRECIPITATION_TABLE BY_FRACTION**. If defined as a length, then only precipitation up to the specified length (L/T) is available for consumption. If defined as a fraction, then it is multiplied by the precipitation to get the effective precipitation length. If the lookup table returns an effective precipitation that is negative, then it is set to zero. Conversely, if the lookup table returns an effective precipitation larger than the actual precipitation, then it is reset to the actual precipitation. To disable limiting the effective precipitation, set the length to 1E30—or any value larger than the actual precipitation. Figure 6.35 is an example that specifies lookup tables of effective precipitation for two soil types and uses a constant for a third soil type.

A

```
#  Example use of the SOIL block keyword EFFECTIVE_PRECIPITATION_TABLE
#
#  NSOIL = 3; [L/T] has model length units in centimeters and time units in days
# Soil 1 and 2 refer to lookup tables in external files
# Soil 3 contains a large number so the effective precipitation is equal to precipitation.
#
#  EFFECTIVE_PRECIPITATION_TABLE specify lookup tables that define the
#     relationship between precipitation and effective precipitation as a lookup table.
#     BY_LENGTH indicates input is expected to be in units of [L/T]
#  ULOAD reads NSOIL Lookup Style input, INTERNAL indicates it is on subsequent lines.
#
EFFECTIVE_PRECIPITATION_TABLE BY_LENGTH INTERNAL
#ID METHOD        NTERM  GENERIC_INPUT
 1 INTERPOLATE   0      TAB1.txt   #  Load table in TAB1.txt; 0 indicates to auto-count rows
 2 INTERPOLATE   5      TAB2.txt   #  Load table in TAB2.txt; Table has 5 rows
 3 CONSTANT 1E30                   #  Table set to a large constant value
```

B

```
#  File: TAB1.txt
#  Values chosen are for illustrative purposes
#  and should not be used in practice
#  LookUp         ReturnValue
   5.0             1.0
  10.0             5.0
  15.0            10.0
  20.0            15.0
  25.0            20.0
# End of file; auto-count sets NTERM = 5
```

C

```
#  File: TAB2.txt
#  Values chosen are for illustrative purposes
#  and should not be used in practice
#  LookUp         ReturnValue
   5.0            10.0
  10.0            10.0
  15.0            10.0
  20.0            15.0
  20.1            1E30
```

Figure 6.35. SOIL block EFFECTIVE_PRECIPITATION_TABLE keyword example that reads using Lookup Style two lookup tables. *A*, Setup required to define EFFECTIVE_PRECIPITATION_TABLE. *B*, File TAB1.txt that contains a lookup table. *C*, File TAB2.txt that contains a lookup table. [[L/T] length per time in model units; NSOIL is the number of soil types; ULOAD is the universal loader]

In figure 6.35, the lookup table defined in TAB2.txt (fig. 6.35*C*) has precipitation lookup values `5`, `10`, and `20.1`, which are less than or equal to their associated effective precipitation (for example, `5<10`). For these three rows the consumption of precipitation is not limited (effective precipitation cannot be larger than actual). For example, a precipitation value of `6` has an effective precipitation of `10`, which is greater than the actual precipitation, so the available precipitation is `6`. Any precipitation greater than `20` has the effective precipitation greater than the actual, so the effective precipitation is always set equal to the actual. For example, precipitation of `20.001` returns an effective precipitation of `1E28`, which is greater than `20.001`, so the available precipitation for consumption is `20.001`.

It should be noted that the effective precipitation only serves as a potential consumption. The effective precipitation may not fully be consumed by the land use. This can occur when the consumptive use is less than the effective precipitation or the consumptive use is satisfied by direct groundwater uptake. Figure 6.36 presents three example cases to illustrate how the **`EFFECTIVE_PRECIPITATION_TABLE`** can limit the consumption. In figure 6.36, the *Potential-Actual Consumptive Use* represents the actual consumptive use [L/T] if there is enough water supply. The *FMP Calculated Transpiration from Groundwater* is determined on the basis of the water-table location, the soil properties, and the root depth. The *FMP Calculated Potential Consumption of Precipitation* is the consumption of precipitation from evaporation and transpiration if there is no limit imposed. The *Potential-Actual Consumption of Precipitation by Evaporation and Transpiration* is the actual maximum consumption of precipitation by evaporation and transpiration given enough precipitation. Each test case shows, in bold, which input becomes the limiting factor for the consumption of precipitation.

The use of the **`EFFECTIVE_PRECIPITATION_TABLE`** keyword is not recommended, unless FMP simulation results present concerns with overconsumption of precipitation. This could occur when a stress period has a long time span compared to the total time that precipitation fell during the stress period. For example, a 30-day stress period that has 1 day with `0.3` meter (m) of precipitation and no precipitation for the rest of the period would have an input of `0.01` meter per day (m/d). The precipitation from a single event is spread out across the month resulting in overconsumption. An example limit on the consumption could be specified by including one of the post-keywords—**`BY_LENGTH`** of `0.002` m or equivalently as **`BY_FRACTION`** of `0.1`. If there is a lack of data about a simulation area and the model tends to overconsume precipitation, then figure 6.37 (Brouwer and Heibloem, 1986) provides rough effective precipitation estimates for a given precipitation per month. Additional empirical methods, including the U.S. Department of Agriculture Soil Conservation Service (USDA SCS) method, for determining effective precipitation are described in Dastane (1978).

If it is desired to use the analytical pseudo steady-state soil moisture, soil-stress concept called *analytical root response* (appendixes 4 and 5), then the soil **`COEFFICIENT`** and the **`LAND_USE`** block keyword **`ROOT_PRESSURE`** must be specified. The soil **`COEFFICIENT`** may be specified using keywords that have predefined soil coefficients or by directly specifying the five soil coefficients (`A`, `B`, `C`, `D`, and `E`). Figure 6.38 presents the soil **`COEFFICIENT`** keywords and their coefficients. Figure 6.39 is an example that specifies the **`SOIL_ID`** and **`COEFFICIENT`** for three soil types.

	Case 1	Case 2	Case 3
FMP Calculated Potential-Actual Consumptive Use [L/T]	1.0	1.0	1.0
FMP Calculated Transpiration from Groundwater [L/T]	0.8	0.3	0.0
FMP Calculated Potential Consumption of Precipitation [L/T]	1.0 – 0.8 = **0.2**	1.0 – 0.3 = 0.7	1.0 – 0.0 = 1.0
FMP Input Precipitation [L/T]	0.4	**0.4**	0.6
Effective Precipitation Returned from the Lookup Table [L/T]	0.5	0.5	**0.5**
Potential-Actual Consumption of Precipitation by Evaporation and Transpiration [L/T]	**0.2**	**0.4**	**0.5**

Figure 6.36. Three example cases that illustrate the effect of the effective precipitation. [Effective precipitation is determined from either the SOIL block EFFECTIVE_PRECIPITATION_TABLE or CLIMATE block PRECIPITATION_POTENTIAL_CONSUMPTION keywords. The use of these keywords is not recommended unless simulation results present concerns with overconsumption of precipitation. This may occur when the number of days of precipitation is small compared to the duration of a stress period and the potential consumptive use is large for the stress period. Abbreviations: FMP, farm process; [L/T], length per time in model units]

Precipitation (mm/month)	Effective Precipitation (mm/month)	Effective Fraction
10	0	0.00
20	2	0.10
30	8	0.27
40	14	0.35
50	20	0.40
60	26	0.43
70	32	0.46
80	39	0.49
90	47	0.52
100	55	0.55
110	63	0.57
120	71	0.59
130	79	0.61
140	87	0.62
150	95	0.63
160	103	0.64
170	111	0.65
180	119	0.66
190	127	0.67
200	135	0.68
210	143	0.68
220	151	0.69
230	159	0.69
240	167	0.70
250	175	0.70

Figure 6.37. Estimated relation of precipitation to effective precipitation from Brouwer and Heibloem (1986). [These values may be used as an estimate for the SOIL block keyword EFFECTIVE_PRECIPITATION_TABLE. The effective precipitation may be used as the BY_LENGTH option and effective fraction may be used as the BY_FRACTION option. The expected units for this are in millimeter per month (mm/month), whereas units in the farm process (FMP) are [L/T], which is length per time in model units].

COEFFICIENT Keyword	Soil Coefficients
SILT	A = 0.3201 B = -0.3286 C = 2.8519 D = 1.3027 E = -2.0416
SANDYLOAM	A = 0.2007 B = -0.1955 C = 3.0831 D = 3.2012 E = -3.9025
SILTYCLAY	A = 0.3481 B = -0.3274 C = 1.7308 D = 0.5298 E = -3.9025
SAND	A = 0.1100 B = -0.0590 C = 3.0900 D = 4.3260 E = -4.0990

Figure 6.38. Analytical, pseudo steady-state soil-moisture, soil-stress concept soil coefficients A, B, C, D, and E for the predefined COEFFICIENT keywords.

```
#  Example use of the SOIL block keywords SOIL_ID and COEFFICIENT
#
#  NSOIL = 3
#  SOIL_ID indicates the location that the soil coefficients are applied to
#
SOIL_ID  INTERNAL
1  1  2  2
1  1  2  2
1  1  3  3
#
#  COEFFICIENT uses ULOAD to read NSOIL List Style records that
#    either specify a soil type keyword: SILT, SANDYLOAM, SILTYCLAY, or SAND
#    or directly specify the soil type coefficients A, B, C, D, and E.
#  ULOAD uses INTERNAL to indicate input is on subsequent lines
#
COEFFICIENT  INTERNAL
1  SANDYLOAM
2  0.3201  -0.3286  2.8519  1.3027  -2.0416  # coefficients are equivalent to "SILT"
3  SILTYCLAY
```

Figure 6.39. SOIL block COEFFICIENT keyword example. [NSOIL is the number of soil types; ULOAD is the universal loader]

Supported SFAC DIMKEY

The **SOIL** block keywords that support **SFAC** **DIMKEY**s are presented in figure 6.40. Only the keywords that support **SFAC** **DIMKEY** are specified, and an "X" is placed under the respective **DIMKEY**. The only keyword that supports a **SFAC** keyword is **CAPILLARY_FRINGE**, and it only supports the keyword "**BySoil**" that loads **NSOIL** scale factors.

Keyword List

Figure 6.41 presents the **SOIL** block with all its supported keywords, their expected input structure, and a commented explanation.

Supply Well Block

The **SUPPLY_WELL** block is an optional block that specifies the properties of groundwater wells that extract or inject water into the groundwater flow process. Supply wells do not specify a pumping rate, but instead specify a maximum pumping capacity (QCAP) of the well, and the pumping rate is dynamically assigned on the basis of satisfying water demand up to the maximum rate, QCAP. The **SUPPLY_WELL** block can be superseded by the **WATER_BALANCE_SUBREGION** block keyword **WATERSOURCE**, which can disable all the supply wells for a WBS during a stress period. In FMP, the block name can be declared as **SUPPLY WELL** or **SUPPLY_WELL**, but the later version, with an underscore, is recommended for clarity.

Supply wells can either be specified as a single model cell—defined by a layer, row, and column—or linked to wells defined in the MNW2 package, which can represent multiple cells. The FMP supply wells that are linked to MNW2 rely on the MNW2 package to perform the simulated pumping, and the FMP determines the desired pumping rate. The FMP-MNW2 linked wells must have their construction information—such as well radius, screen interval, and location—specified in MNW2 input, but the desired pumping rate is assigned by the FMP on the basis of water demand (up to a user specified maximum capacity). The advantage of using MNW2 linked wells is that their desired pumping rate is dynamically set on the basis of water demand, but the simulated pumping rate is constrained by the well construction and antecedent aquifer conditions.

Keyword	DIMKEY			
	ByWBS	ByCrop	ByIrrigate	BySoil
CAPILLARY_FRINGE				X

Figure 6.40. SOIL block keywords that support **SFAC** **DIMKEY**. Only one keyword provides support, and only for one **DIMKEY**—BySoil. [Supported keyword and **DIMKEY** combinations are marked with an X.]

```
BEGIN SOIL
 #
 #  Specifies NSOIL soil types' (integer soil IDs) grid locations.
 #  Input is read with ULOAD to load NROW × NCOL array of INT values.
 #      Optional if NSOIL = 1
 SOIL_ID   ULOAD
 #
 #  Specifies capillary fringe length for the soil. If not specified, then set to zero.
 #  List Style input uses SOIL_ID list to assign the spatial location for the capillary fringe values
 CAPILLARY_FRINGE   LAI[S, A, L]
 #
 #
 DIRECT_RECHARGE {FLUX, RATE} LAI[S, T, A, L]  # Additional recharge added to deep percolation
 #
 #  Only required if LAND_USE Block specifies ROOT_PRESSURES.
 #  In place of specifying the five coefficients for a soil type,
 #      input may use one the predefined keywords SILT, SILTYCLAY, SANDYLOAM, and SAND
 COEFFICIENT LAI[S, L-5]
 #
 # Define for each soil type a precipitation-to-effective precipitation lookup table.
 EFFECTIVE_PRECIPITATION_TABLE {BY_LENGTH, BY_FRACTION} LAI[S, L]
 #
END SOIL
```

Figure 6.41. SOIL block with all supported keywords and their input format. [INT, integer number; NROW is the number of model rows; NCOL is the number of model columns; ULOAD, universal loader; LAI, list-array input with the following letters representing the supported keywords: S for STATIC, T for TRANSIENT, and A for ARRAY]

The **SUPPLY_WELL** block includes three possible input formats (fig. 6.42) to define the spatial location, maximum capacity, and construction of the supply wells. Only one of the three input formats may be used during a simulation. They have the same basic input properties (fig. 6.43), but they differ in how they define the transient (stress period) information. The **SUPPLY_WELL** block must include one of the three input format keywords (fig. 6.42), then on subsequent input lines contain a *Generic_Input* file, one per line, and continues until the next **SUPPLY_WELL** keyword is found or **END SUPPLY_WELL** is reached. This allows for **TIME FRAME** or **LINEFEED** input to be split into multiple files (see appendix 2 for how multiple LineFeed files are handled).

Input Format Keyword	Description
TIME FRAME	Supply well spatial and temporal description are read as one well per line—same well may be repeated multiple times for different time intervals. Each row of input specifies a starting and ending stress period—or date—for when the well is continuously available to provide groundwater supply. This is the most flexible and recommended input style.
LINEFEED WBS	Supply well specified with LineFeed format (appendix 2). The LineFeed temporal section (input read every stress period) specifies the water balance subregion (WBS) that the supply well is associated with. Set the WBS to `0` to disable the well from providing supply for that stress period. This option is best suited when supply wells frequently change the WBS that they provide supply to.
LINEFEED CAPACITY	Supply well specified with LineFeed format (appendix 2). The LineFeed temporal section (input read every stress period) specifies the maximum capacity (QCAP) of the well. Set the QCAP to NaN or `0` to disable the well from providing supply for that stress period. This option is best suited when supply wells have frequently changing maximum capacities.

Figure 6.42. SUPPLY_WELL block keywords that initiate the main supply-well input. [Only one of these keywords may be present during a simulation. Abbreviation: WBS, water-balance subregion]

Supply Well Property	Data type and Description
Well_Name	Name that identifies supply well. It may be any alpha-numeric character with a max length of 20 characters. If supply well is linked to MNW2, then this name must match a MNW2 WELLID, which is the MNW2 well that it is linked to. The name does not have to be unique, but it must be unique among the supply wells in use during a stress period.
WBS	**INT** is the WBS ID that the supply well provides water to. (Desired well pumpage is determined by the WBS demand.)
LAY	**INT** is the layer number of the supply well. If set to "MNW" or 0, then well is linked to MNW2 and Well_Name must match a MNW2 WELLID. (Desired pumping rate is assigned to MNW2 based on the WBS demand.) **INPUT_OPTION** keyword may change LAY to be specified as a **FLOAT**, and to represent either a depth from land surface or elevation.
ROW	**INT** is the row number of the supply well. If well is linked to MNW2, this number is read, but ignored. If the keyword "**INPUT_OPTION XY**" is present, then instead of reading the row number, ROW is read as a **FLOAT** and represents the well's X-Coordinate in the MF-OWHM2 coordinate space.
COL	**INT** is the column number of the supply well. If well is linked to MNW2, this number is read, but ignored. If the keyword "**INPUT_OPTION XY**" is present, then instead of reading the column number, COL is read as a **FLOAT** and represents the well's Y-Coordinate in the MF-OWHM2 coordinate space.
QCAP	**FLOAT** is the maximum pumping capacity of the well [L^3/T]. (Based on the WBS demand, the FMP assigns a desired pumping rate between 0 and QCAP.)

Figure 6.43. SUPPLY_WELL block required base-input properties necessary to define a supply well. [ID, water balance subregion identification number; WBS, water-balance subregion; INT, integer number; FLOAT, floating-point number; [L^3/T], volumetric rate in model units; MNW2, multi-node well package version 2; WELLID is a well name defined in MNW2]

Multi-Node Well Package Version 2 (MNW2) Supply-Well Linkage

To set up an FMP-MNW2 supply-well link it is required to have the MNW2 package enabled and its input set up. At a minimum the MNW2 input (Konikow and others, 2009) must specify all data items defined in the "For Each Simulation" input group—that is, any input that appears before the stress-period data input item `ITMP`, which begins the "For Each Stress Period" datasets. The "For Each Simulation" input identifies each MNW2 with a unique alphanumeric identification label called `WELLID`, which is used by the FMP for locating linked wells. The FMP-MNW2 linkage happens when the FMP supply well has `LAY` set to `0` or "`MNW`" and the FMP input `Well_Name` is identical to a MNW2 `WELLID`.

Additionally, the FMP-MNW2 link requires that the MNW2 well be "active" during the stress periods that the FMP uses it as a supply well. The MNW2 documentation (Konikow and others, 2009) defines a MNW2 as being "active" if it is specified as part of the "For Each Stress Period" `ITMP` well list. An "active" well may extract water, inject water, or remain idle with no pumping. An idle "active" well is still simulated, and if the well extends across multiple layers can include intraborehole flow. The FMP-MNW2 linked wells should always specify the desired pumping (`QDES`) in MNW2 input as zero—because `QDES` is automatically set by the FMP. It also is recommended to have the FMP-MNW2 linked wells made "active" when they are drilled and then "not active"—removed from the simulation—when a well is destroyed. This is not necessarily the same as the period that the well provided supply and allowing it to be idle when not in use. The supply period is defined in the FMP input part (fig. 6.42), which defines when the FMP is allowed to dynamically set the value of `QDES`.

An alternative to directly specifying when MNW2 wells are "active" is to include in the **`SUPPLY_WELL`** block the keyword **`MNW_AUTOMATIC_ON`**. This keyword enables the FMP to automatically set to "active" any MNW2 wells that are not already declared "active" during stress periods that the FMP inputs had defined for using it as a supply. For example, if an FMP supply well is defined as being active and available to provide a supply in stress period 3, then on the third stress period, the FMP checks if the corresponding `WELLID` is "active," and, if not, the **`MNW_AUTOMATIC_ON`** keyword enables it. The MNW2 wells that are made "active" through **`MNW_AUTOMATIC_ON`** remain "active" for the remainder of the simulation and have `QDES` set to zero for stress periods when the FMP does not use them. Conversely, the **`SUPPLY_WELL`** block keyword **`MNW_AUTOMATIC_OFF`** allows the FMP to automatically deactivate, that is, remove from the simulation, an "active" MNW2 well when the FMP supply well is no longer needed to provide supply. The **`MNW_AUTOMATIC_OFF`** option is not suitable for simulating areas where wells are typically abandoned rather than destroyed.

If **`MNW_AUTOMATIC_ON`** is not specified, then the FMP-MNW2 linked wells must be included in the MNW2 "For Each Stress Period" input or specified with LineFeed for the stress periods that they are in use. This is done by including the FMP `Well_Name` as the `WELLID` in the FMP input for the appropriate stress period corresponding to `ITMP` in the MNW2 package input and setting `QDES` to zero. This method is advantageous if the supply well's drilled and destroyed dates are different than the period that they provide supply. Figure 6.44 is an example MNW2 input file used for a simulation of four stress periods (SP) that defines four wells: (1) Industrial1, (2) Industrial2, (3) FMP_MNW_1, and (4) FMP_MNW_2. The wells Industrial1 and Industrial2 start on stress period 1 and have their pumping rate specified by MNW2. The wells FMP_MNW_1 and FMP_MNW_2 are linked in the FMP input and are first "active" in stress periods 2 and 3, respectively. Industrial2 and FMP_MNW_2 are assumed to be destroyed by stress period 4—they are not specified in the list following `ITMP`. Note that in figure 6.44 the corresponding FMP input is not shown.

Time Frame Input Structure

The **`TIME FRAME`** input structure is the recommended method for specifying supply wells in the FMP. The supply-well input specifies the base properties on each line defined in figure 6.43 and then provides starting and ending stress periods—or calendar dates—for when the well is available for pumping. Figure 6.45 is the **`TIME FRAME`** input structure where the supply well is continuously available to supply pumpage between `Date_Start` and `Date_End`. During this interval, a supply well can be disabled at any stress period that has the **`WATER_BALANCE_SUBREGION`** block keyword **`WATERSOURCE`** set to disable groundwater pumping. If a change is needed in the association between a supply well and the WBS that it supplies, then it must be specified twice, and the two input statements will have different values for `Date_Start` and for `Date_End`.

The next line after the keyword **`TIME FRAME`** is *Generic_Input* that specifies where the supply-well input is. Multiple *Generic_Input* files may be specified with one *Generic_Input* file per line. Only one *Generic_Input* is allowed if it is specified with keyword `INTERNAL`, which indicates that the subsequent lines contain the supply-well input. The FMP auto-counts the number of supply wells that are defined in the *Generic_Input* file. If the input is directly within the block (*Generic_Input* is defined as `INTERNAL`), then the auto-counting determines the number of wells on the basis of the number of lines (excluding those that are blank or commented with a "#" symbol) between the keyword **`TIME FRAME`** and either the next **`SUPPLY_WELL`** keyword or **`END SUPPLY_WELL`**, whichever is first. If the input is in a separate file (*Generic_Input* is not defined as `INTERNAL`), then the number of wells is the number of lines in the file that are non-blank and not commented with a "#" symbol.

```
#  For FMP to link to an MNW2 well, the FMP Well_Name must match the MNW2 WELLID.
#  FMP_MNW_1 and FMP_MNW_2 are assumed to be linked to the FMP SUPPLY_WELL block input.
#  Note that the FMP-MNW WELLID can be a 20-character string, but FMP_MNW_ is chosen for clarity.
#  In this example,  FMP can only use for supply FMP_MNW_1 during stress periods (SP) 2, 3, and 4.
#                    FMP can only use for supply FMP_MNW_2 during SP 3
#  If the FMP SUPPLY_WELL block has the keywords MNW_AUTOMATIC_ON and MNW_AUTOMATIC_OFF,
#  then the FMP-MNW linked wells are automatically changed to “active” or “inactive”
#  when the FMP input indicates they are available for supply.
#  This negates the need to define the wells as part of the ITMP portion of this input.
#
#  MNW Construction Part – For Each Simulation
4 0 2                    # MNWMAX  IWL2CB     MNWPRNT
Industrial1  -1          # WELLID     NNODES
  SKIN  0 0 0 0          # LOSSTYPE   PUMPLOC  Qlimi   PPFLAG  PUMPCAP
  0.1  0.35  1.6683      # Rw         Rskin     Kskin
  -75 -50  6   2         # Ztop       Zbot      ROW   COL
Industrial2  -1
  SKIN  0 0 0 0
  0.1  0.35  1.6683
  -70 -55  8   2
FMP_MNW_1  -1
  SKIN  0 0 0 0
  0.1  0.35  1.6683
  -70 -55  10   2
FMP_MNW_2  -1
  SKIN  0 0 0 0
  0.1  0.35  1.6683
  -70 -55  3   3
# For Each Stress Period
  2                      # ITMP - SP 1
  Industrial1   50.0     # WELLID    QDES – Industrial wells "active"
  Industrial2  250.0     # WELLID    QDES
  3                      # ITMP - SP 2
  Industrial1   50.0
  Industrial2  250.0
  FMP_MNW_1      0.0     # FMP linked well now "active", FMP may use it for supply.
  4                      # ITMP - SP 3
  Industrial1   50.0
  Industrial2  250.0
  FMP_MNW_1      0.0
  FMP_MNW_2      0.0     # FMP linked well now "active", FMP may use it for supply.
  2                      # ITMP - SP 4
  Industrial1   50.0     #
  FMP_MNW_1      0.0     # FMP_MNW_2 isn’t specified, making it “inactive”, FMP may not use it for supply
```

Figure 6.44. Example MNW2 input file that contains two wells that are linked to FMP for use as supply wells. These wells can provide supply only if, they are “active” in MNW2 and the FMP SUPPLY_WELL block indicates they are available to provide supply. [MNW2, multi-node well package version 2; FMP, farm process]

A

```
Well_Name  WBS  LAY  ROW  COL  QCAP  Date_Start  Date_End  [xyz]
```

B

Input Format Keyword	Description
Date_Start	Is the starting stress period or date that a supply well is available to provide water to its water-balance subregion (WBS). A date is converted to a stress period by selecting the one that either contains the date or is closest to the date. Dates may be formatted using International Standard Organization (ISO) standard (year-month-day, yyyy-mm-dd) or American style (mm/dd/yyyy) as defined in the appendix 3, "Calendar Dates" section.
Date_End	Is the ending stress period or date that a supply well is available to provide water to its WBS. A date is converted to a stress period by selecting the one that either contains the date or is closest to the date. If set to the word "NPER", then it is automatically set to the end of the simulation—that is, stress period NPER. Dates may be formatted using ISO standard (yyyy-mm-dd) or American style (mm/dd/yyyy) as defined in the appendix 3, "Calendar Dates" section.
XYZ	Represents auxiliary integer (**INT**) flags if an auxiliary option is specified. By default, there are no auxiliary flags read. They are only present when an auxiliary option is set. Multiple auxiliary flags may be required if multiple auxiliary options are in use.

Figure 6.45. SUPPLY_WELL block TIME FRAME general input structure and partial explanation of input variables. *A*, TIME FRAME general input structure. *B*, Explanation of select input variables. [INT, integer number]

Figure 6.46 is a set of simple examples of supply-well inputs using the **TIME FRAME** keyword. In figure 6.46*A*, FMP reads **INTERNAL** input to load the spatiotemporal data for three supply wells. It is a limitation of the **TIME FRAME** input structure that wells must be specified multiple times if either its WBS association or its maximum capacity changes though time. Figure 6.46*B* and figure 6.46*F* illustrate pros and cons of using the **TIME FRAME** option. For example, figure 6.46*B* specifies four lines of well inputs, but only three unique wells are simulated (that is, `Wel_2` is repeated), and in figure 6.46*F*, `WELLA` is repeated three times. Figure 6.46*C* is an example that defines the well input in a separate file (fig. 6.46*E*). Figure 6.46*D* defines the input as contained in three separate files and uses the FMP-MNW2 linked wells defined in figure 6.44 (that is, in fig. 6.46*F* the `Well_Name`s "`FMP_MNW_1`" and "`FMP_MNW_2`"). In figure 6.46*F*, FMP will determine the starting and ending stress period of the `FMP_WEL_1` on the basis of the provided calendar dates. Also in figure 6.46*F*, `WELLA` undergoes changes in its WBS association over time, so it must be repeated in the input table. In this example `WELLA` provides pumpage to WBS 1 from stress period 15 to 25, then to WBS 2 from stress period 36 to 45, and then to WBS 3 from stress period 50 to 55.

A

```
#  Example use of the SUPPLY_WELL block that only specifies the input option TIME FRAME
#  Example assumes NROW = 3, NCOL = 4, NWBS = 5; NPER = 100
#
#  The general input is
#                        TIME FRAME
#                        GENERIC_INPUT
#                        [GENERIC_INPUT]    <-  as many GENERIC_INPUTs as necessary
#
#  where GENERIC_INPUT indicates the location of the TIME FRAME input.
#        [GENERIC_INPUT], if present, indicates that multiple sets of input can be specified.
#  For example:
#                  TIME FRAME
#                  OPEN/CLOSE    WEL_SET1.txt
#                  OPEN/CLOSE    WEL_SET2.txt
#
#  This example specifies one set of input with the INTERNAL keyword (input on subsequent lines).
BEGIN SUPPLY_WELL
  #
  TIME FRAME
  #
  INTERNAL
  # Well_Name  WBS  LAY  ROW  COL   QCAP  Date_Start  Date_End
    Wel_1       3    2    3    2   1000.       1    NPER  # Entire Simulation
    Wel_2       2    1    2    1   1000.       5    NPER  # Stress period 5 to the end
    Wel_3       2    2    2    1   1000.       1    NPER  # Entire Simulation

  # Well count determined by the number of non-blank, uncommented lines between
  # TIME FRAME and END SUPPY_WELL
  #
END SUPPLY_WELL
```

B

```
BEGIN SUPPLY_WELL
  #  Wel_2 supplies water to WBS 3 from stress period 1 to 4, then it supplies water to WBS 2.
  #  Wel_3 is linked to MNW2, which must have defined a WELLID "Wel_3" in the MNW2 input
  TIME FRAME
  #
  INTERNAL
  # Well_Name    WBS  LAY  ROW  COL   QCAP  Date_Start  Date_End
    Wel_1         3    2    3    2   1000.       1    NPER
    Wel_2         3    1    2    1   1000.       1       4  # Supplies WBS 3
    Wel_2         2    1    2    1   1000.       5    NPER  # Supplies WBS 2
    Wel_3         2  MNW    0    0   1000.       1    NPER  # MNW2 Link
  #
END SUPPLY_WELL
```

Figure 6.46. Supply Well block TIME FRAME input examples. *A*, Example input from a single INTERNAL. *B*, Example input from a single INTERNAL that contains wells linked to MNW2. *C*, Example that specifies input in the file "FMP_SUPPLY_WELL.txt". *D*, Example that specifies input in three separate files. *E*, The file "FMP_SUPPLY_WELL.txt". *F*, The file "FMP_SUPPLY_WELL_Group2.txt". [Note that contents of file FMP_SUPPLY_WELL_Group3.txt are not included in this figure. Abbreviations: MNW2, multi-node well package version 2; WBS, water-balance subregion; NPER is the number of stress periods]

C

```
#  This example specifies the TIME FRAME input in a separate file.
#  GENERIC_INPUT specified as "OPEN/CLOSE ./FMP_SUPPLY_WELL.txt"
BEGIN SUPPLY_WELL
  #
  TIME FRAME
  #
  OPEN/CLOSE  ./FMP_SUPPLY_WELL.txt
  #
END SUPPLY_WELL
```

D

```
# Example specifies the TIME FRAME input in three separate files.
BEGIN SUPPLY_WELL
  #
  TIME FRAME
  #
  OPEN/CLOSE ./FMP_SUPPLY_WELL.txt          # First     GENERIC_INPUT
  OPEN/CLOSE ./FMP_SUPPLY_WELL_Group2.txt   # Second GENERIC_INPUT
  OPEN/CLOSE ./FMP_SUPPLY_WELL_Group3.txt   # Third     GENERIC_INPUT
END SUPPLY_WELL
```

E

```
#  File: FMP_SUPPLY_WELL.txt
#
# WBS 2 and 3 have wells, all wells linked to MNW2,
# MNW2 must have defined WELLIDs of "Wel_1", "Wel_2", and "Wel_3"
# Specifying ROW and COL is required but not used—any INT can be set there

# Well_Name  WBS   LAY  ROW  COL   QCAP  Date_Start  Date_End
  Wel_1       3   MNW   3    2   1000.       1    NPER
  Wel_2       2   MNW   2    1   1000.       5    NPER
  Wel_3       2   MNW   2    1   1000.       1    NPER
```

F

```
#  File: FMP_SUPPLY_WELL_Group2.txt
# Well_Name    WBS   LAYER  ROW   COL   QCAP    Date_Start     Date_End
  FMP_WEL_1    3     2      5     25    1000.   3/15/1992      1/31/2015
  FMP_MNW_1    2     MNW    2     5     1000.   2               4
  FMP_MNW_2    2     MNW    2     5     1000.   3               3
  WELLA        1     2      5     25    1000.   15             25
  WELLA        2     2      5     25    1000.   36             45
  WELLA        3     2      5     25    1000.   50             55
```

Figure 6.46. —Continued

LineFeed Water-Balance Subregion (WBS) Input Structure

The **LINEFEED WBS** input structure is beneficial if supply wells either undergo change frequently in their WBS association or are frequently enabled and disabled in a manner that is not well handled by the **WATER_BALANCE_SUBREGION** block keyword **WATERSOURCE** or the **TIME FRAME** keyword. This input relies on LineFeed, described in appendix 2, to load the WBS of the supply wells every stress period—that is, the FeedFiles' temporal input corresponds with the WBS. Any number of FeedFiles, specified with *Generic_Input*, may be specified on the lines after the keyword **LINEFEED WBS**; however, only one *Generic_Input* is allowed per line, and the keyword **INTERNAL** is not allowed. Figure 6.47 is an example that loads three LineFeed files, called FeedFiles.

The LineFeed's FeedFile structure has a "Spatial Input Section" and a "Temporal Input Section" (appendix 2). From the Spatial Input Section, the FMP reads the time invariant information pertaining the **LINEFEED WBS**. Figure 6.48 presents general input structure for the Spatial Input Section. The Spatial Input Section specifies as many wells as necessary until the keyword **TEMPORAL INPUT** is encountered to indicate that the Temporal Input Section (stress-period input) is on the subsequent lines. The temporal input specifies each supply well's WBS. If a supply well is not in use for a stress period, then its WBS must be set to either 0 or NaN. Figure 6.49 illustrates how to set up a **LINEFEED WBS** input FeedFiles.

```
BEGIN SUPPLY_WELL

  LINEFEED WBS
  #
  OPEN/CLOSE ./FWEL_FEEDFILE1.txt
  OPEN/CLOSE ./FWEL_FEEDFILE2.txt
  OPEN/CLOSE ./FWEL_FEEDFILE3.txt
  #
END SUPPLY_WELL
```

Figure 6.47. SUPPLY_WELL block LINEFEED WBS example that specifies three FeedFiles.

```
Well_Name   LAY   ROW   COL   QCAP   [xyz]
```

Figure 6.48. SUPPLY_WELL block LINEFEED WBS general input structure for the LineFeed's Spatial Input Section (time invariant input information).

A

```
#  Supply Well block LINEFEED WBS example FeedFile
#  Example assumes NROW = 3, NCOL = 4, NWBS = 5; NPER = 6
#
#  The FeedFile example is equivalent to the following TIME FRAME input:
#       Wel_1     3  2  3  2  1000.     1  NPER
#       Wel_2     2  1  2  1  1000.     5  NPER
#       Wel_3     2  2  2  1  1000.     1  NPER
#
# Spatial Input Section
#       LAY  ROW  COL    QCAP
Wel_1    2    3    2    1000.
Wel_2    1    2    1    1000.
Wel_3    2    2    1    1000.
#
TEMPORAL INPUT
#
# Wel_1  Wel_2  Wel_3
      3      0      2   # SP 1 – WBS assigned to each supply well
      3      0      2   # SP 2 – WEL_2 not available for supply
      3      0      2   # SP 3
      3      0      2   # SP 4
      3      2      2   # SP 5 – WEL_2 is available for supply
      3      2      2   # SP 6
```

B

```
#  The FeedFile example is equivalent to the following TIME FRAME input:
#  Wel_1     3       2  3  2 1000.     1  NPER
#  Wel_2     3       1  2  1 1000.     1    4
#  Wel_2     2       1  2  1 1000.     5  NPER
#  Wel_3     2  MNW  0   0  1000.     1  NPER
#
# Spatial Input Section
#       LAY  ROW  COL   QCAP
Wel_1    2    3    2   1000.
Wel_2    1    2    1   1000.
Wel_3  MNW   0    0   1000.   # Wel_3 is linked to MNW2
#
TEMPORAL INPUT
#
# Wel_1   Wel_2   Wel_3
      3       3       2   # SP 1 – WBS assigned to each supply well
      3       3       2   # SP 2 – Wel_3 can provide supply only if "active" in MNW2
      3       3       2   # SP 3
      3       3       2   # SP 4
      3       2       0   # SP 5 – No supply from Wel_3 irrelevant of MNW2 status
      3       2       0   # SP 6
```

Figure 6.49. FeedFile examples for the SUPPLY_WELL block LINEFEED WBS. *A*, FeedFile that defines three FMP Supply wells. *B*, FeedFile that defines three FMP supply wells and indicates that one well is linked to MNW2. [MNW2, multi-node well package version 2; WBS, water-balance subregion; NPER, is the number of stress periods]

LineFeed Capacity Input Structure

The **LINEFEED CAPACITY** input structure is beneficial if supply well maximum capacity changes frequently. This input relies on LineFeed, described in appendix 2, to load the maximum capacity, QCAP, of the supply wells every stress period. The input is similar to that described in the "LineFeed WBS Input Structure" section, except that a FeedFile may only be associated with one WBS, and its "Temporal Input Section" specifies the supply wells' QCAP. Each line after the keyword **LINEFEED CAPACITY** specifies a WBS followed by its associated FeedFile, which is loaded with *Generic_Input*. This input is analogous to *List Style* input that reads WBS number as the record ID and the FeedFile location as the input. This input automatically continues to read the input as long as it correctly identifies a record ID. However, there is a limitation of one FeedFile per WBS—that is, a WBS number may only be used as a record ID once. Any WBS not defined as a record ID is assumed to have no supply wells associated. If it is desired to explicitly specify a WBS that does not have any supply wells, then the keyword **NOWELL** is written in the place of the *Generic_Input*.

Figure 6.50 is an example that reads two LineFeed files, with the first one specifying the supply-well capacities for WBS 2, the second is for WBS 3. In this example, any WBS that is not associated with a FeedFile is assumed to not have any supply wells. Additionally, any WBS that has the keyword **NOWELL** in the second column does not have any supply wells.

Figure 6.51 presents general input structure for the Spatial Input Section. The Spatial Input Section specifies data for as many wells as necessary until the keyword **TEMPORAL INPUT** is encountered to indicate that the stress-period input is on the subsequent lines. The Temporal input specifies the QCAP for each supply well. If a supply well does not exist for a stress period, then its QCAP must be set to NaN; however, if instead its QCAP is set to 0.0, then its capacity is reduced to zero, but its existence is still simulated, allowing intraborehole flow for MNW2 linked wells.

```
BEGIN SUPPLY_WELL
  #
  LINEFEED CAPACITY
  #
  # WBS   FeedFile
    2     OPEN/CLOSE ./FWEL_FEEDFILE2.txt
    3     OPEN/CLOSE ./FWEL_FEEDFILE3.txt
    5     NOWELL            #  Explicitly state WBS 5 does not have supply wells
  #
END SUPPLY_WELL
```

Figure 6.50. Supply Well block LINEFEED CAPACITY example that specifies input for three water balance subregions (WBS). This example only specifies FeedFiles for WBS 2 and 3, so all the other WBS do not have supply wells.

```
Well_Name  LAY  ROW  COL  [xyz]
```

Figure 6.51. SUPPLY_WELL block LINEFEED WBS general input structure for the LineFeed FeedFile's Spatial Input Section (time invariant input information).

Optional Keywords

The **SUPPLY_WELL** block includes a set of optional keywords that modify the global behavior of the FMP supply wells. The optional keywords and their descriptions are presented in figure 6.52.

A

Keyword	Argument	Description
SFAC	{**FLOAT**, **ByWBS** ULOAD}	**FLOAT** is a scale factor that is multiplied by all the supply-wells' maximum capacity (QCAP). If followed by the keyword **ByWBS**, then **NWBS** scale factors are read with ULOAD and applied to the respective wells associated with each WBS.
MNW_AUTOMATIC_ON		Automatically enables, makes "active", a MNW2 well that is linked to an FMP supply well. The MNW2 well is made active when the corresponding FMP well is first available to provide supply. If the supply well is no longer available, then the MNW2 well remains "active" with a desired pumping rate of zero until the simulation ends. This serves as a short cut to double specifying the MNW2 well name in its stress period input to mimic the time periods of use by the FMP.
MNW_AUTOMATIC_OFF		Automatically disables, removes from simulation, a MNW2 well that is linked to an FMP supply well. The MNW2 well is removed when the corresponding FMP well is defined to no longer provide supply. This feature should be used when the FMP supply well is considered "destroyed" rather than no longer providing supply to a WBS.

Figure 6.52. SUPPLY_WELL block optional keywords that modify its behavior. Secondary keywords enclosed in curly braces, { }, indicate that only one of the keywords may be selected and used during a simulation. *A*, Recommended keywords. *B*, Advanced keywords. *C*, Advanced keywords. *D*, Advanced keywords. [DIS, discretization package; FMP, farm process; MNW2, multi-node well package version 2; ULOAD, universal loader; WBS, water-balance subregion]

B

Keyword	Argument	Description
QMAXRESET	[**ByWBS** ULOAD]	MNW2 linked supply wells use the FMP QCAP as an initial estimate of the well capacity. This capacity is then reduced by MNW2 for the given aquifer conditions and then reset to the original QCAP at the start of the next stress period. This keyword tells FMP that QCAP should be reset every time step. If followed by the keyword **ByWBS**, then a list of **NWBS** 0s or 1s are read with ULOAD to indicate which WBS should use (1) QMAXRESET or not (0). By default, this setting is True (1), so specifying it is unnecessary unless a user desires to disable the feature for specific WBS's.
NOQMAXRESET		Disables **QMAXRESET** for all MNW2 linked supply wells so their QCAP is only reset at the start of each stress period rather than each time step. This may improve speed of the simulation but at the potential expense of under-estimating pumpage.
NOCIRNOQ	[**ByWBS** ULOAD]	Specifies that supply wells will only provide supply if there is irrigation demand within the row and column where the well is located. That is, if there is zero irrigation demand for a specific row and column, then all wells that have the same row and column will not provide any supply to the WBS. This works best for wells that only supply water to the land surface they are underneath. If followed by the keyword **ByWBS**, then **NWBS** 0 or 1s are read with ULOAD to indicate which WBS should use (1) NOCIRNOQ or not (0).

Figure 6.52. —Continued

C

Keyword	Argument	Description
INPUT_OPTION XY		Indicates that all supply-well input will use the X- and Y-coordinate in the place of the row and column input. The coordinates must be a location within the Cartesian coordinate system defined by the DIS package. The row and column are determined based on what model cell contains the given X- and Y-coordinate.
INPUT_OPTION	{**ELEVATION**, **DEPTH**}	Indicates that all supply-well input will use either the **ELEVATION** or **DEPTH** from land surface in the place of the Layer input. If **ELEVATION** is specified, then the layer number is determined based on the DIS TOP and BOTM input for the supply-well's location. If **DEPTH** is specified, then it is converted to an elevation by subtracting the depth from the **GLOBAL DIMENSION** block's **SURFACE_ELEVATION** for the supply-well's location.
MNW_PUMP_SPREAD	{**BY_COND**, **BY_COUNT**, **BY_TOP**}	This option may improve convergence speed and will not affect the final pumping rate. Wells linked to MNW2 that contain more than one node must assign the desired pumping to all of the well's nodes. MNW2 will then allocate the desired pumpage among the nodes to meet the total desired pumping rate. This option offers different methods for initially applying the desired pumping rate to MNW2 when there is more than one node. **BY_COND** is the default and applies the desired pumpage based on each node's conductance relative to the total. **BY_COUNT** divides the desired pumpage by the total node count and applies that rate to all nodes. **BY_TOP** applies the desired pumpage to the uppermost, top node. This was how FMP3 originally assigned MNW2 linked pumpage.

Figure 6.52. —Continued

D

Keyword	Argument	Description
PRORATE	{ **ByAverage**, **ByCapacity** }	When the demanded water for a WBS is less than its total pumpage capacity (QMAX) the assigned pumpage is prorated across all the supply wells. **ByCapacity** is the default, indicates that the total demand for pumpage is spread across supply wells based on the ratio of each well's capacity, QCAP, to the total capacity of the WBS, QMAX. This has the effect of having higher capacity wells supply more pumpage than lower capacity wells. **ByAverage** calculates an average demand per well and then wells with capacity below this average are set to their QCAP and wells with capacity above the average have an even split of the remaining demand. This has the effect of having lower capacity wells use their full capacity before the larger capacity wells. This was the method used by FMP3.
SMOOTH	{ **ByFRACTION**, **ByTHICK** }	**FLOAT**, Smoothing allows reduction of pumping from non-MWN2 linked wells when the water table approaches the bottom of the model cell. Modifier **ByTHICK** specifies a minimum thickness in model units before smoothing occurs, while **ByFRACTION** specifies a minimum fraction of cell thickness before smoothing occurs.

Figure 6.52. —Continued

Output Keywords

The **SUPPLY_WELL** block has a set of optional, additional output keywords to improve the understanding and analysis of the demand based pumpage. To initiate the output options, write in the **SUPPLY_WELL** block the keyword **PRINT,** followed by the output type, and then the output file, specified with *Generic_Output_OptKey*. The output types available are **LIST**, **INPUT**, **SMOOTHING**, **ByWBS**, **ByWell**, and **ByMNW**. The **LIST** and **INPUT** options print a transcript of the input loaded to either the Listing File or an external file, respectively. If the **SUPPLY_WELL** block input includes the keyword **SMOOTH** and the well undergoes capacity smoothing, the **SMOOTHING** file records the smoothing values applied to the well. The **ByWBS**, **ByWell**, and **ByMNW** output provide detailed information on the requested pumpage and any reductions in pumpage that may result from allotments, smoothing, and MNW. The three output types differ in how the well information is aggregated. **ByWBS**, **ByWell**, and **ByMNW** aggregate the information at the WBS level, for each well, and only for wells linked to MNW2, respectively. The recommended output type is the **ByWBS**, which indicates the groundwater production capacity of each WBS and what resulted in reduction of the capacity. Figure 6.53 presents the output keywords and gives a basic description of each.

Keyword	Argument	Description
PRINT LIST	—	Write to the LIST file a transcript of the supply wells that are in use.
PRINT INPUT	*Generic_Output_OptKey*	Write a transcript of the well input to a separate file.
PRINT SMOOTHING	*Generic_Output_OptKey*	If **SMOOTH** is enabled, then writes any non-MNW2 linked wells that had their capacity reduced because of smoothing. If not enabled, then file will be empty.
PRINT ByWBS	*Generic_Output_OptKey*	Writes to a file at every time step a summary of well information for each WBS.
PRINT ByWELL	*Generic_Output_OptKey*	Writes to a file at every time step a summary of well information for each well in use.
PRINT ByMNW	*Generic_Output_OptKey*	Writes to a file at every time step a summary of well information for each supply well that is linked to MNW2.

Figure 6.53. SUPPLY_WELL block output file keywords. [MNW2, multi-node well package version 2; WBS, water-balance subregion; —, no input values apply]

PRINT ByWBS Header

The keyword **PRINT ByWBS** writes the aggregated pumpage information for each WBS to the *Generic_Output_OptKey* file every time step. The output may be text or binary format and has the header defined in figure 6.54.

A

PER	is the stress period number	
STP	is the time step number	
WBS	is the Water Balance Subregion (Farm) ID number	
Q-CAP-INI	is the initial farm well's pumping capacity as specified by the input	[L^3/T]
Q-CAP-ALLOT	is the farm well's capacity after any allotment constraints are applied	[L^3/T]
Q-CAP-FIN	is the farm well's final pumping capacity	[L^3/T]
Q-FIN	is the final volumetric discharge rate of well during current time step	[L^3/T]
TFDR-INI	is the initial demand for water supply	[L^3/T]
TFDR-FIN	is the demand that was satisfied with water supply	[L^3/T]
DELT	is the time step length	[T]
DYEAR	is the date at the end of the time step as a decimal year	
DATE_START	is the calendar date at the start of the time step (yyyy-mm-ddThh:mm:ss)	

B

DATE_START	CHARACTER(19), starting date formatted as 'yyyy-mm-ddThh:mm:ss'
DYEAR	DOUBLE
DELT	DOUBLE
PER	INTEGER
STP	INTEGER
WBS	INTEGER
Q-CAP-INI	DOUBLE
Q-CAP-FIN	DOUBLE
Q-FIN	DOUBLE
TFDR-INI	DOUBLE
TFDR-FIN	DOUBLE

Figure 6.54. Supply Well block keyword PRINT ByWBS text file header explanation and binary record structure. *A*, Text-header explanation. *B*, Binary-record structure. [[T], unit of time in model units; [L^3/T], volumetric rate in model units; CHARACTER(19) indicates record is 19 characters long (19 bytes); INTEGER is a 4-byte integer record; DOUBLE is a 8-byte floating-point number record.]

PRINT ByWELL Header

The keyword **PRINT ByWELL** writes the pumpage information for each supply well in use for that time step to the *Generic_Output_OptKey* file every time step. The output may be text or binary format and has the header defined in figure 6.55.

A

PER	is the stress period number	
STP	is the time step number	
WBS	is the Water Balance Subregion (Farm) ID number	
WELLID	is the name (WELL_ID) of the specific supply well	
Q-CAP-INI	is the initial well's pumping capacity as specified by the input	[L^3/T]
Q-CAP-FIN	is the well's final pumping capacity	[L^3/T]
Q-ACT	is the final volumetric discharge rate of the well during current time step	[L^3/T]
Q-DES-DMD	is an estimate of the well's initial demand for pumping	[L^3/T]
LAY	is the well's model layer; it is set to zero if linked to MNW2	
ROW	is the well's model row	
COL	is the well's model column	
DELT	is the time step length	[T]
DYEAR	is the date at the end of the time step as a decimal year	
DATE_START	is the calendar date at the start of the time step (yyyy-mm-ddThh:mm:ss)	

B

WELLID	CHARACTER(20)
LAY	INTEGER
ROW	INTEGER
COL	INTEGER
DATE_START	CHARACTER(19), starting date formatted as 'yyyy-mm-ddThh:mm:ss'
DYEAR	DOUBLE
DELT	DOUBLE
PER	INTEGER
STP	INTEGER
WBS	INTEGER
Q-CAP-INI	DOUBLE
Q-CAP-FIN	DOUBLE
Q-FIN	DOUBLE
TFDR-INI	DOUBLE
TFDR-FIN	DOUBLE

Figure 6.55. Supply Well block keyword PRINT ByWELL text file header explanation and binary record structure. *A*, Text-header explanation. *B*, Binary-record structure. [[T], unit of time in model units; [L^3/T], volumetric rate in model units; CHARACTER(20) indicates record is 20 characters long (20 bytes); CHARACTER(19) indicates record is 19 characters long (19 bytes); INTEGER is a 4-byte integer record; DOUBLE is a 8-byte floating-point number record.]

PRINT ByMNW Header

The keyword **PRINT ByMNW** writes the pumpage information for each FMP-MNW2 linked supply well in use for that time step to the *Generic_Output_OptKey* file every time step. The output may be text or binary format and has the header defined in figure 6.56.

A

PER	is the stress period number	
STP	is the time step number	
WBS	is the Water Balance Subregion (Farm) ID number	
WELLID	is the name (WELL_ID) of the specific supply well	
Q-CAP-INI	is the initial well's pumping capacity as specified by the input	[L^3/T]
Q-CAP-FIN	is the well's final pumping capacity	[L^3/T]
Q-ACT	is the final volumetric discharge rate of the well during current time step	[L^3/T]
Q-DES-DMD	is an estimate of the well's initial demand for pumping	[L^3/T]
H-WEL	is the MNW2 calculated well head (h_{well})	[L]
H-CEL-AVE	is MNW2 well's aquifer head (conductance weighted by node)	[L]
DELT	is the time step length	[T]
DYEAR	is the date at the end of the time step as a decimal year	
DATE_START	is the calendar date at the start of the time step (yyyy-mm-ddThh:mm:ss)	

B

WELLID	CHARACTER(20)
DATE_START	CHARACTER(19), starting date formatted as 'yyyy-mm-ddThh:mm:ss'
DYEAR	DOUBLE
DELT	DOUBLE
PER	INTEGER
STP	INTEGER
WBS	INTEGER
Q-CAP-INI	DOUBLE
Q-CAP-FIN	DOUBLE
Q-ACT	DOUBLE
Q-DES-DMD	DOUBLE
H-WEL	DOUBLE
H-CEL-AVE	DOUBLE

Figure 6.56. Supply Well block keyword PRINT ByMNW text file header explanation and binary record structure. *A*, Text-header explanation. *B*, Binary-record structure. [[T], unit of time in model units; [L^3/T], volumetric rate in model units; CHARACTER(20) indicates record is 20 characters long (20 bytes); CHARACTER(19) indicates record is 19 characters long (19 bytes); INTEGER is a 4-byte integer record; DOUBLE is a 8-byte floating-point number record.]

Surface-Water Block

The **SURFACE_WATER** block is an optional block that specifies surface-water related deliveries to a WBS and the destination of aggregated surface runoff. In this release of MF-OWHM2, there are two surface-water delivery options: semi-routed deliveries (SRD) and non-routed deliveries (NRD). The surface runoff may either be aggregated by a WBS and then applied to the streamflow routing package (SFR) as return flow or re-infiltrated as deep percolation. There are two return flow choices, which are semi-routed return flow (SRR) and fully routed return flow (FRR). This sub-section gives an overview of the types of deliveries and return flow. This input block differs from other blocks in that the *List Style* input varies depending on the keyword. In FMP, the block name can be declared as **SURFACE WATER** or **SURFACE_WATER**, but the later version, with an underscore, is recommended for clarity. Figure 6.57 presents the supported keywords for the **SURFACE_WATER** block.

Non-Routed Delivery (NRD)

A non-routed delivery (NRD) is a user-specified amount of water that is available to the WBS as supply. To have an NRD the **GLOBAL DIMENSION** block must specify the keyword **NRD_TYPES**, which indicates the number of NRDs per WBS that will be defined. The NRD itself is defined with the keyword **NON_ROUTED_DELIVERY**. An NRD may be specified as a **VOLUME** (L^3) or **RATE** (L^3/T) per stress period. If the WBS demand is less than the NRD, then the unused portion of the NRD may be directed to one or more of the following: left in the SFR reach at the point of diversion (not be delivered to the WBS), released as runoff to the SFR network, injected into groundwater from the WBS supply wells, or added as deep percolation—groundwater recharge—to specific model locations. If the unused portion is added to deep percolation, then the keyword **NRD_INFILTRATION_LOCATION** LAI[S, T, A] must be included to define where the surplus water is added to deep percolation. It is important to note that an NRD that is specified using keyword **VOLUME** is specified as a volume per stress period and not as a rate nor volume per time step.

The NRD expects three input values per **NRD_TYPES**, called the NRD triplet, which are read for each WBS. The NRD triplet is composed of the quantity of water delivered, its order of consumption amongst the other **NRD_TYPES**, and how the excess water is treated (fig. 6.58). The formal NRD input definition is **NON_ROUTED_DELIVERY** [**VOLUME**, **RATE**] LAI[S, T, L-(3×**NRD_TYPES**)], where the keywords **VOLUME** and **RATE** are optional. The keyword **VOLUME** indicates that the NRD is specified as a volume (L^3) per stress period and the keyword **RATE** indicates that the NRD is specified as a rate (L^3/T) per stress period. If neither **VOLUME** nor **RATE** are specified, then the NRD defaults to volume per stress period—this is the method that previous versions of the FMP processed NRDs.

Figure 6.57. SURFACE_WATER block keyword list. Secondary keywords enclosed in brackets, [ ], indicate that they are optional and at most one may be selected and used during a simulation. Input enclosed in curly braces, { }, indicate that it is optional. *A*, Non-Routed Delivery keywords. *B*, Return flow keywords. *C*, SFR surface water delivery keywords. [Some of the keywords are specified over two lines because of their length, but should be a single, contiguous word in the input. The proper set up is to have on one line the keyword, secondary keyword, and input. SEG and RCH may be specified with using a name defined with the keyword SFR_NAMES in the GLOBAL DIMENSION block. Abbreviations: FLOAT, floating-point number; INT, integer number, ID, water balance subregion identification number; WBS, water-balance subregion; [L], length in model units; [L^3], volume in model units; [L^3/T], volumetric rate in model units; SFR, streamflow routing package; LAI, list-array input with the following letters representing the supported keywords: S for STATIC, T for TRANSIENT, A for ARRAY, and L for LIST]

A

Keyword Secondary Keyword Input	Description
NON_ROUTED_DELIVERY [**VOLUME**, **RATE**] LAI[S, T, L-(3×**NRD_TYPES**)]	Non-Routed Delivery (NRD) is water supply that is delivered to a WBS for a stress period. Use of the secondary "input" keyword (**RATE** or **VOLUME**) is optional, but only one may be specified per simulation. if no "input" keyword is present or specified as **VOLUME**, then the input for NRD volume (NRD_VOL) is a volume [L^3]. The **VOLUME** [L^3] is divided by the stress period length to obtain a rate of water that is available during the stress period. If **RATE** is specified, then NRD_VOL is specified as a volumetric rate [L^3/T]. *List Style* reads **NWBS** records with each record containing an **NRD_TYPES** set of three values, called a "NRD Triplet". The NRD Triplet is: (NRD_VOL, NRD_RANK, NRD_USE) (**FLOAT**, **INT**, **INT**) where NRD_VOL is the quantity of water delivered to the WBS NRD_RANK is the order that water is consumed by the WBS demand. NRD_USE is what to do when delivered water exceeds demand. Accepted values are: 0 indicates only water necessary to meet demand is delivered. 1 indicates excess water is discharged to the WBS semi-routed delivery location. 2 indicates excess water is injected to the WBS supply wells. ≥10 indicates excess water is infiltrated as deep percolation wherever the integer, NRD_USE, is in the array defined in **NRD_INFILTRATION_LOCATION**.
NRD_INFILTRATION_LOCATION LAI[S, T, A]	Identifies location to infiltrate excess water from a **NON_ROUTED_DELIVERY** (NRD). Input is an **INT** array that specifies infiltration locations as either 0 (not used location) or ≥10. An NRD with NRD_USE ≥10 has its unused delivered (excess) water infiltrated at the corresponding ≥10 location in array.

Figure 6.57. —Continued

B

Keyword and Input	Description
NO_RETURN_FLOW { LAI[S, T, L] }	Keyword indicates all surface runoff re-infiltrates as deep percolation. Optionally, the flag can be specified by the WBS with LAI[S, T, L]input reads either a value of 1 —to indicate no return flow— or 0 —to allow return flow.
SEMI_ROUTED_RETURN LAI[S, T, L-3] or LAI[S, T, L-4]	Reads **NSFR_RETURN** records that specify the location in SFR that a WBS's return flow (aggregated surface runoff) flows into. Input expects for each record: ISRR, WBS, SEG, RCH, [FRAC] that is read as **INT**, **INT**, **INT**, **INT**, [**FLOAT**] where ISRR is the record ID, WBS is either 0 to indicate the return flow point is skipped or the WBS that the return flow is coming from, SEG and RCH are the SFR segment and reach that return flow is sent to, and FRAC is the fraction of the WBS return flow that is applied to SEG and RCH. If FRAC is not specified, the return flow is prorated across all specified SEG and RCH based on their length compared to the total length. If the same WBS is defined multiple times, then the return flow is either length prorated or applied based on the specified FRAC. If a WBS has SEG set to 0, then the WBS return flow is removed from the model domain. If RCH is set to 0, then all reaches in SEG receive the return flow. SEG and RCH maybe be specified with a name defined from **SFR_NAMES** in the **GLOBAL DIMENSION** block. If WBS set to 0, then SEG, RCH, [FRAC] are not read.
ROUTED_RETURN_ANY_NON_DIVERSION_REACH	Fully routed return flow is enabled for all WBSs not defined with a **SEMI_ROUTED_RETURN** location. A WBS surface runoff is prorated by the length of all "non-diversion" SFR reaches within the WBS. This keyword is recommended whenever a simulation includes SFR.
ROUTED_RETURN_ANY_REACH	Fully routed return flow is enabled for all WBSs not defined with a **SEMI_ROUTED_RETURN** location. A WBS surface runoff is prorated by the length of all SFR reaches within the WBS.

Figure 6.57. —Continued

C

Keyword and Input	Description
SEMI_ROUTED_DELIVERY LAI[S, T, L-3] or LAI[S, T, L-4]	Reads **NSFR_DELIV** records that each define where a specific WBS has access to SFR surface water deliveries—where a WBS can remove water from SFR to satisfy demand. Input expects for each record: ISRD, WBS, SEG, RCH, [FRAC] that is read as **INT**, **INT**, **INT**, **INT**, [**FLOAT**] where ISRD is the record ID, WBS is either 0 to indicate the delivery is not used or the WBS that can use the delivery point, SEG and RCH are the SFR segment and reach that water is removed from, and FRAC is optional and is the fraction of the WBS demand that the delivery point attempts to satisfy. If FRAC is not specified, then delivery points attempt to satisfy the demand in the order they are specified. If WBS set to 0, then SEG, RCH, [FRAC] are not read.
SEMI_ROUTED_DELIVERY_LOWER_LIMIT LAI[S, T, L]	Optional, reads **NSFR_DELIV** records that define the lower limit for each **SEMI_ROUTED_DELIVERY** in [L^3/T]. Only flow greater than the lower limit is available for delivery. Analogous to an SFR flood-control type diversion (IPRIOR = –3). If this keyword is not specified, then the default value is a lower limit of 0.0—that is any flow present is available for delivery.
SEMI_ROUTED_DELIVERY_UPPER_LIMIT LAI[S, T, L]	Optional, reads **NSFR_DELIV** records that define the upper limit for each **SEMI_ROUTED_DELIVERY** in [L^3/T]. The delivery only has access to water flow up to this upper limit. Analogous to an SFR specified-diversion flow (IPRIOR = 0). If this keyword is not specified, then the default value is an upper limit of 1E100.
SEMI_ROUTED_DELIVERY_CLOSURE_TOLERANCE **FLOAT**	Optional, and specifies a tolerance for delivery rate changes between solver iterations before convergence is allowed. If not specified it has a default value of 0.02 [L^3/T].

Figure 6.57. —Continued

NRD Triplet Definition

(NRD_VOL, NRD_RANK, NRD_USE) × **NRD_TYPES**

where:

NRD_TYPES is the number of Non-Routed Delivery (NRD) defined in the Global Dimension Block.

NRD_VOL is the volume [L^3] or rate [L^3/T] of water delivered to the WBS for the stress period. If the **OPTIONS** Block has the keyword **WELLFIELD**, then this value is ignored.

NRD_RANK is order of consumption of the NRDs, it must be between 1 and **NRD_TYPES**. This only has an effect when a WBS demand is less than the volume of water delivered from all the NRDs.

NRD_USE defines how the unused portion of the NRD is handled. The following are the accepted options:

- <0: is only allowed with **WELLFIELD** option. Its absolute value is the WBS that represents the well field.
- 0: indicates that only the amount necessary to satisfy demand is delivered. No excess water occurs or is brought into the WBS.
- 1: the unused portion of water delivered is discharged back into the WBS's semi-routed delivery locations. If there are more than one location, the excess is divided evening across them.
- 2: the unused portion of water delivered is injected into groundwater using the WBS Supply Wells.
- ≥10: is an integer located within the **NRD_INFILTRATION_LOCATION** array. Any surplus water is added to deep percolation at that integer location. If the integer is located at multiple spots in the array, then the surplus is divided evenly among the locations. If the integer is not located in the array, then an error is raised.

Figure 6.58. Non-Routed Delivery (NRD) triplet input structure and explanation. [[L^3], volume in model units; [L^3/T], volumetric rate in model units]

Figure 6.59 illustrates the structure of the NRD input with an example **GLOBAL DIMENSION** and **SURFACE_WATER** blocks. Figure 6.59*A* has five WBSs and four **NRD_TYPES** defined where the NRD triplets, (4×**NRD_TYPES**), represent water available from a recycled water facility, an imported water pipeline, and two water sources that can be purchased from an external market—Purchase1 and Purchase2. The recycled water facility and imported water pipeline always deliver their water, irrelevant of their consumption, so they have NRD_USE set to 1—to allow the excess to runoff. The recycled water facility provides water to WBS 1, 2, 3, and 4, whereas the imported water pipeline only provides water to WBS 1, 2, and 3. Purchase1 and Purchase2 are available to WBS 1 and 2, but WBS 1 prefers Purchase1 over Purchase2—say they have a contract to pay less for Purchase1, whereas WBS 2 has the opposite preference—preferring Purchase2 over Purchase1. Because the WBS will pay for supplies from Purchase1 and Purchase2 the NRD_USE is set to `0` to only use what is needed. WBS 5 does not have access to any NRD sources. Note that the use of the **STATIC** keyword indicates that these volumes are delivered for every stress period. If **TRANSIENT** is specified, then the TFR file would specify the volumes delivered for each month.

Figure 6.59*B* is an example demonstrating how to use **NRD_INFILTRATION_LOCATION** to have the unused NRD passed to infiltration ponds. This example assumes a model grid of `NROW=3` and `NCOL=4`. For simplicity, **NRD_TYPES** is reduced to 2, with the first source being the recycled water the second source the pipeline water, but the unused portion is passed to deep percolation.

A

```
BEGIN GLOBAL DIMENSION
  NWBS         5  #  Number of water balance subregions (FARMS)
  NRD_TYPES    4  #  Number of non-routed delivery types defined (NRD)
  NCROP        5  #  Assume the following are defined elsewhere
  NSOIL        1
  NIRRIGATE    1
END GLOBAL DIMENSION

BEGIN SURFACE_WATER
  #
  NON_ROUTED_DELIVERY VOLUME STATIC LIST INTERNAL
  #
  #  WBS  (NRD_VOL, NRD_RANK, NRD_USE) × NRD_TYPES
  #
  #   Recycled Water   Pipeline       Purchase1       Purchase2
  1  100.  1  1   50.  2  1  500.  3  0  100.  4  0 # Preference for Purchase1
  2  100.  1  1   50.  2  1  200.  4  0  400.  3  0 # Preference for Purchase2
  3  100.  1  1   50.  2  1    0.  3  0    0.  4  0 # Recycled + Pipeline
  4  100.  1  0    0.  2  0    0.  3  0    0.  4  0 # Only has recycled water
  5    0.  1  0    0.  2  0    0.  3  0    0.  4  0 # No NRD
  #
END SURFACE_WATER
```

Figure 6.59. Non-Routed Delivery input example consisting of the FMP Global Dimension and Surface_Water blocks. *A*, Example defintng 4 NRD_TYPES. *B*, Example defining 2 NRD_TYPES and defines infiltration locations with NRD_INFILTRATION_LOCATION.

B

```
BEGIN GLOBAL DIMENSION
  NWBS         5   # Number of water balance subregions (WBS)
  NRD_TYPES    2   # Number of non-routed delivery types defined (NRD)
  NCROP        5   # Assume the following are defined elsewhere
  NSOIL        1
  NIRRIGATE    1
END GLOBAL DIMENSION

BEGIN SURFACE_WATER
  #
  NRD_INFILTRATION_LOCATION STATIC ARRAY INTERNAL
   0   0  10  10
   0   0  10  10
  11  11   0   0
  #
  NON_ROUTED_DELIVERY VOLUME STATIC LIST INTERNAL
  #
  #   Recycled Water   Pipeline
  1  100.  1  1   50.  2  1    # Pipeline water discharged to SFR
  2  100.  1  1   50.  2  10   # Unused Pipeline is infiltrated at location 10
  3  100.  1  1   50.  2  11   # Unused Pipeline is infiltrated at location 11
  4  100.  1  0    0.  2  0    # Only has recycled water
  5    0.  1  0    0.  2  0    # No NRD
  #
END SURFACE_WATER
```

Figure 6.59. —Continued

Semi-Routed Delivery (SRD)

A semi-routed delivery (SRD) is a user specified SFR segment and reach that a WBS can remove water from to meet its demand. An SRD differs from the NRD in that the conveyance, through the SFR package, to the WBS is simulated—that is, the SRD quantity of water is limited and determined by the amount of flow through the SFR network. The keyword **SEMI_ROUTED_DELIVERY** in the **SURFACE_WATER** block enables SRDs and uses *List Style* input. The **SEMI_ROUTED_DELIVERY** *List Style* input that expects **NSFR_DELIV** records (fig. 6.60).

Figure 6.61 illustrates how to set up an input with **SEMI_ROUTED_DELIVERY** by defines six SRD locations. The simulation has a total of five WBSs, but only WBS 1, 2 and 4 have access to a surface-water delivery to meet their demand. WBS 4 has three delivery locations, and because FRAC is not specified the first delivery location (ISRD=3), attempts to fully meet the WBS demand. If it does not meet the demand, then the second delivery location (ISRD=4) attempts to meet the demand. If the first two SRDs have not met the demand, then the final potential surface-water source (ISRD=5) attempts to meet the WBS demand.

The amount of water that the WBS can remove from the SRD is either equal to its demand or the available flow in the SFR diversion segment and reach, whichever is smaller. There are three methods to impose control over the quantity of water that a WBS can remove from stream flow. The first is to use a “dummy” diversion segment that serves as the SRD. This dummy segment is a small segment that is a diversion from the main SFR flow and immediately connects back to the same segment it diverts water from. Figure 6.62 presents a simple SFR network that has the dummy segment, Segment 3, deliver water to a WBS. This method relies on the SFR diversion options to control the dummy segment’s flow, and consequently, the water delivered to the WBS. Any water that flows into the dummy segment that is not consumed by the WBS returns to the main SFR flow at Segment 4.

A

```
ISRD  WBS  SEG  RCH  [FRAC]
```

B

Input Format Keyword	Description
ISRD	*List Style* input record ID, **INT**, for the Semi-Routed Delivery (SRD) location. Its value is read, but not used. It must be sequential from 1 to **NSFR_DELIV**.
WBS	The WBS ID, **INT**, that can remove water from the specified SEG and RCH. A WBS may be specified in multiple ISRD records allowing for the WBS to have multiple SRD locations. If WBS is set to 0, then the ISRD is not available to deliver water. This allows for fewer than **NSFR_DELIV** to be in use during a stress period. If the WBS is set to 0, then the rest of the input line is ignored.
SEG	The semi-routed delivery's SFR segment, **INT**, that the WBS can remove water from at the SRD location to meet its demand.
RCH	The semi-routed delivery's SFR reach, **INT**, that the WBS can remove water from at the SRD location to meet its demand.
FRAC	An optional input that specifies the fraction, **FLOAT**, of the WBS demand that the semi-routed delivery attempts to satisfy. If a WBS has only one SRD location, then FRAC is set to 1.0. If a WBS has more than one SRD location and FRAC is either not specified or is a negative number, then the delivery to meet demand follows the order in which the SRD locations are specified. That is, the WBS's smallest ISRD first attempts to satisfy the demand with its supply. If there is a deficit, then the WBS's next lowest ISRD attempts to satisfy the demand. This continues until all the WBS's SRDs deliver all their water or the demand is met. An example use of FRAC is a WBS has two SRD locations that have FRAC set to 0.7 and 0.3 and a demand of 60 [L^3/T]. The water that each point delivers 42 and 18 [L^3/T], respectively. If the supply at a SRD location is less than what is calculated by FRAC, the deficit is shifted to the SRD locations to ensure demand is fully met.

Figure 6.60. SURFACE_WATER block SEMI_ROUTED_DELIVERY input definition. A total of NSFR_DELIV records are read with each record containing ISRD, WBS, SEG, RCH, and optionally FRAC. *A*, Expected input structure for each List Style input record. *B*, Explanation of input variables. [SEG and RCH may be replaced with a name defined in the GLOBAL DIMENSION block's keyword SFR_NAMES. Input enclosed in brackets, [ ], indicates it is optional. Abbreviations: FLOAT, floating-point number; INT, integer number; ID, water balance subregion identification number; WBS, water-balance subregion; SFR, streamflow routing package; [L^3/T], volumetric rate in model units]

```
BEGIN GLOBAL DIMENSION
  NWBS          5  #  Number of water balance subregions (WBS)
  NRD_TYPES     0  #  Number of non-routed delivery types defined (NRD)
  NCROP         5  #  Assume the following are defined elsewhere
  NSOIL         1
  NIRRIGATE     1
  NSFR_DELIV    6  #  Number of SEMI_ROUTED_DELIVERY locations defined
  #
  SFR_NAMES INTERNAL
# ID  SFRNAME    ISEG  IRCH
  1 WBS1_SRD     22     1
  2 WBS2_SRD      2     3
  #
END GLOBAL DIMENSION

BEGIN SURFACE_WATER
  #  NSFR_DELIV = 6, so SEMI_ROUTED_DELIVERY reads 6 records (ISRD = 1 to 6)
  #
  SEMI_ROUTED_DELIVERY STATIC LIST INTERNAL
  #
  # ISRD  WBS   SEG    RCH  [FRAC]
    1      1    WBS1_SRD        # Using SFR_NAMES to point to SEG 22, RCH 1
    2      2    WBS2_SRD        # Using SFR_NAMES to point to SEG 2,   RCH 3
    3      4    5     1         # WBS 4 has 3 delivery points. 1st choice to meet demand
    4      4    8     1         # WBS 4 has 3 delivery points. 2nd choice to meet demand
    5      4    12    2         # WBS 4 has 3 delivery points. 3rd choice to meet demand
    6      0    0     0         # ISRD record is ignored because WBS is 0
  #
END SURFACE_WATER
```

Figure 6.61. Semi-Routed Delivery input example consisting of the FMP Global Dimension and Surface_Water blocks.

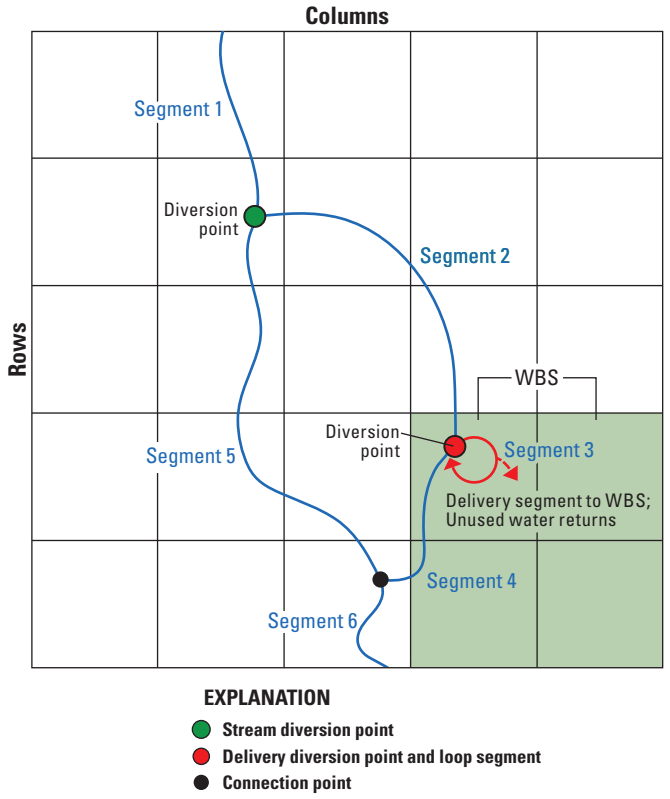


Figure 6.62. Example SFR stream network that delivers water to an FMP semi-routed delivery location (SRD) at segment 3 to meet a WBS water demand. [Water is conveyed from segment 1 to segment 2 to segment 3, then removed by the WBS, and the unused portion returns to segment 4. Abbreviations: SFR, streamflow routing package; WBS, water-balance subregion.]

The second method is to control the water consumed by a WBS is using surface-water allotments described in the "Allotment Block" section of this appendix. A surface-water allotment imposes a total constraint of surface-water supply for a WBS but does not control the available supply at an individual SRD location. The third option is to use the keywords **SEMI_ROUTED_DELIVERY_LOWER_LIMIT** and **SEMI_ROUTED_DELIVERY_UPPER_LIMIT** (fig. 6.57*A*) to limit the available supply from each SRD location.

The **SEMI_ROUTED_DELIVERY_LOWER_LIMIT** and **SEMI_ROUTED_DELIVERY_UPPER_LIMIT** are optional keywords that define the range of available flow for each SRD location. If they are not included in the input, then their default values are `0` and `1E100` [L^3/T], respectively, which imposes no limit on the delivery location. The expected input is `LAI[S, T, L]` that reads **NSFR_DELIV** records. Figure 6.63 is an example input that defines the **SEMI_ROUTED_DELIVERY_LOWER_LIMIT** and **SEMI_ROUTED_DELIVERY_UPPER_LIMIT** as if they were placed in the **SURFACE_WATER** block in figure 6.61.

SEMI_ROUTED_DELIVERY_LOWER_LIMIT is the flow rate that the delivery location must exceed before water is available to the WBS for delivery. This type of limit is analogous to an SFR flood-control type diversion (`IPRIOR` = –3). For example, if the lower limit is set to 10 [L^3/T], then only flows greater than 10 [L^3/T] are available for delivery. If the flow at the delivery segment is 15 [L^3/T], then only 5 [L^3/T] is available for delivery (from 5=15–10). If the flow is 8 [L^3/T], then no water is available for delivery (because 8≤10).

SEMI_ROUTED_DELIVERY_UPPER_LIMIT is an upper flow limit that a delivery location must exceed before water is available to the WBS for delivery. This type of limit is analogous to an SFR specified-diversion flow diversion (`IPRIOR` = `0`). For example, if the upper limit was set to 60 [L^3/T], then a WBS may only remove from that delivery location at most 60 L^3/T minus any imposed lower flow limit. For example, a delivery location that has a lower limit of 10 [L^3/T] and upper limit of 60 [L^3/T] would have a maximum potential delivery of 50 [L^3/T] (from 60 – 10). If the upper limit is set to a negative number, then it is automatically changed to `1E100` to disable it—this assumes that SFR flow is never greater than `1E100` [L^3/T].

```
#  Example assumes that the GLOBAL DIMENSION block defines NSFR_DELIV = 6
⋮
⋮  Example excerpt from SURFACE_WATER block
⋮
#
#  Note that NSFR_DELIV = 6,
#  so SEMI_ROUTED_DELIVERY_LOWER_LIMIT and SEMI_ROUTED_DELIVERY_UPPER_LIMIT
#  each read 6 records. That is, ISRD = 1, 2, 3, 4, 5, and 6.
#
SEMI_ROUTED_DELIVERY_LOWER_LIMIT STATIC LIST INTERNAL
#
# ISRD  Lower_Limit
  1     10.0          # Only flow greater than 10 [L3/T] is available for delivery
  2      0.0          # No lower limit on delivery point
  3     20.0          # Only flow greater than 20 [L3/T] is available for delivery
  4      0.0          # No lower limit on delivery point
  5      0.0          # No lower limit on delivery point
  6      0.0          # No lower limit on delivery point
#
SEMI_ROUTED_DELIVERY_UPPER_LIMIT STATIC LIST INTERNAL
#
# ISRD  Upper_Limit
  1     60.0       # Only flow less than 60 [L3/T] is available, max potential delivery is 50 [L3/T]
  2     75.0       # Only flow less than 75 [L3/T] is available, max potential delivery is 75 [L3/T]
  3     50.0       # Only flow less than 50 [L3/T] is available, max potential delivery is 30 [L3/T]
  4     1E100      # Upper limit so large it cannot be exceeded, so no upper limit on delivery point
  5     -1.0       # Negative number, so no upper limit on delivery point
  6      0.0       # Upper limit set to 0 [L3/T], no flow is available; no delivery is possible
#
⋮
⋮ Continue with the rest of the SURFACE_WATER block
```

Figure 6.63. Example input of the SEMI_ROUTED_DELIVERY_LOWER_LIMIT and SEMI_ROUTED_DELIVERY_UPPER_LIMIT within a Surface Water Block. These impose lower and upper limits on semi-routed delivery locations. [[L^3/T], volumetric rate in model units; SFR, streamflow routing package; SRD, semi-routed delivery; NSFR_DELIV, number of SRD locations; ISRD, number that identifies the SRD, which is between 1 and NSFR_DELIV]

Return Flow

Surface runoff in the FMP is flow that results from excess irrigation or precipitation that is not consumed by the landscape nor infiltrates to groundwater. Surface runoff is calculated for all surface model cells that are assigned a non-zero WBS ID—that is, it is calculated as a surface cell-by-cell. The resulting runoff can either entirely re-infiltrate to groundwater as deep percolation or be aggregated by the WBS. The aggregated surface runoff is called return flow and is applied to the SFR package as runoff. By specifying the keyword **NO_RETURN_FLOW** all surface runoff is designated for re-infiltration to groundwater. If it is desired to specify this disposition of surface runoff individually for each WBS, then the keyword **NO_RETURN_FLOW** should be followed with `LAI[S, T, L]`, which loads **NWBS** *List Style* records that are either a `1` to indicate **NO_RETURN_FLOW** for that WBS and that surface runoff should re-infiltrate or a `0` to indicate that surface runoff should be handled as return flow. This is useful if a WBS is far from any SFR segments or it is known that the agricultural setting does not allow for runoff—which could be required by State or local regulations. If SFR package is not in use—as specified in the Name file—and the keyword **NO_RETURN_FLOW** is not included, then any resulting runoff is removed from the simulation domain and a warning is raised. If the keyword **NO_RETURN_FLOW** is present, then it supersedes all the other return flow keywords.

If surface runoff is aggregated as return flow, then it may either be applied to user-specified SFR segment and reaches or automatically applied to all SFR reaches in a WBS. A semi-routed return (SRR) flow point is an SFR segment and reach that receives the flow that results from surface runoff. As with the semi-routed delivery (SRD) input, a WBS may contain multiple defined semi-routed return flow points and has the same general input structure (fig. 6.64). The **GLOBAL DIMENSION** keyword **NSFR_RETURN** defines the *List Style* input number of records that **SEMI_ROUTED_RETURN** expects. Note that figure 6.64*A* has nearly an identical input structure for each record as figure 6.60*A* (SRD input), but their meanings are slightly different.

Any WBS that is not defined with **NO_RETURN_FLOW** or **SEMI_ROUTED_RETURN** may optionally use fully routed return flow to automatically determine the SFR return flow points. Fully routed return flow is similar to semi-routed return flow, except it automatically identifies all the SFR reaches in a WBS and prorates the return flow by the length of each reach. If fully routed return flow is enabled, but there are no SFR segments or reaches in the WBS, then the return flow is removed from the model—this is the same as specifying a semi-routed return flow to segment zero (that is, setting an `ISRR` with `WBS>0` and `SEG=0`). If a fully routed return flow has the return flow leave the model, then a warning is raised indicating that it be specified as a semi-routed return flow with a segment set to zero. Fully routed return flow is enabled if the keyword **ROUTED_RETURN_ANY_NON_DIVERSION_REACH** or **ROUTED_RETURN_ANY_REACH** is specified in the **SURFACE_WATER** block. The difference between the two is the former applies return flow to all non-diversion SFR reaches, and the latter applies return flow to any reach in the WBS. When SFR is part of the simulation, it is recommended to always include **SEMI_ROUTED_RETURN** and **ROUTED_RETURN_ANY_NON_DIVERSION_REACH**.

Figure 6.65 presents different combinations of return flow keywords that are applied to 5 WBS. Figure 6.65*A* is an example **GLOBAL DIMENSION** keyword that is used by the other figure parts. Figure 6.65*B* illustrates the use of **NO_RETURN_FLOW** and having it apply to all WBSs. Figure 6.65*C* shows how to specify **NO_RETURN_FLOW** by WBS. In this example WBSs 3, 4, and 5 do not have their return flow associated with an SRR point or have a fully-routed return flow keyword present, so if they have any return flow it will be removed from the simulation domain. Figure 6.65*D* enables fully-routed return flow, so if there are any non-diversion reaches in WBSs 3, 4 and 5, then runoff is returned to them. Figure 6.65*E* includes a single semi-routed return flow point for WBS 3. In this example (fig. 6.65*E*), the aggregated surface runoff from WBSs 4 and 5 is not defined with **NO_RETURN_FLOW** or **SEMI_ROUTED_RETURN**, but the fully routed keyword **ROUTED_RETURN_ANY_NON_DIVERSION_REACH** is present, so WBSs 4 and 5 will have their return flow length prorated across any non-diversion reaches in their respective subregions (that is, return flow from WBS 4 returns to reaches in WBS 4, and return flow from WBS 5 returns to reaches in WBS 5). If WBS 4 and 5 do not have any SFR reaches in them, then their return flow is removed from the model and a warning is raised. Figure 6.65*F* does not include **NO_RETURN_FLOW** nor any fully routed return flow keywords, so any WBS that is not defined with **SEMI_ROUTED_RETURN** has its return flow removed from the model. Because figure 6.65*F* does not specify **NO_RETURN_FLOW**, all surface runoff is aggregated as return flow, which is direction specified for WBS 3 and prorated across multiple SFR reaches for WBS 1, 2, 4 and 5.

Supported **SFAC** DIMKEY

NON_ROUTED_DELIVERY is the only **SURFACE_WATER** keyword that supports advanced, **SFAC** **DIMKEY**, scale factors. The scale factors that are loaded are only applied to `NRD_VOL`—the quantity of water delivered. **NON_ROUTED_DELIVERY** only supports the **DIMKEY** **ByWBS** and **ByNRD**. The **DIMKEY** **ByWBS** reads **NWBS** scale factors that are applied by WBS. The **DIMKEY** **ByNRD** reads **NRD_TYPES** scale factors that are applied by each NRD type for all WBS.

A

```
ISRR  WBS  SEG  RCH  [FRAC]
```

B

Input Format Keyword	Description
ISRR	*List Style* input record ID, **INT**, for the semi-routed return (SRR) flow location. Its value is read, but not used. It must be sequential from 1 to **NSFR_RETURN**.
WBS	The WBS ID, **INT**, that has a return flow location defined at SEG and RCH. A WBS may be specified in multiple ISRD records allowing for the WBS to have more than one semi-routed return flow locations. If the WBS is set to 0, then the rest of the input line is ignored. This allows for fewer than **NSFR_RETURN** to be in use during a stress period.
SEG	Is the SFR segment, **INT**, that receives return flow from the defined WBS. If WBS > 0 and SEG = 0, then the WBS's return flow is removed from the model. That is, FMP assumes it flows to a point outside the simulation's domain.
RCH	Is the semi-routed return flow's SFR reach, **INT**, that receives return flow from the defined WBS. If SEG > 0 and RCH = 0, then the return flow is length prorated to all the reaches in SEG. This allows for the ISRR to be defined for multiple segments without having to repeat for every reach.
FRAC	An optional input that specifies the fraction, **FLOAT**, of the total WBS return flow that is passed as runoff to the defined SEG and RCH as runoff. If a WBS only has one semi-routed return location, then FRAC is set to 1.0. For a given WBS, if FRAC is not specified or is set to a negative number, then the fraction is determined by prorating the reach length to the total length of all the WBS' reaches not specified with a FRAC.

Figure 6.64. SURFACE_WATER block SEMI_ROUTED_RETURN input definition. A total of NSFR_RETURN records are read with each record containing ISRR, WBS, SEG, RCH, and optionally FRAC. *A*, Expected input structure for each List Style input record. *B*, Explanation of input variables. [SEG and RCH may be replaced with a name defined in the GLOBAL DIMENSION block's keyword SFR_NAMES. Input enclosed in brackets, [ ], indicates it is optional. Abbreviations: FLOAT, floating-point number; INT, integer number, ID, water balance subregion identification number; WBS, water-balance subregion; SFR, streamflow routing package; [L^3/T], volumetric rate in model units]

A

```
BEGIN GLOBAL DIMENSION

  NWBS          5  #  Number of water balance subregions (FARMS)
  NRD_TYPES     0  #  Number of non-routed delivery types defined (NRD)
  NCROP         5  #  Assume the following are defined elsewhere
  NSOIL         1
  NIRRIGATE     1
  NSFR_DELIV    0  #  No SEMI_ROUTED_DELIVERY locations defined
  NSFR_RETURN  10  #  Number of SEMI_ROUTED_RETURN points defined

END GLOBAL DIMENSION
```

B

```
BEGIN SURFACE_WATER
  #
  #  Keyword forces all surface runoff re-infiltrates to groundwater as deep percolation
  #
  #  It supersedes all other return flow keywords (because there is no return flow)
  #  Note this makes the global dimension NSFR_RETURN = 10 as wasted spaces.
  #
  NO_RETURN_FLOW     # No LAI[S, T, L], indicates keyword applies to all WBS for all time steps
  #
END SURFACE_WATER
```

C

```
BEGIN SURFACE_WATER
  #
  #  The optional [ LAI[S, T, L] ] input is included, which specifies NO_RETURN_FLOW by WBS.
  #
  #  Input indicates that surface runoff re-infiltrates to deep percolation for WBSs 1 and 2
  #
  #  WBSs 3, 4 and 5 do not have specified a return flow point—no return flow,
  #     semi-routed return, or fully-routed return—so their return flow leaves the simulation domain
  #
  NO_RETURN_FLOW STATIC LIST INTERNAL
  1  1
  2  1
  3  0
  4  0
  5  0
  #
END SURFACE_WATER
```

Figure 6.65. Surface Water block return flow keyword examples. *A*, GLOBAL DIMENSION block input used for other figure parts. *B*, Surface Water block input that defines no return flow for the entire model. *C*, Surface Water block input that defines no return flow for select water balance subregions (WBS). *D*, Surface Water block input that defines no return flow for select WBS and includes a fully-routed return flow keyword. *E*, Surface Water block input that includes the three main types of return flow keywords. *F*, Surface Water block input that defines semi-routed return flow.

D

```
BEGIN SURFACE_WATER
  #
  #  All surface runoff re-infiltrates for WBS 1 and 2
  #
  #  WBSs 3, 4 and 5 do not have specified a return flow point,
  #      but the keyword ROUTED_RETURN_ANY_NON_DIVERSION_REACH is present to indicate that
  #      any WBS that does not have its return flow point defined uses fully-routed return flow.
  #      Consequently, any return flow from 3, 4 and 5 is prorated to SFR non-diversion segments
  #
  NO_RETURN_FLOW STATIC LIST INTERNAL
  1  1
  2  1
  3  0
  4  0
  5  0
  #
  ROUTED_RETURN_ANY_NON_DIVERSION_REACH  # Fully-routed return flow keyword
  #
END SURFACE_WATER
```

Figure 6.65. —Continued

E

```
BEGIN SURFACE_WATER
  #
  #  All surface runoff re-infiltrates for WBSs 1 and 2                          (no return flow)
  #  Any runoff from WBS 3 flows to SFR segment 12, reach 2                      (semi-routed return)
  #  Any runoff from WBSs 4 and 5 is prorated to SFR non-diversion segments      (fully-routed return)
  #
  NO_RETURN_FLOW STATIC LIST INTERNAL
  1  1
  2  1
  3  0
  4  0
  5  0
  #
  #  NSFR_RETURN = 10, so SEMI_ROUTED_RETURN reads 10 records (ISRR = 1 to 10)
  #
  SEMI_ROUTED_RETURN STATIC LIST INTERNAL
  #
  # ISRR  WBS  SEG   RCH  [FRAC]
  1      3    12   2        # WBS 3 Return Flow sent to SFR SEG 12, RCH 2 ←
  2      0     0   0        # WBS = 0, ISRR is ignored
  3      0     0   0
  4      0     0   0
  5      0     0   0
  6      0     0   0
  7      0     0   0
  8      0     0   0
  9      0     0   0
  10     0     0   0
  #
  ROUTED_RETURN_ANY_NON_DIVERSION_REACH
  #
END SURFACE_WATER
```

Figure 6.65. —Continued

F

```
BEGIN SURFACE_WATER
  #
  #  NSFR_RETURN = 10, so SEMI_ROUTED_RETURN reads 10 records (ISRR = 1 to 10)
  #
  SEMI_ROUTED_RETURN STATIC LIST INTERNAL
  #
  #ISRR WBS  SEG  RCH  [FRAC]
   1    1    0   0        # WBS 1: return flow leaves the model domain
   2    2    5   0        # WBS 2: return flow length prorated across all reaches of
   3    2    6   0        #          segments 5 and 6 (defined via ISRR 2 and 3).
   4    3    8   0 0.4    # WBS 3: 40% of return flow is sent to segment 8 and
                          #          length prorated among its reaches
   5    3    9   0 0.5    # WBS 3: 50% of return flow is sent to segment 9 and
                          #          length prorated among its reaches
   6    3   10   1 0.1    # WBS 3: 10% of return flow is sent to Seg 10, Reach 1
   7    4   15   4        # WBS 4: All return flow is sent to Seg 15, Reach 4
   8    5    1   1 -1.0   # WBS 5: FRAC  = -1 indicates that return flow is length prorated
   9    5    1   2 -1.0   #          across Seg 1, Reaches 1 and 2. (Note FRAC  = -1 is optional.)
  10    0    0   0        # ISRR is ignored due to WBS = 0
  #
END SURFACE_WATER
```

Figure 6.65. —Continued

Allotment Block

The **ALLOTMENT** block is an optional block that specifies global constraints of water supplies for a WBS. The allotments in the FMP provide a convenient method of imposing a maximum limit on the water sources available to a WBS. The use of allotments as constraints is useful for representing limits such as water rights or operating agreement allocations or for analysis of sustainability by limiting supply in a deficit irrigation option. The latter can be useful for analysis related to meeting statutory requirements such as under California's Sustainable Groundwater Management Act (SGMA; State of California, 2014). With this release of MF-OWHM2, there are two Allotment constraints that can be applied: one for SFR surface-water deliveries and another for total groundwater pumpage. The surface-water allotment imposes a limit on the amount of water a WBS can receive from the **SURFACE_WATER** block's **SEMI_ROUTED_DELIVERY**. The groundwater allotment imposes a limit on the amount of water a WBS can remove from groundwater through wells defined in the **SUPPLY_WELL** block. Allotment constraints are imposed by stress period, to limit the quantity of water withdrawals during a stress period. To convert an allotment to an equivalent volumetric rate is to divide the allotment volume by the stress-period length. Figure 6.66 presents the keywords that are supported in the **ALLOTMENT** block.

In figure 6.66, an X in the **SFAC DIMKEY** column indicates that it is supported in the **ALLOTMENT** block. Keywords not specified in the table do not support the **SFAC DIMKEY**s. Both **SURFACE_WATER** and **GROUNDWATER** support using the **SFAC** keyword, but only allow for the keyword option **ByWBS**. The option loads a list of **NWBS** scale factors.

First Keyword Second Keyword	Input	Description
SURFACE_WATER { **RATE**, **VOLUME**, **HEIGHT** }	LAI[S, T, L]	*List Style* reads **NWBS** records that specify, for each WBS, a total delivery limit from the WBS's **SEMI_ROUTED_DELIVERY**. If the WBS has multiple **SEMI_ROUTED_DELIVERY**'s, then their sum cannot exceed the limit. The second keyword is required and specifies the model units of the surface water allotment. Only one second keyword may be in use during a simulation. **RATE** allotment is a volumetric rate [L^3/T] **VOLUME** allotment is a volume per stress period [L^3] **HEIGHT** allotment is a length per stress period [L] that is multiplied by the WBS's irrigated area to obtain a volume per stress period.
GROUNDWATER { **RATE**, **VOLUME**, **HEIGHT** }	LAI[S, T, L]	*List Style* reads **NWBS** records that specify, for each WBS, a total pumpage limit from the WBS's **SUPPLY_WELLS** (that is, a global limit on a WBS groundwater pumping). The second keyword is required and specifies the model units of the groundwater allotment. Only one second keyword may be in use during a simulation. **RATE** allotment is a volumetric rate [L^3/T] **VOLUME** allotment is a volume per stress period [L^3] **HEIGHT** allotment is a length per stress period [L] that is multiplied by the WBS's irrigated area to obtain a volume per stress period.

Figure 6.66. ALLOTMENT block keyword list. Both the first and second keywords must be present to complete the input. [The first keyword is specified on the first line of each row and the second keyword options are enclosed in { }. Only one of the second keywords may be used during a simulation. Note that VOLUME and HEIGHT second keywords indicate that input is a stress period-based limit rather than a model time unit limit. Abbreviations: FMP, farm process; WBS, water-balance subregion; [L], length in model units; [L^3], volume in model units; [L^3/T], volumetric rate in model units; SFR, streamflow routing package; LAI, list-array input with the following letters representing the supported keywords: S for STATIC, T for TRANSIENT, A for ARRAY, and L for LIST]

First Keyword	DIMKEY		
	ByWBS	**ByCrop**	**ByIrrigate**
SURFACE_WATER	X		
GROUNDWATER	X		

Figure 6.67. ALLOTMENT block keywords that support **SFAC DIMKEY**. The only supported word for **DIMKEY** is ByWBS. [Supported keyword and **DIMKEY** combinations are marked with an X.]

Land-Use Block—Land-Use and Crop Properties

The **Land_Use** (Crop) block defines a land-use type and properties associated with each land use. Please review appendix 4 for a detailed explanation of the properties of the **Land_Use** block. Land use is the main determinant of water consumption in the FMP and consumes groundwater, precipitation, and irrigation water as evaporation and transpiration. In MF-OWHM2, the term "Crop" is synonymous with the term land use. A crop is simulated in the FMP only if it resides in a WBS—that is, its model location is defined with a WBS ID from 1 to **NWBS**. In FMP, the block name can be declared as **Land Use** or **Land_Use**, but the later version, with an underscore, is recommended for clarity (note that input is also not case sensitive).

The **Land_Use** block includes keywords that define crop names and locations, crop coefficients and irrigation methods, root depths and suction pressures, fractions of transpiration and evaporation for each crop/land-use type, losses to runoff from precipitation and irrigation waters, and the type of interaction with groundwater (see "appendix 4" for formal definitions of these properties). The keywords are briefly described here and, unless otherwise noted, all *List Style* input expects **NCROP** records, where **NCROP** is the number of land-use types and specified in the **GLOBAL DIMENSION** block.

Spatial Location Keywords

Land-use types can be specified on a cell-by-cell basis—that is, only one crop per model cell—or defined as a fraction of a model cell—allowing for multiple crops in the same model cell. Land-use fractions—keyword **MULTIPLE_LAND_USE_PER_CELL**—provide a means of simulating multiple crops within a model cell. This can be valuable if constructing composite properties is problematic or if a higher accuracy is desired for cells containing a diverse set of crops. A negative aspect to using land-use fractions is that the input structure becomes more challenging if an array-based input is desired. If the input uses crop fractions and one of its properties requests an array-based input, then it requires a two-dimensional array for each crop—that is, the non-fraction *Array Style* input is NROW × NCOL, but crop fraction *Array Style* input is NCROP × NROW × NCOL. *List Style* input does not present this challenge because it maps each crop's property to any model cell that contains the crop.

The **Land_Use** block must include either the keyword **SINGLE_LAND_USE_PER_CELL** or the keyword **MULTIPLE_LAND_USE_PER_CELL** to indicate if the input assumes one crop per model cell or allows for multiple crops per cell, respectively. If **SINGLE_LAND_USE_PER_CELL** is specified, then the keyword **LOCATION** `LAI[S, T, A]` is required and defines the spatial location of each crop. The **LOCATION** input is specified as a 2D **INT** array that contains the Crop number, from **1** to **NCROP**, where the crop is simulated; however, if a number less than 1 or greater than **NCROP** is input, then bare soil is simulated for that cell, as described in appendix 4. Crops are only simulated in model cells that are associated with a WBS—that is, the same row and column are defined in the **WATER_BALANCE_SUBREGION** **LOCATION** keyword with a WBS number between 1 and **NWBS**. If it is known that a crop does not occupy the entire model cell, then the keyword **LAND_USE_AREA_FRACTION** `LAI[S, T, A]` can be included to specify the fraction of the cell area that the crop is grown on—the fraction must be between **0** and **1**. The remaining part of the model cell, one minus the fraction, is treated as bare soil and only allows for evaporation. The fraction should specify the total cropped area and not the leaf covered area—the leaf covered area is defined with the Fraction of Transpiration as described in appendix 4. The **LAND_USE_AREA_FRACTION** keyword is optional when defining one crop per model cell. Figure 6.68 is an example input that assigns a single crop per model cell for a model grid with three rows and four columns.

The input is structured differently if it is desired to specify multiple crops per model cell. First the keyword **MULTIPLE_LAND_USE_PER_CELL** must be present, and the keyword **LAND_USE_AREA_FRACTION** is required. Because there are multiple crops per model cell the use of the keyword **LOCATION** is not allowed because it signifies assignment of a single crop per model cell. Instead, the **LAND_USE_AREA_FRACTION** usage specifies where the crop is and the fraction of cell area it occupies. As described before, the **LAND_USE_AREA_FRACTION** keyword requires reading **NCROP** NROW × NCOL arrays that are contiguous—making the final array size as NCROP × NROW × NCOL. The cell fractions can be specified with *IXJ Style* input, which is described in the "IXJ Style Input Support" section. If multiple crops are simulated and the sum of the specified cell fractions are not equal to 1, then the remaining area is treated as bare soil. Figure 6.69 is an example input that assigns multiple crops per model cell and is using the previous example's **GLOBAL DIMENSION** and **WATER_BALANCE_SUBREGION** blocks.

Because the FMP inputs allow for comments between input lines it may be advantageous to specify the fraction input with a full-line comment before each crop's input. Figure 6.70 illustrates how you can use full-line comments to break apart the different crops' cell-fraction arrays. Figure 6.71 summarizes the location-based keywords used by the **Land_Use** block.

```
# Example assumes that model grid is 3 by 4 (NROW = 3, NCOL = 4)
# Note only the necessary keywords are defined
BEGIN GLOBAL DIMENSION
  #
  NWBS    5   # Number of water balance subregions (WBS)
  NCROP   2   # Number of land use types (Crops)
  #
END GLOBAL DIMENSION
BEGIN WATER_BALANCE_SUBREGION
  #
  LOCATION  STATIC ARRAY INTERNAL # Only WBSs 1 and 5 are in use during the simulation
  0  1  5  5
  0  1  5  5
  0  5  5  5
  #
END WATER_BALANCE_SUBREGION
BEGIN LAND_USE
  #
  SINGLE_LAND_USE_PER_CELL # Indicates use of LOCATION keyword and only one crop per surface cell
  #
  # Only Crops 1 and 2 are in use
  # Column 1 is not simulated because it is not assigned to a WBS
  # (that is "WATER_BALANCE_SUBREGION LOCATION" = 0 in column 1)
  #
  # Row 3, Columns 2 and 3 are treated as bare soil, having no crop
  #
  LOCATION  STATIC ARRAY INTERNAL
  0  2  2  1
  0  2  2  1
  0  0  0  1
  #
  # The following is optional when using the LOCATION keyword.
  # The fractions define that Crop 2 located at Rows 1 and 2, column 2
  # occupies 60% of the model cell and 40% of the model cell is bare soil.
  # The fraction is meaningless on cells not assigned a Crop number.
  LAND_USE_AREA_FRACTION  STATIC ARRAY INTERNAL
  0  0.6  1.0  1.0
  0  0.6  1.0  1.0
  0  0    0    1.0
  #
END LAND_USE
```

Figure 6.68. Example FMP block input assigns a single crop per surface model cell for a model grid with three rows and four columns. [This example on includes the keywords necessary to specifies crop locations and fractions.]

```
# Example assumes that model grid is 3 by 4 (NROW = 3, NCOL = 4)
# Note only the necessary keywords are defined
BEGIN GLOBAL DIMENSION
  #
  NWBS    5   # Number of water balance subregions (WBS)
  NCROP   2   # Number of land use types (Crops)
  #
END GLOBAL DIMENSION
BEGIN WATER_BALANCE_SUBREGION
  #
  LOCATION  STATIC ARRAY INTERNAL  # Only WBSs 1 and 5 are in use during the simulation
  0  1  5  5
  0  1  5  5
  0  5  5  5
  #
END WATER_BALANCE_SUBREGION
BEGIN LAND_USE
  #
  # Indicates not to use of LOCATION keyword, but instead
  # LAND_USE_AREA_FRACTION specifies each crops location and allows for more than one crop in a surface cell
  MULTIPLE_LAND_USE_PER_CELL
  #
  # The fractions define that Crop 1 as occupying 60% of Rows 1 and 2, column 2;
  # Crop 2 then occupies 40% of Rows 1 and 2, column 2;
  # Crop 1 occupies all of Row 1, Column 3 and Rows 2 and 3, Column 4
  # Crop 1 occupies 50% of Row 1, Column 4 and the remaining area,
  #    except for Row 2, Column 3 (which is fully covered by Crop 2),
  #    is treated as bare soil—since it is undefined.
  #
  # Note that input expects an (NCROP × NROW) by NCOL array.
  #
  LAND_USE_AREA_FRACTION  STATIC ARRAY INTERNAL
  0  0.6  1.0  0.5    # Crop 1 Fractions ----------------------
  0  0.6  0    1.0
  0  0    0    1.0
  0  0.4  0    0      # Crop 2 Fractions ----------------------
  0  0.4  1.0  0
  0  0    0    0
  #
  END LAND_USE
```

Figure 6.69. Example FMP block input specifies multiple crops per surface model cell for a model grid with three rows and four columns. [This example on includes the keywords necessary to specifies crop locations and fractions.]

```
⋮
⋮  Example excerpt from LAND_USE block
⋮
#
LAND_USE_AREA_FRACTION  STATIC ARRAY INTERNAL
#
# Crop 1 Fractions
#
0  0.6  1.0  0.5
0  0.6  0    1.0
0  0    0    1.0
#
# Crop 2 Fractions      – ULOAD can skip commented or empty lines
#
0  0.4  0    0
0  0.4  1.0  0
0  0    0    0
#
⋮
⋮  Continue with the rest of the LAND_USE block
```

Figure 6.70. Example excerpt from the LAND_USE block input to illustrate using comments between lines of input. [ULOAD, the universal loader, automatically skips lines that are either blank or only contain a comment. If it encounters a line with input, then it expects to read the entire input before a comment appears. For example, ULOAD expects four numbers on a line, then if there is one number successful read, then all four must be present before a comment is placed.]

Keyword	Input	Description
{**SINGLE_LAND_USE_PER_CELL** or **MULTIPLE_LAND_USE_PER_CELL**}		One of these two keywords must be present within the **Land_Use** block. The keywords indicate if the FMP should simulate a single-land use per model cell or allow for more than one (multiple) land-use type per model cell.
LOCATION	LAI[S, T, A]	Required if **SINGLE_LAND_USE_PER_CELL** is specified. **INT** array that specifies the spatial location of the land-use types (Crops). Within the array an **INT** between 1 and **NCROP** indicates the crop that is grown there; otherwise, it is assumed to be bare-soil area.
LAND_USE_AREA_FRACTION	LAI[S, T, L]	**FLOAT** array that specifies the fraction of the model cell's surface area that the land use occupies. Optional, if **SINGLE_LAND_USE_PER_CELL** is specified and has a default value of 1.0 for all model cells. Array Style expects an array of fractions from 0.0 to 1.0 structured as NROW × NCOL. The portion of the cell that is not specified by a crop is simulated as bare-soil area. Required, if **MULTIPLE_LAND_USE_PER_CELL** is specified. Array Style expects an array of fractions from 0.0 to 1.0 structured as NCROP · NROW × NCOL. This structure can be thought of as **NCROP** arrays that are stacked one after the next with the dimension of NROW × NCOL. The crop's spatial location is signified by where the fraction is greater than 0. That is, crop 1 is located anywhere the fraction is greater than 0.0 in the first NROW × NCOL array, and crop 2 is located anywhere the fraction is greater than 0.0 in the second NROW × NCOL array. If the fractions do not sum to 1, then the remaining area is assumed to be bare soil.

Figure 6.71. LAND_USE block keywords that define crop locations. [FLOAT, floating-point number; INT, integer; FMP, farm process; WBS, water-balance subregion; LAI, list-array input with the following letters representing the supported keywords: S for STATIC, T for TRANSIENT, A for ARRAY, and L for LIST]

Base Land-Use Properties

The base land-use properties are the necessary features to describe the crop and how to simulate it. Some of the base properties are optional and have default values when not specified, whereas others are conditionally dependent—that is, describing one property may necessitate defining another. Figure 6.72 summarizes the keywords that define the base crop properties, and the rest of this section discusses in greater depth selected keywords.

A simulation that includes the **Land_Use** block must include either the **CONSUMPTIVE_USE** or **CROP_COEFFICIENT** keywords. **CONSUMPTIVE_USE** LAI[S, T, A, L] directly specifies for each crop the potential consumptive use [L/T], which is the same as potential evapotranspiration. **CROP_COEFFICIENT** specifies a crop coefficient (K_c) that is converted to a potential consumptive use (CU) by multiplying it by the crop's reference evapotranspiration (ET_{ref}), which is defined in the **CLIMATE** block with the keyword **REFERENCE_ET**. It is recommended to consult with local agricultural expertise and water districts to obtain crop coefficients for the simulated region. If local crop coefficient data are not available, then Figure 6.73 provides estimates of K_c at three growth stages of selected crops from Allen and others (1998).

If **CONSUMPTIVE_USE** and **CROP_COEFFICIENT** are specified, then the total potential CU is the sum of the two—that is, CU = **CONSUMPTIVE_USE** + **CROP_COEFFICIENT** × **REFERENCE_ET**. If the CU is set to zero, then the FMP assumes the land use does not have any evapotranspiration. The FMP treats CU as the crop demand and attempts to satisfy it directly from groundwater, precipitation, or applied irrigation.

The keyword **IRRIGATION** defines which crops have access to applied water. Applied water is only used when the crop's transpiration demand—calculated from its potential CU—is not fully met from the root uptake of groundwater or precipitation. The **IRRIGATION** keyword may only be specified if the **GLOBAL DIMENSION** block keyword **NIRRIGATE** is greater than **0**, which indicates the number of irrigation types available in the FMP. The input for **IRRIGATION** may be *Array Style* or *List Style* that specifies an integer from **0** to **NIRRIGATE**. If the **IRRIGATION** is set to **0**, then the crop does not have access to applied water—note that this is the opposite of previous versions of the FMP. If the **IRRIGATION** is greater than **0**, then the crop has irrigation and the integer specified refers to the irrigation type. If the keyword **EFFICIENCY**, from the **WATER_BALANCE_SUBREGION** block, uses *List Style* input, then the irrigation type refers to the column in the Irrigation Efficiency input to use (see the "Water Balance Subregion (WBS) Block" section). Additionally, the irrigation type determines the *List Style* record (row) when the keywords **EVAPORATION_IRRIGATION_FRACTION** and **SURFACEWATER_LOSS_FRACTION_IRRIGATION** are followed with the keyword **BY_IRRIGATE**—which indicates that the *List Style* input reads **NIRRIGATE** records. If **EFFICIENCY**, **EVAPORATION_IRRIGATION_FRACTION** and **SURFACEWATER_LOSS_FRACTION_IRRIGATION** are specified with *Array Style*, then the property is determined by the crop's spatial location in the model grid—that is, **IRRIGATION** only serves as a flag to indicate the crop is irrigated.

If a crop has access to irrigation only during part of the simulation, then the **IRRIGATION** flag needs to vary by stress period. During periods when the crop does not receive applied water, the **IRRIGATION** flag is set to **0,** and when it does have applied water, the flag is set to the irrigation type that supplies the crop. Figure 6.73 illustrates the relationship between **WATER_BALANCE_SUBREGION** block **EFFICIENCY** keyword and the **LAND_USE** block **IRRIGATION** keyword—note that the example does not include the complete FMP input for either the **WATER_BALANCE_SUBREGION** or **LAND_USE** input blocks, just the sections pertinent to this example. The keyword **TRANSIENT** with **DATAFILE** IRR_Flag.txt (fig. 6.73*B*) indicates the file IRR_Flag.txt is a direct data file (DDF, appendix 1). That is, the irrigation flag is specified in IRR_Flag.txt for each stress period, consecutively, in the same file (fig. 6.73*C*). It only shows nine monthly stress periods with three crops that are not irrigated in January nor February. It assumes that crop 2 is drip irrigated from March to August and then harvested in September—so there is no longer a need to irrigate. It assumes that Crop 3 has flood irrigation in May and June and then switches to sprinkler irrigation for July and August before being harvested in September.

A

Keyword	Input	Description
CONSUMPTIVE_USE	LAI[S, T, A, L]	**FLOAT** input that defines the potential consumptive use rate of the crop [L/T]. This value is multiplied by the crop area to get a volumetric rate [L^3/T]. Either **CONSUMPTIVE_USE** or **CROP_COEFFICIENT** must be specified if **NCROP** > 0. If both are specified, then their sum is the potential volumetric consumptive-use rate. If the sum is 0, then zero evapotranspiration occurs.
CROP_COEFFICIENT	LAI[S, T, A, L]	**FLOAT** input that defines crop coefficients (K_c) that are multiplied by reference evapotranspiration (ET_{ref}) to determine the potential consumptive use rate [L/T]. If specified, then it is required to specify in the **CLIMATE** block **REFERENCE_ET**. This value is multiplied by the crop area to get a volumetric rate [L^3/T]. Either **CONSUMPTIVE_USE** or **CROP_COEFFICIENT** must be specified if **NCROP** > 0. If both are specified, then their sum is the potential volumetric consumptive-use rate. If the sum is 0, then zero evapotranspiration occurs.
TRANSPIRATION_FRACTION	LAI[S, T, A, L]	**FLOAT** input that defines fraction of consumptive use (FTR) that corresponds to the potential transpiration of the crop. It is formally defined as the ratio of the Basal Crop Coefficient (K_{cb}) divided by the **CROP_COEFFICIENT**, that is FTR = K_{cb} / K_c. It can be viewed as the fraction of a unit-cropped area that is covered by crop foliage.
ROOT_DEPTH	LAI[S, T, A, L]	**FLOAT** input (>0) that defines the root depth below land surface for each crop. If the root depth is less than 1E-30, then it is changed to 1E-30.
CROP_NAME	ULOAD	Optional, **CHAR** input that assigns a name of each crop for output files (max size of 20 characters). If not specified, then the crop names default to "CROP_XX", where the XX is the crop number.

Figure 6.72. LAND_USE block keywords that define the base properties that should be included in a farm process (FMP) simulation. Secondary keywords enclosed in brackets, [ ], are optional and at most one can be used during a simulation. [Some of the keywords are specified over two lines because of their length, but should be a single, contiguous word in the input. Abbreviations: CHAR, character input, FLOAT, floating-point number; INT, integer number; LAI, list-array input with the following letters representing the supported keywords: S for STATIC, T for TRANSIENT, A for ARRAY, and L for LIST.]

B

Keyword	Input	Description
IRRIGATION	LAI[S, T, A, L]	**INT** input that defines whether a crop has irrigation water applied, and the irrigation type. Only specified if **NIRRIGATE** > 0. If **IRRIGATION** is set to 0, then the crop does not have irrigation water; otherwise, input must be specified between 1 and **NIRRIGATE** to indicate that the crop is irrigated with and irrigation efficiency defined by the corresponding column in the **EFFICIENCY** table (OFE). For example, if set to 3, then the crop is irrigated and uses the efficiency defined in the third column in the **EFFICIENCY** table. If not specified, then **IRRIGATION** = 0 for all crops.
EVAPORATION_ IRRIGATION_FRACTION	[**BY_CROP**, **BY_IRRIGATE**] LAI[S, T, A, L]	**FLOAT** input that defines the fraction of the unit-cropped area that is irrigated but is only exposed to sunlight and evaporates (FEI). This portion of the unit-cropped area has negligible transpiration. This fraction should be between 0 and less than one minus the **TRANSPIRATION_FRACTION** (1-FTR). If keyword is not specified or input specifies FEI < 0, then FEI = 1-FTR. FEI can be viewed as the portion of irrigation that is not available for consumption as transpiration (such as the space between row crops) and includes any water that evaporates before reaching the crop or evaporates from atop the crop's leaves. If FTR + FEI > 1, then the final consumptive use will be greater than what is specified in the input due to evaporation of irrigation water. The optional secondary keywords **BY_CROP** and **BY_IRRIGATE** indicate the number of records that *List Style* input reads and applies the input. **BY_CROP** expects **NCROP** records and applies the fractions by crop. **BY_IRRIGATE** expects **NIRRIGATE** records and applies the fractions by irrigation type. If neither of the **BY_** keywords are specified, then the default is **BY_CROP**.

Figure 6.72. —Continued

C

Keyword	Input	Description
SURFACEWATER_LOSS_ FRACTION_PRECIPITATION	LAI[S, T, A, L]	**FLOAT** input is a runoff coefficient for water that originated as precipitation (FIESWP). It is a fraction of precipitation that is not consumed by the crop that becomes surface runoff; one minus this fraction is the portion that becomes deep percolation. For example, a FIESWP value of `0.8` indicates that `80` percent of unconsumed (excess) precipitation becomes runoff and `20` percent infiltrates as deep percolation. If keyword is not specified, then the FIESWP is set to `0.8` if SFR is in use; otherwise, is set to zero.
SURFACEWATER_LOSS_ FRACTION_IRRIGATION	[**BY_CROP**, **BY_IRRIGATE**] LAI[S, T, A, L]	**FLOAT** input is a runoff coefficient for excess irrigation water (FIESWI). The excess irrigation is determined from the quantity of irrigation water and the irrigation type's **EFFICIENCY**. FIESWI specifies the portion of excess irrigation that becomes surface runoff. If keyword is not specified, then FIESWI is set to `0`. For example, a FIESWI value of `0.8` indicates that `80` percent of excess irrigation becomes runoff and `20` percent infiltrates as deep percolation. The optional secondary keywords **BY_CROP** and **BY_IRRIGATE** indicate the number of records that *List Style* input reads and applies the input. **BY_CROP** expects **NCROP** records and applies the fractions by crop. **BY_IRRIGATE** expects **NIRRIGATE** records and applies the fractions by irrigation type. If neither of the **BY_** keywords are specified, then the default is **BY_CROP**.

Figure 6.72. —Continued

Plant Group	Initial stage	Mid-season stage	Late-season stage
Alfalfa hay	0.401	1.201	1.151
Apples, cherries, pears	0.600	0.950	0.752
Apricots, peaches, stone fruit	0.550	0.900	0.652
Berries (bushes)	0.300	1.050	0.500
Cereal crops	0.300	1.150	0.400
Citrus	0.650	0.600	0.650
Cotton	0.000	1.200	0.700
Cucurbitaceae family	0.500	1.000	0.800
Fibre crops	0.350	1.200	0.700
Grapes-wine	0.300	0.700	0.450
Hops	0.300	1.050	0.850
Legumes (leguminosae)	0.400	1.150	0.550
Oil crops	0.350	1.150	0.350
Pistachios	0.400	1.100	0.450
Roots and tubers	0.500	1.100	0.950
Small vegetables	0.700	1.050	0.950
Solanum family	0.600	1.150	0.800
Turf grass	0.800	0.850	0.850
Walnut orchard	0.500	1.100	0.652

Figure 6.73. CROP_COEFFICIENT (K_c) values for select crops (Allen and others, 1998). [Initial stage refers to the start of the crop's life cycle after planting or when the crop ends its winter dormancy, mid-season stage refers to the peak of the crop's life cycle, and late season refers to the end of the crop's life cycle or just before the crop enters its winter dormancy (for more details see Allen and others, 1998).]

The keyword **TRANSPIRATION_FRACTION** is a required input that specifies the fraction of cropped area contributing transpiration (FTR). This fraction must be between `0` and `1` and is formally defined as the ratio of the Basal Crop Coefficient divided by the **CROP_COEFFICIENT** (FTR = K_{cb}/K_c) and is discussed in detail in appendix 4. If FTR is set to 0.0, then the unit cropped area only has evaporation losses and no transpiration. However, if FTR is set to 1.0, then the unit cropped area could potentially only have transpiration. If the actual transpiration is less than the potential CU, then there is a potential for evaporation. It is important to note that FMP only irrigates to satisfy transpiration, so larger values of FTR increase the FMP estimation of irrigation. Typically, in the early stages of plant growth the FTR is small and increases as the plant becomes more developed with more leaf matter. Although it is recommended to investigate locally applicable values of FTR based on the simulated region, a set of example values are presented in figure 6.75. These example values are calculated from basal and crop coefficients published in Allen and others (1998). Fallowed land can have a wide range of FTR depending on type and vigor of the natural vegetation growing on the idle landscape, and the season of year. Values of FTR can range from 0.1 to 0.75 for fallowed land depending on seasonal growth of the natural vegetation. If the land becomes completely devoid of plant life, then setting FTR = `0` is acceptable.

If a crop receives applied water (**IRRIGATION**), then the **EVAPORATION_IRRIGATION_FRACTION** keyword represents the fraction of evaporation from irrigation (FEI; appendix 4) for the crop. The FEI can be viewed as the fraction of a unit cropped area that receives irrigation water, is exposed to the sun and evaporates, but is not consumed as transpiration. The value of FEI is normally less than or equal to `1` – FTR but is not required. If FEI + FTR`>1`, then the final CU can be larger than what is specified in the input. This larger CU is the result of irrigation water that evaporates before it reaches the crop or evaporates while sitting atop of the leaves.

A

```
#  Multipart example to show the relationship between EFFICIENCY and IRRIGATION
#
#  Example simulates 9 stress periods, assumes a model grid that is 3 by 4, with
#     1 Water Balance Subregion (WBS), 3 simulated crops, and 3 irrigation types.
#  That is,  NPER = 9,  NROW = 3,  NCOL = 4,  NWBS = 1,  NCROP = 3,  NIRRIGATE = 3
⋮
⋮  Excerpt from WATER_BALANCE_SUBREGION block to defines EFFICIENCY
⋮
#  EFFICIENCY keyword specifies for each WBS the irrigation efficiency (OFE) for each irrigation type
#  List Style input reads NWBS records that have NIRRIGATE columns
#
#  Note, the irrigation style (drip, sprinkler, flood) is included for perspective on what the OFE represents,
#     but FMP does not simulate the actual irrigation and uses OFE to calculate irrigation losses.
#     In this manner, it is better to think of the irrigation types as "Irrigation Type 1" or "IRR1"
#
EFFICIENCY  STATIC LIST INTERNAL
#         Drip    Sprinkler   Flood
#  WBS    IRR1    IRR2        IRR3
   1      0.9     0.7         0.6
#
⋮
⋮  Continue with the rest of the WATER_BALANCE_SUBREGION block
```

B

```
#  Multipart example to show the relationship between EFFICIENCY and IRRIGATION
⋮
⋮  Excerpt from LAND_USE block to defines EFFICIENCY
⋮
#
#  IRRIGATION keyword specifies which Crops are irrigated and the irrigation type used.
#
#  TRANSIENT indicates input is read by stress period with the Transient File Reader or Direct Data File.
#  DATAFILE indicates "IRR_Flag.txt" is a Direct Data File, so read each stress period (SP) directly from that file.
#
IRRIGATION  TRANSIENT LIST DATAFILE  IRR_Flag.txt
#
⋮
⋮  Continue with the rest of the LAND_USE block
```

Figure 6.74. Example that demonstrates the relationship between FMP irrigation efficiency (OFE) and the irrigation flag. *A*, Partial input of the WATER_BALANCE_SUBREGION block. *B*, Partial input of the LAND_USE block. *C*, Direct Data File that specifies irrigation flags for three crops for nine stress periods.

c

```
#  File: IRR_Flag.txt  —  Direct Data File reads one List Style input at start of each stress period (SP).
#  Input assumes, NCROP = 3, so irrigation is specified for 3 crops  — Note that Crop 1 is never irrigated
# CropID     IrrigationFlag
    1        0       # No irrigation for any crops        SP 1  —  January
    2        0
    3        0
    #  SP 2  —  February
    1        0       # No irrigation for any crops
    2        0
    3        0
    #  SP 3  —  March
    1        0       #
    2        1       # Crop 2 has irrigation 1 (drip), which has EFFICIENCY = 0.9
    3        0       # No irrigation Crop 3
    #  SP 4  —  April
    1        0       #
    2        1       # Crop 2 has irrigation 1 (drip), which has EFFICIENCY = 0.9
    3        0       # No irrigation Crop 3
    #  SP 5  —  May
    1        0       #
    2        1       # Crop 2 has irrigation 1 (drip),    which has EFFICIENCY = 0.9
    3        3       # Crop 3 has irrigation 3 (Flood),  which has EFFICIENCY = 0.6
    #  SP 6  —  June
    1        0       #
    2        1       # Crop 2 has irrigation 1 (drip),    which has EFFICIENCY = 0.9
    3        3       # Crop 3 has irrigation 3 (Flood),  which has EFFICIENCY = 0.6
    #  SP 7  —  July
    1        0       #
    2        1       # Crop 2 has irrigation 1 (drip),        which has EFFICIENCY = 0.9
    3        2       # Crop 3 has irrigation 2 (Sprinkler), which has EFFICIENCY = 0.7
    #  SP 8  —  August
    1        0       #
    2        1       # Crop 2 has irrigation 1 (drip),        which has EFFICIENCY = 0.9
    3        2       # Crop 3 has irrigation 2 (Sprinkler), which has EFFICIENCY = 0.7
    #  SP 9  —  September
    1        0       # Crops have been harvested, so no irrigation for any crops
    2        0
    3        0
```

Figure 6.74. —Continued

Plant Group	Initial stage	Mid-season stage	Late-season stage
Alfalfa hay	0.751	0.958	0.957
Apples, cherries, pears	0.833	0.947	0.934
Apricots, peaches, stone fruit	0.818	0.944	0.923
Berries (bushes)	0.667	0.952	0.800
Cereal crops	0.500	0.957	0.625
Citrus	0.923	0.917	0.923
Cotton	0.100	0.958	0.714
Cucurbitaceae family	0.300	0.950	0.875
Fibre crops	0.429	0.958	0.714
Grapes-wine	0.500	0.929	0.889
Hops	0.500	0.952	0.941
Legumes (leguminosae)	0.375	0.957	0.909
Oil crops	0.429	0.957	0.714
Pistachios	0.500	0.955	0.889
Roots and tubers	0.300	0.909	0.895
Small vegetables	0.214	0.905	0.895
Solanum family	0.250	0.957	0.875
Turf grass	0.938	0.941	0.941
Walnut orchard	0.800	0.955	0.923

Figure 6.75. Example values of TRANSPIRATION_FRACTION (FTR) calculated from crop coefficients presented in Allen and others (1998). [Initial stage refers to the start of the crop's life cycle after planting or when the crop ends its winter dormancy, mid-season stage refers to the peak of the crop's life cycle, and late season refers to the end of the crop's life cycle or just before the crop enters its winter dormancy (for more details, see Allen and others, 1998).]

The FEI can vary on the basis of agricultural practices and the type and quality of irrigation equipment used. Although it is recommended to research locally applicable FEI values based on the simulated region, typical values of FEI are 0.05 to 0.1 for drip irrigation, 0.1 to 0.3 for sprinkler, and 0.3 to 1 – FTR for flood irrigation. If FEI is set to a negative value, then FMP automatically sets FEI = `1` – FTR. That is, setting FEI`<0` ensures that the sum of FEI and FTR is always unity.

As with FTR, increasing FEI increases the FMP estimated irrigation requirement. However, the estimated irrigation requirement is the same for all values of FTR and FEI so long as FEI = `1` – FTR. The reason for this relationship is that the corresponding evaporative loss from irrigation compliments the necessary irrigation to satisfy transpiration.

Excess water originates from precipitation that is not consumed by the crop. Excess water either becomes surface runoff, which is aggregated as return flow, or infiltrates to groundwater as deep percolation. In previous releases of the FMP, the excess water was called an "inefficient loss." The keyword **`SURFACEWATER_LOSS_FRACTION_PRECIPITATION`** (FIESWP) specifies the fraction of the excess water from precipitation that becomes surface runoff; consequently, 1 – FIESWP is the excess water that becomes deep percolation. The choice of FIESWP depends on the properties of the land surface. Typically, areas that are urban or hilly areas have large values of FIESWP (`>0.8`). Large values also can occur if the surface soil is not permeable or known to contain a lot of clay. Small values (`<0.2`) occur when the surface is very permeable, such as sandy beaches or other coarse-grained deposits. Developed agriculture designed to minimize runoff will consequently have very small values of FIESWP. Undeveloped riparian areas that contain phreatophytes also may have very small values. Although it is recommended to investigate locally applicable values of FIESWP for the simulated region, a set of example values are presented in figure 6.76. These values are best viewed as an initial estimate when there is no other information available. Further refinement is advised based on the soil permeability, landscape topography, and degree of development of the landscape.

Land-Use Category or Crop Type	FIESWP
Agricultural trees	0.50
Burned and barren	0.95
Carrots	0.20
Citrus and subtropical	0.90
Corn	0.10
Cotton	0.15
Crucifers	0.10
Deciduous fruits and nuts	0.50
Fallow	0.70
Field crops	0.60
Field crops-irrigated	0.50
Golf-course turf	0.60
Grain and hay crops	0.90
Legumes (leguminosae)	0.10
Native and natural vegetation	0.95
Nurseries	0.95
Pasture	0.95
Phreatophytes	0.10
Rice	0.80
Riparian	0.15
Roots and tubers	0.10
Strawberries	0.10
Tomato	0.10
Upland grasslands	0.90
Upland shrub lands	0.90
Urban	0.95
Vineyards	0.50
Woodland	0.90

Figure 6.76. Example values of SURFACEWATER_LOSS_FRACTION_PRECIPITATION (FIESWP) for different land use and crop types. [The fraction, FIESWP, represents the portion of excess water from precipitation that becomes surface runoff. These values are best viewed as an initial estimate when there is no other information available.]

As with **SURFACEWATER_LOSS_FRACTION_PRECIPITATION**, excess water that originates from irrigation is defined by a fraction of inefficient loss to surface water. The excess irrigation water is determined from the irrigation efficiency (**EFFICIENCY** keyword from the **WATER_BALANCE_SUBREGION** block). The keyword **SURFACEWATER_LOSS_FRACTION_IRRIGATION** specifies the fraction (FIESWI) of the excess water from irrigation that becomes surface runoff—that is, 1 – FIESWI is the fraction of excess water that becomes deep percolation. This input is not used by crops that do not have irrigation (**IRRIGATION** = **0**), and its input is only required when **NIRRIGATE** is greater than zero. **SURFACEWATER_LOSS_FRACTION_IRRIGATION** may optionally be followed by the post-keyword **BY_CROP** or **BY_IRRIGATE** to indicate if *List Style* input is mapped to model cells by crop type or irrigation type. If neither optional keyword is present, then **BY_CROP** is assumed. If the post-keyword is **BY_CROP**, then the FMP reads **NCROP** *List Style* input records and applies the FIESWI by crop type—unless the crop is not irrigated (then the value is never used). If the post-keyword is **BY_IRRIGATE**, then the *List Style* input reads **NIRRIGATE** records and applies the FIESWI by the crop's irrigation type that is defined with the keyword **IRRIGATION**. Typically, values of FIESWI are small (**<0.3**) because most irrigation systems are designed to infiltrate the majority of applied water. In addition, the input values may be small to reflect possible limitations on surface runoff, which could originate from State or local regulations to minimize pesticide runoff to nearby streams. The choice of FIESWI values depends the land surface is development and the properties of the irrigation method. Users are advised to investigate locally applicable values of FIESWI on the basis of the simulated region; nevertheless, a set of example values for different land uses—**BY_CROP** input option—are presented in figure 6.77 and for different irrigation methods—**BY_IRRIGATE** input option—are presented in figure 6.78.

Land-Use Category or Crop Type	FIESWI
Agricultural trees	0.05
Carrots	0.10
Citrus and subtropical	0.70
Corn	0.02
Cotton	0.80
Crucifers	0.05
Deciduous fruits and nuts	0.05
Field crops	0.40
Field crops-irrigated	0.40
Golf-course turf	0.05
Grain and hay crops	0.05
Legumes (leguminosae)	0.05
Nurseries	0.01
Pasture	0.10
Rice	0.75
Roots and tubers	0.05
Strawberries	0.01
Tomato	0.05
Urban	0.35
Vineyards	0.05

Figure 6.77. Example values of SURFACEWATER_LOSS_FRACTION_IRRIGATION (FIESWI) for different land use and crop types. [The fraction, FIESWI, represents the portion of excess water from irrigation that becomes surface runoff. These values are best viewed as an initial estimate when there is no other information available.]

Irrigation Type	FIESWI
Drip	0.01
Sprinkler-developed	0.10
Sprinkler-undeveloped	0.30
Flood-developed	0.10
Flood-undeveloped	0.50

Figure 6.78. Example values of SURFACEWATER_LOSS_FRACTION_IRRIGATION (FIESWI) for different irrigation types. [The fraction, FIESWI, represents the portion of excess water from irrigation that becomes surface runoff. These values are best viewed as an initial estimate when there is no other information available.]

ROOT_DEPTH defines the vertical depth below land surface to which a crop's root system extends, expressed as a positive number. The sum of root depth plus the capillary fringe height (**CAPILLARY_FRINGE** from the **SOIL** block) determines if there is direct uptake of groundwater through the roots and if the potential transpiration is reduced as a result of anoxia and soil-moisture stress. **ROOT_DEPTH** may be set to zero, but it is not recommended. If the advance option **ROOT_PRESSURE**, discussed in the next section, is included, then **ROOT_DEPTH** must be positive; if zero is specified for root depth, then an error is raised and the simulation stops.

Advanced Land-Use Properties

The advanced land-use properties are optional and allow for advanced simulation features or customization of land-use properties. Figure 6.79 presents the list of advanced keywords supported by the **Land_Use** block.

There are three methods available in the FMP to define the groundwater-root uptake and anoxia for each crop. The first method is no groundwater uptake, the second is a linear groundwater-root uptake and anoxia-limitation concept that accounts for the overlap between the capillary fringe and the root depth (*linear root response*), and the third is an analytical pseudo steady-state soil moisture, soil-stress concept (*analytical root response*). If the *analytical root response* is used, then the user is required to specify, with the **ROOT_PRESSURE** keyword, four stress-response root pressures, symbolized as PSI or ψ, and the soil type that the crop is grown in (**COEFFICIENT** from the **SOIL** block). The four PSI values represent the upper limit pressure head at which the root uptake becomes zero because of anoxia, then two pressure heads that represent the range of optimal root uptake, and then a lower limit pressure head that results in wilting (zero root uptake), respectively. It should be noted that for most crops that the **ROOT_PRESSURE** PSI values should all be negative, with the first value being the largest and the fourth value being the most negative—that is, they must be in descending order. Only the first two PSI values may be positive, which indicate that the crop or land use can tolerate ponding of water—that is, a water table above the land surface. If the first value is zero or positive, then the second value also must be zero or positive. If a crop has its four **ROOT_PRESSURE** PSI values set to **0.0**, then the FMP will use the *linear root response* method instead. It is recommended to inquire with local agriculture and water districts to obtain the appropriate pressures for each crop or land use within a simulated region. If locally applicable data are not available, then figure 6.80 provides estimates of **ROOT_PRESSURE** for select crops.

The **GROUNDWATER_ROOT_INTERACTION** keyword defines how a crop's roots directly interact with groundwater. The FMP is capable of simulating five groundwater-root interaction levels that are outlined in figure 6.81. The different levels determine if a crop's root is capable of consuming groundwater (root groundwater uptake = Yes), if it has its CU reduced from anoxia (anoxia reduction = Yes), or a reduction in CU as a result of soil related stresses (soil-stress reduction = Yes). The default setting is level 5 in FMP and was the only option in previous FMP releases. The **GROUNDWATER_ROOT_INTERACTION** keyword only supports *List Style* input that loads for each crop type an **INT**, which must be a **0**, **1**, **2**, **3**, **4**, or **5**. If the level is set to **1** or **2**, then the crop can only meet its CU through precipitation and irrigation. If **GROUNDWATER_ROOT_INTERACTION** is not specified, then it defaults to level 5 for all crops.

The **ADDED_DEMAND** keyword allows for specifying additional irrigation, in addition to the requirement for CU, that is to be applied to a land use. When additional irrigation demand is specified, the FMP applies water to the land use that either becomes runoff or deep percolation. The amount that becomes runoff is defined by the **WATER_BALANCE_SUBREGION** block keyword **ADDED_DEMAND_RUNOFF_SPLIT**. If it is not specified, then by default all additional irrigation water required will be split to have 10 percent become runoff and 90 percent infiltrated—that is, the default **ADDED_DEMAND_RUNOFF_SPLIT** = **0.1**. This keyword is advantageous if it is known that a land use receives additional irrigation, such as pre-wetting a field, beyond meeting the CU requirement of the land use—that is, the demand specified through **CONSUMPTIVE_USE** or **CROP_COEFFICIENT**.

The **ADDED_DEMAND** keyword is identical to the **ADDED_CROP_DEMAND** keyword from the **WATER_BALANCE_SUBREGION** block, except that for **ADDED_DEMAND** the expected *List Style* input is **NCROP** records with **NWBS** columns (the transpose of **ADDED_CROP_DEMAND**) or the **ADDED_DEMAND** can be specified with *Array-Style*. **ADDED_DEMAND** has two possible option keywords that indicate how the input is interpreted. **ADDED_DEMAND** **FLUX** indicates that the input is in units of length (L/T) and multiplied by the crop's irrigated area to determine the additional demand as a volumetric rate (L^3/T). **ADDED_DEMAND** **RATE** indicates that the input is in volumetric rate (L^3/T) units. A simulation may only specify **ADDED_DEMAND** **FLUX** or **ADDED_DEMAND** **RATE**, but not both at the same time. If a simulation includes **ADDED_DEMAND** and the **WATER_BALANCE_SUBREGION** block **ADDED_CROP_DEMAND**, and they both are applied to the same land use, then their sum determines the total additional irrigation that the FMP will apply to the crop.

If a cell's CU is set to zero, then the FMP disables simulation of evaporation and transpiration of the model cell. The **ZERO_CONSUMPTIVE_USE_BECOMES_BARE_SOIL** keyword tells the FMP that if a land use has zero CU, then it should switch to bare-soil calculation. The bare-soil calculation only allows for evaporation of groundwater and precipitation, which is limited by either the **POTENTIAL_EVAPORATION_BARE** or one-half of the **REFERENCE_ET**. The keyword **ZERO_CONSUMPTIVE_USE_BECOMES_BARE_SOIL** may optionally be followed by a *List Style* input that consists of **NCROP** records that either specify a **1**, to indicate the crop is treated as bare soil, or **0** to indicate the crop is not.

A

Keyword	Input	Description
ROOT_PRESSURE	LAI[S, T, L-4]	**FLOAT** input that defines crop stress-response pressures [L] for the *analytical root response* option (appendixes 4 and 5). If keyword is specified, then the **SOIL** block must specify the **COEFFICIENT** keyword. Input expects four root pressures, ψ (PSI), that are species in units of hydraulic head [L]. The four root pressures specify the extent to which the crop is under (1) anoxia conditions, (2 and 3) optimally consumes groundwater, or (4) undergoes wilting. For any crop that has all four root pressures set to zero, or if the keyword is not specified, then FMP uses the *linear root response* concept.
GROUNDWATER_ROOT_INTERACTION	LAI[S, T, L]	**INT**, optional flag that specifies the crop root processes that are simulated, which are a function of the water table elevation and capillary fringe. The first process is the ability to consume groundwater directly called root-groundwater uptake (RGU). No RGU indicates that the crop's potential transpiration can only be satisfied from surface sources (precipitation and irrigation). The other processes reduce in potential transpiration due to anoxia or soil-stress. The flag must be set to one of the following levels: Level RGU Anoxia Soil-Stress 1 no no no 2 no yes yes 3 yes no no 4 yes no yes 5 yes yes yes It is recommended to only use option 1, 3, or 5. If keyword is not specified, the default value is 5.

Figure 6.79. LAND_USE block advanced keywords. All are optional and provide additional land-use properties or alter the behavior or the landscape simulation. Secondary keywords enclosed in curly braces, { }, indicate that only one of the keywords may be selected and used during a simulation. LAI input enclosed in curly braces, { }, indicate that it is optional. [FLOAT, floating-point number; INT, integer number; [L] is length in model units; [L/T], length per time in model units; [L^3/T], volumetric rate in model units; FMP, farm process; WBS, water-balance subregion; NWBS, maximum number of water-balance subregions; LAI, list-array input with the following letters representing the supported keywords: S for STATIC, T for TRANSIENT, A for ARRAY, and L for LIST]

B

Keyword	Input	Description
ADDED_DEMAND	{ **FLUX**, **RATE** } LAI[S, T, A, L-**NWBS**]	**FLOAT** input that defines additional irrigation demand for a crop, specified by each WBS. This is identical to the **WATER_BALANCE_SUBREGION** block keyword, **ADDED_CROP_DEMAND**, except here it is specified with **NCROP** records or *Array-Style*. **FLUX** indicates that input is specified as [L/T] and multiplied by the crop area to get [L^3/T]. **RATE** indicates that input is specified as [L^3/T].
POND_DEPTH	LAI[S, T, L]	**FLOAT** input that defines a depth of water above land surface [L] that a crop can tolerate before death occurs (no transpiration). This only applies to *linear root response* concept crops. That is, **ROOT_PRESSURE** is not specified or all four **ROOT_PRESSURE**'s set to zero. If not specified, the ponding depth tolerance is zero.
MIN_BARE_FRACTION	**FLOAT**	**FLOAT** input that specifies a minimum required fraction of cell area to simulate bare soil. This only pertains to model cells that are not defined with a land use (Crop). For example, if a model cell has two crops that fractionally compose `0.66` and `0.33` of the model-cell area, then the bare-soil area is `0.01` of model-cell area. If the minimum fraction of area is set to `0.02`, then FMP ignores the bare-soil portion of this example model cell. If keyword is not specified, then the default minimum fraction is `1E-12`.

Figure 6.79. —Continued

C

Keyword	**ZERO_CONSUMPTIVE_USE_BECOMES_BARE_SOIL** { LAI[S, T, L] }
Description	Keyword indicates that when a land use has a consumptive use equal to zero, then it should be treated as bare-soil area. This only allows for evaporation with a potential rate equal to half of reference evapotranspiration or the potential evaporation rate. If LAI[S, T, L] is specified, then *List Style* input reads **NCROP** **INT** flags to indicate which crops use this option. The flag must be set to: 0 to not use option for crop. 1 to use the option for crop. If keyword is not specified, then when the consumptive use is set to zero there is no evaporation nor transpiration possible because the potential is zero.
Keyword	**EVAPORATION_IRRIGATION_FRACTION_SUM_ONE_CORRECTION** { LAI[S, T, L] }
Description	Keyword indicates that when the sum of **TRANSPIRATION_FRACTION** (FTR) with **EVAPORATION_IRRIGATION_FRACTION** (FEI) is greater than 1, then FEI should be reduced to make the sum equal 1. That is, if FTR + FEI > 1, then FEI = 1 - FTR If FTR + FEI > 1, then actual consumptive use can be larger than the specified potential consumptive use due to irrigation evaporation. If LAI[S, T, L] is specified, then the *List Style* reads **NCROP** **INT** flags to indicate which crops use this option. The flag must be set to: 0 to not use option for crop. 1 to use the option for crop. If keyword is not specified, then input allows for FTR + FEI > 1.

Figure 6.79. —Continued

By default, if a crop's sum of **TRANSPIRATION_FRACTION** (FTR) with **EVAPORATION_IRRIGATION_FRACTION** (FEI) is greater than 1, no correction is made. The FMP assumes the user intends to simulate over-evaporation from FEI because of irrigation evaporating before reaching the crop or evaporation from the surface of the leaves. If it is desired to never allow the sum to be greater than 1, the keyword **EVAPORATION_IRRIGATION_FRACTION_SUM_ONE_CORRECTION** indicates that a check should be made. With this keyword, if FTR + FEI>1, then FEI is reduced so that FTR + FEI = 1. The keyword **EVAPORATION_IRRIGATION_FRACTION_SUM_ONE_CORRECTION** may optionally be followed by a *List Style* input that reads **NCROP** records that are either a 1, to indicate the crop is to be checked for the correction, or 0 to indicate the crop does not need to be checked.

Land-Use Category or Crop Type	PSI 1 [m]	PSI 2 [m]	PSI 3 [m]	PSI 4 [m]
Alfalfa	–0.15	–0.30	–5.45	–80.0
Apples	–0.13	–0.27	–6.94	–115.0
Apricots	–0.15	–0.30	–5.45	–80.0
Artichokes	–0.15	–0.30	–5.45	–80.0
Barley	–0.15	–0.30	–5.45	–80.0
Berries	–0.15	–0.30	–5.45	–80.0
Broccoli	–0.15	–0.30	–5.45	–80.0
Carrots	–0.13	–0.28	–11.40	–80.0
Cauliflower	–0.13	–0.28	–11.40	–80.0
Cherries	–0.13	–0.28	–11.40	–80.0
Citrus and subtropical	–0.15	–0.30	–6.00	–80.0
Cotton	0.07	0.00	–27.83	–153.3
Deciduous forest	0.13	0.00	–5.45	–80.0
Deciduous fruits and nuts	–0.13	–0.27	–6.94	–115.0
Dry beans	–0.13	–0.27	–6.94	–11.0
Durum wheat	–0.15	–0.30	–5.45	–80.0
Evergreen forest	–0.15	–0.30	–5.45	–80.0
Fallow or idle cropland	–0.15	–0.30	–5.45	–80.0
Field crops	–0.15	–0.30	–30.00	–123.7
Grapes-wine	–0.15	–0.30	–7.25	–80.0
Grassland herbaceous	–0.15	–0.30	–5.45	–80.0
Hay-general	–0.15	–0.30	–30.00	–123.7
Irrigated row and field crops	–0.13	–0.28	–11.40	–80.0
Mixed forest	–0.15	–0.30	–5.45	–80.0
Nectarines	–0.13	–0.27	–11.40	–80.0
Nurseries	–0.15	–0.30	–5.45	–80.0
Oats	–0.15	–0.30	–5.45	–80.0
Olives	0.15	0.10	–0.30	–0.4
Onions	–0.15	–0.30	–7.25	–80.0
Open water	0.50	0.10	–0.30	–0.4
Pasture-grass	–0.15	–0.30	–52.10	–160.1
Peaches	–0.15	–0.30	–5.45	–80.0
Phreatophytes	0.50	0.13	–8.25	–115
Pistachios	–0.15	–0.30	–52.10	–160.1
Potatoes	–0.15	–0.30	–6.00	–80.0
Rice	0.50	0.13	–1.78	–160
Shrubland	–0.15	–0.30	–5.45	–80.0
Strawberries	–0.15	–0.30	–5.45	–80.0
Walnuts	–0.15	–0.30	–5.45	–80.0

Figure 6.80. Example values of ROOT_PRESSURE for different land use and crop types. [The ROOT_PRESSUREs are in meters [m] of hydraulic head. PSI 1, 2, 3, and 4 represent the upper limit pressure head at which the root uptake becomes zero because of anoxia, then two pressure heads that represent the range of optimal root uptake, and then a lower limit pressure head that results in wilting (zero root uptake), respectively. These values are best viewed as an initial estimate when there is no other information available.]

Level	Root Groundwater Uptake	Anoxia Reduction	Soil-Stress Reduction
0	—	—	—
1	No	No	No
2	No	Yes	Yes
3	Yes	No	No
4	Yes	No	Yes
5	Yes	Yes	Yes

Figure 6.81. GROUNDWATER_ROOT_INTERACTION supported options. The List Style input must set the option for each crop to one of the defined Levels. [Root Groundwater Uptake indicates that the farm process (FMP) will evaluate if the water table is close enough to the crop that its roots can take up water directly from groundwater. Anoxia Reduction indicates that the FMP will evaluate if the crop's potential consumptive use is reduced as a result of anoxia conditions. Soil-Stress Reduction indicates that the FMP will evaluate if the crop's potential consumptive use is reduced as a result of non-ideal soil-moisture conditions. Abbreviation: —, no input values apply]

Output Keywords

The **LAND_USE** block additionally has a set of optional, output keywords that provide detailed information of the CU, CU reduction, irrigation, infiltration, and runoff associated with each land use (fig. 6.82). The difference among the output options is how the output data are sampled—for example, it can be presented by WBS, by land-use type, or by each land-use type in each WBS. To initiate the output options, write in the **LAND_USE** block the keyword **PRINT** followed by the output type and then the output file, specified with *Generic_Output_OptKey*. The available output types are specified by the option keywords **ByWBS**, **ByWBS_ByCROP**, **ByCROP**, **BARE**, **ALL**, **ALL_VERBOSE**, and **INPUT**. The keyword **PRINT INPUT** provides an echo of the input for checking if there are input file errors. The keywords **PRINT ByWBS**, **PRINT ByWBS_ByCROP**, and **PRINT ByCROP** provide the same output, but will aggregate it to be by WBS, by CROP or all crops in a WBS. It is recommended for all FMP simulations to include the **PRINT ByWBS_ByCROP** keyword to evaluate each land-use's consumption by WBS.

The keywords **PRINT ALL** and **PRINT ALL_VERBOSE** provide cell-by-cell output for all crops. The difference between the **ALL** and **ALL_VERBOSE** is the amount of output. The **ALL_VERBOSE** option provides output for almost every crop property from its raw input to its simulated results. This can be advantageous for finding errors in the simulation, but caution is advised because the file size can be enormous. To prevent the **PRINT ALL** and **PRINT ALL_VERBOSE** output files from becoming too large, the keyword **SPECIFY_PRINT_ALL_CROPS** `LAI[S, T, L]` allows for specifying which crops should be printed. **SPECIFY_PRINT_ALL_CROPS** input is *List Style* that consists of **NCROP** records that specify a `1` if the crop is to be included in the output or `0` if it is not. This allows the **PRINT ALL** and **PRINT ALL_VERBOSE** output files to only include cell-by-cell output for cells that contain specific crops.

PRINT ByWBS_ByCROP Header

The keyword **PRINT ByWBS_ByCrop** writes to the *Generic_Output_OptKey* file the summarized crops' simulated results for each WBS for every time. If a WBS had bare-soil calculations, then those are included with a Crop ID set to `0` and crop name of `BARE_LAND`. The keyword **PRINT ByWBS** has the same headers, except it does not include the CROP header because it aggregates all crops and bare soil together by WBS. The keyword **PRINT ByCrop** has the same headers, except it does not include the WBS header because it aggregates information across all WBS for each crop. The output for **PRINT ByWBS_ByCrop** may be in text or binary format and has the header defined in figure 6.83.

Keyword Input	Description
PRINT ByWBS_ByCROP	Summarize crop output information by WBS and by crop. Each crop within the WBS is summarized separately. Bare-soil evaporation by WBS also is included in the output (Recommended output option).
PRINT ByWBS	Summarize crop output information by the WBS. All crops in the WBS are summed to report a total quantity. Bare-soil evaporation by WBS also is included in the output.
PRINT ByCROP	Summarize crop output information by crop type across the entire simulation domain.
PRINT BARE	Summarize bare-soil calculation and simulation output for cells that contain bare soil. Output only includes model cells that had bare-soil calculations.
PRINT ALL	Write simulated results for each crop for every model cell that contains the crop.
PRINT ALL_VERBOSE	Write input and simulated results for each crop for every model cell that contains the crop.
PRINT INPUT	Write the land-use block input that is loaded.
SPECIFY_PRINT_ALL_CROPS LAI[S, T, L]	Does not specify an output file, but instead defines which crops should be included in output written for the keywords **PRINT ALL** and **PRINT ALL_VERBOSE**. *List Style* input; FMP reads **NCROP** records that have a 1, if the crop is to be included, or 0 if it is not. This helps prevent the PRINT ALL files from becoming too large.

Figure 6.82. LAND_USE block optional output keywords. [All the PRINT keywords should be followed by a GENERIC_OUTPUT_OptKey, which is the file that the output is written to. The only keyword not associated with an output file is SPECIFY_PRINT_ALL_CROPS, which specifies what crops should be included as part of the PRINT ALL and ALL_VERBOSE options. Abbreviation: WBS, water-balance subregion; LAI, list-array input with the following letters representing the supported keywords: S for STATIC, T for TRANSIENT, and L for LIST]

A

PER	is the stress period number	
STP	is the time step number	
WBS	is the Water Balance Subregion (Farm) ID number	
CROP	is the Land-use (Crop) ID number	
CROP_NAME	is the name of the specific crop	
AREA	is the area of land the crop occupies for the specified WBS	[L^2]
IRRIGATED_AREA	is the area of land the crop occupies that is irrigated for the specified WBS	[L^2]
EFFICIENCY	is the calculated irrigation efficiency (OFE), which is the consumptive irrigation requirement / irrigation water demand.	
CU_INI	is the initial-potential consumptive use of crop, which is the user-specified input CU plus $K_c \cdot ET_{ref}$	[L^3/T]
CU	is the actual crop consumptive use based on water supply	[L^3/T]
CIR_INI	is the crop irrigation requirement to meet CU_INI, assuming no irrigation losses	[L^3/T]
CIR	is the crop irrigation requirement, assuming no irrigation losses, to meet CU_INI after it has been reduced by anoxia and soil-stresses	[L^3/T]
DEMAND_INI	is the irrigation demand to meet CIR_INI considering irrigation losses	[L^3/T]
DEMAND	is the irrigation demand, and actual water supplied, necessary to satisfy CIR (includes irrigation losses).	[L^3/T]
ADDED_DEMAND_INI	is the initial additional irrigation demand specified for the crop	[L^3/T]
ADDED_DEMAND	is the final additional irrigation demand that was satisfied with supply	[L^3/T]
TOT_DEEP_PERC	is the total loss of water to deep percolation (groundwater recharge)	[L^3/T]
TOT_SURF_RUNOFF	is the total loss of water to surface runoff	[L^3/T]
ADDED_DMD_DPERC	is the amount of ADDED_DEMAND irrigation that is lost as deep percolation	[L^3/T]
ADDED_DMD_RUNOFF	is the amount of ADDED_DEMAND irrigation that is lost as surface runoff	[L^3/T]
TRAN_POT	is the potential transpiration from the cropped area for the given CU_INI	[L^3/T]
ANOXIA_LOSS	is the reduction in TRAN_POT due to plant anoxia (high groundwater level)	[L^3/T]
SOIL_STRESS_LOSS	is the reduction in TRAN_POT due to non-idealized soil stress	[L^3/T]
TRAN	is the actual crop transpiration (TRAN_IRR + TRAN_PRECIP + TRAN_GW)	[L^3/T]
TRAN_SURF_INI	is the initial-potential transpiration from precipitation and irrigation water	[L^3/T]
TRAN_SURF	is actual transpiration from precipitation and irrigation water	[L^3/T]
TRAN_IRR	is the portion of transpiration (TRAN) from irrigation water supply	[L^3/T]
TRAN_PRECIP	is the portion of transpiration (TRAN) from precipitation	[L^3/T]
TRAN_GW	is the portion of transpiration (TRAN) from groundwater	[L^3/T]
EVAP_IRR	is the evaporation of water that originated from irrigation water supply	[L^3/T]
EVAP_PRECIP	is the evaporation of water that originated from precipitation	[L^3/T]
EVAP_GW	is the evaporation of water that originated from groundwater	[L^3/T]
PRECIPITATION	is the total precipitation	[L^3/T]
DELT	is the time step length	[T]
DYEAR	is the date at the end of the time step as a decimal year	
DATE_START	is the calendar date at the start of the time step (yyyy-mm-ddThh:mm:ss)	

Figure 6.83. Land_Use block keyword PRINT ByWBS_ByCrop text file header explanation and binary record structure. *A*, Text-header explanation. *B*, Binary-record structure. [[T], unit of time in model units; [L^3/T], volumetric rate in model units; CHARACTER(20) indicates record is 20 characters long (20 bytes); CHARACTER(19) indicates record is 19 characters long (19 bytes); INTEGER is a 4-byte integer record; DOUBLE is a 8-byte floating-point number record.]

B

DATE_START	CHARACTER(19), starting date formatted as 'yyyy-mm-ddThh:mm:ss'
DYEAR	DOUBLE
DELT	DOUBLE
PER	INTEGER
STP	INTEGER
WBS	INTEGER
CROP	INTEGER
CROP_NAME	CHARACTER(20)
AREA	DOUBLE
IRRIGATED_AREA	DOUBLE
EFFICIENCY	DOUBLE
CU_INI	DOUBLE
CU	DOUBLE
CIR_INI	DOUBLE
CIR	DOUBLE
DEMAND_INI	DOUBLE
DEMAND	DOUBLE
ADDED_DEMAND_INI	DOUBLE
ADDED_DEMAND	DOUBLE
TOT_DEEP_PERC	DOUBLE
TOT_SURF_RUNOFF	DOUBLE
ADDED_DMD_DPERC	DOUBLE
ADDED_DMD_RUNOFF	DOUBLE
TRAN_POT	DOUBLE
ANOXIA_LOSS	DOUBLE
SOIL_STRESS_LOSS	DOUBLE
TRAN	DOUBLE
TRAN_SURF_INI	DOUBLE
TRAN_SURF	DOUBLE
TRAN_IRR	DOUBLE
TRAN_PRECIP	DOUBLE
TRAN_GW	DOUBLE
EVAP_IRR	DOUBLE
EVAP_PRECIP	DOUBLE
EVAP_GW	DOUBLE
PRECIPITATION	DOUBLE

Figure 6.83. —Continued

PRINT ALL Header

The keyword **`PRINT ALL`** writes to the *Generic_Output_OptKey* file the crop and bare-soil information for every model cell. Bare soil is identified with a crop ID of `0` and crop name of `BARE_LAND`. The output for **`PRINT ALL`** may be in text or binary format. The output may be text or binary format and has the header defined in figure 6.84.

A

PER	is the stress period number	
STP	is the time step number	
CROP	is the Land-use (Crop) ID number	
CROP_NAME	is the name of the specific crop	
WBS	is the Water Balance Subregion (Farm) ID number	
ROW	is the model row number	
COL	is the model column number	
AREA	is the area of land the crop occupies for the specified WBS	[L^2]
EFFICIENCY	is the irrigation efficiency (OFE)	
CU_INI	is the initial-potential consumptive use of crop, which is the user-specified input CU plus $K_c \cdot ET_{ref}$	[L^3/T]
CU	is the actual crop consumptive use based on water supply	[L^3/T]
CIR_INI	is the crop irrigation requirement to meet CU_INI, assuming no irrigation losses	[L^3/T]
CIR	is the crop irrigation requirement, assuming no irrigation losses, to meet CU_INI after it has been reduced by anoxia and soil-stresses	[L^3/T]
DEMAND_INI	is the irrigation demand to meet CIR_INI considering irrigation losses	[L^3/T]
DEMAND	is the irrigation demand necessary to satisfy CIR (includes irrigation losses).	[L^3/T]
ADDED_DEMAND_INI	is the initial additional irrigation demand specified for the crop	[L^3/T]
ADDED_DEMAND	is the final additional irrigation demand that was satisfied with supply	[L^3/T]
TOT_DEEP_PERC	is the total loss of water to deep percolation (groundwater recharge)	[L^3/T]
TOT_SURF_RUNOFF	is the total loss of water to surface runoff	[L^3/T]
ADDED_DMD_DPERC	is the amount of ADDED_DEMAND irrigation that is lost as deep percolation	[L^3/T]
ADDED_DMD_RUNOFF	is the amount of ADDED_DEMAND irrigation that is lost as surface runoff	[L^3/T]
TRAN_POT	is the potential transpiration from the cropped area for the given CU_INI	[L^3/T]
ANOXIA_LOSS	is the reduction in TRAN_POT due to plant anoxia (high groundwater level)	[L^3/T]
SOIL_STRESS_LOSS	is the reduction in TRAN_POT due to non-idealized soil stress	[L^3/T]
TRAN	is the actual crop transpiration (TRAN_IRR + TRAN_PRECIP + TRAN_GW)	[L^3/T]
TRAN_SURF_INI	is the initial-potential transpiration from precipitation and irrigation water	[L^3/T]
TRAN_SURF	is actual transpiration from precipitation and irrigation water	[L^3/T]
TRAN_IRR	is the portion of transpiration (TRAN) from irrigation water supply	[L^3/T]
TRAN_PRECIP	is the portion of transpiration (TRAN) from precipitation	[L^3/T]
TRAN_GW	is the portion of transpiration (TRAN) from groundwater	[L^3/T]
EVAP_IRR	is the evaporation of water that originated from irrigation water supply	[L^3/T]
EVAP_PRECIP	is the evaporation of water that originated from precipitation	[L^3/T]
EVAP_GW	is the evaporation of water that originated from groundwater	[L^3/T]
PRECIPITATION	is the total precipitation	[L^3/T]
GW_HEAD	is the groundwater elevation (head) beneath the crop	[L]
GSE	is the ground surface elevation	[L]
ROOT_ELEV	is the lowest elevation of the crop's root zone	[L]
LOW_EXTINC_ELEV	is the lowest elevation for root-groundwater uptake (ROOT_ELEV − capillary fringe)	[L]
DELT	is the time step length	[T]
DYEAR	is the date at the end of the time step as a decimal year	
DATE_START	is the calendar date at the start of the time step (yyyy-mm-ddThh:mm:ss)	

Figure 6.84. Land_Use block keyword PRINT ALL text file header explanation and binary record structure. *A*, Text-header explanation. *B*, Binary-record structure. [[T], unit of time in model units; [L^3/T], volumetric rate in model units; CHARACTER(20) indicates record is 20 characters long (20 bytes); CHARACTER(19) indicates record is 19 characters long (19 bytes); INTEGER is a 4-byte integer record; DOUBLE is a 8-byte floating-point number record.]

B

DATE_START	CHARACTER(19), starting date formatted as 'yyyy-mm-ddThh:mm:ss'
DYEAR	DOUBLE
DELT	DOUBLE
PER	INTEGER
STP	INTEGER
CROP	INTEGER
CROP_NAME	CHARACTER(20)
WBS	INTEGER
ROW	INTEGER
COL	INTEGER
AREA	DOUBLE
EFFICIENCY	DOUBLE
CU_INI	DOUBLE
CU	DOUBLE
CIR_INI	DOUBLE
CIR	DOUBLE
DEMAND_INI	DOUBLE
DEMAND	DOUBLE
ADDED_DEMAND_INI	DOUBLE
ADDED_DEMAND	DOUBLE
TOT_DEEP_PERC	DOUBLE
TOT_SURF_RUNOFF	DOUBLE
ADDED_DMD_DPERC	DOUBLE
ADDED_DMD_RUNOFF	DOUBLE
TRAN_POT	DOUBLE
ANOXIA_LOSS	DOUBLE
SOIL_STRESS_LOSS	DOUBLE
TRAN	DOUBLE
TRAN_SURF_INI	DOUBLE
TRAN_SURF	DOUBLE
TRAN_IRR	DOUBLE
TRAN_PRECIP	DOUBLE
TRAN_GW	DOUBLE
EVAP_IRR	DOUBLE
EVAP_PRECIP	DOUBLE
EVAP_GW	DOUBLE
PRECIPITATION	DOUBLE
GW_HEAD	DOUBLE
GSE	DOUBLE
ROOT_ELEV	DOUBLE
LOW_EXTINC_ELEV	DOUBLE

Figure 6.84. —Continued

PRINT ALL_VERBOSE Header

The keyword **PRINT ALL_VERBOSE** writes to the *Generic_Output_OptKey* file the crop and bare-soil information for every model cell—including crop input features. Bare soil is identified with a crop ID of `0` and crop name of `BARE_LAND`. The output for **PRINT ALL_VERBOSE** may be in text or binary format. The output may be text or binary format and has the header defined in figure 6.85.

A

PER	is the stress period number	
STP	is the time step number	
CROP	is the Land-use (Crop) ID number	
CROP_NAME	is the name of the specific crop	
WBS	is the Water Balance Subregion (Farm) ID number	
ROW	is the model row number	
COL	is the model column number	
LAY	is the upper most active model layer	
GW_INT	is root groundwater interaction level (0 – 5)	
IRR	is the irrigation flag and type	
AREA	is the area of land the crop occupies for the specified WBS	[L^2]
EFFICIENCY	is the irrigation efficiency (OFE)	
FRAC_TRAN	is the TRANSPIRATION_FRACTION (FTR)	
FRAC_EVAP_IRR	is the EVAPORATION_IRRIGATION_FRACTION (FEI)	
FRAC_SW_LOSS_PRE	is the SURFACEWATER_LOSS_FRACTION_PRECIPITATION	
FRAC_SW_LOSS_IRR	is the SURFACEWATER_LOSS_FRACTION_IRRIGATION	
CU_INI	is the initial-potential consumptive use of crop, which is the user-specified input CU plus $K_c \cdot ET_{ref}$	[L^3/T]
CU	is the actual crop consumptive use based on water supply	[L^3/T]
CIR_INI	is the crop irrigation requirement to meet CU_INI, assuming no irrigation losses	[L^3/T]
CIR	is the crop irrigation requirement, assuming no irrigation losses, to meet CU_INI after it has been reduced by anoxia and soil-stresses	[L^3/T]
DEMAND_INI	is the irrigation demand to meet CIR_INI considering irrigation losses	[L^3/T]
DEMAND	is the irrigation demand necessary to satisfy CIR (includes irrigation losses).	[L^3/T]
ADDED_DEMAND_INI	is the initial additional irrigation demand specified for the crop	[L^3/T]
ADDED_DEMAND	is the final additional irrigation demand that was satisfied with supply	[L^3/T]
TOT_DEEP_PERC	is the total loss of water to deep percolation (groundwater recharge)	[L^3/T]
TOT_SURF_RUNOFF	is the total loss of water to surface runoff	[L^3/T]
ADDED_DMD_DPERC	is the amount of ADDED_DEMAND irrigation that is lost as deep percolation	[L^3/T]
ADDED_DMD_RUNOFF	is the amount of ADDED_DEMAND irrigation that is lost as surface runoff	[L^3/T]
TRAN_POT	is the potential transpiration from the cropped area for the given CU_INI	[L^3/T]
ANOXIA_LOSS	is the reduction in TRAN_POT due to plant anoxia (high groundwater level)	[L^3/T]
SOIL_STRESS_LOSS	is the reduction in TRAN_POT due to non-idealized soil stress	[L^3/T]
⋮		

Figure 6.85. Land_Use block keyword PRINT ALL_VERBOSE text file header explanation and binary record structure. *A*, Text-header explanation. *B*, Binary-record structure. [[T], unit of time in model units; [L^3/T], volumetric rate in model units; CHARACTER(20) indicates record is 20 characters long (20 bytes); CHARACTER(19) indicates record is 19 characters long (19 bytes); INTEGER is a 4-byte integer record; DOUBLE is a 8-byte floating-point number record.]

***A** (continued)*

⋮		
TRAN	is the actual crop transpiration (TRAN_IRR + TRAN_PRECIP + TRAN_GW)	[L^3/T]
TRAN_SURF_INI	is the initial-potential transpiration from precipitation and irrigation water	[L^3/T]
TRAN_SURF	is actual transpiration from precipitation and irrigation water	[L^3/T]
TRAN_IRR	is the portion of transpiration (TRAN) from irrigation water supply	[L^3/T]
TRAN_PRECIP	is the portion of transpiration (TRAN) from precipitation	[L^3/T]
TRAN_GW	is the portion of transpiration (TRAN) from groundwater	[L^3/T]
EVAP_IRR	is the evaporation of water that originated from irrigation water supply	[L^3/T]
EVAP_PRECIP	is the evaporation of water that originated from precipitation	[L^3/T]
EVAP_GW	is the evaporation of water that originated from groundwater	[L^3/T]
PRECIPITATION	is the total precipitation	
GW_HEAD	is the groundwater elevation (head) beneath the crop	[L]
POND_DEPTH_TOL	is the the total depth of ponded water that crop can tolerate	[L]
PSI_1	is the root pressure that crop dies from anoxia occurs	[L]
PSI_2	is the upper bounding root pressure that is the optimal for groundwater-root	[L]
PSI_3	is the lower bounding root pressure that is the optimal for groundwater-root	[L]
PSI_4	is the wilting-root pressure	[L]
GSE	is the ground surface elevation	[L]
UP_EXTINC_ELEV	is the lowest groundwater elevation where crop death occurs as a result of anoxia	[L]
MID_EXTINC_ELEV	is the groundwater elevation where optimal uptake transitions to anoxia	[L]
ROOT_ELEV	is the lowest elevation of the crop's root zone	[L]
LOW_EXTINC_ELEV	is the lowest elevation for root-groundwater uptake (ROOT_ELEV – capillary fringe)	[L]
DELT	is the time step length	[T]
DYEAR	is the date at the end of the time step as a decimal year	
DATE_START	is the calendar date at the start of the time step (yyyy-mm-ddThh:mm:ss)	

Figure 6.85. —Continued

B

DATE_START	CHARACTER(19), starting date formatted as 'yyyy-mm-ddThh:mm:ss'
DYEAR	DOUBLE
DELT	DOUBLE
PER	INTEGER
STP	INTEGER
CROP	INTEGER
CROP_NAME	CHARACTER(20)
WBS	INTEGER
ROW	INTEGER
COL	INTEGER
LAY	INTEGER
GW_INT	INTEGER
IRR	INTEGER
AREA	DOUBLE
EFFICIENCY	DOUBLE
FRAC_TRAN	DOUBLE
FRAC_EVAP_IRR	DOUBLE
FRAC_SW_LOSS_PRE	DOUBLE
FRAC_SW_LOSS_IRR	DOUBLE
CU_INI	DOUBLE
CU	DOUBLE
CIR_INI	DOUBLE
CIR	DOUBLE
DEMAND_INI	DOUBLE
DEMAND	DOUBLE
ADDED_DEMAND_INI	DOUBLE
ADDED_DEMAND	DOUBLE
TOT_DEEP_PERC	DOUBLE
TOT_SURF_RUNOFF	DOUBLE
ADDED_DMD_DPERC	DOUBLE
ADDED_DMD_RUNOFF	DOUBLE
TRAN_POT	DOUBLE
ANOXIA_LOSS	DOUBLE
SOIL_STRESS_LOSS	DOUBLE
⋮	

⋮	
TRAN	DOUBLE
TRAN_SURF_INI	DOUBLE
TRAN_SURF	DOUBLE
TRAN_IRR	DOUBLE
TRAN_PRECIP	DOUBLE
TRAN_GW	DOUBLE
EVAP_IRR	DOUBLE
EVAP_PRECIP	DOUBLE
EVAP_GW	DOUBLE
PRECIPITATION	DOUBLE
GW_HEAD	DOUBLE
POND_DEPTH_TO	DOUBLE
PSI_1	DOUBLE
PSI_2	DOUBLE
PSI_3	DOUBLE
PSI_4	DOUBLE
GSE	DOUBLE
UP_EXTINC_ELEV	DOUBLE
MID_EXTINC_ELEV	DOUBLE
ROOT_ELEV	DOUBLE
LOW_EXTINC_ELEV	DOUBLE

Figure 6.85. —Continued

Supported SFAC DIMKEY

In figure 6.86, an X signifies by keyword each **SFAC** **DIMKEY** that is supported. Keywords not specified in the table do not support the **SFAC** **DIMKEY**s.

Keyword	DIMKEY		
	ByWBS	ByCrop	ByIrrigate
LAND_USE_AREA_FRACTION	X	X	
CONSUMPTIVE_USE	X	X	X
CROP_COEFFICIENT	X	X	X
TRANSPIRATION_FRACTION	X	X	X
EVAPORATION_IRRIGATION_FRACTION	X	X	X
SURFACEWATER_LOSS_FRACTION_PRECIPITATION	X	X	
SURFACEWATER_LOSS_FRACTION_IRRIGATION	X	X	X
ROOT_DEPTH	X	X	X
ROOT_PRESSURE		X	
POND_DEPTH	X	X	X
ADDED_DEMAND	X	X	X

Figure 6.86. LAND_USE block keywords that support **SFAC** **DIMKEY**. The supported words that can be used for **DIMKEY** are ByWBS, ByCrop, and ByIrrigate. [Supported keyword and **DIMKEY** combinations are marked with an X.]

IXJ Style Input Support

At the time of this report's publication, 10 of the keywords in the **LAND_USE** block allow for the use of the *IXJ Style* input that serves as an alternative input to the *Array Style* input. If the *IXJ Style* input is used, then the FMP expects to load records containing three integers (`I`) and one floating-point number (`X`) and does not read any subsequent integers (`J`). The three integers are the Crop ID, the model row, and model column. The floating-point number is the keyword property (`Prop`). For example, for the **LAND_USE_AREA_FRACTION**, the keyword property is the crop's fraction of cell area. The exception to the expected input records being as described applies to the **LOCATION** keyword, where the property is the crop's location in the model grid; it does not include a Prop because the property is fully defined by crop ID, model row, and column. The use of *IXJ Style* input is advantageous when the model input has included the **MULTIPLE_LAND_USE_PER_CELL** keyword and it is necessary to define an input using *Array Style*. This is because `IXJ` can replicate the *Array Style* input in a more compact manner that avoids an `NCROP×NROW` by `NCOL` array input. It is recommended to use *List Style* input whenever possible as the main input option. Figure 6.87 provides a list of the keywords that support *IXJ Style* input and their expected input.

Keyword List

Figure 6.88 presents the **LAND_USE** block with keywords that are relevant to most simulations; the basic definition of each keyword is included as comments.

Keywords that support IXJ Input Style	Expected Input
LOCATION	Crop Row Column
LAND_USE_AREA_FRACTION	Crop Row Column Prop
CONSUMPTIVE_USE	Crop Row Column Prop
CROP_COEFFICIENT	Crop Row Column Prop
TRANSPIRATION_FRACTION	Crop Row Column Prop
EVAPORATION_IRRIGATION_FRACTION	Crop Row Column Prop
SURFACEWATER_LOSS_FRACTION_PRECIPITATION	Crop Row Column Prop
SURFACEWATER_LOSS_FRACTION_IRRIGATION	Crop Row Column Prop
ROOT_DEPTH	Crop Row Column Prop
ADDED_DEMAND	Crop Row Column Prop

Figure 6.87. LAND_USE block keywords that support IXJ input as a surrogate for Array Style input. Expected Input uses the structure expected by IXJ. [Crop is the crop ID; Row is the model row; Column is the model column; and Prop is the associated property defined by the keyword.]

```
BEGIN LAND_USE
  # A simulation must include one of the two following keywords
  SINGLE_LAND_USE_PER_CELL        # Crop location defined by LOCATION
  MULTIPLE_LAND_USE_PER_CELL      # Crop location defined by LAND_USE_AREA_FRACTION
  #
  # Define the spatial location of each crop and fraction of cell area it occupies.
  LOCATION                 LAI[S, T, A]  # Do not specify  MULTIPLE_LAND_USE_PER_CELL
  LAND_USE_AREA_FRACTION   LAI[S, T, A]  # Optional if      SINGLE_LAND_USE_PER_CELL
  #
  PRINT ByWBS_ByCROP   Generic_Output_OptKey      # Recommended output option.
  #
  # At least one of the following two keywords should be defined
  CONSUMPTIVE_USE   LAI[S, T, A, L]   # Specify crop consumptive use directly [L/T]
  CROP_COEFFICIENT  LAI[S, T, A, L]   # Specify crop consumptive use as a crop coefficient
  #
  ROOT_DEPTH  LAI[S, T, A, L]   # Define the root depth of crop [L].
  #
  # Define which crops receive irrigation. If 0, then crop is not irrigated.
  # If greater than zero, then the integer indicates the irrigation type used for the crop.
  # If NIRRIGATE = 0, then keyword is not required.
  IRRIGATION  LAI[S, T, A, L]
  #
  TRANSPIRATION_FRACTION  LAI[S, T, A, L]   # Define the fraction of transpiration (Kcb/Kc).
  #
  # Define the fraction of cropped area that is irrigated but exposed to sunlight and evaporates.
  # This fraction can by specified with NCROP records (BY_CROP) or NIRRIGATE records (BY_IRRIGATE).
  # If NIRRIGATE = 0, then keyword is not required.
  # Only one of the following two keywords may be in use during a simulation.
  EVAPORATION_IRRIGATION_FRACTION BY_CROP      LAI[S, T, A, L]
  EVAPORATION_IRRIGATION_FRACTION BY_IRRIGATE  LAI[S, T, A, L]
  #
  # Define the fraction of unconsumed precipitation that becomes surface runoff.
  SURFACEWATER_LOSS_FRACTION_PRECIPITATION  LAI[S, T, A, L]
  #
  # Define the fraction of unconsumed precipitation that becomes surface runoff.
  # This fraction can by specified with NCROP records (BY_CROP) or NIRRIGATE records (BY_IRRIGATE).
  # If NIRRIGATE = 0, then keyword is not required.
  # Only one of the following two keywords may be in use during a simulation.
  SURFACEWATER_LOSS_FRACTION_IRRIGATION BY_CROP      LAI[S, T, A, L]
  SURFACEWATER_LOSS_FRACTION_IRRIGATION BY_IRRIGATE  LAI[S, T, A, L]
  #
END LAND_USE
```

Figure 6.88. LAND_USE block with most of the supported keywords and their input format. [NIRRIGATE, number of irrigation types; NCROP, number of land use types; [L/T], length per time in model units; [L^3/T], volumetric rate in model units; 1, list-array input with the following letters representing the supported keywords: S for STATIC, T for TRANSIENT, A for ARRAY, and L for LIST]

Salinity Flush Irrigation Block

The **SALINITY_FLUSH_IRRIGATION** block optionally allows users to simulate additional irrigation demand for flushing soils to prevent salt accumulation. Note that the block name can be declared as **SALINITY FLUSH IRRIGATION** or **SALINITY_FLUSH_IRRIGATION**, but the later version, with underscores, is recommended for clarity. Additional irrigation water can be set to a specific value, use a predefined salinity-flush irrigation function called the Rhoades equation, or use a user supplied custom function. Typically, the quantity of additional applied water is dependent upon the salt tolerance of the specific crop and the salinity of the source water. This requires that the input specify a soil salinity (EC_e) threshold for each crop, which is the point when a crop's yield is reduced as a result of soil salinity.

The soil salinity is measured using average electrical conductivity (EC) of a soil sample with units of decisiemens per meter (dS/m). Electrical conductivity represents, within the range of 0.1–5 dS/m, 640 milligrams per liter (mg/L) of total dissolved solids (TDS) per 1 dS/m. For measurements greater than 5 dS/m, EC represents approximately 800 mg/L of TDS per 1 dS/m. Figure 6.89 presents example EC_e values for select crops in dS/m; soil salinities with higher EC result in less than optimal yield of the crop. The salinity of the sources of irrigation water increases the salinity of the soil. The FMP has four potential sources of irrigation water: **NON_ROUTED_DELIVERY** (NRD) from the **SURFACE_WATER** block, **SEMI_ROUTED_DELIVERY** (SRD) from the **SURFACE_WATER** block, groundwater pumpage from the **SUPPLY_WELL** block, and external water (EXT) sources that make up a supply shortfall when the **DEFICIENCY_SCENARIO** is set to `0` (Zero-Scenario deficit irrigation) from the **WATER_BALANCE_SUBREGION** block. This requires specifying the EC, in dS/m, for the four sources of water.

Crop Common Name	Soil Salinity Threshold, EC_e (dS/m)
Alfalfa	2.0
Almond	1.5
Barley	8.0
Broccoli	2.8
Cabbage	1.8
Carrot	1.0
Celery	1.8
Corn	1.7
Garlic	3.9
Lemon	1.5
Lettuce	1.3
Peach	1.7
Potato	1.7
Spinach	2.0
Strawberry	1.0
Tomato	2.5
Wheat, durum	5.9

Figure 6.89. List of common agricultural crops and their soil salinity (EC_e) threshold (Tanji and Kielen, 2002). [The threshold value represents the point when the potential crop yield starts to decrease because of soil salinity. Abbreviation: dS/m, decisiemens per meter]

The Rhoades equation may be used for determining the additional irrigation demand and requires solving two sets of equations. The first equation involves determining the fraction of total irrigation (applied) water that must pass through the soil to prevent the soil salinity from reaching the tolerance of the crop. This unitless fraction is called the Leaching Requirement (LR). The LR is determined from the salinity concentration of irrigation water (EC_w) and from the crop tolerance to soil salinity. Both EC_w and EC_e are measured as electrical conductivity (dS/m). From Ayers and Wescott (1985), the leaching requirement can be calculated as follows:

$$LR = \frac{EC_w}{(5 \cdot EC_e) - EC_w} \tag{6.1}$$

where

LR is the minimum leaching requirement needed to control salts, with $0 \leq LR < 1$ (–);

EC_w is the salinity of the applied irrigation water, calculated from the mixture of irrigation water used by the FMP to meet the irrigation demand (dS/m); and

EC_e is the average soil salinity (dS/m) tolerated by the crop as measured on a soil saturation extract. It can be viewed as the desired soil salinity after additional irrigation is applied.

The leaching requirement cannot exceed nor equal one, so salinity flushing is possibly only when $EC_w < 5 \times EC_e$. The choice of EC_e is based on the desired, or obtainable, relative yields of the crop. For example, figure 6.89 represents EC_e values that calculate a leaching requirement that could obtain a 100-percent relative yield. Once the leaching requirement is determined, the second part of the Rhoades equation uses the crop irrigation requirement—the necessary irrigation to satisfy a crop's evapotranspiration demand—and specifies the total irrigation necessary for salinity flushing:

$$AW = \frac{CIR}{1 - LR} \tag{6.2}$$

where

CIR is the crop irrigation requirement under perfect irrigation efficiency (L^3/T), and

AW is the necessary applied water for salinity flushing under perfect irrigation efficiency (L^3/T).

$$D_{irrigation} = AW/OFE \tag{6.3}$$

where

$D_{irrigation}$ is the irrigation necessary to satisfy the crop's transpiration and sufficiently provide salinity flushing under actual irrigation efficiency (L^3/T).

Figure 6.89 presents a list of the soil salinity threshold for common agricultural crops.

How uniformly, in the vertical direction, irrigation is applied can affect the salt accumulation and flushing of salts from the soil. The second part of the Rhoades equation can be modified to include the effects of irrigation uniformity (*DU*).

$$AW = \frac{CIR/DU}{1 - LR} \tag{6.4}$$

where

DU is the irrigation uniformity—how evenly the irrigation method is spread across the root zone—as a fraction from 0 to 1 (–). A value of 1 indicates that the irrigation is perfectly uniform through the root zone.

DU can be set to 1 to not include this uniformity consideration as part of the simulation, but *DU* should never be set to zero.

Keyword List

Figure 6.90 summarizes the keywords that the **SALINITY_FLUSH_IRRIGATION** block supports. They are arranged in order of importance, with required keywords first and optional keywords last.

A

Keyword Input	Description
CROP_SALINITY_TOLERANCE LAI[S, T, L]	Required input that specifies each Crop's salinity tolerance (EC_e) as an Electrical Conductivity (EC) for the desired yield of the crop. *List Style* input expects **NCROP** records that read the EC_e as a **FLOAT**. Input must be in deciSiemens per meter (dS/m).
CROP_LEACHING_REQUIREMENT LAI[S, T, L]	Required input that specifies each Crop's leaching requirement (LR) as either a **FLOAT**, Rhoades equation, or a custom expression. *List Style* input expects **NCROP** records that specify for each crop either: the LR as a fraction (**FLOAT**) between 0 to 1, or the word **RHOADES** to use the Rhoades equation, or a custom expression whose result is the LR, or the word **SKIP** to set LR to 0.
CROP_SALINITY_APPLIED_WATER LAI[S, T, L]	Required input that defines for each Crop the additional irrigation (AW) for salinity flushing [L^3/T]. This is the amount of water added in addition to the irrigation requirement to ensure salinity flushing. AW may be specified as **FLOAT** value, calculated with Rhoades equation, which is a function of LR, or determined by a user-specified custom expression. *List Style* input expects **NCROP** records that specify for each crop either: the AW directly, or the word **RHOADES** to use the Rhoades equation, or a custom expression whose result is the AW, or the word **SKIP** to set LR to 0. Note that Rhoades equation is dependent on the leaching ratio (LR) calculated by the **CROP_LEACHING_REQUIREMENT** keyword

Figure 6.90. Salinity_Flush_Irrigation block keywords. [The number of records read for List Style input depends on the keyword. Keywords that begin with WBS (water-balance subregion) expect NWBS records, and keywords that begin with CROP expect NCROP records. Note that some of the input is NOT in model units, but instead in parts per million (PPM) or decisiemens per meter (dS/m). Some of the keyword and input are specified over two lines because of their length, but should be specified on the same line in FMP. Abbreviations: [FLOAT, floating-point number; INT, integer; ID, water balance subregion identification number; WBS, water-balance subregion; [L], length in model units; [L^3/T], volumetric rate in model units; *Generic_Output_OptKey* specifies the path and file name for an output file; LAI, list-array input with the following letters representing the supported keywords: S for STATIC, T for TRANSIENT, and L for LIST]

B

Keyword Input	Description
WBS_SUPPLY_SALT_CONCENTRATION LAI[S, T, L-4]	Required input that specifies the source water supply's salt concentration in PPM (milligram/liter). This is not in model units. *List Style* reads **NWBS** records that specify the salt concertation as a **FLOAT** from four sources of water: Non-Routed Delivery (NRD), Semi-routed Delivery (SRD), Groundwater Pumping, and External Sources (FMP Zero-Scenario water). Supports use of **SFAC DIMKEY** with **BySource** option, which reads four scale that are applied to each of the water sources.
WBS_IRRIGATION_UNIFORMITY LAI[S, T, L-**NIRRIGATE**]	Optional input that specifies for each WBS the vertical, across the root zone, irrigation uniformity (DU). The fraction is specified in the range 0 < DU ≤ 1. A value of 1 indicates that the irrigation is perfectly uniform and a value of 0.5 indicates that the irrigation is half uniform. DU may not be set to 0. *List Style* input expects **NWBS** records that read the DU for each irrigation type (**NIRRIGATE**) as a **FLOAT**. If not specified, then DU is set to 1.0 for all irrigation types.
CROP_HAS_SALINITY_DEMAND LAI[S, T, L]	Optional input that specifies which crops should have additional irrigation for salinity flushing. *List Style* reads **NCROP** **INT** flags to indicate if the crop includes additional irrigation for salinity flushing: 0 to not have salinity flush irrigation 1 to have salinity flush irrigation If not specified, then all irrigated crops (**IRRIGATION** > 0) will have additional irrigation for salinity flushing.

Figure 6.90. —Continued

C

Keyword	Input	Description
CROP_MAX_LEACHING_REQUIREMENT	LAI[S, T, L	Optional input that specifies each crop's max allowed leaching requirement (LR_{Max}). The leaching requirement (LR) is specified or calculated with the keyword **CROP_LEACHING_REQUIREMENT** and must be less than 1. If $LR > LR_{Max}$, then $LR = LR_{Max}$. *List Style* input expects **NCROP** records that read LR_{Max} as a **FLOAT** between 0 and 0.9999. If keyword is not specified, then the the default is LR_{Max} = 0.99 for all crops.
EXPRESSION_LINE_LENGTH	**INT**	Optional **INT** input that specifies the maximum length of a custom expression that may be read by **CROP_LEACHING_REQUIREMENT** and **CROP_SALINITY_APPLIED_WATER**. If not specified, then the default is 20 characters long. That is the largest expression can be "12345678901234567890", which includes white spaces.
EXPRESSION_VARIABLE_NEARZERO	**FLOAT**	Optional **FLOAT** input that specifies the tolerance for converting a custom expression variable to zero. For example, if the tolerance is set to 0.1 and P is a variable in the expression, then its value is set to zero if $-0.1 \leq P \leq 0.1$. If keyword is not specified, then the default tolerance is 0.0.

Figure 6.90. —Continued

D

Keyword and Input		Description
PRINT ByWBS_ByCROP	*Generic_Output_OptKey*	Writes salinity irrigation flush output that is summarized for each crop within each WBS.
PRINT ByWBS	*Generic_Output_OptKey*	Writes salinity irrigation flush output that is summarized for each WBS.
PRINT ByCROP	*Generic_Output_OptKey*	Writes salinity irrigation flush output that is summarized for each crop across the entire model domain.
PRINT ALL	*Generic_Output_OptKey*	Writes salinity irrigation flush output and simulated results for each crop for every model cell that contains the crop with salinity flush irrigation.
PRINT INPUT	*Generic_Output_OptKey*	Writes the input that is loaded.

Figure 6.90. —Continued

The keywords **CROP_LEACHING_REQUIREMENT** and **CROP_SALINITY_APPLIED_WATER** allow defining custom expression for calculating the leaching requirement and additional applied irrigation water. A custom expression is passed to the ExpressionParser and accepts a set of predefined variable names whose values are set at runtime. The list of supported variable names available, at the time of this report's publication, are presented in figure 6.91.

Custom expressions may contain an *Inline IF* statement. The basic syntax is IF[Condition, TrueResult, FalseResult], for example, "`IF[ ETc>ETr, (ECe/ECw)*LOG(CIR) + CIR, 0.1·CIR ]`". This example checks if the crop evapotranspiration from precipitation and irrigation is greater than the reference evapotranspiration. If it is true, then it returns the result of a custom expression, `(ECe/ECw)·LOG(CIR) + CIR,` and, if false, returns one-tenth the crop irrigation requirement (`0.1·CIR`).

Figure 6.92 is an example of **SALINITY_FLUSH_IRRIGATION** block input for three crops, two WBSs, and one irrigation type (note the irrigation type is not used in the input example). This example is for demonstration purposes only. Generally, users can specify **CROP_LEACHING_REQUIREMENT** and **CROP_SALINITY_APPLIED_WATER** with either a single **FLOAT** or use the Rhoades equation.

A

Variable Name	Description—Value Stored in Variable Name
LR	Leaching requirement [-] Value is set by **CROP_LEACHING_REQUIREMENT**.
ECw	Calculated electrical conductivity of irrigation water (dS/m).
ECe	Crop electrical conductivity tolerance (dS/m).
CU	Potential consumptive use of crop (total potential consumption of water); [L^3/T].
ETr	Reference evapotranspiration; [L^3/T].
ETc	Crop evapotranspiration from precipitation and irrigation; [L^3/T]. Assumes that irrigation needs are fully met.
ETp	Crop evapotranspiration from precipitation; [L^3/T].
ETi	Crop evapotranspiration from irrigation. Determined as DMD*EFF; [L^3/T].
CIR	Crop irrigation requirement that assumes perfect irrigation efficiency; [L^3/T]. Note, CIR = ETi, if efficiency improvement is disabled (**EFFICIENCY_IMPROVEMENT**= 0).
DMD	Crop irrigation demand that includes irrigation efficiency. This irrigation is necessary to meet the crop's consumptive use. Note, DMD = CIR if irrigation efficiency is 1.0; [L^3/T].
P	Precipitation that falls on the crop; [L^3/T]
AREA	Crop area; [L^2].
Tgw	Transpiration from root-groundwater uptake; [L^3/T].
Tp	Transpiration from precipitation; [L^3/T].
Ti	Transpiration from irrigation; [L^3/T].

Figure 6.91. SALINITY_FLUSH_IRRIGATION block custom expression variable name support for calculating CROP_LEACHING_REQUIREMENT and CROP_SALINITY_APPLIED_WATER. [The variable names listed are replaced at runtime with their described values. Abbreviations: dS/m, electrical conductivity in decisiemens per meter; [L], length in model units; [L^2], area in model units; [L^3/T], volumetric rate in model units; [-], indicates that property is unitless]

B

Variable Name	Description—Value Stored in Variable Name
EFF	Irrigation efficiency (OFE, appendix 4); [-].
ROOT	Crop root depth; [L].
CapF	Capillary fringe length; [L].
DP_p	Deep percolation from precipitation; [L^3/T].
DP_i	Deep percolation from irrigation, if irrigated demand is met; [L^3/T]. This includes other additional irrigation demands, such as **ADDED_CROP_DEMAND**, but not deep percolation from salinity irrigation.
DP	Total potential deep percolation from precipitation and irrigation; [L^3/T].
ADRS	Crop added demand runoff split (**ADDED_DEMAND_RUNOFF_SPLIT**).
ADMD	Sum of added demands, excluding salinity added demand, for the crop; [L^3/T].

Figure 6.91. —Continued

```
#  Example Salinity Flush Irrigation Block input
#
#  Assume GLOBAL Dimension block defines: NWBS = 2 and NCROP = 3
#
BEGIN SALINITY_FLUSH_IRRIGATION
  #
  #  Define the salinity concentration of the irrigation water
  WBS_SUPPLY_SALT_CONCENTRATION STATIC LIST INTERNAL
  #WBS  SW    GW    NRD   External
   1    310.  210.  520.  640.   # (mg/L)
   2    300.  200.  510.  640.
  #
  #  Define the soil salinity acceptable tolerance for the crops. Using values for 100% yield.
  CROP_SALINITY_TOLERANCE STATIC LIST INTERNAL
  #Crop  ECe (dS/m)
   1    1.7   # Corn
   2    8.0   # Barley
   3    1.0   # Strawberry
  #
  #  Define how the leaching requirement is calculated.
  CROP_LEACHING_REQUIREMENT STATIC LIST INTERNAL
  #Crop  Expression
   1    0.2                 # Corn – Define directly a leaching requirement of 0.2
   2    RHOADES             # Barley – Use Rhoades equation for the leaching requirement
   3    ECw/(5*ECe - ECw)   # Strawberry – Custom expression that is identical to Rhoades.
  #
  # Define how the additional irrigation for salinity flushing is calculated.
  # The variable LR and Rhoades equation use
  # the leaching requirement set by CROP_LEACHING_REQUIREMENT
  # Note that the result is only the additional irrigation, not the total irrigation.
  CROP_SALINITY_APPLIED_WATER STATIC LIST INTERNAL
  #Crop  Expression
   1    SKIP                # Corn – Does not have additional irrigation (no salinity flush).
   2    RHOADES             # Barley – Use Rhoades equation for additional irrigation
   3    CIR/(1-LR) - CIR    # Strawberry – Custom expression that is identical to Rhoades.
  #
END SALINITY_FLUSH_IRRIGATION
```

Figure 6.92. SALINITY_FLUSH_IRRIGATION block input example. [Example assumes that the GOBAL DIMENSION block set the number of water balance subregions (NBS) to 2 and the number of crops (NCROP) to 3]

PRINT Headers

The keyword **PRINT ByWBS_ByCrop** writes to the *Generic_Output_OptKey* file the summarized salinity irrigation flush results for each WBS for every time. The keyword **PRINT ByWBS** has the same headers, except it does not include the CROP header because it aggregates all crops together by WBS. The keyword **PRINT ByCrop** has the same headers, except it does not include the WBS header because it aggregates information across all WBSs for each crop. The keyword **PRINT ByCrop** has the same headers, except it is written for every model cell and contains the header ROW and COL, for the model row and column, after the header FARM. The output for **PRINT ByWBS_ByCrop** may be in text or binary format and has the header defined in figure 6.93.

A

PER	is the stress period number	
STP	is the time step number	
WBS	is the Water Balance Subregion (Farm) ID number	
CROP	is the Land-use (Crop) ID number	
CROP_NAME	is the name of the specific crop	
DEMAND%CHANGE	is the percentage increase in irrigation for salinity flushing	[percent]
CROP_AREA	is the area of land the crop occupies	[L^2]
IRRIGATED_AREA	is the area of land the crop occupies that is irrigated	[L^2]
SALINITY_AREA	is the area of irrigated land that has salinity flushing	[L^2]
PRECIPITATION	is the total precipitation	[L^3/T]
TOT_IRRIGATION	is the total irrigation applied to the crop	[L^3/T]
TOT_DEEP_PERC	is the total loss of water to deep percolation (groundwater recharge)	[L^3/T]
LEACH_FRACTION	is the leaching fraction (LF)	
CU	is actual crop consumptive use based on water supply	[L^3/T]
ET_IRR	is evapotranspiration of water that originated from irrigation water supply	[L^3/T]
SALT_REQ_IRR	is the necessary, perfect efficiency, irrigation for the salinity flushing requirement	[L^3/T]
SALT_IRR	is the final irrigation with perfect efficiency applied for salinity flushing	[L^3/T]
SALT_REQ_DEMAND	is the necessary irrigation demand for the salinity flushing requirement	[L^3/T]
SALT_DEMAND	is the final irrigation applied for salinity flushing	[L^3/T]
LEACH_REQ	is the calculated leaching requirement	[L^3/T]
ECe	is the crop electrical conductivity tolerance	[dS/m]
ECw	is the irrigation water's electrical conductivity	[dS/m]
IRR_UNIFORMITY	is the irrigation uniformity of the all the irrigation applied	[L^3/T]
DELT	is the time step length	[T]
DYEAR	is the date at the end of the time step as a decimal year	
DATE_START	is the calendar date at the start of the time step (yyyy-mm-ddThh:mm:ss)	

Figure 6.93. SALINITY_FLUSH_IRRIGATION block keyword PRINT ByWBS_ByCrop text file header explanation and binary record structure. *A*, Text-header explanation. *B*, Binary-record structure. [[T], unit of time in model units; [L^3/T], volumetric rate in model units; [dS/m], decisiemens per meter; CHARACTER(20) indicates record is 20 characters long (20 bytes); CHARACTER(19) indicates record is 19 characters long (19 bytes); INTEGER is a 4-byte integer record; DOUBLE is a 8-byte floating-point number record.]

B

```
DATE_START          CHARACTER(19), starting date formatted as 'yyyy-mm-ddThh:mm:ss'
DYEAR               DOUBLE
DELT                DOUBLE
PER                 INTEGER
STP                 INTEGER
WBS                 INTEGER
CROP                INTEGER
CROP_NAME           CHARACTER(20)
DEMAND%CHANGE       DOUBLE
CROP_AREA           DOUBLE
IRRIGATED_AREA      DOUBLE
SALINITY_AREA       DOUBLE
PRECIPITATION       DOUBLE
TOT_IRRIGATION      DOUBLE
TOT_DEEP_PERC       DOUBLE
LEACH_FRACTION      DOUBLE
CU                  DOUBLE
ET_IRR              DOUBLE
SALT_REQ_IRR        DOUBLE
SALT_IRR            DOUBLE
SALT_REQ_DEMAND     DOUBLE
SALT_DEMAND         DOUBLE
LEACH_REQ           DOUBLE
ECe                 DOUBLE
ECw                 DOUBLE
IRR_UNIFORMITY      DOUBLE
```

Figure 6.93. —Continued

Data Requirements

The data requirements vary depending on the level of complexity and detail desired for the FMP4 simulation. To keep this section simple, it is broken into a set of input questions and responses that indicate which keyword needs to be set for the simulation to incorporate it. There is some redundancy in answers, but that is to include the widest range of questions possible. The first planning decision concerns the number of WBSs that are simulated and their spatial layout—**WATER_BALANCE_SUBREGION** block **LOCATION**. The next decision is the total number of land uses simulated, their spatial location, and if there is more than one land use in a model cell—**LAND_USE** block **SINGLE_LAND_USE_PER_CELL**, **MULTIPLE_LAND_USE_PER_CELL**, **LAND_USE_AREA_FRACTION**, and **LOCATION**. The WBS and crops can vary with time, but the best practice is to estimate the maximum number in use at one time—**GLOBAL DIMENSION** block **NWBS** and **NCROP**. Figure 6.94 is a flow chart that works through basic considerations of the potentially necessary datasets for an FMP simulation.

Example Farm Process (FMP) Input File

There are many input options in the latest version of the FMP. Figure 6.95 provides an example showing the minimum input required for an FMP simulation and some examples of internal, list, and array input options. Items in green are the named input blocks. Items in blue are keywords. Items in red indicate how the input is read. Items in black are input values such as values for each grid cell or for named output files. Lines with "#" symbols are comment lines. The items in this section are not discussed at length as each of the input types and options is described in detail in the respective report section corresponding to the block name.

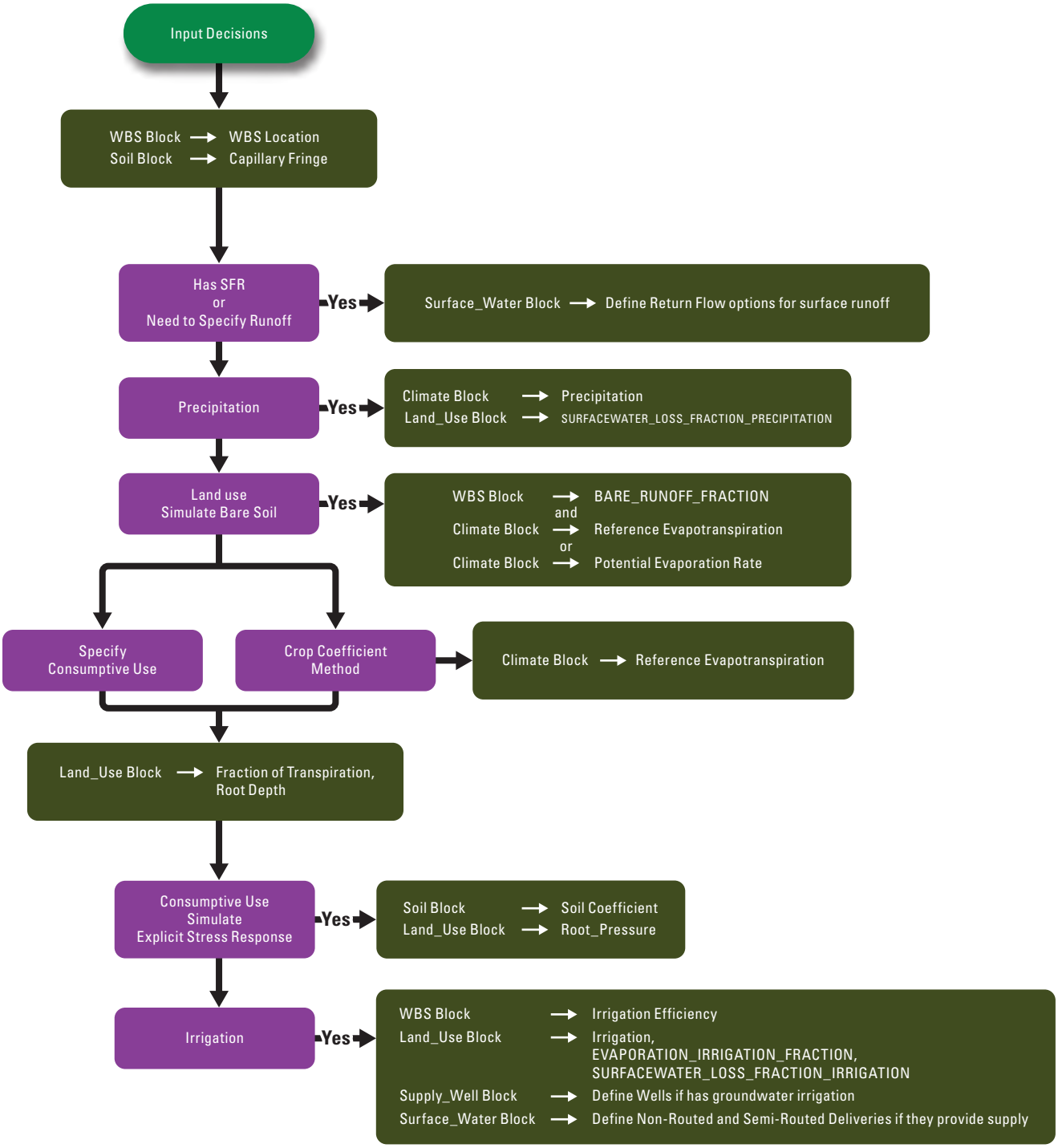


Figure 6.94. Work flow for determining what input is necessary.

A

```
BEGIN GLOBAL DIMENSION          # Model grid is 3 by 4, such that: NROW = 3, NCOL =4
   #
   NFARM      3
   NCROP      3
   NSOIL      2
   NIRRIGATE 2
   NRD_TYPES 1
   #
   NSFR_DELIV   2
   NSFR_RETURN  1
   #
   SURFACE_ELEVATION INTERNAL
      100.   101.  102.  103.
      100.5  101.  102.  103.
      100.7  101.  102.  103.
END
#
BEGIN WATER_BALANCE_SUBREGION    #WBS
   #
   LOCATION  STATIC ARRAY INTERNAL
    1  1  2  2
    1  1  2  2
    3  3  2  2
   #
   EFFICIENCY STATIC LIST INTERNAL
   1   0.7  0.9 #[WBS,  Irrigation Type 1,  Irrigation Type 2]
   2   0.7  0.9
   3   0.7  0.9
   #
END WATER_BALANCE_SUBREGION
#
BEGIN OUTPUT
   #
   FARM_WELL_CBC           0
   FARM_NET_RECHARGE_CBC   0
   #
   FARM_DEMAND_SUPPLY_SUMMARY ./FDS.OUT
   FARM_BUDGET                ./FB_DETAILS.OUT
END
#
```

Figure 6.95. Example FMP input file. *A*, Global Dimension, Water Balance Subregion, and Output block input example. *B*, Soil, Climate, and Surface_Water block input example. *C*, Supply_Well block input example. *D*, Land_Use block input example. [This figure is split into multiple parts, but the actual input is one single file.]

B

```
BEGIN SOIL
   #
   SOIL_ID  INTERNAL
    1 1 2 2
    1 1 2 2
    1 1 2 2
   #
   CAPILLARY_FRINGE STATIC LIST INTERNAL
   1  1.00  # [Soil_ID,  Capillary_Fringe]
   2  0.75
   #
END SOIL
#
BEGIN CLIMATE
   #
   PRECIPITATION  CONSTANT 0.1
   #
   REFERENCE_ET  STATIC ARRAY INTERNAL # Only required if using Crop Coefficient
      1.1  1.2  1.3  1.4
      1.1  1.2  1.3  1.4
      1.1  1.2  1.3  1.4
END CLIMATE
#
BEGIN SURFACE_WATER
   #
   NON_ROUTED_DELIVERY STATIC LIST INTERNAL
   1  100.  1  0  # [WBS,  NRD_VOL,  NRD_RANK,  NRD_USE]
   2    0.  1  0
   3    0.  1  0
   #
   SEMI_ROUTED_DELIVERY STATIC LIST INTERNAL
   1  1  15 1 # [ISRD,  WBS,  Segment,  Reach]
   2  2  22 1
   #
   SEMI_ROUTED_RETURN STATIC LIST INTERNAL
   1  1  15 1 # [ISRR,  WBS,  SEGMENT,  REACH]
   #
   ROUTED_RETURN_ANY_NON_DIVERSION_REACH # Fully-routed return flow keyword
END
#
```

Figure 6.95. —Continued

C

```
BEGIN SUPPLY_WELL
   #
   QMAXRESET
   #
   MNW_AUTOMATIC_ON
   MNW_AUTOMATIC_OFF
   # Output Options
   PRINT BYWELL       ./Farm_Well_Info.txt
   PRINT BYFARM       ./Farm_Well_ByFarm_Info.txt
   #
   TIME FRAME
   # Well_Name      WBS      LAY    ROW COL  QMAX    Date_Start    Date_End
     Well1          1        2      1    2   1000.   1             NPER
     Well2          1        2      2    2   1000.   5             10
     WEL_MNW_LNK    2       MNW     2    3   2000.   1             NPER
   #
END SUPPLY_WELL
#
```

Figure 6.95. —Continued

D

```
BEGIN LAND_USE
  #
  PRINT BYFARM_BYCROP ./Crop_Info.txt
  PRINT BYFARM        ./Crop_Info_ByFarm.txt
  #
  LOCATION   STATIC ARRAY INTERNAL
   2  1  1  2
   2  1  1  2
   3  3  2  2
  #
  CROP_COEFFICIENT STATIC LIST INTERNAL
  1  0.60  # [Crop_ID,  Crop_Coefficient]
  2  1.10
  3  0.15
  #
  ROOT_DEPTH STATIC LIST INTERNAL
  1  0.5  # [Crop_ID,  Root_Depth]
  2  1.6
  3  0.1
  #
  IRRIGATION STATIC LIST INTERNAL
  1  2  # [Crop_ID,  Irrigation Type]
  2  1  # Uses user-specified irrigation type 1
  3  0  # Not irrigated
  #
  TRANSPIRATION_FRACTION STATIC LIST INTERNAL
  1   0.6  # [Crop_ID,  FTR]
  2   0.8
  3   0.4
  #
  EVAPORATION_IRRIGATION_FRACTION By_Crop  STATIC LIST INTERNAL
  1  0.15  # [Crop_ID,  FEI]
  2  0.15
  3  0.00  # Never used because it is not irrigated
  #
  SURFACEWATER_LOSS_FRACTION_PRECIPITATION  STATIC LIST INTERNAL
  1  0.2  # [Crop_ID,  FIESWP]
  2  0.1
  3  0.9
  #
  SURFACEWATER_LOSS_FRACTION_IRRIGATION By_Crop STATIC LIST INTERNAL
  1  0.01  # [Crop_ID,  FIESWI]
  2  0.05
  3  0.00
  #
END LAND_USE
```

Figure 6.95. —Continued

References Cited

Allen, R.G., Pereira, L.S., Raes, D., and Smith, M., 1998, Crop evapotranspiration—Guidelines for computing crop water requirements: Rome, Italy, Food and Agriculture Organization of the United Nations, Irrigation and Drainage Paper 56, 300 p., http://www.fao.org/docrep/X0490E/X0490E00.htm.

Ayers, R.S., and Wescott, D.W., 1985, Water quality for agriculture: Rome, Italy, Food and Agriculture Organization of the United Nations Irrigation and Drainage Paper 29, rev. 1, variously paginated, http://www.fao.org/docrep/003/t0234e/t0234e00.htm.

Brouwer, C., and Heibloem, M., Irrigation water management—Irrigation water needs, training manual no. 3: Rome, Italy, Food and Agriculture Organization (FAO), 102 p., http://www.fao.org/3/S2022E/s2022e00.htm.

Dastane, N.G., 1978, Effective rainfall in irrigated agriculture: Rome, Italy, FAO irrigation and Drainage paper, 65 p., http://www.fao.org/3/x5560e/x5560e00.htm.

Hanson, R.T., Boyce, S.E., Schmid, W., Hughes, J.D., Mehl, S.M., Leake, S.A., Maddock, T., III, and Niswonger, R.G., 2014, One-Water Hydrologic Flow Model (MODFLOW-OWHM): U.S. Geological Survey Techniques and Methods 6–A51, 120 p., http://dx.doi.org/10.3133/tm6A51.

Konikow, L.F., Hornberger, G.Z., Halford, K.J., and Hanson, K.J., 2009, Revised multi-node well (MNW2) package for MODFLOW ground-water flow model: U.S. Geological Survey Techniques and Methods 6–A30, 67 p., https://pubs.usgs.gov/tm/tm6a30/.

Schmid, W., Hanson, R.T., Maddock, T., III, and Leake, S.A., 2006, User guide for the farm process (FMP1) for the U.S. Geological Survey's modular three-dimensional finite-difference ground-water flow model, MODFLOW-2000: U.S. Geological Survey Techniques and Methods 6–A17, 127 p., https://water.usgs.gov/nrp/gwsoftware/mf2005_fmp/tm6A17.pdf.

Schmid, W., King, J.P., and Maddock, T.M., III, 2009, Conjunctive surface-water/ground-water model in the southern Rincon Valley using MODFLOW-2005 with the farm process: Las Cruces, N. Mex., New Mexico Water Resources Research Institute Completion Report no. 350, 65 p., https://nmwrri.nmsu.edu/tr350/.

State of California, 2014, Sustainable Groundwater Management Act [and related statutory provisions from SB1168 (Pavley), AB1739 (Dickinson), and SB1319 (Pavley) as chaptered]: State of California, 52 p., http://leginfo.legislature.ca.gov/faces/codes_displayText.xhtml?lawCode=WAT&division=6.&title=&part=2.74.&chapter=2.&article=.

Tanji, K.K., and Kielen, N.C., 2002, Agricultural drainage water management in arid and semi-arid areas: Rome, Italy, United Nations Food and Agriculture Organization, Irrigation and Draining Paper 61, variously paginated, http://www.fao.org/docrep/005/y4263e/y4263e00.htm.

Appendix 7. Conduit Flow Process Updates and Upgrades (CFP2)

The conduit flow process allows for the coupled simulation of pipes or other preferential flow paths within an aquifer (that is, a discrete-continuum model). The aquifer volume is represented by a porous medium (that is, matrix) and preferential flow paths (for example, conduits or mine adits). Preferential flow paths are simulated as linear pathways in the aquifer. In this version of conduit flow process (CFP2), several modifications and increased capabilities were developed for conduit boundary conditions, time-series analysis (TSA) and output, conduit storage, and length-dependent exchange. The newly added boundary conditions and conduit storage features available in CFP2 include the following:

- Time series as input files (for example, for boundaries—important because of highly conductive pipes).
- Additional mixed boundary conditions (Cauchy, combined Dirichlet-Neumann).
- Additional direct storage for discrete pipe structures simulated as Conduit Accessible Drainable Storage (CADS).
- Water release by dewatering discrete elements through partially filled pipe storage.

This appendix summarizes the conceptual framework of each of these new features in the second version of the Conduit Flow Process (CFP2). Instructions to implement the new features are presented in appendix 8.

Time-Dependent Boundary Conditions

Flow through karst conduits can respond to changed boundary conditions very quickly and sensitively because of the high conductivity. Consequently, transient models can require time-dependent (TD) boundary conditions, such as pumping rates (well boundary conditions) or river stages (Cauchy boundary conditions). As originally implemented in CFP1 (Shoemaker and others, 2008), time-dependent input was applied by dividing the model time step into several periods. For karst systems, however, this can result in inefficient input files for a number of such periods. CFP2 Mode 1 was modified to read time-dependent data from an external file and to compute boundary conditions for each model time step to account for time-dependent boundary conditions in one model stress period. This functionality is useful for various boundary conditions:

- Fixed head boundary
- Fixed head limited flow boundary (FHLQ)
- Well boundary (QWELL)
- Cauchy boundary

The approach to linear interpolation of temporally varying boundary conditions across model time-steps is as follows:

$$x_t = \frac{t - t_p}{t_n - t_p}\left(x_n - x_p\right) + x_p \tag{7.1}$$

where

x_t is the interpolated boundary condition value at time, t;
t_p is the time of previous data entry;
t_n is the time of next data entry;
x_p is the previous data entry boundary condition value; and
x_n is the next data entry boundary condition value.

Measured data from a large-scale pumping test (Maréchal and others, 2008), introduced in the "Conduit-Associated Drainable Storage" section, are used to demonstrate the functionality enabled by TD boundary conditions. Hence, measured pumping rates are treated as time-dependent data having a resolution of 3,600 seconds. Results are presented in figure 7.1 and show that (1) variable pumping rates are correctly considered by the numerical model using TD data and (2) relatively small variability in pumping can cause large variation in well drawdown.

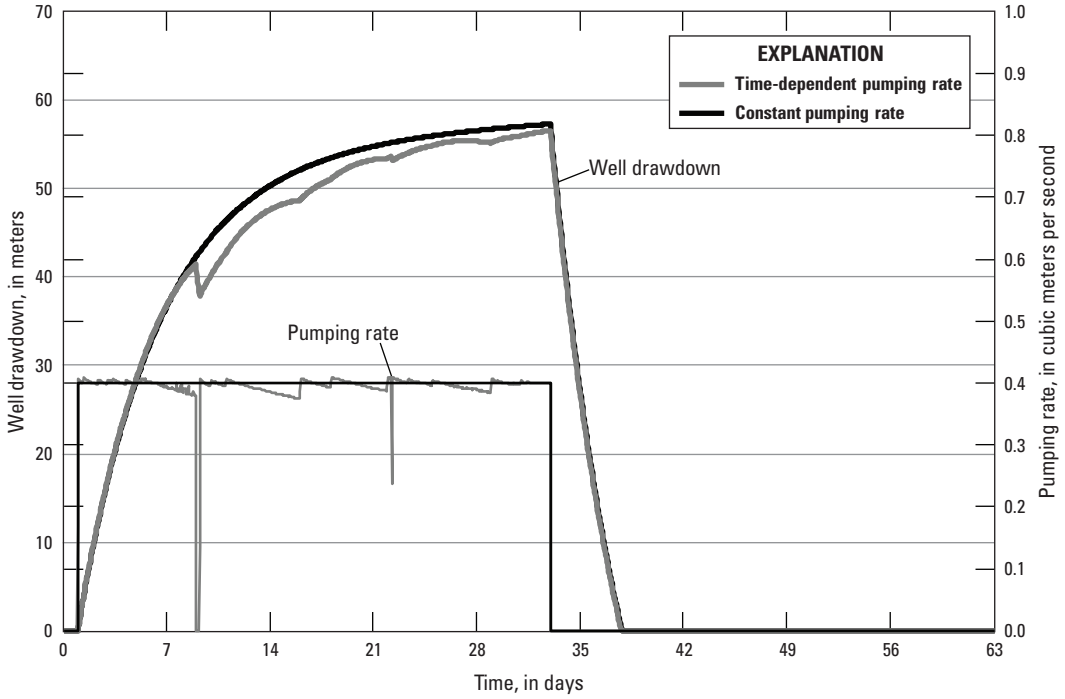


Figure 7.1. Simulation using CFP2 with time-dependent boundary conditions (pumping rate from Maréchal and others, 2008).

Fixed Head Limited Flow (FHLQ) Boundary Condition

A conduit with fixed head boundary condition can strongly affect inflow or outflow of the highly permeable conduit network. For example, water extraction from a network of conduits with sufficiently large diameters almost always results in water inflow through the fixed head boundary. Often, these exchange rates are limited by hydrologic properties of the porous matrix (for example, hydraulic conductivity). Therefore, it may be useful for some simulations to constrain the flow rate to the conduit network across a fixed head boundary and limit exchanges between the conduit network and the porous matrix. The FHLQ boundary condition is intended to limit inflow or outflow at constant head boundaries. If a user-defined limiting flow rate (threshold) is exceeded, the boundary condition switches from a fixed head to a constant-flow boundary, which results in variable head (Bauer and others, 2005):

$$FHLQ = \begin{Bmatrix} h = H, & Q \le Q_L \\ Q = Q_L, & \text{else} \end{Bmatrix} \sqrt{a^2 + b^2} \tag{7.2}$$

where

h is the head at the conduit node,
H is the fixed head value,
Q is the discharge at the boundary (negative values denote outflow), and
Q_L is the limiting flow rate (LQ).

This boundary condition was not included in the previous version of the conduit flow process (CFP1). The fixed head limited flow boundary condition existed in the discrete continuum model Carbonate Aquifer Void Evolution (CAVE; Liedl and others, 2003). The CAVE model provided the theoretical basis for CFP Mode 1. The FHLQ boundary condition has been added to CFP2 for Mode 1.

Well Boundary Condition

This boundary condition can be used to apply pumping to a conduit node. Prior to CFP2, this was done by using the recharge (RCH) package along with direct recharge percentage (CRCH). This update allows recharge to be apportioned to a conduit and extracted from the conduit to improve accounting of pumpage by conduits in the budget files. This is a newly developed feature of CFP2 to use with Mode 1.

Cauchy Boundary Condition

The Cauchy boundary condition (CYLQ) can be used to provide head-dependent flow to a node, like the River GHB package in MODFLOW. Flow at the Cauchy boundary, Q_{cy}, is computed as follows:

$$Q_{cy} = c_{cy}\left(h - h_{cy}\right) \tag{7.3}$$

where

c_{cy} is the Cauchy conductance, where 0 is less than c_{cy}, which is less than infinity, and is normal to the boundary;
h is the hydraulic head in the conduit; and
h_{cy} is the Cauchy head.

Negative values for Q_{cy} denote inflow into the conduit system. Additionally, inflow can be limited by a CYLQ condition, as for the FHLQ boundary described in this appendix (see the "Fixed Head Limited Flow (FHLQ) Boundary Condition" section).

$$CYLQ = \begin{cases} Q = Q_{cy}, & Q \le Q_L \\ Q = Q_L, & \text{else} \end{cases} \tag{7.4}$$

where

Q_L is the limiting flow rate (LQ).

Limited Head Boundary Condition (LH)

This boundary condition can be used to provide a limitation for the node-head, for example, to represent a flooded sinkhole for which the limited head is equivalent to the ground surface (fig. 7.2). This is newly developed feature of CFP2 for use with Mode 1. The hydraulic head for the limited head (LH) boundary condition is computed as follows:

$$h_{LH} = \begin{cases} h, & h \le H_{LH} \\ h = H_{LH}, & \text{else} \end{cases} \tag{7.5}$$

where

h is the computed hydraulic head
h_{LH} is the head at the limited head boundary node, and
H_{LH} is the limited head.

Note that LH boundaries only apply to groundwater-outflow situations and, therefore, are equivalent to FHLQ boundaries where LQ = 0, but they allow for improved representation of water flow in water budgets.

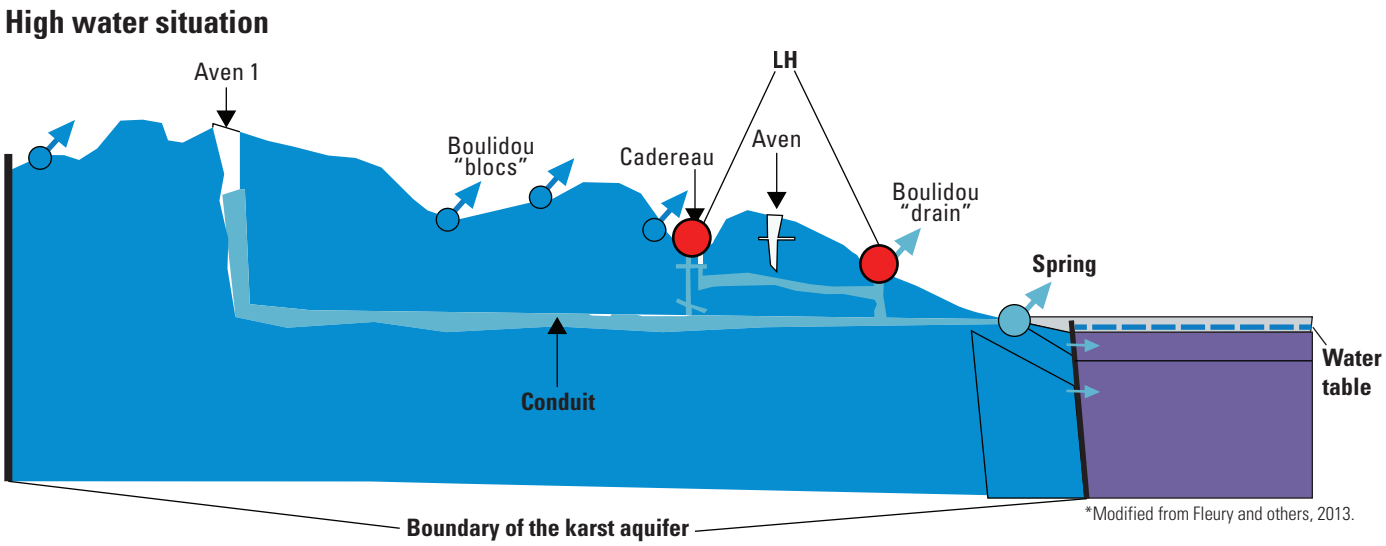


Figure 7.2. Conceptual diagram of the application of the limited head (LH) boundary (from Fleury and others, 2013).

Conduit-Associated Drainable Storage (CADS)

Karst aquifers can be conceptualized in several ways. One common way is to represent the karst system by highly permeable conduits embedded in a low permeability matrix continuum (for example, Király, 1997). Others describe karst systems as pipes draining associated karstic storage, and the matrix storage is negligible (for example, Mangin, 1994). Depending on the investigated karst system, as well as the considered time scale, three karst features can predominate:

- Pipes
- Drainable storage (like caves, large fractures, and fissures)
- The matrix continuum.

An example is the large-scale pumping test by Maréchal and others (2008), which demonstrates the drawdown reaction of conduit and matrix heads in response to long-term pumping. Results from this test provide arguments for the presence and importance of karst conduits, associated karstic storage (responsible for conduit drawdown), and matrix storage (responsible for matrix drawdown).

To consider drainable storage in CFP2 Mode 1, the CADS package (conduit-associated drainable storage) was developed. The CADS package simulates an independent storage zone with an independently calculated hydraulic head. The CADS storage is assumed to be in direct hydraulic contact with draining conduits:

$$h_{conduit} = h_{CADS} \tag{7.6}$$

where

$h_{conduit}$ is the head at the conduit node, and
h_{CADS} is the CADS head, which is also related to the conduit node.

It is assumed that water released from the CADS due to head variations immediately enters the conduit, resulting in additional discharge. The resulting flow rate from and to the CADS, Q_{CADS}, is as follows:

$$Q_{CADS} = \frac{V_t - V_{t-\Delta t}}{\Delta t} \tag{7.7}$$

where

V_t is the volume of the CADS at time t, and
Δt is the time step size.

Finally, the volume of the CADS, V_{CADS}, is computed:

$$V_{CADS} = L_{CADS} W_{CADS} \left(h_{CADS} - z_{bottom} \right); h_{CADS} > z_{bottom} \tag{7.8}$$

where

L_{CADS} is the length, which is assumed to equal the length of the conduit;
W_{CADS} is the width of the CAD storage; and
z_{bottom} is the conduit base elevation.

The CADS package was newly developed for CFP2 for use with Mode 1. Figure 7.3 shows the conceptual implementation of CADS for a karst catchment.

Potential application of the CADS functionality in CFP2 Mode1 is demonstrated here by simulating the large-scale pumping test by Maréchal and others (2008) in a very preliminary, simplified manner (intended only to demonstrate CADS functionality). Measured data used to develop this demonstration model are available in Maréchal and others (2008); a conceptual sketch of the demonstration model is shown in figure 7.4. Parameters used are presented in table 7.1. Results of computed and measured drawdown at the conduit and in the matrix are presented in figure 7.5 and resulting computed flows for several compartments of the model are listed in table 7.2.

Simulated drawdown and flow terms compared reasonably well with the measured data from Maréchal and others (2008). This comparison illustrates the new functionality of the CADS function in CFP2 Mode 1 and is for demonstration purposes only. This demonstration, which was not rigorously calibrated, shows the potential applicability of CFP MODE 1 with CADS and the FHLQ boundary condition to represent long-term pumping from karst water resources.

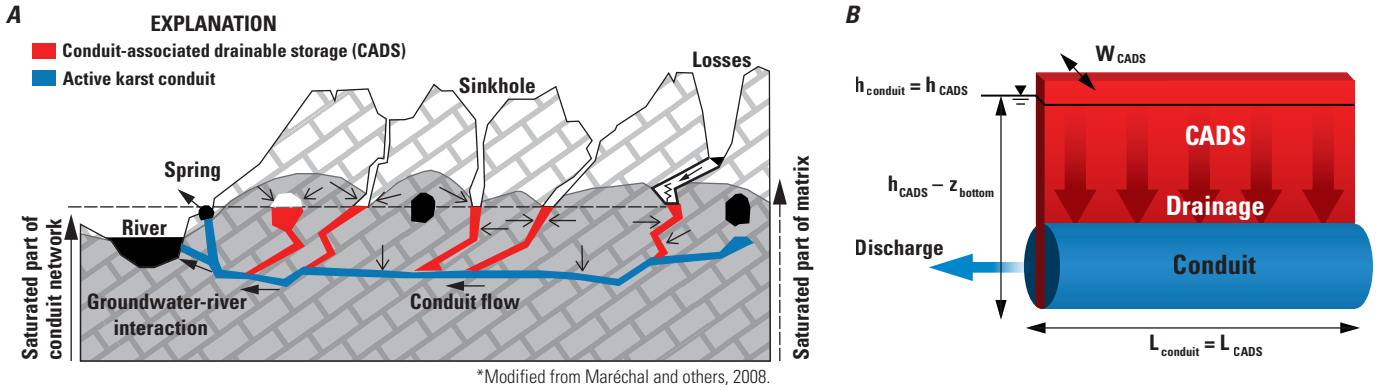


Figure 7.3. Conceptual implementation of CADS for a karst catchment as *A*, conceptual cross section (diagram modified from Maréchal and others, 2008) and *B*, simplification to illustrate terms used in equation 7.8.

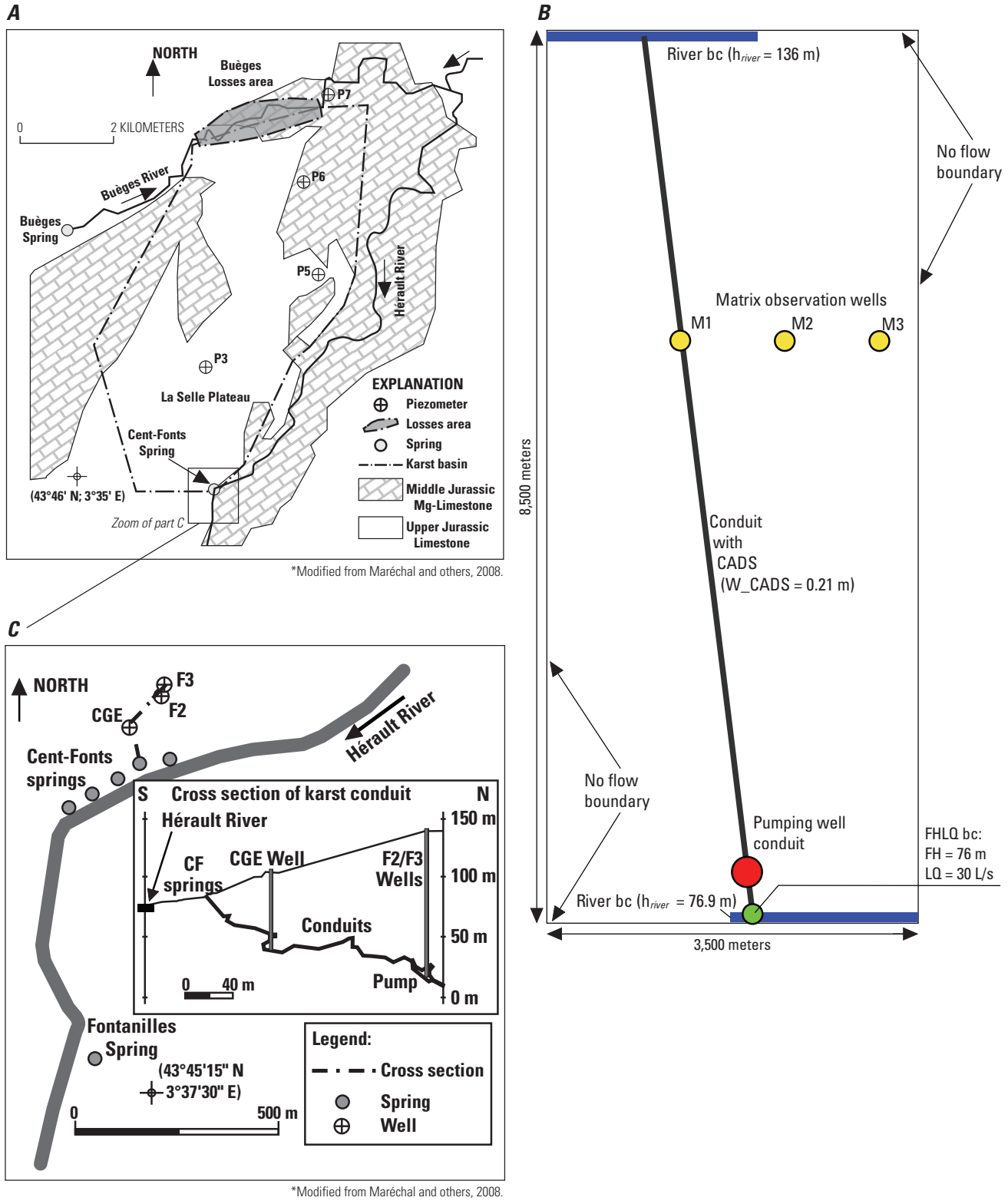


Figure 7.4. Map and diagram of the model demonstration of a large-scale pumping test: *A*, area of catchment where test was done and location of pumping well F3 and Cent-Fonts spring inside the inset black box (map from Maréchal and others, 2008); *B*, conceptual representation in planimetric view of the modeled scenario of the test showing the river boundary condition (River bc) where the head in the river (h_{river}) is specified as meters above sea level (masl), and fixed head limited flow boundary condition (FHLQ bc) where the fixed head (FH) is 76 masl and the limited flux (LQ) is specified as 30 liters per second (L/s); and *C*, Cent-Fonts spring area showing Cent-Fonts springs, Fontanilles Spring, Hérault River; CGE, F2, and F3 wells, and cross section of karst conduits.

Table 7.1. Parameters used for the large-scale pumping test model comparing CFP2 MODE 1 with Conduit Accessible Drainable Storage (CADS) to the data from Maréchal and others (2008).

[a^{-1}, per annum; h, hours; km, kilometers; km^2, square kilometers; m, meter; m^2, square meters; mm, millimeters; s^{-1}, per second; P1, P2, and P3 refer to stress period time step length with recharge rates defined by recharge rates R1, R2, and R3, respectively; —, no data]

Parameter	Data from Maréchal and others (2008)	Parameter for CFP MODE 1/CADS
Matrix continuum		
Area	30 km^2	3.5 km × 8.5 km = 29.75 km^2; 35 x 85 cells with $\Delta x = \Delta y$ = 100 m
top/bottom	—	250 m / –150 m
$T¬m$	1.6×10^{-5} m^2s^{-1}	—
K_m	—	9.0×10^{-6} ms^{-1}
S_m	0.007	0.007 (S_S = 0.00001)
Conduit		
Length	—	~ 9.1 km
Diameter	—	3.5 m
Roughness	—	0.01 m
S_c (free surface area of dewatering conduit network)	1,900 m^2	$S_C \sim A_{CADS}$; W_{CADS} = 1,900 m^2 / 9,100 m = 0.21 m
pipe conductance (α)	—	4.5×10^{-5} m^2s^{-1}
time discretization	—	P1: 1 day initialization (steady state)
		P2: 32 days pumping (transient, dt = 1h)
		P3: 32 days recovery (transient, dt = 1h)
diffuse recharge	—	R1: $6.3376 \times 10^{-9}¬$ ms^{-1} (200 mm a^{-1})
		R2: $6.3376 \times 10^{-9}¬$ ms^{-1}
		R3: $6.3376 \times 10^{-9}¬$ ms^{-1}

Table 7.2. Comparison between flow rates from Maréchal and others (2008) and the CFP2 MODE 1 with Conduit Accessible Drainable Storage (CADS) simulated results.

[bc, boundary condtion; CADS Conduit Accessible Drainable Storage; FHLQ, Fixed Head Limited Flow; m, meter; m^3, cubic meters; s^{-1}, per second]

Flow compartment	Data from Maréchal and others (2008)	Computed by CFP MODE 1/CADS
Spring discharge (steady state)	0.250 m^3s^{-1}	0.148 m^3s^{-1}
Inflow Hérault (river bc, at the karst spring)	0.030 m^3s^{-1}	0.030 m^3s^{-1} (FHLQ bc)
Bueges losses (river bc, at the origin of the conduit)	0.010 m^3s^{-1}	0.005 m^3s^{-1} (steady state) up to 0.010 m^3s^{-1} (during pumping)
Matrix water inflow to conduits	0.240 m^3s^{-1}	0.147 m^3s^{-1} (steady state) up to 0.367 m^3s^{-1} (during pumping)

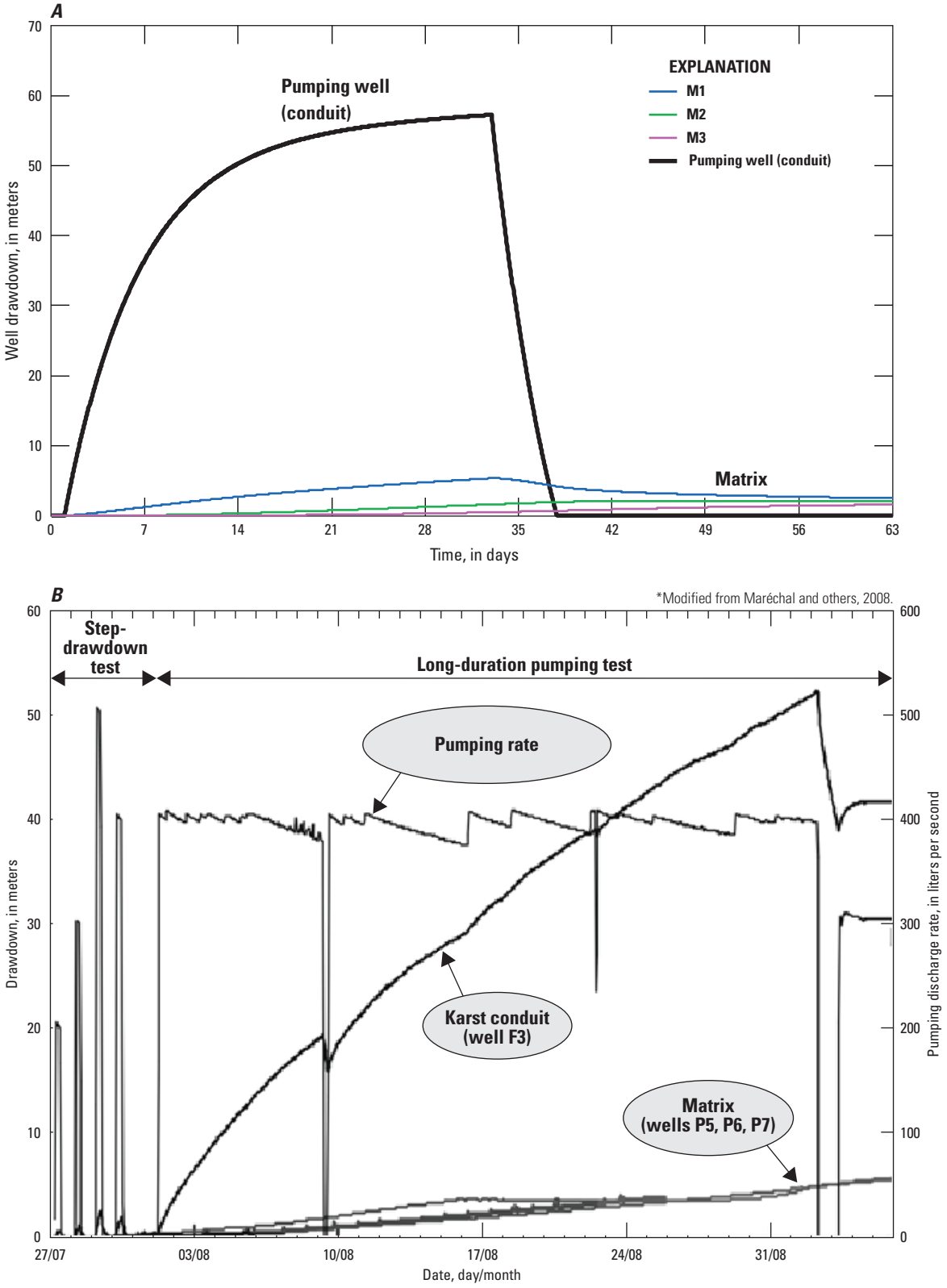


Figure 7.5. Drawdown results from a large-scale pumping test: *A*, computed drawdown for matrix wells M1, M2 and M3 and conduit shown in figure 7.4*B*; and *B*, measured drawdown from field test (from Maréchal and others, 2008).

Multi-Layer CADS (CADSML)

This new option allows a CADS volume to vary as a function of elevation. Hence, CADS width W_{CADS} is no longer uniform with depth. This allows Multi-Layer CADS (CADSML) users to limit CADS to specific areas (that is, restrict CADS to below the ground surface or to layers with a free water table inside CADS storage).

The numerical approach to implement CADSML is like that of partially filled pipe storage (PFPS), which is presented in a separate section of this appendix (equations 7–9 and 7–10). The concept of CADSML is illustrated in figure 7.6. Although the system shown in figure 7.6 has only two layers with non-zero width, CADSML can account for up to four different CADS layers ($z1$ / $W_{CADS}1$ … $z4$ / $W_{CADS}4$).

The numerical solution of multilayer CADS can result in oscillations, which is likely for situations with rapidly varying boundary conditions (for example, reduction or stopping of pumping) and subsequent conduit head change along flowpaths where values of W_{CADS} vary notably. The CFP2 considers oscillations by analyzing five successive iterations and, if there are oscillations, changes the relaxation parameter (line 20 of the CFP2 input file, Shoemaker and others, 2008, p. 27) to 5 percent of the initial value until the next iteration loop.

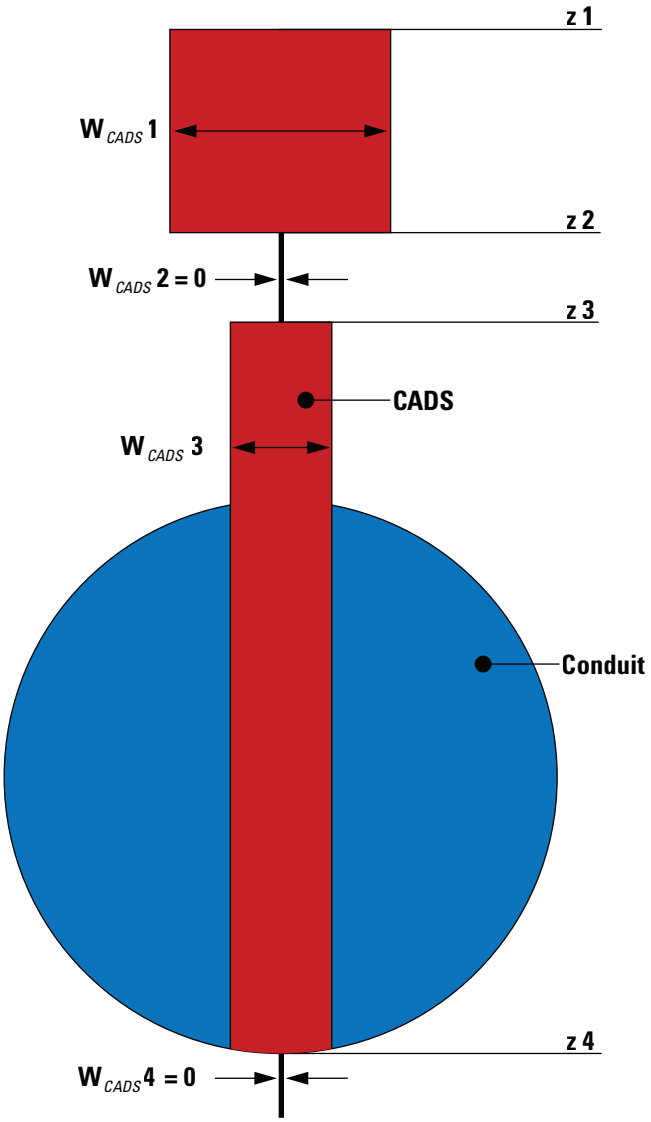


Figure 7.6. Conceptual diagram for multi-layer conduit-associated drainable storage (CADSML) package.

CADS Recharge

The CADS recharge allows defining the percentage of diffuse areal recharge that is directly routed into CADS. By definition, the CADS heads are equal to conduit heads, and therefore, CADS recharge results in immediate flow from the CADS-associated node into tubes. In subsequent iterations, this additional tube discharge causes head variations in conduits (and CADS). By implementing CADS as described, CADS recharge is hydraulically equal to conduit recharge.

Partially Filled Pipe Storage (PFPS)

The CFP2 was modified from the previous version to account for flow through partially filled pipes. If pipes are partially filled (for example, pipe bottom is less than head and head is less than pipe top), the CFP2 Mode 1 corrects the flow computation to apply appropriately to partially filled pipes (Shoemaker and others, 2008). The water volume coming from and going to the pipe storage is computed for tracking budget terms. This newly developed feature for the CFP2 Mode 1 improves representation of pipe storage in the water budget (that is, PFPS is part of the active flow system resulting in additional discharge). Figure 7.7 is a conceptual sketch of the PFPS implementation.

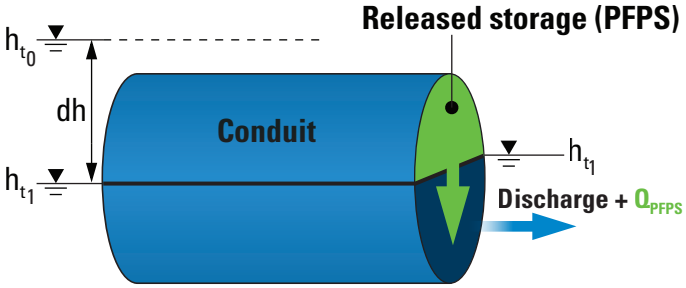


Figure 7.7. Conceptual implementation of partially filled pipe storage (PFPS). Drawdown, dh, equals the change in hydraulic head during the time step (t_0 until t_1); the outflow discharge at t_1 is the sum of conduit inflow and the flow rate, Q_{PFPS}, of released storage as the pipe transitions from filled to partially filled.

The flow rate in and out, or change in storage (Q), in PFPS is estimated similarly as in CADS (see the "Conduit Associated Drainable Storage" section):

$$Q_{PFPS} = \frac{V_{t2} - V_{t0}}{\Delta t} \tag{7.9}$$

where

V_t is the volume at time, t, and
V_{t0} is the initial volume at the start of the time step.

Unlike in CADS, the volume is no longer a linear function of head (h). The finite difference of volume resulting from head change can be computed, however, as follows:

$$V_{t1} - V_{t0} = f\left(h_{t1}\right) - f\left(h_{t0}\right) = \frac{f\left(h_{t1}^{kiter-1}\right) - f\left(h_{t0}\right)}{h_{t1}^{kiter-1} - h_{t0}}\left(h_{t1}^{kiter} - h_{t0}\right) \tag{7.10}$$

where

$h_{t1}^{kiter-1}$ is the hydraulic head in the previous iteration at time, t;
h_{t1}^{kiter} is the hydraulic head in the current iteration; and
h_{t0} is the hydraulic head in the previous iteration at the start of the time step.

The functional behavior of $V = f(h)$ is described by Shoemaker and others (2008, p. 8) and is beyond the scope of this document. Please note that PFPS does not include dry pipes; if the head falls below the conduit base, the pipe is no longer partly full, and PFPS is not active.

It should be noted that CFP2 Mode 1 is not designed to compute karst hydraulics in partially filled conduits. The free-surface flow processes in partially filled pipes can result in unsteady flow with considerable effects due to inertial forces and momentum. The steady-state approach implemented in CFP Mode 1 neglects dynamic processes (for example, water released from PFPS is immediately part of the active flow system, resulting in an immediate change of discharge); therefore, flow computation with PFPS considered can be unstable for some situations with too much water release from PFPS. For instance, large head changes or large conduit diameters can result in increasing discharge ($Q_{overall} + Q_{PFPS}$), increasing heads, filled conduits, and, therefore, $Q_{PFPS} = 0$.

References Cited

Bauer, S., Liedl, R., and Sauter, M., 2005, Modeling the influence of epikarst evolution on karst aquifer genesis—A time-variant recharge boundary condition for joint karst-epikarst development: Water Resources Research, v. 41, no. 9, 12 p., https://doi.org/10.1029/2004wr003321.

Fleury, P., Maréchal, J.-C., and Ladouche, B., 2013, Karst flash-flood forecasting in the city of Nîmes (southern France): Engineering Geology, v. 164, p. 26–35. https://doi.org/10.1016/j.enggeo.2013.06.007.

Király, L., 1997, Modelling karst aquifers by the combined discrete channel and continuum approach, 6th Conference on limestone hydrology and fissured aquifers: Université de Franche-Comté, Sciences et Technique de l'Environment, La Chaux-de-Fonds, p. 1–26.

Liedl, R., Sauter, M., Hückinghaus, D., Clemens, T., and Teutsch, G., 2003, Simulation of the development of karst aquifers using a coupled continuum pipe flow model: Water Resources Research, v. 39, no. 3, 11 p., https://doi.org/10.1029/2001WR001206.

Mangin, A., 1994, Karst hydrogeology, *in* Stanford, J., Gibert, J., and Danielopol, D., eds., Ground-water ecology: San Diego, Calif., Academic Press, p. 43–67.

Maréchal, J.-C., Ladouche, B., Dörfliger, N., and Lachassagne, P., 2008, Interpretation of pumping tests in a mixed flow karst system: Water Resources Research, v. 44, no. 5, 18 p., https://doi.org/10.1029/2007WR006288.

Shoemaker, W.B., Kuniansky, E.L., Birk, S., Bauer, S., and Swain, E.D., 2008, Documentation of a conduit flow process (CFP) for MODFLOW-2005: U.S. Geological Survey Techniques and Methods Book, Chapter A24, 50 p., https://pubs.usgs.gov/tm/tm6a24/pdf/tm6-A24.pdf.

Appendix 8. Conduit Flow Process (CFP2) Input File Documentation for New Capabilities of CFP2 Mode 1—Discrete Conduits

This appendix summarizes the variables, input data, and input file modifications needed for each of the new features in the second version of the Conduit Flow Process (CFP2). The CFP2 is backward compatible with the previous version of CFP (CFP1) if minor modifications are made to the CFP input file. These modifications are described in the "Modification and Increased Capabilities for Specifying Conduit Boundary Conditions" section. The updates in this version of CFP focus on CFP2 Mode 1, the CFP mode of operation in which users specify a discrete conduit network for the aquifer. Other CFP modes are described in the original CFP documentation (Shoemaker and others, 2008). Three different input files exist for Mode 1:

- CFP, the Conduit Flow Process input file (see Shoemaker and others, 2008, p. 27).
- CRCH, the Conduit Recharge input file (see Shoemaker and others, 2008, p. 30).
- COC, the Conduit Output Control file (Shoemaker and others, 2008, p. 30).

To clarify, item numbers in the following sections are not line numbers; they signify all required input data for each item. Thus, each item may be composed of more than one line of data. The number of lines for each item depends on the number of parameters that must be specified.

New Capabilities for Simulating Conduit Storage and Flow

This section provides input instructions for implementing recent improvements to conduit storage and flow in CFP2 Mode 1. The purpose of CFP2 Mode 1 is to represent a conduit network as discrete elements (for example, tubes) that are connected through nodes within the model domain. The storage and flow improvements for CFP2 Mode 1 include Conduit Associated Drainable Storage (CADS), multiple-layer CADS (CADSML), and length-dependent exchange. In the input files, item 0 is a "#Required comment line" that can be useful for annotating package input. Item 1 is an integer value that specifies the CFP mode of operation. All newly added features in this version of CFP are for CFP2 Mode 1, thus item 1 of the input files has a value of 1.

Conduit-Associated Drainable Storage (CADS)

The Conduit-Associated Drainable Storage (CADS) functionality (see Reimann and others, 2014) is activated by the keyword CADS in item 2 of the CFP input file. The theoretical basis for CADS is provided in appendix 7 and shown in figure 7.3. Item 28 is an optional comment line to document data provided in item 29. Item 29 defines the node number (NO_N) and exchange factor (K_EXCHANGE) for each CFP node in both versions of CFP. In CFP2 with CADS, the width of the CADS is defined in the third column of input (W_CADS). With this format, item 29 reads values for three variables: NO_N, K_EXCHANGE, and W_CADS.

Example input file with W_CADS = 0.25 meter (m):

```
0.      #optional comment line
1.      1
2.      CADS
...
28.     #NODE K_EXCHANGE W_CADS
29.     1       0.000050        0.25
        2       0.000100        0.25
        3       0.000100        0.25
        4       0.000100        0.25
        5       0.000100        0.25
        6       0.000050        0.25
```

Multiple-Layer CADS (CADSML)

Multiple-layer CADS (CADSML) is activated by the keyword CADSML in item 2 of the CFP input file. Item 29 is repeated for each CFP node. The values on each line of item 29 are NO_N, K_EXCHANGE, and paired values of depth *z*, and W_CADS, the associated width of the CAD storage (above the specified depth *z*) for each vertical interval of CADS that is specified. Using this format, item 29 contains values for 10 variables: NO_N, K_EXCHANGE, Z1, W_CADS1, Z2, W_CADS2, Z3, W_CADS3, Z4, and W_CADS4. In CFP2, CADSML is limited to four CADS-layers, and layers without CADS must have W_CADS defined as zero.

Example input file with W_CADS = 0.25 meter (m) for depth interval from 150 m to 30 m and W_CADS = 0.50 m from 30 m to 0 m:

```
0.      #Required comment line
1.      1
2.      CADSML
...
28.     #NODE K_EXCHANGE W_CADS
29.     1       0.000050        150 0.25 30 0.5 0 0 0 0
        2       0.000100        150 0.25 30 0.5 0 0 0 0
...
        5       0.000100        150 0.25 30 0.5 0 0 0 0
        6       0.000050        150 0.25 30 0.5 0 0 0 0
```

CADS Recharge

The CADS recharge is parameterized in a second input file for CFP, the CRCH input file (see Shoemaker and others, 2008, p. 30). Item 3 of the CRCH input contains input for each CFP node; a node number and a value for the fraction of diffuse recharge, P_CRCH, are specified on each line. CADS recharge is activated for a node by specifying an additional optional variable, P_CDSRCH, on the line after the P_CRCH value. The P_CDSRCH variable is a real number equal to the fraction of diffuse areal recharge (entered in the MODFLOW-2005 RCH package) partitioned directly into CADS. If P_CDSRCH is not set in the CRCH input file, the default value of 0.0 is used. In the following example, a value of P_CRCH equal to 0.500 and a value of P_CDSRCH equal to 1.0 indicate 100 percent of diffuse areal recharge is routed to CADS in node 1. For nodes 2 through 6, however, the default input for P_CDSRCH effectively routes zero diffuse areal recharge to CADS.

Example CRCH input file (100 percent of diffuse areal recharge routed to CADS in node 1):

```
0.      #Required comment line
1.        1
2. #node P_CRCH P_CDSRCH
3.      1 0.5000 1.0
        2 0.500
        3 0.500
        4 0.500
        5 0.500
        6 0.500
```

Length-Dependent Exchange

This option allows the water-transfer coefficient, α_{ex}, to be applied per unit length l; for example, input for this exchange is provided as $\alpha_{ex}\, l^{-1}$. Hence, it is not necessary to specify water-transfer coefficients to each node for which the node-associated conduit length varies (for example, if the spatial discretization of matrix cells is varying). The length-dependent exchange is activated by setting the SA-EXCHANGE flag (see Shoemaker and others, 2008, p. 28) to a value of 2 (item 14 in the example shown). In the following case with a 100-unit length tube, the α_{ex} is 0.002 and the $\alpha_{ex}\, l^{-1}$ is equal to the α_{ex} divided by the length; thus, $\alpha_{ex}\, l^{-1}$ is equal to 0.00002.

Example CFP input file (5 tubes with 100 length units, 6 nodes, α_{ex} intended as 0.002 and $\alpha_{ex}\, l^{-1}$ input as 0.00002)

```
13.     #SA-EXCHANGE flag setting follows: 2 = length-dependent exchange is active
14.     2
15.     #Required comment line
…
28.     #Required comment line
29.     1 0.00002
        2 0.00002
        …
        6 0.00002
```

Modification and Increased Capabilities for Specifying Conduit Boundary Conditions

CFP2 Mode 1 has several new boundary conditions that can be applied to each conduit. The new boundary conditions are fixed head-limited flow (FHLQ), well boundary and specified pumping from conduits by stress period, Cauchy boundary, limited head (LH) boundary, and time-dependent (TD) boundary. Each of these boundaries has a specific keyword to activate it. The keywords and input for each boundary condition are described in the corresponding sections. All additional boundary conditions, like the FHLQ boundary condition, are activated and defined in the CFP input file.

Activation of boundary conditions in one or more nodes in the simulation is enabled by specifying the keyword FBC (further boundary conditions) in item 2 of the CFP input file. Several keywords can be linked with a space or underscore, "_" (for example, FBC_CADS or FBC CADS). In item 27 in CFP1 Mode 1, the input parameters are the node number (NO_N) and initial hydraulic head value (N_HEAD). In CFP2 Mode 1, an additional keyword can be supplied to define the boundary condition type (BC_TYPE). If no boundary conditions are defined for that node, the keyword "x" is mandatory to indicate no further boundary conditions need be considered for that node. For nodes that are specified with a boundary condition in item 27, BC_TYPE (boundary condition type at specified node), the optional boundary condition value and related parameters must be specified.

The specific keywords and inputs for each boundary condition are provided in the boundary condition description for each boundary condition type. The CFP2 is backward compatible with models built using CFP1 after some minor modifications to the CFP input file. For models built using CFP1, item 27 must have the keyword "x" added to the third column to specify that none of the new boundary conditions are to be applied to the specified node number on that line.

Fixed-Head Limited-Flow (FHLQ) Boundary Condition

For the fixed-head limited-flow boundary condition, the keyword "FHLQ" defines the node as having the FHLQ boundary. For each node with the boundary condition, after the FHLQ keyword, the limited-flow value (LQ) value must be provided. In the FHLQ format, item 27 for each node reads values for three or four variables: NO_N, N_HEAD, BC_TYPE, and the optional LQ for the BC_TYPE of FHLQ. N_HEAD (column 2) can be defined as a fixed head by using positive values, or values less than or equal to –1 can be used to indicate that heads will be calculated during simulation.

Example CFP input file (fixed-head limited-flow boundary in node 6):

```
0.      #Required comment line
1.      1
2.      FBC
...
26.     #Node Head
27.     1       -1      x
        2       -1      x
        3       -1      x
        4       -1      x
        5       -1      x
        6       50.0    FHLQ 0.045
```

Well Boundary Condition

The well boundary condition can be specified for one or more nodes in item 27 by using the keyword “WELL” in the third column of input values. The BC_TYPE keyword “WELL” defines the associated node as containing a well boundary condition.

Example CFP input file (well boundary condition in node 5):

```
0.      #Required comment line
1.      1
2.      FBC
...
26.     #Node Head
27.     1       -1      x
        2       -1      x
        3       -1      x
        4       -1      x
        5       -1      WELL
        6       50.0    x
```

Specified Pumping from Conduits

The CRCH input file allows definition of pumping rates for various stress periods. The intended pumping rates, variable QWELL (positive for inflow, negative for outflow), are added to item 2 formatted input rows following the value for P_CRCH (see the “CADS Recharge” section). In the following example, a pumping rate of 0.25 length cubed per time is specified from a well boundary condition specified on node 5 (see well boundary condition example provided earlier). The input reads values for three variables: the node identifier (NODE_NUMBERS), the percentage of diffuse conduit recharge (P_CRCH), and the flow rate (QWELL). Please note that for conduits with the defined well boundary, QWELL needs a specified value greater than or equal to 0 (zero) for each time step in case the well is not active (that is, during a recovery period).

Example CRCH input file:

```
0.      #Required comment line
1.        1
   #node P_CRCH QWELL
2.      1 0.000
        2 0.000
        3 0.000
        4 0.000
        5 0.000 -0.25
        6 0.000
```

Cauchy Boundary Condition

The CAUCHY functionality is activated by the keyword FBC in item 2 of the CFP input file, and boundary conditions are defined for individual nodes in item 27 by the keyword CAUCHY. In each line of item 27, the node identifier and hydraulic head at the specified node (for example, the Cauchy head, HCY, for nodes with the Cauchy boundary) are specified as the first two columns of input. For nodes where Cauchy boundaries are indicated, the keyword CAUCHY appears in column 3 and, when present, is followed by required values of the Cauchy conductivity (CCY) and the Cauchy limited inflow (CYLQ). For this format, each line in item 27 reads NO_N, HCY, BC_TYPE, CCY, and CYLQ (CCY and CYLQ are only required if the BC_TYPE is CAUCHY).

Example CFP input file (Cauchy boundary in node 6):

```
0.      #Required comment line
1.      1
2.      FBC
...
26.     #Node Head
27.     1       -1      x
        2       -1      x
        3       -1      x
        4       -1      x
        5       -1      x
        6       50.0    CAUCHY 2E-2 0.045
```

Limited Head Boundary Condition (LH)

The LH functionality is activated by the keyword FBC in item 2 of the CFP input file, and boundary conditions are defined for individual nodes in item 27 (previously only "NO_N, N_HEAD" were input) by the keyword "LH" in third column. One line is provided for each node and the selected boundary condition. The head (HLH) value on line 1 of item 27 represents the limited head, H_{LH}, when the keyword LH follows the head value. Subsequently, each line in item 27 specifies values for three variables: NO_N, HLH, and BC_TYPE.

Example CFP input file (Limited Head boundary in node 3):

```
0.      #Required comment line
1.      1
2.      FBC
...
26.     #Node Head
27.     1       -1      x
        2       -1      x
        3       60.0    LH
        4       -1      x
        5       -1      x
        6       50.0    x
```

Time-Dependent Boundary Conditions (TD)

Time-dependent input data need to be saved in an external ASCII file. The name of this external file must be included in the MODFLOW Name File (NAM). In the following example, time-dependent boundary condition data are contained in the file named EXT1.txt. The first part of this example shows the contents of EXT1.txt. The second part of the example shows declaration of EXT1.txt in the MODFLOW NAM file.

For the time-dependent boundary condition file, item 1 specifies the number of specified boundary condition timesteps. Item 2 contains one line with two entries for each boundary condition timestep. Column 1 specifies the time, (t), as model time beginning at t = 0 and continuing with one line for each model timestep for each dataset. Ensure that time-dependent data enclose the overall model time. The second entry on each line is the value for the boundary condition associated with that time. Boundary condition data are linearly interpolated between specified periods.

Example time-dependent boundary conditions data file:

```
1.      3
   # T-elapsd BndyCondn
2.      0       50
        1800    55
        3600    50
```

For the MODFLOW NAM file, the file EXT1.txt must be specified with the rest of the model input files with a unit number as shown in the example.

Example NAM file (file type DATA; here time-dependent data are provided by the file EXT1.txt with unit number 70):

```
CFP     57 SC.cfp
COC     59 SC.coc
DATA    70 EXT1.txt
```

The time-dependent nature of boundary conditions is activated in the CFP input files (Shoemaker and others, 2008, p. 27) by the keyword “TD” (time-dependent). The TD symbol is directly appended to the boundary-type keyword (that is, to the value of BC_TYPE—for example, FHLQTD). To specify a TD boundary condition for any boundary condition other than a well, replace the value of the variable that is to be time-dependent with the unit number of the external input file. For the well boundary condition with a TD boundary, the unit number is specified after the well boundary keyword (WELLTD). The examples that follow demonstrate the activation of TD input functionality for all supported types of boundary conditions.

Examples of CFP input files for Time-Dependent Boundary Conditions;
file unit number for external TD boundary conditions input data is 70:

```
# Fixed head TD boundary in node 6 read from file unit 70 (Shoemaker and others., 2008, p. 27)
...
26.     #Node Head
27.     1       -1      x
        ...
        6       70      TD
# FHLQ TD boundary in node 6 read from file unit 70
...
26.     #Node Head
27.     1       -1      x
        ...
        6       70      FHLQTD 1.0
# Cauchy TD boundary in node 6 read from file unit 70
...
26.     #Node Head
27.     1       -1      x
        ...
        6       70      CAUCHYTD 0.005 1.0

# Well TD boundary in node 5 read from file unit 70
...
26.     #Node Head
27.     1       -1      x
        ...
        5       -1      WELLTD 70
```

Modification and Increased Capabilities of CFP2 Output Files

Time-Series Analysis (TSA) Output

This functionality provides an additional output file, TSA.out, containing a list that has all inflow and outflow terms of the conduits for every output time step specified in the file used to control CFP output, the conduit output control file (COC). The TSA output is specified in the COC file by assigning a negative sign to the NNODES value in item 2. NNODES is an integer number the absolute value of which is equal to the number of nodes for which flow and head values are desired in separate output files (see Shoemaker and others, 2008, p. 30). In the following example, NNODES is set to −6, signifying that the inflows and outflows for six nodes are to be written to the time-series analysis output file (TSA.out):

Example COC input file to provide TSA output (for 6 nodes)

```
0.      #Required comment line
1.      # NNODES value follows
2.      -6
```

Time-Series Analysis Along Nodes (TSAN) and Along Tubes (TSAT)

This functionality allows the generation of files that contain output data (time series) resulting from simulated processes along pathways of user defined nodes (TSAN) and tubes (TSAT), such as for all nodes and tubes of a specific conduit branch. To activate the TSAN functionality, item 2 of the COC input file (Shoemaker and others, 2008, p. 30), the number of TSAN nodes (NTSAN), is specified after the number of nodes (NNODES). To activate the TSAT functionality, item 8 of the COC input must contain the number of pipes (NPIPES) and the number of TSAT tubes (NTSAT). The default value for NTSAN and NTSAT is 0, the effect of which suppresses the writing of TSAN and TSAT output, respectively. Following item 13 in the COC input file, the intended node numbers for TSAN output (TSAN_NODE_NUMBERS), the TSAN output frequency (TSAN_NTS), the intended tube numbers for TSAT output (TSAT_TUBE_NUMBERS), and the TSAT output frequency (TSAT_NTS) need to be added.

The example that follows demonstrates the modifications to activate TSAN / TSAT (shown in red). In the example, N_NTS is an integer value equal to the time-step interval for the output of node head and flow values. Here, N_NTS equal to 2 activates node-specific output at each of the specified nodes (NODE_NUMBERS) every two time-steps, and T_NTS is an integer value equal to the time-step interval for output of pipe-flow rates and Reynolds numbers. In the example, T_NTS equal to 2 activates pipe-specific output at each of the selected pipes (PIPE_NUMBERS) every two time-steps.

Example COC input file to provide TSAN and TSAT output
(NNODES is 6, TSAN for nodes 1, 2, and 4, and TSAT for tubes 2 and 3)

```
0.      #Required comment line
1.      #Required comment line
2.      6 3   #NNODES, NTSAN
3.      #Required comment line
4.      1 #NODE_NUMBERS
        …
        6
5.      #Required comment line
6.      1 # N_NTS
7.      #Required comment line
8.      5 2  # NPIPES, NTSAT
9.      #Required comment line
10      1  # PIPE_NUMBERS
        …
        5
11.     #Required comment line
12.     1 #T_NTS
13.     #Required comment line / following TSAN_NODE_NUMBERS
14.     1
        2
        4
15.     #Required comment line / following TSAN_NTS
16.     1
17.     #Required comment line / following TSAT_TUBE_NUMBERS
18.     2
        3
19.     #Required comment line / following TSAT_NTS
20.     1
```

References Cited

Reimann, T., Giese, M., Geyer, T., Liedl, R., Maréchal, J.C., and Shoemaker, W.B., 2014, Representation of water abstraction from a karst conduit with numerical discrete-continuum models: Hydrology and Earth System Sciences, v. 18, no. 1, p. 227–241, https://doi.org/10.5194/hess-18-227-2014.

Shoemaker, W.B., Kuniansky, E.L., Birk, S., Bauer, S., and Swain, E.D., 2008, Documentation of a conduit flow process (CFP) for MODFLOW-2005: U.S. Geological Survey Techniques and Methods 6–A24, 50 p., https://pubs.usgs.gov/tm/tm6a24/pdf/tm6-A24.pdf.

For more information concerning the research in this report, contact the
Director, California Water Science Center
U.S. Geological Survey
6000 J Street, Placer Hall
Sacramento, California 95819
https://ca.water.usgs.gov

Publishing support provided by the U.S. Geological Survey
Science Publishing Network, Sacramento Publishing Service Center

ISSN 2328-0328 (online)
https://doi.org/10.3133/tm6a60