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.

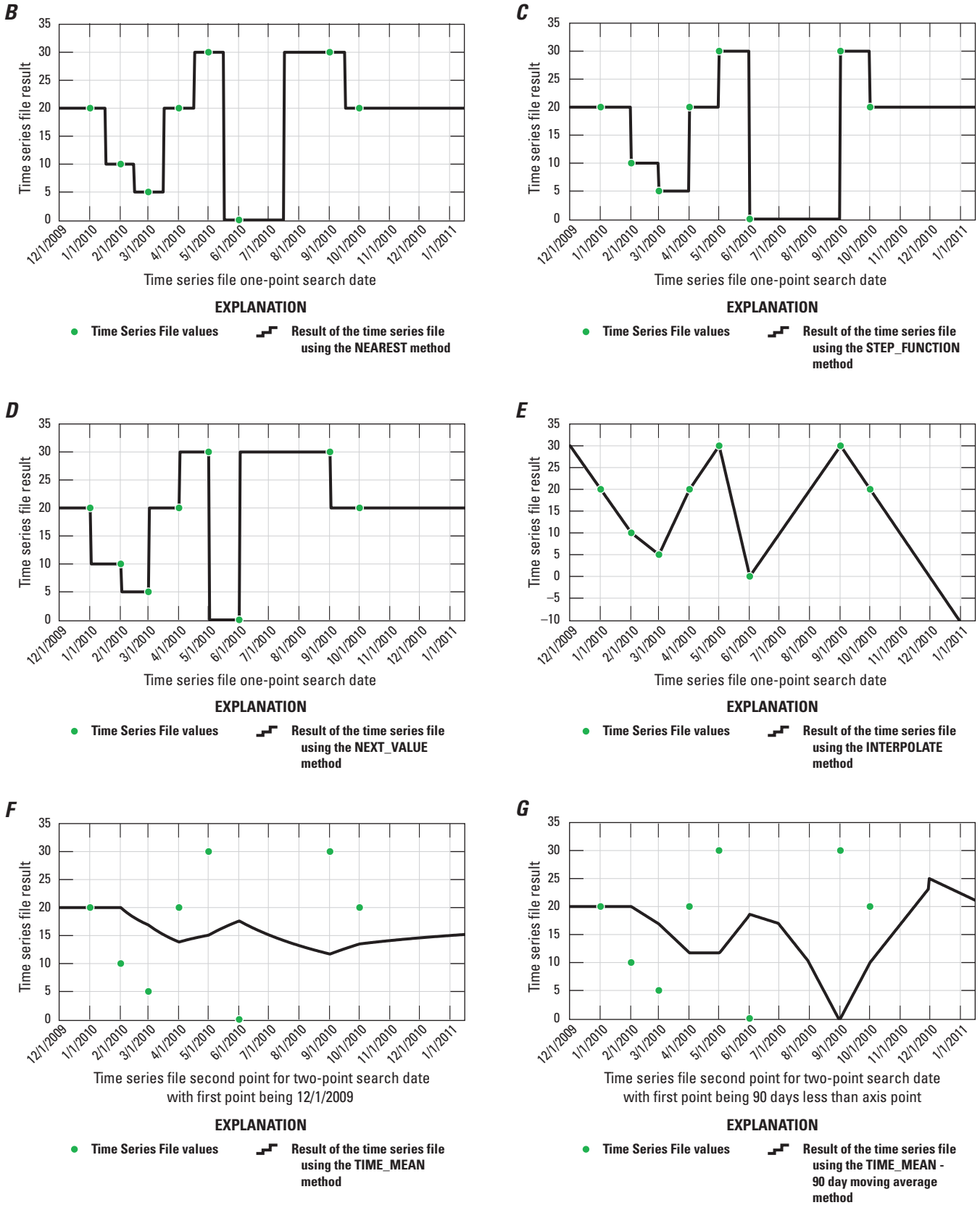

Figure 2.5. —Continued

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.

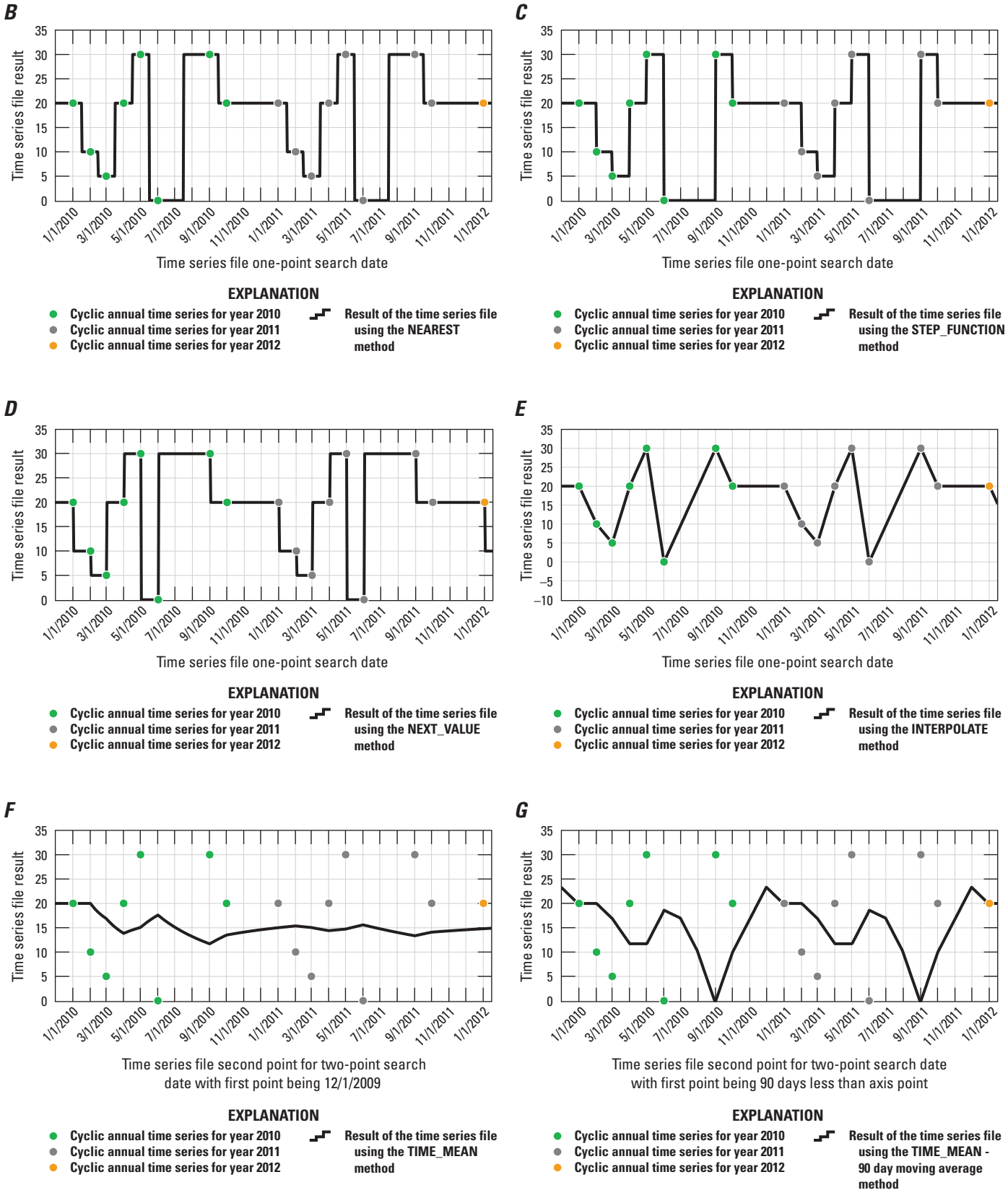

Figure 2.6. —Continued

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.