# Simulation of Canal and Control-Pond Operation at the Quivira National Wildlife Refuge, South-Central Kansas

By XIAODONG JIAN

---

U.S. GEOLOGICAL SURVEY
Water-Resources Investigations Report 97–4289

Prepared in cooperation with the
KANSAS GEOLOGICAL SURVEY

Lawrence, Kansas
1998

U.S. DEPARTMENT OF THE INTERIOR
BRUCE BABBITT, Secretary

U.S. GEOLOGICAL SURVEY
THOMAS J. CASADEVALL, Acting Director



For additional information write to:

District Chief

U.S. Geological Survey
4821 Quail Crest Place
Lawrence, Kansas 66049–3839

Copies of this report can be purchased from:

U.S. Geological Survey
Information Services
Box 25286
Federal Center
Denver, CO 80225–0826

# CONTENTS

## FIGURES



## TABLES

TABLES—Continued

## CONVERSION FACTORS, ABBREVIATION, AND DEFINITIONS

| Multiply | By | To obtain |
|---|---|---|
| acre | 4,047 | square meter |
| acre-foot (acre-ft) | 1,233 | cubic meter |
| acre-foot per day (acre-ft/d) | 1,233 | cubic meter per day |
| cubic foot per day ($ft^3/d$) | 0.02832 | cubic meter per day |
| cubic foot per second ($ft^3/s$) | 0.02832 | cubic meter per second |
| foot (ft) | 0.3048 | meter |
| foot per day (ft/d) | 0.3048 | meter per day |
| foot per second (ft/s) | 0.3048 | meter per second |
| foot per second squared ($ft/s^2$) | 0.3048 | meter per second squared |
| inch (in.) | 2.54 | centimeter |
| inch per day (in/d) | 25.4 | millimeter per day |
| inch per year (in/yr) | 25.4 | millimeter per year |
| mile (mi) | 1.609 | kilometer |
| mile per day (mi/d) | 1.609 | kilometer per day |
| millimeter (mm) | 0.03937 | inch |
| millimeter per day (mm/d) | 0.03937 | inch per day |
| square foot ($ft^2$) | 0.09290 | square meter |
| square mile ($mi^2$) | 2.590 | square kilometer |

Temperature can be converted to degrees Celsius (°C) or degrees Fahrenheit (°F) by the equations:

$$°C = 5/9\ (°F - 32),$$
$$°F = 9/5\ (°C) + 32.$$

**Sea level**: In this report, "sea level" refers to the National Geodetic Vertical Datum of 1929—a geodetic datum derived from a general adjustment of the first-order level nets of the United States and Canada, formerly called Sea Level Datum of 1929.

**Water year:** Water year is the 12-month period October 1 through September 30. The water year is designated by the calender year in which it ends. Thus, the year ending September 30, 1991, is called the "1991 water year."

# Simulation of Canal and Control-Pond Operation at the Quivira National Wildlife Refuge, South-Central Kansas

*By* Xiaodong Jian

## Abstract

Efficient water management of the Quivira National Wildlife Refuge, located in the Rattlesnake Creek Basin of south-central Kansas, is a complicated task. In a cooperative study with the Kansas Geological Survey, the U.S. Geological Survey developed a computer-based water-budget and flow-routing model to assist the U.S. Fish and Wildlife Service in determining the outcome of possible water-management options. The computer model uses network analysis to determine the optimal operation of canals and control ponds on the refuge. Applications of the model are presented that investigate the daily operation of canals and control ponds on the refuge using historical discharges and pond water levels.

Simulation of the daily operation of the canal and control-pond system at the refuge from June 11 through December 11, 1996, was conducted using the computer model. Simulated pond water levels matched well with measured ones. The root mean square error (RMSE) between the simulated and measured water levels in ponds was less than 0.13 foot except for two ponds. Water storage in ponds during the simulation period was substantially reduced due to water-surface evaporation and canal-flow transmission losses. Simulation of canal and control-pond operation was also conducted with different target pond water levels. This simulation used 1991 discharge, precipitation, and water-surface evaporation data to consider model results during drought conditions. Results indicated that lowering target pond water levels reduced water-surface evaporation, resulted in more water stored in ponds at the north end of the refuge, and caused a substantial decrease in the final volume of water stored in the main water-storage unit at the south end of the refuge (Little Salt Marsh).

## INTRODUCTION

The Quivira National Wildlife Refuge is located in the Rattlesnake Creek Basin of south-central Kansas (fig. 1). The refuge was established in 1953 to provide food, water, and a resting place for waterfowl and certain endangered species during their annual migrations. To provide the proper type of feeding and resting areas for wildlife, water is diverted from Rattlesnake Creek into a system of canals and impoundments. There are more than 30 control marshes and ponds (collectively referred to hereinafter as ponds) (water units) ranging in size from 7 to 1,768 acres currently (1997) on the refuge, three main canals, and numerous smaller canals and waterways (fig. 2).

Water is managed to provide a mixture of marsh and wet-meadow habitat in and adjacent to the control ponds. Large ponds, such as Little Salt Marsh (water unit 5, fig. 2) at the south end of the refuge, provide habitat and serve as the main water-storage units on the refuge. It is difficult for the refuge manager to determine optimally how much water should be stored in the large ponds instead of being released to smaller ponds. Habitat losses occur when water is too deep in the larger ponds and also when there is insufficient water to supply the smaller ponds.

An additional complication in the surface-water management of the refuge results from ground-water inflows to the area. The north part of the refuge is within a ground-water discharge area, and some of the

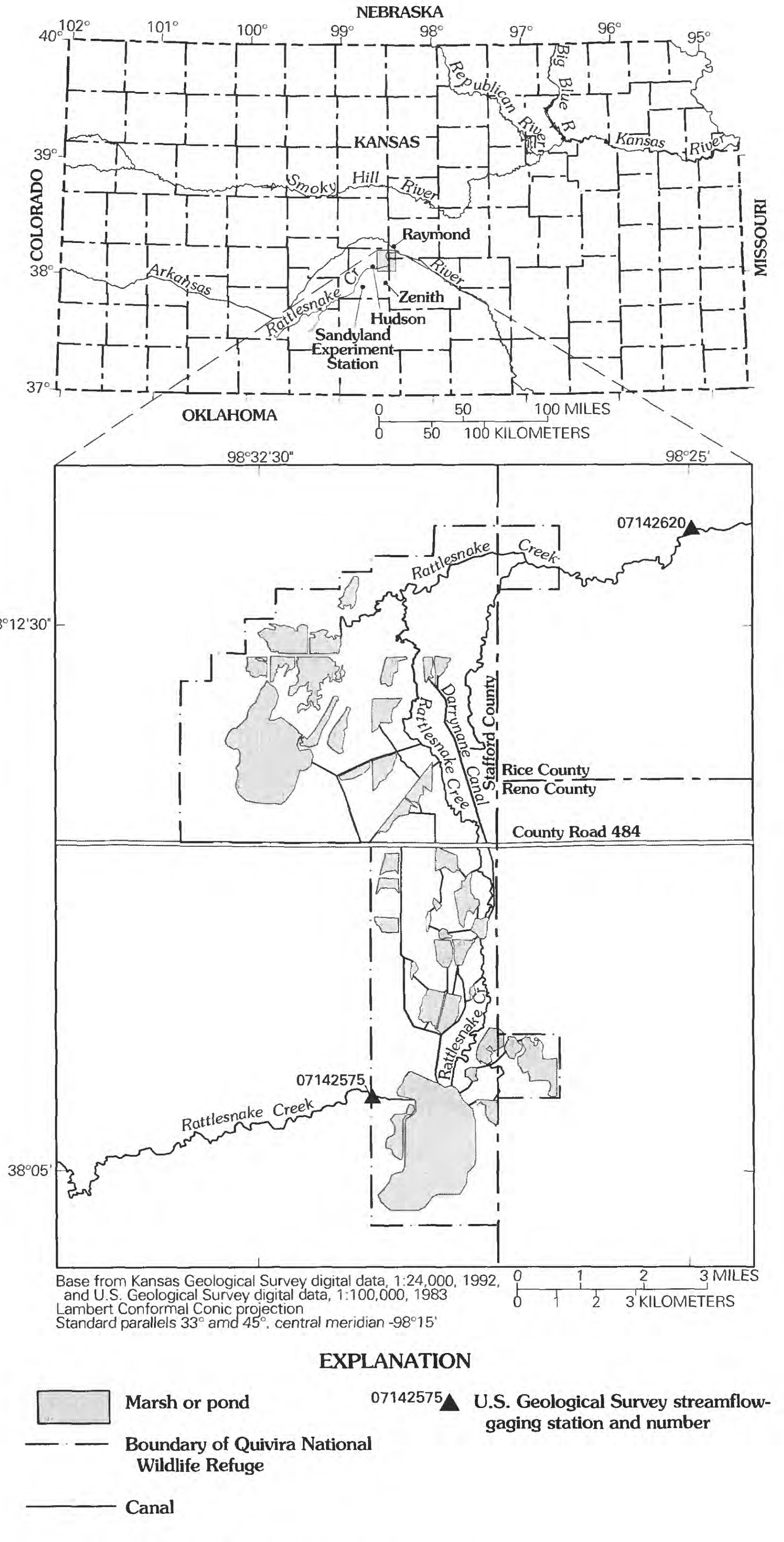
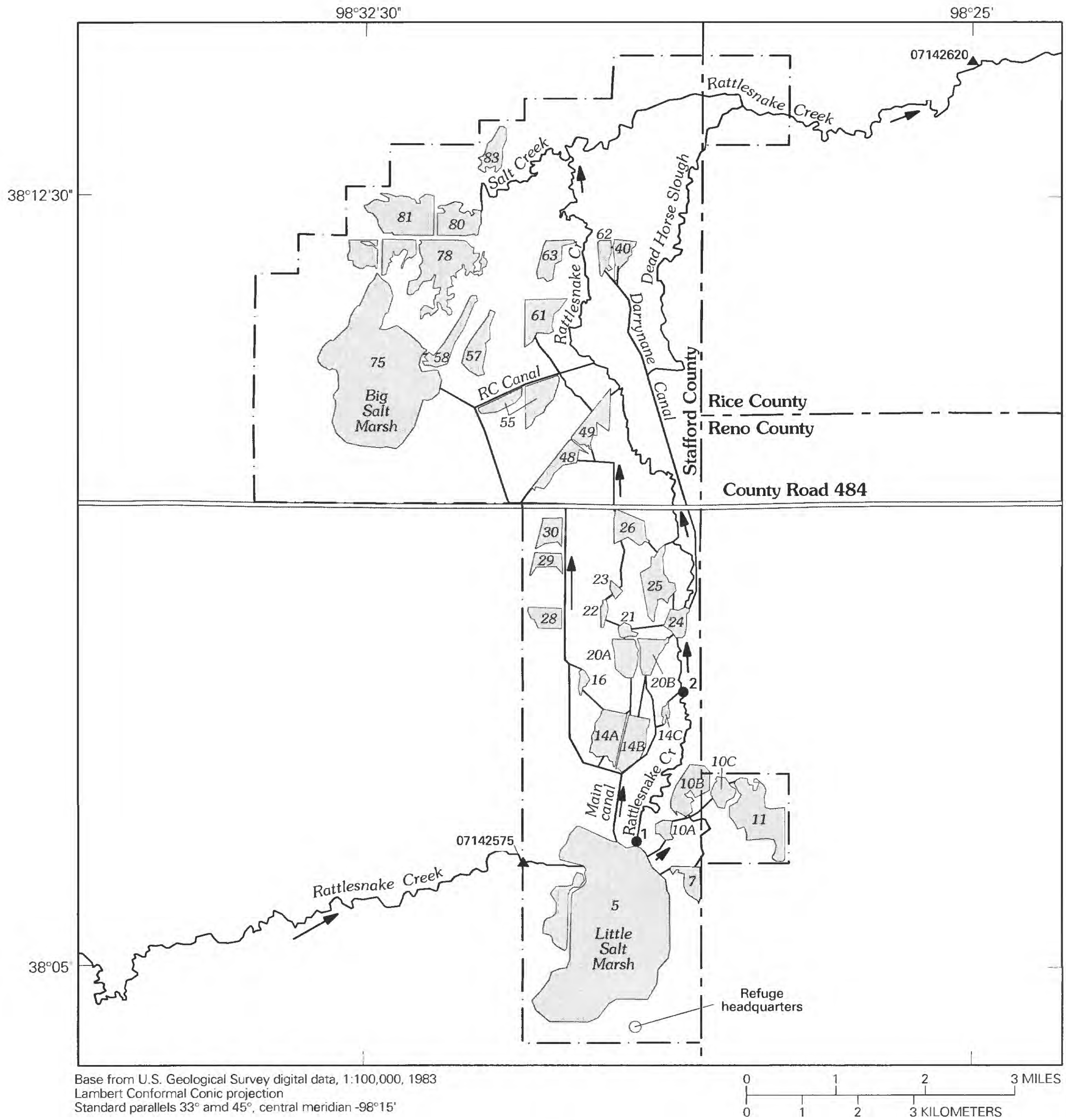



**Figure 1.** Location of Quivira National Wildlife Refuge, south-central Kansas.




## EXPLANATION

5 **Marsh or pond**—Number is water-unit number used for identification in tables

**Boundary of Quivira National Wildlife Refuge**

**Canal**

**Direction of flow**

07142575 **U.S. Geological Survey streamflow-gaging station and number**

1 **Seepage test site and number**

**Figure 2.** Ponds and canals at Quivira National Wildlife Refuge.

ponds receive a large part of their total water supply from ground-water sources (Megan Estep-Johnston, U.S. Fish and Wildlife Service, written commun., 1995). This is especially true of Big Salt Marsh (water unit 75, fig. 2). It is very difficult to determine the volume of water that is contributed to the refuge water supply from ground-water sources without a comprehensive simulation study.

Beginning in 1995, a 3-year study to develop a water-budget and flow-routing model to assist the U.S. Fish and Wildlife Service (USFWS) in determining the outcome of possible water-management options was begun by the U.S. Geological Survey (USGS) in cooperation with the Kansas Geological Survey (KGS). The objectives of the study were:

1. To develop a network flow model that incorporates linear-programming techniques to determine efficient management strategies for water use.
2. To provide simulation results using historic streamflow and water-level data.

The network flow model developed for this study can be adapted to any configuration of canals and ponds for similar water-management problems.

## Purpose and Scope

The purpose of this report is to describe the development of a computer model to simulate the water budget and surface-water flow routing for the Quivira National Wildlife Refuge to help determine the effects of possible water-management options on the distribution of available water within the refuge. The report includes a description of a network model to simulate the water budget of the refuge and the routing of water throughout the refuge. The network flow model consisted of nodes and arcs, where a node was any location in the refuge where the water budget was computed. Nodes represented all ponds, joints of canals, and any other terrestrial areas of interest. Arcs connected the nodes and allowed simulation of water movement from one node to another. A linear-network flow technique was used to simulate flow through arcs.

This report also presents results of simulations of daily operation of canals and ponds based on 1996 and 1991 conditions. The simulation for 1996 was used to investigate pond operation with measured pond levels. The simulation for 1991 was used to investigate the operation of ponds under drought flow conditions with different simulated pond target levels.

## Acknowledgments

The author thanks Marios Sophocleous and Greg Pouch of the Kansas Geological Survey for providing valuable data and suggestions. The author also thanks Megan Estep-Johnston of the U.S. Fish and Wildlife Service and Quivira National Wildlife Refuge personnel for providing crucial information and essential hydrologic data. Special thanks are given to David M. Wolock and Andrew C. Ziegler of the U.S. Geological Survey for their advice throughout the study.

# PHYSICAL AND HYDROLOGIC FEATURES OF QUIVIRA NATIONAL WILDLIFE REFUGE

## Description of Refuge

Quivira National Wildlife Refuge is located in south-central Kansas in northeast Stafford County (fig. 1). The refuge covers about 32 $mi^2$. It contains more than 30 control marshes and ponds ranging in size from 7 to 1,768 acres and about 21 mi of canals ranging in length from 0.1 to 7 mi (fig. 2).

The nearest climatic station to the refuge is at Hudson, 9 mi west of the refuge (fig. 1). The Hudson station records daily precipitation and temperature. The Sandyland Experiment Station (approximately 18 mi southwest of the refuge, fig. 1) is operated by the Kansas State University Agricultural Extension Office in Manhattan, Kansas, and has been recording hourly precipitation and temperature data for the last several years. Precipitation data also have been collected at the refuge headquarters (fig. 2) since 1996 and at the USGS streamflow-gaging station near Zenith (station 07142575, fig. 1).

## Surface Water

The major source of water to the refuge is Rattlesnake Creek. Hourly streamflow of Rattlesnake Creek is recorded at USGS streamflow-gaging stations near Zenith (station 07142575) and near Raymond (station 07142620). The long-term, lowest, and highest annual mean discharges for Rattlesnake Creek near Zenith for the 1973 through 1995 water years were 50.6, 6.59, and 186 $ft^3/s$, respectively (Putnam and others, 1996). For Rattlesnake Creek near Raymond, the long-term, lowest, and highest annual mean discharges for the same period were 48.4, 2.77, and 190 $ft^3/s$, respectively (Putnam and others, 1996).

In addition to the water supplied by Rattlesnake Creek, surface runoff to ponds generated by precipitation also plays an important role. The delineated drainage areas of the ponds are listed in table 1 (Marios Sophocleous, Kansas Geological Survey, written commun., 1997). These drainage areas were used for the calculation of overland surface runoff to the ponds. Surface runoff was estimated using the SCS curve-number method (Soil Conservation Service, 1985). SCS curve numbers for control ponds at Quivira National Wildlife Refuge also are listed in table 1. A description of the SCS curve-number method is found in the section "Estimation of Direct Overland Surface Runoff."

The refuge currently diverts water from the Little Salt Marsh (water unit 5), which is supplied by Rattlesnake Creek, into the main canal and into water units 7, 10A, 10B, 10C, and 11 (fig. 2). Water also flows from the Little Salt Marsh back into Rattlesnake Creek. Water in the creek flows north to water unit 24, where part of the water is diverted into the Darrynane Canal and into units 21 and 25. Some water flows into Rattlesnake Creek north of unit 24 and is transported to the west and north into the units north of County Road 484.

## Ground Water

The ponds in the north part of the refuge are within a ground-water discharge area. Table 2 shows the estimated monthly ground-water discharge from shallow aquifers to ponds for 1994 (Marios Sophocleous, Kansas Geological Survey, written commun., 1997). These values were estimated using a previous ground-water simulation done by Sophocleous and Perkins (1992) and the delineated drainage area of the ponds (table 1). The total ground-water discharge to ponds for 1994 was about 6,200 acre-ft.

## Physical Features of Control Ponds

Bottom elevations and full-pond capacities of control ponds are listed in table 3 (Megan Estep-Johnston, U.S. Fish and Wildlife, written commun., 1995). To express mathematically the elevation-volume-area relation of a pond, the pond storage was first divided into several water-depth zones. The number of zones

**Table 1.** Drainage area and SCS curve number for control ponds at Quivira National Wildlife Refuge, south-central Kansas

[SCS, U.S. Soil Conservation Service. Drainage areas and SCS curve numbers are from the Kansas Geological Survey (Marios Sophocleous, written commun., 1997)]

| Water-unit number (fig. 2) | Drainage area (acres) | SCS curve number |
|---|---|---|
| 5 | 1,890.7 | 74.020 |
| 7 | 140.8 | 42.680 |
| 10A | 84.9 | 48.397 |
| 10B | 201.4 | 47.575 |
| 10C | 84.5 | 47.575 |
| 11 | 341.4 | 47.575 |
| 14A | 149.9 | 71.217 |
| 14B | 124.9 | 71.693 |
| 14C | 59.5 | 33.552 |
| 16 | 180.0 | 40.753 |
| 20A | 179.4 | 58.461 |
| 20B | 116.4 | 73.101 |
| 21 | 60.0 | 76.469 |
| 22 | 82.5 | 45.543 |
| 23 | 43.8 | 46.615 |
| 24 | 259.9 | 51.636 |
| 25 | 226.7 | 55.711 |
| 26 | 194.7 | 67.952 |
| 28 | 228.4 | 38.659 |
| 29 | 78.9 | 60.995 |
| 30 | 69.0 | 61.968 |
| 40 | 207.2 | 42.018 |
| 48 | 305.6 | 73.499 |
| 49 | 137.4 | 71.000 |
| 55 | 582.8 | 72.250 |
| 57 | 257.5 | 69.910 |
| 58 | 186.7 | 62.831 |
| 61 | 258.8 | 71.481 |
| 62 | 90.4 | 52.546 |
| 63 | 201.4 | 71.000 |
| 75 | 5,621.7 | 69.583 |
| 78 | 635.9 | 70.544 |
| 80 | 187.1 | 70.544 |
| 81 | 620.9 | 70.544 |
| 83 | 149.4 | 70.544 |
| **Total** | **14,240.5** | |

**Table 2.** Ground-water discharge to selected control ponds at Quivira National Wildlife Refuge for 1994

[Data from Kansas Geological Survey (Marios Sophocleous, written commun., 1997). Negative numbers indicate that pond gains water from shallow aquifers ]

| Water-unit number (fig. 2) | Monthly ground-water discharge (acre-feet per day) | | | | | | | | | | | | Total discharge (acre-feet) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | January | February | March | April | May | June | July | August | September | October | November | December | |
| 5 | 1.30 | 1.36 | 1.42 | 1.51 | 1.46 | 1.49 | 1.60 | 1.51 | 1.52 | 1.60 | 1.57 | 1.59 | 545.00 |
| 10B | .16 | .17 | .17 | .18 | .18 | .18 | .19 | .18 | .19 | .19 | .18 | .18 | 65.46 |
| 14B | -.05 | -.05 | -.05 | -.05 | -.05 | -.05 | -.05 | -.05 | -.05 | -.05 | -.05 | -.05 | -18.00 |
| 14C | .09 | .09 | .09 | .09 | .09 | .09 | .10 | .10 | .10 | .10 | .10 | .09 | 34.33 |
| 20B | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | 15.16 |
| 24 | .56 | .57 | .56 | .59 | .56 | .56 | .60 | .56 | .55 | .57 | .54 | .54 | 206.02 |
| 25 | .24 | .23 | .23 | .24 | .23 | .23 | .24 | .22 | .22 | .22 | .21 | .21 | 82.66 |
| 26 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .03 | 13.70 |
| 40 | -.35 | -.32 | -.29 | -.27 | -.26 | -.24 | -.22 | -.21 | -.19 | -.18 | -.17 | -.16 | -86.52 |
| 49 | .03 | .04 | .04 | .05 | .05 | .05 | .07 | .06 | .06 | .07 | .07 | .07 | 20.33 |
| 58 | -.49 | -.49 | -.48 | -.49 | -.48 | -.47 | -.48 | -.47 | -.47 | -.46 | -.46 | -.46 | -173.42 |
| 61 | -.36 | -.35 | -.33 | -.32 | -.31 | -.30 | -.28 | -.29 | -.28 | -.26 | -.26 | -.25 | -109.43 |
| 62 | -.24 | -.23 | -.22 | -.21 | -.20 | -.19 | -.18 | -.18 | -.17 | -.17 | -.16 | -.16 | -70.29 |
| 63 | -.51 | -.50 | -.48 | -.46 | -.46 | -.44 | -.42 | -.43 | -.42 | -.40 | -.40 | -.39 | -161.82 |
| 75 | -14.20 | -14.05 | -13.95 | -14.02 | -13.86 | -13.70 | -13.81 | -13.55 | -13.45 | -13.40 | -13.28 | -13.22 | -5,006.67 |
| 78 | -1.67 | -1.65 | -1.64 | -1.65 | -1.63 | -1.61 | -1.62 | -1.59 | -1.58 | -1.57 | -1.56 | -1.55 | -587.37 |
| 80 | -.49 | -.49 | -.48 | -.48 | -.48 | -.47 | -.48 | -.47 | -.46 | -.46 | -.46 | -.46 | -172.86 |
| 81 | -1.85 | -1.83 | -1.82 | -1.83 | -1.81 | -1.78 | -1.80 | -1.76 | -1.75 | -1.74 | -1.73 | -1.72 | -651.70 |
| 83 | -.39 | -.39 | -.38 | -.39 | -.38 | -.38 | -.38 | -.37 | -.37 | -.37 | -.37 | -.36 | -137.91 |
| **Total** | **-18.14** | **-17.81** | **-17.53** | **-17.43** | **-17.27** | **-16.95** | **-16.84** | **-16.66** | **-16.47** | **-16.23** | **-16.15** | **-16.03** | **-6,193.33** |

**Table 3.** Full-pond elevations, water-surface areas, and capacities for selected control ponds at Quivira National Wildlife Refuge

[Data from U.S. Fish and Wildlife Service (Megan Estep-Johnston, written commun., 1995). BM, bench mark; ft, feet]

| Water-unit number (fig. 2) | Bottom elevation (feet above sea level) | Full-pond elevation, in feet above sea level (datum location) | | Full-pond surface area (acres) | Full-pond capacity (acre-feet) |
|---|---|---|---|---|---|
| 5 | 1,780 | 1,783 | (SPILLWAY) | 864 | 1,866 |
| 7 | 1,774 | 1,778 | (TOP OF STOPLOG SLOT) | 26 | 40 |
| 10A&10B | 1,774 | 1,779 | (TOP OF STOPLOG SLOT) | 64 | 145 |
| 10C | 1,772 | 1,774.4 | (TOP OF GAGE) | 11 | 13 |
| 11 | 1,754 | 1,774.9 | (SPILLWAY) | 90 | 338 |
| 14A | 1,772 | 1,778 | (SPILLWAY) | 87 | 196 |
| 14B | 1,772 | 1,776.7 | (SPILLWAY) | 65 | 96 |
| 14C | 1,774 | 1,777 | (14C[1] BM–0.67 ft) | 7 | 16 |
| 16 | 1,768 | 1,775 | (TOP OF STOPLOG SLOT) | 31 | 80 |
| 20A | 1,767 | 1,770.7 | (SPILLWAY) | 138 | 195 |
| 20B | 1,767 | 1,770.7 | (SPILLWAY) | 138 | 195 |
| 21 | 1,764 | 1,770 | (TOP OF STOPLOG SLOT) | 30 | 81 |
| 22 | 1,764 | 1,766 | (22A[1] BM–0.6 ft) | 10 | 13 |
| 23 | 1,762 | 1,764.3 | (TOP OF GAGE) | 9 | 15 |
| 24 | 1,765 | 1,769.4 | (SPILLWAY) | 31 | 35 |
| 25 | 1,762 | 1,768.4 | (TOP OF GAGE) | 94 | 296 |
| 26 | 1,758 | 1,762 | (SPILLWAY) | 59 | 111 |
| 28 | 1,762 | 1,768 | (28A[1] BM–0.86 ft) | 85 | 153 |
| 29 | 1,757 | 1,762 | (29C[1] BM–0.58 ft) | 61 | 91 |
| 30 | 1,756 | 1,759 | high water | 78 | 119 |
| 40 | 1,736 | 1,742.5 | (40B[1] BM–0.65 ft) | 32 | 66 |
| 48 | 1,750 | 1,754.4 | (SPILLWAY) | 89 | 113 |
| 49 | 1,750 | 1,754.2 | (SPILLWAY) | 95 | 159 |
| 57 | 1,740 | 1,743.5 | (57A[1] BM–0.6 ft) | 127 | 212 |
| 58 | 1,736 | 1,742 | (58B[1] BM–0.5 ft) | 99 | 251 |
| 61 | 1,740 | 1,745.5 | (62B[1] BM–0.58 ft) | 218 | 498 |
| 62 | 1,735 | 1,744 | (TOP OF STOPLOG SLOT) | 47 | 120 |
| 63 | 1,736 | 1,741.2 | (TOP OF GAGE) | 154 | 339 |
| 75 | 1,736 | 1,740.8 | (SPILLWAY) | 1,768 | 2,446 |
| | | | **Total** | **4,607** | **8,298** |

[1]Letters indicate structure names where water levels are measured.

were different for different ponds. For example, the number of zones for water unit 5 and water unit 24 were two and five, respectively (table 4). The bottom elevation (above sea level) of each zone was called the zonal elevation base ($Z_b$) for the corresponding zone. The elevation-volume-area relation of a pond was rep-

**Table 4.** Zonal elevation base and regression coefficients for elevation-volume-area relations of selected control ponds at Quivira National Wildlife Refuge

[Data from U.S. Fish and Wildlife Service (Megan Estep-Johnston, written commun., 1995)]

| Water-unit number (fig. 2) | Zone number | Zonal elevation base, $Z_b$ (feet above sea level) | Regression coefficients | | |
|---|---|---|---|---|---|
| | | | A1 | A2 | A3 |
| 5 | 1 | 1,780 | 1.0000 | 308.1200 | 110.4650 |
| | 2 | 1,782 | 1,059.0999 | 749.9802 | 56.9399 |
| 7 | 1 | 1,774 | 0 | .1800 | 1.6575 |
| | 2 | 1,776 | 6.9900 | 6.8100 | 4.7775 |
| | 3 | 1,778 | 39.7200 | 25.9200 | 7.2600 |
| 10A | 1 | 1,774 | 0 | 6.2900 | 4.5450 |
| | 2 | 1,776 | 30.7600 | 24.4700 | 3.6325 |
| | 3 | 1,778 | 94.2300 | 39.0000 | 12.2525 |
| 10B | 1 | 1,774 | 0 | 6.2900 | 4.5450 |
| | 2 | 1,776 | 30.7600 | 24.4700 | 3.6325 |
| | 3 | 1,778 | 94.2300 | 39.0000 | 12.2525 |
| 10C | 1 | 1,772 | 0 | 3.6700 | .6825 |
| | 2 | 1,774 | 10.0700 | 6.4000 | 5.3450 |
| 11 | 1 | 1,754 | 0 | .3000 | .5050 |
| | 2 | 1,756 | 2.6200 | 2.3200 | .5700 |
| | 3 | 1,758 | 9.5400 | 4.6000 | .5875 |
| | 4 | 1,760 | 21.0900 | 6.9500 | .6975 |
| | 5 | 1,762 | 37.7800 | 9.7400 | .7775 |
| | 6 | 1,764 | 60.3700 | 12.8500 | .7150 |
| | 7 | 1,766 | 88.9300 | 15.7100 | 1.5075 |
| | 8 | 1,768 | 126.3800 | 21.7400 | 2.3025 |
| | 9 | 1,770 | 179.0700 | 30.9500 | 3.0025 |
| | 10 | 1,772 | 252.9800 | 42.9600 | 1.2850 |
| 14A | 1 | 1,772 | 0 | 3.6700 | 1.7150 |
| | 2 | 1,774 | 14.2000 | 10.5300 | 7.9625 |
| | 3 | 1,776 | 67.1100 | 42.3800 | 11.0625 |
| 14B | 1 | 1,772 | 0 | .0800 | 2.5050 |
| | 2 | 1,774 | 10.1800 | 10.1000 | 9.5525 |
| | 3 | 1,776 | 68.5900 | 48.3100 | 12.1175 |
| 14C | 1 | 1,774 | 0 | .3000 | 2.6500 |
| | 2 | 1,775 | 2.9500 | 5.6000 | .3400 |
| 16 | 1 | 1,768 | 0 | .4700 | .9075 |
| | 2 | 1,770 | 4.5700 | 4.1000 | .9850 |
| | 3 | 1,772 | 16.7100 | 8.0400 | 4.5925 |
| | 4 | 1,774 | 51.1600 | 26.4100 | 2.3550 |

**Table 4.** Zonal elevation base and regression coefficients for elevation-volume-area relations of selected control ponds at Quivira National Wildlife Refuge—Continued

| Water-unit number (fig. 2) | Zone number | Zonal elevation base, $Z_b$ (feet above sea level) | Regression coefficients | | |
|---|---|---|---|---|---|
| | | | A1 | A2 | A3 |
| 20A | 1 | 1,767 | 0 | 0.8800 | 0.8400 |
| | 2 | 1,768 | 1.7200 | 2.5600 | 20.9950 |
| | 3 | 1,769 | 25.2750 | 44.5500 | 35.8750 |
| | 4 | 1,770 | 105.7000 | 116.3000 | 15.8000 |
| 20 B | 1 | 1,767 | 0 | .8800 | .8400 |
| | 2 | 1,768 | 1.7200 | 2.5600 | 20.9950 |
| | 3 | 1,769 | 25.2750 | 44.5500 | 35.8750 |
| | 4 | 1,770 | 105.7000 | 116.3000 | 15.8000 |
| 21 | 1 | 1,764 | 0 | 1.4200 | 1.3675 |
| | 2 | 1,766 | 8.3100 | 6.8900 | 2.8300 |
| 22 | 1 | 1,764 | 0 | 3.4700 | 1.6350 |
| | 2 | 1,766 | 13.4800 | 10.0100 | 1.2825 |
| 23 | 1 | 1,762 | 0 | 3.7900 | 1.1625 |
| | 2 | 1,764 | 12.2300 | 8.4400 | 1.0625 |
| 24 | 1 | 1,765 | 0 | .1600 | .3700 |
| | 2 | 1,766 | .5300 | .9000 | .6200 |
| | 3 | 1,767 | 2.0500 | 2.1400 | 3.5750 |
| | 4 | 1,768 | 7.7650 | 9.2900 | 6.8600 |
| | 5 | 1,769 | 23.9150 | 23.0100 | 10.3950 |
| 25 | 1 | 1,762 | 0 | .4600 | 2.3875 |
| | 2 | 1,764 | 10.4700 | 10.0100 | 15.9925 |
| | 3 | 1,766 | 94.4600 | 73.9800 | 4.1675 |
| 26 | 1 | 1,758 | 0 | 2.4800 | 5.4875 |
| | 2 | 1,760 | 26.9100 | 24.4300 | 8.7050 |
| 28 | 1 | 1,762 | 0 | .0400 | .8125 |
| | 2 | 1,764 | 3.3300 | 3.2900 | 6.7775 |
| | 3 | 1,766 | 37.0200 | 30.4000 | 13.7275 |
| 29 | 1 | 1,757 | 0 | .0600 | .2650 |
| | 2 | 1,758 | .3250 | .5900 | 1.6500 |
| | 3 | 1,759 | 2.5650 | 3.8900 | 5.2800 |
| | 4 | 1,760 | 11.7350 | 14.4500 | 13.6450 |
| | 5 | 1,761 | 39.8300 | 41.7400 | 9.4300 |
| | 6 | 1,762 | 91.0000 | 60.6000 | 12.8350 |
| 30 | 1 | 1,756 | 0 | 1.6200 | 12.7325 |
| 40 | 1 | 1,736 | 0 | .1900 | .2350 |

**Table 4.** Zonal elevation base and regression coefficients for elevation-volume-area relations of selected control ponds at Quivira National Wildlife Refuge—Continued

| Water-unit number (fig. 2) | Zone number | Zonal elevation base, $Z_b$ (feet above sea level) | Regression coefficients A1 | A2 | A3 |
|---|---|---|---|---|---|
| 40 | 2 | 1,738 | 1.4400 | 1.2500 | 2.2725 |
| | 3 | 1,740 | 13.0300 | 10.3400 | 4.3375 |
| | 4 | 1,742 | 51.0600 | 27.6900 | 4.4375 |
| 48 | 1 | 1,750 | 0 | .2700 | .4750 |
| | 2 | 1,751 | .7450 | 1.2200 | 2.1400 |
| | 3 | 1,752 | 4.1050 | 5.5000 | 14.3300 |
| | 4 | 1,753 | 23.9350 | 34.1600 | 21.3150 |
| | 5 | 1,754 | 79.4100 | 76.7900 | 15.6400 |
| 49 | 1 | 1,750 | 0 | .4600 | 1.6450 |
| | 2 | 1,751 | 2.1050 | 3.7500 | 11.2350 |
| | 3 | 1,752 | 17.0900 | 26.2200 | 19.3450 |
| | 4 | 1,753 | 62.6550 | 64.9100 | 12.4750 |
| 57 | 1 | 1,740 | 0 | 5.5300 | 14.5825 |
| | 2 | 1,742 | 69.3900 | 63.8600 | 20.9075 |
| 58 | 1 | 1,736 | 0 | 2.2800 | 3.9775 |
| | 2 | 1,738 | 20.4700 | 18.1900 | 9.5700 |
| | 3 | 1,740 | 95.1300 | 56.4700 | 10.6425 |
| 61 | 1 | 1,740 | 0 | 10.2900 | 6.9975 |
| | 2 | 1,742 | 48.5700 | 38.2800 | 25.7175 |
| 62 | 1 | 1,735 | 0 | .0100 | .1000 |
| | 2 | 1,736 | .1100 | .2100 | .1900 |
| | 3 | 1,737 | .5100 | .5900 | .3150 |
| | 4 | 1,738 | 1.4150 | 1.2200 | 1.0550 |
| | 5 | 1,739 | 3.6900 | 3.3300 | 2.4350 |
| | 6 | 1,740 | 9.4550 | 8.2000 | 5.4750 |
| | 7 | 1,741 | 23.1300 | 19.1500 | 4.0800 |
| | 8 | 1,742 | 46.6360 | 27.3100 | 4.8150 |
| 63 | 1 | 1,736 | 0 | .3800 | 6.0175 |
| | 2 | 1,738 | 24.8300 | 24.4500 | 24.6250 |
| | 3 | 1,740 | 172.2300 | 122.9500 | 13.0675 |
| 75 | 1 | 1,736 | 0 | .3100 | 96.7650 |
| | 2 | 1,737 | 97.0750 | 193.8400 | 84.3200 |
| | 3 | 1,738 | 375.2350 | 362.4800 | 76.1350 |
| | 4 | 1,739 | 813.8500 | 514.7500 | 113.3850 |
| | 5 | 1,740 | 1,441.9851 | 741.5198 | 641.6354 |

resented by stepwise regression equations in terms of zonal water depth (Megan, Estep-Johnston, U.S. Fish and Wildlife, written commun., 1995):

$$V = A1 + A2\ X + A3\ X^2, \text{ and} \qquad (1)$$

$$A = A2 + (A3 + A3)\ X, \qquad (2)$$

where $X$ is the zonal water depth and is equal to the difference between pond water-surface elevation ($Z$), in feet, and the corresponding zonal elevation base ($Z_b$), in feet; that is, $X = Z - Z_b$. $A1$, $A2$, and $A3$ are regression coefficients, volume ($V$) is in acre-feet, and water-surface area ($A$) is in acres.

Table 4 summarizes the zonal elevation bases and corresponding regression coefficients of selected control ponds (Megan Estep-Johnston, U.S. Fish and Wildlife Service, written commun. 1995).

As an illustration of the use of these regression equations, consider water unit 5 as an example. Let the water-surface elevation $Z$ be 1,782.5 ft. From table 4, the water surface is located in zone 2 because the water-surface elevation of 1,782.5 ft is higher than the zonal elevation base ($Z_b$) of 1,782 ft. Therefore, the zonal water depth ($X$) is 1,782.5 - 1,782.0 = 0.5 ft with the regression coefficients ($A1$, $A2$, and $A3$) of 1,059.0999, 749.9802, and 56.9399, respectively. Using equations 1 and 2, the corresponding water volume ($V$) and water-surface area ($A$) are 1,448.32 acre-ft and 806.92 acres, respectively. However, if the water-surface elevation $Z$ is 1,781.0 ft, the corresponding zone number now is 1 with a zonal elevation base ($Z_b$) of 1,780 ft and regression coefficients ($A1$, $A2$, and $A3$) of 1.0000, 308.1200, and 110.4650, respectively. Therefore, the corresponding water volume ($V$) and water-surface area ($A$) are 419.59 acre-ft and 529.05 acres, respectively. Figure 3 shows the elevation-volume-area curves for Little Salt Marsh (water unit 5, fig. 2) using equations 1 and 2.

# LINEAR-NETWORK FLOW MODEL

The optimal operation of the control ponds at the Quivira National Wildlife Refuge can be formulated mathematically as a linear-network flow problem (Jian, 1988; Yu and others, 1989). In this section, the mathematical formulation of a linear-network flow model (Jian, 1988) is modified and expanded. The concepts of a rule curve for a pond and the zoning of pond storage and canal flow are introduced in this section. These two concepts are bases for formulating a network flow problem. The operating policy for a pond system in terms of priority and cost-penalty coefficients is also discussed. By combining the concepts of a rule curve and zoning and the operating policy, the problem of the operation of pond storage and flow routing can be formulated as a minimum-cost network flow problem, which is a typical topic in network flow analysis.

## Network Representation of Flow Systems

To apply network flow analysis to the Quivira National Wildlife Refuge, the flow systems shown in figure 2 were conceptually represented by a network of nodes and arcs (fig. 4). The network was comprised of 67 nodes, of which 34 nodes are pond nodes (oval shape in fig. 4). Water unit 55 was shown as an oval in figure 4 and treated as a canal node because there was no pond information for that unit. Ninety-seven (97) arcs were used to represent canals or waterways on the refuge. Water unit 34 in figure 4 is a proposed pond for future use and is not currently (1997) in operation.

## Rule-Curve and Zoning Concepts

A rule curve designated a target water level in a control pond. Using the zoning concept, a control pond at the refuge was divided into four storage zones—extended upper zone, upper zone, lower zone, and inactive zone—and the rule curve was set at the top of the lower zone (fig. 5). The extended upper zone was used during periods of flood. The upper zone and lower zone were called conservation zones and were used to represent normal use. The inactive zone represented the storage area filled up by sediment accumulation. The selection of the number of zones in a particular pond was based on management needs. For example, if the objective of management was to maximize water yields, the target water level (that is, rule curve) was set at the highest elevation of a pond so that high water levels could be maintained after satisfying downstream flow requirements and water demands.

Similarly, flow in a canal was also divided into an upper zone (that is, above-normal zone), a normal zone, and a lower zone (that is, below-normal zone) as shown in figure 6. The selection of the number of canal flow zones was also dependent on management needs.

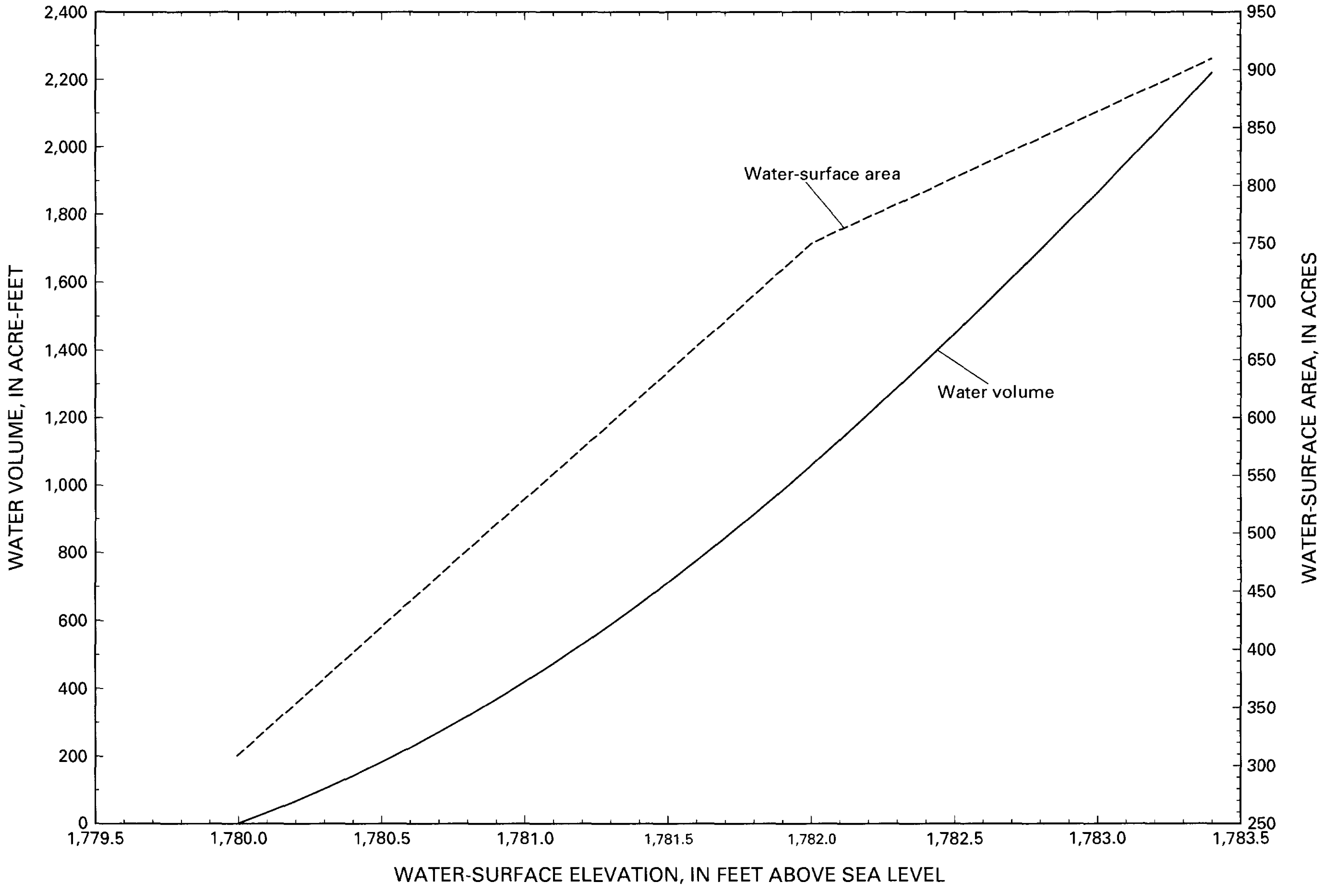



**Figure 3.** Relations of water-surface elevation, water volume, and water-surface area for Little Salt Marsh (water unit 5, fig. 2).

Because flows in canals at the refuge are not regulated, one flow zone (normal) was used in the flow model development. In model simulations, canal flows were maintained in the normal flow zone as long as possible.

## Operating Policy

Under ideal inflow conditions, all pond levels would be maintained at the target water levels (rule curves), and all canal flows would be maintained in the normal flow ranges in addition to satisfying water-management requirements such as minimum desirable streamflow (Kansas Water Office, written commun., 1996). In reality, ideal inflow conditions rarely occur. If a pond water level was higher or lower than its rule curve, a "cost" or "penalty" was assessed to the water storage or depletion deviation from the rule curve. The penalty depended on the amount of water deviation from the target level and the penalty coefficient (cost per unit water deviation from the target level). A penalty was also assessed to canal flows. In other words, penalty coefficients were assigned to each storage zone of a pond and each flow zone in a canal to assess penalty.

Different penalty coefficients were assigned according to management priorities related to each storage zone of a pond. Penalty coefficients for canal flows were specified in a similar way. Higher penalty coefficients were assigned to the extended upper zone and inactive zone, and smaller penalty coefficients were assigned to the conservation zone (the lower zone and the upper zone) because water levels needed to be maintained in the conservation zone for normal use. The penalty coefficient in the normal-flow zone in a canal was generally zero or less than the penalty coefficient of the pond conservation zone. A higher penalty coefficient was assigned for violation of normal flow range; that is, the higher values of penalty coefficients were assigned to the upper and lower flow zones.

To optimally operate the canal and pond system at the refuge, it was necessary for some interpond rela-

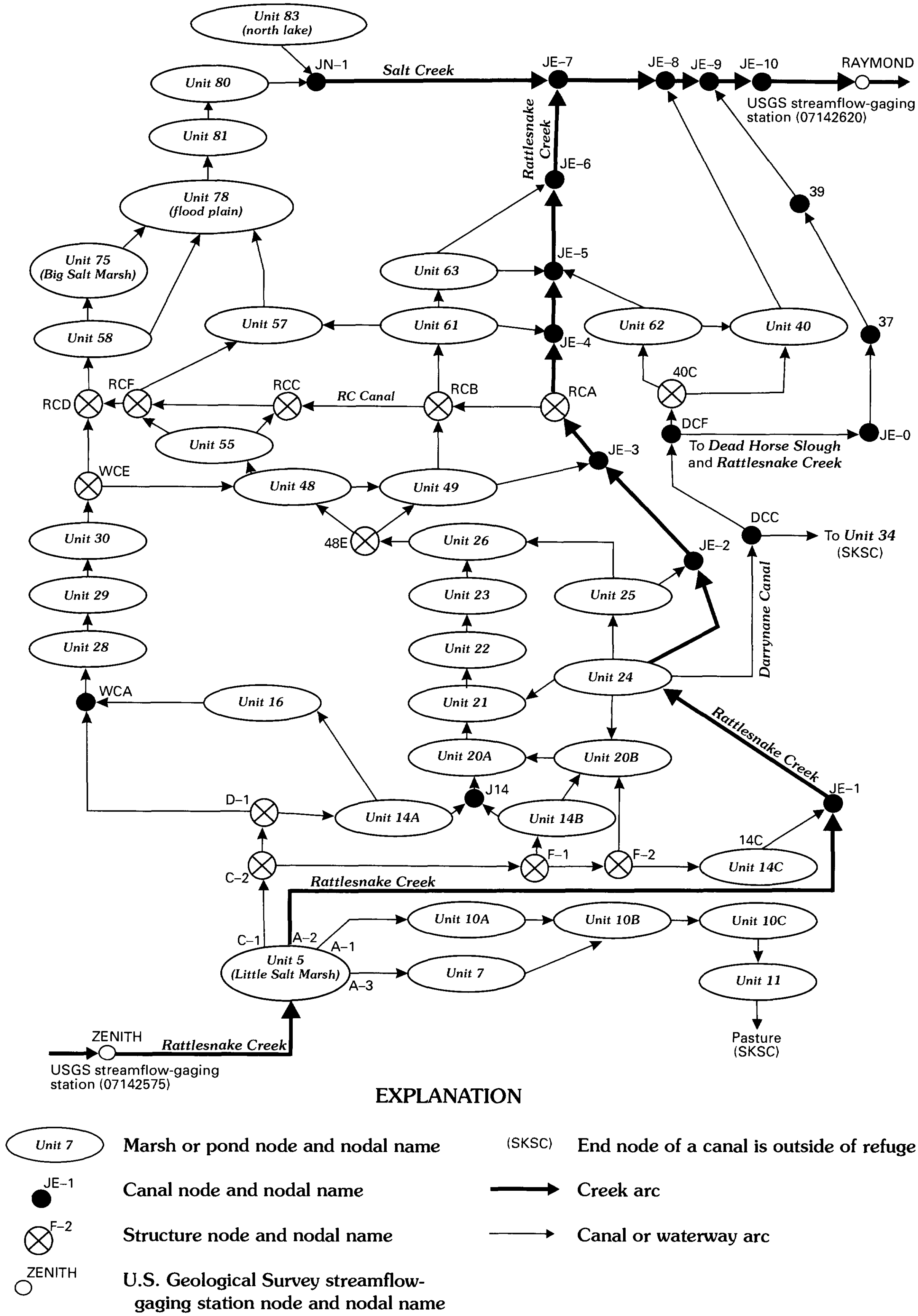



**Figure 4.** Network representation of flow systems at Quivira National Wildlife Refuge.

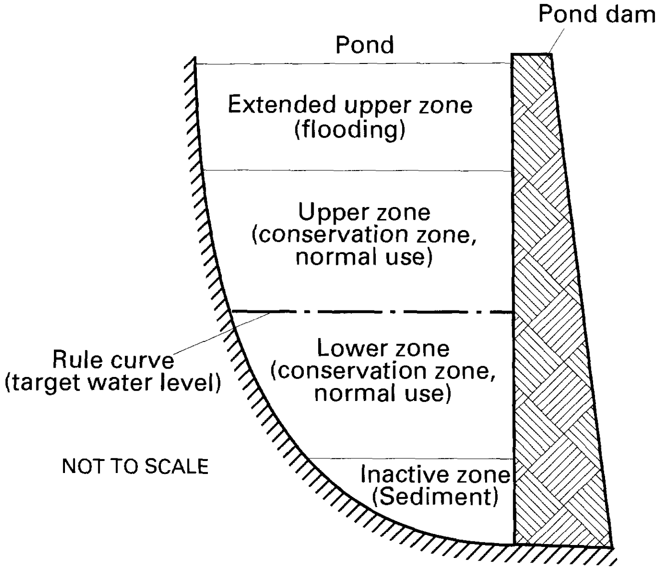



**Figure 5.** Concepts of rule curve and pond zoning.

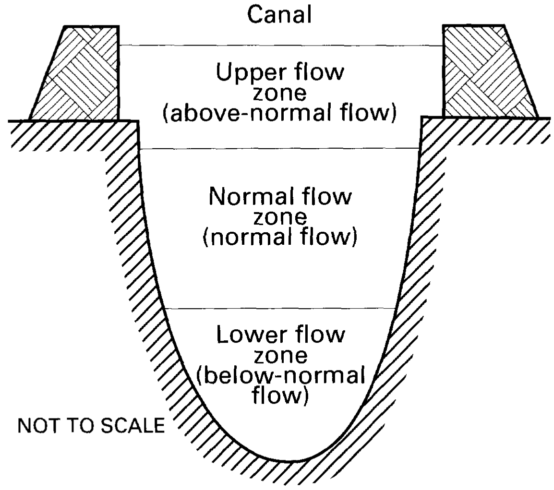



**Figure 6.** Concepts of canal-flow zoning.

tions to be incorporated into the flow model. One of the relations was priority ranking of the ponds. Ponds were ranked according to some specified criteria by assigning different penalty coefficients to storage zones. The lowest priority pond was assigned the smallest penalty coefficient for the same-purpose storage zone; higher priority ponds had higher penalty coefficients. Using this relation, violation of the rule curve first occurred in the pond with the lowest priority. It was common for rule-curve violations to occur first in the downstream ponds rather than the upstream ones. This procedure minimized unnecessary spilling at the most downstream pond in the event of high lateral flows; that is, flows that did not enter the system through upstream ponds. The optimal operation of the canal and pond system minimized the total penalty assessed on the deviations of pond storage from the rule curve and of canal flows from specified normal flows.

## Mathematical Expression of Linear-Network Flow Model

The flow network consisted of nodes and directed arcs. A node represented a location where the computation of the water budget was needed, such as at ponds and at canals where diversion of water occurred. An arc represented a stream or a canal along which water moved from one location to another. An arc was also used to represent a storage deviation of a pond or canal, and other additional contributions such as evaporation, seepage, and runoff.

The linear-network flow model was expressed mathematically as a linear programming problem of minimizing the total cost or penalty as follows:

$$\text{Minimize} \sum_i \sum_j C_{ij} Q_{ij} \text{ for all (i,j) arcs,} \tag{3}$$

$$\text{subject to} \sum_j Q_{ji} - \sum_j Q_{ij} = 0 \text{ for all i nodes, and} \tag{4}$$

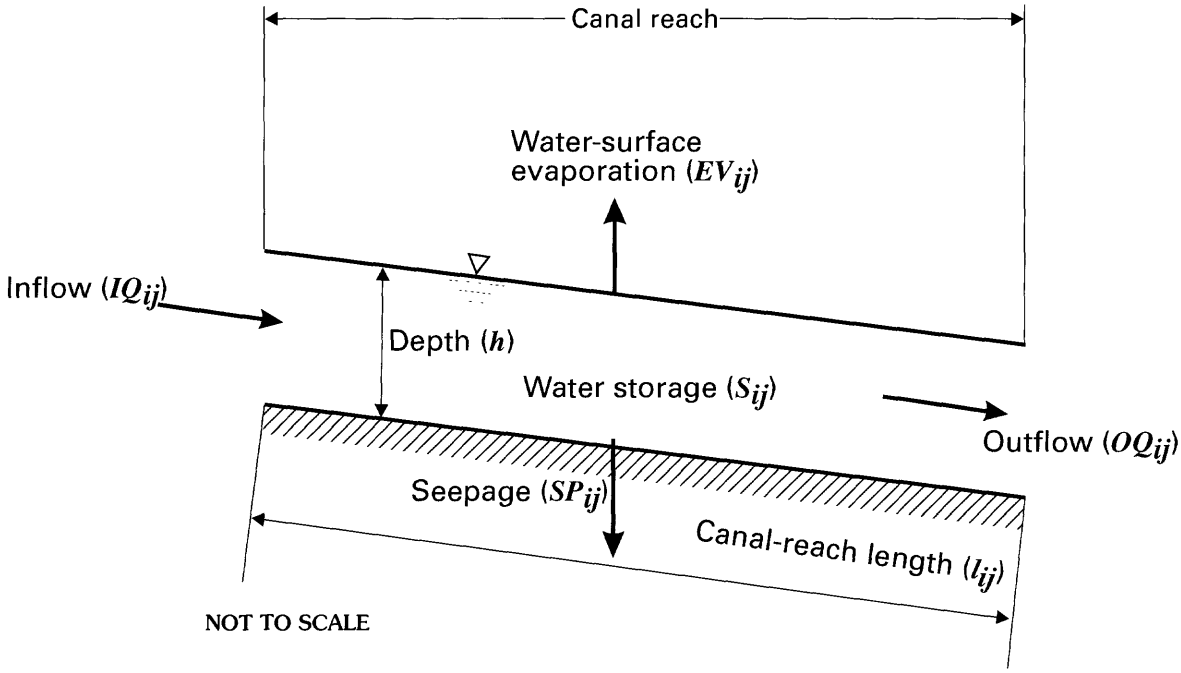
$$L_{ij} \leq Q_{ij} \leq U_{ij} \text{ for all (i,j) arcs,} \tag{5}$$

where

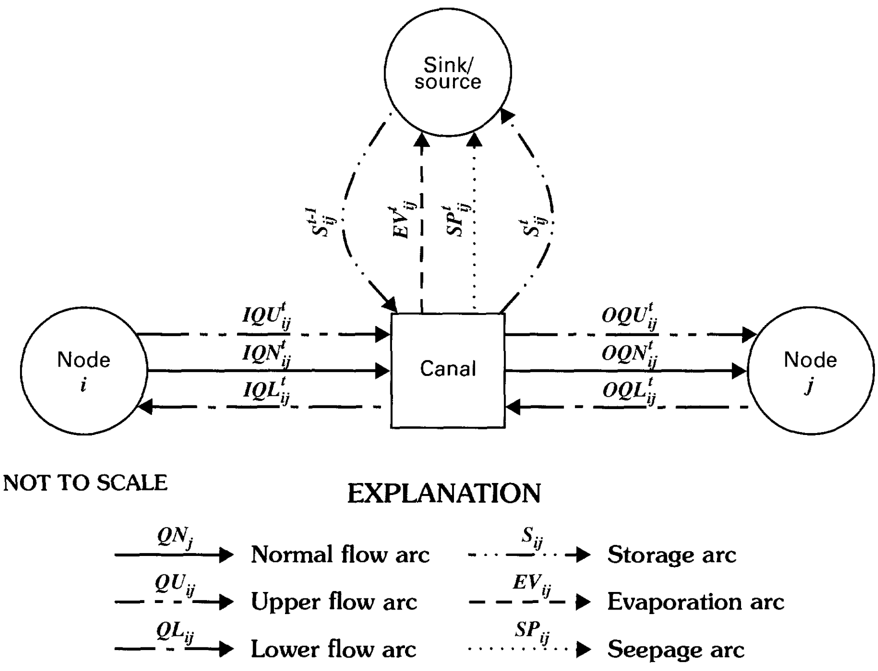
$Q_{ij}$ = flow in arc $(i,j)$ from node $i$ to node $j$;
$C_{ij}$ = cost per unit flow in arc $(i,j)$, also called the penalty coefficient;
$L_{ij}$ = the lower flow boundary in any arc $(i,j)$; and
$U_{ij}$ = the upper flow boundary in any arc $(i,j)$.

Any flow (choice of the $Q_{ij}$'s) satisfying the constraints in equation 4 was called a conserving flow, accounting for mass conservation at the nodes. A conserving flow that satisfied the remaining constraints in equation 5 was a feasible flow (solution).

The objective of equation 3 for the operation of canals and control ponds was to minimize the total cost due to deviations from specified rule curves and canal flows. Equations 3–5 needed to incorporate the con-




*A.* **Water balance in a canal reach**




*B.* **Arc-node representation of canal-flow routing**

**Figure 7.** *(A)* Water balance in a canal reach and *(B)* arc-node representation of canal-flow routing.

cepts of rule curve and zoning to obtain the appropriate water-balance equation for any pond or canal reach.

### Canal Routing

Considering a canal reach $(i,j)$ with a length $l_{ij}$, the water balance in the canal (fig. 7A) during time period $(t)$ was expressed as:

$$IQ_{ij}^t - OQ_{ij}^t - (SP_{ij}^t + EV_{ij}^t) = S_{ij}^t - S_{ij}^{t-1}, \qquad (6)$$

where

$IQ_{ij}^t$ = inflow to a canal reach $(i,j)$ during time period $t$;

$OQ_{ij}^t$ = outflow from a canal reach during time period $t$;

$SP_{ij}^t$ = seepage along a canal reach during time period $t$. The amount of seepage depended on canal flow and hydraulic parameters. See section "Estimation of Canal-Flow Transmission Losses" later in this report.

$EV_{ij}^t$ = canal surface-water evaporation, which was estimated by:

$$EV_{ij}^t = e_{ij}^t A_{ij}^t, \qquad (7)$$

in which $e_{ij}^t$ was the water-surface evaporation coefficient for time period $t$, and $A_{ij}^t$ was the water-surface area given by $A_{ij}^t = l_{ij}\, b_{ij}$, where $l_{ij}$ and $b_{ij}$ were the canal length and the width of the water surface, respectively.

$S_{ij}^t$ = water storage in a canal reach at the end of time period $t$. Canal storage $S_{ij}^t$ depended on canal inflow and outflow, and canal hydrologic parameters. See section "Estimation of Canal Water Storage" using Muskingum's method (McCuen, 1989) later in this report.

$S_{ij}^{t-1}$ = water storage in a canal reach at the beginning of time period $t$.

In the concept of canal-flow zoning, canal flow may be in the normal, upper, or lower flow zone. The actual flow in a canal was denoted by $Q$ and the normal flow by $QN$. The range of flow in the normal flow zone was $0 \le \underline{QN} \le QN \le \overline{QN}$, where $\underline{QN}$ and $\overline{QN}$ were the lower and upper boundaries of the normal flow range, respectively. If canal flow $(Q)$ was in the upper flow zone, then the upper flow $(QU)$ was defined as the flow deviation from the upper boundary of the normal flow range; that is, $QU = Q - \overline{QN}$, and $0 \le QU \le \overline{QU}$, where $\overline{QU}$ was the magnitude of the upper flow zone. In this case, $QN = \overline{QN}$, and $Q = QN + QU$. If canal flow $(Q)$ was in the lower flow zone, then the lower flow $(QL)$ was defined as the flow deviation from the lower boundary of the normal flow zone; that is, $QL = \underline{QN} - Q$, and $0 \le QL \le \overline{QL}$, where $\overline{QL}$ was the magnitude of the lower flow zone. In this case, $QN = \underline{QN}$, and $Q = QN - QL$. Therefore, the actual flow $Q$ in a canal reach could be expressed as normal flow $(QN)$, plus the upper flow $(QU)$, and minus the lower flow $(QL)$; that is,

$$Q = QN + QU - QL. \qquad (8)$$

If canal flow was in the normal flow zone, both $QU$ and $QL$ were equal to zero. If canal flow was in the upper zone, $QL$ was zero. On the other hand, if canal flow was in the lower flow zone, $QU$ was zero. Therefore, equation 8 represented all flow states in a canal.

Substituting equation 8 into equation 6 gave the canal water-balance equation as follows:

$$(IQN_{ij}^t + IQU_{ij}^t - IQL_{ij}^t) - (OQN_{ij}^t + OQU_{ij}^t - OQL_{ij}^t) - (SP_{ij}^t + EV_{ij}^t) = S_{ij}^t - S_{ij}^{t-1}, \qquad (9)$$

where

$IQN_{ij}^t$ = normal inflow $(0 \le \underline{QN}_{ij} \le IQN_{ij}^t \le \overline{QN}_{ij})$,

$IQU_{ij}^t$ = upper inflow $(0 \le IQU_{ij}^t \le \overline{QU}_{ij})$,

$IQL_{ij}^t$ = lower inflow $(0 \le IQL_{ij}^t \le \overline{QL}_{ij})$,

$OQN_{ij}^t$ = normal outflow $(0 \le \underline{QN}_{ij} \le OQN_{ij}^t \le \overline{QN}_{ij})$,

$OQU_{ij}^t$ = upper outflow $(0 \le OQU_{ij}^t \le \overline{QU}_{ij})$, and

$OQL_{ij}^t$ = lower outflow $(0 \le OQL_{ij}^t \le \overline{QL}_{ij})$.

Each item in equation 9 could be represented by flow through a distinct arc (fig. 7B). If there was no water loss or storage change along the arc $(i,j)$, then the actual flow between two neighboring nodes $i$ and $j$ was simplified as:

$$IQ_{ij}^t = OQ_{ij}^t = IQN_{ij}^t + IQU_{ij}^t - IQL_{ij}^t. \qquad (10)$$

## Pond-Storage Routing

Using the concepts of rule curve and zoning of a pond, the actual storage of a pond $i$ at time $t$, $S_i^t$, was represented as the sum of the rule-curve storage, $RC_i^t$, plus the storage deviation from the rule-curve storage, $D_i^t$; that is,

$$S_i^t = RC_i^t + D_i^t, \qquad (11)$$

where

subscript $i$ was the pond node index, and
superscript $t$ was the time period index.

In the concept of pond zoning, the storage deviation, $D_i^t$, was expressed as:

$$D_i^t = SU_i^t - SL_i^t, \qquad (12)$$

where

$SU_i^t$ = the actual storage deviation above the rule curve in the upper zone; that is, $SU_i^t = S_i^t - RC_i^t$, and $0 \le SU_i^t \le \overline{SU}_i$, where $\overline{SU}_i$ was the total capacity of the upper zone in control pond $i$. If there were $m$ upper zones, the upper deviation $SU_i^t$ was calculated by:

$$SU_i^t = \sum_{k=1}^{m} SU_{i,k}^t, \qquad (13)$$

where

$SU^t_{i,k}$ was the water storage in the upper zone $k$.

$SL_i^t$ = the actual storage deviation from the rule curve in the lower zone; that is, $SL_i^t = RC_i^t - S_i^t$, and $0 \le SL_i^t \le \overline{SL}_i$, where $\overline{SL}_i$ was the total capacity of the lower zone in pond $i$. If there were $n$ lower zones, the lower deviation $SL_i^t$ was calculated by:

$$SL_i^t = \sum_{k=1}^{n} SL_{i,k}^t, \qquad (14)$$

where

$SL^t_{i,k}$ was the water storage in the lower zone $k$.

Only one of the two terms on the right side of equation 12 could be nonzero. In other words, if the pond water level was in the upper zone, then $SL_i^t = 0$. On the other hand, if the pond water level was in the lower zone, then $SU_i^t = 0$.

Substituting equation 12 into equation 11 gave:

$$S_i^t = RC_i^t + SU_i^t - SL_i^t. \qquad (15)$$

The water-balance equation (equation 3) for pond node $i$ could then be rewritten as:

$$\sum_j OQ_{ji}^t - \sum_j IQ_{ij}^t + I_i^t + RN_i^t - EV_i^t - SP_i^t - W_i^t = S_i^t - S_i^{t-1}, \qquad (16)$$

where

$OQ_{ji}^t$ = canal inflow from the upstream node $j$ during time period $t$.

$IQ_{ij}^t$ = water release to downstream node $j$ during time period $t$. Release was determined in terms of a downstream flow requirement, pond stage, and outlet control structure. See section "Flow Through Hydraulic Structures" later in this report.

$I_i^t$ = local net inflow to pond $i$ during time period $t$.

$RN_i^t$ = precipitation falling onto the water surface ($P_i^t$) plus the direct overland surface runoff ($RF_i^t$) during time period $t$, given by:

$$RN_i^t = P_i^t + RF_i^t. \qquad (17)$$

Precipitation falling onto the water surface, in acre-feet, $P_i^t$, was estimated by:

$$P_i^t = 0.0833\, r_i^t A_i^t \quad , \qquad (18)$$

in which $r_i^t$ was rainfall during time period $t$, in inches, and $A_i^t$ was the water-surface area, in acres. Direct overland surface runoff ($RF_i^t$) was estimated using a SCS curve-number method (Soil Conservation Service, 1985). See section "Estimation of Direct Overland Surface Runoff" later in this report.

$EV_i^t$ = water-surface evaporation of pond $i$ during time period $t$ was estimated by:

$$EV_i^t = e_i^t A_i^t \quad , \qquad (19)$$

in which $e_i^t$ was the water-surface evaporation coefficient for time period $t$, and $A_i^t$ was the pond water-surface area at the beginning of time period $t$. If the evaporation rate $a_i^t$ was in inches per day and the water-surface area $A_i^t$ was in acres, then total surface evaporation $EV_i^t$, in acre-feet, during time period $t$ with time length of $\Delta t$ days was calculated as follows:

$$EV_i^t = 0.0833\ a_i^t A_i^t\ \Delta t. \tag{20}$$

$SP_i^t$ = seepage through the pond bottom, in acre-feet. Seepage was estimated using Darcy's equation:

$$SP_i^t = K_i \frac{Zsw_i^t - Zgw_i^t}{d_i} A_i^t \Delta t, \tag{21}$$

in which $K_i$ was the bottom hydraulic conductivity of pond $i$, in feet per day; $Zsw_i^t$ was the surface-water elevation, in feet above sea level; $Zgw_i^t$ was the ground-water elevation below the pond bottom, in feet above sea level; and $d_i$ was the pond-bottom thickness, in feet.

$W_i^t$= water withdrawal during time period $t$, in acre-feet.

$S_i^t$ = pond storage at the end of time period $t$, in acre-feet.

$S_i^{t-1}$ = pond storage at the beginning of time period $t$, in acre-feet.

Substituting equations 8 and 15 into equation 16 gave:

$$\sum_j \left( OQN_{ji}^t + OQU_{ji}^t - OQL_{ji}^t \right) - \sum_j \left( IQN_{ij}^t + IQU_{ij}^t - IQL_{ij}^t \right) + \tag{22}$$
$$I_i^t + RN_i^t - SP_i^t - W_i^t = \left( RC_i^t + SU_i^t \right) - S_i^{t-1}.$$

Rearranging equation 22 gave:

$$\sum_j \left( OQN_{ji}^t + OQU_{ji}^t - OQL_{ji}^t \right) - \tag{23}$$
$$\sum_J \left( IQN_{ij}^t + IQU_{ij}^t - IQL_{ij}^t \right) - SU_i^t + SL_i^t$$
$$+ (S_i^{t-1} + I_i^t + RN_i^t - RC_i^t - SP_i^t - EV_i^t - W_i^t) = 0.$$

At the beginning of time $t$, the values of $S_i^{t-1}$, $I_i^t$, $RN_i^t$, $RC_i^t$, $SP_i^t$, $EV_i^t$, and $W_i^t$ were known or could be estimated using previous time-period data. If $NV_i^t = S_i^{t-1} + I_i^t + RN_i^t - RC_i^t - SP_i^t - EV_i^t - W_i^t$, the pond water-balance equation became:

$$\sum_j \left( OQN_{ji}^t + OQU_{ji}^t - OQL_{ji}^t \right) - \tag{24}$$
$$\sum_j \left( IQN_{ij}^t + IQU_{ij}^t - IQL_{ij}^t \right) - SU_i^t + SL_i^t + NV_i^t = 0.$$

Each term in equation 24 was represented by flow through a distinct arc in the linear-network flow model. Among these arcs, the term $NV_i^t$ was simply called a net-value arc (*NV*). Upper storage deviation arcs (*SU*), lower storage deviation arcs (*SL*), and *NV* arcs were connected to a sink/source node (fig. 8). The direction of *SU* arcs was from node $i$ to the sink/source node. The direction of *SL* arcs was from the sink/source node to node $i$. The direction of *NV* arcs depended on the sign of the value of $NV_i^t$. If the value of $NV_i^t$ was positive, the direction of the *NV* arc was from the sink/source node to the pond node $i$, and the reverse was true if the value of $NV_i^t$ was negative (fig. 8).

### General Node

A general node was designated where the calculation of water balance was needed (for example, at joints of canals). The difference between a pond node and a general node was that there was no water storage associated with a general node. The water balance at general node $i$ during time $t$ was given by:

$$\sum_j OQ_{ji}^t - \sum_j IQ_{ij}^t + I_i^t - W_i^t = 0, \tag{25}$$

where

$OQ_{ji}^t$ = inflow from upstream node $j$ to node $i$ during time period $t$;

$IQ_{ij}^t$ = outflow from a general node $i$ to the downstream node $j$ during time period $t$;

$I_i^t$ = local net incremental inflow to node $i$, such as surface runoff; and

$W_i^t$ = water withdrawal at node $i$ during time $t$. $W_i^t$ was expressed as follows:

$$W_i^t = TR_i^t - DW_i^t, \tag{26}$$

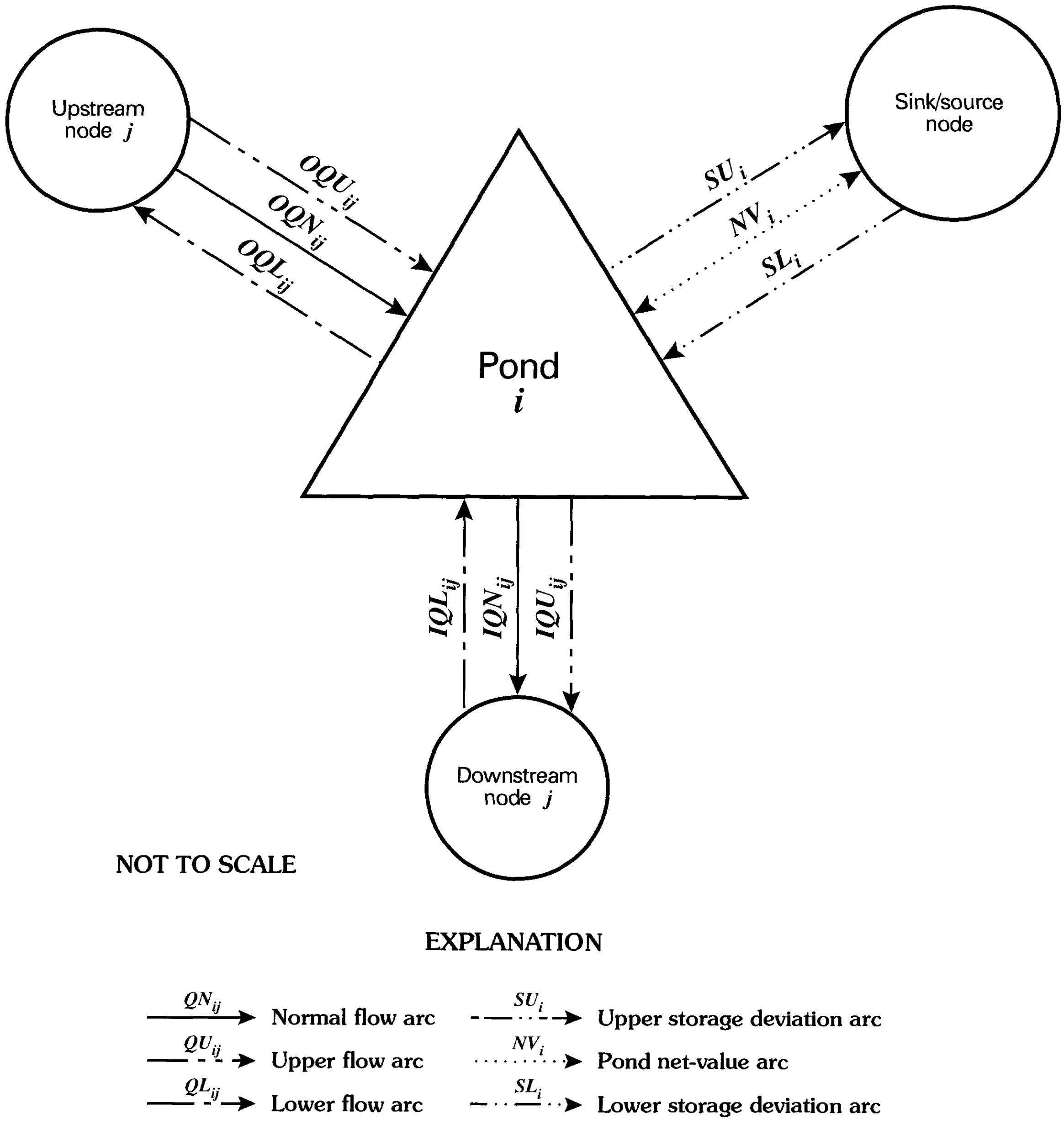



**Figure 8.** Arc-node representation for pond-storage routing.

in which $TR_i^t$ was the target water-withdrawal from node $i$ during time period $t$, and $DW_i^t$ was the water withdrawal deviation for node $i$ during time period $t$, $0 \le DW_i^t \le TR_i^t$.

Substituting equations 8 and 26 into equation 25 gave:

$$\sum_j \left(OQN_{ji}^t + OQU_{ji}^t - OQL_{ji}^t\right) - \sum_j \left(IQN_{ij}^t + IQU_{ij}^t - IQL_{ij}^t\right) + I_i^t - \left(TR_i^t - DW_i^t\right) = 0 \,. \quad (27)$$

Equation 27 is consistent with equation 4, and each term of equation 27 was represented by flow through a distinct arc (fig. 9). Among flows through these arcs, flows $I_i^t$ and $TR_i^t$ were known at the beginning of time $t$. The direction of the $I_i^t$ arc depended on the sign of the value $I_i^t$. If it was positive, then the arc was directed toward the water demand node from the sink/source node. The reverse was true for a negative $I_i^t$. Because $I_i^t$ and $TR_i^t$ were known, the penalty coefficients of $I_i^t$ and $TR_i^t$ arcs were assigned to be zero.

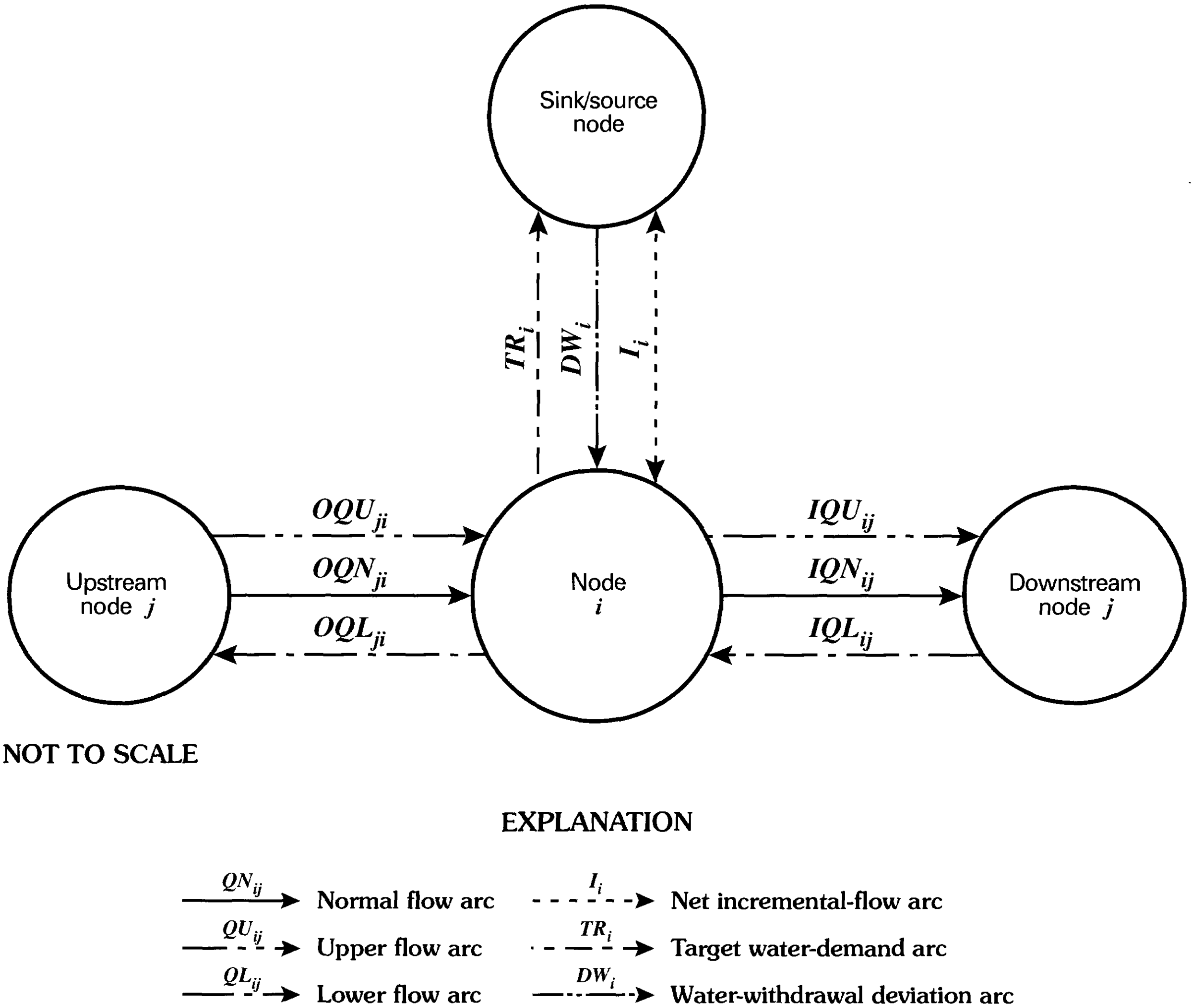



**Figure 9.** Arc-node representation for a general node.

### Sink/Source Node

The sink/source node mentioned in previous sections was an introduced node that made it possible to form a closed-loop network. As a sink, this node accounted for: (1) flows from canal water loss (seepage and evaporation) and final water storage (fig. 7*B*), (2) flows from the storage deviation above the rule curve (fig. 8), (3) canal flows at the downstream end of the system, and (4) water withdrawal from a node (fig. 9). As a source, the sink/source node accounted for: (1) flows for the canal initial storages (fig. 7*B*), (2) flows for the storage deviations below the rule curves (fig. 8), (3) net inflows to ponds (fig. 8), and (4) net incremental flows to general nodes (fig. 9). The water balance for canal reaches, pond nodes, and general nodes guaranteed that mass conservation at a sink/source node was satisfied.

## Linear-Network Optimization Flow Model

The linear-network flow model given by equations 3–5 was rewritten for the operation of canals and control ponds as follows:

Minimize

$$\sum_i \left( C_i^u SU_i + C_i^l SL_i \right)^t + \qquad (28)$$

$$\sum_i \sum_j \left( C_{ij}^n QN_{ij} + C_{uh}^u QU_{ij} - C_{ij}^l QL_{ij} \right)^t + \sum_i C_i^w DW_i^t ,$$

subject to

$$\sum (OQN_{ji} + OQU_{ji} - OQL_{ji})^t - \sum_j \left( IQN_{ij} + IQU_{ij} - IQL_{ij} \right) - \left( SU_i - SL_i \right)^t + NV^t_i = 0 \quad (29)$$

for all pond nodes $i$,

$$\sum_j \left( OQN^t_{ji} + OQU^t_{ji} - OQL^t_{ji} \right) - \sum_j \left( IQN^t_{ij} + IQU^t_{ij} - IQL^t_{ij} \right) + I^t_i - \left( TR^t_i - DW^t_i \right) = 0 \quad (30)$$

for all general nodes $i$,

$$\left( IQN^t_{ij} + IQU^t_{ij} - IQL^t_{ij} \right) - \left( OQN^t_{ij} + OQU^t_{ij} - OQL^t_{ij} \right) - EV^t_i - SP^t_i = S^t_i - S^{t-1}_i \quad (31)$$

for canal flow arcs $(i,j)$,

$$0 \le SU^t_i \le \overline{SU}_i, \quad (32)$$

$$0 \le SL^t_i \le \overline{SL}_i \;, \quad (33)$$

$$0 \le \underline{QN}_{ij} \le QN^t_{ij} \le \overline{QN}_{ij}, \quad (34)$$

$$0 \le QU^t_i \le \overline{QU}_i, \quad (35)$$

$$0 \le QL^t_i \le \overline{QL}_i, \quad (36)$$

$$0 \le DW^t_i \le TR^t_i, \quad (37)$$

where $C_i^u$ and $C_i^l$ were the penalty coefficients for cost per unit; and the upper bars and lower bars in equations 32–37 were upper and lower flow boundaries of an associated arc. Water storage deviated from the rule curve at pond node $i$ for the upper zone and lower zone, respectively. $C_{ij}^n$, $C_{ij}^u$, and $C_{ij}^l$ denoted the penalty coefficients for cost per unit flow in canal $ij$ for the normal, upper, and lower flow zones, respectively.

The linear-network optimization flow model given by equations 28–37 was a typical minimum-cost flow problem in network analysis. Several algorithms exist for solving a minimum-cost flow problem. One of the algorithms, called the out-of-kilter algorithm (Fulkerson, 1961; Bazaraa and others, 1990), was used in developing the computer program called OPONDS (the optimal Operation of a system of PONDS) developed for this study (see Appendices for the description and listing of the computer program).

## Model Supplements

In the following section, the methods used in OPONDS to estimate canal water storage, canal-flow transmission loss, surface runoff, and flow through hydraulic structures are described.

### Estimation of Canal Water Storage

The Muskingum's method (McCuen, 1989) was used to estimate canal storage in this study. The method assumes that, for given a reach, canal storage $(S)$ can be expressed in terms of inflow and outflow rates as follows:

$$S = K[xI + (1-x)O], \quad (38)$$

where

$K$ = the storage constant defined by the ratio of storage to discharge. The storage constant $K$ has the dimension of time; therefore, $K$ is often called traveltime. The coefficients of $K$ and $x$ are generally determined using historical discharge data (Wu and others, 1985; Chow and others, 1988; McCuen,1989);

$x$ = the dimensionless weighting factor for the storage effect of inflow and outflow. The value of $x$ is usually between 0 and 0.5;

$I$ = inflow rate; and

$O$ = outflow rate.

### Estimation of Canal-Flow Transmission Losses

To estimate canal-flow transmission losses to an aquifer, two approximation methods were included in the computer program OPONDS. The first one was based on Darcy's equation given by:

$$q = K_b \frac{(Z_{sw} - Z_{gw})}{d} LB \quad , \quad (39)$$

where

$q$ = canal seepage rate along canal reach;

$K_b$ = canal-bottom hydraulic conductivity;

$Z_{sw}$ = average canal surface-water elevation, which is the average water depth ($h$) plus the canal-bottom elevation ($Z_b$);

$Z_{gw}$ = average ground-water elevation below the canal bottom, which is the average value along the canal;

$d$ = average canal-bottom thickness;

$L$ = canal-reach length; and

$B$ = canal water-surface width.

If a canal gained water from the aquifer, the seepage ($q$) in equation 39 was a negative number. If the ground-water elevation $Z_{gw}$ was lower than the average canal-bottom elevation, then the seepage rate, $q$, was simplified as:

$$q = K_b hLB, \tag{40}$$

where $h$ was the average water depth in a canal reach.

The water depth $h$ was estimated iteratively using Manning's equation (Henderson, 1966):

$$v = \frac{1.486R^{2/3}J^{1/2}}{n}, \tag{41}$$

in which $v$ was the average velocity, in feet per second [$v = Q / A(h)$]; $R$ was the hydraulic radius [$R = A(h) / P(h)$], in feet; $A(h)$ was the cross-section area, in square feet; $P(h)$ was the wetted perimeter in feet; $J$ was the hydraulic slope; and $n$ was the roughness coefficient, which is dependent on canal bottom materials.

The second approximation method (Jordan, 1977) assumed that the rate of canal-flow transmission loss at any point was proportional to the flow at that point and that the canal characteristics were uniform for a given reach; that is,

$$\frac{dQ_x}{dx} = -kQ_x, \tag{42}$$

where $x$ was the distance coordinate, and $k$ was the transmission loss per unit length of canal [1/L] and was simply called transmission loss coefficient.

For a given canal reach of length $L$, the transmission loss then was estimated by:

$$q = (1.0 - e^{-kL})IQ = c\,IQ, \tag{43}$$

where $IQ$ was the inflow entering a canal, and $c$ was a transmission loss rate for a given canal reach of length $L$ and was estimated using seepage test data with a least-squares technique or other techniques.

## Estimation of Direct Overland Surface Runoff

The Soil Conservation Service (1985) developed a method for estimating direct overland surface runoff depth from precipitation. The runoff depth $Q$ generated by precipitation $P$ was given by:

$$Q = \frac{(P - 0.2S)^2}{P + 0.8S}, \tag{44}$$

where $S$ was the potential maximum retention (the amount of rain not converted to runoff after runoff begins) given by:

$$S = \frac{1000}{CN} - 10, \tag{45}$$

in which $CN$ was the SCS curve number. The SCS curve number ($CN$) is an index that represents the combination of hydrologic soil group and land use. $CN$ is a function of three factors—soil group, land-cover type, and antecedent moisture conditions. The range of $CN$ is from 0 to 100. The curve number for average antecedent soil-moisture conditions (AMC II) can be interpreted for given soil properties and land-cover type (Soil Conservation Service, 1985; McCuen, 1989). For dry conditions (AMC I) and wet conditions (AMC III), equivalent curve numbers can be computed using the following equations (Chow and others, 1988):

$$CN(I) = \frac{4.2CN(II)}{10 - 0.058CN(II)}, \tag{46}$$

and

$$CN(III) = \frac{23CN(II)}{10 + 0.13CN(II)}, \tag{47}$$

where $CN(I)$, $CN(II)$, and $CN(III)$ are the curve numbers for the dry, average, and wet conditions, respectively.

The range of antecedent moisture conditions for each class is shown in table 5 (Chow and others, 1988). The SCS curve numbers for average soil-moisture conditions for the Quivira National Wildlife Refuge are summarized in table 4 in an earlier section.

## Flow Through Hydraulic Structures

Water releases from ponds are through hydraulic control structures. The amount of water release depends on several factors, such as the pond water level, hydraulic-structure types and sizes, and the operation of structures. In the following sections, flows through four types of structures are discussed.

### Flow Over Sharp-Crested Weir

A sharp-crested weir consists of a vertical plate mounted at right angles to the flow and having a sharp-edged crest (fig. 10*A*). The discharge equation is:

$$Q = mb\sqrt{2g}H_o^{1.5}, \qquad (48)$$

where

$Q$ = discharge over weir, in cubic feet per second;

$m$ = discharge coefficient, which is dimensionless;

$b$ = weir length, in feet;

$H_o$ = total energy head $(= H + v_o^2/2g)$, in feet. If approaching velocity $v_o \approx 0$, then $H_o = H$, where $H$ is the static water head on a weir, referred to as the weir crest; and

$g$ = gravity acceleration (= 32.17 $ft/s^2$).

The discharge coefficient ($m$) for free discharge is a function of certain dimensionless ratios that describe the geometry of the canal and the weir (Hulsing, 1967). One simple expression for free discharge with no side contraction is (Henderson, 1966):

$$m = 0.4073 + 0.0533\,(H/P),\ \text{where } 0 < H/P < 5, \qquad (49)$$

in which $P$ is the weir height (fig. 10*A*).

### Flow Under Gate on Broad-Crested Weir

Flow under a vertical sluice gate on a broad-crested weir (fig. 10*B*) was calculated by:

**Table 5.** Classification of antecedent soil-moisture conditions (AMC) for SCS curve-number method of rainfall abstractions

[From Chow and others, 1988]

| AMC | Total 5-day antecedent rainfall (inches): Dormant season | Growing season |
|---|---|---|
| I | Less than 0.5 | Less than 1.4 |
| II | 0.5–1.1 | 1.4–3.1 |
| III | More than 1.1 | More than 3.1 |

$$Q = mbe\sqrt{2gH_o}, \qquad (50)$$

where $e$ was the gate opening height, and the other terms had the same definitions as in equation 48.

If $e/H > 0.65$, flow was not affected by the gate. The discharge coefficient for the free outflow under the gate depended on the relative gate opening height ($e/H$) and was approximated by (Swamee, 1992):

$$m = 0.611\left(\frac{1-\frac{e}{H}}{1+15\frac{e}{H}}\right)^{0.072}, \qquad (51)$$

where $e/H < 0.65$.

### Flow Under Gate on Spillway

Flow under a gate on a spillway was calculated by:

$$Q = mbe\sqrt{2gH_o}. \qquad (52)$$

The definition of variables in equation 52 is the same as equation 50. The discharge coefficient ($m$) for a standard spillway depended not only on the relative gate opening height ($e/H$) but also on the design water head ($H_d$) and design discharge coefficient ($m_d$) (U. S. Army Engineer Waterways Experiments Station, 1972). In the case that design water head and design coefficient were not available, the free outfall flow with a flat gate and sharp-crested edge of the gate facing downstream (fig. 10*C*) was approximated using the following equation (Chengdu Science and Technology University, 1979):

$$m = 0.65 - 0.186\,(e/H). \qquad (53)$$

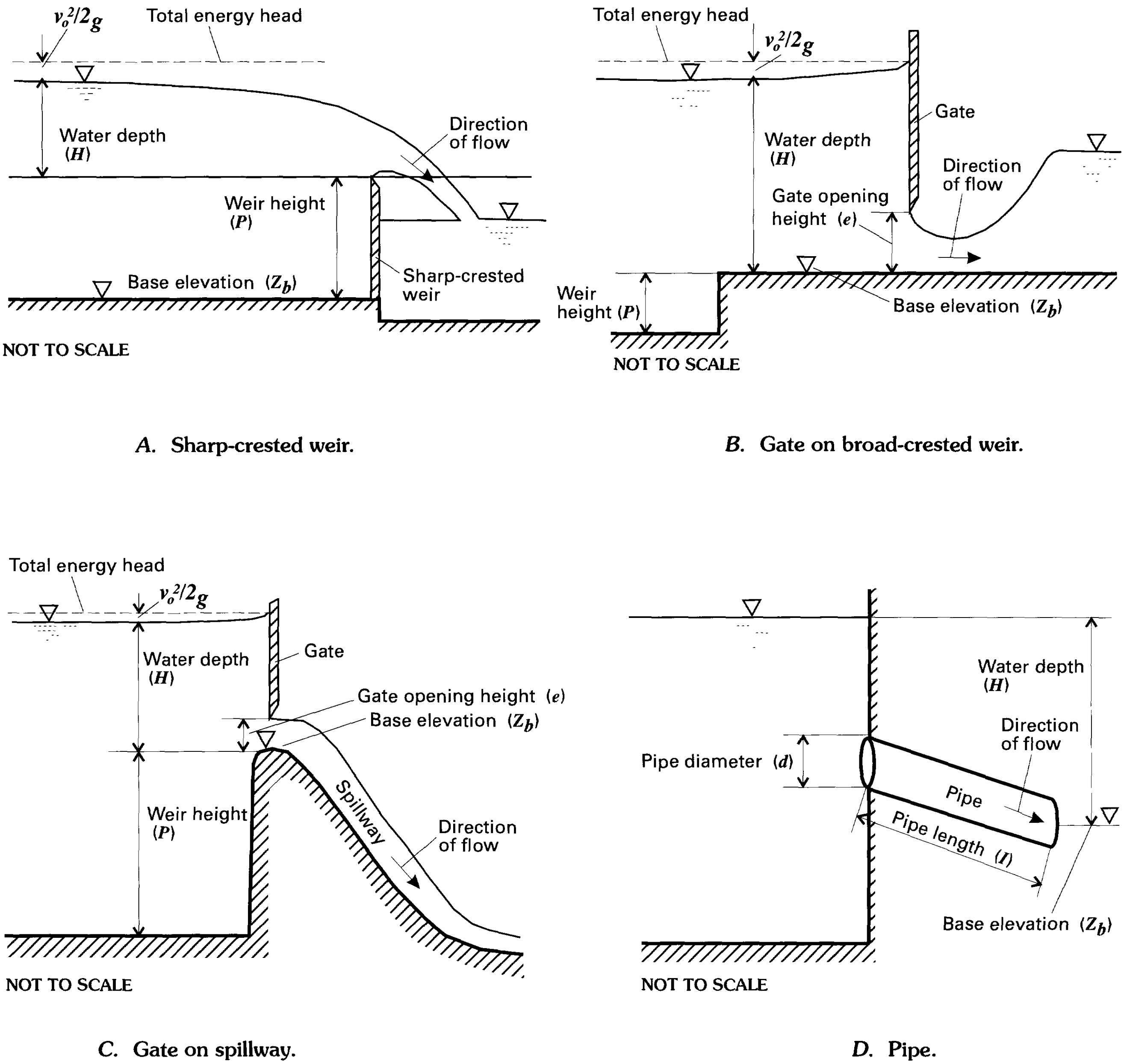



**Figure 10.** Flow through hydraulic structures.

### Pipe Outflow

For free flow through a pipe (fig. 10*D*), the discharge was estimated by:

$$Q = mA\sqrt{2gH} \ , \tag{54}$$

where $A$ was the area of cross section of the pipe, $H$ was the water depth above the water outlet, and the discharge coefficient ($m$) was given by (Chengdu Science and Technology University, 1979):

$$m = \frac{1}{\sqrt{1 + \lambda\frac{l}{d} + \Sigma\zeta}} \ , \tag{55}$$

where $l$ was the length of the pipe, $d$ was the diameter of the pipe, $\lambda$ was the pipe friction coefficient that was determined by pipe materials, and $\zeta$ was the entrance loss coefficient that was determined by the shape of the entrance.

# SIMULATION OF CANAL AND CONTROL-POND OPERATION FOR 1996

From June 11 through December 11, 1996, personnel at the Quivira National Wildlife Refuge measured water-surface levels about four or five times a month for most ponds. Streamflow discharges at the Rattlesnake Creek near Zenith and Raymond streamflow-gaging stations were measured continuously by USGS; however, the discharges in canals on the refuge were not measured. A simulation was conducted to determine the operation of canals and control ponds under 1996 conditions. The major objective was to determine the operation policy for canals and control ponds on the refuge so that the simulated pond water levels would match well with the measured water levels. The basic approach was to use the measured pond water levels as the pond rule curve, to set up pond zoning and the priority relations of control ponds, to determine pond releases to canals or other ponds to satisfy the measured discharges of Rattlesnake Creek near Raymond, and to examine simulated water levels for those ponds without water-level measurements. In the following sections, the data and the related necessary assumptions needed to conduct the simulation are discussed, the operation policy of ponds is discussed, and the simulation results are presented.

## Data Preparation

In this section, data needed for the simulation are discussed. Measurement data were used if available. If some data were not available, reasonable values were estimated from other sources.

### Precipitation

The amount of precipitation directly affects the surface runoff to ponds. Daily precipitation measured at the refuge headquarters from June 11 through December 11, 1996, is shown in figure 11*A* (Marios Sophocleous, Kansas Geological Survey, written commun., 1997). The total amount of precipitation for the period was 13.96 in.

### Water-Surface Evaporation

The daily potential evapotranspiration (PET) (Marios Sophocleous, Kansas Geological Survey, written commun., 1997) is shown in figure 11*B*. It was assumed that the daily water-surface evaporation rates on the refuge were the same as the corresponding daily potential evapotranspiration. The total water-surface evaporation for the simulation period was 25.21 in.

### Canal Discharge

Discharge for Rattlesnake Creek measured at the USGS streamflow-gaging stations near Zenith and Raymond from June 11 through December 11, 1996, is shown in figure 11*C*. The mean daily discharge rates for the simulation period were 48.72 and 47.73 $ft^3/s$ for the Zenith and Raymond stations, respectively.

For this simulation, the daily mean discharges observed at the USGS Zenith station were used as water supply from Rattlesnake Creek to Little Salt Marsh. The daily mean discharges observed at the USGS Raymond station were used as the required stream outflow from the refuge through Rattlesnake Creek.

### Canal-Flow Transmission Losses

Flow transmission losses from canals on the refuge were difficult to estimate. Personnel from the refuge did four seepage tests (table 6) along a 15,129-ft reach of Rattlesnake Creek downstream from Little Salt Marsh during 1996 (see fig. 2). Applying the least-squares method to equation 43, the estimated transmission loss coefficient ($k$) (equation 42) was equal to $9.16 \times 10^{-6}\ ft^{-1}$. Due to a lack of data for the remaining canals on the refuge, this value of $k$ was used for the estimation of flow transmission loss rate $c$ (equation 43) for all canals south of the RC Canal. Because canals north of the RC Canal are located in the ground-water discharge area, no canal-flow transmission losses occurred for these canals. The ground-water discharge to these canals was included in discharge to ponds (see table 2).

### Ground-Water Discharge to Ponds

Ground-water discharge to ponds on the refuge during the simulation period was not available. Instead, ground-water discharges to ponds based on information provided by Marios Sophocleous (Kansas

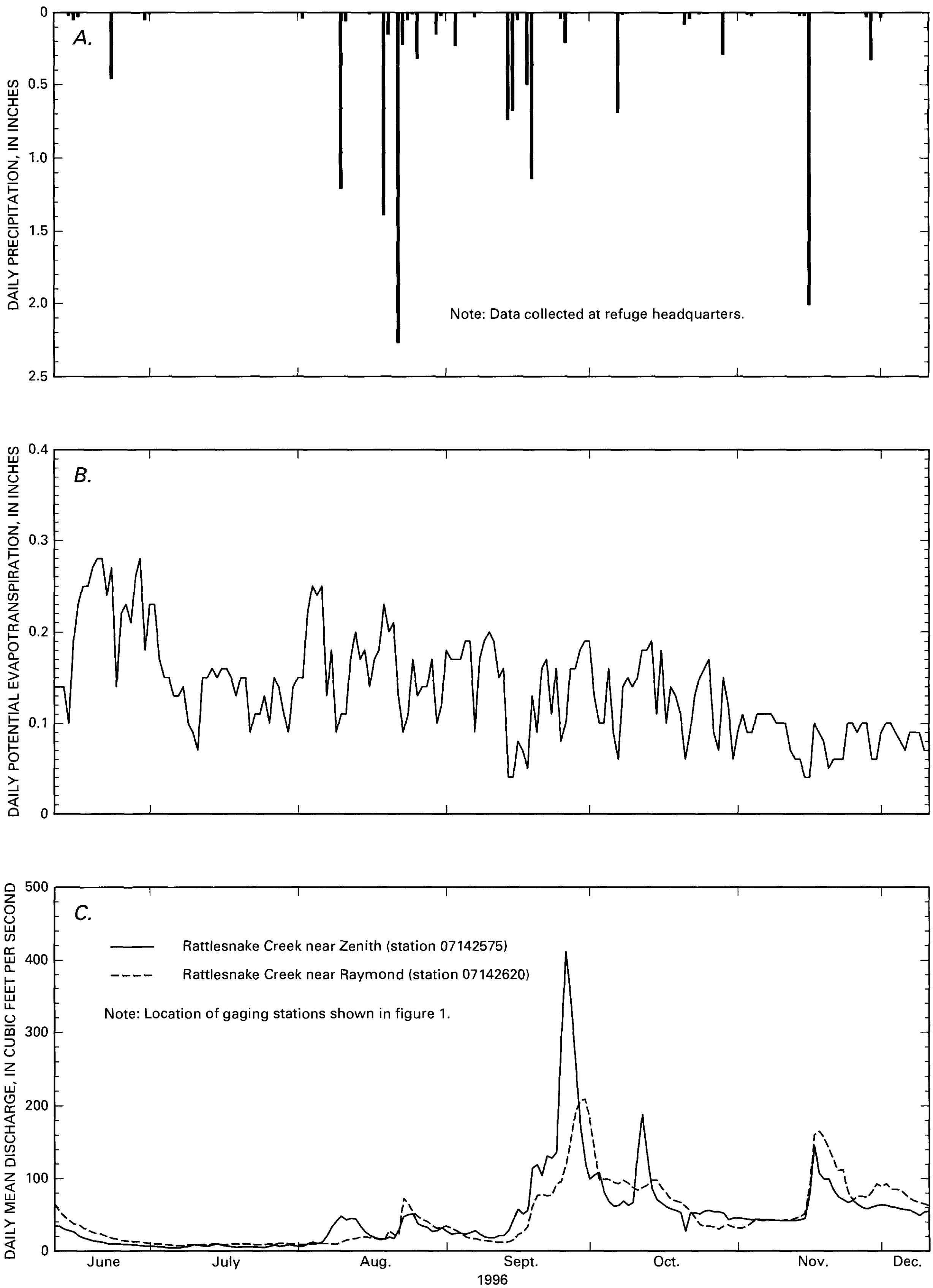



**Figure 11.** *(A)* Daily precipitation, *(B)* daily potential evapotranspiration, *(C)* daily mean discharge for Rattlesnake Creek, June 11 through December 11, 1996. Precipitation and potential evapotranspiration data from the Kansas Geological Survey (Marios Sophocleous, written commun., 1997).

**Table 6.** Results of seepage tests along Rattlesnake Creek, 1996, at Quivira National Wildlife Refuge

[Data from U.S. Fish and Wildlife Service (Megan Estep-Johnston, written commun., 1996). $ft^3/s$, cubic feet per second]

| Date (month/day/year) | Discharge ($ft^3/s$) Upstream test site 1 (fig. 2) | Downstream test site 2 (fig. 2) |
|---|---|---|
| 06/26/96 | 6.93 | 6.15 |
| 07/17/96 | 8.98 | 7.46 |
| 07/24/96 | 3.13 | 2.92 |
| 09/09/96 | 5.52 | 5.12 |

Geological Survey, written commun., 1997) for 1994 were used (table 2).

### Pond Water-Surface Elevations

Water-surface elevations for selected control ponds were measured during the simulation period. Table 7 lists the ponds with measured water-surface elevations, the number of measurements, and the minimum and maximum water-surface elevations for the ponds. Because the water levels may be at the bottom of ponds or above the staff gage at ponds, the number of observations of water-surface elevations listed in table 7 may be different than the number of measurements listed. The difference between the number of observations and the number of measured elevations is the number of records without measurements. Those water-surface elevations observed outside the range of measurement on pond staff gages were treated accordingly as the pond-bottom elevation or the full-pond elevation in this simulation.

## Pond Zoning and Operating Policy

Each control pond was divided into four storage zones—inactive zone, lower zone, upper zone, and extended upper zone as shown in figure 5. Target water levels (rule curves) were set at the top of the lower zone. For ponds with measured water-surface elevations (table 7), the measured water elevations were used as their rule curves, which indicates that the rule curves changed during the simulation period and so did the storage capacity of lower and upper zones. For those ponds without measured water levels, the rule curves were set at 95 percent of their corresponding full-pond storage capacities. The capacity of the inactive zone of a pond was set at 20 percent of full-pond storage capacity (selected in consultation with the U.S. Fish and Wildlife Service). Some of the rule curves for some ponds with measured water levels were located in the inactive zone during the simulation. In this case, the top boundary of the inactive zone capacity was set at the rule curve, and the capacity of the lower zone was set to zero. The top boundary of the upper zone was set at the full-pond elevation. The top boundary of the extended upper zone was set 0.5 ft higher than its full-pond elevation. For ponds whose maximum measured water levels were higher than the full-pond elevation plus 0.5 ft, the top boundaries of the extended upper zone were set at the maximum measured water level. Pond zoning expressed as pond storage is summarized in table 8.

To operate the system of canals and control ponds on the refuge, it was necessary to establish the priority of the ponds. Because Little Salt Marsh (water unit 5), which is supplied by Rattlesnake Creek, serves as the principal water-storage unit for the entire refuge, the highest operational priorities were given to its storages zones. Water units (75, 78, 80, 81, and 83, see fig. 2) in the north part of the refuge were given the lowest operational priorities because these ponds are at the downstream end of the refuge and control less drainage area. The remaining ponds were given priorities in between the highest and the lowest priorities. Under this operating policy, water to satisfy the downstream water requirements was released first (1) from the lowest priority ponds when water levels at the highest priority ponds were below the rule curve so that high-priority pond water levels were as close as possible to their rule curves, or (2) from the highest priority ponds when their water levels were higher than the rule curves so that the water levels would decrease to as close to their rule curves as possible. To represent priorities of ponds, different penalty coefficients were assigned to each of the storage zones of the ponds. The higher the priority, the higher the penalty coefficient assigned. It should be noted that the relative magnitudes, not the absolute values, of the penalty coefficients determined the optimal operation of the system. Different combinations of assigned values of penalty coefficients were tested for the control ponds on the refuge. Typical values of penalty coefficients used in this simulation are summarized in table 8.

Because there are no flow requirements such as minimum-required flow for canals on the refuge, there

**Table 7.** Summary of water-surface elevations for selected ponds at Quivira National Wildlife Refuge, June 11 through December 11, 1996

[Data from U.S. Fish Wildlife Service, written commun., 1997]

| Water-unit number (fig. 2) | Number of observations | Number of measurements | Maximum measured elevation (feet above sea level) | Minimum measured elevation (feet above sea level) |
|---|---|---|---|---|
| 5 | 30 | 30 | 1,783.30 | 1,782.62 |
| 7 | 30 | 23 | 1,778.96 | 1,777.33 |
| 10A | 30 | 16 | 1,778.67 | 1,777.03 |
| 10B | 30 | 29 | 1,778.89 | 1,777.32 |
| 10C | 32 | 32 | 1,774.86 | 1,773.22 |
| 11 | 32 | 14 | 1,773.91 | 1,771.95 |
| 14A | 30 | 30 | 1,777.92 | 1,776.30 |
| 14B | 30 | 30 | 1,777.36 | 1,774.90 |
| 16 | 30 | 28 | 1,774.46 | 1,772.72 |
| 20A | 29 | 29 | 1,770.84 | 1,769.56 |
| 21 | 29 | 17 | 1,769.09 | 1,767.00 |
| 22 | 29 | 29 | 1,767.17 | 1,764.91 |
| 23 | 29 | 27 | 1,764.78 | 1,763.02 |
| 24 | 30 | 30 | 1,770.46 | 1,769.61 |
| 25 | 33 | 20 | 1,766.92 | 1,763.16 |
| 26 | 28 | 28 | 1,762.06 | 1,760.14 |
| 28 | 30 | 17 | 1,767.81 | 1,764.10 |
| 29 | 30 | 26 | 1,761.83 | 1,757.20 |
| 30 | 30 | 17 | 1,760.01 | 1,756.48 |
| 40 | 29 | 22 | 1,742.59 | 1,738.58 |
| 48 | 29 | 25 | 1,754.28 | 1,750.88 |
| 49 | 29 | 29 | 1,754.13 | 1,750.25 |
| 58 | 30 | 30 | 1,740.90 | 1,739.59 |
| 61 | 29 | 29 | 1,743.89 | 1,742.54 |
| 62 | 29 | 29 | 1,742.64 | 1,739.55 |
| 63 | 29 | 29 | 1,740.73 | 1,739.17 |
| 75 | 29 | 7 | 1,740.17 | 1,739.55 |

**Table 8.** Initial storage, zoning, and penalty coefficients assigned to control ponds at Quivira National Wildlife Refuge, June 11 through December 11, 1996

| Water-unit number (fig. 2) | Initial storage (acre-feet) | Upper boundary of extended upper zone (acre-feet) | Penalty coefficient for extended upper zone | Upper boundary of upper zone (acre-feet) | Penalty coefficient for upper zone | Lower boundary of lower zone (acre-feet) | Penalty coefficient for lower zone | Lower boundary of inactive zone (acre-feet) | Penalty coefficient for inactive zone |
|---|---|---|---|---|---|---|---|---|---|
| 5 | 1,988.26 | 2,312.18 | 2,000 | 1,866.00 | 1,500 | 373.20 | 1,500 | 1.00 | 5,000 |
| 7 | 39.72 | 72.90 | 2,000 | 40.00 | 1,500 | 8.00 | 1,000 | 1.00 | 2,000 |
| 10A | 145.48 | 180.30 | 2,000 | 145.00 | 1,500 | 29.00 | 1,000 | 1.00 | 2,000 |
| 10B | 145.48 | 180.30 | 2,000 | 145.00 | 1,500 | 29.00 | 1,000 | 1.00 | 2,000 |
| 10C | 19.54 | 21.81 | 2,000 | 13.00 | 1,500 | 2.60 | 1,000 | 1.00 | 2,000 |
| 11 | 388.37 | 440.07 | 2,000 | 338.00 | 1,500 | 67.60 | 1,000 | 1.00 | 2,000 |
| 14A | 161.70 | 242.20 | 2,000 | 196.00 | 1,500 | 39.20 | 1,000 | 1.00 | 2,000 |
| 14B | 93.40 | 185.74 | 2,000 | 96.00 | 1,500 | 19.20 | 1,000 | 1.00 | 2,000 |
| 14C | 15.51 | 19.07 | 2,000 | 16.00 | 500 | 3.20 | 500 | 1.00 | 2,000 |
| 16 | 62.67 | 96.07 | 2,000 | 80.00 | 1,500 | 16.00 | 1,000 | 1.00 | 2,000 |
| 20A | 163.88 | 268.01 | 2,000 | 195.00 | 1,500 | 39.00 | 1,000 | 1.00 | 2,000 |
| 20B | 163.88 | 268.01 | 2,000 | 195.00 | 1,500 | 39.00 | 1,000 | 1.00 | 2,000 |
| 21 | 34.34 | 96.62 | 2,000 | 81.00 | 1,500 | 16.20 | 1,000 | 1.00 | 2,000 |
| 22 | 2.30 | 18.81 | 2,000 | 13.00 | 1,500 | 2.60 | 1,000 | 1.00 | 2,000 |
| 23 | 15.41 | 19.66 | 2,000 | 15.00 | 1,500 | 3.00 | 1,000 | 1.00 | 2,000 |
| 24 | 132.55 | 139.01 | 2,000 | 35.00 | 1,500 | 7.00 | 1,000 | 1.00 | 2,000 |
| 25 | 18.00 | 344.05 | 2,000 | 296.00 | 1,500 | 59.20 | 1,000 | 1.00 | 2,000 |
| 26 | 91.48 | 142.39 | 2,000 | 111.00 | 1,500 | 22.20 | 1,000 | 1.00 | 2,000 |
| 28 | 6.11 | 198.82 | 2,000 | 153.00 | 1,500 | 30.60 | 1,000 | 1.00 | 2,000 |
| 29 | 0.20 | 124.51 | 2,000 | 91.00 | 1,500 | 18.20 | 1,000 | 1.00 | 2,000 |
| 30 | 2.82 | 161.64 | 2,000 | 119.00 | 1,500 | 23.80 | 1,000 | 1.00 | 2,000 |
| 40 | 55.91 | 83.19 | 2,000 | 66.00 | 1,500 | 13.20 | 1,000 | 1.00 | 2,000 |
| 48 | 3.94 | 161.19 | 2,000 | 113.00 | 1,500 | 22.60 | 1,000 | 1.00 | 2,000 |
| 49 | 51.63 | 209.05 | 2,000 | 159.00 | 1,500 | 31.80 | 1,000 | 1.00 | 2,000 |
| 57 | 212.22 | 280.74 | 2,000 | 212.00 | 1,500 | 42.40 | 1,000 | 1.00 | 2,000 |

**Table 8.** Initial storage, zoning, and penalty coefficients assigned to control ponds at Quivira National Wildlife Refuge, June 11 through December 11, 1996—Continued

| Water-unit number (fig. 2) | Initial storage (acre-feet) | Upper boundary of extended upper zone (acre-feet) | Penalty coefficient for extended upper zone | Upper boundary of upper zone (acre-feet) | Penalty coefficient for upper zone | Lower boundary of lower zone (acre-feet) | Penalty coefficient for lower zone | Lower boundary of inactive zone (acre-feet) | Penalty coefficient for inactive zone |
|---|---|---|---|---|---|---|---|---|---|
| 58 | 146.39 | 302.82 | 2,000 | 251.00 | 1,500 | 50.20 | 1,000 | 1.00 | 2,000 |
| 61 | 212.80 | 613.17 | 2,000 | 498.00 | 1,500 | 99.60 | 1,000 | 1.00 | 2,000 |
| 62 | 48.58 | 145.00 | 2,000 | 120.00 | 1,500 | 24.00 | 1,000 | 1.00 | 2,000 |
| 63 | 268.98 | 419.01 | 2,000 | 339.00 | 1,500 | 67.80 | 1,000 | 1.00 | 2,000 |
| 75 | 2,445.85 | 3,490.32 | 1,000 | 2,446.00 | 500 | 489.20 | 500 | 1.00 | 2,000 |
| 78 | 5,270.43 | 6,091.37 | 1,000 | 5,270.00 | 500 | 1,054.00 | 500 | 1.00 | 2,000 |
| 80 | 355.20 | 474.34 | 1,000 | 355.00 | 500 | 71.00 | 500 | 1.00 | 2,000 |
| 81 | 25.31 | 60.68 | 1,000 | 25.00 | 500 | 5.00 | 500 | 1.00 | 2,000 |
| 83 | 314.34 | 419.31 | 1,000 | 314.00 | 750 | 62.80 | 750 | 1.00 | 2,000 |

was only one flow zone for canals designated in this simulation. It was assumed that flow through a canal reach ranged in magnitude from zero to the full capacity of the canal. Because of the complexity of the canal flow network on the refuge, flows could reach the same location through different routes of canals. Different penalty coefficients were assigned to the flow zones of canals so that the most efficient route could be determined by minimizing the total penalty applied to canal flows. However, costs of transporting water through canals were not available. Because Rattlesnake Creek is used as the major route to distribute water to the refuge and because other canals are used only when necessary, flows through Rattlesnake Creek and canals downstream from control ponds were assigned penalty coefficients of zero, and the remaining canals were assigned nonzero penalty coefficients as shown in table 9 (see figures 2 and 4 for nodal names, location, and flow network).

## Results

The simulation of canal and control-pond operation at the Quivira National Wildlife Refuge for June 11 through December 11, 1996, was conducted using the following specifications for pond zoning, operating policy, and canal outflow from the refuge: (1) four storage zones for each pond, with the inactive storage of 20 percent of full-pond storage capacity; (2) rule curves set at the measured water levels for ponds with measurements, otherwise at 95 percent of full-pond storage capacity; (3) initial storage in ponds interpreted from the water levels measured on June 10, 1996, for ponds with measurements, otherwise set at 95 percent of full-pond storage capacity; and (4) outflows from the refuge through Rattlesnake Creek near the USGS streamflow-gaging station near Raymond equal to the observed discharges at the streamflow-gaging station (fig. 11*C*).

Figures 12*A-D* show the water-budget components simulated for the operation of water unit 5. Similar figures also can be generated for other control ponds. Inflows shown in figure 12*A* are upstream inflows from Rattlesnake Creek, which are equal to the discharges observed at the USGS streamflow-gaging station near Zenith. Total downstream releases shown in figure 12*C* are the summations of releases to all downstream nodes (water units 7 and 10A, and nodes C–2 and JE–1, see figure 4). Ground-water seepage during the simulation period shown in figure 12*B* is almost the same for the whole simulation period (the values were estimated for 1994, see table 2). Figure 12*E* shows the simulated and measured water stages and depths. From July 9 to August 8, even though there were no releases from the pond, the simulated water stages were lower than the

**Table 9.** Penalty coefficients for canal flows at Quivira National Wildlife Refuge, June 11 through December 11, 1996

| Canal | | | Canal | | | Canal | | |
|---|---|---|---|---|---|---|---|---|
| From-node name (fig. 4) | To-node name (fig. 4) | Penalty coefficient | From-node name (fig. 4) | To-node name (fig. 4) | Penalty coefficient | From-node name (fig. 4) | To-node name (fig. 4) | Penalty coefficient |
| Zenith | Unit 5 | 0 | Unit 24 | Unit 21 | 10 | Unit 61 | JE–4 | 0 |
| Unit 5 | Unit 7 | 10 | Unit 24 | Unit 20B | 1,000 | Unit 63 | JE–5 | 0 |
| Unit 5 | Unit 10A | 10 | Unit 25 | JE–2 | 10 | Unit 63 | JE–6 | 0 |
| Unit 5 | JE–1 | 0 | Unit 25 | Unit 26 | 10 | Unit 75 | Unit 78 | 10 |
| Unit 5 | C–2 | 10 | Unit 26 | 48E | 10 | Unit 78 | Unit 81 | 10 |
| Unit 7 | Unit 10B | 10 | 48E | Unit 48 | 0 | Unit 81 | Unit 80 | 10 |
| Unit 10A | Unit 10B | 10 | 48E | Unit 49 | 0 | Unit 80 | JN–1 | 0 |
| Unit 10B | Unit 10C | 10 | Unit 28 | Unit 29 | 10 | Unit 83 | JN–1 | 0 |
| Unit 10C | Unit 11 | 10 | Unit 29 | Unit 30 | 10 | JN–1 | JE–7 | 0 |
| Unit 11 | SKSC[1] | 1,250 | Unit 30 | WCE | 10 | DCC | DCF | 100 |
| C–2 | F–1 | 0 | WCE | Unit 48 | 0 | DCC | SKSC[1] | 5,000 |
| C–2 | D–1 | 0 | WCE | RCD | 0 | DCF | 40C | 100 |
| F–1 | F–2 | 0 | Unit 48 | Unit 49 | 10 | DCF | JE–0 | 100 |
| F–1 | Unit 14B | 0 | Unit 48 | Unit 55 | 10 | JE–0 | 37 | 100 |
| D–1 | Unit 14A | 0 | Unit 49 | RCB | 10 | 37 | 39 | 100 |
| D–1 | WCA | 0 | Unit 49 | JE-3 | 10 | 39 | JE–9 | 100 |
| Unit 14A | UNIT 16 | 10 | Unit 55 | RCC | 0 | 40C | Unit 40 | 0 |
| Unit 14A | J14 | 10 | Unit 55 | RCF | 0 | 40C | Unit 62 | 0 |
| Unit 14B | J14 | 10 | JE–1 | Unit 24 | 0 | Unit 40 | JE–8 | 10 |
| Unit 14B | Unit 20B | 10 | JE–2 | JE–3 | 0 | Unit 62 | JE–5 | 10 |
| J14 | Unit 20A | 0 | JE–3 | RCA | 0 | Unit 62 | Unit 40 | 10 |
| F–2 | Unit 14C | 0 | RCA | RCB | 0 | JE–4 | JE–5 | 0 |
| F–2 | Unit 20B | 0 | RCA | JE–4 | 0 | JE–5 | JE–6 | 0 |
| Unit 14C | JE–1 | 0 | RCB | Unit 61 | 0 | JE–6 | JE–7 | 0 |
| Unit 16 | WCA | 10 | RCB | RCC | 0 | JE–7 | JE–8 | 0 |
| WCA | Unit 28 | 10 | RCC | RCF | 0 | JE–8 | JE–9 | 0 |
| Unit 20A | Unit 21 | 10 | RCF | Unit 57 | 0 | JE–9 | JE–10 | 0 |
| Unit 20B | Unit 20A | 10 | RCF | RCD | 0 | JE–10 | RAYMOND | 0 |
| Unit 21 | Unit 22 | 10 | RCD | Unit 58 | 0 | RAYMOND | SKSC[1] | 0 |
| Unit 22 | Unit 23 | 10 | Unit 57 | Unit 78 | 10 | | | |
| Unit 23 | Unit 26 | 10 | Unit 58 | Unit 75 | 10 | | | |
| Unit 24 | Unit 25 | 10 | Unit 58 | Unit 78 | 10 | | | |
| Unit 24 | JE–2 | 0 | Unit 61 | Unit 57 | 10 | | | |
| Unit 24 | DCC | 100 | Unit 61 | Unit 63 | 10 | | | |

[1]Nodal name SKSC is used to specify that the end node of a canal is outside the refuge.

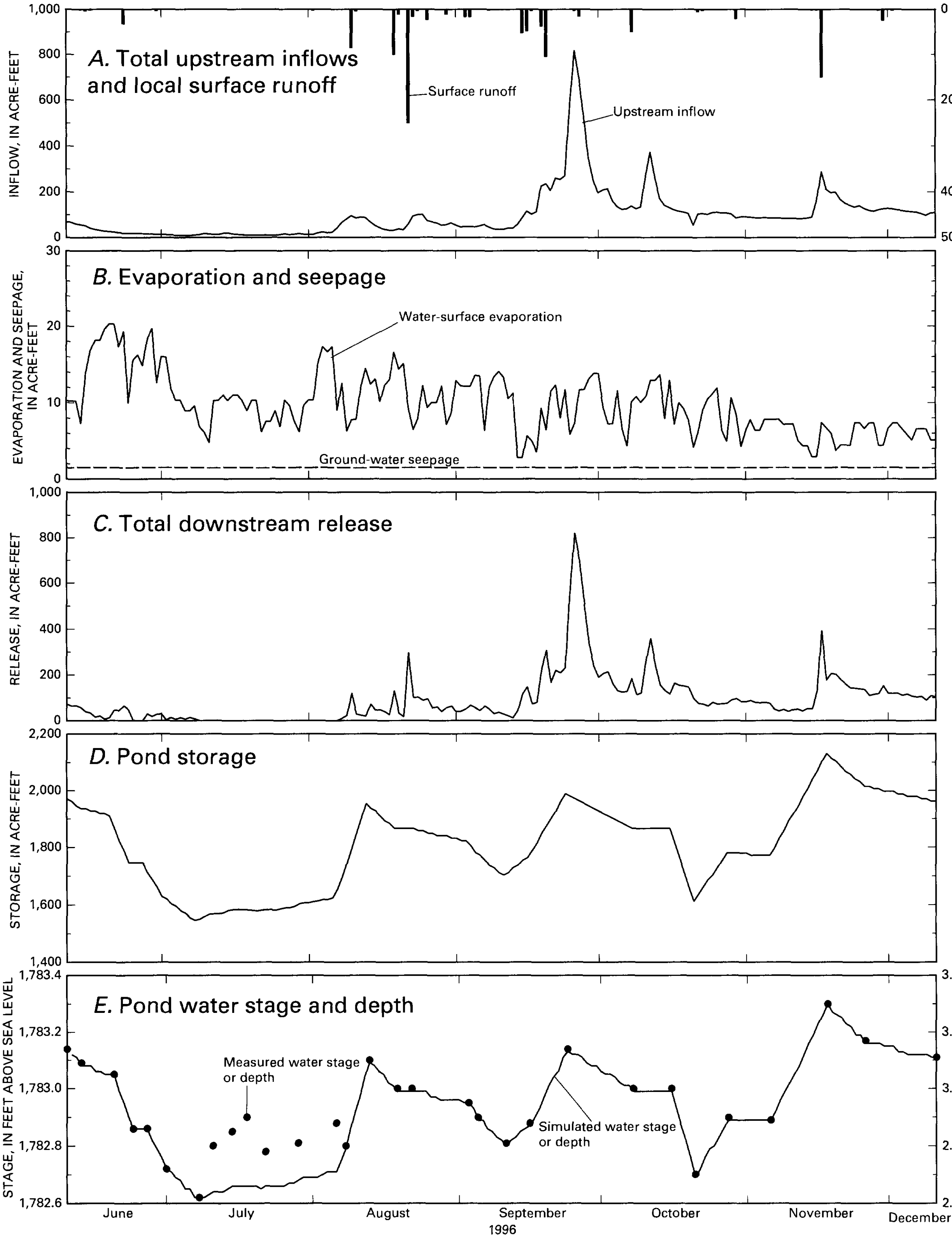



**Figure 12.** Water budget simulated for water unit 5, June 11 through December 11, 1996.

measured ones. The differences in stages were about 0.1 to 0.2 ft. The cause of these differences might be errors in reading stage and in estimating water-surface evaporation. The simulated water levels matched well with measured ones for the simulation period. The root mean square error (RMSE) between the simulated ($\hat{Z}$) and measured ($Z$) water levels was 0.08 ft for water unit 5 (see equation 56; *n* is the number of comparisons):

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}\left(\hat{z}_i - z\right)^2}{n}} . \qquad (56)$$

Similar results were found for other ponds. The RMSEs for other ponds were less than 0.13 ft except water units 24 and 30 for which RMSEs were 0.49 and 0.40 ft, respectively. In other words, the current specification for pond zoning and rule curves simulated the operation of ponds well.

Table 10 summarizes water-budget components of the ponds for the entire simulation period. For each pond, the water-balance equation was:

*Initial water storage + upstream inflow + surface runoff - water-surface evaporation - ground-water seepage - total downstream release = final storage,* (57)

where *upstream inflow* was the total inflow to a pond from upstream canals; *surface runoff* was the total runoff calculated using the measured precipitation data and SCS curve numbers; *water-surface evaporation* was the total surface-water evaporation loss, which was estimated in terms of the water-surface area and potential evapotranspiration coefficients; *ground-water seepage* was the total water loss to an aquifer (positive values) or total water gain from an aquifer (negative values); *downstream release* was the total amount of water released to downstream canals from a pond; and *final storage* was the water stored in a pond at the end of the simulation period. It was shown that equation 57 was satisfied for all ponds.

Another way to examine the water budget is by viewing a whole flow system as a "system node," which combines canals and control ponds with inflow from the Zenith node and outflows from nodes Raymond, water unit 11, and DCC (see figure 4). The overall water budget for the entire canal and control-pond system is summarized in table 11 for the entire simulation period. In table 11, initial storage was the summation of pond storage at the beginning of the simulation (13,102.68 acre-ft), which was interpreted from the measured water levels on June 10, 1996. Total stream inflow was the inflow from Rattlesnake Creek to water unit 5 (17,782.21 acre-ft), which was measured at the USGS streamflow-gaging station near Zenith. Local water gain to the system included surface runoff due to precipitation (6,559.04 acre-ft, 6,499.77 acre-ft of which were to ponds) and ground-water seepage to ponds (3,035.56 acre-ft) and was equal to 9,594.60 acre-ft. The outflow was the summation of outflows released from node Raymond (17,421.22 acre-ft) and from node water unit 11 (400.85 acre-ft). Total stream inflow to the system was almost the same as the stream outflow from the system. Although the total inflow to the system (stream inflow, runoff, and ground-water seepage to ponds) was much larger than the stream outflow from the system, the final water storage in the system was significantly reduced from the initial storage of 13,102.68 acre-ft to 9,211.88 acre-ft due to a large amount of local water loss through water-surface evaporation from ponds (10,683.74 acre-ft) and canal-flow transmission losses (2,761.79 acre-ft). The water loss due to water-surface evaporation was larger than the total local water gain within the refuge.

## SIMULATION OF CANAL AND CONTROL-POND OPERATION FOR 1991 WATER YEAR

A simulation was conducted to determine the operation of the system of canals and control ponds under drought flow conditions as occurred during the 1991 water year (October 1, 1990, through September 30, 1991) with different rule curves. Discharge during the 1991 water year was used as simulated discharge because this water year was the driest in terms of total discharges in Rattlesnake Creek for water years 1973 through 1995 (Putnam and others, 1996). In the following sections, the data needed to conduct the simulation and the necessary assumptions about these data are discussed, and then the simulation results are presented.

**Table 10.** Water budgets simulated for selected control ponds at Quivira National Wildlife Refuge, June 11 through December 11, 1996

[All values are in acre-feet; --, not applicable]

| Water-unit number (fig. 2) | Initial storage | + Total upstream inflow | + Total surface runoff | - Water-surface evaporation | - Ground-water seepage | - Total downstream release | = Final storage |
|---|---|---|---|---|---|---|---|
| 5 | 1,988.26 | 17,782.22 | 1,117.84 | 1,795.73 | 286.00 | 16,844.87 | 1,961.72 |
| 7 | 39.72 | 486.93 | 31.98 | 57.25 | 0 | 431.67 | 69.71 |
| 10A | 145.48 | 169.66 | 57.79 | 109.78 | 0 | 117.67 | 145.48 |
| 10B | 145.48 | 531.21 | 55.60 | 98.40 | 34.04 | 467.15 | 132.70 |
| 10C | 19.54 | 459.95 | 9.25 | 20.34 | 0 | 449.93 | 18.47 |
| 11 | 388.37 | 448.42 | 54.08 | 101.65 | 0 | 400.85 | 388.37 |
| 14A | 161.70 | 214.52 | 96.82 | 152.05 | 0 | 152.25 | 168.74 |
| 14B | 93.40 | 197.43 | 150.80 | 120.57 | -9.20 | 179.99 | 150.27 |
| 14C | 15.51 | 102.95 | 6.88 | 12.81 | 18.09 | 91.24 | 3.20 |
| 16 | 62.67 | 114.02 | 29.02 | 51.88 | 0 | 92.57 | 61.26 |
| 20A | 163.88 | 493.65 | 142.78 | 250.73 | 0 | 353.34 | 196.24 |
| 20B | 163.88 | 433.68 | 141.17 | 249.69 | 7.36 | 285.45 | 196.23 |
| 21 | 34.34 | 531.79 | 31.32 | 48.36 | 0 | 494.16 | 54.93 |
| 22 | 2.30 | 489.96 | 13.63 | 22.24 | 0 | 460.62 | 23.03 |
| 23 | 15.41 | 458.46 | 10.26 | 17.77 | 0 | 448.68 | 17.68 |
| 24 | 132.55 | 12,741.83 | 63.05 | 93.63 | 103.47 | 12,666.91 | 73.42 |
| 25 | 18.00 | 676.23 | 55.85 | 72.37 | 40.89 | 494.09 | 142.73 |
| 26 | 91.48 | 862.43 | 63.69 | 93.92 | 7.25 | 829.71 | 86.72 |
| 28 | 6.11 | 635.03 | 44.60 | 69.63 | 0 | 497.53 | 118.58 |
| 29 | .20 | 482.72 | 38.38 | 53.35 | 0 | 404.38 | 63.57 |
| 30 | 2.82 | 396.78 | 62.17 | 97.04 | 0 | 196.68 | 168.05 |
| 40 | 55.91 | 77.90 | 14.03 | 34.19 | -36.27 | 83.58 | 66.34 |
| 48 | 3.94 | 316.42 | 73.47 | 72.85 | 0 | 245.36 | 75.62 |
| 49 | 51.63 | 567.52 | 64.72 | 102.76 | 11.87 | 437.18 | 132.06 |
| 57 | 212.22 | 1,193.02 | 156.57 | 257.91 | 0 | 1,102.51 | 201.39 |
| 58 | 146.39 | 1,388.52 | 83.47 | 135.49 | -86.07 | 1,429.80 | 139.16 |
| 61 | 212.80 | 340.11 | 123.31 | 209.47 | -50.68 | 380.83 | 136.60 |
| 62 | 48.58 | 59.57 | 17.64 | 33.97 | -31.89 | 57.62 | 66.09 |
| 63 | 268.98 | 132.23 | 129.33 | 232.08 | -76.44 | 243.73 | 131.17 |
| 75 | 2,445.85 | 1,177.96 | 1,445.34 | 2,043.86 | -2,484.88 | 4,004.79 | 1,505.38 |
| 78 | 5,270.43 | 5,359.15 | 1,728.99 | 3,161.14 | -291.63 | 7,252.18 | 2,236.88 |
| 80 | 355.20 | 7,542.65 | 155.93 | 363.20 | -85.77 | 7,705.34 | 71.01 |
| 81 | 25.31 | 7,252.18 | 41.67 | 94.73 | -323.22 | 7,542.65 | 5.00 |
| 83 | 314.34 | 0 | 188.34 | 352.91 | -68.48 | 14.14 | 204.11 |
| **Total** | **13,102.68** | -- | **6,499.77** | **10,683.75** | **-3,035.56** | -- | **9,211.91** |

**Table 11.** Water budget simulated for entire canal and control-pond system at Quivira National Wildlife Refuge, June 11 through December 11, 1996

[All values are in acre-feet; --, not applicable]

| Water-budget component | Storage | Inflow | Outflow |
|---|---|---|---|
| Initial storage | 13,102.68 | -- | -- |
| Stream inflow | -- | 17,782.21 | -- |
| Surface runoff | -- | 6,559.04 | -- |
| Water-surface evaporation | -- | -- | 10,683.74 |
| Net ground-water seepage | -- | 3,035.56 | -- |
| Canal-flow transmission loss | -- | -- | 2,761.79 |
| Outflow from Raymond node | -- | -- | 17,822.07 |
| **Final storage** | **9,211.88** | -- | -- |

## Data Preparation

In this section, data needed for the simulation are discussed. Measurement data were used if available. If some data were not available, reasonable values were interpreted on the basis of other related data.

### Precipitation

One of the major factors affecting the generation of direct overland surface runoff to ponds is the amount of precipitation. Figure 13*A* shows the daily precipitation measured at the Sandyland Experiment Station and at the USGS streamflow-gaging station near Zenith (fig. 1). Precipitation data from October 1, 1990, through May 20, 1991, were measured at the Sandyland Experiment Station. Precipitation data from May 21 through September 30, 1991, were measured at the USGS streamflow-gaging station near Zenith. The total amount of precipitation during the 1991 water year was 13.43 in.

### Water-Surface Evaporation

The daily potential evapotranspiration (PET) estimated with the Penman method using the climatic data collected at the Sandyland Experiment Station (Marios Sophocleous, Kansas Geological Survey, written commun., 1996) is shown in figure 13*B*. The total amount of PET was 61.23 in. for the 1991 water year. For the 1991 water year simulation, it was assumed that the daily water-surface evaporation rate for ponds on the refuge was equal to the corresponding daily potential evapotranspiration at the Sandyland Experimental Station.

### Canal Discharge

Discharges for Rattlesnake Creek measured at the USGS streamflow-gaging stations near Zenith and Raymond (fig. 1) from October 1, 1990, through September 30, 1991, are shown in figure 13*C* (Geiger and others, 1992). The mean daily discharges for the Zenith and Raymond stations during the 1991 water year were 6.59 and 2.77 $ft^3/s$, respectively, which are much smaller than the long-term means of 50.6 $ft^3/s$ (1973–95 water years) and 48.8 $ft^3/s$ (1960–95 water years), respectively. As shown in the figure 13*C*, there was almost no flow during late September 1991.

For this simulation, the daily mean discharge observed at the USGS streamflow-gaging station near Zenith station was used as daily inflows to Little Salt Marsh from Rattlesnake Creek. The daily mean discharge observed at the USGS streamflow-gaging station near Raymond was used as the streamflow requirement for Rattlesnake Creek near Raymond.

### Canal-Flow Transmission Losses

Canal-flow transmission loss was difficult to estimate. Because there were no data available to estimate the canal-flow transmission loss coefficient for the canals on the refuge during the simulation period, the estimated transmission loss coefficient ($k$ in equation 42) of $9.16 \times 10^{-6}$ $ft^{-1}$ for the 1996 simulation period was used for this simulation. Similar to 1996, canal-flow transmission losses occurred only in canals south of the RC Canal.

### Ground-Water Discharge to Ponds

No monthly data for ground-water discharge to ponds were available for the simulation period. The study conducted using MODFLOW by Marios Sophocleous (Kansas Geological Survey, written commun., 1996) shows that the amount of annual ground-water discharge to ponds on the refuge was almost the same from 1975 through 1990. Consequently, the monthly ground-water-discharge data obtained from Marios Sophocleous (Kansas Geological Survey, written commun., 1997) for 1994 were used (see table 2).

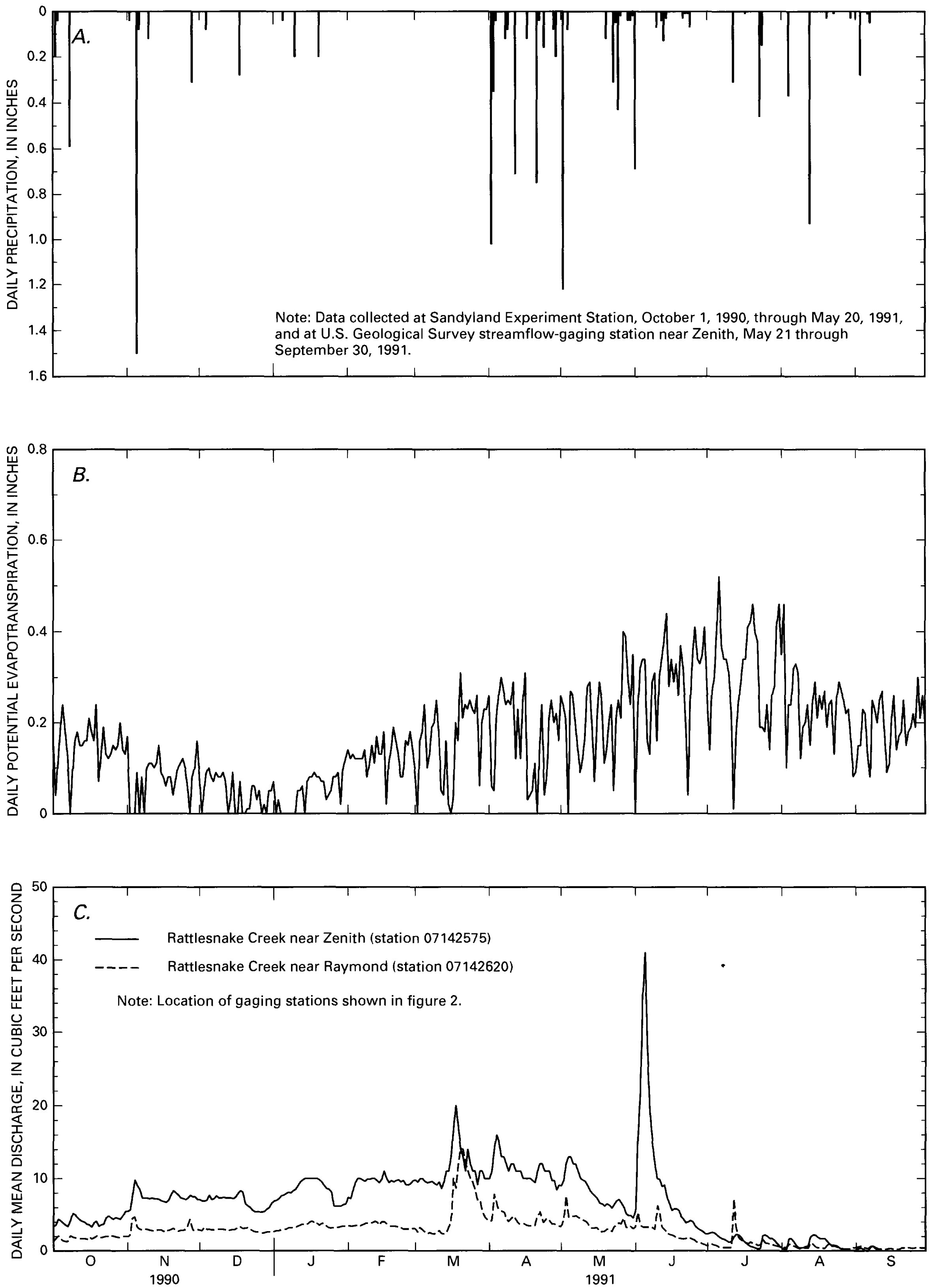



**Figure 13.** *(A)* Daily precipitation, *(B)* daily potential evapotranspiration, and *(C)* daily mean discharge for Rattlesnake Creek, 1991 water year. Part of the precipitation data and potential evapotranspiration data are from the Kansas Geological Survey (Marios Sophocleous, written commun., 1996), and the discharge data are from Geiger and others (1992).

### Initial Water Storage in Ponds

The amount of initial water storage in the control ponds affects the operation of the ponds and the final water budgets. The amount of water stored in the control ponds on September 30, 1990, was not known. The 1991 water year simulation was used to evaluate the daily operation of ponds with different rule curves during drought conditions. Therefore, the initial water storage in a pond was simply set at 80 percent of full-pond capacity for the 1991 simulation (see table 12) to be consistent with the study by Marios Sophocleous (Kansas Geological Survey, written commun., 1996).

## Pond Zoning and Operating Policy

Each control pond was divided into four storage zones—inactive zone, lower zone, upper zone, and extended upper zone as shown in figure 5. Normal operating storage of a pond consisted of water stored between the lower zone and upper zone and was set between 20 and 100 percent of full-pond storage capacity for this simulation. In the other words, the lower boundary of the lower storage zone was set at 20 percent of full-pond capacity, and the top boundary of the upper storage zone was set at 100 percent of full-pond capacity. The rule curve was set within this operating storage range. Four different rule curves corresponding to different simulations were set at 60, 70, 80, and 90 percent of full-pond capacity, respectively. The capacity of the inactive zone of a pond was set at 20 percent of full-pond storage capacity (selected in consultation with the U.S. Fish and Wildlife Service). The top boundary of the extended upper zone was set 0.5 ft higher than corresponding full-pond capacity. Pond zoning expressed as pond storage along with the rule curve at 90 percent of full-pond capacity are summarized in table 12.

The priority of pond operation for the 1991 water year was the same as for 1996 (see the discussion of pond priority for the 1996 simulation). Typical values for penalty coefficients used in the 1991 water year simulation are also summarized in table 12. The canal-flow zoning and the assignment of penalty coefficients were the same as those used in the simulation for 1996 (see table 9).

## Results

Four different simulations of canal and control-pond operation at the refuge were conducted with the rule curve of a pond set at 60, 70, 80, and 90 percent of full-pond capacity, respectively. Other specifications for pond zoning and canal outflows from the refuge were (1) the initial storage of a pond was set at the 80 percent of full-pond capacity; (2) the inactive storage of a pond was set at the 20 percent of full-pond capacity; and (3) outflows of Rattlesnake Creek near the USGS streamflow-gaging station near Raymond were fixed and equal to the discharges observed for the 1991 water year.

Results of operating the canals and control ponds using a rule curve of 90 percent of full-pond capacity are described first. The simulated water budget for water unit 5 is shown in figure 14. Figure 14*A* shows the inflows from the upstream Zenith node to water unit 5 (also see figure 4), which are equal to the discharges observed at the USGS streamflow-gaging station near Zenith. The total releases to all downstream nodes (water units 7 and 10A, and canal joints C–2 and JE–1) from unit 5 are shown in figure 14*C*. The simulated pond water stage corresponding to water storage (fig. 14*D*) is shown in figure 14*E*. Similar figures could also be generated for the remaining control ponds. These figures reflect the operation of a single pond during an entire simulation period with the current operating policy. These figures also can be used to evaluate whether some specifications, such as the target water level, in the operating policy are satisfied. Water storage after mid-June 1991 decreased and reached the inactive zone (fig. 14*D*) and could not be maintained at the target level due to insufficient inflow and water-surface evaporation. In other words, if the target level in water unit 5 was set too low, water unit 5 could be dry at the end of the period under inflow conditions that were simulated.

To show the water budget of a control pond during the simulation period, table 13 summarizes water-budget components of ponds with the rule curve at 90 percent of full-pond capacity. It is seen from table 13 that the final storage value for all ponds at the end of the simulation period was much smaller than the initial storage values. Many small ponds were dry at the end of the simulation period. Total water-surface evaporation for all ponds was much larger than other water-budget components (runoff, ground-water seepage).

**Table 12.** Initial storage, rule curve, zoning, and penalty coefficients assigned to control ponds at Quivira National Wildlife Refuge, 1991 water year

[acre-ft, acre-feet]

| Water-unit number (fig. 2) | Initial storage at 80 percent of full-pond capacity (acre-ft) | Rule curve at 90 percent of full-pond capacity (acre-ft) | Upper boundary of extended upper zone (acre-ft) | Penalty coefficient for extended upper zone | Upper boundary of upper zone (acre-ft) | Penalty coefficient for upper zone | Lower boundary of lower zone (acre-ft) | Penalty coefficient for lower zone | Lower boundary of inactive zone (acre-ft) | Penalty coefficient for inactive zone |
|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 1,492.80 | 1,679.40 | 2,312.18 | 3,000 | 1,866.00 | 2,000 | 373.20 | 3,000 | 0 | 6,000 |
| 7 | 32.00 | 36.00 | 54.49 | 3,000 | 40.00 | 1,500 | 8.00 | 1,500 | 0 | 3,000 |
| 10A | 116.00 | 130.50 | 180.30 | 3,000 | 145.00 | 1,500 | 29.00 | 1,500 | 0 | 3,000 |
| 10B | 116.00 | 130.50 | 180.30 | 3,000 | 145.00 | 1,500 | 29.00 | 1,500 | 0 | 3,000 |
| 10C | 10.40 | 11.70 | 14.61 | 3,000 | 13.00 | 1,500 | 2.60 | 1,500 | 0 | 3,000 |
| 11 | 270.40 | 304.20 | 413.90 | 3,000 | 338.00 | 1,500 | 67.60 | 1,500 | 0 | 3,000 |
| 14A | 156.80 | 176.40 | 242.20 | 3,000 | 196.00 | 1,500 | 39.20 | 1,500 | 0 | 3,000 |
| 14B | 76.80 | 86.40 | 144.01 | 3,000 | 96.00 | 1,500 | 19.20 | 1,500 | 0 | 3,000 |
| 14C | 12.80 | 14.40 | 19.07 | 3,000 | 16.00 | 1,500 | 3.20 | 1,500 | 0 | 3,000 |
| 16 | 64.00 | 72.00 | 96.07 | 3,000 | 80.00 | 1,500 | 16.00 | 1,500 | 0 | 3,000 |
| 20A | 156.00 | 175.50 | 268.01 | 3,000 | 195.00 | 1,500 | 39.00 | 1,500 | 0 | 3,000 |
| 20B | 156.00 | 175.50 | 268.01 | 3,000 | 195.00 | 1,500 | 39.00 | 1,500 | 0 | 3,000 |
| 21 | 64.80 | 72.90 | 96.62 | 3,000 | 81.00 | 1,500 | 16.20 | 1,500 | 0 | 3,000 |
| 22 | 10.40 | 11.70 | 18.81 | 3,000 | 13.00 | 1,500 | 2.60 | 1,500 | 0 | 3,000 |
| 23 | 12.00 | 13.50 | 19.66 | 3,000 | 15.00 | 1,500 | 3.00 | 1,500 | 0 | 3,000 |
| 24 | 28.00 | 31.50 | 53.04 | 3,000 | 35.00 | 1,500 | 7.00 | 1,500 | 0 | 3,000 |
| 25 | 236.80 | 266.40 | 344.05 | 3,000 | 296.00 | 1,500 | 59.20 | 1,500 | 0 | 3,000 |
| 26 | 88.80 | 99.90 | 142.39 | 3,000 | 111.00 | 1,500 | 22.20 | 1,500 | 0 | 3,000 |
| 28 | 122.40 | 137.70 | 198.82 | 3,000 | 153.00 | 1,500 | 30.60 | 1,500 | 0 | 3,000 |
| 29 | 72.80 | 81.90 | 124.51 | 3,000 | 91.00 | 1,500 | 18.20 | 1,500 | 0 | 3,000 |
| 30 | 95.20 | 107.10 | 161.64 | 3,000 | 119.00 | 1,500 | 23.80 | 1,500 | 0 | 3,000 |
| 40 | 52.80 | 59.40 | 83.19 | 3,000 | 66.00 | 1,500 | 13.20 | 1,500 | 0 | 3,000 |
| 48 | 90.40 | 101.70 | 161.19 | 3,000 | 113.00 | 1,500 | 22.60 | 1,500 | 0 | 3,000 |
| 49 | 127.20 | 143.10 | 209.05 | 3,000 | 159.00 | 1,500 | 31.80 | 1,500 | 0 | 3,000 |
| 57 | 169.60 | 190.80 | 280.74 | 3,000 | 212.00 | 1,500 | 42.40 | 1,500 | 0 | 3,000 |
| 58 | 200.80 | 225.90 | 302.82 | 3,000 | 251.00 | 1,500 | 50.20 | 1,500 | 0 | 3,000 |
| 61 | 398.40 | 448.20 | 613.17 | 3,000 | 498.00 | 1,500 | 99.60 | 1,500 | 0 | 3,000 |
| 62 | 96.00 | 108.00 | 145.00 | 3,000 | 120.00 | 1,500 | 24.00 | 1,500 | 0 | 3,000 |
| 63 | 271.20 | 305.10 | 419.01 | 3,000 | 339.00 | 1,500 | 67.80 | 1,500 | 0 | 3,000 |
| 75 | 1,956.80 | 2,201.40 | 3,490.32 | 3,000 | 2,446.00 | 1,500 | 489.20 | 1,500 | 0 | 3,000 |
| 78 | 4,216.00 | 4,743.00 | 6,091.37 | 2,000 | 5,270.00 | 1,000 | 1,054.00 | 1,000 | 0 | 3,000 |
| 80 | 284.00 | 319.50 | 474.34 | 2,000 | 355.00 | 1,000 | 71.00 | 1,000 | 0 | 3,000 |
| 81 | 20.00 | 22.50 | 60.68 | 2,000 | 25.00 | 1,000 | 5.00 | 1,000 | 0 | 3,000 |
| 83 | 251.20 | 282.60 | 419.31 | 3,000 | 314.00 | 1,500 | 62.80 | 1,500 | 0 | 3,000 |

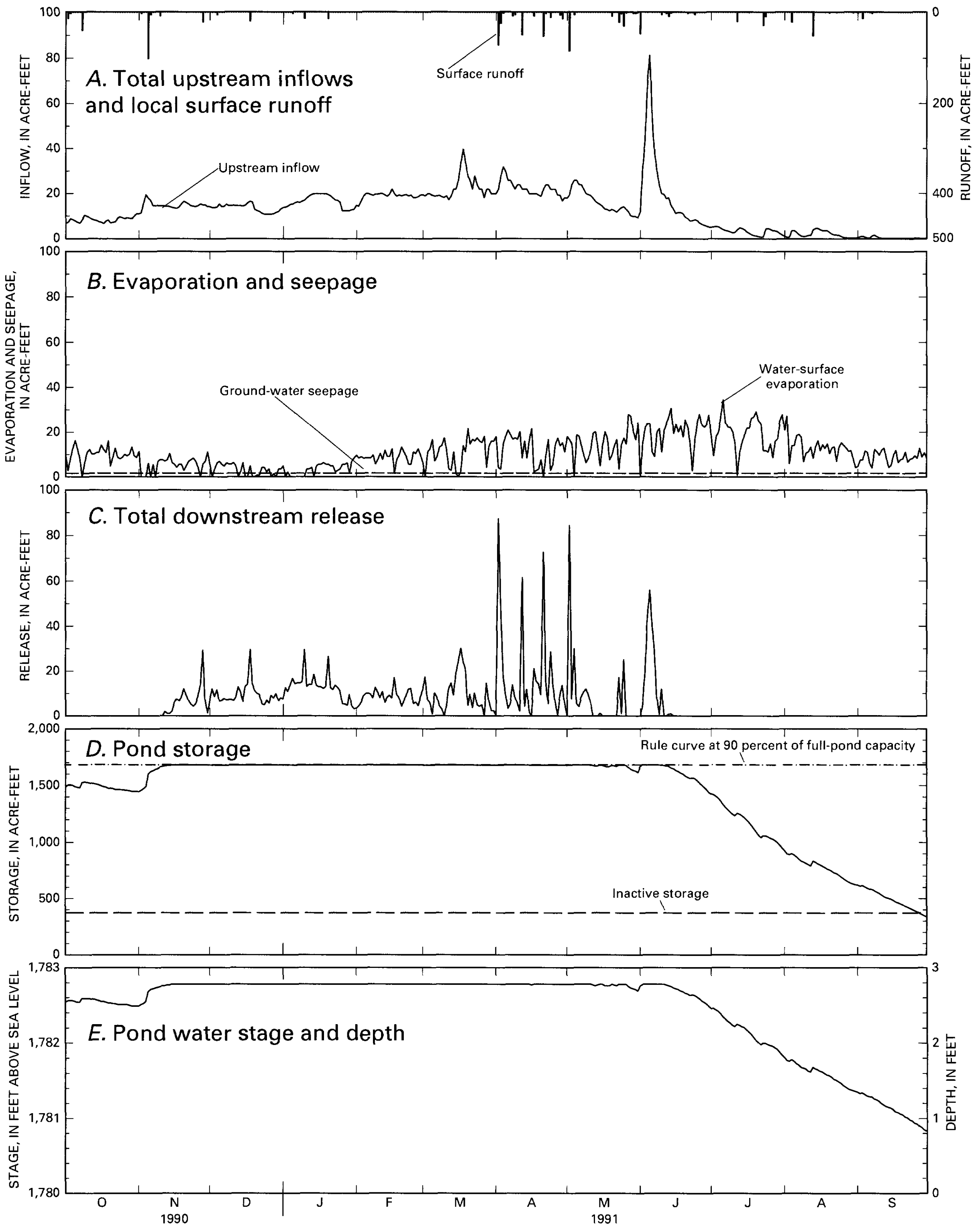



**Figure 14.** Water budget simulated for water unit 5, 1991 water year.

**Table 13.** Water budgets simulated for control ponds at Quivira National Wildlife Refuge with rule curve at 90 percent of full-pond capacity, 1991 water year

[All values are in acre-feet; --, not applicable]

| Water-unit number (fig. 2) | Initial storage at 80 percent of full-pond capacity + | Total upstream inflow + | Total surface runoff - | Water-surface evaporation - | Ground-water seepage[1] - | Total downstream release = | Final storage |
|---|---|---|---|---|---|---|---|
| 5 | 1,492.80 | 4,772.63 | 900.16 | 3,970.32 | 545.66 | 2,312.24 | 337.37 |
| 7 | 32.00 | 27.62 | 17.03 | 60.87 | 0 | 15.78 | 0 |
| 10A | 116.00 | 81.59 | 48.37 | 196.59 | 0 | 41.33 | 8.04 |
| 10B | 116.00 | 56.16 | 31.97 | 132.51 | 64.84 | 6.78 | 0 |
| 10C | 10.40 | 6.68 | 5.53 | 22.61 | 0 | 0 | 0 |
| 11 | 270.40 | 0 | 41.11 | 175.91 | 0 | 0 | 135.60 |
| 14A | 156.80 | 117.93 | 67.12 | 252.34 | 0 | 80.85 | 8.66 |
| 14B | 76.80 | 64.16 | 44.27 | 171.41 | -18.25 | 27.51 | 4.56 |
| 14C | 12.80 | 56.63 | 5.13 | 17.70 | 24.97 | 31.89 | 0 |
| 16 | 64.00 | 48.64 | 19.21 | 72.22 | 0 | 54.60 | 5.03 |
| 20A | 156.00 | 100.37 | 78.10 | 303.93 | 0 | 29.97 | .57 |
| 20B | 156.00 | 196.85 | 96.66 | 380.01 | 13.88 | 55.62 | 0 |
| 21 | 64.80 | 51.83 | 22.11 | 85.13 | 0 | 50.50 | 3.11 |
| 22 | 10.40 | 50.07 | 6.45 | 23.09 | 0 | 43.83 | 0 |
| 23 | 12.00 | 43.62 | 6.40 | 23.31 | 0 | 38.71 | 0 |
| 24 | 28.00 | 1,541.96 | 17.28 | 62.83 | 149.33 | 1,375.08 | 0 |
| 25 | 236.80 | 94.39 | 73.72 | 268.01 | 75.04 | 61.85 | .01 |
| 26 | 88.80 | 99.09 | 30.52 | 121.74 | 13.50 | 83.17 | 0 |
| 28 | 122.40 | 61.71 | 44.38 | 172.26 | 0 | 53.03 | 3.20 |
| 29 | 72.80 | 51.45 | 31.35 | 123.71 | 0 | 30.83 | 1.06 |
| 30 | 95.20 | 30.25 | 42.70 | 167.16 | 0 | .99 | 0 |
| 40 | 52.80 | 0 | 30.79 | 134.70 | -86.83 | 8.19 | 27.53 |
| 48 | 90.40 | 35.50 | 43.20 | 167.99 | 0 | .12 | .99 |
| 49 | 127.20 | 45.13 | 52.38 | 205.68 | 19.03 | 0 | 0 |
| 57 | 169.60 | 343.12 | 114.11 | 485.65 | 0 | 102.75 | 38.43 |
| 58 | 200.80 | 362.88 | 92.46 | 394.82 | -173.34 | 390.50 | 44.16 |
| 61 | 398.40 | 565.28 | 208.54 | 902.49 | -109.08 | 283.94 | 94.87 |
| 62 | 96.00 | 0 | 40.37 | 173.96 | -70.19 | 3.15 | 29.45 |
| 63 | 271.20 | 189.92 | 150.50 | 650.00 | -161.39 | 59.07 | 63.94 |
| 75 | 1,956.80 | 254.24 | 1,478.91 | 6,103.74 | -5,002.59 | 1,642.84 | 945.96 |
| 78 | 4,216.00 | 1,881.86 | 1,262.39 | 5,406.83 | -587.57 | 1,486.99 | 1,054.00 |
| 80 | 284.00 | 2,028.75 | 147.51 | 646.47 | -172.74 | 1,915.53 | 71.00 |
| 81 | 20.00 | 1,486.99 | 37.92 | 162.60 | -651.44 | 2,028.75 | 5.00 |
| 83 | 251.20 | 0 | 133.69 | 522.11 | -137.75 | 0 | .53 |
| **Total** | **11,525.60** | -- | **5,422.34** | **22,760.70** | **-6,264.92** | -- | **2,883.07** |

[1]The positive values of ground-water seepage indicate that ponds lost water to the aquifer. The negative values of ground-water seepage indicate that ponds gained water from the aquifer.

**Table 14.** Water budget simulated for entire canal and control-pond system at Quivira National Wildlife Refuge with rule curve at 90 percent of full-pond capacity, 1991 water year

[All values are in acre-feet; --, not applicable]

| Water-budget component | Storage | Inflow | Outflow |
|---|---|---|---|
| Initial storage | 11,525.60 | -- | -- |
| Stream inflow | -- | 4,772.63 | -- |
| Surface runoff | -- | 5,422.34 | -- |
| Water-surface evaporation | -- | -- | 22,760.70 |
| Net ground-water seepage | -- | 6,264.92 | -- |
| Canal-flow transmission loss | -- | -- | 336.51 |
| Outflow from Raymond node | -- | -- | 2,005.24 |
| **Final storage** | **2,883.07** | **--** | **--** |

To examine the water budget for the whole flow system at the refuge, table 14 summarizes the overall water budget for the entire canal and control-pond system with the rule curve set at 90 percent of full-pond capacity. It can be seen from this table that although there were total inflows of 16,459.89 acre-ft, of which 4,772.63 acre-ft were from Rattlesnake Creek at Zenith node, 5,422.34 acre-ft from direct surface runoff, and 6,264.92 acre-ft from the ground-water seepage to ponds, the final water storage in the system was substantially reduced from the initial storage of 11,525.60 acre-ft, which was set at 80 percent of full-pond capacity, to 2,883.07 acre-ft due to the outflows from the Raymond node, water-surface evaporation, and canal-flow transmission loss. Total water out of the system (outflow, evaporation, and canal-flow transmission loss) from the system was 25,102.45 acre-ft, of which 22,760.70 acre-ft (or 91 percent of water outflow from the system) was due to water-surface evaporation. At the end of simulation period, 30 out of 34 ponds, including water unit 5, had water stored only in the inactive zone or were dry due to the large amount of water-surface evaporation.

To compare the operation of canal and control ponds with the rule curve at 90 percent of full-pond capacity, simulations were also conducted with the rule curves at 80, 70, and 60 percent of full-pond capacity. All of simulations were conducted with the same model specification except for the rule curves.

Figure 15 shows the change in water storage for water units 5 and 78, respectively, with different rule curves. As the rule curve was reduced from 90 to 60 percent of full-pond capacity, water storage in water unit 5 during the simulation period decreased, and the final pond storage was also reduced from 337 to 48 acre-ft (fig. 15*A*). Because water unit 5 had the highest priority and because the initial storage was higher than the rule curve, water was released immediately downstream as shown in figure 15*A*. On the other hand, water storage in water unit 78 increased during the simulation period (fig. 15*B*). Because water unit 78 had the lowest priority and because water storage in the upstream higher priority ponds was in the upper zone, water was released from these higher priority ponds to maintain their rule curves, and water released from the upstream pond was stored in the unit 78, which caused the water storage to reach full-pond capacity (fig. 15*B*). After mid-June 1991, there were not enough inflow (upstream inflow plus surface runoff) to water unit 5 to maintain water levels at the rule curve, and water levels decreased due to water-surface evaporation. At the end of simulation, the water level in water unit 5 was located in the inactive zone (figs. 14 and 15*A*). Similar changes in water storages were also observed for other control ponds.

The simulated water budget for the entire canal and control-pond system for the 1991 water year with different rule curves is summarized in table 15. As the rule curves were reduced from 90 to 60 percent of full-pond capacity, surface runoff, water-surface evaporation, and ground-water seepage from ponds were reduced, and stream outflow and final storage increased (see table 15). The reduction of the rule curve of a pond generally caused a lower pond water level to be maintained for the higher priority ponds. In other words, the total water-surface evaporation and rainfall onto the water-surface area of a pond were reduced for the same evaporation rate and precipitation depth. When initial pond storage was higher than the rule curve (initial storage was set at 80 percent of full-pond capacity), water was released from the ponds with higher priority to meet the rule-curve water level, which caused more canal-flow transmission losses along the canals in the south part of the refuge and increased outflows from water unit 11. The final pond water storage also increased due to storage increases in water unit 78 (fig.15*B*) and other ponds in the north part of the refuge.

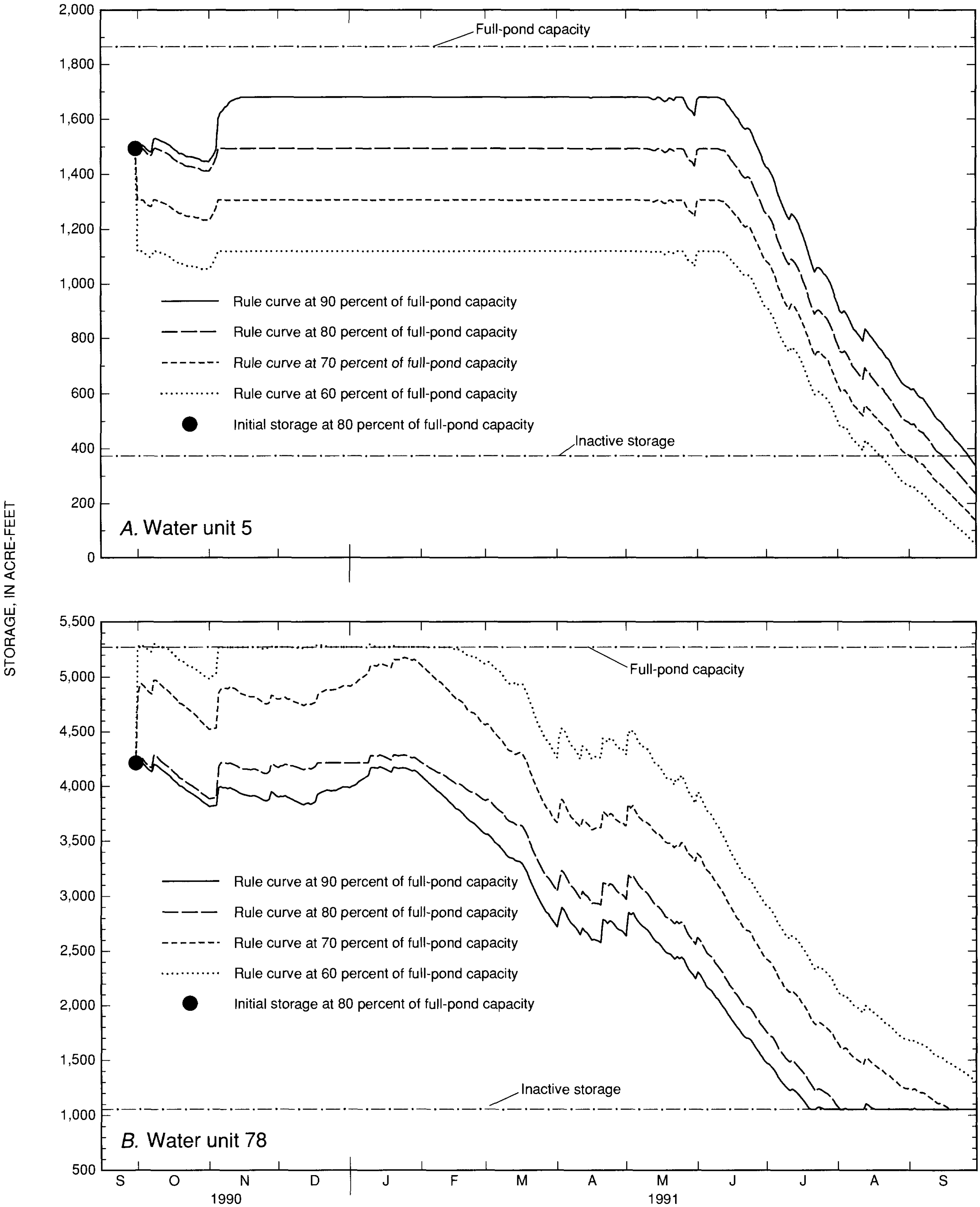



**Figure 15.** Simulated pond water storage with different rule curves for *(A)* water unit 5 and *(B)* water unit 78, with initial storage at 80 percent of full-pond capacity, 1991 water year.

**Table 15.** Water budgets simulated for entire canal and control-pond system at Quivira National Wildlife Refuge using different rule curves, 1991 water year

[all values are in acre-feet]

| Water-budget component | Rule curve set at 90 percent of full-pond capacity | Rule curve set at 80 percent of full-pond capacity | Rule curve set at 70 percent of full-pond capacity | Rule curve set at 60 percent of full-pond capacity |
|---|---|---|---|---|
| Initial pond storage | 11,525.60 | 11,525.60 | 11,525.60 | 11,525.60 |
| **Inflow:** | | | | |
| Stream inflow from Rattlesnake Creek | 4,772.63 | 4,772.63 | 4,772.63 | 4,772.63 |
| Surface runoff | 5,422.34 | 5,393.31 | 5,260.83 | 4,981.23 |
| Net ground-water seepage, including canal-flow transmission loss | -5,928.41 | -5,900.37 | -5,854.23 | -5,854.04 |
| **Outflow:** | | | | |
| Water-surface evaporation | 22,760.70 | 22,694.41 | 22,238.25 | 21,299.74 |
| Total outflow | 2,005.24 | 2,005.28 | 2,073.30 | 2,547.26 |
| **Final pond storage** | **2,883.07** | **2,892.22** | **3,101.74** | **3,286.50** |

Simulation results for the 1991 water year indicate that water-surface evaporation was the major factor in lowering water storage in ponds. Storing more water in the ponds in the north part of the refuge by reducing the rule curve for higher priority ponds may reduce the overall water-surface evaporation. However, this will also cause water unit 5 to dry out quickly if there is not enough upstream inflow as was the case during the 1991 water year. Maintaining high water levels in water unit 5 depends upon the rule curve in water unit 5 being set at a high level. The simulation results discussed for the 1991 water year were obtained based on a number of assumptions, such as the initial storage in ponds. If the specifications for the simulation model change, the results may be much different.

## SUMMARY

In 1995, a 3-year study was undertaken to develop a water budget and flow-routing model to assist the U.S. Fish and Wildlife Service in determining the outcome of possible water-management options at the Quivira National Wildlife Refuge, south-central Kansas. The study was done by the U.S. Geological Survey in cooperation with the Kansas Geological Survey. A computer program OPONDS, written in FORTRAN, was developed using network flow analysis to determine the optimal operation of a system of canals and control ponds. Applications of the model are presented that investigate the daily operation of canals and control ponds on the refuge using historical discharge and pond water levels.

The daily operation of a system of canals and control ponds at the refuge in the Rattlesnake Creek Basin was simulated for June 11 through December 11, 1996, using a linear-network flow model. In this simulation, some management requirements included the measured water levels of control ponds as the target management pond levels and the observed stream discharges in Rattlesnake Creek near Raymond as the outflow requirement from the refuge. Measured precipitation and calculated potential evapotranspiration were used to compute the surface runoff to ponds and water-surface evaporation, respectively. The operating policy was determined by using selected storage zones within a pond and prioritization of the ponds by using the relative magnitude of penalty coefficients within the computer model to adjust pond storages and canal flows. Results of the 1996 simulation indicate that the current specification for pond zoning and rule curves, with water unit 5 given the highest priority and ponds in the north part of the refuge given the lowest priorities, simulated pond levels that matched well with observed ones. Root mean square errors between simulated and measured water levels were less than 0.13 ft except for water units 24 and 30. Water storage in ponds during the simulation period was substantially reduced due to water-surface evaporation and canal-flow transmission losses.

Simulation of canal and control-pond operation under drought conditions during the 1991 water year was also conducted with different target pond water levels. This simulation used 1991 measured stream discharges, precipitation, and potential evapotranspiration data and 1994 ground-water seepage to ponds to investigate the operation of the canals and control ponds. The operating policy used four pond storage zones and the prioritization of ponds, with water unit 5 having the highest priority and ponds in the north part of the refuge having the lowest priority. Results showed that under the same initial water storage of 80 percent of full-pond capacity lowering target pond water levels reduced water-surface evaporation, resulted in more water stored in ponds in the north part of the refuge, and caused a substantial decrease in the final water storage in water unit 5. In other words, to maintain high water storage in water unit 5, the target water level in this unit should be high. To reduce the total water-surface evaporation loss, the target water level should be low for unit 5 so that water is stored in the ponds in the north part of the refuge. It should be noted that results of the 1991 water year simulation were obtained with the same initial storage of ponds and measured discharges of Rattlesnake Creek near Raymond as the 1996 simulation. The optimal operation of a system of canals and control ponds depends on having a well-defined operating policy and accurate data and may require several combinations of model specifications to obtain optimum results.

The OPONDS model can be applied to other operations at the Quivira Refuge simply by modifying the conceptual flow-network configuration and changing the operating policy through pond-storage and canal-flow zoning and corresponding penalty coefficients. The OPONDS model can be applied to operational matters such as the determination of target water levels and pond water releases, the operation of the outlet structures, and canal flow and routing.

The OPONDS model is a simplification of a complex canal-pond network flow system and is limited in simulating the operation of the flow system by the accuracy of data used in the model and some assumptions. Nonetheless, the OPONDS model is a useful tool for estimating the effects of possible water-management options for the Quivira National Wildlife Refuge.

## REFERENCES

Bazaraa, M.S., Jarvis, J.J., and Sherali, H.D., 1990, Linear programming and network flows (2nd ed.): New York, John Wiley & Sons, 684 p.

Chengdu Science and Technology University, 1979, Hydraulics: Chengdu, China, Chengdu Science and Technology University, 512 p.

Chow, V.T., Maidment, D.R., and Mays, L.W., 1988, Applied hydrology: New York, McGraw-Hill, Inc., 572 p.

Fulkerson, D.R., 1961, An out-of-kilter method for minimal cost flow problem: Journal of the Society for Industrial and Applied Mathematics, v. 9, p.18–27.

Geiger, C.O, Lacock, D.L., Schneider, D.R., Carlson, M.D., and Pabst, B.J., 1992, Water resources data, Kansas, water year 1991: U.S. Geological Survey Water-Data Report KS–91–1, 358 p.

Henderson, F.M., 1966, Open channel flow: New York, MacMillan Publ. Co., Inc., 522 p.

Hulsing, Harry, 1967, Measurement of peak discharge at dams by indirect method: U.S. Geological Survey Techniques of Water-Resources Investigations, book 3, chap. A5, 29 p.

Jian, Xiaodong, 1988, Simulation and optimization of the operation of the Kansas River Basin reservoirs: Lawrence, University of Kansas, unpublished master's thesis, 121 p.

Jordan, P.R., 1977, Streamflow transmission losses in western Kansas: American Society of Civil Engineers, Journal of the Hydraulic Division, v. 103, no. HY8, p. 905–919.

McCuen, R.H., 1989, Hydrologic analysis and design: Englewood Cliffs, New Jersey, Prentice-Hall, 867 p.

Putnam, J.E., Lacock, D.L., Schneider, D.R., Carlson, M.D., and Dague, B.J., 1996, Water resources data, Kansas, water year 1995: U.S. Geological Survey Water-Data Report KS–95–1, 488 p.

Soil Conservation Service, 1985, SCS national engineering handbook, section 4, hydrology: U.S. Department of Agriculture, various pages.

Sophocleous, Marios, and Perkins, S.P., 1992, Stream-aquifer and mineral intrusion modeling of the lower Rattlesnake Creek basin with emphasis on the Quivira National Wildlife Refuge: Kansas Geological Survey Open-File Report 92–6, 205 p.

Swamee, P.K., 1992, Sluice-gate discharge equations: Journal of Irrigation and Drainage Engineering, v. 118, no. 1, p. 56–60.

U.S. Army Engineer Waterways Experiment Station, 1972, Overflow spillways, site-discharge relation for uncontrolled flow: Vicksburg, Miss., Hydraulic Design Chart 113–3/3.

Wu, J.S., King, E.L., and Wang, Michael, 1985, Optimal identification of Muskingum routing coefficients: Water Resources Bulletin, v. 21, no. 3, p. 417–421.

Yu, Yun-Sheng, Jian, Xiaodong, Pogge, E.C., and Heidari, Manoutchehr, 1989, Management of the Kansas River Basin—a systems approach phase II: Lawrence, Kansas, Water Resources Research Institute Contribution No. 276, 119 p.

# APPENDICES

# APPENDIX A. GENERAL DESCRIPTION OF OPONDS COMPUTER PROGRAM

The computer program OPONDS (The optimal Operation of a system of PONDS) is written in FORTRAN 77. The main purpose of the program is to simulate the operation of a system of canals and ponds using various management requirements. Some examples of management requirements are target water levels (rule curves) of ponds, target releases from ponds, minimum required canal flow, maximum allowed canal flow, target water withdrawals, prioritization of ponds, and so forth. The program combines the concepts of pond zoning and rule curves together with the prioritization of ponds to determine operation of a system of canals and ponds using a linear programming technique. The resulting model is very flexible and can be easily adapted to any configuration of a canal-pond system. The introduction of penalty coefficients to the model allows model users to switch easily from one policy of operation to another by simply altering values of the penalty coefficients assigned to prioritize the various ponds.

The modeling approach converts the canal-pond operation into a minimum-cost network flow problem. Some management requirements become constraints in the network flow problem. After the minimum-cost flows are determined, these flows are transferred back to their corresponding pond-storage or canal-flow values.

The overall OPONDS program structure is shown in figure 16. In terms of functions, the whole program can be divided into three parts: (1) build and modify a flow network, (2) determine the flow in the network, and (3) output water budgets in nodes and arcs.

The first part of the program builds a basic flow network. The arcs in this network do not change throughout the simulation period and include pond-storage arcs and canal-flow arcs. For each time period, time-dependent contribution data, such as the net incremental inflow to nodes, precipitation, target water demand, water-surface evaporation, and ground-water seepage, are needed, and arcs representing these contribution data are generated and added into the basic network. If time-dependent management requirements such as seasonal flow boundary and pond rule curves are needed, the basic network can be expanded to represent these time-dependent data.

After the flow network is built, the flows in the network can be determined using a linear network flow algorithm called the out-of-kilter algorithm (Fulkerson, 1961). If no flows can be determined, the program execution terminates.

Once flows are determined for a network, water budgets for canals and ponds are computed. These water budgets are output for each time step. Time-series output of water budgets for selected canals and ponds are also provided.

The program source codes are listed in Appendix E of this report. The electronic form of the source codes may be obtained by contacting the U.S. Geological Survey in Lawrence, Kansas.

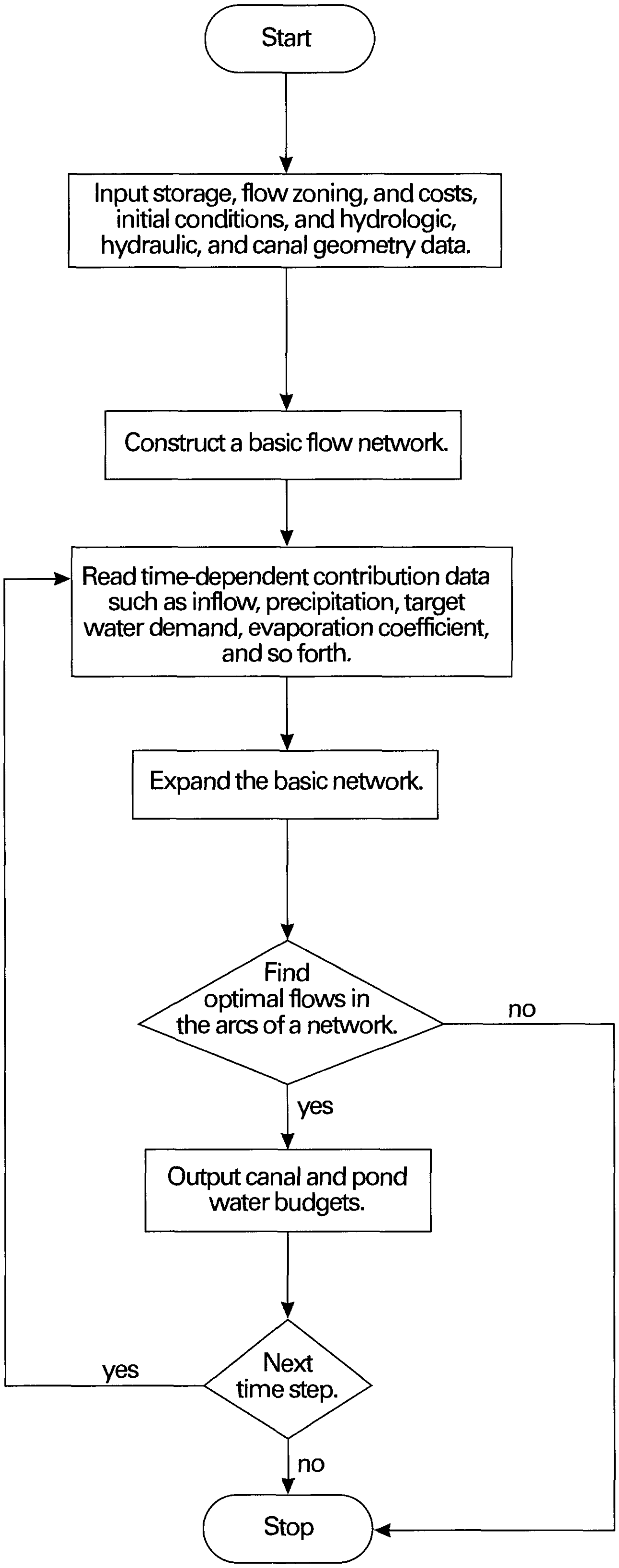



**Figure 16.** Overall structure of OPONDS computer program.

# APPENDIX B. INPUT/OUTPUT INSTRUCTIONS FOR OPONDS COMPUTER PROGRAM

Because there are more than 20 input and output data files, all input- and output-data file names and associated file-identification codes are listed in the master data file. File-identification codes here are used to distinguish data files. Table 16 lists all available file-identification codes and descriptions of associated data files. The master data-file format is listed in table 17. Instruction file formats for different input data are summarized in tables 18 through 36. Most input data files consist of four parts of information—title area (five title lines), data unit code, nodal name list, and data matrix. Data are input with free format; that is, the data are delimited by spaces.

The number of input data files is dependent on the study need. The essential files to run the program are the master data file, the general network configuration and parameter file, and the network flow-configuration file if data in this file are not included in the general network configuration and parameter file. The other data files are added only if needed. For example, if a study involves the operation of pond(s), then files for relations of elevation-volume-area of ponds and pond-storage zoning are needed.

Most input data are related to a nodal name. Nodal names are limited to 12 characters and are not case sensitive. For example, POND_1 and pump_1 are valid nodal names. Commas and spaces are not allowed in a nodal name.

**Table 16.** List of file codes and descriptions for OPONDS computer program

[--, not applicable]

| File code | File description | File format |
|---|---|---|
| | **Input data files** | |
| 0 | General network configuration and parameter file | See table 18 |
| 1 | Pond zoning file | See table 19 |
| 2 | Network flow-configuration file | See table 20 |
| 3 | Canal geometry file | See table 21 |
| 4 | Outlet hydraulic-structure file | See table 22 |
| 5 | Surface-runoff parameter file | See table 23 |
| 9 | Pond elevation-volume-area file | See tables 24 and 25 |
| 10 | Seasonal target water-demand file | See table 26 |
| 11 | Seasonal water-surface evaporation file | See table 27 |
| 12 | Seasonal flow-boundary file | See table 28 |
| 13 | Seasonal rule-curve file | See table 29 |
| 16 | Local net incremental inflow file | See table 30 |
| 17 | Precipitation file | See table 31 |
| 18 | Time-dependent, evaporation file | See table 32 |
| 19 | Time-dependent, target water-demand file | See table 33 |
| 20 | Time-dependent, rule-curve elevation file | See table 34 |
| 21 | Time-dependent, flow-boundary file | See table 35 |
| 22 | Ground-water-elevation file | See table 36 |
| 23 | Fixed-flow file | See table 37 |
| | **Output files** | |
| 26 | Network configuration output | -- |
| 27 | Nodal budget output | -- |
| 28 | Arc budget output | -- |
| 29 | Operation of hydraulic-structure output | -- |
| 30 | File for listing nodal names for nodal water-budget output in time-series format | -- |
| 31 | File for listing canal upstream and downstream nodal names for canal water-budget results in time-series format | -- |

**Table 17.** File format for master file in OPONDS computer program

[--, not applicable]

| Statement number | Information at statement | Variable | Definition | Variable type | Statement number | Information at statement | Variable | Definition | Variable type |
|---|---|---|---|---|---|---|---|---|---|
| 1–5 | Title lines | -- | Title and variable descriptions. | -- | 6 | File code and file names | CD<br>FILNM | File code (see table 16). File name. | integer<br>character |

The time-series output of water budgets for a node will use the nodal name as a part of the output file name. If the program is running on a personal computer (PC) and time-series output for a nodal water budget is needed, the corresponding nodal name is limited to five characters because a file name, not including the file extension, on PC MS–DOS systems is limited to eight characters.

Seasonal data here mean that values change seasonally (monthly, weekly, even daily) within a year and do not change over years. Some examples of seasonal data are target water demands, pond rule curves (target water level), and evaporation coefficients. These data can be input either as seasonal data or nonseasonal data depending on the length of the simulation period. If the whole simulation period is multiple years, the seasonal data can be specified in the seasonal data file. However, if the simulation period is less than 1 year, seasonal data can be input as nonseasonal data because this may result in smaller input files.

## General Network-Configuration and Parameter File

The general network-configuration and parameter file is used for specifying the basic simulation information, such as length of the simulation period, the number of seasons of a year, and accuracy of output results (see items 1 through 7 in table 18). In addition to the simulation information, data for constructing a basic network, such as pond zoning, canal zoning, canal-flow directions, canal hydrologic and geometry data, and seasonal input data, also can be included in this file. The part numbers are designed to input these data (see instructions in table 18). Note that these data may be specified in separate files (see tables 19 through 23 and 26 through 29).

**Table 18.** File format for general network configuration and parameters (file-identification code 0 used in OPONDS computer program)

[<, less than; SCS, Soil Conservation Service; $ft^3/s$, cubic feet per second; ft, feet; $ft^3/d$, cubic feet per day; ft/d, feet per day; acre-ft, acre-feet; in/d, inch per day; in., inch; mm/d, millimeter per day; mm, millimeter; --, not applicable; >, greater than; <, less than]

| Statement number | Information at statement | Variable name | Definition | Variable type | Default value | Unit |
|---|---|---|---|---|---|---|
| 1–2 | Title lines | SYSNAM | Canal-pond system name. | character | | |
| 3 | Time step and seasons | PERD<br>NPER | Number of days in a time period.<br>Number of periods in a year. | real<br>integer | <br>-- | day<br>-- |
| 4 | Starting season | STMO<br>YR | Starting season.<br>Starting year. | integer<br>integer | <br>-- | --<br>-- |
| 5 | Length of simulation periods | NSPS | Number of simulation periods. | | | -- |
| 6 | Convergence | RTERM<br>LDIRT | Flow convergence criterion.<br>Maximum number of iterations. | real<br>integer | <br>-- | $ft^3/s$<br>-- |
| 7 | Output budget accuracy | XP | Number of decimal points in acre-ft. | integer | | |

**Table 18.** File format for general network configuration and parameters (file-identification code 0 used in OPONDS computer program)—Continued

| Statement number | Information at statement | Variable name | Definition | Variable type | Default value | Unit |
|---|---|---|---|---|---|---|
| 8 | Save options | SAVOPT | Save options for general output (0—all; 1—input data; 2—network configuration; 9—none). | integer | 0 | |
| 9 | **Part 1:** | PART | Part. | character | PART | |
| | **Pond zones and bottom information** | N | Part number. | integer | 1 | -- |
| 10 | List of variables | -- | -- | string | | |
| 11 | Data | NAME | Pond node name. | character | | |
| | | UNITCD | Elevation or storage unit code (0—ft; 1—acre-ft). | integer | -- | -- |
| | | INST | Initial pond elevation or storage. | real | -- | -- |
| | | BOT | Bottom elevation. | real | -- | ft |
| | | KY | Bottom-bed hydraulic conductivity. | real | -- | ft/d |
| | | B | Bottom-bed thickness. | real | -- | ft |
| | | RC | Rule-curve elevation or storage. | real | -- | -- |
| | | Z(I) | Zone elevation or storage. | real | -- | -- |
| | | COST(I) I = 1, NZONE) | Penalty coefficients. NZONE is the number of operational storage zones of a pond. | integer | -- | -- |
| 12 | Finish | FINISH | Finish. | character | finish | -- |
| 13 | **Part 2:** | PART | Part. | character | PART | -- |
| | **Flow network** | N | Part number. | integer | 2 | -- |
| 14 | List of variables | -- | -- | string | -- | -- |
| 15 | Data | F_NODE | From-node name. | character | -- | -- |
| | | T_NODE | To-node name. | char | -- | -- |
| | | LBND | Lower flow boundary. | real | -- | $ft^3/s$ |
| | | UBND | Upper flow boundary. | real | -- | $ft^3/s$ |
| | | COST | Penalty coefficient for flow zone. | real | -- | -- |
| | | INST | Initial canal storage. | real | -- | acre-ft |
| | | K | Traveltime through routing canal. | real | -- | day |
| | | X | Weighting factor between 0 and 0.5. | real | -- | -- |
| | | SP | Canal-seepage coefficient (<1.0); if < 0, use Darcy's law. | real | -- | -- |
| | | EV | Evaporation coefficient. | real | -- | in/d |
| 16 | Finish | FINISH | Finish. | character | finish | -- |
| 17 | **Part 3: Canal** | PART | Part. | character | PART | -- |
| | **geometry data** | N | Part number. | integer | 3 | -- |
| 18 | List of variables | -- | -- | character | -- | -- |

**Table 18.** File format for general network configuration and parameters (file-identification code 0 used in OPONDS computer program)—Continued

| Statement number | Information at statement | Variable name | Definition | Variable type | Default value | Unit |
|---|---|---|---|---|---|---|
| 19 | Data | F_NODE | From-node name. | character | | -- |
| | | T_NODE | To-node name. | character | | -- |
| | | N | Canal roughness coefficient. | real | | -- |
| | | L | Canal length. | real | | ft |
| | | J | Canal average slope. | real | | -- |
| | | W | Canal width. | real | | ft |
| | | M | Canal side slope. | real | 0 | -- |
| | | D | Canal maximum depth. | real | | ft |
| | | KY | Canal riverbed hydraulic conductivity. | real | | ft/d |
| | | | | | | -- |
| | | THICK | Riverbed thickness. | real | | ft |
| | | ELEV | Riverbed elevation at entry of canal. | real | | ft |
| 20 | Finish | FINISH | Finish. | character | finish | -- |
| 21 | **Part 4: Hydraulic** | PART | Part. | character | PART | -- |
| | **structure** | N | Part number. | integer | 4 | -- |
| 22 | List of variables | -- | -- | string | -- | -- |
| 23 | Information for | NAME | Structure name. | character | -- | -- |
| | outlet hydraulic | F_NODE | From-node name. | character | -- | -- |
| | structure—pipe | T_NODE | To-node name. | character | -- | -- |
| | | TYPE | Structure type code (1—sharp-crested weir; 2—gate on spillway; 3—gate on broad-crested weir). | integer | -- | -- |
| | | BELEV | Base elevation. | real | -- | ft |
| | | WLEN | Weir length. | real | -- | ft |
| | | WHITE | Weir height if sharp-crested and broad-crested weir, or the design water head if standard spillway. | real | -- | ft |
| | | GHITE | Gate opening height. | real | -- | ft |
| | | WB | Weir width (thickness). | real | -- | ft |
| 23 | Information for | NAME | Structure name. | character | -- | -- |
| | outlet hydraulic | F_NODE | From-node name. | character | -- | -- |
| | structure—pipe | T_NODE | To-node name. | character | -- | -- |
| | | TYPE | Structure type code (= 6). | integer | -- | -- |
| | | BELEV | Base elevation. | real | -- | ft |
| | | WIDTH | Pipe diameter. | real | -- | ft |
| | | LENG | Pipe length. | real | -- | ft |
| | | FRIC | Pipe-friction factor. | real | 0.025 | -- |
| | | ENLOS | Pipe-entrance loss factor. | real | 0.5 | -- |
| 24 | Finish | -- | -- | character | -- | -- |
| 25 | **Part 5: Surface-** | PART | Part. | character | PART | -- |
| | **runoff** | N | Part number. | integer | 5 | -- |
| | **parameters** | | | | | |
| 26 | List variable | -- | -- | -- | -- | -- |

**Table 18.** File format for general network configuration and parameters (file-identification code 0 used in OPONDS computer program)—Continued

| Statement number | Information at statement | Variable name | Definition | Variable type | Default value | Unit |
|---|---|---|---|---|---|---|
| 27 | Data | NAME | Nodal name. | character | -- | -- |
| | | A5DR0 | Initial total antecedent 5-day rainfall. | real | -- | in. |
| | | A5DRI | Antecedent 5-day rainfall for dry conditions. | real | -- | in. -- |
| | | A5DRIII | Antecedent 5-day rainfall for wet conditions. | real | -- | in. |
| | | SCSCN | SCS curve number for average condition. | real | -- | -- |
| | | AREA | Drainage area. | real | -- | acres |
| 28 | Finish | -- | -- | -- | -- | -- |
| 29 | **Part 10: Seasonal target water demands** | PART | Part. | character | PART | -- |
| | | N | Part number. | integer | 10 | -- |
| 30 | Unit | WSUNIT | Target water-demand unit code (0—acre-ft; 1—$ft^3/s$; 2—$ft^3/d$). | integer | 0 | -- |
| 31 | List of variables | TIME | Time step. | character | -- | -- |
| | | (NAME (I), I = 1, NWSND) | Nodal names. NWSND is the number of nodes with seasonal target water demands. | character | -- | -- |
| 32 | Target water demands for each season N, N = 1, NPER | TIME | Season. | character | -- | -- |
| | | (WSTB (N,J), J = 1, NWSND) | Seasonal target water demands. | real | -- | -- |
| 33 | Finish | FINISH | Finish. | character | finish | -- |
| 34 | **Part 11: Seasonal water-surface evaporation coefficient** | PART | Part. | character | PART | -- |
| | | N | Part number. | integer | 11 | -- |
| 35 | Unit | EVUNIT | Surface-water evaporation coefficient unit code (0—mm/d; 1—in/d; 2—ft/d). | integer | -- | -- |
| 36 | List of variables | TIME | Time step. | character | -- | -- |
| | | (NAME (I), I = 1, NEV) | Nodal names. NEV is the number of nodes with surface-water evaporation coefficients. | character | -- | -- |
| 37 | Water-surface evaporation coefficients for each season N, N = 1, NPER | TIME | Season. | char | -- | -- |
| | | (EVTB (N,J), J=1, NEV) | Evaporation coefficients. | real | -- | -- |
| 38 | Finish | FINISH | Finish. | character | finish | -- |
| 39 | **Part 12: Seasonal flow boundaries** | PART | Part. | character | PART | -- |
| | | N | Part number. | integer | 12 | -- |
| 40 | Unit | FBUNIT | Flow unit code (0—acre-ft; 1—$ft^3/s$; 2— $ft^3/d$). | integer | -- | -- |

**Table 18.** File format for general network configuration and parameters (file-identification code 0 used in OPONDS computer program)—Continued

| Statement number | Information at statement | Variable name | Definition | Variable type | Default value | Unit |
|---|---|---|---|---|---|---|
| 41 | List of nodal names | (NAME (J), J = 1, NFBAR) | From-nodal names. NFBAR is the number of arcs. | character | -- | -- |
| 42 | List of nodal names | (NAME (J), J = 1, NFBAR) | To-nodal names. | character | -- | -- |
| 43 | Zone index | TIME, | Season. | character | -- | -- |
| | | (FBIDX (J), J = 1, NFBAR) | Flow-zone index (-1, lower boundary of normal flow zone; +1, upper boundary of normal flow zone, <-2, lower extended flow zone; >+2, upper extended flow zone; that is, $ndex = zone \pm$). | integer | -- | -- |
| 44 | Flow boundaries for each season N, N = 1, NPER | TIME, | Season. | character | -- | -- |
| | | (FBTB (N,J), J = 1, NFBAR) | Seasonal flow boundaries. | real | -- | -- |
| 45 | Finish | FINISH | Finish. | character | finish | -- |
| 46 | **Part 13: Seasonal rule curve** | PART | Part. | character | PART | -- |
| | | N | Part number. | integer | 13 | -- |
| 47 | Unit | RCUNIT | Rule-curve elevation unit code (0—ft; 1—in.; 2—mm). | integer | -- | -- |
| 48 | List of variables | TIME | Time step. | character | -- | -- |
| | | (NAME (I), I = 1, NRCND) | Pond nodal names. NRCND is the number of pond nodes with seasonal rule curves. | character | -- | -- |
| 49 | Rule curves for each season N, N = 1, NPER | N | Season. | character | -- | -- |
| | | (RCTB (N,J), J=1, NRCND) | Rule-curve elevation. | real | -- | -- |
| 50 | Finish | FINISH | Finish. | character | finish | -- |

## Pond-Zoning File

The pond-zoning file is used to specify pond-storage zones and bottom hydraulic parameters. Initial storage, rule curve, pond zoning, and penalty coefficients are specified in this file. The bottom hydraulic and geometry data include hydraulic conductivity, average thickness of bottom layer, and bottom elevation. Input instructions and explanations of variables are listed in table 19. The file-identification code is 1. All data specified in this file can also be included in the general network-configuration and parameter file (table 18).

**Table 19.** File format for initial pond condition and zoning (file-identification code 1 used in OPONDS computer program)

[ft, feet; acre-ft, acre-feet; ft/d, feet per day; --, not applicable]

| Statement number | Information at statement | Variable name | Definition | Variable type | Unit |
|---|---|---|---|---|---|
| 1–5 | Title lines | | Specify data information about source, etc. | | |
| 6 | Pond zoning and bottom hydraulic parameter for each pond | NAME | Pond node name. | character | -- |
| | | UNITCD | Stage unit code (0—ft, 1—acre-ft). | integer | -- |
| | | INST | Pond initial elevation or storage. | real | -- |
| | | BOT | Bottom elevation. | real | ft |
| | | KY | Bottom layer hydraulic conductivity. | real | ft/d |
| | | B | Bottom layer thickness. | real | ft |
| | | RC | Rule-curve stage. | real | -- |
| | | (Z(I) | Zone stage (see note 1). | real | -- |
| | | COST(I) I = 1, NZONE) | Penalty coefficients (see note 1). NZONE is the number of storage zones of a pond. | integer | -- |

Note:

1. If there is more than one zone, Z and COST must be specified for each zone. Zone boundaries (that is, elevations) or corresponding storage are specified at the beginning of the zones next to the rule curve.

## Network Flow-Configuration File

The network flow-configuration file contains data for determining the directions of canal-flow arcs and canal-routing coefficients. Each record in the file represents an arc in a network. Because each flow zone in the canal is represented by an arc, there is one record for each extended flow zone. Canal-routing coefficients (initial canal storage, traveltime, weighting factor, and seepage coefficient) are optional. If canal routing is not needed, routing coefficients do not need to be specified or the values are set equal to zero. Input instructions and explanations of variables are listed in table 20. The file-identification code is 2. All data specified in this file also can be included in the general network-configuration and parameter file (table 18). Network-flow configuration data need to be specified in order to run the OPONDS program.

**Table 20.** File format for network flow configuration (file-identification code 2 used in OPONDS computer program)

[ft$^3$/s, cubic feet per second; acre-ft, acre-feet; in/d, inches per day; --, not applicable; <, less than]

| Statement number | Information at statement | Variable | Definition | Variable type | Unit |
|---|---|---|---|---|---|
| 1–5 | Title lines | -- | -- | -- | -- |
| 6 | Canal-flow zoning and routing coefficients for each flow zone | F_Node | From-node name. | character | -- |
| | | T_Node | To-node name (see note 1). | character | -- |
| | | LBND | Lower flow boundary (see note 2). | real | ft$^3$/s |
| | | UBND | Upper flow boundary (see note 2). | real | ft$^3$/s |
| | | COST | Penalty coefficient of flow zone. | integer | -- |
| | | INST | Initial canal storage. | real | acre-ft |
| | | K | Traveltime through routing canal . | real | day |
| | | x | Weighting factor between 0 to 0.5. | real | -- |
| | | SP | Canal-seepage coefficient (<1.0) (see note 3). | real | -- |
| | | EV | Evaporation coefficient (see note 4). | real | in/d |

Notes:

1. If there is not a physical downstream node (that is, the downstream node is SINK), use node name SKSC.
2. If there are more than one flow zone, the normal flow zone must be specified first.

One record is specified for each flow zone, including normal, lower, and upper zones.

3. If SP < 0, the seepage will be estimated using Darcy's law. The average depth of water is estimated using Manning's equation. The hydraulic and geometry parameters used in Manning's equation must be specified.
4. Only the evaporation (EV) occurring in the normal flow range will be estimated.

If EV > 0, this value will be used for entire simulation period.

If EV < 0, the evaporation coefficient will be interpreted in terms of coefficients in upstream and downstream nodes if available. To estimate the surface-water evaporation for canals, canal geometry parameters (length, width, and side slopes) must be specified in the geometry and riverbed hydraulic parameter file. If the canal geometry data are not specified, the canal water-surface evaporation will not be calculated.

## Canal-Geometry and Riverbed Hydraulic Parameter File

The canal-geometry and riverbed hydraulic parameter file is used for specifying canal cross-section data, riverbed hydraulic parameters, canal lengths, and canal-entry bottom elevations. These data are used mainly in canal routing for determining canal seepage and surface-water evaporation. These data are optional. Input instructions and explanations of variables are listed in table 21. The file-identification code is 3. All data specified in this file also can be included in the general network-configuration and parameter file (table 18).

**Table 21.** File format for canal geometry and riverbed hydraulic parameters (file-identification code 3 used in OPONDS computer program)

[ft, feet; ft/d, feet per day; --, not applicable]

| Statement number | Information at statement | Variable | Definition | Variable type | Unit |
|---|---|---|---|---|---|
| 1–5 | Title lines | -- | -- | -- | -- |
| 6 | Canal geometry data | F_NODE | From-node name. | character | -- |
| | | T_NODE | To-node name. | character | -- |
| | | N | Canal roughness coefficient. | real | -- |
| | | L | Canal length. | real | ft |
| | | J | Canal average slope. | real | -- |
| | | W | Canal bottom width. | real | ft |
| | | M | Canal side slope. | real | -- |
| | | D | Canal maximum depth. | real | ft |
| | | KY | Canal riverbed hydraulic conductivity. | real | ft/d |
| | | THICK | Riverbed thickness. | real | ft |
| | | ELEV | Riverbed elevation at entry. | real | ft |

## Outlet Hydraulic-Structure Parameter File

Outlet hydraulic structures included in the OPONDS program are rectangular sharp-crested weirs, vertical sluice gates on broad-crested weirs, vertical flat gates on spillways, and pipes. Flow through gated weirs and sharp-crested weirs is assumed to be controllable by adjusting gate opening heights or sharp-crested weir heights. The data needed and input instructions and explanations of variables are listed in table 22. The file-identification code is 4. All data specified in this file also can be included in the general network-configuration and parameter file (table 18).

**Table 22.** File format for outlet hydraulic-structure parameters (file-identification code 4 used in OPONDS computer program)

[ft, feet; --, not applicable]

| Statement number | Information at statement | Variable | Definition | Variable type | Default value | Unit |
|---|---|---|---|---|---|---|
| 1–5 | Title lines | -- | -- | -- | -- | -- |
| 6 | Information for each weir | NAME | Structure name, | character | -- | -- |
| | | F_NODE | Nodal name (see note 4). | character | -- | -- |
| | | T_NODE | Downstream nodal name (see note 4). | character | -- | -- |
| | | TYPE | Structure type code (see note 1). | integer | -- | -- |
| | | BELEV | Base elevation (see note 2). | real | -- | ft |
| | | WLEN | Weir length. | real | -- | ft |
| | | WHITE | Weir height if sharp-crested and broad-crested weir, or the design water head if standard spillway. | real | -- | ft |
| | | GHITE | Gate opening height. | real | -- | ft |
| | | WB | Weir width (thickness). | real | -- | ft |
| | Information for each outlet pipe | NAME | Structure name. | character | -- | -- |
| | | F_NODE | Nodal name (see note 4). | character | -- | -- |
| | | T_NODE | Downstream nodal name (see note 4). | character | -- | -- |
| | | TYPE | Structure type code (see note 1). | integer | -- | -- |
| | | BELEV | Base elevation (see note 2). | real | -- | ft |
| | | WIDTH | Pipe diameter. | real | -- | ft |
| | | LENG | Pipe length (see note 3). | real | -- | ft |
| | | FRIC | Pipe-friction factor. | real | 0.025 | -- |
| | | ENLOS | Pipe-entrance loss factor. | real | 0.5 | -- |

Notes:

1. Hydraulic-structure type code:
   1—Rectangular sharp-crested weir ($0 < H/P < 5$).
   2—Vertical flat gate on spillway. Gate edge is facing downstream.
   3—Vertical sluice gate on broad-crested weir.
   6—Pipe.
2. Base elevation: (1) bottom of a weir if sharp-crested weir, (2) top of a weir if spillway or broad-crested weir, and (3) center of a pipe at entry if pipe.
3. If sharp-crested weir height is less than zero (< 0), the weir height is adjustable, and the absolute value is the maximum height allowed.
4. There is only one flow zone downstream from the structure. No extended flow zones are allowed.

## Surface-Runoff Parameter File

Surface runoff is calculated using the SCS curve-number method (Soil Conservation Service, 1985). Data needed are drainage area, curve number for average condition, initial antecedent 5-day rainfall, and criteria for wet/dry conditions. Input instructions and explanations of variables are listed in table 23. The file-identification code is 5. All data specified in this file also can be included in the general network-configuration and parameter file (table 18).

**Table 23.** File format for surface-runoff parameters (file-identification code 5 used in OPONDS computer program)

[in., inches; --, not applicable]

| Statement number | Information at statement | Variable | Definition | Variable type | Default value | Unit |
|---|---|---|---|---|---|---|
| 1–5 | Title lines | -- | -- | -- | -- | -- |
| 6 | Data | NAME | Nodal name. | character | -- | -- |
| | | A5DR0 | Initial antecedent 5-day rainfall. | real | -- | in. |
| | | A5DRI | Antecedent 5-day rainfall for dry conditions (I) (see note 1). | real | 0.5 | in. |
| | | A5DRIII | Antecedent 5-day rainfall for wet conditions (III) (see note 2). | real | 1.1 | in. |
| | | SCSCN | SCS curve number for average conditions (II). | real | -- | -- |
| | | AREA | Drainage area. | real | -- | acres |

Notes:

1. The suggested values are less than 0.5 in. for dormant season and less than 1.4 in. for growing season (Soil Conservation Service, 1985; McCuen, 1989).

2. The suggested values are greater than 1.1 in. for dormant season and greater than 2.1 in. for growing season (Soil Conservation Service, 1985; McCuen, 1989).

## Pond Elevation-Volume-Area Relation File

If there is any pond operation involved, the pond elevation-volume-area relation file is used. The relations among water-surface elevation, volume, and water-surface area of ponds can be expressed either in tabular form or in the regression equations for the Quivira National Wildlife Refuge (tables 24 and 25). The file-identification code is 9.

**Table 24.** File format for relations among water-surface elevation (*Z*), volume (*V*), and water-surface area (*A*) of ponds (file-identification code 9 used in OPONDS computer program)

[ft; feet; acre-ft, acre-feet; --, not applicable]

| Statement number | Information at statement | Variable | Definition | Variable type | Unit |
|---|---|---|---|---|---|
| 1–5 | Title lines | -- | -- | -- | -- |
| 6 | Data source | ZVAMTH | Data source index for *Z-V-A* data (= 0). | integer | -- |
| 7 | Pond name | NAME | Pond nodal name. | character | -- |

**Table 24.** File format for relations among water-surface elevation (*Z*), volume (*V*), and water-surface area (*A*) of ponds (file-identification code 9 used in OPONDS computer program)—Continued

| Statement number | Information at statement | Variable | Definition | Variable type | Unit |
|---|---|---|---|---|---|
| 8 | Pond characteristic curves among elevation, capacity, and area | ELE | Water-surface elevation. | real | ft |
| | | CAP | Water volume of a pond at the current elevation. | real | acre-ft |
| | | AREA | Water-surface area. | real | acre |
| 9 | Empty line | -- | Move to next pond node. | | -- |
| 10 | Finish | FINISH | Finish reading pond table. | character | -- |

**Table 25.** File format for regression relations of water-surface elevation (*Z*), volume (*V*), and water-surface area (*A*) of ponds (file-identification code 9 used in OPONDS computer program)

[ft, feet; --, not applicable]

| Statement number | Information at statement | Variable | Definition | Variable type | Unit |
|---|---|---|---|---|---|
| 1–5 | Title lines | -- | -- | -- | -- |
| 6 | Data source | ZVAMTH | Data source index for *Z*-*V*-*A* data (= 1). | integer | -- |
| 7 | Pond name | NAME | Pond nodal name. | character | -- |
| 8 | Base and coefficients for each regression equation | N | Equation sequential number. | integer | -- |
| | | ZB | Zonal elevation base. | real | ft |
| | | A1 | Coefficient A1. | real | -- |
| | | A2 | Coefficient A2. | real | -- |
| | | A3 | Coefficient A3. | real | -- |
| 9 | Empty line | -- | Move to next pond node. | -- | -- |
| 10 | Finish | FINISH | Finish reading coefficients of regression equations. | character | -- |

## Seasonal Target Water-Demand File

Target water demand in the OPONDS program means that water will be withdrawn from a node; that is, water will be taken out of the canal and control-pond system. Input instructions and explanations of variables are listed in table 26. The file-identification code is 10. All data specified in this file also can be included in the general network-configuration and parameter file (table 18).

**Table 26.** File format for seasonal target water demands (file-identification code 10 used in OPONDS computer program)

[acre-ft, acre-feet; $ft^3/s$, cubic feet per second; $ft^3/d$, cubic feet per day; --, not applicable]

| Statement number | Information at statement | Variable | Definition | Variable type |
|---|---|---|---|---|
| 1–5 | Title lines | -- | -- | -- |
| 6 | Water-demand unit | WSUNIT | Water-demand unit code (0—acre-ft, 1—$ft^3/s$, 2—$ft^3/d$). | integer |
| 7 | List of nodal names | TIME<br>NAME(I), I = 1, NWSND | Time step.<br>Nodal names (see note 1). NWSND is the number of nodes with water demands. | character<br>character |
| 8 | Target water demands for each season N, N = 1, NPER | TIME<br>(WSTB (N,J), J = 1, NWSND) | Season.<br>Seasonal target water demands. | character<br>real |

Note:
1. Use the nodal name DEFAULT for nodes with the same target water demands. The DEFAULT node must follow the other specified nodes (that is, in last column).

## Seasonal Water-Surface Evaporation File

Seasonal water-surface evaporation from a pond node or canal segment is calculated using evaporation coefficients and water-surface area. Input instructions and explanations of variables for evaporation coefficients are listed in table 27. The file-identification code is 11. All data specified in this file also can be included in the general network-configuration and parameter file (table 18).

**Table 27.** File format for seasonal water-surface evaporation coefficients (file-identification code 11 used in OPONDS computer program)

[mm/d; millimeters per day; in/d, inches per day; ft/d, feet per day; --, not applicable]

| Statement number | Information at statement | Variable | Definition | Variable type |
|---|---|---|---|---|
| 1–5 | Title lines | -- | -- | -- |
| 6 | Unit | EVUNIT | Water-surface evaporation coefficient (0—mm/d; 1—in/d; 2—ft/d). | integer |
| 7 | List of nodal names | TIME<br>(NAME(J), J = 1, NEV) | Time.<br>Nodal names (see note 1). NEV is the number of nodes with evaporation coefficients. | character<br>character |

**Table 27.** File format for seasonal water-surface evaporation coefficients (file-identification code 11 used in OPONDS computer program)—Continued

| Statement number | Information at statement | Variable | Definition | Variable type |
|---|---|---|---|---|
| 8 | Evaporation coefficients for each season N, N = 1, NPER | TIME<br>(EVTB (N,J), J=1, NEV) | Season index.<br>Evaporation coefficients. | character<br>real |

Note:

1. Use the nodal name DEFAULT for nodes with the same coefficients. The DEFAULT node must follow the other specified nodes (that is, in last column).

## Seasonal Flow-Boundary File

The flow requirements in canals may be different for different seasons. In the linear network setting, these flow requirements are represented by flow boundaries in the associated arcs. Input instructions and explanations of variables for seasonal flow boundaries are listed in table 28. The file-identification code is 12. All data specified in this file can also be included in the general network-configuration and parameter file (table 18).

**Table 28.** File format for seasonal flow boundaries (file-identification code 12 used in OPONDS computer program)

[acre-ft, acre-feet; $ft^3/s$, cubic feet per second; $ft^3/d$, cubic feet per day; --, not applicable; >, greater than; <, less than]

| Statement number | Information at statement | Variable | Definition | Variable type |
|---|---|---|---|---|
| 1–5 | Title lines | | -- | -- |
| 6 | Unit | FBUNIT | Flow unit code (0—acre-ft; 1—$ft^3/s$; 2—$ft^3/d$). | integer |
| 7 | List of upstream nodal names | (NAME(J),J = 1, NFBAR) | From-node names of arcs. NFBAR is the number of arcs with seasonal flow boundaries. | character |
| 8 | List of downstream nodal names | (NAME(J),J = 1, NFBAR) | To-node names of arcs. | character |
| 9 | Zone index | TIME,<br>(FBIDX(J), J = 1, NFBAR) | Season.<br>Flow-zone index (-1, lower boundary of normal flow zone; +1, upper boundary of normal flow zone; <-2, lower extended flow zone; >+2, upper extended flow zone; that is, $ndex = zone \pm$). | character<br>integer |
| 10 | Flow boundaries for each season N, N = 1, NPER | TIME,<br>(FBTB (N,J), J = 1, NFBAR), N = 1, NPER | Season.<br>Seasonal flow boundary (see note 1). | character<br>real |

Note:

1. For lower or upper extended flow zones, flow boundaries are equal to flow-zone capacities.

## Seasonal Pond Rule-Curve File

Management water levels in a pond may change seasonally. The seasonal water levels are represented by the different rule curves in the linear-network flow model. Input instructions and explanations of variables for seasonal rule curves are listed in table 29. The file-identification code is 13. All data specified in this file also can be included in the general network-configuration and parameter file (table 18).

**Table 29.** File format for seasonal pond rule curves (file-identification code 13 used in OPONDS computer program)

[ft, feet; in., inches; mm, millimeters; --, not applicable]

| Statement number | Information at statement | Variable | Definition | Variable type |
|---|---|---|---|---|
| 1–5 | Title lines | -- | -- | -- |
| 6 | Unit | RCUNIT | Rule-curve elevation unit code (0—ft; 1—in.; 2—mm). | integer |
| 7 | List of nodal names | TIME | Time. | character |
| | | (NAME(J), J = 1, NRCND) | Pond nodal names (see note 1). NRCND is the number of pond nodes with seasonal rule curves. | character |
| 8 | Rule curves for each season N, N = 1, NPER | TIME | Season. | character |
| | | (RCTB (N,J), J=1, NRCND) | Rule-curve elevations. | real |

Note:

1. Use the nodal name DEFAULT for pond nodes with the same rule curves. The DEFAULT node must follow the other specified nodes (that is, in last column).

## Local Net Incremental Inflow File

The local net incremental inflow file is used to specify nodes and their local net inflows. Local net incremental inflow is the water locally attributed to a node. Input instructions and explanations of variables for local net inflows are listed in table 30. The file-identification code is 16.

**Table 30.** File format for local net incremental inflows (file-identification code 16 used in OPONDS computer program)

[acre-ft, acre-feet; $ft^3/s$, cubic feet per second; $ft^3/d$, cubic feet per day; --, not applicable]

| Statement number | Information at statement | Variable | Definition | Variable type |
|---|---|---|---|---|
| 1–5 | Title lines | -- | -- | -- |
| 6 | Flow unit | IFWCD | Flow unit code (0—acre-ft; 1—$ft^3/s$; 2—$ft^3/d$). | integer |
| 7 | List of nodal names | TIME | Time. | character |
| | | (NAME(J), J = 1, NIFW) | Nodal names (see note 1). NIFW is the number of nodes with net incremental inflows. | character |

**Table 30.** File format for local net incremental inflows (file-identification code 16 used in OPONDS computer program)—Continued

| Statement number | Information at statement | Variable | Definition | Variable type |
|---|---|---|---|---|
| 8 | Net inflows to node for each time period | TIME<br>QIN(I), I = 1, NIFW | Time.<br>Net inflow to nodes (see note 2). | character<br>real |

Notes:

1. Use the nodal name DEFAULT for nodes with the same net inflows. The DEFAULT node must follow the other specified nodes (that is, in last column).
2. Net inflow to a node can be either a positive or a negative value.

## Precipitation File

The precipitation file is used to assign precipitation data to nodes. The precipitation data are used to calculate surface runoff to nodes. Input instructions and explanations of variables for precipitation data are listed in table 31. The file-identification code is 17.

**Table 31.** File format for precipitation data (file-identification code 17 used in OPONDS computer program)

[ft, feet; in., inches; mm, millimeters; acre-ft/d, acre-feet per day; $ft^3/s$, cubic feet per second; $ft^3/d$, cubic feet per day; --, not applicable]

| Statement number | Information at statement | Variable | Definition | Variable type |
|---|---|---|---|---|
| 1–5 | Title lines | -- | -- | -- |
| 6 | Data unit | RNUNIT<br>RNTYPE | Data unit code (if rntype = 1, 0 —ft, 1—in., 2—mm; if rntype = 2, 0—acre-ft/d, 1—$ft^3/s$, 2—$ft^3/d$).<br>Date type index (1—depth; 2—flux). | integer<br>integer |
| 7 | List of nodal names | TIME<br>(NAME(J), J = 1, NRAIN) | Time.<br>Nodal names (see note 1). NRAIN is the number of nodes with precipitation. | character<br>character |
| 8 | Precipitation data for each time period | TIME<br>RAIN(I), I = 1, NRAIN | Time.<br>Precipitation depth or flux rate to nodes. | character<br>real |

Note:

1. Use the nodal name DEFAULT for nodes with the same amount of precipitation. The DEFAULT node must follow the other specified nodes (that is, in last column).

## Time-Dependent, Water-Surface Evaporation File

In general, water-surface evaporation coefficients change with time. Even for the same season and the same place, the evaporation coefficients may be significantly different for different years. This file is designed to input time-dependent, water-surface evaporation coefficients for selected nodes. Input instructions and explanations of variables are listed in table 32. The file-identification code is 18.

**Table 32.** File format for time-dependent, water-surface evaporation coefficients (file-identification code 18 used in OPONDS computer program)

[mm/d; millimeters per day; in/d, inches per day; ft/d, feet per day; --, not applicable]

| Statement number | Information at statement | Variable | Definition | Variable type |
|---|---|---|---|---|
| 1–5 | Title lines | -- | -- | -- |
| 6 | Date unit | EVUNIT | Evaporation coefficient unit code (0—mm/d, 1—in/d, 2 —ft/d). | integer |
| 7 | List of nodal names | TIME<br>(NAME(J), J = 1, NEV) | Time.<br>Nodal names (see note 1). NEV is the number of nodes with water-surface evaporation coefficients. | character<br>character |
| 8 | Water-surface evaporation for each time period | TIME<br>(EVTB (0,J), J=1, NEV) | Time.<br>Evaporation coefficients (see note 2). | character<br>real |

Notes:

1. Use the nodal name DEFAULT for nodes with the same coefficients. The DEFAULT node must follow the other specified nodes (that is, in last column).

2. If both seasonal and nonseasonal coefficients are specified for the same node, only nonseasonal coefficients are used.

## Time-Dependent, Target Water-Demand File

The time-dependent, target water-demand file is used to input target water demands for selected nodes for which water withdrawals change with time. Input instructions and explanations of variables are listed in table 33. The file-identification code is 19.

**Table 33.** File format for time-dependent, target water demand (file-identification code 19 used in OPONDS computer program)

[acre-ft, acre-feet; $ft^3/s$, cubic feet per second; $ft^3/d$, cubic feet per day; --, not applicable]

| Statement number | Information at statement | Variable | Definition | Variable type |
|---|---|---|---|---|
| 1–5 | Title lines | -- | -- | -- |
| 6 | Units of water demand | WSUNIT | Water-demand unit code (0—acre-ft; 1—$ft^3/s$; 2—$ft^3/d$). | integer |

**Table 33.** File format for time-dependent, target water demand (file-identification code 19 used in OPONDS computer program)—Continued

| Statement number | Information at statement | Variable | Definition | Variable type |
|---|---|---|---|---|
| 7 | List of nodal names | TIME | Time step. | character |
| | | NAME(I), I = 1, NWSND | Nodal names (see note 1). NWSND is the number of nodes with target water demands. | character |
| 8 | Target water demand for each time period | TIME | Time. | character |
| | | (WSTB (0,J), J = 1, NWSND) | Target water demands (see note 2). | real |

Notes:

1. Use the nodal name DEFAULT for nodes with the same target water demands. The DEFAULT node must follow the other specified nodes (that is, in last column).

2. If both seasonal and nonseasonal values are specified for the same node, only nonseasonal values are used.

## Time-Dependent, Pond Rule-Curve File

The target water-surface elevation of a pond changes not only seasonally but also with time (nonseasonal). It is assumed in the OPOND program that the top level of the upper zone and bottom level of the lower zone are kept unchanged. The rule curve of a pond changes between the top level of the upper zone and the bottom level of the lower zone with time. Input instructions and explanations of variables for rule-curve elevations are listed in table 34. The file-identification code is 20.

**Table 34.** File format for time-dependent, pond rule-curve elevations (file-identification code 20 used in OPONDS computer program)

[ft, feet; in., inches; mm, millimeters; --, not applicable]

| Statement number | Information at statement | Variable | Definition | Variable type |
|---|---|---|---|---|
| 1–5 | Title lines | -- | -- | -- |
| 6 | Unit | RCUNIT | Rule-curve elevation unit code (0—ft; 1—in.; 2—mm). | integer |
| 7 | List of nodal names | TIME | Time. | character |
| | | NAME(J), J = 1, NRCND | Pond nodal names (see note 1). NRCND is the number of nodes with time-dependent rule curves. | character |
| 8 | Rule-curve elevations for each time period | TIME | Time. | character |
| | | (RCTB (0,J), J=1, NRCND) | Rule-curve elevations (see note 2). | real |

Notes:

1. Use the nodal name DEFAULT for nodes with the same pond rule curves. The DEFAULT node must follow the other specified nodes (that is, in last column).

2. If both seasonal and nonseasonal values are specified for the same node, only nonseasonal values are used.

## Time-Dependent, Flow-Boundary File

The time-dependent, flow-boundary file is used to specify flow boundaries that change with time for selected flow zones in canals. Input instructions and explanations of variables are listed in table 35. The file-identification code is 21.

**Table 35.** File format for time-dependent flow boundaries (file-identification code 21 used in OPONDS computer program)

[acre-ft, acre-feet; $ft^3/s$, cubic feet per second; $ft^3/d$, cubic feet per day; --, not applicable; >, greater than; <, less than]

| Statement number | Information at statement | Variable | Definition | Variable type |
|---|---|---|---|---|
| 1–5 | Title lines | -- | -- | -- |
| 6 | Unit | FBUNIT | Flow unit code (0—acre-ft; 1—$ft^3/s$; 2—$ft^3/d$). | integer |
| 7 | List of upstream nodal names | NAME(J), J = 1, NFBAR | From-node names. NFBAR is the number of arcs with flow boundaries. | character |
| 8 | List of downstream nodal names | NAME(J), J = 1, NFBAR | To-node names. | character |
| 9 | Flow-zone index | TIME,<br>(FBIDX(J), J = 1, NFBAR) | Time.<br>Flow-zone index (-1, lower boundary of normal flow zone; +1, upper boundary of normal flow zone; <-2, lower extended flow zone; >+2, upper extended flow zone; that is, $index = zone \pm 1$ ). | character<br>integer |
| 10 | Flow boundaries for each time period | TIME,<br>(FBTB(0,J), J = 1, NFBAR), | Time.<br>Flow boundaries (see notes 1 and 2). | character<br>real |

Notes:

1. For lower or upper extended flow zone, set flow boundaries equal to corresponding flow-zone capacities.
2. If both seasonal and nonseasonal values are specified for the same flow zone in the same flow arc, only nonseasonal values are used.

## Ground-Water Elevation File

Ground-water data are used to estimate seepage from ponds and canals. Ground-water elevations are conceptually specified at nodes. Input instructions and explanations of variables for ground-water data are described in table 36. The file-identification code is 22.

**Table 36.** File format for ground-water elevations (file-identification code 22 used in OPONDS computer program)

[ft, feet; in., inches; mm, millimeters; acre-ft/d, acre-feet per day; $ft^3/s$, cubic feet per second; $ft^3/d$, cubic feet per day; --, not applicable]

| Statement number | Information at statement | Variable | Definition | Variable type |
|---|---|---|---|---|
| 1–5 | Title lines | -- | -- | -- |
| 6 | Data unit and type | GWUNIT | Data unit code (if gwtype = 1, 0—ft; 1—in.; 2—mm; if gwtype = 2, 0—acre-ft/d; 1—$ft^3/s$; 2—$ft^3/d$). | integer |
| | | GWTYPE | Data type code (1—level; 2—flux). | integer |
| 7 | List of nodal names | TIME | Time. | character |
| | | (NAME(J), J = 1, NGWND) | Nodal names (see note 1). NGWND is the number of nodes with ground-water data. | character |
| 8 | Ground-water level or flux for each time period | TIME, | Time. | character |
| | | GWLVL(I), I = 1, NGWND | Ground-water elevation or flux (see note 2). | real |

Notes:

1. Use the nodal name DEFAULT for nodes with the same values. The DEFAULT node must follow the other specified nodes (that is, in last column).
2. If the flux is a negative value, it means the pond gains water from an aquifer.

## Fixed-Flow File

One special case of canal flows is when flows are fixed to a certain amount for a given time period. This implies that there are no extended flow zones and that flow in the normal flow zone is constant in the network model. The fixed-flow file is used to specify fixed flows for selected canals. Input instructions and explanations of variables are described in table 37. The file-identification code is 23.

**Table 37.** File format for fixed flows (file-identification code 23 used in OPONDS computer program)

[acre-ft, acre-feet; $ft^3/s$, cubic feet per second; $ft^3/d$, cubic feet per day; --, not applicable]

| Statement number | Information at statement | Variable | Definition | Variable type |
|---|---|---|---|---|
| 1–5 | Title lines | -- | -- | -- |
| 6 | Flow unit | FIXCD | Flow unit code (0—acre-ft; 1—$ft^3/s$; 2—$ft^3/d$). | integer |
| 7 | List of upstream nodes | (NAME(J), J = 1, NFIX) | Upstream nodal names. NFIX is the number of arcs with fixed flows. | character |
| 8 | List of downstream nodes | TIME | Time. | character |
| | | NAME(J), J = 1, NFIX | Downstream nodal names. | character |

**Table 37.** File format for fixed flows (file-identification code 23 used in OPONDS computer program)—Continued

| Statement number | Information at statement | Variable | Definition | Variable type |
|---|---|---|---|---|
| 9 | Fixed flow for each time period | TIME | Time. | character |
| | | Q(I), I = 1, NFIX | Fixed flow. | real |

## Network-Configuration Output File

The network-configuration output file is used to summarize input data and the basic network configuration and to store error messages during program execution. The file-identification code is 26. The output file name with the file-identification code 26 must be specified in the master data file. See a sample output in Appendix D for the sample problem.

## Nodal-Budget Output File

The nodal-budget output file is used to store nodal budgets for each time period. The output for general nodes includes upstream inflows, local net inflows, surface runoff, water withdrawals, and total outflows from nodes. In addition to these budget terms, budget items for initial storage, surface-water evaporation, bottom seepage, and final storage are added for the pond-budget output. The file-identification code is 27. The output file name with the file-identification code 27 must be specified in the master data file. If the file name is not specified, no nodal water-budget output will be generated.

## Canal-Routing Output File

The canal-routing output file summarizes the canal-routing results, which include canal initial storage, inflow, canal seepage, water-surface evaporation, canal final storage, and outflow from each canal. The file-identification code is 28. The output file name with the file-identification code 28 must be specified in the master data file (see tables 16 and 17). If the file name is not specified, no canal water-budget output will be generated.

## Outlet Hydraulic-Structure Output File

The output file includes outlet hydraulic-structure information, flow through structure, and structure operation. The structure information includes structure name, type, location, and size. Structure operation means the gate opening height or sharp-crested weir height. The file-identification code is 29. The output file name with the file-identification code 29 must be specified in the master data file (see tables 16 and 17).

## List of Nodes for Time-Series Output of Water Budget

This file is used to list selected nodes for which time-series output of water budgets are needed. For the nodal listing, the file-identification code is 30. To specify a node, one nodal name is one record in the file. Two time-series output files for each of the specified nodes are generated. The first output file is for the nodal water budget with the output file as "nb_{nodal_name}.dat", where {nodal_name} is the specified nodal name. The second output file is for downstream releases from a specified node. The name of this output file is "nr_{nodal_name}.dat". If the program runs on PC MS-DOS, nodal names are limited to five characters because a file name in PC DOS is limited to eight characters.

## List of Canals for Time-Series Output of Canal-Routing Results

This file is used to list selected canals for which time-series output of routing results are needed. For the canal listing, the file-identification code is 31. A canal is represented by upstream and downstream nodal names. One record is specified for each selected canal in the file. The time-series output of the water budget will be saved in separated files for each selected canal. The output file name is of the form "arbud###.out", where ### is the sequential number of selected canals in the file.

# APPENDIX C. LIST OF SELECTED VARIABLES AND THEIR DEFINITIONS

Selected variables from the OPONDS program and their definitions are listed along with the variable type, which is presented in parentheses. If a variable is an array, the array dimension is also presented in parentheses.

**ARBFLG**—Logical indicator for arc budget output (LOGICAL).
**ARC**—Index variable for the current arc (INTEGER).
**ARCBUD**—List of budget terms for canal routing (unsigned) (**LDARC** X 0:6; INTEGER).
0—Downstream node of a canal reach.
1—Inflow.
2—Initial storage.
3—Canal seepage.
4—Canal water-surface evaporation.
5—Final storage.
6—Outflow.
**ARCS**—Number of arcs in a basic network (that is, number of physical arcs in the original network) (INTEGER).
**ARTYP**—List of arc types (**LDARC** X 1; INTEGER).
1—Canal-flow arc: abcd.
a = + if normal or upper extended flow zone;
- if lower extended flow zone.
b = 1.
c = 0 if not a stream arc;
1 if a stream arc.
d = 0 if normal flow zone:
> 0 if extended flow zone number.
2—Pond-storage arc:
a = + upper storage zone;
- lower storage zone.
b = 2.
c = 0.
d = zone number (1, 2, ... ).
3—Pond net value arc.
4—Pond evaporation arc or seepage arc.
5—Surface-runoff arc.
6—Water-demand arc.
7—Local inflow arc.
8—Canal initial-storage arc.
9—Canal final-storage arc.
10—Canal-seepage arc.
11—Canal-evaporation arc.

**CARBT**—Cumulative arc budgets for whole system (6 x 1; REAL).
1—Inflow.
2—Initial storage.
3—Canal seepage.
4—Canal water-surface evaporation.
5—Final storage.
6—Outflow from the system.
**CNDBT**—Cumulative nodal budget for whole system (0:10 x 1; REAL).
0—Initial storage (acre-feet).
1—Upstream inflow (acre-feet).
2—Local inflow (acre-feet).
3—Evaporation loss for a pond node (acre-feet).

4—Precipitation gain (acre-feet).
5—Seepage loss for pond node (acre-feet).
6—Water withdrawal (acre-feet).
7—Total downstream release (acre-feet).
8—Final storage for pond node (acre-feet).

**COLSTR**—Array of strings for general purposes (**LDCOL** X 1; CHARACTER*30).

**CONST**—Constant coefficient converting acre-feet per month to cubic feet per second (= 86,400 / 43,560 = 1.98347) (REAL).

**COST**—Array of penalty coefficients of arcs (**LDARC** X 1; INTEGER).

**CSARBT**—Array of cumulative water budgets for arcs (**LDARC** x 0:6; REAL).
0—Downstream node of a canal reach.
1—Inflow.
2—Initial storage.
3—Canal seepage.
4—Canal water-surface evaporation.
5—Final storage.
6—Outflow.

**CSNDBT**—Array of cumulative nodal budgets for nodes (**LDND** x 0:10; REAL).
0—Initial storage (acre-feet).
1—Upstream inflow (acre-feet).
2—Local inflow (acre-feet).
3—Evaporation loss for pond node (acre-feet).
4—Precipitation gain (acre-feet).
5—Seepage loss for pond node (acre-feet).
6—Water withdrawal (acre-feet).
7—Total downstream release (acre-feet).
8—Final storage for pond node (acre-feet).

**CTARFW**—List of control arc and lower and upper flow boundaries (**LDCTAR** X 3; INTEGER).
0—Control arc number.
1—Lower flow boundary.
2—Upper flow boundary.

**CTERM**—A temporary string (char*500).

**DEBUG**—Logical variable for optimal solution (LOGICAL).

**ERR**—Error flag (LOGICAL).

**EVFIL**—Logical indicator for time-dependent evaporation from a file (LOGICAL).

**EVFLAG**—Logical indicator for pond water-surface evaporation (LOGICAL).

**EVND**—List of nodes with water-surface evaporation (**LDEV** X 3; INTEGER).
1—Nodal number.
2—Data unit code (0—millimeter per day; 1—inches per day; 2—feet per day).
3—Sequential number for time-dependent data.
If zero, seasonal data will be used. If negative, the default value is used.

**EVTB**—Array of seasonal water-surface evaporation coefficients (0:**LDP** X **LDEV**; REAL).

**EVUNIT**—Evaporation coefficient unit code (INTEGER).

**FBAR**—List of flow-boundary arc information (0:7 X **LDFBAR**; INTEGER).
0—Signed flow-boundary arc.
1—Upstream node.
2—Downstream node.
3—Flow-zone index. index = zone number $\pm 1$ .
(-1 lower boundary of normal flow range;
+1 upper boundary of normal flow range;
<= -2 lower extended flow zone;

>= 2 upper extended flow zone).
4—Next extended arc 1.
5—Next extended arc 2.
6—Data unit code (0—acre-feet; 1—cubic feet per second; 2—cubic feet per day).
7—Seasonal or time-dependent index (0—seasonal, >0—time dependent).

**FBFIL**—Logical indicator for time-dependent flow boundary from a file (LOGICAL).
**FBFLAG**—Logical indicator for existing flow-boundary arcs (LOGICAL).
**FBTB**—List of seasonal or time-dependent flow boundaries for selected arcs (0:**LDFBTB** x **LDFBAR**; REAL).
**FBUNIT**—Unit code for flow boundaries (INTEGER).
(0—acre-feet; 1—cubic feet per second; 2—cubic feet per day).
**FCRIT**—Flow-convergence criterion (INTEGER).
**FILNAM**—List of data and output file names (0:**LDFIL** X 1; CHAR*30).
**FLAG**—Logical indicator (LOGICAL).
**FLOW**—Array of flows in arcs (**LDARC** X 1; INTEGER).
**FLWARC**—Logical indicator for canal routing (LOGICAL).
**FWFLAG**—Indicator for existing local incremental inflow (LOGICAL).
**FXAR**—List of fixed-flow arc and associated upstream and downstream nodes (0:2 X **LDFXAR**; INTEGER).
0—Arc number.
1—Upstream node.
2—Downstream node.
**FXFLAG**—Logical variable for fixed flow (LOGICAL).
**FXUNIT**—Fixed-flow unit code (INTEGER).
(0—acre-feet; 1—cubic feet per second; 2—cubic feet per day).

**GWFLAG**—Logical indicator for ground-water discharge (LOGICAL).
**GWLVL**—List of ground-water elevations (**LDGWND** X 1; REAL).
**GWND**—List of nodes with ground-water seepage (**LDGWND** X 3; INTEGER).
1—Nodal number.
2—Data unit code (0—feet; 1—inches; 2—millimeters).
3—Sequential number for time-dependent data. If zero, seasonal data will be used.
**GWTYPE**—Ground-water data type (INTEGER).
(1—elevation; 2—flux).
**GWUNIT**—Ground-water-data unit (0:2 x2; CHARACTER*6).

**HI**—Array of upper flow boundaries of arcs (**LDARC** X 1; INTEGER).
**HYBFLG**—Logical indicator for outputting water budgets through outlet hydraulic structures (LOGICAL).
**HYDAT**—List of hydraulic-structure parameters (**LDHY** X 5; REAL).
1—Base elevation (feet).
2—Weir length or pipe diameter (feet).
3—Weir height or pipe length (feet).
4—Gate opening height (feet) or pipe-friction factor.
5—Pipe-loss factor in constrictions and entrances.
**HYDIR**—Array for structure and associated upstream and downstream nodal names (**LDHY** X 0:2; CHAR*12).
0—Structure name.
1—Upstream nodal name.
2—Downstream nodal name.
**HYOUT**—Array for results for hydraulic-structure operation (**LDHY** X 3; REAL).
1—Flow-through structure (cubic feet per second).
2—Water level of an upstream node (feet).
3—Gate opening height or weir height (feet).
**HYTP**—Array of names of hydraulic-structure types available (0:**LDHYTP** X 1; CHAR*20).
**HYTPCD**—Array of structure type codes and downstream arcs (**LDHY** X 0:1; INTEGER).
0—Structure type code.
1—Rectangle sharp-crested weir.
2—Vertical flat gate on spillway.

3—Vertical flat gate on broad-crested weir.
6—Pipe.
1—Downstream arc number.

**I**—Index variable (INTEGER).
**IFAULT**—Index variable for fault code (INTEGER).
**IFWCD**—Local net inflow unit code (INTEGER).
0—Acre-feet.
1—Cubic feet per second (cubic feet per second).
2—Cubic feet per day (cubic feet per day).
**IFWND**—List of nodes with local incremental flows (**LDIFW** X 3; INTEGER).
1—Nodal number.
2—Data unit code (0—acre-feet; 1—cubic feet per second; 2—cubic feet per day).
3—Sequential number for time-dependent data. IF < 0, use default value.
**II**—An array of upstream nodes for directed arcs (**LDARC** X 1; INTEGER).
**IN**—FORTRAN file unit for general network data file (INTEGER).
**INST**—Array of initial storages at each time step (**LDRES** X 1; REAL).
**IN_EV**—FORTRAN file unit for evaporation coefficient (INTEGER).
**IN_FB**—FORTRAN file unit for time-dependent flow boundaries (INTEGER).
**IN_FX**—FORTRAN file unit for fixed-flow data (INTEGER).
**IN_GW**—FORTRAN file unit for ground-water-level data (INTEGER).
**IN_IFW**—FORTRAN file unit for local incremental inflow (INTEGER).
**IN_RC**—FORTRAN file unit for time-dependent rule curve (INTEGER).
**IN_RN**—FORTRAN file unit for precipitation data (INTEGER).
**IN_WS**—FORTRAN file unit for target water-demand data (INTEGER).
**ISGN**—INTEGER function for the sign of an INTEGER (INTEGER).

**J**—Index variable (INTEGER).
**JJ**—Array of downstream nodes of directed arcs (**LDARC** X 1; INTEGER).

**KARC**—The current out-of-kilter arc (INTEGER).

**LAST**—Logical indicator at the ending of a file (LOGICAL).
**LDARC**—Leading dimension for number of arcs in a network (INTEGER).
**LDCOL**—Leading dimension for the array **COLSTR** (INTEGER).
**LDCTAR**—Leading dimension for number of control arcs (INTEGER).
**LDEV**—Leading dimension for number of nodes with water-surface evaporation (INTEGER).
**LDFBAR**—Leading dimension for flow-boundary arcs (INTEGER).
**LDFBTB**—Leading dimension for flow-boundary data matrix (INTEGER).
**LDFIL**—Leading dimension for number of input-output files (INTEGER).
**LDFXAR**—Leading dimension for number of fixed-flow arcs (INTEGER).
**LDGWND**—Leading dimension for number of nodes with ground-water data (INTEGER).
**LDHY**—Leading dimension for number of hydraulic structures (INTEGER).
**LDHYTP**—Leading dimension for number of outlet-structure types (INTEGER).
**LDIFW**—Leading dimension for number of nodes with local incremental inflow (INTEGER).
**LDITR**—Leading dimension for number of iterations (INTEGER).
**LDND**—Leading dimension for number of nodes (INTEGER).
**LDP**—Leading dimension for number of seasons in a year (INTEGER).
**LDPL**—Leading dimension for pool space (INTEGER).
**LDRAIN**—Leading dimension for number of nodes with precipitation (INTEGER).
**LDRC**—Leading dimension for number of nodes with time-dependent rule curves (INTEGER).
**LDRCTB**—Leading dimension for rule-curve table (INTEGER).
**LDREAR**—Leading dimension for number of pond arcs (INTEGER).
**LDRES**—Leading dimension for number of pond nodes (INTEGER).
**LDRETB**—Leading dimension for number of pond-characteristic table (INTEGER).

**LDRFTB**—Leading dimension for runoff data table **RNOFTB** (INTEGER).
**LDRNOF**—Leading dimension for number of nodes with the surface-runoff data (INTEGER).
**LDSABL**—Leading dimension for number of single-arc budget lists (INTEGER).
**LDSNBL**—Leading dimension for number of single-node budget lists (INTEGER).
**LDSTR**—Leading dimension for number of stream arcs (INTEGER).
**LDSTRM**—Leading dimension for number of stream-routing arcs (INTEGER).
**LDUNIT**—Leading dimension for number of units used (INTEGER).
**LDWS**—Leading dimension for number of water-demand nodes (INTEGER).
**LO**—Array of lower flow boundaries (**LDARC** X 1; INTEGER).

**MNTH**—Current season in characters (CHARACTER*5).
**MTH**—Current season in numbers such as 1, 2, ... (INTEGER).
**MXCST**—Maximum value of penalty coefficients (INTEGER).

**NARCND**—Number of arc nodes (that is, stream nodes) (INTEGER).
**NARCS**—Number of arcs in a network (INTEGER).
**NDBFLG**—Logical indicator for outputting the nodal water budgets (LOGICAL).
**NDDWAR**—Array of downstream flow arcs from nodes (**LDARC** X 1; INTEGER).
**NDNAM**—List of nodal names (**LDND** X 1; CHAR*12).
**NDSEQ**—List of nodal sequential numbers (**LDND** X 1; INTEGER).
**NDTYP**—List of nodal types (**LDND** X 1; INTEGER).
1—Pond node.
2—General node.
**NDWAR**—Number of downstream arcs (INTEGER).
**NDXAR**—List of signed extensional arcs associated with a node (**LDND** X 6; INTEGER).
1—Pond net-value arc or local net inflow arc.
2—Water-surface-evaporation arc.
3—Precipitation arc.
4—Pond-seepage arc.
5—Target water-demand arc.
6—Target water-demand deviation arc.
**NEV**—Number of evaporation nodes (INTEGER).
**NFBAR**—Number of flow-boundary arcs (INTEGER).
**NFXAR**—Number of arcs with time-dependent, fixed flows (INTEGER).
**NGWND**—Number of ground-water nodes (INTEGER).
**NHY**—Number of hydraulic structures (INTEGER).
**NHYTP**—Number of outlet structure types supported (INTEGER).
**NIFW**—Number of nodes with local incremental inflows (INTEGER).
**NITR**—Number of iteration steps (INTEGER).
**NNODS**—Number of active nodes in a network (that is, original nodes) (INTEGER).
**NODBUD**—List of budget terms for nodal budgets (**LDND** X 0:10; INTEGER).
0—Initial storage for a pond node (acre-feet).
1—Upstream inflow (acre-feet).
2—Local inflow (acre-feet).
3—Evaporation loss for pond node (acre-feet).
4—Precipitation runoff (acre-feet).
5—Seepage loss for pond node (acre-feet).
6—Water withdrawal (acre-feet).
7—Total downstream release (acre-feet).
8—Final storage for pond node (acre-feet).
**NOP**—Index for the current time period (INTEGER).
**NOTCOV**—Logical convergence indicator (LOGICAL).
**NPER**—Number of seasons in a year (INTEGER).
**NRAIN**—Number of nodes with precipitation (INTEGER).
**NRCND**—Number of nodes with time-dependent, rule curves (INTEGER).

**NRES**—Number of pond nodes (INTEGER).
**NRNOF**—Number of surface-runoff nodes (INTEGER).
**NSABL**—Number of arcs needed to output single-arc budget (INTEGER).
**NSNBL**—Number of nodes needed to output nodal budget (INTEGER).
**NSPS**—Number of simulation periods (INTEGER).
**NSTR**—Number of stream reaches with geometrical data (INTEGER).
**NSTRM**—Number of stream-routing arcs (INTEGER).
**NTLS**—Number of non-data lines in a data file (INTEGER).
**NWSND**—The number of water-demand nodes (INTEGER).

**OARCS**—Old arc (INTEGER).
**OCF**—Array of old stream coefficients (canal storage) (**LDSTRM** X 1; REAL).
**OFLOW**—Array of old flows (**LDARC** X 1; INTEGER).
**OHI**—Array of old high boundaries of arcs (**LDARC** X 1; INTEGER).
**OINST**—List of old initial pond storages (**LDRES** X 1; REAL).
**OU_AR**—FORTRAN unit for outputting arc budgets (INTEGER).
**OU_HY**—FORTRAN unit for outputting operating hydraulic structures (INTEGER).
**OU_ND**—FORTRAN unit for outputting nodal budgets (INTEGER).
**OU_NT**—FORTRAN unit for outputting network configuration (INTEGER).
**OU_SA**—FORTRAN unit for file listing arcs for which single-arc budgets are needed (INTEGER).
**OU_SN**—FORTRAN unit for file listing nodes for which single-node budgets are needed (INTEGER).

**PERD**—Number of days for each time period (REAL).
**PN**—Part number (INTEGER).
**PTDWAR**—Pointer of downstream arcs from a node (**LDND** X 2; INTEGER).
 1—Pointer for first downstream arc.
 2—Pointer for the last downstream arc.
**PTRE**—Pointer of pond-storage arcs for pond nodes (**LDRES** X 1; INTEGER).
**PTRES**—Pointer of pond nodes (that is, list of pond nodes) (**LDRES** X 1; INTEGER).

**RAINND**—List of nodes with precipitation (LDRAIN X 3; INTEGER).
 1—Nodal number.
 2—Data unit code (0—feet; 1—inches; 2—millimeters).
 3—Sequential number for time-dependent data. If zero, seasonal data will be used.
**RAINTY**—Surface-runoff data type (INTEGER).
 (1—precipitation; 2—flux)
**RC**—Array of pond rule curves in volume (**LDRES** X 1; REAL).
**RCFIL**—Logical indicator for time-dependent, rule curves from a file (LOGICAL).
**RCFLAG**—Logical indicator for time-dependent, rule curves (LOGICAL).
**RCND**—List of nodes with time-dependent, rule curves (**LDRC** X 3; INTEGER).
 1—Nodal number.
 2—Data unit code (0—feet; 1—inches; 2—millimeters).
 3—Sequential number for time-dependent data. If zero, seasonal data will be used.
**RCTB**—Array of time-dependent, rule curves (0:**LDRCTB** X **LDRC**; INTEGER).
**RCUNIT**—Rule-curve unit code (INTEGER).
**REAR**—Array of pond-storage arcs (**LDREAR** X 1; INTEGER).
**RESDAT**—Pond-bottom information (**LDRES** X 0:2; REAL).
 0—Pond-bottom elevation (feet).
 1—Pond-bed hydraulic conductivity (feet per day).
 2—Pond-bed thickness (feet).
**REZN**—Array of pond-storage zone capacities (**LDREAR** X 1; REAL).
**RNFLAG**—Indicator for existing rain data (LOGICAL).
**RNOFND**—List of nodes with surface runoff (**LDRNOF** X 1; INTEGER).
**RNOFTB**—Array of surface-runoff data (**LDRNOF** X 0:**LDRFTB**; REAL).
 0—Initial 5-day antecedent rainfall, in inches.

1—5-day antecedent rainfall for dry condition, in inches.
2—5-day antecedent rainfall for wet condition, in inches.
3—Drainage area, in acres.
4—SCS curve number for average condition (II).

**RNUNIT**—Precipitation unit code (INTEGER).
(0—feet; 1—inches; 3—millimeters)

**RPOOL**—Pool space (**LDPL** X 1; REAL).

**RTERM**—Temporary REAL number (REAL).

**SABLFG**—Logical indicator for single-arc budget list (LOGICAL).

**SABLND**—List of arcs with single-arc budget list (**LDSABL** X 3; INTEGER).
1—Upstream node number.
2—Downstream node number.
3—FORTRAN unit for output budget.

**SAVOPT**—Save option for general output (INTEGER).
0—Both input data-file information and network configuration.
1—Input data-file information.
2—Network configuration.

**SKSC**—Sink/source node (INTEGER).

**SNBLFG**—Logical indicator for single-node budget list (LOGICAL).

**SNBLND**—List of nodes with single-node budget list (**LDSNBL** X 3; INTEGER).
1—Nodal number.
2—FORTRAN unit for water-budget output.
3—FORTRAN unit for downstream water releases.

**STMO**—Starting season of simulation period in number (INTEGER).

**STRDAT**—List of canal-geometry data (**LDSTR** X 9; REAL).
1—Canal-bed roughness (dimensionless).
2—Canal-reach length (feet).
3—Average canal-bed slope (dimensionless).
4—Canal-base width (feet).
5—Canal-side slope (dimensionless).
6—Canal depth (feet).
7—Canal-bed hydraulic conductivity (feet per day).
8—Thickness of canal bed (feet).
9—Canal-bed elevation at reach entry (feet above sea level).

**STRDIR**—List of nodes associated with stream arcs (**LDSTR** X 3; INTEGER).
1—Upstream node.
2—Downstream node.
3—Middle routing node.

**STRMAR**—List of arcs for canal routing. Arc number is signed (**LDSTRM** X 0:6; INTEGER).
0—Stream nodal number.
1—Signed inflow arc number.
2—Signed initial canal-storage arc number.
3—Canal-seepage arc number.
4—Evaporation arc.
5—Final storage arc.
6—Outflow arc.

**STRMCF**—List of canal-routing coefficients (**LDSTRM** X 0:4; REAL).
0—Canal initial storage (acre-feet).
1—Traveltime ($K$) (days).
2—Weighting factor ($x$) (0–0.5, dimensionless).
3—Seepage coefficient (dimensionless).
$> 0$, in percentage of inflow ($<1.0$, $0.01 \sim 0.02$).
$< 0$, seepage will be calculated by Darcy's law.
4—Evaporation coefficient (inches per day).

**SYSNAM**—Name of a system of ponds (CHAR*30).

**UNITNM_1**—Unit name for discharge (0:2 X 1; CHAR*6).
0—acre-feet.
1—cubic feet per second.
2—cubic feet per day.
**UNITNM_2**—Unit name for length (0:2 X 1; CHAR*6).
0—feet.
1—inches.
2—millimeters.
**UNITNM_3**—Unit name for rate (0:2 X 1; CHAR*6).
0—millimeters per day.
1—inches per day.
2—feet per day.

**WSFIL**—Logical indicator for target water demand from a file (LOGICAL).
**WSFLAG**—Logical indicator for target water demand (LOGICAL).
**WSND**—List of nodes with target water demand (LDWS X 3; INTEGER).
1—Nodal number.
2—Data unit code (0—acre-feet; 1—cubic feet per second; 2—cubic feet per day).
3—Sequential number for time-dependent data. If zero, seasonal data will be used.
**WSTB**—Array of time-dependent or seasonal target water demands (0:**LDP** X **LDWS**; REAL).
**WSUNIT**—Target water-demand unit code (INTEGER).
(0—acre-feet; 1—cubic feet per second; 2—cubic feet per day).

**XF**—Exaggerator factor (REAL).
**XP**—Number of decimal points for water budget, in acre-feet (INTEGER).

**YEAR**—Current year (INTEGER).
**YES**—Logical variable for yes or no (LOGICAL).
**YR**—Year (INTEGER).

**ZEROFG**—Logical indicator for zero (LOGICAL).

# APPENDIX D. CONCEPTUAL SAMPLE PROBLEM

The conceptual sample problem is intended to illustrate the use of the computer program OPONDS for operation of a system of canals and control ponds, especially input data files and output from the program. The conceptual flow system used in this sample problem contains three ponds, three pumping stations for water supply, and three hydraulic structures to control water releases from the ponds (fig. 17). The sample problem specifies that three ponds should be operated to satisfy the downstream water demands at three pumping stations and the minimum flow requirements in channels and to keep pond water levels as close to target levels as possible. In the following section, the input data files for the sample problem are discussed first, and then output files are presented. The data in this sample problem are hypothetical.

## Input Data for Conceptual Sample Problem

There are 18 input data files for this sample problem. Contents of these files are described in the following section. For the file format and variable definitions, refer to the input/output instructions in Appendix B.

### Master Data File (mdf.dat)

The master data file is used to list all file names for input and output and their associated file-identification codes. The contents of master data file for the sample problem are listed as follows:

```
List of sample data files for the sample problem
 File
 code   File name      File Description
 ------ -----------    -----------------------------
      0 nsim.dat     : Network-configuration file
      1 pdzn.dat     : Pond-zoning file
      2 ntwk.dat     : Network flow-configuration file
      3 geom.dat     : Canal-geometry file
      4 hydr.dat     : Hydraulic-structure file
      5 rnof.dat     : Runoff-related data file
      9 pzva.dat     : Pond-characteristics table file
     10 tws.dat      : Pond seasonal target water-demand file
     11 pet.dat      : Pond seasonal water-surface evaporation file
     13 rc.dat       : Seasonal rule-curve file
     16 ifw.dat      : Local incremental flow file
     17 rain.dat     : Precipitation file
     18 pet2.dat     : Water-surface evaporation file
     19 tws2.dat     : Target water-demand file
     20 rc2.dat      : Rule-curve data
     22 gwl.dat      : Ground-water-level file
     26 nsim.out     : Network-configuration output
     27 ndbt.out     : Nodal-budget output
     28 arbt.out     : Arc-budget output
     29 hydr.out     : Flow-through hydraulic-structure output
     30 snbl.dat     : List of nodes for time-series output
     31 sabl.dat     : List of arcs for time-series output
```

### General Network-Configuration and Parameter File (nsim.dat)

This file is used to specify the simulation period, accuracy of results, and so forth. It is noted that data for network construction and seasonal data also can be included in this file.

```
Operation of a river system
This is a sample problem.
30.42  12        ! Time interval, and number of time steps in a year
10 1996          ! Starting time period and year.
12               ! Number of simulation periods.
0.1    100       ! Flow-convergence criterion and maximum iteration steps.
2                ! Output accuracy. Number of decimal point in acre-feet.
0                ! Output option.
```

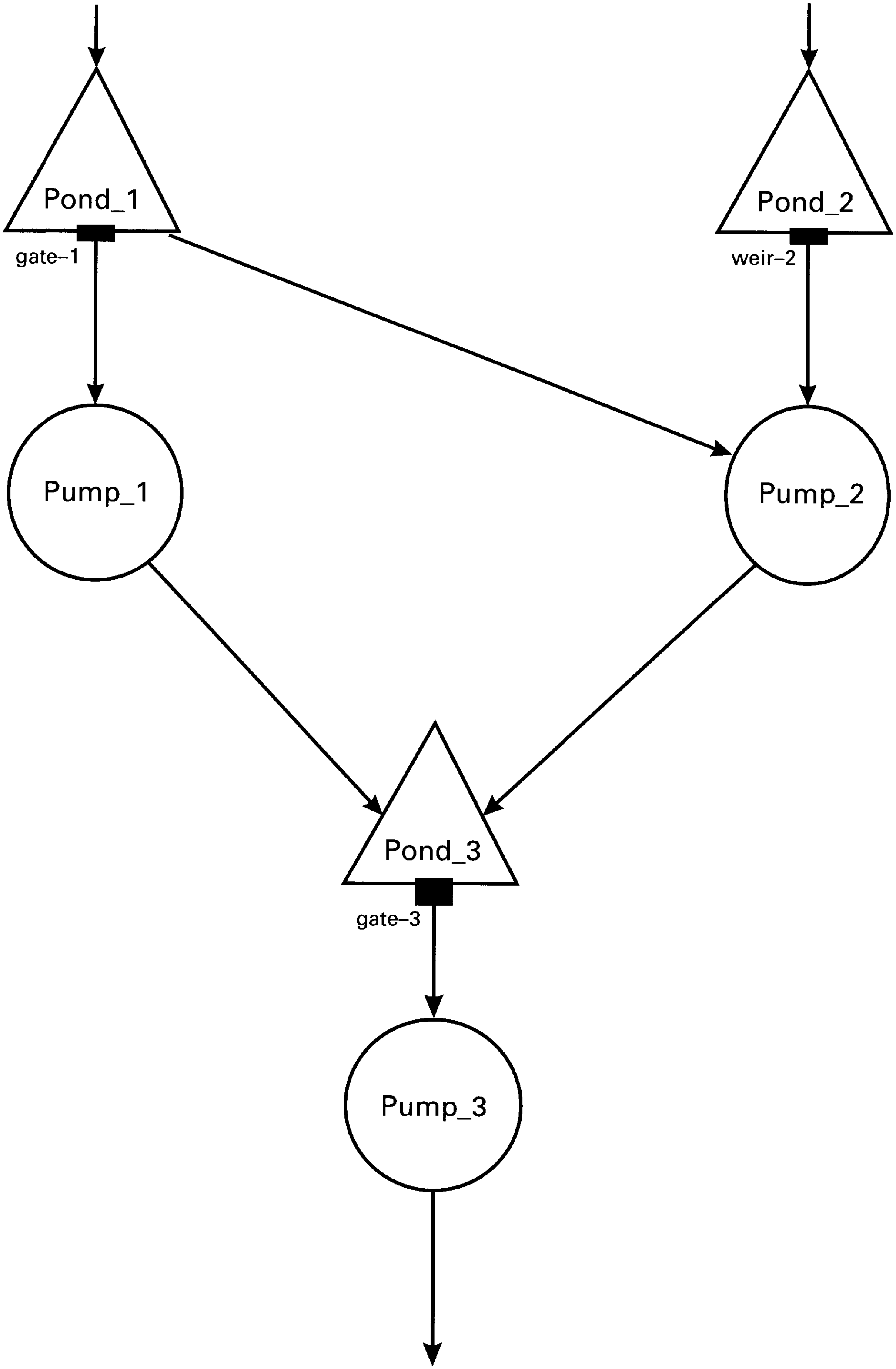



**Figure 17.** Network representation of flow system for sample problem.

## Pond Rule-Curve and Zoning File (pdzn.dat)

Storages of pond_1 and pond_2 are divided into two storage zones (lower zone and upper zone). However, the storage of pond_3 is divided into three zones (lower zone, upper zone, and extended upper zone). The rule curves for the three ponds are located at the top of the lower zones. Pond-bottom hydraulic data also are included in this data file. It is noted that units of storage zone, rule curve, and initial storage must be the same. The units can be either feet or acre-feet and are specified in the unit code.

Sample pond rule curve and zoning.

| Name | Unit | Initial (feet) | Bottom elevation (feet) | Hydraulic cond. (ft/d) | Bottom thickness (feet) | Rule-curve elev. (feet) | Zone_1 elev. (feet) | Zone_1 coef. | Zone_2 elev. (feet) | Zone_2 coef. | Zone_3 elev. (feet) | Zone_3 coef. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| pond_1 | 0 | 1274.00 | 1239.0 | 0. | 1.0 | 1274.00 | 1240.00 | 100 | 1280.00 | 1000 | | |
| POND_2 | 0 | 1350.50 | 1308.0 | 0.0001 | 1.0 | 1350.50 | 1320.00 | 100 | 1365.00 | 1000 | | |
| POND_3 | 0 | 1039.00 | 1010.0 | 0.0001 | 1.0 | 1039.00 | 1020.00 | 100 | 1050.00 | 1000 | 1060.00 | 2000 |

## Network Flow Configuration File (ntwk.dat)

There are three flow zones from pond_1 node to pump_1 node. Canal-routing coefficients, seepage coefficients, and water-surface-evaporation coefficients also are included in this file.

Network flow configuration for sample problem

| F_node | T_node | Lower boundary (ft^3/s) | Upper boundary (ft^3/s) | Cost coef. | Initial storage (acre-ft) | Travel time (day) | Weight factor | Seepage coef. | Evap. coef. (in/d) |
|---|---|---|---|---|---|---|---|---|---|
| pond_1 | PUMP_1 | 0. | 16000. | 10 | 0.0 | 1 | 0.2 | 0.01 | 0.0 |
| pond_1 | PUMP_2 | 0. | 16000. | 100 | 0.0 | 1 | 0.2 | 0.01 | 0.0 |
| POND_2 | PUMP_2 | 0. | 16000. | 10 | 0.0 | 1 | 0.2 | 0.01 | 0.0 |
| POND_3 | PUMP_3 | 0. | 16000. | 10 | 0.0 | 1 | 0.2 | 0.01 | 0.0 |
| PUMP_1 | POND_3 | 10.0 | 16000. | 10 | 0.0 | 1 | 0.2 | -0.01 | -1.0 |
| PUMP_2 | POND_3 | 10.0 | 16000. | 10 | 0.0 | 1 | 0.2 | 0.01 | 0.0 |
| PUMP_3 | sksc | 10.0 | 16000. | 10 | | | | | |

## Canal Geometry Data File (geom.dat)

Canal geometry data are used for estimating water-surface evaporation and water depth of a selected canal.

Canal geometry for the sample problem

| F-Node | T-Node | Rough. coef. | Canal length (feet) | Channel slope | Canal width (feet) | Side slope | Canal depth (feet) | Hydr. cond. (ft/d) | Riverbed thickness (feet) | Bottom elevation (feet) |
|---|---|---|---|---|---|---|---|---|---|---|
| pond_1 | pump_1 | 0.025 | 1000 | 0.01 | 50 | 0 | 15 | 1.0 | 1.0 | 1240.0 |
| pump_1 | pond_3 | 0.025 | 5000 | 0.001 | 50 | 0 | 10 | 1.0 | 1.0 | 1140.0 |
| pump_2 | pond_3 | 0.025 | 500 | 0.01 | 50 | 0 | 30 | 1.0 | 1.0 | 1140.0 |

## Outlet Hydraulic-Structure File (hydr.dat)

Three kinds of hydraulic structures—sharp-crested weir, gate on spillway, and gate on broad-crested weir—are include in the sample problem.

Hydraulic-structure parameters for the sample problem

| Name | F_Node | T_Node | Type | Base elevation (feet) | Weir length or pipe diameter (feet) | Weir height or pipe length (feet) | Gate opening or pipe friction (feet) |
|---|---|---|---|---|---|---|---|
| Gate-1 | Pond_1 | Pump_1 | 2 | 1240.00 | 2.0 | 0 | 1.0 |
| Weir-2 | Pond_2 | Pump_2 | 1 | 1340.00 | 2.0 | -20 | |
| Gate-3 | Pond_3 | Pump_3 | 3 | 1020.00 | 10.0 | 0 | |

## Surface-Runoff File (rnof.dat)

```
Surface-runoff data for the sample problem
        Antecedent 5-   AMC for dry  AMC for wet    SCS     Drainage
        day rainfall    conditions   conditions     curve   area
Name     (inches)       (inches)      (inches)      number  (acres)
------  -------------   -----------  -----------    ------  --------
pump_3       1.0            0.5          1.1          85      10000
```

## Elevation-Volume-Area Relation File (pzva.dat)

```
Elevation-capacity-area relations of three ponds for the sample problem

File format: pond_name
             elevation volume area
Elevation is in feet, volume is in acre-feet; and area is in acres.
0           !Z-V-A type(0--table; 1 -- QNWR regression equations)
pond_1
    1239.0       0.0       0.0
    1240.0      14.0      42.0
    1241.0      87.3     110.0
    1242.0     200.3     116.0
    1243.0     349.0     184.0
    1244.0     536.0     190.0
    1245.0     762.0     264.0
    1246.0    1032.0     276.0
    1247.0    1351.0     364.0
    1248.0    1724.9     384.0
    1249.0    2181.9     534.0
    1250.0    2784.5     600.0
    1251.0    3428.9     764.0
    1252.0    4210.8     800.0
    1253.0    5076.9     934.0
    1254.0    6028.9     970.0
    1255.0    7062.2    1104.0
    1256.0    8187.1    1140.0
    1257.0    9393.5    1274.0
    1258.0   10700.4    1340.0
    1259.0   12121.6    1504.0
    1260.0   13668.4    1590.0
    1261.0   15339.7    1754.0
    1262.0   17146.4    1860.0
    1263.0   19087.9    2024.0
    1264.0   21164.6    2130.0
    1265.0   23376.1    2294.0
    1266.0   15722.9    2400.0
    1267.0   28204.5    2560.0
    1269.0   33513.1    2774.0
    1270.0   36315.1    2830.0
    1271.0   39211.8    2964.0
    1272.0   42210.7    3034.0
    1273.0   45308.5    3162.0
    1274.0   48506.5    3234.0
    1275.0   58118.1    3390.0
    1276.0   55258.0    3490.0
    1277.0   58827.7    3650.0
    1278.0   62532.6    3760.0
    1279.0   66372.3    3920.0
    1280.0   70347.0    4030.0
    1281.0   74456.0    4190.0
    1282.0   78701.8    4300.0
    1283.0   83076.0    4450.0
    1284.0   87578.0    4560.0
    1285.0   92213.0    4710.0
    1286.0   96978.0    4820.0
    1287.0  101892.0    5010.0
    1288.0  106982.0    5170.0
    1289.0  112266.0    5400.0
    1290.0  117764.0    5560.0
    1290.0  123380.0    5710.0
    1292.0  129165.0    5860.0
    1293.0  135095.0    6000.0
    1294.0  141174.0    6160.0
    1295.0  147414.0    6320.0
    1296.0  153823.0    6500.0
    1297.0  160413.0    6680.0
    1298.0  167207.0    6910.0
    1299.0  174212.0    7100.0
    1300.0  181438.0    7340.0
    1301.0  188867.8    7520.0
```

```
     1302.0  196507.5    7760.0
     1303.0  204347.0    7920.0
     1304.0  212397.0    8228.0
     1305.0  220656.8    8340.0
     1306.0  229126.0    8600.0
     1307.0  237831.3    8810.0
     1308.0  246776.0    9080.0
     1309.0  255960.0    9290.0
     1310.0  265405.8    9601.0

pond_2
     1308.0       0.0       0.0
     1309.0       1.0       3.0
     1310.0       4.5       4.0
     1311.0       9.0       5.0
     1312.0      14.5       6.0
     1313.0      21.4       8.0
     1314.0      31.4      12.0
     1315.0      45.3      16.0
     1316.0      64.3      22.0
     1317.0      93.0      36.0
     1318.0     142.3      64.0
     1319.0     227.3     108.0
     1320.0     363.3     166.0
     1321.0     567.1     244.0
     1322.0     853.0     330.0
     1323.0    1240.5     448.0
     1324.0    1744.4     562.0
     1325.0    2384.7     722.0
     1326.0    3175.7     862.0
     1327.0    4137.9    1066.0
     1328.0    5271.2    1202.0
     1329.0    6573.9    1406.0
     1330.0    8037.5    1522.0
     1331.0    9660.4    1726.0
     1332.0   11451.0    1856.0
     1333.0   13419.0    2082.0
     1334.0   15561.7    2204.0
     1335.0   17898.0    2472.0
     1336.0   20442.1    2616.0
     1337.0   23199.8    2902.0
     1338.0   26183.4    3066.0
     1339.0   29401.2    3372.0
     1340.0   32865.8    3558.0
     1341.0   36583.7    3880.0
     1342.0   40528.5    4010.0
     1343.0   44707.3    4350.0
     1344.0   49103.2    4442.0
     1345.0   53730.0    4814.0
     1346.0   58594.9    4916.0
     1347.0   63712.6    5322.0
     1348.0   69090.5    5434.0
     1349.0   74733.2    5854.0
     1350.0   80864.1    5976.0
     1351.0   86849.7    6430.0
     1352.0   93343.6    6558.0
     1353.0  100140.2    7038.0
     1354.0  107256.0    7094.0
     1355.0  114710.5    7718.0
     1356.0  122532.2    7926.0
     1357.0  130752.5    8518.0
     1358.0  139402.1    8782.0
     1359.0  148449.8    9316.0
     1360.0  157889.5    9584.0
     1361.0  167748.3   10116.0
     1362.0  178022.9   10434.0
     1363.0  188741.0   11006.0
     1364.0  199911.2   11334.0
     1365.0  211534.5   11915.0
     1366.0  223636.1   12249.0

pond_3
     1008.0       0.0       0.0
     1009.0       0.0       0.0
     1010.0       3.0       8.4
     1011.0      20.0      27.4
     1012.0      50.0      32.7
     1013.0      84.0      35.3
     1014.0     122.0      40.8
     1015.0     164.0      43.3
     1016.0     212.0      52.9
     1017.0     269.0      61.2
```

```
    1018.0     335.0      70.9
    1019.0     411.0      81.2
    1020.0     505.0     107.4
    1021.0     666.0     221.4
    1022.0     966.0     386.2
    1023.0    1451.0     591.0
    1024.0    2171.0     857.2
    1025.0    3167.0    1141.5
    1026.0    4467.0    1465.2
    1027.0    6089.0    1784.1
    1028.0    8054.0    2151.7
    1029.0   10517.0    2788.0
    1030.0   13576.0    3338.2
    1031.0   17332.0    4189.9
    1032.0   21788.0    4727.5
    1033.0   27038.0    5790.4
    1034.0   32947.0    6028.4
    1036.0   46511.0    7193.9
    1037.0   54164.0    8125.5
    1038.0   62433.0    8417.4
    1039.0   71284.0    9291.8
    1040.0   80723.0    9587.0
    1041.0   90745.0   10463.4
    1042.0  101314.0   10795.4
    1043.0  112626.0   11714.8
    1044.0  124525.0   12084.1
    1045.0  137072.0   13015.6
    1046.0  150271.0   13383.2
    1047.0  164118.0   14316.0
    1048.0  178617.0   14682.8
    1049.0  193774.0   15636.2
    1050.0  209603.0   16022.6
    1051.0  226110.0   16996.2
    1052.0  243309.0   17402.6
    1053.0  261221.0   18426.3
    1054.0  279875.0   18882.6
    1055.0  299292.0   19956.3
    1056.0  319495.0   20450.7
    1057.0  340497.0   21558.2
    1058.0  362316.0   22080.9
    1059.0  384963.0   23217.9
    1060.0  408461.0   23779.2
    1061.0  432828.0   24959.5
    1062.0  458087.0   25559.6
    1063.0  484273.0   26817.4
    1064.0  514131.0   27500.0
    1065.0  539588.0   28819.1
    1066.0  568766.0   29538.4
    1067.0  598982.0   30898.7
    1068.0  930250.0   31638.7
    1069.0  662487.0   32839.0
    1070.0  695834.0   33857.6
    1071.0  730152.0   34780.5
    1073.0  802016.0   36862.2
    1074.0  839592.0   38294.3
    1075.0  878315.0   39153.3
    1076.0  918206.0   40633.3
    1077.0  959294.0   41544.4
    1078.0 1001594.0   43063.0
    1079.0 1045132.0   44017.6
    1080.0 1089926.0   45574.9

FINISH
```

## Seasonal Target Water-Demand File (tws.dat)

```
Seasonal water demands for the sample problem
(values are in acre-feet.
>
>
>
0     :unit code (0 -- acre-ft, 1 -- ft^3/s, 2 -- ft^3/d)
Date  PUMP_2     pump_1  PUMP_3
Jan.     343        619    2580
Feb.     310        599    2330
March    343        619    2580
April    332        599    2496
May      343        619    2580
June     332        599    2496
July     343        619    2580
```

```
Aug.     343       619    2580
Sept.    332       599    2496
Oct.     343       619    2580
Nov.     332       599    2496
Dec.     343       619    2580
```

## Seasonal Water-Surface-Evaporation Coefficient File (pet.dat)

```
Sample seasonal pond water-surface evaporation coefficients
(Values are in inches per day)
>
>
>
1  :unit code(0 -- mm/d, 1 -- in/d, 2 -- ft/d)
Date      pond_2   pond_1
Jan.       0.1       0.01
Feb.       0.1       0.01
March      0.1       0.01
April      0.1       0.01
May        0.1       0.01
June       0.1       0.01
July       0.1       0.01
Aug.       0.1       0.01
Sept.      0.1       0.01
Oct.       0.1       0.01
Nov.       0.1       0.01
Dec.       0.1       0.01
```

## Seasonal Rule-Curve File (rc.dat)

```
Seasonal rule-curve data for the sample problem
>
>
>
>
0   :Unit code(0 -- feet; 1 -- inches; 2 -- millimeters)
Date     Pond_2
Jan.     1350.5
Feb.     1355
March    1360
April    1355
May      1350
June     1350
July     1360
Aug.     1360
Sept.    1360
Oct.     1360
Nov.     1360
Dec.     1360
```

## Local Net Inflow File (ifw.dat)

```
Local net inflows to three ponds for the sample problem
<
<
<
<
0    :Data unit code (0 -- acre-ft, 1 -- ft^3/s, 2 -- ft^3/d)
Date       POND_1    POND_2    POND_3
Oct.      1729.16   1494.33  36962.78
Nov.      1593.61   1274.00  32579.50
Dec.     13222.35   9886.98  66895.89
Jan.     19412.21  13013.84 171310.33
Feb.      6358.12   7925.43 203225.77
March      723.75   2390.50 118321.28
April     -407.64     64.59  67050.85
May       2926.71    122.61  31301.72
June      -840.34  -1525.57  22908.99
July      -435.46  -1355.91   8527.78
Aug.      -126.75     94.50     66.27
Sept.       77.64    314.43   7617.22
```

## Precipitation File (rain.dat)

```
Total rainfall to ponds for the sample problem
These value are in inches.
<
<
<
1          :Unit code(0 -- feet; 1 -- inches; 2 -- millimeters)
Date     pond_2  Default
Oct.      2.0     0.1
```

## File for Time-Dependent, Water-Surface-Evaporation Coefficients (pet2.dat)

```
Time-dependent pond water-surface
evaporation coefficients for the sample problem
<
<
<
1     :Unit code (0 -- mm/d, 1 -- in/d, 2 -- ft/d)
Date  pond_3     pond_1 Default
Oct.    0.15        0.1   0.01
Nov.    0.15        0.1   0.01
Dec.    0.15        0.1   0.01
Jan.    0.15        0.1   0.01
Feb.    0.15        0.1   0.01
March   0.15        0.1   0.01
April   0.15        0.1   0.01
May     0.15        0.1   0.01
June    0.15        0.1   0.01
July    0.15        0.1   0.01
Aug.    0.15        0.1   0.01
Sept.   0.15        0.1   0.01
```

## File for Time-Dependent, Target Water Demands (tws2.dat)

```
Time-dependent water demands for the sample problem
<
<
<
<
0     :Unit code (0 -- acre-ft; 1 -- ft^3/s; 2 -- ft^3/d)
Date        pond_3  PUMP_3
Oct.          2580    2600
Nov.          2330    2330
Dec.          2580    2580
Jan.          2496    2496
Feb.          2580    2580
March         2496    2496
April         2580    2580
May           2580    2580
June          2496    2496
July          2580    2580
Aug.          2496    2496
Sept.         2580    2580
```

## File for Time-Dependent, Pond Rule Curves (rc2.dat)

```
Time-dependent rule curves for the sample problem
<
<
<
<
0   :Unit code(0 -- feet; 1 -- inches; 2 -- millimeters)
Date  Pond_1
Oct.  1274.0
Nov.  1270.0
Dec.  1265.0
Jan.  1275.0
Feb.  1275.0
March 1275.0
April 1275.0
May   1275.0
June  1275.0
July  1275.0
Aug.  1275.0
Sept. 1275.0
```

### File for Ground-Water Elevations (gwl.dat)

```
Ground-water elevations for the sample problem
Elevations are in feet above sea level.
<
<
<
0   :Unit code(0 -- feet; 1 -- inches; 2 -- millimeters)
Date      POND_1    POND_2    POND_3    Pump_1    Pump_2    pump_3
Oct.      1276.0    1355.0    1045.0    1250.0    1250.0    1150.0
Nov.      1276.0    1355.0    1045.0    1250.0    1250.0    1150.0
Dec.      1276.0    1355.0    1045.0    1250.0    1250.0    1150.0
Jan.      1276.0    1355.0    1045.0    1250.0    1250.0    1150.0
Feb.      1276.0    1355.0    1045.0    1250.0    1250.0    1150.0
March     1276.0    1355.0    1045.0    1250.0    1250.0    1150.0
April     1276.0    1355.0    1045.0    1250.0    1250.0    1150.0
May       1276.0    1355.0    1045.0    1250.0    1250.0    1150.0
June      1276.0    1355.0    1045.0    1250.0    1250.0    1150.0
July      1276.0    1355.0    1045.0    1250.0    1250.0    1150.0
Aug.      1276.0    1355.0    1045.0    1250.0    1250.0    1150.0
Sept.     1276.0    1355.0    1045.0    1250.0    1250.0    1150.0
```

### File for List of Nodes for Time-Series Water-Budget Output (snbl.dat)

This file is used to list nodes for which the time-series output of water budgets is needed. For each node specified, two output files are generated—one for nodal water budget (for example, bpond_1.out) and the other for water releases to downstream nodes (for example, rpond_1.out).

```
pond_1
pond_2
pond_3
pump_1
pump_2
pump_3
```

### File for List of Canals for Water-Budget Output (sabl.dat)

This file is used to list canal reaches for which time-series output of water budgets is needed. In this sample problem, only three canals are listed for the time-series output of water budgets. The names of output files are automatically generated from the OPONDS program with the form arbud###.dat, where ### is the sequential number such as 001, 002, and 003 in the list file. For example, the output file name for canal reach from pond_1 to pump_1 is arbud001.dat.

```
pond_1  pump_1
pond_2  pump_2
pump_1  pond_3
```

## Sample Problem Output

Four general output files are generated from the OPONDS program for the sample problem. The first one is for the input information and network configuration, the second one is for nodal water budgets, the third one is for canal water budgets, and the final one is for hydraulic-structure operations. The output file names are specified in the master data file. In additional to general outputs, there are two kinds of optional water-budget outputs in time-series format for specified nodes and canal reaches. It should not be expected that the outputs from running the same problem on different computers will match exactly. Small variations in output can be caused by differences in the way real numbers are stored and calculated. Storage and calculation of real numbers depend on the specific computer hardware, the FORTRAN compiler, and the math library that is loaded with the compiled program. Output variations among computers also depend on the size of the problem, the number of iterations required for solution, and the precision used when printing results.

## Output File for Input Information and Network Configuration (nsim.out)

```
Operation of a river system.

Summary of simulation period:
============================
          Length of a time step:   30.42 days
  Number of time steps of a year:   12
                Starting season:   10
                  Starting year: 1996
 Number of simulation time steps:   12
      Flow-convergence criterion:    0.100 cubic feet per second
        Maximum iteration steps:  100
        Number of decimal points:    2


Summary of pond elevation-volume-area relations:
================================================
                              Minimum    Maximum
                    Number of elevation   elevation
   No  Name           records    (feet)    (feet)
  ---  ------------  --------- --------- ---------
    1  POND_1              71   1239.00   1310.00
    2  POND_2              59   1308.00   1366.00
    3  POND_3              71   1008.00   1080.00


Part 1: Pond-storage zoning information
======================================

POND_1                1                  :Pond name and corresponding node number
        1239.00    0.00        1.00      :Bottom elevation, hydraulic conductivity, and thickness
       48506.50                          :Initial storage, in acre-feet
        1274.00                          :Rule-curve elevation, in feet
    1   1240.00    100                   :Zone number, elevation, and penalty coefficient
    2   1280.00   1000                   :Zone number, elevation, and penalty coefficient

POND_2                2                  :Pond name and corresponding node number
        1308.00    0.00        1.00      :Bottom elevation, hydraulic conductivity, and thickness
       83856.91                          :Initial storage, in acre-feet
        1350.50                          :Rule-curve elevation, in feet
    1   1320.00    100                   :Zone number, elevation, and penalty coefficient
    2   1365.00   1000                   :Zone number, elevation, and penalty coefficient

POND_3                3                  :Pond name and corresponding node number
        1010.00    0.00        1.00      :Bottom elevation, hydraulic conductivity, and thickness
       71284.00                          :Initial storage, in acre-feet
        1039.00                          :Rule-curve elevation, in feet
    1   1020.00    100                   :Zone number, elevation, and penalty coefficient
    2   1050.00   1000                   :Zone number, elevation, and penalty coefficient
    3   1060.00   2000                   :Zone number, elevation, and penalty coefficient

Part 2: Network flow configuration
=====================================
```

| | | Lower boundary (cubic feet per second) | Upper boundary (cubic feet per second) | Penalty coefficient | Initial storage (acre-feet) | Travel-time (day) | Weighting factor | Seepage coefficient | Evaporation coefficient (inches per day) |
|---|---|---|---|---|---|---|---|---|---|
| POND_1 | PUMP_1 | 0.00 | 16000.00 | 10 | 0.00 | 1.00 | 0.20 | 0.01 | 0.00 |
| POND_1 | PUMP_2 | 0.00 | 16000.00 | 100 | 0.00 | 1.00 | 0.20 | 0.01 | 0.00 |
| POND_2 | PUMP_2 | 0.00 | 16000.00 | 10 | 0.00 | 1.00 | 0.20 | 0.01 | 0.00 |
| POND_3 | PUMP_3 | 0.00 | 16000.00 | 10 | 0.00 | 1.00 | 0.20 | 0.01 | 0.00 |
| PUMP_1 | POND_3 | 10.00 | 16000.00 | 10 | 0.00 | 1.00 | 0.20 | -0.01 | -1.00 |
| PUMP_2 | POND_3 | 10.00 | 16000.00 | 10 | 0.00 | 1.00 | 0.20 | 0.01 | 0.00 |
| PUMP_3 | SKSC | 10.00 | 16000.00 | 10 | | | | | |

```
PART 3: Canal geometry data
===========================
```

| F-node name | T-node name | Roughness coefficient | Canal length (feet) | Canal slope | Bottom width (feet) | Side slope | Canal depth (feet) | Riverbed hydraulic conductivity (feet per day) | Riverbed thickness (feet) | Entry elevation (feet) |
|---|---|---|---|---|---|---|---|---|---|---|
| POND_1 | PUMP_1 | 0.0250 | 1000.00 | 0.010000 | 50.00 | 0.000000 | 15.00 | 1.00 | 1.00 | 1240.00 |
| PUMP_1 | POND_3 | 0.0250 | 5000.00 | 0.001000 | 50.00 | 0.000000 | 10.00 | 1.00 | 1.00 | 1140.00 |
| PUMP_2 | POND_3 | 0.0250 | 500.00 | 0.010000 | 50.00 | 0.000000 | 30.00 | 1.00 | 1.00 | 1140.00 |

Part 4: Hydraulic-structure data

| Structure name | F-node name | T-node name | Structure type | Base elevation (feet) | Weir width or pipe diameter (feet) | Weir height or pipe length (feet) | Pipe-friction factor | Pipe-entry loss factor |
|---|---|---|---|---|---|---|---|---|
| Gate-1 | Pond_1 | Pump_1 | Gate spillway | 1240.00 | 2.00 | 0.00 | 1.00000 | 0.00000 |
| Weir-2 | Pond_2 | Pump_2 | Sharp-crested weir | 1340.00 | 2.00 | -20.00 | 0.00000 | 0.00000 |
| Gate-3 | Pond_3 | Pump_3 | Sluice gate | 1020.00 | 10.00 | 0.00 | 0.00000 | 0.00000 |

Part 5: Surface-runoff data

| Name | Antecedent 5-day rainfall (inches) | Initial criterion 5-day rainfall: Dry condition (inches) | Initial criterion 5-day rainfall: Wet condition (inches) | SCS curve number | Drainage area (acres) |
|---|---|---|---|---|---|
| PUMP_3 | 1.00 | 0.50 | 1.10 | 85.00 | 10000.00 |

Part 10: Seasonal target water demands, in acre-feet

| SEASON | PUMP_2 | PUMP_1 | PUMP_3 |
|---|---|---|---|
| 1 | 343.00 | 619.00 | 2580.00 |
| 2 | 310.00 | 599.00 | 2330.00 |
| 3 | 343.00 | 619.00 | 2580.00 |
| 4 | 332.00 | 599.00 | 2496.00 |
| 5 | 343.00 | 619.00 | 2580.00 |
| 6 | 332.00 | 599.00 | 2496.00 |
| 7 | 343.00 | 619.00 | 2580.00 |
| 8 | 343.00 | 619.00 | 2580.00 |
| 9 | 332.00 | 599.00 | 2496.00 |
| 10 | 343.00 | 619.00 | 2580.00 |
| 11 | 332.00 | 599.00 | 2496.00 |
| 12 | 343.00 | 619.00 | 2580.00 |

Part 11: Seasonal water-surface evaporation coefficients, in inches per day

| SEASON | POND_2 | POND_1 |
|---|---|---|
| 1 | 0.10 | 0.01 |
| 2 | 0.10 | 0.01 |
| 3 | 0.10 | 0.01 |
| 4 | 0.10 | 0.01 |
| 5 | 0.10 | 0.01 |
| 6 | 0.10 | 0.01 |
| 7 | 0.10 | 0.01 |
| 8 | 0.10 | 0.01 |
| 9 | 0.10 | 0.01 |
| 10 | 0.10 | 0.01 |
| 11 | 0.10 | 0.01 |
| 12 | 0.10 | 0.01 |

Part 13: Seasonal rule-curve elevations, in feet

| SEASON | POND_2 |
|---|---|
| 1 | 1350.50 |
| 2 | 1355.00 |
| 3 | 1360.00 |
| 4 | 1355.00 |
| 5 | 1350.00 |
| 6 | 1350.00 |
| 7 | 1360.00 |
| 8 | 1360.00 |
| 9 | 1360.00 |
| 10 | 1360.00 |
| 11 | 1360.00 |
| 12 | 1360.00 |

```
Summary for local incremental inflow data file:
===============================================

                    File name: ifw.dat
                    Data unit: acre-feet
              Number of nodes:    3
            Number of records:   12
          List of nodal names: POND_1     POND_2     POND_3


Summary for precipitation data file:
====================================

                    File name: rain.dat
                    Data unit: inches
              Number of nodes:    6
            Number of records:    1
           List of nodal name: POND_2       DEFAULT


Summary for water-surface evaporation coefficient data file:
============================================================

                    File name: pet2.dat
                    Data unit: inches per day
              Number of nodes:    6
            Number of records:   12
           List of nodal name: POND_3     POND_1       DEFAULT


Summary for target water-demand data file:
==========================================

                    File name: tws2.dat
                    Data unit: acre-feet
              Number of nodes:    2
            Number of records:   12
          List of nodal names: POND_3     PUMP_3


Summary for rule-curve elevation data file:
===========================================

                    File name: rc2.dat
                    Data unit: feet
              Number of nodes:    1
            Number of records:   12
          List of nodal names: POND_1


Summary for ground-water-level data file:
=========================================

                    File name: gwl.dat
                    Data unit: feet
              Number of nodes:    6
            Number of records:   12
          List of nodal names: POND_1     POND_2     POND_3     PUMP_1     PUMP_2
                               PUMP_3


Node name and type of basic network
===================================
             Node   Node         Node
           number   name         type
           ------   ------       ------
                1   POND_1       POND
                2   POND_2       POND
                3   POND_3       POND
                4   PUMP_1       GENERAL
                5   PUMP_2       GENERAL
                6   PUMP_3       GENERAL
```

Basic network

| Arc | From-node | To-node | Flow boundary Lower (cubic feet per second) | Flow boundary Upper (cubic feet per second) | Penalty coefficient | Type |
|---|---|---|---|---|---|---|
| 1 | SKSC | POND_1 | 0.00 | 803.69 | 100 | -201 |
| 2 | POND_1 | SKSC | 0.00 | 361.97 | 1000 | 201 |
| 3 | SKSC | POND_2 | 0.00 | 1383.78 | 100 | -201 |
| 4 | POND_2 | SKSC | 0.00 | 2116.07 | 1000 | 201 |
| 5 | SKSC | POND_3 | 0.00 | 1173.06 | 100 | -201 |
| 6 | POND_3 | SKSC | 0.00 | 2292.43 | 1000 | 201 |
| 7 | POND_3 | SKSC | 0.00 | 3295.78 | 2000 | 202 |
| 8 | POND_1 | STRM001004 | 0.00 | 16000.00 | 10 | 110 |
| 9 | STRM001004 | PUMP_1 | 0.00 | 16000.00 | 10 | 110 |
| 10 | POND_1 | STRM001005 | 0.00 | 16000.00 | 100 | 110 |
| 11 | STRM001005 | PUMP_2 | 0.00 | 16000.00 | 100 | 110 |
| 12 | POND_2 | STRM002005 | 0.00 | 16000.00 | 10 | 110 |
| 13 | STRM002005 | PUMP_2 | 0.00 | 16000.00 | 10 | 110 |
| 14 | POND_3 | STRM003006 | 0.00 | 16000.00 | 10 | 110 |
| 15 | STRM003006 | PUMP_3 | 0.00 | 16000.00 | 10 | 110 |
| 16 | PUMP_1 | STRM004003 | 10.00 | 16000.00 | 10 | 110 |
| 17 | STRM004003 | POND_3 | 10.00 | 16000.00 | 10 | 110 |
| 18 | PUMP_2 | STRM005003 | 10.00 | 16000.00 | 10 | 110 |
| 19 | STRM005003 | POND_3 | 10.00 | 16000.00 | 10 | 110 |
| 20 | PUMP_3 | SKSC | 10.00 | 16000.00 | 10 | 100 |

Pond-zoning penalty coefficients

| Name | Zone 1 | Zone 2 | Zone 3 |
|---|---|---|---|
| POND_1 | 100 | 1000 | |
| POND_2 | 100 | 1000 | |
| POND_3 | 100 | 1000 | 2000 |

Flow-zoning penalty coefficients

| From-node | To node | Normal flow zone | Lower flow zone | Upper flow zone |
|---|---|---|---|---|
| POND_1 | PUMP_1 | 10 | | |
| POND_1 | PUMP_2 | 100 | | |
| POND_2 | PUMP_2 | 10 | | |
| POND_3 | PUMP_3 | 10 | | |
| PUMP_1 | POND_3 | 10 | | |
| PUMP_2 | POND_3 | 10 | | |
| PUMP_3 | SKSC | 10 | | |

Nodal water budgets for whole simulation

| Node name | Node type | Initial storage (acre-feet) | Upstream inflow (acre-feet) | Local net inflow (acre-feet) | Evap-ration (acre-feet) | Rainfall (acre-feet) | Seepage (acre-feet) | Withdrawal (acre-feet) | Downstream release (acre-feet) | Final storage (acre-feet) |
|---|---|---|---|---|---|---|---|---|---|---|
| POND_1 | 1 | 48506.50 | 0.00 | 44233.36 | 9070.00 | 26.95 | 0.00 | 0.00 | 49592.00 | 34104.81 |
| POND_2 | 1 | 83856.91 | 0.02 | 33699.75 | 2006.36 | 1033.83 | -730.60 | 0.00 | 42006.13 | 75308.61 |
| POND_3 | 1 | 71284.00 | 98219.99 | 766768.44 | 43382.32 | 77.43 | -1911.17 | 30374.00 | 797618.75 | 66885.85 |
| PUMP_1 | 2 | 0.00 | 49056.46 | 0.00 | 0.00 | 0.00 | 0.00 | 7328.00 | 41728.46 | 0.00 |
| PUMP_2 | 2 | 0.00 | 41555.05 | 0.00 | 0.00 | 0.00 | 0.00 | 4039.00 | 37516.05 | 0.00 |
| PUMP_3 | 2 | 0.00 | 789537.75 | 0.00 | 0.00 | 0.00 | 0.00 | 30394.00 | 759143.75 | 0.00 |

Canal water budgets for whole simulation

| No. | From | To | Inflow (acre-feet) | Initial storage (acre-feet) | Canal seepage (acre-feet) | Surface evapo-ration (acre-feet) | Final storage (acre-feet) | Outflow (acre-feet) |
|---|---|---|---|---|---|---|---|---|
| 1 | POND_1 | PUMP_1 | 49592.00 | 1576.56 | 495.93 | 0.00 | 39.61 | 49056.46 |
| 2 | POND_1 | PUMP_2 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 3 | POND_2 | PUMP_2 | 42006.13 | 1338.02 | 420.07 | 0.00 | 31.01 | 41555.05 |
| 4 | POND_3 | PUMP_3 | 797618.75 | 25902.82 | 7976.17 | 0.00 | 104.85 | 789537.75 |
| 5 | PUMP_1 | POND_3 | 41728.46 | 1817.42 | -19448.18 | 13.97 | 63.69 | 61098.98 |
| 6 | PUMP_2 | POND_3 | 37516.05 | 1203.05 | 375.17 | 0.00 | 19.88 | 37121.00 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 7 | PUMP_3 | SKSC | 759143.75 | 0.00 | 0.00 | 0.00 | 0.00 759143.75 |

System water budgets
===================

| Budget | Pond | Canal | Total |
|---|---|---|---|
| Initial storage: | 203647.41 | 0.00 | 203647.41 |
| Total net inflow: | 844701.56 | 0.00 | 844701.56 |
| Runoff: | 1138.22 | 0.00 | 1138.22 |
| Evaporation: | 54458.68 | 13.97 | 54472.65 |
| Ground-water seepage: | -2641.76 | -10180.84 | -12822.60 |
| Water withdrawal: | 72135.00 | 0.00 | 72135.00 |
| Outflow: | 0.00 | 759143.75 | 759143.75 |
| Final storage: | 176299.28 | 259.03 | 176558.31 |
| | | In - Out = | 0.13 |

## Output File for Nodal Water Budgets (ndbt.out)

Water budgets for time step: 1
==============================
[--, not applicable]

| No. | Node name | Node type | Initial storage (acre-feet) | Upstream inflow (acre-feet) | Local Net inflow (acre-feet) | Evapo-ration (acre-feet) | Runoff (acre-feet) | Seepage (acre-feet) | Withdrawal (acre-feet) | Downstream release (acre-feet) | Final storage (acre-feet) | Final stage (feet) | Water depth (feet) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | POND_1 | 1 | 48506.50 | 0.00 | 1729.16 | 819.82 | 26.95 | 0.00 | 0.00 | 1275.66 | 48167.13 | 1273.89 | 34.89 |
| 2 | POND_2 | 1 | 83856.91 | 0.00 | 1494.33 | 157.25 | 1033.83 | -84.91 | 0.00 | 1015.08 | 85297.66 | 1350.74 | 42.74 |
| 3 | POND_3 | 1 | 71284.00 | 2813.19 | 36962.78 | 3533.21 | 77.43 | -169.59 | 2580.00 | 33909.79 | 71284.00 | 1039.00 | 29.00 |
| 4 | PUMP_1 | 2 | -- | 1222.37 | 0.00 | -- | 0.00 | -- | 619.00 | 603.37 | -- | -- | -- |
| 5 | PUMP_2 | 2 | -- | 972.68 | 0.00 | -- | 0.00 | -- | 343.00 | 629.68 | -- | -- | -- |
| 6 | PUMP_3 | 2 | -- | 32493.18 | 0.00 | -- | 0.00 | -- | 2600.00 | 29893.18 | -- | -- | -- |

Water budgets for time step: 2
==============================
[--, not applicable]

| No. | Node name | Node type | Initial storage (acre-feet) | Upstream inflow (acre-feet) | Local Net inflow (acre-feet) | Evapo-ration (acre-feet) | Runoff (acre-feet) | Seepage (acre-feet) | Withdrawal (acre-feet) | Downstream release (acre-feet) | Final storage (acre-feet) | Final stage (feet) | Water depth (feet) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | POND_1 | 1 | 48167.13 | 0.00 | 1593.61 | 817.88 | 0.00 | 0.00 | 0.00 | 12627.76 | 36315.10 | 1270.00 | 31.00 |
| 2 | POND_2 | 1 | 85297.66 | 0.00 | 1274.00 | 160.02 | 0.00 | -81.79 | 0.00 | 949.57 | 85543.86 | 1350.78 | 42.78 |
| 3 | POND_3 | 1 | 71284.00 | 13259.98 | 32579.50 | 3533.21 | 0.00 | -169.59 | 2330.00 | 40145.87 | 71284.00 | 1039.00 | 29.00 |
| 4 | PUMP_1 | 2 | -- | 12139.73 | 0.00 | -- | 0.00 | -- | 599.00 | 11540.73 | -- | -- | -- |
| 5 | PUMP_2 | 2 | -- | 941.33 | 0.00 | -- | 0.00 | -- | 332.00 | 609.33 | -- | -- | -- |
| 6 | PUMP_3 | 2 | -- | 39518.70 | 0.00 | -- | 0.00 | -- | 2330.00 | 37188.70 | -- | -- | -- |

Water budgets for time step: 3
==============================
[--, not applicable]

| No. | Node name | Node type | Initial storage (acre-feet) | Upstream inflow (acre-feet) | Local Net inflow (acre-feet) | Evapo-ration (acre-feet) | Runoff (acre-feet) | Seepage (acre-feet) | Withdrawal (acre-feet) | Downstream release (acre-feet) | Final storage (acre-feet) | Final stage (feet) | Water depth (feet) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | POND_1 | 1 | 36315.10 | 0.00 | 13222.35 | 717.41 | 0.00 | 0.00 | 0.00 | 25443.95 | 23376.10 | 1265.00 | 26.00 |
| 2 | POND_2 | 1 | 85543.86 | 0.01 | 9886.98 | 160.49 | 0.00 | -81.24 | 0.00 | 962.47 | 94389.13 | 1352.15 | 44.15 |
| 3 | POND_3 | 1 | 71284.00 | 25655.43 | 66895.89 | 3533.21 | 0.00 | -169.59 | 2580.00 | 86607.71 | 71284.00 | 1039.00 | 29.00 |
| 4 | PUMP_1 | 2 | -- | 24773.01 | 0.00 | -- | 0.00 | -- | 619.00 | 24154.01 | -- | -- | -- |
| 5 | PUMP_2 | 2 | -- | 952.47 | 0.00 | -- | 0.00 | -- | 343.00 | 609.47 | -- | -- | -- |
| 6 | PUMP_3 | 2 | -- | 84259.55 | 0.00 | -- | 0.00 | -- | 2580.00 | 81679.55 | -- | -- | -- |

Water budgets for time step: 4
==============================
[--, not applicable]

| No. | Node name | Node type | Initial storage (acre-feet) | Upstream inflow (acre-feet) | Local Net inflow (acre-feet) | Evapo-ration (acre-feet) | Runoff (acre-feet) | Seepage (acre-feet) | Withdrawal (acre-feet) | Downstream release (acre-feet) | Final storage (acre-feet) | Final stage (feet) | Water depth (feet) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | POND_1 | 1 | 23376.10 | 0.00 | 19412.21 | 581.53 | 0.00 | 0.00 | 0.00 | 443.09 | 41763.69 | 1271.85 | 32.85 |
| 2 | POND_2 | 1 | 94389.13 | 0.00 | 13013.84 | 168.12 | 0.00 | -57.42 | 0.00 | 962.09 | 106330.19 | 1353.86 | 45.86 |
| 3 | POND_3 | 1 | 71284.00 | 3598.11 | 171310.34 | 3533.21 | 0.00 | -169.59 | 2496.00 | 154318.00 | 86014.83 | 1040.52 | 30.52 |
| 4 | PUMP_1 | 2 | -- | 1222.37 | 0.00 | -- | 0.00 | -- | 619.00 | 603.37 | -- | -- | -- |
| 5 | PUMP_2 | 2 | -- | 952.47 | 0.00 | -- | 0.00 | -- | 343.00 | 609.47 | -- | -- | -- |
| 6 | PUMP_3 | 2 | -- | 150585.31 | 0.00 | -- | 0.00 | -- | 2496.00 | 148089.31 | -- | -- | -- |

Water budgets for time step: 5

[--, not applicable]

| No. | Node name | Node type | Initial storage (acre-feet) | Upstream inflow (acre-feet) | Local Net inflow (acre-feet) | Evaporation (acre-feet) | Runoff (acre-feet) | Seepage (acre-feet) | Withdrawal (acre-feet) | Downstream release (acre-feet) | Final storage (acre-feet) | Final stage (feet) | Water depth (feet) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | POND_1 | 1 | 41763.69 | 0.00 | 6358.12 | 766.47 | 0.00 | 0.00 | 0.00 | 1219.14 | 46136.20 | 1273.25 | 34.25 |
| 2 | POND_2 | 1 | 106330.19 | 0.00 | 7925.43 | 179.65 | 0.00 | -24.36 | 0.00 | 927.65 | 113172.68 | 1354.79 | 46.79 |
| 3 | POND_3 | 1 | 86014.83 | 2892.46 | 203225.77 | 3821.42 | 0.00 | -136.71 | 2580.00 | 173304.08 | 112564.27 | 1042.99 | 32.99 |
| 4 | PUMP_1 | 2 | -- | 1202.37 | 0.00 | -- | 0.00 | -- | 599.00 | 603.37 | -- | -- | -- |
| 5 | PUMP_2 | 2 | -- | 919.47 | 0.00 | -- | 0.00 | -- | 310.00 | 609.47 | -- | -- | -- |
| 6 | PUMP_3 | 2 | -- | 170911.70 | 0.00 | -- | 0.00 | -- | 2580.00 | 168331.70 | -- | -- | -- |

Water budgets for time step: 6

[--, not applicable]

| No. | Node name | Node type | Initial storage (acre-feet) | Upstream inflow (acre-feet) | Local Net inflow (acre-feet) | Evaporation (acre-feet) | Runoff (acre-feet) | Seepage (acre-feet) | Withdrawal (acre-feet) | Downstream release (acre-feet) | Final storage (acre-feet) | Final stage (feet) | Water depth (feet) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | POND_1 | 1 | 46136.20 | 0.00 | 723.75 | 806.29 | 0.00 | 0.00 | 0.00 | 1235.36 | 44818.30 | 1272.84 | 33.84 |
| 2 | POND_2 | 1 | 113172.68 | 0.00 | 2390.50 | 192.39 | 0.00 | -4.76 | 0.00 | 963.20 | 114412.36 | 1354.96 | 46.96 |
| 3 | POND_3 | 1 | 112564.27 | 2874.73 | 118321.29 | 4452.65 | 0.00 | -71.44 | 2496.00 | 155599.08 | 71284.00 | 1039.00 | 29.00 |
| 4 | PUMP_1 | 2 | -- | 1222.37 | 0.00 | -- | 0.00 | -- | 619.00 | 603.37 | -- | -- | -- |
| 5 | PUMP_2 | 2 | -- | 952.47 | 0.00 | -- | 0.00 | -- | 343.00 | 609.47 | -- | -- | -- |
| 6 | PUMP_3 | 2 | -- | 154588.78 | 0.00 | -- | 0.00 | -- | 2496.00 | 152092.78 | -- | -- | -- |

Water budgets for time step: 7

[--, not applicable]

| No. | Node name | Node type | Initial storage (acre-feet) | Upstream inflow (acre-feet) | Local Net inflow (acre-feet) | Evaporation (acre-feet) | Runoff (acre-feet) | Seepage (acre-feet) | Withdrawal (acre-feet) | Downstream release (acre-feet) | Final storage (acre-feet) | Final stage (feet) | Water depth (feet) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | POND_1 | 1 | 44818.30 | 0.00 | -407.64 | 796.43 | 0.00 | 0.00 | 0.00 | 1213.84 | 42400.39 | 1272.06 | 33.06 |
| 2 | POND_2 | 1 | 114412.36 | 0.00 | 64.59 | 195.02 | 0.00 | -0.94 | 0.00 | 950.60 | 113332.27 | 1354.81 | 46.81 |
| 3 | POND_3 | 1 | 71284.00 | 2874.27 | 67050.84 | 3533.21 | 0.00 | -169.59 | 2580.00 | 63981.50 | 71284.00 | 1039.00 | 29.00 |
| 4 | PUMP_1 | 2 | -- | 1202.37 | 0.00 | -- | 0.00 | -- | 599.00 | 603.37 | -- | -- | -- |
| 5 | PUMP_2 | 2 | -- | 941.47 | 0.00 | -- | 0.00 | -- | 332.00 | 609.47 | -- | -- | -- |
| 6 | PUMP_3 | 2 | -- | 66266.83 | 0.00 | -- | 0.00 | -- | 2580.00 | 63686.83 | -- | -- | -- |

Water budgets for time step: 8

[--, not applicable]

| No. | Node name | Node type | Initial storage (acre-feet) | Upstream inflow (acre-feet) | Local Net inflow (acre-feet) | Evaporation (acre-feet) | Runoff (acre-feet) | Seepage (acre-feet) | Withdrawal (acre-feet) | Downstream release (acre-feet) | Final storage (acre-feet) | Final stage (feet) | Water depth (feet) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | POND_1 | 1 | 42400.39 | 0.00 | 2926.71 | 771.11 | 0.00 | 0.00 | 0.00 | 1235.39 | 43320.60 | 1272.35 | 33.35 |
| 2 | POND_2 | 1 | 113332.27 | 0.00 | 122.61 | 192.73 | 0.00 | -4.28 | 0.00 | 32402.33 | 80864.10 | 1350.00 | 42.00 |
| 3 | POND_3 | 1 | 71284.00 | 31742.23 | 31301.72 | 3533.21 | 0.00 | -169.59 | 2580.00 | 57100.34 | 71284.00 | 1039.00 | 29.00 |
| 4 | PUMP_1 | 2 | -- | 1222.37 | 0.00 | -- | 0.00 | -- | 619.00 | 603.37 | -- | -- | -- |
| 5 | PUMP_2 | 2 | -- | 31078.96 | 0.00 | -- | 0.00 | -- | 343.00 | 30735.96 | -- | -- | -- |
| 6 | PUMP_3 | 2 | -- | 56822.88 | 0.00 | -- | 0.00 | -- | 2580.00 | 54242.88 | -- | -- | -- |

Water budgets for time step: 9

[--, not applicable]

| No. | Node name | Node type | Initial storage (acre-feet) | Upstream inflow (acre-feet) | Local Net inflow (acre-feet) | Evaporation (acre-feet) | Runoff (acre-feet) | Seepage (acre-feet) | Withdrawal (acre-feet) | Downstream release (acre-feet) | Final storage (acre-feet) | Final stage (feet) | Water depth (feet) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | POND_1 | 1 | 43320.60 | 0.00 | -840.34 | 780.74 | 0.00 | 0.00 | 0.00 | 1213.84 | 40485.67 | 1271.42 | 32.42 |
| 2 | POND_2 | 1 | 80864.10 | 0.00 | -1525.57 | 151.49 | 0.00 | -90.89 | 0.00 | 0.00 | 79277.94 | 1349.74 | 41.74 |
| 3 | POND_3 | 1 | 71284.00 | 3866.78 | 22908.99 | 3533.21 | 0.00 | -169.59 | 2496.00 | 20916.16 | 71284.00 | 1039.00 | 29.00 |
| 4 | PUMP_1 | 2 | -- | 1202.37 | 0.00 | -- | 0.00 | -- | 599.00 | 603.37 | -- | -- | -- |
| 5 | PUMP_2 | 2 | -- | 1003.94 | 0.00 | -- | 0.00 | -- | 332.00 | 671.94 | -- | -- | -- |
| 6 | PUMP_3 | 2 | -- | 21864.21 | 0.00 | -- | 0.00 | -- | 2496.00 | 19368.21 | -- | -- | -- |

Water budgets for time step: 10

[--, not applicable]

| No. | Node name | Node type | Initial storage (acre-feet) | Upstream inflow (acre-feet) | Local Net inflow (acre-feet) | Evaporation (acre-feet) | Runoff (acre-feet) | Seepage (acre-feet) | Withdrawal (acre-feet) | Downstream release (acre-feet) | Final storage (acre-feet) | Final stage (feet) | Water depth (feet) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | POND_1 | 1 | 40485.67 | 0.00 | -435.46 | 758.91 | 0.00 | 0.00 | 0.00 | 1235.39 | 38055.91 | 1270.60 | 31.60 |
| 2 | POND_2 | 1 | 79277.94 | 0.01 | -1355.91 | 150.69 | 0.00 | -95.09 | 0.00 | 960.76 | 76905.67 | 1349.35 | 41.35 |
| 3 | POND_3 | 1 | 71284.00 | 2894.30 | 8527.78 | 3533.21 | 0.00 | -169.59 | 2580.00 | 5478.47 | 71284.00 | 1039.00 | 29.00 |
| 4 | PUMP_1 | 2 | -- | 1222.37 | 0.00 | -- | 0.00 | -- | 619.00 | 603.37 | -- | -- | -- |
| 5 | PUMP_2 | 2 | -- | 946.37 | 0.00 | -- | 0.00 | -- | 343.00 | 603.37 | -- | -- | -- |
| 6 | PUMP_3 | 2 | -- | 5943.89 | 0.00 | -- | 0.00 | -- | 2580.00 | 3363.89 | -- | -- | -- |

Water budgets for time step: 11

[--, not applicable]

| No. | Node name | Node type | Initial storage (acre-feet) | Upstream inflow (acre-feet) | Local Net inflow (acre-feet) | Evapo-ration (acre-feet) | Runoff (acre-feet) | Seepage (acre-feet) | Withdrawal (acre-feet) | Downstream release (acre-feet) | Final storage (acre-feet) | Final stage (feet) | Water depth (feet) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | POND_1 | 1 | 38055.91 | 0.00 | -126.75 | 737.82 | 0.00 | 0.00 | 0.00 | 1234.71 | 35956.62 | 1269.87 | 30.87 |
| 2 | POND_2 | 1 | 76905.67 | -0.01 | 94.50 | 149.49 | 0.00 | -101.28 | 0.00 | 961.74 | 75990.21 | 1349.20 | 41.20 |
| 3 | POND_3 | 1 | 71284.00 | 2874.25 | 66.27 | 3533.21 | 0.00 | -169.59 | 2496.00 | 3038.96 | 65325.95 | 1038.32 | 28.32 |
| 4 | PUMP_1 | 2 | -- | 1222.37 | 0.00 | -- | 0.00 | -- | 619.00 | 603.37 | -- | -- | -- |
| 5 | PUMP_2 | 2 | -- | 951.97 | 0.00 | -- | 0.00 | -- | 343.00 | 608.97 | -- | -- | -- |
| 6 | PUMP_3 | 2 | -- | 3099.37 | 0.00 | -- | 0.00 | -- | 2496.00 | 603.37 | -- | -- | -- |

Water budgets for time step: 12

[--, not applicable]

| No. | Node name | Node type | Initial storage (acre-feet) | Upstream inflow (acre-feet) | Local Net inflow (acre-feet) | Evapo-ration (acre-feet) | Runoff (acre-feet) | Seepage (acre-feet) | Withdrawal (acre-feet) | Downstream release (acre-feet) | Final storage (acre-feet) | Final stage (feet) | Water depth (feet) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | POND_1 | 1 | 35956.62 | 0.00 | 77.64 | 715.59 | 0.00 | 0.00 | 0.00 | 1213.86 | 34104.81 | 1269.21 | 30.21 |
| 2 | POND_2 | 1 | 75990.21 | 0.00 | 314.43 | 149.03 | 0.00 | -103.64 | 0.00 | 950.63 | 75308.61 | 1349.09 | 41.09 |
| 3 | POND_3 | 1 | 65325.95 | 2874.26 | 7617.22 | 3309.39 | 0.00 | -176.67 | 2580.00 | 3218.86 | 66885.85 | 1038.50 | 28.50 |
| 4 | PUMP_1 | 2 | -- | 1202.37 | 0.00 | -- | 0.00 | -- | 599.00 | 603.37 | -- | -- | -- |
| 5 | PUMP_2 | 2 | -- | 941.47 | 0.00 | -- | 0.00 | -- | 332.00 | 609.47 | -- | -- | -- |
| 6 | PUMP_3 | 2 | -- | 3183.37 | 0.00 | -- | 0.00 | -- | 2580.00 | 603.37 | -- | -- | -- |

## Output File for Canal Water Budgets (arbt.out)

Canal water budgets for time step: 1

| No. | From | To | Inflow (acre-feet) | Initial storage (acre-feet) | Canal seepage (acre-feet) | Surface evapo-ration (acre-feet) | Final storage (acre-feet) | Outflow (acre-feet) | Outflow (cubic feet per second) |
|---|---|---|---|---|---|---|---|---|---|
| 1 | POND_1 | PUMP_1 | 1275.66 | 0.00 | 12.76 | 0.00 | 40.53 | 1222.37 | 20.26 |
| 2 | POND_1 | PUMP_2 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 3 | POND_2 | PUMP_2 | 1015.08 | 0.00 | 10.15 | 0.00 | 32.25 | 972.68 | 16.12 |
| 4 | POND_3 | PUMP_3 | 33909.79 | 0.00 | 339.10 | 0.00 | 1077.51 | 32493.18 | 538.53 |
| 5 | PUMP_1 | POND_3 | 603.37 | 0.00 | -1669.70 | 1.16 | 62.09 | 2209.82 | 36.62 |
| 6 | PUMP_2 | POND_3 | 629.68 | 0.00 | 6.30 | 0.00 | 20.01 | 603.37 | 10.00 |
| 7 | PUMP_3 | SKSC | 29893.18 | 0.00 | 0.00 | 0.00 | 0.00 | 29893.18 | 495.44 |

Canal water budgets for time step: 2

| No. | From | To | Inflow (acre-feet) | Initial storage (acre-feet) | Canal seepage (acre-feet) | Surface evapo-ration (acre-feet) | Final storage (acre-feet) | Outflow (acre-feet) | Outflow (cubic feet per second) |
|---|---|---|---|---|---|---|---|---|---|
| 1 | POND_1 | PUMP_1 | 12627.76 | 40.53 | 126.28 | 0.00 | 402.28 | 12139.73 | 201.20 |
| 2 | POND_1 | PUMP_2 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 3 | POND_2 | PUMP_2 | 949.57 | 32.25 | 9.50 | 0.00 | 31.00 | 941.33 | 15.60 |
| 4 | POND_3 | PUMP_3 | 40145.87 | 1077.51 | 401.46 | 0.00 | 1303.22 | 39518.70 | 654.96 |
| 5 | PUMP_1 | POND_3 | 11540.73 | 62.09 | -1463.68 | 1.16 | 408.72 | 12656.61 | 209.76 |
| 6 | PUMP_2 | POND_3 | 609.33 | 20.01 | 6.09 | 0.00 | 19.87 | 603.37 | 10.00 |
| 7 | PUMP_3 | SKSC | 37188.70 | 0.00 | 0.00 | 0.00 | 0.00 | 37188.70 | 616.35 |

Canal water budgets for time step: 3

| No. | From | To | Inflow (acre-feet) | Initial storage (acre-feet) | Canal seepage (acre-feet) | Surface evapo-ration (acre-feet) | Final storage (acre-feet) | Outflow (acre-feet) | Outflow (cubic feet per second) |
|---|---|---|---|---|---|---|---|---|---|
| 1 | POND_1 | PUMP_1 | 25443.95 | 402.28 | 254.44 | 0.00 | 818.78 | 24773.01 | 410.58 |
| 2 | POND_1 | PUMP_2 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 3 | POND_2 | PUMP_2 | 962.47 | 31.00 | 9.63 | 0.00 | 31.38 | 952.47 | 15.79 |
| 4 | POND_3 | PUMP_3 | 86607.71 | 1303.22 | 866.08 | 0.00 | 2785.30 | 84259.55 | 1396.48 |
| 5 | PUMP_1 | POND_3 | 24154.01 | 408.72 | -1308.12 | 1.16 | 817.63 | 25052.06 | 415.20 |
| 6 | PUMP_2 | POND_3 | 609.47 | 19.87 | 6.09 | 0.00 | 19.88 | 603.37 | 10.00 |
| 7 | PUMP_3 | SKSC | 81679.55 | 0.00 | 0.00 | 0.00 | 0.00 | 81679.55 | 1353.72 |

Canal water budgets for time step: 4

| No. | From | To | Inflow (acre-feet) | Initial storage (acre-feet) | Canal seepage (acre-feet) | Surface evaporation (acre-feet) | Final storage (acre-feet) | Outflow (acre-feet) | Outflow (cubic feet per second) |
|---|---|---|---|---|---|---|---|---|---|
| 1 | POND_1 | PUMP_1 | 443.09 | 818.78 | 4.43 | 0.00 | 35.06 | 1222.37 | 20.26 |
| 2 | POND_1 | PUMP_2 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 3 | POND_2 | PUMP_2 | 962.09 | 31.38 | 9.62 | 0.00 | 31.37 | 952.47 | 15.79 |
| 4 | POND_3 | PUMP_3 | 154318.00 | 2785.30 | 1543.18 | 0.00 | 4974.80 | 150585.31 | 2495.73 |
| 5 | PUMP_1 | POND_3 | 603.37 | 817.63 | -1657.57 | 1.16 | 82.67 | 2994.74 | 49.63 |
| 6 | PUMP_2 | POND_3 | 609.47 | 19.88 | 6.09 | 0.00 | 19.88 | 603.37 | 10.00 |
| 7 | PUMP_3 | SKSC | 148089.31 | 0.00 | 0.00 | 0.00 | 0.00 | 148089.31 | 2454.36 |

Canal water budgets for time step: 5

| No. | From | To | Inflow (acre-feet) | Initial storage (acre-feet) | Canal seepage (acre-feet) | Surface evaporation (acre-feet) | Final storage (acre-feet) | Outflow (acre-feet) | Outflow (cubic feet per second) |
|---|---|---|---|---|---|---|---|---|---|
| 1 | POND_1 | PUMP_1 | 1219.14 | 35.06 | 12.19 | 0.00 | 39.64 | 1202.37 | 19.93 |
| 2 | POND_1 | PUMP_2 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 3 | POND_2 | PUMP_2 | 927.65 | 31.37 | 9.28 | 0.00 | 30.28 | 919.47 | 15.24 |
| 4 | POND_3 | PUMP_3 | 173304.08 | 4974.80 | 1733.04 | 0.00 | 5634.14 | 170911.70 | 2832.61 |
| 5 | PUMP_1 | POND_3 | 603.37 | 82.67 | -1668.38 | 1.16 | 64.17 | 2289.09 | 37.94 |
| 6 | PUMP_2 | POND_3 | 609.47 | 19.88 | 6.09 | 0.00 | 19.88 | 603.37 | 10.00 |
| 7 | PUMP_3 | SKSC | 168331.70 | 0.00 | 0.00 | 0.00 | 0.00 | 168331.70 | 2789.85 |

Canal water budgets for time step: 6

| No. | From | To | Inflow (acre-feet) | Initial storage (acre-feet) | Canal seepage (acre-feet) | Surface evaporation (acre-feet) | Final storage (acre-feet) | Outflow (acre-feet) | Outflow (cubic feet per second) |
|---|---|---|---|---|---|---|---|---|---|
| 1 | POND_1 | PUMP_1 | 1235.36 | 39.64 | 12.35 | 0.00 | 40.27 | 1222.37 | 20.26 |
| 2 | POND_1 | PUMP_2 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 3 | POND_2 | PUMP_2 | 963.20 | 30.28 | 9.63 | 0.00 | 31.38 | 952.47 | 15.79 |
| 4 | POND_3 | PUMP_3 | 155599.08 | 5634.14 | 1555.99 | 0.00 | 5088.44 | 154588.78 | 2562.08 |
| 5 | PUMP_1 | POND_3 | 603.37 | 64.17 | -1668.68 | 1.16 | 63.70 | 2271.35 | 37.64 |
| 6 | PUMP_2 | POND_3 | 609.47 | 19.88 | 6.09 | 0.00 | 19.88 | 603.37 | 10.00 |
| 7 | PUMP_3 | SKSC | 152092.78 | 0.00 | 0.00 | 0.00 | 0.00 | 152092.78 | 2520.71 |

Canal water budgets for time step: 7

| No. | From | To | Inflow (acre-feet) | Initial storage (acre-feet) | Canal seepage (acre-feet) | Surface evaporation (acre-feet) | Final storage (acre-feet) | Outflow (acre-feet) | Outflow (cubic feet per second) |
|---|---|---|---|---|---|---|---|---|---|
| 1 | POND_1 | PUMP_1 | 1213.84 | 40.27 | 12.14 | 0.00 | 39.60 | 1202.37 | 19.93 |
| 2 | POND_1 | PUMP_2 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 3 | POND_2 | PUMP_2 | 950.60 | 31.38 | 9.51 | 0.00 | 31.01 | 941.47 | 15.60 |
| 4 | POND_3 | PUMP_3 | 63981.50 | 5088.44 | 639.82 | 0.00 | 2163.30 | 66266.83 | 1098.28 |
| 5 | PUMP_1 | POND_3 | 603.37 | 63.70 | -1668.68 | 1.16 | 63.69 | 2270.90 | 37.64 |
| 6 | PUMP_2 | POND_3 | 609.47 | 19.88 | 6.09 | 0.00 | 19.88 | 603.37 | 10.00 |
| 7 | PUMP_3 | SKSC | 63686.83 | 0.00 | 0.00 | 0.00 | 0.00 | 63686.83 | 1055.52 |

Canal water budgets for time step: 8

| No. | From | To | Inflow (acre-feet) | Initial storage (acre-feet) | Canal seepage (acre-feet) | Surface evaporation (acre-feet) | Final storage (acre-feet) | Outflow (acre-feet) | Outflow (cubic feet per second) |
|---|---|---|---|---|---|---|---|---|---|
| 1 | POND_1 | PUMP_1 | 1235.39 | 39.60 | 12.35 | 0.00 | 40.27 | 1222.37 | 20.26 |
| 2 | POND_1 | PUMP_2 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 3 | POND_2 | PUMP_2 | 32402.33 | 31.01 | 324.02 | 0.00 | 1030.36 | 31078.96 | 515.09 |
| 4 | POND_3 | PUMP_3 | 57100.34 | 2163.30 | 571.00 | 0.00 | 1869.75 | 56822.88 | 941.76 |
| 5 | PUMP_1 | POND_3 | 603.37 | 63.69 | -1668.68 | 1.16 | 63.69 | 2270.88 | 37.64 |
| 6 | PUMP_2 | POND_3 | 30735.96 | 19.88 | 307.36 | 0.00 | 977.13 | 29471.35 | 488.44 |
| 7 | PUMP_3 | SKSC | 54242.88 | 0.00 | 0.00 | 0.00 | 0.00 | 54242.88 | 899.00 |

Canal water budgets for time step: 9

| No. | From | To | Inflow (acre-feet) | Initial storage (acre-feet) | Canal seepage (acre-feet) | Surface evaporation (acre-feet) | Final storage (acre-feet) | Outflow (acre-feet) | Outflow (cubic feet per second) |
|---|---|---|---|---|---|---|---|---|---|
| 1 | POND_1 | PUMP_1 | 1213.84 | 40.27 | 12.14 | 0.00 | 39.60 | 1202.37 | 19.93 |
| 2 | POND_1 | PUMP_2 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 3 | POND_2 | PUMP_2 | 0.00 | 1030.36 | 0.00 | 0.00 | 26.42 | 1003.94 | 16.64 |
| 4 | POND_3 | PUMP_3 | 20916.16 | 1869.75 | 209.18 | 0.00 | 712.52 | 21864.21 | 362.37 |
| 5 | PUMP_1 | POND_3 | 603.37 | 63.69 | -1668.68 | 1.16 | 63.69 | 2270.88 | 37.64 |
| 6 | PUMP_2 | POND_3 | 671.94 | 977.13 | 6.73 | 0.00 | 46.44 | 1595.90 | 26.45 |
| 7 | PUMP_3 | SKSC | 19368.21 | 0.00 | 0.00 | 0.00 | 0.00 | 19368.21 | 321.00 |

Canal water budgets for time step: 10

| No. | From | To | Inflow (acre-feet) | Initial storage (acre-feet) | Canal seepage (acre-feet) | Surface evaporation (acre-feet) | Final storage (acre-feet) | Outflow (acre-feet) | Outflow (cubic feet per second) |
|---|---|---|---|---|---|---|---|---|---|
| 1 | POND_1 | PUMP_1 | 1235.39 | 39.60 | 12.35 | 0.00 | 40.27 | 1222.37 | 20.26 |
| 2 | POND_1 | PUMP_2 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 3 | POND_2 | PUMP_2 | 960.76 | 26.42 | 9.61 | 0.00 | 31.20 | 946.37 | 15.68 |
| 4 | POND_3 | PUMP_3 | 5478.47 | 712.52 | 54.78 | 0.00 | 192.32 | 5943.89 | 98.51 |
| 5 | PUMP_1 | POND_3 | 603.37 | 63.69 | -1668.68 | 1.16 | 63.69 | 2270.88 | 37.64 |
| 6 | PUMP_2 | POND_3 | 603.37 | 46.44 | 6.03 | 0.00 | 20.36 | 623.41 | 10.33 |
| 7 | PUMP_3 | SKSC | 3363.89 | 0.00 | 0.00 | 0.00 | 0.00 | 3363.89 | 55.75 |

Canal water budgets for time step: 11

| No. | From | To | Inflow (acre-feet) | Initial storage (acre-feet) | Canal seepage (acre-feet) | Surface evaporation (acre-feet) | Final storage (acre-feet) | Outflow (acre-feet) | Outflow (cubic feet per second) |
|---|---|---|---|---|---|---|---|---|---|
| 1 | POND_1 | PUMP_1 | 1234.71 | 40.27 | 12.35 | 0.00 | 40.26 | 1222.37 | 20.26 |
| 2 | POND_1 | PUMP_2 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 3 | POND_2 | PUMP_2 | 961.74 | 31.20 | 9.62 | 0.00 | 31.36 | 951.97 | 15.78 |
| 4 | POND_3 | PUMP_3 | 3038.96 | 192.32 | 30.41 | 0.00 | 101.50 | 3099.37 | 51.37 |
| 5 | PUMP_1 | POND_3 | 603.37 | 63.69 | -1668.68 | 1.16 | 63.69 | 2270.88 | 37.64 |
| 6 | PUMP_2 | POND_3 | 608.97 | 20.36 | 6.09 | 0.00 | 19.87 | 603.37 | 10.00 |
| 7 | PUMP_3 | SKSC | 603.37 | 0.00 | 0.00 | 0.00 | 0.00 | 603.37 | 10.00 |

Canal water budgets for time step: 12

| No. | From | To | Inflow (acre-feet) | Initial storage (acre-feet) | Canal seepage (acre-feet) | Surface evaporation (acre-feet) | Final storage (acre-feet) | Outflow (acre-feet) | Outflow (cubic feet per second) |
|---|---|---|---|---|---|---|---|---|---|
| 1 | POND_1 | PUMP_1 | 1213.86 | 40.26 | 12.15 | 0.00 | 39.61 | 1202.37 | 19.93 |
| 2 | POND_1 | PUMP_2 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 3 | POND_2 | PUMP_2 | 950.63 | 31.36 | 9.51 | 0.00 | 31.01 | 941.47 | 15.60 |
| 4 | POND_3 | PUMP_3 | 3218.86 | 101.50 | 32.14 | 0.00 | 104.85 | 3183.37 | 52.76 |
| 5 | PUMP_1 | POND_3 | 603.37 | 63.69 | -1668.68 | 1.16 | 63.69 | 2270.88 | 37.64 |
| 6 | PUMP_2 | POND_3 | 609.47 | 19.87 | 6.09 | 0.00 | 19.88 | 603.37 | 10.00 |
| 7 | PUMP_3 | SKSC | 603.37 | 0.00 | 0.00 | 0.00 | 0.00 | 603.37 | 10.00 |

## Output File for Operation of Hydraulic Structures (hydr.out)

Parameters for hydraulic structures: 1

[-999.99, not flow under gate]

| Structure name | Upstream node name | Downstream node name | Structure type | Base elevation (feet) | Weir length (feet) | Discharge (cubic feet per second) | Upstream Water elevation (feet) | Gate-opening or weir height (feet) |
|---|---|---|---|---|---|---|---|---|
| Gate-1 | Pond_1 | Pump_1 | Gate spillway | 1240.00 | 2.00 | 21.14 | 1274.00 | 0.35 |
| Weir-2 | Pond_2 | Pump_2 | Sharp-crested weir | 1340.00 | 2.00 | 16.82 | 1350.50 | 8.65 |
| Gate-3 | Pond_3 | Pump_3 | Sluice gate | 1020.00 | 10.00 | 562.00 | 1039.00 | 2.90 |

Parameters for hydraulic structures: 2

[-999.99, not flow under gate]

| Structure name | Upstream node name | Downstream node name | Structure type | Base elevation (feet) | Weir length (feet) | Discharge (cubic feet per second) | Upstream Water elevation (feet) | Gate-opening or weir height (feet) |
|---|---|---|---|---|---|---|---|---|
| Gate-1 | Pond_1 | Pump_1 | Gate spillway | 1240.00 | 2.00 | 209.29 | 1273.89 | 3.55 |
| Weir-2 | Pond_2 | Pump_2 | Sharp-crested weir | 1340.00 | 2.00 | 15.74 | 1350.74 | 8.97 |
| Gate-3 | Pond_3 | Pump_3 | Sluice gate | 1020.00 | 10.00 | 665.36 | 1039.00 | 3.48 |

Parameters for hydraulic structures: 3

[-999.99, not flow under gate]

| Structure name | Upstream node name | Downstream node name | Structure type | Base elevation (feet) | Weir length (feet) | Discharge (cubic feet per second) | Upstream Water elevation (feet) | Gate-opening or weir height (feet) |
|---|---|---|---|---|---|---|---|---|
| Gate-1 | Pond_1 | Pump_1 | Gate spillway | 1240.00 | 2.00 | 421.70 | 1270.00 | 7.99 |
| Weir-2 | Pond_2 | Pump_2 | Sharp-crested weir | 1340.00 | 2.00 | 15.95 | 1350.78 | 9.00 |
| Gate-3 | Pond_3 | Pump_3 | Sluice gate | 1020.00 | 10.00 | 1435.40 | 1039.00 | 8.07 |

Parameters for hydraulic structures: 4

[-999.99, not flow under gate]

| Structure name | Upstream node name | Downstream node name | Structure type | Base elevation (feet) | Weir length (feet) | Discharge (cubic feet per second) | Upstream Water elevation (feet) | Gate-opening or weir height (feet) |
|---|---|---|---|---|---|---|---|---|
| Gate-1 | Pond_1 | Pump_1 | Gate spillway | 1240.00 | 2.00 | 7.34 | 1265.00 | 0.14 |
| Weir-2 | Pond_2 | Pump_2 | Sharp-crested weir | 1340.00 | 2.00 | 15.95 | 1352.15 | 10.37 |
| Gate-3 | Pond_3 | Pump_3 | Sluice gate | 1020.00 | 10.00 | 2557.59 | 1039.00 | -999.99 |

Parameters for hydraulic structures: 5

[-999.99, not flow under gate]

| Structure name | Upstream node name | Downstream node name | Structure type | Base elevation (feet) | Weir length (feet) | Discharge (cubic feet per second) | Upstream Water elevation (feet) | Gate-opening or weir height (feet) |
|---|---|---|---|---|---|---|---|---|
| Gate-1 | Pond_1 | Pump_1 | Gate spillway | 1240.00 | 2.00 | 20.21 | 1271.85 | 0.34 |
| Weir-2 | Pond_2 | Pump_2 | Sharp-crested weir | 1340.00 | 2.00 | 15.37 | 1353.87 | 12.12 |
| Gate-3 | Pond_3 | Pump_3 | Sluice gate | 1020.00 | 10.00 | 2872.26 | 1040.53 | -999.99 |

Parameters for hydraulic structures: 6

[-999.99, not flow under gate]

| Structure name | Upstream node name | Downstream node name | Structure type | Base elevation (feet) | Weir length (feet) | Discharge (cubic feet per second) | Upstream Water elevation (feet) | Gate-opening or weir height (feet) |
|---|---|---|---|---|---|---|---|---|
| Gate-1 | Pond_1 | Pump_1 | Gate spillway | 1240.00 | 2.00 | 20.47 | 1273.26 | 0.34 |
| Weir-2 | Pond_2 | Pump_2 | Sharp-crested weir | 1340.00 | 2.00 | 15.96 | 1354.79 | 13.00 |
| Gate-3 | Pond_3 | Pump_3 | Sluice gate | 1020.00 | 10.00 | 2578.83 | 1042.99 | 13.84 |

Parameters for hydraulic structures: 7

[-999.99, not flow under gate]

| Structure name | Upstream node name | Downstream node name | Structure type | Base elevation (feet) | Weir length (feet) | Discharge (cubic feet per second) | Upstream Water elevation (feet) | Gate-opening or weir height (feet) |
|---|---|---|---|---|---|---|---|---|
| Gate-1 | Pond_1 | Pump_1 | Gate spillway | 1240.00 | 2.00 | 20.12 | 1272.84 | 0.34 |
| Weir-2 | Pond_2 | Pump_2 | Sharp-crested weir | 1340.00 | 2.00 | 15.75 | 1354.96 | 13.18 |
| Gate-3 | Pond_3 | Pump_3 | Sluice gate | 1020.00 | 10.00 | 1060.40 | 1039.00 | 5.76 |

Parameters for hydraulic structures: 8

[-999.99, not flow under gate]

| Structure name | Upstream node name | Downstream node name | Structure type | Base elevation (feet) | Weir length (feet) | Discharge (cubic feet per second) | Upstream Water elevation (feet) | Gate-opening or weir height (feet) |
|---|---|---|---|---|---|---|---|---|
| Gate-1 | Pond_1 | Pump_1 | Gate spillway | 1240.00 | 2.00 | 20.47 | 1272.06 | 0.35 |
| Weir-2 | Pond_2 | Pump_2 | Sharp-crested weir | 1340.00 | 2.00 | 537.02 | 1354.82 | 1.95 |
| Gate-3 | Pond_3 | Pump_3 | Sluice gate | 1020.00 | 10.00 | 946.35 | 1039.00 | 5.09 |

Parameters for hydraulic structures: 9

[-999.99, not flow under gate]

| Structure name | Upstream node name | Downstream node name | Structure type | Base elevation (feet) | Weir length (feet) | Discharge (cubic feet per second) | Upstream Water elevation (feet) | Gate-opening or weir height (feet) |
|---|---|---|---|---|---|---|---|---|
| Gate-1 | Pond_1 | Pump_1 | Gate spillway | 1240.00 | 2.00 | 20.12 | 1272.36 | 0.34 |
| Weir-2 | Pond_2 | Pump_2 | Sharp-crested weir | 1340.00 | 2.00 | 0.00 | 1350.00 | 1.95 |
| Gate-3 | Pond_3 | Pump_3 | Sluice gate | 1020.00 | 10.00 | 346.65 | 1039.00 | 1.74 |

Parameters for hydraulic structures: 10

[-999.99, not flow under gate]

| Structure name | Upstream node name | Downstream node name | Structure type | Base elevation (feet) | Weir length (feet) | Discharge (cubic feet per second) | Upstream Water elevation (feet) | Gate-opening or weir height (feet) |
|---|---|---|---|---|---|---|---|---|
| Gate-1 | Pond_1 | Pump_1 | Gate spillway | 1240.00 | 2.00 | 20.47 | 1271.42 | 0.35 |
| Weir-2 | Pond_2 | Pump_2 | Sharp-crested weir | 1340.00 | 2.00 | 15.92 | 1349.74 | 7.96 |
| Gate-3 | Pond_3 | Pump_3 | Sluice gate | 1020.00 | 10.00 | 90.80 | 1039.00 | 0.43 |

Parameters for hydraulic structures: 11

[-999.99, not flow under gate]

| Structure name | Upstream node name | Downstream node name | Structure type | Base elevation (feet) | Weir length (feet) | Discharge (cubic feet per second) | Upstream Water elevation (feet) | Gate-opening or weir height (feet) |
|---|---|---|---|---|---|---|---|---|
| Gate-1 | Pond_1 | Pump_1 | Gate spillway | 1240.00 | 2.00 | 20.46 | 1270.60 | 0.36 |
| Weir-2 | Pond_2 | Pump_2 | Sharp-crested weir | 1340.00 | 2.00 | 15.94 | 1349.35 | 7.58 |
| Gate-3 | Pond_3 | Pump_3 | Sluice gate | 1020.00 | 10.00 | 50.37 | 1039.00 | 0.24 |

Parameters for hydraulic structures: 12

[-999.99, not flow under gate]

| Structure name | Upstream node name | Downstream node name | Structure type | Base elevation (feet) | Weir length (feet) | Discharge (cubic feet per second) | Upstream Water elevation (feet) | Gate-opening or weir height (feet) |
|---|---|---|---|---|---|---|---|---|
| Gate-1 | Pond_1 | Pump_1 | Gate spillway | 1240.00 | 2.00 | 20.12 | 1269.87 | 0.35 |
| Weir-2 | Pond_2 | Pump_2 | Sharp-crested weir | 1340.00 | 2.00 | 15.76 | 1349.21 | 7.44 |
| Gate-3 | Pond_3 | Pump_3 | Sluice gate | 1020.00 | 10.00 | 53.35 | 1038.33 | 0.26 |

## Output File for Time-Series Output of Water Budget for Node Pond_1 (bpond_1.out)

Water Budgets for POND_1

| No. | Initial storage (acre-feet) | Upstream inflow (acre-feet) | Local net inflow (acre-feet) | Evapo-ration (acre-feet) | Runoff (acre-feet) | Seepage (acre-feet) | Withdrawal (acre-feet) | Downstream release (acre-feet) | Final storage (acre-feet) | Final stage (feet) | depth (feet) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 48506.50 | 0.00 | 1729.16 | 819.82 | 26.95 | 0.00 | 0.00 | 1275.66 | 48167.13 | 1273.89 | 34.89 |
| 2 | 48167.13 | 0.00 | 1593.61 | 817.88 | 0.00 | 0.00 | 0.00 | 12627.76 | 36315.10 | 1270.00 | 31.00 |
| 3 | 36315.10 | 0.00 | 13222.35 | 717.41 | 0.00 | 0.00 | 0.00 | 25443.95 | 23376.10 | 1265.00 | 26.00 |
| 4 | 23376.10 | 0.00 | 19412.21 | 581.53 | 0.00 | 0.00 | 0.00 | 443.09 | 41763.69 | 1271.85 | 32.85 |
| 5 | 41763.69 | 0.00 | 6358.12 | 766.47 | 0.00 | 0.00 | 0.00 | 1219.14 | 46136.20 | 1273.25 | 34.25 |
| 6 | 46136.20 | 0.00 | 723.75 | 806.29 | 0.00 | 0.00 | 0.00 | 1235.36 | 44818.30 | 1272.84 | 33.84 |
| 7 | 44818.30 | 0.00 | -407.64 | 796.43 | 0.00 | 0.00 | 0.00 | 1213.84 | 42400.39 | 1272.06 | 33.06 |
| 8 | 42400.39 | 0.00 | 2926.71 | 771.11 | 0.00 | 0.00 | 0.00 | 1235.39 | 43320.60 | 1272.35 | 33.35 |
| 9 | 43320.60 | 0.00 | -840.34 | 780.74 | 0.00 | 0.00 | 0.00 | 1213.84 | 40485.67 | 1271.42 | 32.42 |
| 10 | 40485.67 | 0.00 | -435.46 | 758.91 | 0.00 | 0.00 | 0.00 | 1235.39 | 38055.91 | 1270.60 | 31.60 |
| 11 | 38055.91 | 0.00 | -126.75 | 737.82 | 0.00 | 0.00 | 0.00 | 1234.71 | 35956.62 | 1269.87 | 30.87 |
| 12 | 35956.62 | 0.00 | 77.64 | 715.59 | 0.00 | 0.00 | 0.00 | 1213.86 | 34104.81 | 1269.21 | 30.21 |

## Output File for Time-Series Output of Release from Node Pond_1 (rpond_1.dat)

Outflows from POND_1

| | PUMP_1 | | PUMP_2 | |
|---|---|---|---|---|
| No. | (acre-feet) | (cubic feet per second) | (acre-feet) | (cubic feet per second) |
| 1 | 1275.66 | 21.14 | 0.00 | 0.00 |
| 2 | 12627.76 | 209.29 | 0.00 | 0.00 |
| 3 | 25443.95 | 421.70 | 0.00 | 0.00 |
| 4 | 443.09 | 7.34 | 0.00 | 0.00 |
| 5 | 1219.14 | 20.21 | 0.00 | 0.00 |
| 6 | 1235.36 | 20.47 | 0.00 | 0.00 |
| 7 | 1213.84 | 20.12 | 0.00 | 0.00 |
| 8 | 1235.39 | 20.47 | 0.00 | 0.00 |
| 9 | 1213.84 | 20.12 | 0.00 | 0.00 |
| 10 | 1235.39 | 20.47 | 0.00 | 0.00 |
| 11 | 1234.71 | 20.46 | 0.00 | 0.00 |
| 12 | 1213.86 | 20.12 | 0.00 | 0.00 |

## Output File for Time-Series Output of Arc Water Budget (arbud001.out)

Canal water budgets from POND_1 to PUMP_1

| No. | Inflow (acre-feet) | Initial storage (acre-feet) | Canal seepage (acre-feet) | Water-surface evaporation (acre-feet) | Final storage (acre-feet) | Outflow (acre-feet) | Outflow (cubic feet per second) |
|---|---|---|---|---|---|---|---|
| 1 | 1275.66 | 0.00 | 12.76 | 0.00 | 40.53 | 1222.37 | 20.26 |
| 2 | 12627.76 | 40.53 | 126.28 | 0.00 | 402.28 | 12139.73 | 201.20 |
| 3 | 25443.95 | 402.28 | 254.44 | 0.00 | 818.78 | 24773.01 | 410.58 |
| 4 | 443.09 | 818.78 | 4.43 | 0.00 | 35.06 | 1222.37 | 20.26 |
| 5 | 1219.14 | 35.06 | 12.19 | 0.00 | 39.64 | 1202.37 | 19.93 |
| 6 | 1235.36 | 39.64 | 12.35 | 0.00 | 40.27 | 1222.37 | 20.26 |
| 7 | 1213.84 | 40.27 | 12.14 | 0.00 | 39.60 | 1202.37 | 19.93 |
| 8 | 1235.39 | 39.60 | 12.35 | 0.00 | 40.27 | 1222.37 | 20.26 |
| 9 | 1213.84 | 40.27 | 12.14 | 0.00 | 39.60 | 1202.37 | 19.93 |
| 10 | 1235.39 | 39.60 | 12.35 | 0.00 | 40.27 | 1222.37 | 20.26 |
| 11 | 1234.71 | 40.27 | 12.35 | 0.00 | 40.26 | 1222.37 | 20.26 |
| 12 | 1213.86 | 40.26 | 12.15 | 0.00 | 39.61 | 1202.37 | 19.93 |

# APPENDIX E. COMPUTER PROGRAM LISTING

```
*=================================================================
* Name:         OPONDS
* Purpose:      Optimal operation of a system of ponds using linear
*               network flow model.
* Platform:     Unix/DG Aviion
* Version:      1.0
* Author:       Xiaodong Jian
*               U.S. Geological Survey
*               4821 Quail Crest Place
*               Lawrence, KS 66049
* Date:         July, 1997
*=================================================================
      PROGRAM   OPONDS
      IMPLICIT  NONE
      INTEGER   LDRES, LDND, LDREAR, LDARC, LDRETB, LDCTAR, LDP, LDFIL,
     &          LDFXAR, NTLS
      PARAMETER (LDRES = 100, LDND = 300, LDREAR = 200, LDARC = 1000,
     &           LDRETB = 5000, LDCTAR = 10, LDP = 365, LDFIL = 35,
     &           LDFXAR = 10, NTLS = 5)
      INTEGER   II(LDARC), JJ(LDARC), HI(LDARC), LO(LDARC), COST(LDARC),
     &          FLOW(LDARC), ARTYP(LDARC), ARCBUD(LDARC, 0:6)
      INTEGER   OHI(LDARC)
      INTEGER   REAR(LDREAR), PTRE(LDRES), PTDWAR(LDND, 2),
     &          NDDWAR(LDARC)
      REAL      REZN(LDREAR)
      CHARACTER NDNAM(LDND)*12
      INTEGER   NDTYP(LDND), NDSEQ(LDND), NDXAR(LDND,6),
     &          NODBUD(LDND,0:10)
      INTEGER   NNODS, NDWAR, SKSC, NARCS
      REAL      APRX, CONST, PERD
      LOGICAL   ZEROFG
      INTEGER   LDRC, LDRCTB
      PARAMETER (LDRC = LDRES, LDRCTB = LDP)
      REAL      INST(LDRES), RC(LDRES)
      INTEGER   NRCND, RCND(LDRC, 3), RCUNIT
      REAL      RCTB(0:LDRCTB, LDRC)
      LOGICAL   RCFLAG, RCFIL
      CHARACTER SYSNAM*40, MNTH*5
      INTEGER   PN
      REAL      OINST(LDRES)
      INTEGER   NPER
      INTEGER   NOP, NSPS, STMO, YR, I, NRES
      INTEGER   ARCS
      INTEGER   ARC, YEAR, MTH, KARC
      INTEGER   J, MXCST
      INTEGER   FCRIT, OFLOW(LDARC), NITR, LDITR, OARCS
      INTEGER   ISGN
      LOGICAL   FLWARC, NOTCOV
      INTEGER   IN, IN_IFW, IN_RN, IN_EV, IN_WS, IN_RC, IN_GW, IN_FX
      INTEGER   IN_FB
      INTEGER   OU_NT, OU_ND, OU_AR, OU_HY, OU_SN, OU_SA
      LOGICAL   NDBFLG, ARBFLG, HYBFLG
      INTEGER   CTARFW(LDCTAR, 3), NCTAR
      CHARACTER FILNAM(0:LDFIL)*30
      INTEGER   LDSTRM
      PARAMETER (LDSTRM = LDARC)
      INTEGER   STRMAR(LDSTRM, 0:6), NSTRM, NARCND
      REAL      STRMCF(LDSTRM, 0:4) ! OCF(LDSTRM)
      INTEGER   LDSTR
      PARAMETER (LDSTR = LDSTRM)
      INTEGER   NSTR, STRDIR(LDSTR, 3)
      REAL      STRDAT(LDSTR, 9)
      INTEGER   LDHY, LDHYTP
      PARAMETER (LDHY = LDND, LDHYTP = 6)
      CHARACTER HYDIR(LDHY, 0:2)*12
      INTEGER   NHY, HYTPCD(LDHY, 0:1), NHYTP
      REAL      HYDAT(LDHY, 5), HYOUT(LDHY, 3)
      CHARACTER HYTP(0:LDHYTP)*20
      INTEGER   PTRES(LDRES)
      REAL      RESDAT(LDRES, 0:2)
      INTEGER   XP
      REAL      XF
      INTEGER   LDIFW
      PARAMETER (LDIFW = LDND)
      INTEGER   NIFW, IFWND(LDIFW, 3), IFWCD
      INTEGER   NFXAR, FXAR(0:2, LDFXAR), FXUNIT
      LOGICAL   FWFLAG, FXFLAG
      INTEGER   LDWS
      PARAMETER (LDWS = LDND)
      INTEGER   NWSND, WSND(LDWS, 3), WSUNIT
      REAL      WSTB(0:LDP, LDWS)
      LOGICAL   WSFLAG, WSFIL
      INTEGER   LDEV
      PARAMETER (LDEV = LDRES)
      INTEGER   NEV, EVND(LDEV, 3), EVUNIT
      REAL      EVTB(0:LDP, LDEV)
      LOGICAL   EVFLAG, EVFIL
      INTEGER   LDGWND
```

```
      PARAMETER (LDGWND = LDND)
      INTEGER   NGWND, GWND(LDGWND, 3), GWTYPE
      REAL      GWLVL(LDGWND)
      LOGICAL   GWFLAG
      CHARACTER GWUNIT(0:2, 2)*21
      INTEGER   LDRAIN, LDRNOF, LDRFTB
      PARAMETER (LDRAIN = LDND, LDRNOF = LDND, LDRFTB = 4)
      INTEGER   NRAIN, RAINND(LDRAIN, 3), RAINTY
      LOGICAL   RNFLAG
      INTEGER   NRNOF, RNOFND(LDRNOF)
      REAL      RNOFTB(LDRNOF, 0:LDRFTB), A5DR(LDRNOF, 5)
      CHARACTER RNUNIT(0:2, 2)*21
      INTEGER   LDFBAR, LDFBTB
      PARAMETER (LDFBAR = LDARC, LDFBTB = LDP)
      INTEGER   NFBAR, FBAR(0:7, LDFBAR), FBUNIT
      REAL      FBTB(0:LDFBTB, LDFBAR)
      LOGICAL   FBFLAG, FBFIL
      INTEGER   LDSNBL, LDSABL
      PARAMETER (LDSNBL = LDRES, LDSABL = 100)
      INTEGER   NSNBL, SNBLND(LDSNBL, 3), NSABL, SABLND(LDSABL, 3)
      LOGICAL   SNBLFG, SABLFG
      INTEGER   LDUNIT
      PARAMETER (LDUNIT = 2)
      CHARACTER UNITNM_1(0:2)*21, UNITNM_2(0:2)*21, UNITNM_3(0:2)*21
      REAL      CNDBT(0:10), CARBT(6), CSNDBT(LDND, 0:10),
     &          CSARBT(LDARC,0:6)
      INTEGER   SAVOPT
      INTEGER   LDPL, LDCOL
      PARAMETER (LDPL = LDND, LDCOL = 50)
      CHARACTER CTERM*500, COLSTR(LDCOL)*30
      REAL      RTERM, RPOOL(LDPL)
      INTEGER   IFAULT
      LOGICAL   FLAG
      LOGICAL   ERR, LAST, DEBUG, CHECK
      COMMON    /NDNAME/ NDNAM
      COMMON    /SAVOPT/ SAVOPT
      COMMON    /CHECK/ CHECK
*
      DATA      UNITNM_1/ 'acre-feet', 'cubic feet per second',
     &                     'cubic feet per day'/
      DATA      UNITNM_2/ 'feet', 'inches', 'millimeters'/
      DATA      UNITNM_3/ 'millimeters per day', 'inches per day',
     &                     'feet per day'/
      DATA      GWUNIT / 'feet', 'inches', 'meter',
     &                   'acre-feet per day', 'cubic feet per second',
     &                   'cubic feet per day'/
      DATA      RNUNIT / 'feet', 'inches', 'millimeters',
     &                   'acre feet per day', 'cubic feet per second',
     &                   'cubic feet per day'/
*
      ZEROFG = .TRUE.
*
*          Open data files and assign associated fortran file units
*
      CALL OPMDF(FILNAM, LDFIL, CTERM, COLSTR, LDCOL)
      PRINT *, 'Please wait! Processing data files'
      IN = 8
      CALL IO_OPFIL(IN, 1, FILNAM(0), 'ENTER NETWORK FILE: ')
*
*          Open output files
*
      OU_NT = 26       ! GENERAL OUPUT.
      OU_ND = 27       ! NODAL BUDGET
      OU_AR = 28       ! ARC BUDGET
      OU_HY = 29       ! FLOW IN STRUCTURE.
      OU_SN = 30       ! SINGLE NODAL BUDGET LIST.
      OU_SA = 31       ! SINGLE ARC BUDGET LIST.
      CALL IO_OPFIL(OU_NT, 3, FILNAM(OU_NT), 'ENTER GENERAL OUTPUT: ' )
      IF (FILNAM(OU_ND) .EQ. ' ') THEN
                                   NDBFLG = .FALSE.
      ELSE
                                   NDBFLG = .TRUE.
                       CALL IO_OPFIL(OU_ND, 3, FILNAM(OU_ND),
     &            'ENTER NODAL BUDGET OUTPUT: ')
      ENDIF
      IF (FILNAM(OU_AR) .EQ. ' ') THEN
                                   ARBFLG = .FALSE.
      ELSE
                                   ARBFLG = .TRUE.
                       CALL IO_OPFIL(OU_AR, 3, FILNAM(OU_AR),
     &                'ENTER ARC BUDGET OUTPUT: ')
      ENDIF
      IF (FILNAM(OU_HY) .EQ. ' ') THEN
                                   HYBFLG = .FALSE.
      ELSE
                                   HYBFLG = .TRUE.
                       CALL IO_OPFIL(OU_HY, 3, FILNAM(OU_HY),
     &                'ENTER STRUCTURE OUTPUT: ')
      ENDIF
*
*------------Assign fortran file units for input data files.
*
      IN_IFW = 16
      IN_RN  = 17
```

```
      IN_EV   = 18
      IN_WS   = 19
      IN_RC   = 20
      IN_FB   = 21
      IN_GW   = 22
      IN_FX   = 23
*
*             Initialize some arrays and assign constants.
*
      CONST = 86400.0 / 43560.0
      APRX = .5
*
*---------------Initialization
*
      NOP = 0
      NARCS = 0
      NCTAR = 0
      LAST = .FALSE.
      ERR = .FALSE.
      FLWARC = .FALSE.
      DO I = 1, LDARC
                                          FLOW(I) = 0
                                         OFLOW(I) = 0
                                          COST(I) = 0
                                          REAR(I) = 0
                                        NDDWAR(I) = 0
      ENDDO
      DO I = 1, LDND
                                         NDNAM(I) = ' '
                                       PTDWAR(I, 1) = 0
                                       PTDWAR(I, 2) = 0
      ENDDO
      DO 13 I = 1, LDRES
                                          PTRE(I) = 0
   13 CONTINUE
      SKSC = LDND
      NDNAM(SKSC) = 'SKSC'
      NDWAR = 0
      NSTRM = 0
      NITR = 0
*
*             Set default value
*
      PERD = 30.42
      NPER = 12
      XP = 0
      XF = 10**XP
      FCRIT = INT(0.1 * PERD * CONST * XF)
      LDITR = 100
      NIFW = 0
      NWSND = 0
      NEV = 0
      NRCND = 0
      NRAIN = 0
      NGWND = 0
      NFBAR = 0
      NFXAR = 0
*
*             Read symtem name, simulation period, time period length, and
*             output accuracy in no. of decimal points in ac-ft.
*
      READ (IN, '(A)') SYSNAM
      READ (IN, '(A)')
      READ (IN, *) PERD, NPER
      READ (IN, *) STMO, YR
      READ (IN, *) NSPS
      READ (IN, *) RTERM, LDITR
      READ (IN, *) XP
*
      SAVOPT = 0
      CALL GETINT(SAVOPT, IN, IFAULT)
*
      WRITE (OU_NT, 802) SYSNAM, PERD, NPER, STMO, YR,
     &                   NSPS, RTERM, LDITR, XP
  802 FORMAT(A,
     &   //, 'Summary of simulation period: ',
     &    /, '=============================',
     &    /, '            Length of a time step: ', F7.2, ' days',
     &    /, '  Number of time steps of a year: ', I4,
     &    /, '                 Starting season: ', I4,
     &    /, '                   Starting year: ', I4,
     &    /, ' Number of simulation time steps: ', I4,
     &    /, '      Flow-convergence criterion: ', F8.3,
     &       ' cubic feet per second',
     &    /, '         Maximum iteration steps: ', I4,
     &    /, '        Number of decimal points: ', I4)
*
      XP = XP + 1
      XF = 10**XP
      IF (RTERM .NE. 0) THEN
                            FCRIT = INT(RTERM * PERD * CONST * XF)
      ENDIF
*
*------------READ RESERVOIR CAPACITY TABLES
```

```
*
      CALL SETZVA(NRES, NNODS, NDNAM, NDTYP, LDND,
     &             FILNAM(9), NTLS, OU_NT)
*
*            Read reservoir zoning, network configuration,
*            other physical parameters, and seasonal-dependent
*            data.
*
*
*-----------------1. READ RESERVOIR ZONES (BOUNDS) AND CREATE THE BASIC
*                      storage arcs of reservoirs.
      PN = 1
      CALL  NETWK_1(NDNAM, NDSEQ, LDND,
     &               NARCS, II, JJ, HI, LO, COST, ARTYP, LDARC,
     &               NRES, PTRES, RC, INST, RESDAT, LDRES,
     &               PTRE, LDRES, REAR, REZN, LDREAR,
     &               SKSC, XF, FILNAM(PN), NTLS,
     &               IN, OU_NT, PN)
*
*--------------2. Channeel flow bounds and stream routing
*                 coefficients
      PN = 2
      CALL  NETWK_2(NNODS, NDNAM, NDTYP, LDND,
     &               NARCS, II, JJ, HI, LO, COST, ARTYP, LDARC,
     &               NARCND, NSTRM, STRMCF, STRMAR, LDSTRM,
     &               PTDWAR, LDND, NDWAR, NDDWAR, LDARC,
     &               NCTAR, CTARFW, LDCTAR,
     &               PERD, CONST, XF, FILNAM(PN), NTLS,
     &               IN, OU_NT, PN)
      IF (NSTRM .GT. 0) FLWARC = .TRUE.
      ARCS = NARCS
      MXCST = -9999
      DO I = 1, NARCS
                           IF (COST(I) .GT. MXCST) MXCST = COST(I)
      ENDDO
      DO 16 I = 1, NRES
                                      OINST(I) = INST(I)
 16   CONTINUE
*
*----------------3. Read Channel geometry data such as length, roughtness
*                      slope, riverbed hydraulic conductivity.
      PN = 3
      CALL STRM_DAT(NDNAM, LDND, II, JJ, LDARC,
     &               NSTRM, STRMAR, LDSTRM,
     &               NSTR, STRDIR, STRDAT, LDSTR,
     &               FILNAM(PN), NTLS, IN, OU_NT, PN)
      CALL  STRM_KX(NSTRM, STRMAR, STRMCF, LDSTRM,
     &               NSTR,  STRDIR, STRDAT, LDSTR)
*
*----------------4. Read parameters for hydraulic structures
*
      PN = 4
      CALL HYDR_DAT(NNODS, NDNAM, LDND, JJ, LDARC,
     &               PTDWAR, LDND, NDDWAR, LDARC,
     &               NSTRM, STRMAR, LDSTRM,
     &               NHYTP, HYTP, LDHYTP,
     &               NHY, HYDIR, HYTPCD, HYDAT, LDHY,
     &               FILNAM(PN), NTLS, IN, OU_NT, PN,
     &               CTERM, COLSTR, LDCOL)
*
*-----------------5. Surface runoff parameters
*
      PN = 5
      CALL RNOF_DAT(NDNAM, LDND,
     &           NRNOF, RNOFND, LDRNOF, RNOFTB, LDRFTB, A5DR,
     &           FILNAM(PN), NTLS, IN, OU_NT, PN,
     &           CTERM, COLSTR, LDCOL)
*
*-----------------10. Seasonal target water demands
*
      PN = 10
      WSFLAG = .FALSE.
      CALL TWS_DAT(NNODS, NDNAM, LDND, NWSND, WSND, LDWS, NPER, WSTB,
     &             LDP, WSUNIT, WSFLAG, PN, FILNAM(PN), NTLS, IN, OU_NT,
     &             UNITNM_1, LDUNIT, CTERM, COLSTR, LDCOL)
*
*-------------------11. Seasonal surface water evaporation coefficients
*
      PN = 11
      CALL RESEV_DAT (NNODS, NDNAM, LDND, NEV, EVND, LDEV,
     &                NPER, EVTB, LDP, EVUNIT, EVFLAG,
     &                FILNAM(PN), NTLS, IN, OU_NT, PN,
     &                UNITNM_3, LDUNIT, CTERM, COLSTR, LDCOL)
*
*------------------12. Seasonal flow bounds
*
      PN = 12
      CALL FB_DAT(II, JJ, ARTYP, LDARC, NDNAM, LDND,
     &                  NFBAR, FBAR, LDFBAR, FBTB,  LDFBTB,
     &                  FBUNIT, FBFLAG, FILNAM(PN), NTLS, IN, PN, OU_NT,
     &                  PTDWAR, LDND, NDDWAR, LDARC,
     &                  UNITNM_1, LDUNIT, CTERM, COLSTR, LDCOL)
*
*------------------13. Seasonal rule curve
```

```
*
      PN = 13
      CALL RC_DAT(NNODS, NDNAM, LDND,
     &            NRCND, RCND, LDRC, NPER, RCTB, LDRCTB, RCUNIT, RCFLAG,
     &            FILNAM(PN), NTLS, IN, OU_NT, PN,
     &            UNITNM_2, LDUNIT, CTERM, COLSTR, LDCOL)
*
*----------Open time-dependent data file
*
*
*----------1. Open local incremental inflow file
*
      CALL FIL_HEAD(NNODS, NDNAM, LDND, NIFW, IFWND, LDIFW, IFWCD,
     &              UNITNM_1, LDUNIT, FWFLAG, FLAG, FILNAM(IN_IFW),
     &              NTLS, IN_IFW, OU_NT, 'local incremental inflow',
     &              CTERM, COLSTR, LDCOL)
*
*----------2. Open a rainfall data file
*
      CALL FIL_HEAD(NNODS, NDNAM, LDND, NRAIN, RAINND, LDRAIN, RAINTY,
     &              RNUNIT, LDUNIT, RNFLAG, FLAG, FILNAM(IN_RN),
     &              NTLS, IN_RN, OU_NT, 'precipitation',
     &              CTERM, COLSTR, LDCOL)
*
*----------Open water surface coefficient file
*
      CALL FIL_HEAD(NNODS, NDNAM, LDND, NEV, EVND, LDEV, EVUNIT,
     &              UNITNM_3, LDUNIT, EVFLAG, EVFIL, FILNAM(IN_EV),
     &              NTLS, IN_EV, OU_NT,
     &              'water-surface evaporation coefficient',
     &              CTERM, COLSTR, LDCOL)
*
*----------Open target water demand file
*
      CALL FIL_HEAD(NNODS, NDNAM, LDND, NWSND, WSND, LDWS, WSUNIT,
     &              UNITNM_1, LDUNIT, WSFLAG, WSFIL, FILNAM(IN_WS),
     &              NTLS, IN_WS, OU_NT, 'target water-demand',
     &              CTERM, COLSTR, LDCOL)
*
*-------------Open rule curve data file
*
      CALL FIL_HEAD(NNODS, NDNAM, LDND, NRCND, RCND, LDRC, RCUNIT,
     &              UNITNM_2, LDUNIT, RCFLAG, RCFIL, FILNAM(IN_RC),
     &              NTLS, IN_RC, OU_NT, 'rule-curve elevation',
     &              CTERM, COLSTR, LDCOL)
*
*-------------Open flow bound data file
*
      CALL FB_FIL(NDNAM, LDND, II, JJ, ARTYP, LDARC,
     &            PTDWAR, LDND, NDDWAR, LDARC,
     &            NFBAR, FBAR, LDFBAR, FBUNIT,
     &            UNITNM_1, LDUNIT, FBFLAG, FBFIL, FILNAM(IN_FB),
     &            NTLS, IN_FB, OU_NT, CTERM, COLSTR, LDCOL)
*
*-------------Open a groundwater level data file
*
      CALL FIL_HEAD(NNODS, NDNAM, LDND, NGWND, GWND, LDGWND, GWTYPE,
     &              GWUNIT, LDUNIT, GWFLAG, FLAG, FILNAM(IN_GW),
     &              NTLS, IN_GW, OU_NT, 'ground-water-level',
     &              CTERM, COLSTR, LDCOL)
*
*-------------Open a fixed flow file
*
      CALL FX_FIL(NDNAM, LDND, II, JJ, ARTYP, LDARC,
     &            PTDWAR, LDND, NDDWAR, LDARC,
     &            NFXAR, FXAR, LDFXAR,
     &            FXUNIT, UNITNM_1, LDUNIT, FXFLAG,
     &            FILNAM(IN_FX), NTLS, IN_FX, OU_NT,
     &            CTERM, COLSTR, LDCOL)
*
*-------------WRITE BASIC NETWORK INFORMATION
*
      IF (SAVOPT .EQ. 0 .OR. SAVOPT .EQ. 1) THEN
                CALL PRTINF (NDNAM, NDTYP, NDSEQ, NRES, 0, LDND,
     &             LDARC, LDRES,
     &             II, JJ, HI, LO, COST, ARTYP, PTRE, REAR,
     &             NNODS, PTDWAR, NDDWAR,
     &             PERD, NARCS, CONST, XF, OU_NT)
      ENDIF
*
*-------------Open single nodal/arc budget list files
*
      CALL SNBL(NNODS, NDNAM, LDND, II, JJ, LDARC,
     &          PTDWAR, LDND, NDDWAR, LDARC,
     &          NSTRM, STRMAR, LDSTRM,
     &          NSNBL, SNBLND, LDSNBL, SNBLFG, FILNAM(OU_SN))
      CALL SABL(NDNAM, LDND, PTDWAR, II, JJ, NDDWAR, LDARC,
     &          NSTRM, STRMAR, LDSTRM,
     &          NSABL, SABLND, LDSABL, SABLFG,
     &          FILNAM(OU_SA))
*
      DO I = 1, NARCS
                                OHI(I) = HI(I)
      ENDDO
```

```
*
*-------------BEGIN TO CALCULATION
*
      PRINT *, 'Begin simulation'
 30   CONTINUE
      NOTCOV = .TRUE.
      NOP = NOP + 1
      IF (NOP .GT. 1) THEN
                 PRINT '(A, I4)', ' Finish simulation time period:', NOP-1
      ENDIF
      IF (NOP .GT. NSPS) GOTO 80
      MTH = NOP + STMO - 1
      MTH = MOD (MTH, NPER)
      IF (MTH .EQ. 0) MTH = NPER
*
*           PREPARE THE TIME FOR OUTPUT
*
      IF (STMO .EQ. 1 .AND. NOP .EQ. 1) THEN
                                        YEAR = YEAR - 1
      END IF
      IF (MTH .EQ. 1) THEN
                                        YEAR = YEAR + 1
      END IF
      IF (NPER .EQ. 12) THEN
                                   CALL MONTH(2, MTH, MNTH)
      ELSE
                                   WRITE(MNTH, '(I4)') MTH
      ENDIF
      DO I = 1, LDND
                                         DO J = 1, 6
                                         NDXAR(I, J) = 0
                                            ENDDO
      ENDDO
*
*           CALL SUBROUTINE NVAR TO GET THE NET INFLOW TO NODES
*           AND CREATE THE CORESPONGING NET INFLOW ARCS.
*
      CALL NVAR (NNODS, NDNAM, NDTYP, NDSEQ, LDND,
     &           NARCS, II, JJ, HI, LO, COST, ARTYP, LDARC,
     &           MTH, IN_IFW, LAST, LDRES,
     &           INST, RC, SKSC,
     &           NDXAR, PERD, XF,
     &           NIFW, IFWND, LDIFW,
     *           PTRE, LDRES, REAR, REZN, LDREAR,
     *           NRCND, RCND, LDRC, RCTB, LDRCTB, RCFLAG, RCFIL,
     *           IN_RC,
     &           APRX, OU_NT,
     &           RPOOL, LDPL, CTERM, COLSTR, LDCOL)
      IF (ERR) GO TO 99
      IF (FBFLAG) THEN
                               CALL FB_ARC(LO, HI, ARTYP, LDARC,
     &               NFBAR, FBAR, LDFBAR, FBTB, LDFBTB, FBFIL,
     &               MTH, PERD, XF, IN_FB,
     &               CTERM, COLSTR, LDCOL)
      ENDIF
*
*------------Define rainfall arcs
*
      IF (RNFLAG) THEN
                               CALL  RNOF_ARC(NDTYP, NDSEQ, LDND,
     &                  NARCS, II, JJ, LO, HI, COST, ARTYP, LDARC,
     &                  INST,   LDRES,
     &                  NDXAR, LDND, NRAIN, RAINND, LDRAIN, RAINTY,
     &                  NRNOF, RNOFND, LDRNOF, RNOFTB, LDRFTB,
     &                  A5DR, PERD,
     &                  SKSC, XF, IN_RN,
     &                  CTERM, COLSTR, LDCOL)
      ENDIF
*
*-------------Define reservoir water surface EV arc
*
      IF (EVFLAG) THEN
                              CALL  RESEV_ARC(NDTYP, NDSEQ, LDND,
     &                  NARCS, II, JJ, LO, HI, COST, ARTYP, LDARC,
     &                  INST, LDRES,
     &                  NDXAR, LDND, NEV, EVND, LDEV, EVTB, LDP,
     &                  MTH, SKSC, PERD, EVFIL, XF, IN_EV,
     &                  CTERM, COLSTR, LDCOL)
      ENDIF
*
*-------------Target water demand arcs
*
      IF (WSFLAG) THEN
                  CALL TWS_ARC(NARCS, II, JJ, LO, HI, COST, ARTYP, LDARC,
     &                NWSND, WSND, LDWS, WSTB, LDP, WSFIL,
     &                NDXAR, LDND, MTH, SKSC, PERD, MXCST, XF, IN_WS,
     &                CTERM, COLSTR, LDCOL)
      ENDIF
*
*-------------Define Reservoir seepage arcs
*
      IF (GWFLAG) THEN
                  CALL GW_DAT(NGWND, GWND, GWLVL, LDGWND, GWTYPE, IN_GW,
     &               CTERM, COLSTR, LDCOL)
```

```
      ENDIF
      CALL RESLOS(
     &             NARCS, II, JJ, LO, HI, COST, ARTYP, LDARC,
     &             NRES, PTRES, INST, RESDAT, LDRES,
     &             NGWND, GWND, GWLVL, LDGWND,   GWTYPE,
     &             NDXAR, LDND, SKSC, PERD, XF)
*
*------------Calculate the overflow over weir
*
      CALL  HYDR_OFW(NDSEQ, LDND,
     &                II, HI, LO, OHI, LDARC,
     &                NHY, HYTPCD, HYDAT, HYOUT, LDHY,
     &                OINST, LDRES,
     &                PERD, CONST, XF, OU_NT)
*
*------------Assign flows for fixed flow arcs
*
      IF (FXFLAG) THEN
              CALL FX_ARC(HI, LO, COST, LDARC, NFXAR, FXAR, LDFXAR, FXUNIT,
     &                 XF, PERD, IN_FX,
     &                 RPOOL, LDPL, CTERM, COLSTR, LDCOL)
      ENDIF
*
*          Flow dependent arcs such as Channel seepage arcs
*
      IF (FLWARC) THEN
                                                NITR = 0
                                             OARCS = NARCS
                                            DO I = 1, LDARC
                                               OFLOW(I) = 0
                                                 ENDDO
      ENDIF
 40   CONTINUE
      IF (FLWARC) THEN
                                             NARCS = OARCS
                                            IF (EVFIL) THEN
                                                  I = 0
                                                 ELSE
                                                 I = MTH
                                                 ENDIF
*
*
                                            CALL STRM_ROUT(
     &           NARCS, II, JJ, HI, LO, COST, FLOW, ARTYP, LDARC,
     &           NSTRM, STRMCF, STRMAR, LDSTRM,
     &           NSTR, STRDIR, STRDAT, LDSTR,
     &           NGWND, GWND, GWLVL, LDGWND,
     &           NEV, EVND, LDEV, I, EVTB, LDP,
     &           SKSC, PERD, XF)
      ENDIF
*
*
*           SOLVE THE NETWORK BY USING THE OUT OF KILTER TECHNIQUE
*
      CALL KLTR(II, JJ, HI, LO, COST, FLOW, LDND, NARCS, DEBUG, KARC,
     &          IFAULT)
      IF (IFAULT .NE. 0) THEN
                                                 STOP
      ENDIF
      IF (DEBUG) THEN
                           IF (FLWARC .AND. NITR .LT. LDITR) THEN
                                         NITR = NITR + 1
                                                GOTO 40
                                                ELSE
                    CALL OUTPUT2(NDNAM, LDND, II, JJ, HI, LO, COST, FLOW,
     &                   ARTYP, NARCS)
                          PRINT *, '  ******SOLUTION IS INFEASIBLE******'
                     PRINT *, 'CHECK THE ARC FROM ', NDNAM(II(IABS(KARC))),
     &                ' TO ', NDNAM(JJ(IABS(KARC))), 'WITH ARC NUMBER:',
     &               ABS(KARC)
                           PRINT *, 'THE CURRENT TIME PERIOD IS ', NOP
                             PRINT *, 'THE LOCAL ITERATION IS ', NITR
                                                   STOP
*           STOP !CALL EXIT
                                                ENDIF
      END IF
      IF (FLWARC) THEN
                                        CALL FLWCK(NNODS, LDND,
     &               PTDWAR, NDDWAR, LDARC,
     &               FLOW, OFLOW, FCRIT, NOTCOV)
*
                                          IF (NOTCOV) THEN
                                            NITR = NITR + 1
                                       IF (NITR .LE. LDITR) THEN
                                                  GOTO 40
                                                  ELSE
                                                  GOTO 43
                                                 ENDIF
                                                ENDIF
      ENDIF
 43   CONTINUE
*
*
*           CALCULATE STAGES OF PONDS
```

```
*
      CALL RESSTG(NNODS, NDTYP, NDSEQ, LDND,
     &            RC, INST, PTRE, LDRES, REAR, LDREAR,
     &            FLOW, LDARC, XF)
*
*          Calculate nodal budgets and save into a file
*
      CALL NDBUD(FLOW, LDARC, RC, OINST, INST, RESDAT, LDRES,
     &            NNODS, NDTYP, NDSEQ, PTDWAR, LDND, NDDWAR, LDARC,
     &            NDXAR, NODBUD, XF, OU_NT)
      CALL PRTND(NNODS, NDNAM, NDTYP, NODBUD, LDND, NOP, XF, XP,
     &           NSNBL, SNBLND, LDSNBL, SNBLFG, NDBFLG, ZEROFG, OU_ND)
*
*           Calculate arc budgets
*
      CALL ARBUD(NNODS, PTDWAR, LDND, NDDWAR, LDARC,
     &             II, JJ, FLOW, ARCBUD, LDARC,
     &             NSTRM, STRMAR, LDSTRM)
      CALL  PTARBD(II, JJ, ARCBUD, NDDWAR, LDARC,
     &            NNODS, NDNAM, PTDWAR, LDND,
     &            NSABL, SABLND, LDSABL,
     &            NSNBL, SNBLND, LDSNBL, SNBLFG,
     &            NOP, XF, XP, ARBFLG, PERD, CONST, OU_AR)
*
*          Flow in the hydraulic structures
*
      IF (NHY .GT. 0) THEN
                                CALL HYDR_HITE(
     &                ARCBUD, LDARC,
     &
     &                NHY, HYTPCD, HYDAT, HYOUT, LDHY,
     &                PERD, CONST, XF)
                          CALL HYDR_PRN(HYTP, LDHYTP,
     &                NHY, HYDIR, HYTPCD, HYDAT, HYOUT, LDHY, NOP,
     &                HYBFLG, OU_HY)
      ENDIF
*
*------------System water budget
*
      CALL SAVBUD(NNODS, NDNAM, NDTYP, NODBUD, PTDWAR, LDND,
     &            II, JJ, ARCBUD, NDDWAR, LDARC,
     &            NOP, NSPS, XF, XP, OU_NT,
     &            CNDBT, CARBT, CSNDBT, LDND, CSARBT, LDARC)
*
300   CONTINUE
      DO I = 1, NRES
                              OINST(I) = INST(I)
      ENDDO
      DO I = 1, NSTRM
                              ARC = STRMAR(I, 5)
                    STRMCF(I, 0) = ISGN(ARC)*FLOW(IABS(ARC))/XF
      ENDDO
      IF (NARCS .GT. ARCS) THEN
                            DO I = ARCS + 1, NARCS
                                    II(I) = 0
                                    JJ(I) = 0
                                    HI(I) = 0
                                    LO(I) = 0
                                  FLOW(I) = 0
                                  COST(I) = 0
                                   ENDDO
      ENDIF
      NARCS = ARCS
      GO TO 30
 80   CONTINUE
      CALL SVINFO(FILNAM, LDFIL, NHY, NSNBL, NSABL,
     &            NDBFLG, ARBFLG, HYBFLG)
 99   STOP
      END
*=================================================================
      SUBROUTINE GETINT(IVAR, IN, IFAULT)
      IMPLICIT   NONE
      INTEGER    IVAR, IN, IFAULT
*
      CHARACTER  CTERM*50
*
      CTERM = ' '
      IFAULT = 1
      READ (IN, '(a)', END = 5) CTERM
 5    CONTINUE
      IF (CTERM .NE. ' ') THEN
                                BACKSPACE(IN)
                              READ (IN, *) IVAR
                                 IFAULT = 0
      ENDIF
      RETURN
      END
*=================================================================
* Name:       arbud
* Purpose:    Calculate the water budget for arcs.
* Author:     Xiaodong Jian
* Date:       1/16/95
*=================================================================
      SUBROUTINE ARBUD(NND, PTDWAR, LDND, DWAR, LDDWAR,
```

```
     &                  II, JJ, FLOW, ARCBUD, LDARC,
     &                  NSTRM, STRMAR, LDSTRM)
      IMPLICIT    NONE
      INTEGER     NND, LDND, PTDWAR(LDND, 2), LDDWAR, DWAR(LDDWAR)
      INTEGER     LDARC, II(LDARC), JJ(LDARC), FLOW(LDARC),
     &            ARCBUD(LDARC, 0:6)
      INTEGER     NSTRM, LDSTRM, STRMAR(LDSTRM, 0:6)
*
      INTEGER     I, J, N, STRM, DWND, ARC, LIM1, LIM2, ARC2
      INTEGER     ISGN
      LOGICAL     UNSTRM
*
      DO 100 N = 1, NND
                                     LIM1 = PTDWAR(N, 1)
                                     LIM2 = PTDWAR(N, 2)
                                    DO 50 J = LIM1, LIM2
                                         ARC = DWAR(J)
*
*              Check if the current arc is a stream arc.
*
                                        UNSTRM = .TRUE.
                                             I = 1
                              DO WHILE (I .LE. NSTRM .AND. UNSTRM)
                                  IF (ARC .EQ. STRMAR(I, 1)) THEN
                                       UNSTRM = .FALSE.
                                           STRM = I
                                               ENDIF
                                             I = I + 1
                                             ENDDO
*
*----------Calculate the arc water budget.
*
                                        IF (UNSTRM) THEN
                                        IF (ARC .GT. 0) THEN
                                        DWND = JJ(ARC)
                                               ELSE
                                          ARC = -ARC
                                        DWND = II(-ARC)
                                              ENDIF
                                       ARCBUD(ARC, 0) = DWND
                                    ARCBUD(ARC, 1) = FLOW(ARC)
                                        ARCBUD(ARC, 2) = 0
                                        ARCBUD(ARC, 3) = 0
                                        ARCBUD(ARC, 4) = 0
                                        ARCBUD(ARC, 5) = 0
                                    ARCBUD(ARC, 6) = FLOW(ARC)
                                             ELSE
                                      ARC2 = STRMAR(STRM, 6)
                                       IF (ARC2 .GT. 0) THEN
                                        DWND = JJ(ARC2)
                                    ELSE IF (ARC2 .LT. 0) THEN
                                       DWND = II(-ARC2)
                                              ENDIF
                                         ARC = IABS(ARC)
                                       ARCBUD(ARC, 0) = DWND
                                           DO I = 1, 6
                                    ARC2 = STRMAR(STRM, I)
                                    IF (ARC2 .NE. 0) THEN
                         ARCBUD(ARC, I) = ISGN(ARC2) * FLOW(IABS(ARC2))
                                            ELSE
                                       ARCBUD(ARC, I) = 0
                                            ENDIF
                                              ENDDO
                                             ENDIF
 50       CONTINUE
 100  CONTINUE
      RETURN
      END
*===================================================================
* Name:       ptarbd
* Purpose:    print arc water budget into a file.
* Author:     Xiaodong Jian
* Date:       1/16/96
*===================================================================
      SUBROUTINE PTARBD(II, JJ, ARCBUD, NDDWAR, MXARC,
     &                  NNODS, NDNAM, PTDWAR, MXND,
     &                  NSABL, SABLND, LDSABL,
     &                  NSNBL, SNBLND, LDSNBL, SNBLFG,
     &                  NOP, XF, XP, ARBFLG, PERD, CONST, IOUT)
      IMPLICIT    NONE
      INTEGER     MXARC, MXND, NNODS, NOP, XP, IOUT
      INTEGER     II(MXARC), JJ(MXARC), ARCBUD(MXARC, 0:6)
      INTEGER     PTDWAR(MXND, 2), NDDWAR(MXARC)
      REAL        XF, PERD, CONST
      CHARACTER   NDNAM(MXND)*(*)
*
      INTEGER     NSABL, LDSABL, SABLND(LDSABL, 3)
      INTEGER     NSNBL, LDSNBL, SNBLND(LDSNBL, 3)
      LOGICAL     SNBLFG, ARBFLG
*
      INTEGER     I, J, K, L, N, OJ, ARC, LIM1, LIM2, BUD(6)
      INTEGER     OU, NOFW, OFWID, OFW(20), NARC
*
      CHARACTER   FMT*40, FMT2*30, OFWFMT*30
```

```
      LOGICAL    OFWFLG, CN2INT, CNINT
*
      NARC = 0
      IF (XP .GT. 0) THEN
                           FMT = '(I4, 1X, 2A12, T30, 6F10.0, F10.2)'
                                WRITE(FMT(26:26), '(I1)') XP - 1
                                FMT2 = '(I4, 1X, 6F10.0, F10.2)'
                               WRITE(FMT2(15:15), '(I1)') XP - 1
                             OFWFMT = '(I4, 1X, 20(F10.0, F10.2))'
                              WRITE(OFWFMT(17:17), '(I1)') XP - 1
      ELSE
                            FMT = '(I4, 1X, 2A12, T30, 6I10, F10.2)'
                                 FMT2 = '(I4, 1X, 6I10, F10.2)'
                              OFWFMT = '(I4, 1X, 20(I10, F10.2))'
      ENDIF
*
      IF (ARBFLG) THEN
                                      WRITE(IOUT, 900) NOP
      ENDIF
      DO 100 N = 1, NNODS
                                       LIM1 = PTDWAR(N, 1)
                                       LIM2 = PTDWAR(N, 2)
                                              OJ = 0
*
                 IF (SNBLFG .AND. CNINT(N, NSNBL, SNBLND, OFWID)) THEN
                                                NOFW = 0
                                           OFWFLG = .TRUE.
                                                ELSE
                                           OFWFLG = .FALSE.
                                                ENDIF
*
                                      DO 50 K = LIM1, LIM2
                                         ARC = NDDWAR(K)
                                       IF (ARC .GT. 0) THEN
                                              I = II(ARC)
                                                 ELSE
                                              I = JJ(-ARC)
                                                ENDIF
                                     J = ARCBUD(IABS(ARC), 0)
                                         IF (J .NE. OJ) THEN
                                          IF (K .NE. LIM1) THEN
                                          IF (ARBFLG) THEN
                                            NARC = NARC + 1
                                          IF (XP .GT. 0) THEN
                           WRITE(IOUT, FMT) NARC, NDNAM(I), NDNAM(OJ),
     &                         (BUD(L)/XF, L=1,6), BUD(6)/XF/PERD/CONST
                                                 ELSE
                           WRITE(IOUT, FMT) NARC, NDNAM(I), NDNAM(OJ),
     &                          (BUD(L), L=1,6), BUD(6)/PERD/CONST
                                                ENDIF
                                               ENDIF
*
*-----------------Print arc budget for selected arcs
*
                 IF (CN2INT(I,OJ,NSABL,SABLND(1,1),SABLND(1,2),L)) THEN
                                          OU = SABLND(L, 3)
*
                                           IF (XP .GT. 0) THEN
                                       WRITE(OU, FMT2) NOP,
     &                          (BUD(L)/XF, L=1,6), BUD(6)/XF/PERD/CONST
                                                 ELSE
                                       WRITE(OU, FMT2) NOP,
     &                          (BUD(L), L=1,6), BUD(6)/PERD/CONST
                                                ENDIF
                                               ENDIF
*
*----------------Save downstream release.
*
                                           IF (OFWFLG) THEN
                                             NOFW = NOFW + 1
                                           OFW(NOFW) = BUD(1)
                                               ENDIF
                                                ENDIF
                                                 OJ = J
                                              DO L = 1, 6
                                            BUD(L) = 0
                                                ENDDO
                                              ENDIF
                                            DO L = 1, 6
                         BUD(L) = BUD(L) + ARCBUD(IABS(ARC), L)
                                              ENDDO
                                      IF (K .EQ. LIM2) THEN
                                         IF (ARBFLG) THEN
                                          NARC = NARC + 1
                                      IF (XP .GT. 0) THEN
                             WRITE(IOUT, FMT) NARC, NDNAM(I), NDNAM(OJ),
     &                       (BUD(L)/XF, L = 1, 6), BUD(6)/XF/PERD/CONST
                                               ELSE
                             WRITE(IOUT, FMT) NARC, NDNAM(I), NDNAM(OJ),
     &                       (BUD(L), L = 1, 6), BUD(6)/PERD/CONST
                                              ENDIF
                                                ENDIF
*
*------------------Print arc budget for selected arcs
```

```
*
                 IF (CN2INT(I,OJ,NSABL,SABLND(1,1),SABLND(1,2),L)) THEN
                                     OU = SABLND(L, 3)
                                    IF (XP .GT. 0) THEN
                                     WRITE(OU, FMT2) NOP,
     &                    (BUD(L)/XF, L=1,6), BUD(6)/XF/PERD/CONST
                                           ELSE
                                     WRITE(OU, FMT2) NOP,
     &                          (BUD(L), L=1,6), BUD(6)/PERD/CONST
                                           ENDIF
                                             ENDIF
*
*-----------------Save downstream release.
*
                                        IF (OFWFLG) THEN
                                      NOFW = NOFW + 1
                                    OFW(NOFW) = BUD(1)
                                             ENDIF
                                           ENDIF
 50       CONTINUE
                                     IF (OFWFLG) THEN
                                    OU = SNBLND(OFWID, 3)
                                     IF (XP .GT. 0) THEN
                                     WRITE(OU, OFWFMT) NOP,
     &              (OFW(L)/XF, OFW(L)/XF/PERD/CONST, L = 1, NOFW)
                                            ELSE
                                     WRITE(OU, OFWFMT) NOP,
     &              (OFW(L), OFW(L)/PERD/CONST, L = 1, NOFW)
                                           ENDIF
                                         ENDIF
 100   CONTINUE
       RETURN
 901   FORMAT (I4, 1X, 2A12, T30, 6F10.2)
 900   FORMAT(/,T20, 'Canal water budgets for time step:', I3,
     &        /,T20, '====================================',
     & /,T30,'                                        Surface',
     & /,T30,'           Initial     Canal    evapo-     Final',
     &        '          Outflow',
     & /,T30,'    Inflow   storage   seepage    ration   storage',
     &        '   Outflow    (cubic',
     & /,T30,'    (acre-    (acre-    (acre-    (acre-    (acre-',
     &        '    (acre-  feet per',
     & /,' No. From', T18, 'To',
     &   T30,'     feet)     feet)     feet)     feet)     feet)',
     &        '     feet)   second)',
     & /,' --- ----', T18, '----',
     &   T30,' --------- --------- --------- --------- ---------',
     &        ' ---------  --------')
       END
*=====================================================================
* Name:       setzva
* Purpose:    Get pond characteristic data of elevation, volume,,
*             and surface area from a file.
* Author:     Xiaodong Jian
* Date:       04/07/97
*=====================================================================
       SUBROUTINE SETZVA(NRES, NNODS, NDNAM, NDTYP, LDND,
     &                   FILNAM, NTLS, OU)
       IMPLICIT   NONE
       INTEGER    NRES, NNODS, LDND, NDTYP(LDND), NTLS, OU
       CHARACTER  NDNAM(LDND)*(*), FILNAM*(*)
*
       INTEGER    IU, I
       INTEGER    ZVAMTH, SAVOPT
       COMMON     /ZVAWAY/ ZVAMTH
       COMMON     /SAVOPT/ SAVOPT
*
*
*---------------Open a data file
*
       IU = 9
       CALL IO_OPFIL(IU, 1, FILNAM,
     &               'Enter pond characteristics file: ')
*
*--------------Skip title lines
*
       DO I = 1, NTLS
                                          READ (IU, *)
       ENDDO
*
*-------------Read data index
*
       READ (IU, *, ERR = 99) ZVAMTH
*
*------------Read corresponding data according to data index
*
       IF (ZVAMTH .EQ. 0) THEN
                         CALL SETTAB(NRES, NNODS, NDNAM, NDTYP, LDND,
     &                 IU, OU, SAVOPT)
       ELSE IF (ZVAMTH .EQ. 1) THEN
                         CALL SETEQS(NRES, NNODS, NDNAM, NDTYP, LDND,
     &                 IU, OU, SAVOPT)
       ENDIF
 99    RETURN
```

```
      END
*=====================================================================
* Name:          getzva
* Purpose:       Logial function to interprete Z-V-A for given one of
*                Z, V, or, A for a given node.
* Author:        Xiaodong Jian
* Date:          04/07/97
*=====================================================================
      LOGICAL FUNCTION GETZVA(RESND, ZVAIDX, ZVA, OU)
      IMPLICIT NONE
      INTEGER  RESND, ZVAIDX, OU
      REAL     ZVA(3)
*
      INTEGER  ZVAMTH
      COMMON   /ZVAWAY/ ZVAMTH
*
      IF (ZVAMTH .EQ. 0) THEN
                        CALL GETTAB(GETZVA, RESND, ZVAIDX, ZVA, OU)
      ELSE IF (ZVAMTH .EQ. 1) THEN
                        CALL GETEQS(GETZVA, RESND, ZVAIDX, ZVA, OU)
      ENDIF
      RETURN
      END
*=====================================================================
* Name:          Settab
* Purpose:       Read the Z-V-A table into the program from the file.
* Author:        Xiaodong Jian
* Date:          04/09/97
*=====================================================================
      SUBROUTINE SETTAB(NRES, NNODS, NDNAM, NDTYP, LDND,
     &                  IU, OU, SAVOPT)
      IMPLICIT   NONE
      INTEGER    NRES, NNODS, LDND, NDTYP(LDND), IU, OU, SAVOPT
      CHARACTER  NDNAM(LDND)*(*)
*
*------------Local variables
*
      REAL       ELE, CAP, AREA, MAXV, MINV
      INTEGER    J, K, NWND, NCOL
      CHARACTER  NAME*30, CTERM*100
      LOGICAL    CN
*
*------------Common block for Z-V-A table. This common block must be
*            the same as the subroutine gettab.
*   ptretb    - Pointer of reservoir characteristics table (2 x 100).
*   retb      - Array of reservoir elevation-volume-area table (3 x 5000)
*               1 -- elevation in feet
*               2 -- volume in acre-feet
*               3 -- area in  acre.
*
      INTEGER  PTRETB(2, 100)
      REAL     RETB(3, 5000)
      COMMON   /ZVADAT/ PTRETB, RETB
      PTRETB(2,1) = 1
      NRES = 0
      NNODS = 0
      NWND = 0
      J = 0
*
*----------------Set output file titles
*
      IF (SAVOPT .EQ. 0 .OR. SAVOPT .EQ. 1) THEN
                                             WRITE(OU, 800)
 800     FORMAT(
     &   //, 'Summary of pond elevation-volume-area relations:',
     &    /, '================================================',
     &    /, '                                 Minimum    Maximum',
     &    /, '                         Number of elevation   elevation',
     &    /, '  No  Name                 records    (feet)     (feet)',
     &    /, '  ---  ------------        ---------  ---------  ---------')
      ENDIF
  5   CONTINUE
      READ (IU, '(A)') NAME
      IF (CN ('FINISH', NAME, 1)) THEN
                                         GOTO 90
      ENDIF
*
      MINV =  99999.0
      MAXV = -99999.0
*
*-----------Add a new node.
*
      NRES = NRES + 1
      NNODS = NNODS + 1
      NWND =  NNODS
      CALL STR_CORS(NAME, 1)
      NDNAM(NWND) = NAME
      NDTYP(NWND) = 1
      PTRETB(1, NRES) = NWND
*
*----------Read the characteristics table
*
      NCOL = 0
 15   READ (IU, '(A)') CTERM
```

```
      IF (CTERM .NE. ' ') THEN
                                    NCOL = NCOL + 1
                                     K = J + NCOL
                                     BACKSPACE(IU)
                              READ(IU, *) ELE, CAP, AREA
                                    RETB(1,K) = ELE
                                    RETB(2,K) = CAP
                                    RETB(3,K) = AREA
                            IF (ELE .LT. MINV) MINV = ELE
                            IF (ELE .GT. MAXV) MAXV = ELE
                                       GO TO 15
      END IF
      PTRETB(2, NRES+1) = PTRETB(2, NRES) + NCOL
      J = J + NCOL
*
      IF (SAVOPT .EQ. 0 .OR. SAVOPT .EQ. 1) THEN
                    WRITE (OU, 801) NRES, NAME, NCOL, MINV, MAXV
 801     FORMAT(I5, 2X, A12, 1X, I10, 2F10.2)
      ENDIF
      GO TO 5
 90   CONTINUE
      CLOSE(IU)
      RETURN
      END
*=================================================================
* Name:        gettab
* Purpose:     Get the Z-V-A table to interprete the Z-V-A for a given
*              node.
* Author:      Xiaodong Jian
* Date:        4/9/97
*=================================================================
      SUBROUTINE GETTAB(GETZVA, RESND, ZVAIDX, ZVA, OU)
      IMPLICIT NONE
      INTEGER  RESND,  ZVAIDX, OU
      REAL     ZVA(3)
      LOGICAL  GETZVA
*
*-------------Local variables
*
      REAL       X, X1, Y1, X2, Y2, VAL, VAL1, DIF
      INTEGER    I, N, K, LIM
      CHARACTER ZVANAM(3)*9
      DATA       ZVANAM/ 'ELEVATION', 'VOLUME', 'AREA'/
*
*-------------Common block for Z-V-A table. This common block must be
*             the same as the subroutine settab.
*
      INTEGER    PTRETB(2, 100)
      REAL       RETB(3, 5000)
      CHARACTER  NDNAM(300)*12
      COMMON     /ZVADAT/ PTRETB, RETB
      COMMON     /NDNAME/ NDNAM
*
*--------------Check whether there is Z-V-A for the current node
*
      N = 1
      DO WHILE (PTRETB(1,N) .NE. RESND .AND. PTRETB(1, N) .NE. 0)
                                       N = N + 1
      ENDDO
      IF (PTRETB(1,N) .NE. RESND) THEN
                                     GETZVA = .FALSE.
                                          GOTO 99
      ENDIF
      GETZVA  = .TRUE.
      K = PTRETB(2, N)
      LIM = PTRETB(2, N+1) - 1
*
      VAL = ZVA(ZVAIDX)
      IF (VAL .LT. 0) VAL = 0.0
      IF (RETB(ZVAIDX, K) .GT. VAL .OR. RETB(ZVAIDX, LIM) .LT. VAL) THEN
                                     GETZVA = .FALSE.
                                 PRINT '(A)', CHAR(7)
                            CALL STR_LEN(ZVANAM(ZVAIDX), I)
              WRITE (OU, 901) ZVANAM(ZVAIDX)(1:I), VAL, NDNAM(RESND), RESND,
     &                   ZVANAM(ZVAIDX)(1:I), RETB(ZVAIDX,K),
     &                   ZVANAM(ZVAIDX)(1:I), RETB(ZVAIDX, LIM)
              WRITE (*, 901) ZVANAM(ZVAIDX)(1:I), VAL, NDNAM(RESND), RESND,
     &                   ZVANAM(ZVAIDX)(1:I), RETB(ZVAIDX,K),
     &                   ZVANAM(ZVAIDX)(1:I), RETB(ZVAIDX, LIM)
 901     FORMAT('***ERROR****'
     &     /, ' TRANSFORMING ', A,  ' = ', F10.2,
     &     /, ' TO ITS CORRESPONDING PARAMETERS AT POND: ', A,
     &     /, ' WITH THE NODAL NUMBER: ', I2,
     &     /, ' THE MINIMUM ', A, ' IN Z-V-A TABLE = ', F10.2,
     &     /, ' THE MAXIMUM ', A, ' IN Z-V-A TABLE = ', F10.2)
                                         STOP
*        goto 99
      ENDIF
 10   CONTINUE
      VAL1 = RETB(ZVAIDX,K)
      DIF = VAL1 - VAL
      IF (DIF .LT. 0) THEN
                                  IF (K .GT. LIM) THEN
                                     GETZVA = .FALSE.
```

```
                                        GO TO 99
                                        ENDIF
                                     K = K + 1
                                      GO TO 10
      ELSE IF (DIF .EQ. 0) THEN
                                     DO I = 1, 3
                                  ZVA(I) = RETB(I, K)
                                        ENDDO
                                      GO TO 50
      ELSE
                                        X = VAL
                                X1 = RETB(ZVAIDX,K-1)
                                 X2 = RETB(ZVAIDX,K)
                                     DO I = 1, 3
                                   Y1 = RETB(I, K-1)
                                    Y2 = RETB(I, K)
                     ZVA(I) = Y1 + (Y2 - Y1) * (X - X1) / (X2 - X1)
                                        ENDDO
                                      GO TO 50
      END IF
 99   CONTINUE
 50   RETURN
      END
*=====================================================================
* Name:         seteqs
* Purpose:      Read the Z-V-A regression equation coefficients into the
*               program from the file.
* Author:       Xiaodong Jian
* Date:         04/09/97
*=====================================================================
      SUBROUTINE SETEQS(NRES, NNODS, NDNAM, NDTYP, LDND,
     &                  IU, OU, SAVOPT)
      IMPLICIT   NONE
      INTEGER    NRES, NNODS, LDND, NDTYP(LDND), IU, OU, SAVOPT
      CHARACTER  NDNAM(LDND)*(*)
*
*-------------Local variables
*
      INTEGER    I, N, NWND, NEQS
      CHARACTER  NAME*30, CTERM*100
      LOGICAL    CN
*
*-------------Common block for Z-V-A equations for QNWR.
*               This common block must be the same as the subroutine geteqs
*
      INTEGER  ZVAPTR(2, 100)
      REAL     ZVAEQS(0:3, 10, 100)
      COMMON   /ZVAQNWR/ZVAPTR, ZVAEQS
*
      NRES = 0
      NNODS = 0
      NWND = 0
c      J = 0
*
*-----------------Set output file titles
*
      IF (SAVOPT .EQ. 0 .OR. SAVOPT .EQ. 1) THEN
                                   WRITE(OU, 800)
 800     FORMAT(
     &    //, 'SUMMARY OF POND ELEVATION-CAPACITY-AREA RELATIONS:',
     &     /, '==================================================',
     &     /, 'Z-V-A relations is expresses as regression equation')
      ENDIF
*
*-----------Read pond-node name
*
  5   CONTINUE
      NAME = ' '
      DO WHILE (NAME .EQ. ' ')
                              READ (IU, '(A)', END = 90) NAME
      ENDDO
      IF (CN ('FINISH', NAME, 1)) THEN
                                        GOTO 90
      ENDIF
*
*------------Add a new node.
*
      NRES = NRES + 1
      NNODS = NNODS + 1
      NWND =  NNODS
      CALL STR_CORS(NAME, 1)
      NDNAM(NWND) = NAME
      NDTYP(NWND) = 1
      ZVAPTR(1, NRES) = NWND
      ZVAPTR(2, NRES) = 0
*
*-----------Read the base elevations, and coefficients of regression equations.
*
      NEQS = 0
 15   READ(IU, '(a)') CTERM
      DO WHILE (CTERM .NE. ' ')
                                    NEQS = NEQS + 1
                                     BACKSPACE(IU)
                 READ (IU, *) N, (ZVAEQS(I, NEQS, NRES), I = 0, 3)
```

```
                              READ (IU, '(a)', END= 20) CTERM
      ENDDO
 20   IF (NEQS .NE. 0) THEN
                                   ZVAPTR(2, NRES) = NEQS
      ENDIF
*
*-------------Read another pond node
*
      GOTO 5
*
 90   CONTINUE
      CLOSE(IU)
      RETURN
      END
*=======================================================================
* Name:         geteqs
* Purpose:      Get the equations to interprete the Z~V~A for a given
*               node.
* Author:       Xiaodong Jian
* Date:         4/9/97
*=======================================================================
      SUBROUTINE GETEQS(GETZVA, RESND, ZVAIDX, ZVA, OU)
      IMPLICIT NONE
      INTEGER  RESND,  ZVAIDX, OU
      REAL     ZVA(3)
      LOGICAL  GETZVA
*
*-------------Local variables
*
      INTEGER   N
      INTEGER   RES, NEQS, IFAULT
*
*-------------Common block for Z-V-A equations for QNWR.
*             This common block must be the same as the main program oponds
*             and subroutine getzva
*
      INTEGER   ZVAPTR(2, 100)
      REAL      ZVAEQS(0:3, 10, 100)
      CHARACTER NDNAM(300)*12
      COMMON    /ZVAQNWR/ ZVAPTR, ZVAEQS
      COMMON     /NDNAME/ NDNAM
*
*--------------Check whether there is Z-V-A equations for the current node
*
      N = 1
      DO WHILE (ZVAPTR(1,N) .NE. RESND .AND. ZVAPTR(1, N) .NE. 0)
                                         N = N + 1
      ENDDO
      IF (ZVAPTR(1,N) .EQ. RESND .AND. ZVAPTR(2,N) .GT. 0) THEN
                                          RES = N
                                    NEQS = ZVAPTR(2,N)
      ELSE
                                     GETZVA = .FALSE.
                                          GOTO 99
      ENDIF
*
*---------------Get Z, V, and A
*
      CALL CALZVA(NEQS, ZVAEQS(0, 1, RES), ZVAIDX, ZVA, IFAULT)
      IF (IFAULT .NE. 0) THEN
                                     GETZVA = .FALSE.
                                       WRITE(OU, 900)
900      FORMAT('***Errorr*** in transforming Z-V-A')
      ELSE
                                      GETZVA = .TRUE.
      ENDIF
*
  99  CONTINUE
 50   RETURN
      END
      SUBROUTINE CALZVA(NEQS, A, ZVAIDX, ZVAOUT,IERR)
      IMPLICIT    NONE
      INTEGER     NEQS, ZVAIDX, IERR
      REAL        A(0:3, NEQS)
      REAL        ZVAOUT(3)
*
      REAL        VAL, ZVA(3)
      REAL        Z1, Z2
      INTEGER     I
*
      IF (ZVAIDX .EQ. 1) THEN    ! Z --> V, AND A.
                                    ZVA(1) = ZVAOUT(1)
                              CALL Z2VA(NEQS, A, ZVA, IERR)
      ELSE IF (ZVAIDX .EQ. 2 .OR. ZVAIDX .EQ. 3) THEN
                                   VAL = ZVAOUT(ZVAIDX)
                                       Z1 = A(0, 1)
*
*------------find z2
*
                                   ZVA(1) = A(0, NEQS)
                              CALL Z2VA(NEQS, A, ZVA, IERR)
                            DO WHILE (ZVA(ZVAIDX) .LT.  VAL)
                                     ZVA(1) = ZVA(1) + 1
                                CALL Z2VA(NEQS, A, ZVA, IERR)
```

```
                                            ENDDO
                                        Z2 = ZVA(1)
*
*------------Binary search for the elevation.
*
                              DO WHILE(ABS(Z2 - Z1) .GT. 0.005)
                                  ZVA(1) = 0.5 * (Z1 + Z2)
                                CALL Z2VA(NEQS, A, ZVA, IERR)
                                IF (ZVA(ZVAIDX) .LT. VAL) THEN
                                        Z1 = ZVA(1)
                              ELSE IF (ZVA(ZVAIDX) .GT. VAL) THEN
                                        Z2 = ZVA(1)
                                          ELSE
                                         GOTO 100
                                         ENDIF
                                        ENDDO
      ENDIF
 100  CONTINUE
      DO I = 1, 3
                                   IF (I .NE. ZVAIDX) THEN
                                      ZVAOUT(I) = ZVA(I)
                                        ENDIF
      ENDDO
      RETURN
      END
      SUBROUTINE Z2VA(NEQS, A, ZVA, IERR)
      IMPLICIT   NONE
      INTEGER    NEQS, IERR
      REAL       A(0:3, NEQS), ZVA(3)
*
      REAL       X
      INTEGER    I
*
      IERR = 0
      DO 20 I = NEQS, 1, -1
                                 IF (ZVA(1) .GE. A(0,I)) THEN
                                     X  = ZVA(1) - A(0,I)
                           ZVA(2) = (A(3, I) * X + A(2,I)) * X + A(1,I)
                             ZVA(3) = A(2,I) + X * (A(3,I) + A(3,I))
                                         GOTO 99
                                        ENDIF
 20   CONTINUE
      IERR = 1
 99   RETURN
      END
*==================================================================
* Name:       chdep
* Purpose:    Compute depth of flow using Manning's Equation
* Author:     Xiaodong Jian
* Date:       5/9/96
*==================================================================
      SUBROUTINE CHDEP(ROUGH, DISCH, SLOPE, WIDTH, ML, MR, HMAX, H,
     &                 CONST, TOL, ITMX, IERR)
      IMPLICIT   NONE
      REAL       ROUGH, DISCH, SLOPE, WIDTH, ML, MR, HMAX, H, TOL,
     &           CONST
      INTEGER    IERR, ITMX
*
      REAL       H1, H2, AREA, RAD, C, C1, EPS
      INTEGER    I
      DATA       EPS /0.000001/
*
      IERR = 0
      IF (DISCH .LT. EPS) THEN
                                       H = 0.0
                                      GOTO 999
      ENDIF
*
*-----------------Compute quotient and initialize iteration values
*
      C = ROUGH * DISCH / SQRT(SLOPE) / CONST
      H1 = HMAX
      H = H1
      H2 = 0.0
*
*-----------------Start Iterations
*
      IERR = 0
      DO 100 I = 1, ITMX
*
*            compue area and hydraulic radius
*
                         AREA = H * (WIDTH + 0.5 * H * (ML + MR))
                RAD = AREA /(WIDTH + H*(SQRT(1.0+ML*ML) + SQRT(1.0+MR*MR)))
*
*             Compute approximate quotient
*
                                  C1 = AREA * RAD**(2.0/3.0)
*
*             Check convergence
*
                              IF (ABS((C1-C)/C) .LE. TOL) THEN
                                        GOTO 999
                                        ENDIF
```

```
*
*             if No.
*
                                            IF (C1 .GT. C) THEN
                                            H = H2 + (H - H2) / 2.0
                                            ELSE
                                            IF (H+EPS .GE. H1) THEN
                                            IERR = 1
                                            H1 = H1 * 2
                                            ELSE
                                            H2 = H
                                            H = H2 + (H1 - H2) / 2.0
                                            ENDIF
                                            ENDIF
 100  CONTINUE
      IERR = 2
 999  RETURN
      END
*======================================================================
* Name:       fil_head
* Purpose:    Open a time series file and read headers for nodal data.
* Author:     Xiaodong Jian
*======================================================================
      SUBROUTINE FIL_HEAD(NNODS, NDNAM, LDND, NDTND, DTND, LDDTND,
     &              DTTYPE, UNITNM, LDUNIT, DTFLAG, DTFIL, FILNAM, NTLS,
     &              IU, OU, CLASS, CTERM, COLSTR, LDCOL)
      IMPLICIT    NONE
      INTEGER     NNODS, LDND, NDTND, LDDTND, DTND(LDDTND, 3), DTTYPE,
     &            LDUNIT
      CHARACTER*(*)  NDNAM(LDND), UNITNM(0:LDUNIT, 1), FILNAM, CLASS
      INTEGER     NTLS, IU, OU
      LOGICAL     DTFLAG, DTFIL
*
      INTEGER     LDCOL
      CHARACTER   CTERM*(*), COLSTR(LDCOL)*(*), MESS*50
*
      INTEGER     DTUNIT
      INTEGER     I, J, K, L, N, ND, NREC, NNDS, SL
      LOGICAL     ERR, CN, CNINT, DUMMY, ISNUM
      INTEGER     SAVOPT
      COMMON      /SAVOPT/ SAVOPT
*
      DTTYPE = 1
      DUMMY = .FALSE.
      MESS = 'ENTER DATA FILE FOR '//CLASS
      DTFIL = .FALSE.
      IF (FILNAM .EQ. ' ') THEN
                                            GOTO 999
      ENDIF
      CALL IO_OPFIL(IU, 1, FILNAM, MESS)
*
*-------------Skip title lines
*
      DO I = 1, NTLS
                                    READ (IU, *, END = 999)
      ENDDO
*
*-------------Read data unit code and type
*
      READ (IU, '(A)', END = 999) CTERM
                  CALL STR_DIVD(CTERM, I, COLSTR, LDCOL, 0, ' ,')
                          READ (COLSTR(1), '(I3)') DTUNIT
                            IF (ISNUM(COLSTR(2))) THEN
                           READ (COLSTR(2), '(I3)') DTTYPE
                                            ENDIF
*
*-------------Read nodal names
*
      NNDS = 0
      READ (IU, '(A)', END = 999) CTERM
      CALL STR_DIVD(CTERM, NNDS, COLSTR, LDCOL, 0, ' ,')
*
      N = 0
      DO 30 I = 2, NNDS
                         IF (CN(COLSTR(I), 'DEFAULT', 1)) THEN
*
*-----------Seasonal data are no longer valid.
*
                                            DUMMY = .TRUE.
                                            IF (NNDS .EQ. 2) THEN
                                            DO J = 1, NNODS
                                            DTND(J, 1) = J
                                            DTND(J, 2) = DTUNIT
                                            DTND(J, 3) = -J
                                            ENDDO
                                            ELSE
*-------------
                                            IF (NDTND .GT. 0) THEN
                                            DO 25 J = 1, NNDS - 2
                                            DO L = 1, NDTND + N
                                            IF (DTND(L, 3) .EQ. J) THEN
                                            DO K = 1, 3
                                            DTND(J, K) = DTND(L, K)
                                            ENDDO
```

```
                                        GOTO 25
                                       ENDIF
                                         ENDDO
 25                CONTINUE
                                 DO J = NNDS-2+1, NNODS
                                       DO K = 1, 3
                                    DTND(J,K) = 0
                                         ENDDO
                                      ENDDO
                                          ENDIF
*--------------
                                       L = NNDS - 2
                                      DO J = 1, NNODS
                         IF (.NOT. CNINT(J, NNDS-2, DTND(1,1), K)) THEN
                                        L = L + 1
                                     DTND(L, 1) = J
                                   DTND(L, 2) = DTUNIT
                                     DTND(L, 3) = -L
                                      ENDIF
                                          ENDDO
                                       ENDIF
                                     NDTND = NNODS
                                       GOTO 50
                                       ELSE
                          CALL NAMNUM(LDND, NDNAM, COLSTR(I), ND, 0, ERR)
                                     IF (ERR) THEN
                                WRITE(*, 901) COLSTR(I), FILNAM
 901            FORMAT( '***ERROR***NODAL NAME: ', A12,
     &                 ' IN THE FILE: ',A,
     &            /, ' NOT FOUND IN THE NETWORK CONFIGURATION.')
                                       CALL EXIT
                                       ENDIF
                                  IF (NDTND .GT. 0) THEN
                                      DO J = 1, NDTND
                              IF (DTND(J, 1) .EQ. ND) THEN
                                       GOTO 20
                                      ENDIF
                                          ENDDO
                                        N = N + 1
                                     J = NDTND + N
 20             CONTINUE
                                        ELSE
                                       J = I - 1
                                        ENDIF
                                    DTND(J, 1) = ND
                                  DTND(J, 2) = DTUNIT
                                   DTND(J, 3) = I - 1
                                        ENDIF
 30    CONTINUE
       NNDS = NNDS - 1
       IF (NDTND .GT. 0) THEN
                                  NDTND = NDTND + N
       ELSE
                                    NDTND = NNDS
       ENDIF
*
 50    CONTINUE
       DTFLAG = .TRUE.
       DTFIL = .TRUE.
*
*------------Find the number of data records in the data file
*
       CALL NORECS(IU, NREC)
                                     REWIND(IU)
                                  DO I = 1, NTLS+2
                                      READ(IU, *)
                                       ENDDO
*
*------------Save the data file information.
*
       IF (SAVOPT .EQ. 0 .OR. SAVOPT .EQ. 1) THEN
                                 CALL STR_LEN(CLASS, J)
                      WRITE (OU, 800) CLASS, ('=', I = 1, J + 11)
 800      FORMAT (
     &      //, 'Summary for ', A, ' data file:'
     &       /, '===========', 100A)
                                     IF (DUMMY) THEN
                WRITE (OU, 805) FILNAM, UNITNM(DTUNIT, DTTYPE), NDTND, NREC
                WRITE (OU, 806) (NDNAM(DTND(I, 1)), I=1,NNDS-2), 'DEFAULT'
                                       ELSE
                WRITE (OU, 805) FILNAM, UNITNM(DTUNIT, DTTYPE), NNDS, NREC
                                     CTERM = ' '
                                        J = 0
                                        L = 1
                                    DO I = 1, NNDS
                      IF (CNINT(I, NDTND, DTND(1, 3), K)) THEN
                                      J = J + 1
                 WRITE(CTERM((J-1)*10+1:J*10),'(A10)') NDNAM(DTND(K,1))
                                        ENDIF
                                   IF (J .EQ. 5) THEN
                                  IF (L .EQ. 1) THEN
                                  WRITE(OU, 807) CTERM
                                        ELSE
                                  WRITE(OU, 808) CTERM
```

```
                                          ENDIF
                                       CTERM = ' '
                                          J = 0
                                        L = L + 1
                                             ENDIF
                                           ENDDO
                              IF (SL(CTERM) .GT. 0 ) THEN
                                    IF (L .EQ. 1) THEN
                                WRITE (OU, 807) CTERM
                                             ELSE
                                WRITE (OU, 808) CTERM
                                             ENDIF
                                         CTERM = ' '
                                           ENDIF
                                         ENDIF
      ENDIF
*
 999  RETURN
 805  FORMAT (
     &     /, 10X, '          File name: ', A
     &     /, 10X, '          Data unit: ', A
     &     /, 10X, '    Number of nodes: ', I4,
     &     /, 10X, '  Number of records: ', I4)
 806  FORMAT( 10X, ' List of nodal name: ', 5A10, 10(/, 30X, 5A10))
 807  FORMAT( 10X, 'List of nodal names: ', A50)
 808  FORMAT( 31X, A50)
      END
*=====================================================================
      SUBROUTINE NORECS(IU, NREC)
      IMPLICIT   NONE
      INTEGER    IU, NREC, I
*
      NREC = 0
 5    READ (IU, *, END = 10)
      NREC = NREC + 1
      GOTO 5
 10   DO I = 1, NREC + 1
                                     BACKSPACE(IU)
      ENDDO
      RETURN
      END
*====================================================================
* Name:    Isnum
* Purpose: Check whether a string is a float or integer number.
*====================================================================
      LOGICAL FUNCTION ISNUM(STR)
      IMPLICIT  NONE
      CHARACTER STR*(*)
*
      INTEGER   I, L, ICD
      LOGICAL   DECFLG, SIGNFG
      ISNUM = .FALSE.
      DECFLG = .FALSE.
      SIGNFG = .FALSE.
      IF (STR .EQ. ' ') GOTO 999
      L = LEN(STR)
*
      DO I = 1, L
                              IF (STR(I:I) .NE. ' ') THEN
                                  ICD = ICHAR(STR(I:I))
               IF (ICD .GE. 48 .AND. ICD .LE. 57) THEN      ! DIGITS 0 - 9.
                                          CONTINUE
             ELSE IF (ICD .EQ. 46 .AND. .NOT. DECFLG) THEN  ! DECIMAL POINT.
                                       DECFLG = .TRUE.
                        ELSE IF ((ICD .EQ. 43 .OR. ICD .EQ. 45)
     &          .AND. .NOT. SIGNFG) THEN  ! + OR -
                                       SIGNFG = .TRUE.
                                           ELSE
                                          GOTO 999
                                          ENDIF
                                        ENDIF
      ENDDO
      ISNUM = .TRUE.
999   RETURN
      END
*=================================================================
      LOGICAL FUNCTION INLIST(ND, NLST, LSTND, LOC)
      IMPLICIT  NONE
      INTEGER   ND, NLST, LSTND(NLST), LOC
      INTEGER   I
      INLIST = .FALSE.
      DO I = 1, NLST
                               IF (ND .EQ. LSTND(I)) THEN
                                         LOC = I
                                    INLIST = .TRUE.
                                       GOTO 999
                                       ENDIF
      ENDDO
 999  RETURN
      END
*=====================================================================
* Name:        gw_dat
* Purpose:     Read groundwater altitude or flux data.
* Author:      Xiaodong Jian
```

```
*=====================================================================
      SUBROUTINE GW_DAT(NGWND, GWND, GWLVL, LDGWND, GWTYPE, IU,
     &                  CTERM, COLSTR, LDCOL)
      IMPLICIT   NONE
      INTEGER    NGWND, LDGWND, GWTYPE, IU
      INTEGER    GWND(LDGWND, 3)
      REAL       GWLVL(LDGWND)
*
      INTEGER    LDCOL
      CHARACTER  CTERM*(*), COLSTR(LDCOL)*(*)
*
      INTEGER    I, J, NNDS
      REAL       UC(0:2, 2)
*
*------------Check groundwater data type
*
      IF (GWTYPE .LT. 1 .OR. GWTYPE .GT. 2) THEN
                PRINT *, '***ERROR*** INVALID GROUDWATER DATA TYPE: ', GWTYPE
                                    STOP
                                  ENDIF
*
*------------Unit coversion factor: 0,1 ft --> ft; 1,1 in --> ft; 2,1 m  --> ft
*               0,2 aft/d -> aft/d; 1,2 cfs->aft/d; 2,2 cfd->aft/d
*
      UC(0,1) = 1.0
      UC(1,1) = 1.0 / 12.0
      UC(2,1) = 3.281
      UC(0,2) = 1.0                        ! AC-FT/D --> AC-FT/D
      UC(1,2) = 86400.0 / 43560.0          ! CFS     --> AC-FT/D
      UC(2,2)   = 1.0 / 43560.0            ! CFD     --> AC-FT/D
*----------------Read time-dependent ground-levels
*
      READ (IU, '(A)', END = 99) CTERM
      CALL  STR_DIVD(CTERM, NNDS, COLSTR, LDCOL, 0, ' ,')
*
      DO I = 1, NGWND
                           IF (GWND(I,3) .GT. 0) THEN
                                 J = GWND(I,3) + 1
                         READ (COLSTR(J), '(F15.0)') GWLVL(I)
                          ELSE IF(GWND(I, 3) .LT. 0) THEN
                        READ (COLSTR(NNDS), '(F15.0)') GWLVL(I)
                                     ENDIF
                  GWLVL(I) = GWLVL(I) * UC(GWND(I,2), GWTYPE)
      ENDDO
 99   RETURN
      END
*=====================================================================
* Name:       reslos
* Purpose:    Calculate reservoir seepage loss.
* Author:     Xiaodong Jian
*=====================================================================
      SUBROUTINE RESLOS(NARC, II, JJ, LO, HI, COST, ARTYP, LDARC,
     &                  NRES, PTRES, INST, RESDAT, LDRES,
     &                  NGWND, GWND, GWLVL, LDGWND, GWTYPE,
     &                  NDXAR, LDXAR, SKSC, PERD, XF)
      IMPLICIT   NONE
      INTEGER    NARC, LDARC,
     &           II(LDARC), JJ(LDARC), LO(LDARC), HI(LDARC),
     &           COST(LDARC), ARTYP(LDARC)
      INTEGER    NRES, LDRES, PTRES(LDRES)
      REAL       INST(LDRES), RESDAT(LDRES, 0:2)
      INTEGER    NGWND, LDGWND, GWND(LDGWND,3), GWTYPE
      REAL       GWLVL(LDGWND)
      INTEGER    LDXAR, NDXAR(LDXAR, 6)
      INTEGER    SKSC
      REAL       PERD, XF
*
      INTEGER    ND, RES, NV, GWNO
      REAL       HSW, HGW, AREA, KY, DL, ZVA(3), EPS, RTERM
      LOGICAL    GETZVA, GWFLG, INLIST
      CHARACTER  NDNAM(300)*12
      COMMON     /NDNAME/ NDNAM
      DATA       EPS /0.000001/
*
      DO 100 RES = 1, NRES
                              ND = PTRES(RES)
                            KY = RESDAT(RES, 1)
                            DL = RESDAT(RES, 2)
                 IF (KY .LT. EPS .AND. GWTYPE .EQ. 1) GOTO 100
*
*------------Is the current node a ground-water node, i.e.
*            the node with ground-water data.
*
              HGW = RESDAT(RES, 0)      !BOTTOM ELEVATION OF RESERVOIR.
                    IF (INLIST(ND, NGWND, GWND, GWNO)) THEN
                               GWFLG = .TRUE.
                IF (GWTYPE .EQ. 1 .AND. GWLVL(GWNO) .LT. HGW) THEN
                                  CONTINUE
                                   ELSE
                               HGW = GWLVL(GWNO)
                                  ENDIF
                                 ELSE
                              GWFLG = .FALSE.
                                 ENDIF
```

```
*
*----------------Calculate seepage from a pond.
*
                ZVA(2) = INST(RES)                   !INITIAL POND STORAGE
                           IF (GWFLG .AND. GWTYPE .EQ. 2) THEN
*
*-------------If required ground-water seepage is greater than
*              the available water in a pond. It is assumed that
*              the maximum ground-water seepage is 1/3 of the available
*              water.
*
                                      RTERM = HGW * PERD
                                   NV = NINT(HGW * PERD * XF)
                         IF (HGW .GT. 0.0 .AND. RTERM .GT. ZVA(2)) THEN
                                   NV = NINT(ZVA(2) / 3.0 * XF)
                                              ELSE
                                    NV = NINT(HGW * PERD * XF)
                                             ENDIF
                                            ELSE
*
*---------------Find pond water-surface elevation and area
*
                                IF (GETZVA(ND, 2, ZVA, 26)) THEN
                                           HSW = ZVA(1)
                                          AREA = ZVA(3)
                                             ENDIF
                         RTERM = KY * (HSW - HGW) / DL * AREA * PERD
                      NV = NINT(KY * (HSW - HGW) / DL * AREA * PERD * XF)
*
*-------------If calculated  ground-water seepage is greater than
*              the available water in a pond. It is assumed that
*              the maximum ground-water seepage is 1/3 of the available
*              water.
*
                                  IF (RTERM .GT. ZVA(2)) THEN
                                   NV = NINT(ZVA(2) / 3.0 * XF)
                                              ELSE
                                      NV = NINT(RTERM * XF)
                                             ENDIF
                                            ENDIF
*
*------------Create the ground-water arc
*
                      IF (NV .GT. 0) THEN       ! LOSS WATER TO AQUIFER
                        CALL ARCVAL(NARC, II, JJ, LO, HI, COST, ARTYP,
     &                 LDARC, ND, SKSC, NV, NV, 0, 4)
                                     NDXAR(ND, 4) = NARC
                     ELSE IF (NV .LT. 0) THEN ! GAIN WATER FROM AQUIFER.
                        CALL ARCVAL(NARC, II, JJ, LO, HI, COST, ARTYP,
     &                 LDARC, SKSC, ND, -NV, -NV, 0, -4)
                                    NDXAR(ND, 4) = -NARC
                                            ENDIF
 100  CONTINUE
      RETURN
      END
*=====================================================================
* Name:        KLTR
* Purpose:     Out-of-kilter algorithm to find the optimal flow in a net-
*              work.
*=====================================================================
      SUBROUTINE KLTR (I, J, HI, LO, COST, FLOW, NNODS, NARCS,
     &                 DEBUG, KARC, IFAULT)
      IMPLICIT    NONE
      INTEGER     NARCS, NNODS, I(NARCS), J(NARCS), KARC, IFAULT
      INTEGER     HI(NARCS), LO(NARCS), COST(NARCS), FLOW(NARCS)
      LOGICAL     DEBUG
*
      INTEGER     MXNODS
      PARAMETER   (MXNODS = 500)
      INTEGER     PI(MXNODS), NA(MXNODS), NB(MXNODS)
      INTEGER     A, AOK, COK, C, DEL, E, EPS, SRC, SNK
*
      INTEGER     M, K, INF, IA, JA, N, LAB, CCOK, NI, NJ
*
      KARC = 0
      IFAULT = 0
      DEBUG = .FALSE.
*
*--------------Check nodal numbers which cannot be negative, zero, or
*              greater than MXNODS.
*
      DO K = 1, NARCS
                           IF (I(K) .LE. 0 .OR. J(K) .LE. 0) THEN
                                         PRINT *, CHAR(7)
                                     WRITE(*, 805) I(K), J(K)
 805        FORMAT('**ERROR** NODAL NUMBER CANNOT BE NEGATIVE OR LESS',
     &             'THAN ZERO.'
     &           /, 'NODAL NUMBERS ARE ', I4,',', I4)
                                          STOP !CALL EXIT
                                            ENDIF
                      IF (I(K) .GT. MXNODS .OR. J(K) .GT. MXNODS) THEN
                                         PRINT *, CHAR(7)
                                  WRITE(*, 806) MXNODS, I(K), J(K)
 806        FORMAT('**ERROR** NODAL NUMBER CANNOT BE GREATER THAN', I4,
```

```
&         /,    'NODAL NUMBERS ARE ', I4,',', I4)
                                        STOP !CALL EXIT
                                           ENDIF
      ENDDO
      DO 1 A = 1, NARCS
                                         FLOW (A) = 0
  1   CONTINUE
      DO 2 M = 0, NNODS
                                         PI (M) = 0
  2   CONTINUE
      DO 3 K = 0, 100
                                         NA (K) = 0
  3   CONTINUE
      DO 4 A = 1, NARCS
                                 IF (LO (A) .GT. HI (A)) THEN
                                          IFAULT = 1
                  PRINT '(a, i5)', '**** ERROR *** Lower bound is higher than'
     &                // ' the upper bound for arc: ', A
                       PRINT '(a, i10)', '    The lower bound is ', LO(A)
                       PRINT '(a, i10)', '    The upper bound is ', HI(A)
                                           END IF
  4   CONTINUE
      IF (IFAULT .NE. 0) THEN
                                           RETURN
      ENDIF
      INF = 999999
      AOK = 0
 10   CONTINUE
      DO 20 A = 1, NARCS
                                         IA = I(A)
                                         JA = J(A)
                                 C = COST(A) + PI(IA) - PI(JA)
                  IF ((FLOW(A) .LT. LO(A)) .OR. (C .LT. 0 .AND. FLOW(A) .LT.
     &           HI(A))) THEN
                                          SRC = J(A)
                                          SNK = I(A)
                                            E = +1
                                          GO TO 30
                ELSE IF ((FLOW(A) .GT. HI(A)) .OR. (C .GT. 0 .AND. FLOW(A) .GT.
     &                LO(A))) THEN
                                          SRC = I(A)
                                          SNK = J(A)
                                            E = -1
                                          GO TO 30
                                          END IF
 20   CONTINUE
      GO TO 100
 30   IF ((A .EQ. AOK) .AND. (NA(SRC) .NE. 0)) GO TO 25
      AOK = A
      DO 5 N = 1, NNODS
                                         NA(N) = 0
                                         NB(N) = 0
  5   CONTINUE
      NA(SRC) = IABS(SNK) * E
      NB(SRC) = IABS(AOK) * E
 25   COK = C
 40   LAB = 0
      DO 35 A = 1, NARCS
                                         IA = I(A)
                                         JA = J(A)
*     !DETERMINE THE CANDIDATE ARC FOR THE TREE
                  IF ((NA(IA) .EQ. 0 .AND. NA(JA) .EQ. 0) .OR. (NA(IA) .NE. 0
     &           .AND. NA(JA) .NE. 0)) GO TO 35
                                 C = COST(A) + PI(IA) - PI(JA)
                                   IF (NA(IA) .NE. 0) THEN
                                  IF (FLOW(A) .GE. HI(A). OR.
     &              (FLOW(A) .GE. LO(A) .AND. C .GT. 0)) GO TO 35
                                        NA(JA) = I(A)
                                          NB(JA) = A
                                 ELSE IF (NA(IA) .EQ. 0) THEN
                                  IF (FLOW(A) .LE. LO(A) .OR.
     &              (FLOW(A). LE. HI(A) .AND. C .LT. 0)) GO TO 35
                                           IA = I(A)
                                        NA(IA) = -J(A)
                                          NB(IA) = -A
                                           END IF
                                          LAB = 1
                                 IF (NA(SNK) .NE. 0) GO TO 50
 35   CONTINUE
      IF (LAB .NE. 0) GO TO 40
      DEL = INF
      DO 45 A = 1, NARCS
                                         IA = I(A)
                                         JA = J(A)
                  IF ((NA(IA) .EQ. 0 .AND. NA(JA) .EQ. 0) .OR. (NA(IA) .NE. 0
     &           .AND. NA(JA) .NE. 0)) GO TO 45
                                 C = COST(A) + PI(IA) - PI(JA)
*
                 IF (NA(JA) .EQ. 0 .AND. FLOW(A) .LT. HI(A)) DEL = MIN0(DEL,C)
                IF (NA(JA) .NE. 0 .AND. FLOW(A) .GT. LO(A)) DEL = MIN0(DEL,-C)
*
 45   CONTINUE
      CCOK = COK
      IF (DEL .EQ. INF .AND. (FLOW(AOK) .EQ. HI(AOK) .OR. FLOW(AOK) .EQ.
```

```
     &      LO(AOK))) DEL = ABS(CCOK)
      IF (DEL .EQ. INF) THEN
                                          DEBUG = .TRUE.
                                          KARC = AOK * E
                                             GO TO 99
      END IF
      DO 6 N = 1, NNODS
                              IF (NA(N) .EQ. 0) PI(N) = PI(N) + DEL
 6    CONTINUE
      GO TO 10
 50   EPS = INF
      NI = SRC
 60   NJ = IABS(NA(NI))
      A = IABS(NB(NI))
      C = COST(A) + PI(NI) - PI(NJ)
      IF (NB(NI) .GT. 0) THEN
                                  IF (FLOW(A) .LT. LO(A)) THEN
                                 EPS = MINO(EPS, LO(A) - FLOW(A))
                                             END IF
                         IF (C .LE. 0 .AND. FLOW(A) .LT. HI(A)) THEN
                                 EPS = MINO(EPS, HI(A) - FLOW(A))
                                             END IF
      ELSE IF (NB(NI) .LT. 0) THEN
                                  IF (FLOW(A) .GT. HI(A)) THEN
                                 EPS = MINO(EPS, FLOW(A) - HI(A))
                                             END IF
                         IF (C .GE. 0 .AND. FLOW(A) .GT. LO(A)) THEN
                                 EPS = MINO(EPS, FLOW(A) - LO(A))
                                             END IF
      END IF
      NI = NJ
      IF (NI .NE. SRC) GO TO 60
 70   NJ = IABS (NA(NI))
      A = IABS(NB(NI))
      FLOW(A) = FLOW(A) + ISIGN(EPS, NB(NI))
      NI = NJ
      IF (NI .NE. SRC) GO TO 70
      AOK = 0
      GO TO 10
 99   CONTINUE
 100  IF (.NOT. DEBUG) GO TO 90
 90   RETURN
 800  FORMAT ('SOLUTION INFEASIBLE')
      END
*=====================================================================
* Name:         PRTINF
* Purpose:      Print the basic network information.
* Author:       Xiaodong Jian
*=====================================================================
      SUBROUTINE PRTINF (NDNAM, NDTYP, NDSEQ, NRES, TYPE, MXND,
     &             MXARC, MXR,
     &             II, JJ, HI, LO, COST, ARTYP, PTRE, REAR,
     &             NNODS, PTDWAR, NDDWAR,
     &             PERD, NARCS, C, XF, IOUT1)
      IMPLICIT    NONE
      INTEGER     MXARC, MXR, MXND
      INTEGER     II(MXARC), JJ(MXARC), HI(MXARC), LO(MXARC),COST(MXARC),
     &            ARTYP(MXARC), PTRE(MXR), REAR(MXR),
     &            PTDWAR(MXND, 2), NDDWAR(MXARC)
      INTEGER     NNODS, NARCS, IOUT1
      REAL        C, XF, PERD, X
      CHARACTER   NDNAM(MXND)*(*)
      INTEGER     NDTYP(MXND), NDSEQ(MXND)
      INTEGER     NRES, TYPE, N, I, J, RES, LIM1, LIM2, L
      INTEGER     ND, OND
      INTEGER     ARC, ZONE
      CHARACTER   TYP*4
      CHARACTER*10  NCST, UCST, LCST, ONCST
*
      X = PERD * C * XF
      IF (TYPE .EQ. 0 .OR. TYPE .EQ. 1) THEN
*
*         WRITE THE NODE NAME AND TYPE
*
                                         WRITE (IOUT1,21)
 21       FORMAT (
     &        //, 'Node name and type of basic network',
     &         /, '===================================',
     &         /,  10X,  '   Node    Node        Node',
     &         /,  10X,  ' number    name        type',
     *         /,  10X,  ' ------    ------      ------')
                                        DO 2 N = 1, NNODS
                                     IF (NDTYP(N) .EQ. 1) THEN
                                WRITE (IOUT1, 23) N, NDNAM(N), 'POND'
 23             FORMAT (10X, I7, 3X, A12, A)
                                              ELSE
                               WRITE (IOUT1,23) N, NDNAM(N), 'GENERAL'
                                             END IF
 2        CONTINUE
      END IF
*
      IF (TYPE .EQ. 0 .OR. TYPE .EQ. 2) THEN
*
*         WRITE THE ARCS AND BOUNDS OF A NETWORK
```

```
*
                                   WRITE (IOUT1, 27)
 27      FORMAT (
     &     //,  'Basic network',
     &      /,  '=============',
     &      /, T45, '      Flow boundary ',
     &      /, T45, ' -------------------',
     &      /, T45, '     Lower     Upper                  ',
     &      /, T45, '    (cubic    (cubic ',
     &      /, 10X, '    From-          To-',
     &         T45, '  feet per  feet per     Penalty            ',
     &      /, 10X, 'Arc  node          node',
     &         T45, '   second)   second)   coefficient      Type',
     &      /, 10X, '---  ----          ---- ',
     &         T45, ' --------- ---------   -----------      -----')
                                   DO 3 J = 1, NARCS
                         WRITE (IOUT1, 29) J, NDNAM(II(J)), NDNAM(JJ(J)),
     &                        LO(J)/X, HI(J)/X, COST(J), ARTYP(J)
 29        FORMAT (10X, I3, 2X, T16, A, T28, A, T45, 2F10.2,I15,I10)
 3       CONTINUE
      END IF
*
      IF (TYPE .EQ. 0 .OR. TYPE .EQ. 3) THEN
*
*        WRITE THE PENALTY COEFFICIENTS OF RESERVOIR ZONES
*
                                   WRITE (IOUT1, 31)
 31      FORMAT (
     &   //, 'Pond-zoning penalty coefficients',
     &    /, '================================')
*
*             Find the maximum number of zones of a pond
*
                                       J = 0
                                  DO 36 RES = 1, NRES
                              L = PTRE(RES+1) - PTRE(RES)
                                   IF (L .GT. J) J = L
 36      CONTINUE
                       WRITE (IOUT1, 940) 'Name', ('Zone ', I, I = 1, J)
 940     FORMAT(10X, A, T31, 10(4X, A4,I2))
                           WRITE (IOUT1, 942) ('------', I = 1, J+1)
 942     FORMAT(10X, A4, T31, 10(4X, A6))
                                  DO 39 RES = 1, NRES
                                    DO 4 N = 1, NNODS
                       IF (NDTYP(N) .EQ. 1 .AND. NDSEQ(N) .EQ. RES) GO TO 33
 4          CONTINUE
 33         CONTINUE
                                    LIM1 = PTRE(RES)
                                  LIM2 = PTRE(RES+1) - 1
                               WRITE (IOUT1, 945) NDNAM(N),
     &                       (COST (IABS (REAR(L)) ), L= LIM1, LIM2)
 945        FORMAT (10X, A, T31, 4I10/)
 39      CONTINUE
*
*        WRITE OUT THE PENALTY COEFFICIENTS OF CHANNELS
*
                                   WRITE (IOUT1, 950)
 950     FORMAT(
     &   //, 'Flow-zoning penalty coefficients',
     &    /, '================================',
     &    /, 10X, T35, '    Normal      Lower       Upper',
     &    /, 10X, 'From-', T23, 'To',
     &            T35, '     flow       flow       flow',
     &    /, 10X, 'node', T23, 'node',
     &            T35, '     zone       zone       zone',
     &    /, 10X, '----', T23, '----',
     &            T35, '    ------     ------     ------')
                                   DO 49 N = 1, NNODS
                                    LIM1 = PTDWAR(N, 1)
                                    LIM2 = PTDWAR(N, 2)
                                          OND = 0
                                       ONCST = ' '
                                        NCST = ' '
                                        LCST = ' '
                                        UCST = ' '
                                    DO I = LIM1, LIM2
                                       ARC = NDDWAR(I)
                              WRITE(TYP, '(I4)') ARTYP(ABS(ARC))
*
*-----------------downstream node
*
                                IF (TYP(3:3) .EQ. '1') THEN
                                 IF (ARC .GT. 0) THEN
                                      ND = JJ(ARC+1)
                                          ELSE
                                      ND = II(-ARC+1)
                                         ENDIF
                                             ELSE
                                 IF (ARC .GT. 0) THEN
                                        ND = JJ(ARC)
                                          ELSE
                                       ND = II(-ARC)
                                         ENDIF
                                            ENDIF
```

```
*
*-----------------flow zone
*
                                   READ(TYP(4:4), '(I1)') ZONE
                                      IF (ZONE .EQ. 0) THEN
                               WRITE(NCST, '(I10)') COST(ARC)
                                IF (I .EQ. LIM1) ONCST = NCST
                                    ELSE IF (ARC .GT. 0) THEN
                               WRITE(UCST, '(I10)') COST(ARC)
                                    ELSE IF (ARC .LT. 0) THEN
                              WRITE(LCST, '(I10)') COST(-ARC)
                                              ENDIF
                                    IF (I .EQ. LIM1) OND = ND
                                      IF (ND .NE. OND) THEN
                          WRITE (IOUT1, 43) NDNAM(N), NDNAM(OND),
     &                              ONCST, LCST, UCST
 43             FORMAT (10X, 2A12, T35, 3A10)
                                          OND = ND
                                        ONCST = NCST
                                         LCST = ' '
                                         UCST = ' '
                                              ENDIF
                                            ENDDO
                           WRITE (IOUT1, 43) NDNAM(N), NDNAM(ND),
     &                       ONCST, LCST, UCST
 49      CONTINUE
      END IF
*
      RETURN
      END
*======================================================================
* Name:       PRTTL
* Purpose:    Print output titles for hydrographs.
*======================================================================
      SUBROUTINE PRTTL (NDNAM, NDTYP, NDSEQ,  SYSNAM, STMH, STYR,
     &                  NNODS, NRES, INST, RESNAM,
     &                  IOUT1, IOUT2, MXND, MXR, PRTFLW)
      IMPLICIT   NONE
      INTEGER    MXND, MXR, NNODS, NRES, IOUT1, IOUT2
      CHARACTER  NDNAM(MXND)*(*)
      INTEGER    NDTYP(MXND), NDSEQ(MXND)
      CHARACTER*(*) STMH, STYR, SYSNAM * 40
      INTEGER    I, J
      CHARACTER  RESNAM(MXR)*(*)
      REAL       INST(MXR)
      LOGICAL    PRTFLW
      IF (.NOT. PRTFLW) THEN
                          WRITE (IOUT1, 1001) SYSNAM, STMH, STYR
                       WRITE (IOUT1, 1002) (NDNAM(J), J = 1, NNODS)
                                   WRITE (IOUT1, *)
      ENDIF
*
      DO 11 I = 1, NRES
                                     DO 9 J = 1, NNODS
                     IF ((NDTYP(J) .EQ. 1) .AND. (NDSEQ(J) .EQ. I)) THEN
                                       RESNAM(I) = NDNAM(J)
                                            END IF
9        CONTINUE
11    CONTINUE
      IF (.NOT. PRTFLW) THEN
      WRITE (IOUT2, 1011) SYSNAM, STMH, STYR
      WRITE (IOUT2, 1012) (RESNAM(J), J = 1, NRES)
      WRITE (IOUT2, 1013) ' TIME ', ('    INFLOW    VOLUME   OUTFLOW',
     &                              I = 1, NRES)
      WRITE (IOUT2, 1013) ' STEP ', ('    (A-FT)    (A-FT)    (A-FT) ',
     $                              I = 1, NRES)
      WRITE (IOUT2, 19)0,(NINT(INST(I)), I = 1, NRES)
 19   FORMAT (/, I3, 2X, 5(10X, I10,10X))
      ENDIF
      RETURN
 1001 FORMAT (///, 20X, 'DOWNSTREAM FLOW FROM EACH NODE IN ', A40, /,
     &              20X, ' (FROM ', A4, A5, ')', /,
     &              20X, ' (UNIT: ACRE-FEET)', //)
 1002 FORMAT (1X, '  TIME', 3X, 15(1X, A9))
 1011 FORMAT (///, 20X, 'RESERVOIR OPERATIONS IN ', A40, /,
     &              20X, ' FROM ', A4, A5 //)
 1012 FORMAT (5X, 5(12X, A, 6X))
 1013 FORMAT (15A)
      END
*======================================================================
* Name:       arcflw
* Purpose:    print the downstream flow from a node for current arc.
*======================================================================
      SUBROUTINE ARCFLW(ARC, FLOW, II, JJ, MXARC, NDNAM, MXND,IOUT,
     &                  FLW, NOP, OJ, FLW1, FLWTL, LAST)
      IMPLICIT   NONE
      INTEGER    ARC, MXARC, MXND, IOUT, NOP
      INTEGER    FLOW(MXARC), II(MXARC), JJ(MXARC)
      REAL       FLW
      INTEGER    ARC1, OJ, FLW1
      CHARACTER  NDNAM(MXND)*(*)
      LOGICAL    FLWTL, LAST
*
      INTEGER    FN, TN, K
```

```
*
      IF (FLWTL) THEN
                                      WRITE(IOUT, 950) NOP
 950      FORMAT(
     &    /,20X, 'CHANNEL FLOW FOR TIME STEP: ', I3,
     &    /,20X, '==============================',
     &    /,
     &           T50, 'FLOW',  T65,'TOTAL'
     &    /,10X, 'FROM-NODE', T28, 'TO-NODE',
     &           T50, '(A-FT)',T65,'FLOW',
     &    /,10X, '---------', T28, '-------',
     &           T50,'------', T65,'-----')
                                         FLWTL = .FALSE.
      ENDIF
      IF (ARC .GT. 0) THEN
                                                K = 1
                                            FN = II(ARC)
                                            TN = JJ(ARC)
      ELSE
                                                K = -1
                                             ARC = -ARC
                                            FN = JJ(ARC)
                                            TN = II(ARC)
      ENDIF
      IF (TN .NE. OJ) THEN
                                               OJ = TN
                                          FLW1 = K*FLOW(ARC)
      ELSE
                                   FLW1 = FLW1 + K*FLOW(ARC)
      ENDIF
      IF (.NOT. LAST) THEN
                                           ARC1 = ARC + 1
                   IF ( (FN .EQ. II(ARC1) .AND. TN .EQ. JJ(ARC1)) .OR.
     &         (FN .EQ. JJ(ARC1) .AND. TN .EQ. II(ARC1)) ) GOTO 99
      ENDIF
      WRITE(IOUT, 900) NDNAM(FN), NDNAM(TN), FLW1, INT(FLW)
 900  FORMAT(10X, A12, 5X, A12, 5X, I10, 5X, I10)
 99   RETURN
      END
*=====================================================================
* Name:       OUTPUT
* Purpose:    Print the arc flow status.
*=====================================================================
      SUBROUTINE OUTPUT (M, N, HI, LO, COST, FLOW, NARCS)
      IMPLICIT    NONE
      INTEGER     NARCS, M(NARCS), N(NARCS), COST(NARCS), HI(NARCS),
     &            LO(NARCS), FLOW(NARCS)
*
      INTEGER     I
      WRITE (*, '(8X, A/)') 'NETWORK STATUS AND MINIMUM COST FLOW'
      WRITE (*, 905)
 905  FORMAT (3X, 'ARC', 5X, 'I', 5X, 'J', 6X, 'COST', 5X, 'L.BND', 5X,
     &         'U.BND', 6X, 'FLOW', /)
      DO 7 I = 1, NARCS
                WRITE(*, 910) I, M(I), N(I), COST(I), LO(I), HI(I), FLOW(I)
 7    CONTINUE
 910  FORMAT (1X, I5, 2I6, 4I10)
      RETURN
      END
*==========================================================================
      SUBROUTINE OUTPUT2 (NDNAM, LDND, M, N, HI, LO, COST, FLOW,
     &                    ARTYP, NARCS)
      IMPLICIT    NONE
      INTEGER     LDND, NARCS, M(NARCS), N(NARCS), COST(NARCS),
     &            HI(NARCS), LO(NARCS), ARTYP(NARCS), FLOW(NARCS)
      CHARACTER   NDNAM(LDND)*(*)
*
      INTEGER     I
      WRITE (*, '(8X, A/)') 'NETWORK STATUS AND MINIMUM COST FLOW'
      WRITE (*, 905)
 905  FORMAT (3X, 'ARC', 1X, 'TYPE', 1X, 'FROM', 8X, 'TO', 16X, 'COST',
     &        5X, 'L.BND', 5X, 'U.BND', 6X, 'FLOW', /)
      DO 7 I = 1, NARCS
              WRITE(*, 910) I,ARTYP(I), NDNAM(IABS(M(I))), NDNAM(IABS(N(I))),
     &                  COST(I), LO(I), HI(I), FLOW(I)
 7    CONTINUE
 910  FORMAT(1X, 2I5, 1X, 2A12, 4I10)
      RETURN
      END
*=====================================================================
* Name:       FLWCK
* Purpose:    Check flow convergence.
*=====================================================================
      SUBROUTINE FLWCK(NNODS, MXND,
     &                  PTDWAR, NDDWAR, MXARC,
     &                  FLOW, OFLOW, FCRIT, NOTCOV)
      IMPLICIT    NONE
      INTEGER     NNODS, MXND, MXARC
      INTEGER     PTDWAR(MXND, 2), NDDWAR(MXARC)
      INTEGER     FLOW(MXARC)
      INTEGER     OFLOW(MXARC), FCRIT
      LOGICAL     NOTCOV
*
      INTEGER     I, N, LIM1, LIM2, ARC
```

```
*
*         Check convergence
*
      NOTCOV = .FALSE.
      DO 50 N = 1, NNODS
                              LIM1 = PTDWAR(N, 1)
                              LIM2 = PTDWAR(N, 2)
                               DO I = LIM1, LIM2
                              ARC = IABS (NDDWAR(I))
                 IF (ABS(FLOW(ARC) - OFLOW(ARC)) .LT. FCRIT) THEN
                                      CONTINUE
                                        ELSE
                                   NOTCOV = .TRUE.
                                       ENDIF
                              OFLOW(ARC) = FLOW(ARC)
                                     ENDDO
 50   CONTINUE
      RETURN
      END
*==================================================================
* Name:      CTFLWB
* Purpose:   Adjust the lower flow bound for control arcs.
* Author:    Xiaodong Jian
* Date:      11/27/95
*==================================================================
      SUBROUTINE CTFLWB(ARC, CTAR, NCTAR, CTARFW, LDCTAR, LO, LDARC,
     &                  DLO, CTRFLW)
      IMPLICIT   NONE
      INTEGER    LDCTAR, LDARC, ARC, CTAR
      INTEGER    NCTAR, CTARFW(LDCTAR, 3), LO(LDARC), DLO, DFB
      LOGICAL    CTRFLW
*
      INTEGER    I, A
*
      CTRFLW = .FALSE.
      DO I = 1, NCTAR
                               A = CTARFW(I, 1)
                        DFB = CTARFW(I,3) - CTARFW(I,2)
                   IF ( (DFB .LT. DLO .AND. ARC .EQ. 0) .OR.
     &        (DFB .LT. DLO .AND. A .NE. CTAR) ) THEN
                                     CONTINUE
                                      ELSE
                                     CTAR = A
                                  CTRFLW = .TRUE.
                                  A = CTARFW(I, 1)
                                IF (ARC .EQ. 0) THEN
                                 CTARFW(I, 2) = LO(A)
                   LO(A) = INT(0.5 * (CTARFW(I, 2) + CTARFW(I,3)))
                                        ELSE
                                  CTARFW(I,3) = LO(A)
                   LO(A) = INT(0.5 * (CTARFW(I, 2) + CTARFW(I,3)))
                                       ENDIF
                                      GOTO 99
                                     ENDIF
      ENDDO
 99   CONTINUE
      RETURN
      END
*==================================================================
* Name:      locflw
* Purpose:   Open a local inflow file.
*==================================================================
      SUBROUTINE LOCFLW(NND, NDNAM, LDND, NIFW, IFWND, LDIFW, IFWCD,
     &                  FILNAM, IU, OU, CTERM, COLSTR, LDCOL)
      IMPLICIT   NONE
      INTEGER    NND, LDND, NIFW, LDIFW, IFWND(LDIFW), IFWCD, IU, OU
      CHARACTER  NDNAM(LDND)*(*), FILNAM*(*)
*
      INTEGER    LDCOL
      CHARACTER  CTERM*(*), COLSTR(LDCOL)*(*)
      INTEGER    I, ND, NREC
      LOGICAL    ERR
      CHARACTER  UNITNM(0:2)*7
      DATA       UNITNM/'ACRE-FT', 'FT^3/S', 'FT^3/D'/
*
      CALL IO_OPFIL(IU, 1, FILNAM, 'ENTER LOCAL NET INFLOW FILE: ')
*
*         Skip title lines
*
      NIFW = 0
      DO I = 1, 2
                            READ (IU, *, END = 999)
      ENDDO
*
*         Read flow unit code
*
      READ (IU, *) IFWCD
*
*         Read the nodal name with local net inflow
*
      READ (IU, '(A)', END = 999) CTERM
      CALL STR_DIVD(CTERM, NIFW, COLSTR, LDCOL, 0, ' ,')
      DO I = 2, NIFW
                    CALL NAMNUM(NND, NDNAM, COLSTR(I), ND, 0, ERR)
```

```
                              IF (ERR) THEN
                          WRITE(*, 901) COLSTR(I), FILNAM
 901        FORMAT( '***ERROR***NODAL NAME: ', A12, ' IN THE FILE: ',A,
     &            /, ' NOT FOUND IN THE NETWORK CONFIGURATION.')
                              STOP !CALL EXIT
                                  ELSE
                              IFWND(I-1) = ND
                                  ENDIF
      ENDDO
      NIFW = NIFW - 1
*
      CALL NORECS(IU, NREC)
      WRITE (OU, 800) FILNAM, UNITNM(IFWCD), NIFW,
     &                NREC, (NDNAM(IFWND(I)), I = 1, NIFW)
 800  FORMAT (
     &    //, '========================',
     &     /, 'LOCAL INCREMENTAL INFLOWS:',
     &     /, '========================',
     &     /, 20X, '        FILE NAME: ', A
     &     /, 20X, '       FLOW UNITS: ', A
     &     /, 20X, 'NUMBER OF  NODES: ', I4,
     &     /, 20X, 'NUMBER OF RECORD: ', I4,
     &     / ,20X, '   LIST OF NODES: ', 4A10, 10(/, 36X, 4A10))
 999  RETURN
      END
*=======================================================================
* Name:        opmdf
* Purpose:     Open master data file.
*=======================================================================
      SUBROUTINE OPMDF(FILNAM, LDFIL, CTERM, COLSTR, LDCOL)
      IMPLICIT   NONE
      INTEGER    LDFIL, LDCOL
      CHARACTER  FILNAM(0:LDFIL)*(*), CTERM*(*), COLSTR(LDCOL)*(*)
*
      INTEGER    CD, L, N, IU
      CHARACTER  FILNM*30
      LOGICAL    CN, EXIST
*
      IU = 9
      FILNM = ' '
      DO N = 1, LDFIL
                              FILNAM(N) = ' '
      ENDDO
*
*--------------Open master data file
*
      IF (EXIST('OPONDS.CTR')) THEN
                    OPEN(10, FILE = 'OPONDS.CTR', STATUS = 'OLD')
                              READ (10, '(A)') FILNM
                         CLOSE(10, STATUS = 'delete')
      ENDIF
      CALL IO_RDSTR('Enter master file:', FILNM)
      CALL IO_OPFIL(IU, 1, FILNM, 'ENTER MASTER FILE: ')
      OPEN(11, FILE = 'OPONDS.CTR', STATUS = 'UNKNOWN')
      WRITE(11, '(A)') FILNM
      CLOSE(11)
*
      L = 0
 5    CONTINUE
      READ (IU, '(A)', END = 99) CTERM
      L = L + 1
      IF (CTERM .EQ. ' ') GOTO 5
      CALL STR_DIVD(CTERM, N, COLSTR, LDCOL, 0, ' ,')
      READ (COLSTR(1), '(I2)', ERR = 5) CD
      IF (CD .LT. 0 .OR. CD .GT. LDFIL) THEN
                              PRINT *, CHAR(7)
                           WRITE(*, 801) CD, FILNM
 801     FORMAT('***ERROR***INVALID FILE CODE: ', I2,
     &           ' IN THE FILE: ', A)
                              STOP !CALL EXIT
      ENDIF
      IF (CN(COLSTR(2), ':', 1)) GOTO 5
      FILNAM(CD) = COLSTR(2)
      IF (.NOT. EXIST(FILNAM(CD)) .AND. CD .LE. 20) THEN
                              PRINT *, CHAR(7)
                         CALL STR_LEN(FILNAM(CD), N)
                    WRITE(*,  802) FILNAM(CD)(1:N), CD, L, FILNM
 802     FORMAT('***ERROR***FILE: ', A, ' WITH FILE CODE ', I2,
     &       /, '  DOES NOT EXIST. CHECK RECORDS IN THE LINE ', I2,
     &           ' IN THE FILE: ', A)
                              STOP !CALL EXIT
      ENDIF
      GOTO 5
 99   CLOSE (IU)
      RETURN
      END
*=======================================================================
* Name:        arcval
* Purpose:     Create an arc and assign values.
*=======================================================================
      SUBROUTINE ARCVAL(NARCS, II, JJ, LO, HI, COST, ARTYP,
     &                  MXARC, I, J, L, U, C, TYP)
      IMPLICIT   NONE
      INTEGER    NARCS, MXARC
```

```
      INTEGER    II(MXARC), JJ(MXARC), LO(MXARC), HI(MXARC),
     &          COST(MXARC), ARTYP(MXARC)
      INTEGER    I, J, L, U, C, TYP
*
      NARCS = NARCS + 1
      II(NARCS) = I
      JJ(NARCS) = J
      LO(NARCS) = L
      HI(NARCS) = U
      COST(NARCS) = C
      ARTYP(NARCS) = TYP
*
      RETURN
      END
*==========================================================================
* Name:       Month
* Purpose:    TRANSFORM THE MONTH WITH NUMBER OR CHAR.
*==========================================================================
      SUBROUTINE MONTH (TY, NUM, CHA)
      IMPLICIT  NONE
      CHARACTER * 4, MOTH(12), CHA
      INTEGER   TY, NUM, I
      DATA MOTH / 'JAN.', 'FEB.', 'MAR.', 'APR.', 'MAY',  'JUN.',
     &            'JUL.', 'AUG.', 'SEP.', 'OCT.', 'NOV.', 'DEC.'/
      IF (TY .EQ. 1) THEN
                                 DO 1 I = 1, 12
                            IF (CHA .EQ. MOTH(I)) THEN
                                      NUM = I
                                     GO TO 10
                                     END IF
 1       CONTINUE
      ELSE IF (TY .EQ. 2) THEN
                                CHA = MOTH(NUM)
      ELSE
                 WRITE (*, *) ' ***MONTH TRANSFORM IS FAILED IN TYPE' , TY
      END IF
 10   CONTINUE
      RETURN
      END
*=======================================================================
      SUBROUTINE PNCK(PN, IU, FLAG, CTERM, COLSTR, LDCOL)
      IMPLICIT   NONE
      INTEGER    PN, LDCOL, IU
      LOGICAL    FLAG
      CHARACTER  CTERM*(*), COLSTR(LDCOL)*(*)
      INTEGER    N
      LOGICAL    CN
*
      FLAG = .TRUE.
      CTERM = ' '
      DO WHILE (CTERM .EQ. ' ')
                          READ (IU, '(A)', END = 999) CTERM
      ENDDO
      CALL STR_DIVD(CTERM, N, COLSTR, LDCOL, 0, ' ,')
      READ (COLSTR(2), '(I2)') N
      IF (CN(COLSTR(1), 'PART', 1) .AND. N .EQ. PN) THEN
                                 FLAG = .FALSE.
      ELSE
                IF (CN(COLSTR(1), 'PART', 1) .AND. N .GT. PN)THEN
                                   BACKSPACE(IU)
                                     GOTO 999
                                     ENDIF
                                PRINT *, CHAR(7)
                                WRITE(*, 801) PN
 801     FORMAT( '***ERROR***ERROR IN PART ', I1,
     &          ' OF NETWORK DATA FILE')
                             CALL STR_LEN(CTERM, N)
                        PRINT *, 'RECORD  = ', CTERM(1:N)
                                STOP !CALL EXIT
      ENDIF
 999  RETURN
      END
*=====================================================================
      LOGICAL  FUNCTION CNINT(ISEED, NOIX, IX, LOC)
      IMPLICIT NONE
      INTEGER  ISEED, NOIX, IX(NOIX), LOC
*
      INTEGER   I
*
      CNINT = .FALSE.
      DO I = 1, NOIX
                           IF (ISEED .EQ. IX(I)) THEN
                                  CNINT = .TRUE.
                                     LOC = I
                                     GOTO 99
                                     ENDIF
      ENDDO
 99   RETURN
      END
*========================================================================
      LOGICAL  FUNCTION CN2INT(ISD1, ISD2, NOIX, IX1, IX2, LOC)
      IMPLICIT NONE
      INTEGER  ISD1, ISD2, NOIX, IX1(NOIX), IX2(NOIX), LOC
*
```

```
      INTEGER    I
*
      CN2INT = .FALSE.
      I = 1
      DO WHILE (I .LE. NOIX .AND. .NOT. CN2INT)
                     IF (ISD1 .EQ. IX1(I) .AND. ISD2 .EQ. IX2(I)) THEN
                                       CN2INT = .TRUE.
                                           LOC = I
                                          ENDIF
                                         I = I + 1
      ENDDO
 99   RETURN
      END
*======================================================================
* Name:       fdwnd
* Purpose:    Find DownStream Node.
*======================================================================
      SUBROUTINE FDWND(ARC, II, JJ, LDARC, NSTRM, STRMAR, LDSTRM, DWND)
      IMPLICIT   NONE
      INTEGER    ARC, LDARC, II(LDARC), JJ(LDARC)
      INTEGER    LDSTRM, NSTRM, STRMAR(LDSTRM,0:6)
      INTEGER    DWND
*
      INTEGER    I, ARC2
      LOGICAL    CNINT
*
      IF (CNINT(ARC, NSTRM, STRMAR(1,1), I)) THEN
                                  ARC2 = STRMAR(I, 6)
      ELSE
                                      ARC2 = ARC
      ENDIF
      IF (ARC2 .GT. 0) THEN
                                    DWND = JJ(ARC2)
      ELSE
                                   DWND = II(-ARC2)
      ENDIF
      RETURN
      END
*======================================================================
* Name:       dattb
* Purpose:    Read seasonal data matrix.
* Author:     Xiaodong Jian
* Date:       02/29/95
*======================================================================
      SUBROUTINE DATTB(NNODS, NDNAM, LDND, NDTND, DTND, LDDTND, UNITCD,
     &                 NREC, DTTB, LDDTTB, DTFLAG,
     &                 UNITNM, LDUNIT, FILNAM, IU, OU, PN, ENDFIL,
     &                 TITLE, CTERM, COLSTR, LDCOL)
      IMPLICIT   NONE
      INTEGER    NNODS, LDND, NDTND, LDDTND, DTND(LDDTND, 3), NREC,
     &           LDDTTB
      REAL       DTTB(0:LDDTTB, LDDTND)
      INTEGER    UNITCD, LDUNIT, IU, OU, PN
      CHARACTER  NDNAM(LDND)*(*), UNITNM(0:LDUNIT)*(*), FILNAM*(*)
      LOGICAL    DTFLAG, ENDFIL
      INTEGER    LDCOL
      CHARACTER  TITLE*(*), CTERM*(*), COLSTR(LDCOL)*(*)
*
      INTEGER    I, J, K, L, ND, SL, NNDS
      LOGICAL    ERR, CN, CNINT, DUMMY
      INTEGER    SAVOPT
      COMMON     /SAVOPT/ SAVOPT
*
      DUMMY = .FALSE.
      DTFLAG = .FALSE.
      ENDFIL = .TRUE.
      NDTND = 0
      NREC = 0
*
*-------------Read data unit and nodal names
*
      READ (IU, *, END = 999) UNITCD
      READ (IU, '(A)', END = 999) CTERM
      CALL STR_DIVD(CTERM, NNDS, COLSTR, LDCOL, 0, ' ,')
      DO I = 2, NNDS
                        IF (CN(COLSTR(I), 'DEFAULT', 1)) THEN
                                     DUMMY = .TRUE.
                                      NDTND = NNODS
                                IF (NNDS .EQ. 2) THEN
                                        DO J = 1, NNODS
                                     DTND(J, 1) = J
                                DTND(J, 2) = UNITCD
                                     DTND(J, 3) = 0
                                              ENDDO
                                             ELSE
                                          L = NNDS - 2
                                        DO J = 1, NNODS
                    IF (.NOT. CNINT(J, NNDS-2, DTND(1,1), K)) THEN
                                            L = L + 1
                                       DTND(L, 1) = J
                                   DTND(L, 2) = UNITCD
                                       DTND(L, 3) = 0
                                           ENDIF
                                              ENDDO
```

```
                                                  ENDIF
                                                GOTO 50
                                                ELSE
                    CALL NAMNUM(LDND, NDNAM, COLSTR(I), ND, 0, ERR)
                                              IF (ERR) THEN
                                       WRITE(*, 850) COLSTR(I), FILNAM
 850              FORMAT( '***ERROR***NODAL NAME: ',A12,' IN THE FILE: ',A,
     &              /, ' NOT FOUND IN THE NETWORK CONFIGURATION.')
                                              STOP !CALL EXIT
                                                  ELSE
                                             DTND(I-1, 1) = ND
                                           DTND(I-1, 2) = UNITCD
                                             DTND(I-1, 3) = 0
                                                  ENDIF
                                                ENDIF
      ENDDO
      NDTND = NNDS - 1
*
*-------------Read data table
*
 50   CONTINUE
      READ (IU, '(A)', END = 99) CTERM
      IF (CN(CTERM, 'FINISH', 1)) GOTO 99
      CALL  STR_DIVD(CTERM, J, COLSTR, LDCOL, 0, ' ,')
      IF (J .NE. NNDS ) THEN
                         PRINT *, '***ERROR*** FILE = ', FILNAM
                        PRINT *, '        FOR TIME = ',COLSTR(1)
                                    STOP !CALL EXIT
      ENDIF
*
      NREC = NREC + 1
      DO J = 1, NNDS - 1
                     READ (COLSTR(J+1), '(F15.0)') DTTB(NREC, J)
      ENDDO
      GOTO 50
*
*-------------Output Title
*
 99   CONTINUE
      IF (DUMMY) THEN
                                     DO I = 1, NREC
                                     DO J = NNDS, NDTND
                                 DTTB(I, J) = DTTB(I, NNDS-1)
                                          ENDDO
                                        ENDDO
      ENDIF
      IF (NREC .GT. 0) THEN
                                   ENDFIL = .FALSE.
                                    DTFLAG = .TRUE.
                     IF (SAVOPT .EQ. 0 .OR. SAVOPT .EQ. 1) THEN
                                 CALL STR_LEN(TITLE, J)
                             CALL STR_LEN(UNITNM(UNITCD), K)
                 WRITE(OU, 801) PN, TITLE (1:J), UNITNM(UNITCD)(1:K),
     &                ('=', I = 1, J+K+5)
 801          FORMAT(//, 'Part ', I2,': Seasonal ', A, ', in ', A,
     &           /, '=================', 100A)
*
                                  IF (.NOT. DUMMY) THEN
                                     DO K = 1, NDTND, 8
                               IF ((K+7) .GT. NDTND) THEN
                                          L = NDTND
                                           ELSE
                                          L = K + 7
                                          ENDIF
                                      WRITE(OU, 805)
     &               (NDNAM(DTND(J,1))(1:SL(NDNAM(DTND(J,1)))),J = K,L)
                                      DO I = 1, NREC
                         WRITE (OU, 810) I, (DTTB(I, J), J = K, L)
                                           ENDDO
                                             ENDDO
                                            ELSE
                 WRITE(OU, 805) (NDNAM(DTND(J,1))(1:SL(NDNAM(DTND(J,1)))),
     &                       J = 1, NNDS-2), 'DEFAULT'
                                         DO I = 1, NREC
                     WRITE (OU, 810) I, (DTTB(I, J), J = 1, NNDS-1)
                                             ENDDO
                                           ENDIF
 805          FORMAT(/, 'SEASON', T10, 10A10)
 810          FORMAT (I5, T10, 10F10.2)
                                          ENDIF
      ENDIF
*
 999  CONTINUE
      RETURN
      END
*===================================================================
* Name:       RESSTG
* Purpose:    Calculate reservoir stges.
*===================================================================
      SUBROUTINE RESSTG(NND, NDTYP, NDSEQ, LDND,
     &                  RC, INST, PTREAR, LDRES, REAR, LDREAR,
     &                  FLOW, LDARC, XF)
      IMPLICIT    NONE
      INTEGER     LDND, LDRES, LDREAR, LDARC
```

```
      INTEGER     NND, NDTYP(LDND), NDSEQ(LDND)
      INTEGER     PTREAR(LDRES), REAR(LDREAR)
      INTEGER     FLOW(LDARC)
      REAL        RC(LDRES), INST(LDRES), XF
*
      INTEGER     N, I, LIM1, LIM2, RES, ARC, TYP
      REAL        STG
      DO 200 N = 1, NND
                                    TYP = NDTYP(N)
                                 IF (TYP .EQ. 1) THEN
                                      RES = NDSEQ(N)
                                      STG = RC(RES)
                                    LIM1 = PTREAR(RES)
                               LIM2 = PTREAR(RES + 1) - 1
                                    DO 45 I = LIM1, LIM2
                                        ARC = REAR(I)
                                    IF (ARC .GT. 0) THEN
                           STG = STG + REAL(FLOW(ARC)) / XF
                                            ELSE
                                       ARC = -ARC
                           STG = STG - REAL(FLOW(ARC)) / XF
                                           END IF
 45         CONTINUE
                                     INST(RES) = STG
                                         ENDIF
 200  CONTINUE
      RETURN
      END
*==================================================================
* Name:       ndbud
* Purpose:    calculate nodal water budgets.
*==================================================================
      SUBROUTINE NDBUD(FLOW, LDARC, RC, OINST, INST, RESDAT, LDRES,
     &                 NND, NDTYP, NDSEQ, PTDWAR, LDND, DWAR, LDDWAR,
     &                 NDXAR, NODBUD, XF, OU)
      IMPLICIT    NONE
      INTEGER     LDARC, LDRES, LDND, LDDWAR,
     &            FLOW(LDARC), NND, NDTYP(LDND), NDSEQ(LDND),
     &            PTDWAR(LDND, 2), DWAR(LDDWAR)
      INTEGER     NODBUD(LDND, 0:10), NDXAR(LDND, 6), OU
      REAL        RC(LDRES), OINST(LDRES), INST(LDRES), RESDAT(LDRES,0:2)
      REAL        XF
*
      INTEGER     ISGN
*
      REAL        FLW, ZVA(3)
      INTEGER     I, J, N, TYP, LIM1, LIM2, ARC, RES
      LOGICAL     GETZVA
*
      DO 200 N = 1, NND
*
*----------------Initialize budget array
*
                                      DO J = 0, 8
                                    NODBUD(N, J) = 0
                                         ENDDO
                                    TYP = NDTYP(N)
*
*----------------Total releases from a current node.
*
                                       FLW = 0.0
                                  LIM1 = PTDWAR(N, 1)
                                  LIM2 = PTDWAR(N, 2)
                                 DO 50 J = LIM1, LIM2
                                      ARC = DWAR(J)
                                  IF (ARC .GT. 0) THEN
                               FLW = FLW + REAL (FLOW(ARC))
                                          ELSE
                              FLW = FLW - REAL (FLOW(-ARC))
                                         END IF
 50      CONTINUE
                                 IF (TYP .EQ. 1) THEN
*
*------------------1. Reservoir budget
*
                                    RES = NDSEQ(N)
                         NODBUD(N, 0) = NINT(OINST(RES) * XF)
*
*------------------2. Local gain and loss
*
                                      DO I = 1, 5
                                    ARC = NDXAR(N,I)
                                  IF (ARC .NE. 0) THEN
                                  IF (I .EQ. 1) THEN
                       NODBUD(N,2) = ISGN(ARC) * FLOW(IABS(ARC))
     &                            - NINT((OINST(RES) - RC(RES))*XF)
                                ELSE IF (I .EQ. 2) THEN
                       NODBUD(N,3) = ISGN(ARC) * FLOW(IABS(ARC))
                                ELSE IF (I .EQ. 3) THEN
                       NODBUD(N,4) = ISGN(ARC) * FLOW(IABS(ARC))
                                ELSE IF (I .EQ. 4) THEN
                       NODBUD(N,5) = ISGN(ARC) * FLOW(IABS(ARC))
                                ELSE IF (I .EQ. 5) THEN
                NODBUD(N,6) = FLOW(NDXAR(N, 5)) - FLOW(NDXAR(N, 6))
```

```
                                   ENDIF
                                      ENDIF
                                   ENDDO
*
*-------------------3. Downsteam release
*
                              NODBUD(N,7) = FLW
*
*-------------------4. Final storage and reservoir elevatoin.
*
                        NODBUD(N,8) = NINT(INST(RES) * XF)
                              ZVA(2) = INST(RES)
                        IF (GETZVA(RES, 2, ZVA, OU)) THEN
                           NODBUD(N,9) = INT(ZVA(1)*100.0)
                  NODBUD(N,10) = INT(NODBUD(N,9) - RESDAT(RES,0)*100.0)
                                   ELSE
                        PRINT *, '***ERROR*** RESERVOIR STAGE'
                              STOP !CALL EXIT
                                   ENDIF
*
*-------------------5. Upstream Inflow: Q = EV + L + W + R + S - I - P - So
*
                NODBUD(N,1) = NODBUD(N,3) + NODBUD(N, 5) + NODBUD(N,6)
     &                 + NODBUD(N,7) + NODBUD(N, 8)
     &                 - NODBUD(N,2) - NODBUD(N,4) - NODBUD(N,0) !INFLOW
                           ELSE IF (TYP .EQ. 2) THEN
*
*-----------Water supply node budget
*
*-------------------1. Local Gain and Loss
*
                                   DO I = 1, 6
                                   ARC = NDXAR(N,I)
                              IF (ARC .NE. 0) THEN
                              IF (I .EQ. 1) THEN
                      NODBUD(N,2) = ISGN(ARC) * FLOW(IABS(ARC))
                              ELSE IF (I .EQ. 3) THEN
                      NODBUD(N,4) = ISGN(ARC) * FLOW(IABS(ARC))
                              ELSE IF (I .EQ. 5) THEN
                  NODBUD(N,6) = FLOW(NDXAR(N,5)) - FLOW(NDXAR(N,6))
                                   ENDIF
                                      ENDIF
                                   ENDDO
*
*------------------2. downstream release
*
                              NODBUD(N,7) = FLW
*
*------------------3. Upstream Inflow: Q = W + R - I - Runoff
*
                NODBUD(N,1) = NODBUD(N,6) + NODBUD(N,7) - NODBUD(N,2)
     &                 - NODBUD(N,3)
                                   ENDIF
 200  CONTINUE
      RETURN
      END
*=====================================================================
* Name:       prtnd
* Purpose:    Print nodal budget.
*=====================================================================
      SUBROUTINE PRTND(NND, NDNAM, NDTYP, NODBUD, LDND, NOP, XF, XP,
     &                 NSNBL, SNBLND, LDSNBL, SNBLFG, NDBFLG,
     &                 ZEROFG, IOUT)
      IMPLICIT   NONE
      INTEGER    LDND, NND, NDTYP(LDND), NODBUD(LDND, 0:10), NOP, IOUT,
     &           XP, LDSNBL
      REAL       XF
      INTEGER    NSNBL, SNBLND(LDSNBL, 2)
      LOGICAL    SNBLFG, NDBFLG, ZEROFG
      CHARACTER  NDNAM(LDND)*(*)
*
      INTEGER    I, N, K, OU
      REAL       EPS
      CHARACTER  FMT2*60, CTERM*150
      LOGICAL    CNINT
*
      EPS = 5.0 * 10**(-XP)
      ZEROFG = .TRUE.
*
*----------------Output time step
*
      IF (NDBFLG) THEN
                              WRITE(IOUT, 990) NOP
 990     FORMAT (/,T30, 'Water budgets for time step:', I3,
     &           /,T30, '==============================',
     &           /,T30, '   [--, not applicable]',
     &   /,                    T21, '   Initial  Upstream Local Net',
     &'    Evapo-                          Downstream     Final',
     & /,                      T21, '   storage    inflow    inflow',
     &'    ration    Runoff   Seepage Withdrawal  release   storage',
     &'     Final     Water',
     & /,'    Node',T16, 'Node',T21, '    (acre-    (acre-    (acre-',
     &'   (acre-    (acre-    (acre-    (acre-    (acre-    (acre-',
     &'     stage     depth',
```

```
     & /,'No. name',T16, 'type', T21, '     feet)     feet)     feet)',
     &'     feet)     feet)     feet)     feet)     feet)',
     &'    (feet)    (feet)',
     & /,'--- ----',T16, '----', T21,  ' --------- --------- ---------',
     &' --------- --------- --------- --------- --------- ---------',
     &' --------- ---------')
      ENDIF
*
*-----------Save nodal water budgets
*
      IF (XP .GT. 0) THEN
                               FMT2 = '(F10.0)'
                       WRITE(FMT2(6:6), '(I1)') XP - 1
                                 DO N = 1, NND
                                   CTERM = ' '
*
*           Nodal name and type
*
                    WRITE(CTERM(1:20), '(I3, 1X, A12, 1X, I2, 2X)')
     &                        N, NDNAM(N), NDTYP(N)
*
*           Water budgets
*
                                   DO I = 0, 8
                    IF (NDTYP(N) .EQ. 2 .AND. (I .EQ. 0 .OR. I .EQ. 3
     &              .OR. I .EQ. 5 .OR. I .EQ. 8)) THEN
                 WRITE(CTERM((I+2)*10+1:(I+3)*10), '(6x,a2,2x)') '--'
                    ELSE IF (ABS(NODBUD(N, I)) .GT. 0 .OR. ZEROFG) THEN
                      WRITE(CTERM((I+2)*10+1:(I+3)*10), FMT2)
     &                REAL(NODBUD(N,I)) / XF
                                       ENDIF
                                      ENDDO
                             CALL STR_LEN(CTERM, I)
*
*           Final stage and water depth
*
                                 IF (NDBFLG) THEN
                               IF (NDTYP(N) .EQ. 1) THEN
                        WRITE(IOUT, '(A, 2F10.2)') CTERM(1:I),
     &                REAL(NODBUD(N,9))/100.,REAL(NODBUD(N,10))/100.
                                        ELSE
                 WRITE(IOUT,'(A, 2x,2(6x,a2,2x))')CTERM(1:I),'--','--'
                                       ENDIF
                                      ENDIF
*
                  IF (SNBLFG .AND. CNINT(N, NSNBL, SNBLND, K) ) THEN
                                  OU = SNBLND(K, 2)
                                     CTERM = ' '
                                     DO I = 0, 8
                        IF (ABS(NODBUD(N, I)) .GT. EPS) THEN
                          WRITE(CTERM(I*10+1:(I+1)*10), FMT2)
     &                  REAL(NODBUD(N,I)) / XF
                                        ELSE
                        WRITE(CTERM(I*10+1:(I+1)*10), FMT2) 0.0
                                     ENDIF
                                       ENDDO
                               CALL STR_LEN(CTERM, I)
                             IF (NDTYP(N) .EQ. 1) THEN
                         WRITE(OU, '(I4, 6X, A, 2F10.2)')
     &                  NOP, CTERM(1:I), REAL(NODBUD(N,9))/100.,
     &                                   REAL(NODBUD(N,10))/100.
                                        ELSE
                  WRITE(OU, '(I4, 6X, A, F10.2)') NOP, CTERM(1:I)
                                       ENDIF
                                      ENDIF
*
                                     ENDDO
      ELSE
                                 DO N = 1, NND
                                   CTERM = ' '
                    WRITE(CTERM(1:20), '(I3, 1X, A12, 1X, I2, 2X)')
     &              N, NDNAM(N), NDTYP(N)
                                   DO I = 0, 8
                      IF (ABS(NODBUD(N, I)) .GT. 0 .OR. ZEROFG) THEN
                WRITE(CTERM((I+2)*10+1:(I+3)*10), '(I10)') NODBUD(N,I)
                                       ENDIF
                                      ENDDO
                             CALL STR_LEN(CTERM, I)
                                 IF (NDBFLG) THEN
                               IF (NDTYP(N) .EQ. 1) THEN
                        WRITE(IOUT, '(A, 2F10.2)') CTERM(1:I),
     &              REAL(NODBUD(N,9))/100.0,REAL(NODBUD(N,10))/100.0
                                        ELSE
                          WRITE(IOUT, '(A)') CTERM(1:I)
                                       ENDIF
                                      ENDIF
*
                  IF (SNBLFG .AND. CNINT(N, NSNBL, SNBLND, K) ) THEN
                                  OU = SNBLND(K, 2)
                                     CTERM = ' '
                                     DO I = 0, 8
                        IF (ABS(NODBUD(N, I)) .GT. EPS) THEN
                         WRITE(CTERM(I*10+1:(I+1)*10), '(I10)')
     &                                      NODBUD(N,I)
```

```
                                        ELSE
                       WRITE(CTERM(I*10+1:(I+1)*10), '(I10)') 0
                                       ENDIF
                                          ENDDO
                               CALL STR_LEN(CTERM, I)
                              IF (NDTYP(N) .EQ. 1) THEN
                  WRITE(OU, '(I4, 6X, A, 2F10.2)') NOP, CTERM(1:I),
     &              REAL(NODBUD(N,9))/100.,REAL(NODBUD(N,10))/100.
                                          ELSE
                   WRITE(OU, '(I4, 6X, A, F10.2)') NOP, CTERM(1:I)
                                          ENDIF
                                        ENDIF
*
                                       ENDDO
      ENDIF
      RETURN
      END
*====================================================================
* Name:       NETWK_1
* Purpose:    READ IN THE RESERVOIR ZONES AND CREATE THE CORRESPONDING
*             ARCS TO/FROM SINK/SOURCE NODE OR RIVERS
*====================================================================
      SUBROUTINE NETWK_1(NDNAM, NDSEQ, LDND,
     &                   NARCS, II, JJ, HI, LO, COST, ARTYP, LDARC,
     &                   NRES, PTRES, RC, INST, RESDAT, LDRES,
     &                   PTREAR, LDPTRE, REAR, REZN, LDREAR,
     &                   SKSC, XF, FILNAM, NTLS,
     &                   IN, OU, PARTNO)
      IMPLICIT   NONE
      INTEGER    LDND, NDSEQ(LDND)
      CHARACTER  NDNAM(LDND)*(*)
      INTEGER    NARCS, LDARC, II(LDARC), JJ(LDARC), HI(LDARC),
     &           LO(LDARC), COST(LDARC), ARTYP(LDARC)
      INTEGER    NRES, LDRES, PTRES(LDRES)
      REAL       RC(LDRES), INST(LDRES),
     &           RESDAT(LDRES, 0:2)
      INTEGER    LDPTRE, PTREAR(LDPTRE), LDREAR, REAR(LDREAR)
      REAL       REZN(LDREAR)
      INTEGER    SKSC, NTLS, IN, IU, OU, PARTNO
      REAL       XF
*
      REAL       RCEL, BON, LVL1, LVL2, DIF, ZVA(3), INEL, EPS
      INTEGER    ND, CST, I, N, L, U, NZONE, NLZ, NUZ, NZ, UNITCD
      CHARACTER  FILNAM*(*)
      LOGICAL    CN, GETZVA, ERR, FLAG
*
      INTEGER    NCOL, LDCOL
      PARAMETER  (LDCOL = 15)
      CHARACTER  CTERM * 150, COLSTR(LDCOL)*15
      INTEGER    SAVOPT
      COMMON     /SAVOPT/ SAVOPT
      DATA       EPS /0.0001/
*
*----------Open reservoir storage zone file
*
      IF (FILNAM .NE. ' ') THEN
                                  FLAG = .TRUE.
                                      IU = 9
                CALL IO_OPFIL(IU, 1, FILNAM, 'ENTER RESERVOIR FILE: ')
                                 DO I = 1, NTLS
                                    READ (IU, *)
                                      ENDDO
      ELSE
                                 FLAG = .FALSE.
                                     IU = IN
                                   CTERM = ' '
                            DO WHILE (CTERM .EQ. ' ')
                          READ (IU, '(A)', END = 100) CTERM
                                      ENDDO
                 CALL STR_DIVD(CTERM, I, COLSTR, LDCOL, 0, ' ,')
                             READ (COLSTR(2), '(I2)') N
                IF (CN(COLSTR(1), 'PART', 1) .AND. N .EQ. PARTNO) THEN
                         READ (IU, *) !SKIP THE VARIABLE LIST
                                       ELSE
                 IF (CN(COLSTR(1), 'PART', 1) .AND. N .GT. PARTNO )THEN
                                   BACKSPACE(1)
                                     GOTO 100
                                      ENDIF
                                  PRINT *, CHAR(7)
                  PRINT *, '***ERROR***ERROR IN RESERVOIR ZONING DATA'
                   PRINT *, 'PART = ', COLSTR(1), ' NO = ', COLSTR(2)
                                  STOP !CALL EXIT
                                      ENDIF
      ENDIF
      NRES = 0
      NZ = 0       ! TOTAL NUMBER OF EXISTING ZONES.
*
      IF (SAVOPT .EQ. 0 .OR. SAVOPT .EQ. 1) THEN
                                WRITE (OU, 800)
 800     FORMAT (//, 'Part 1: Pond-storage zoning information',
     &            /, '======================================')
      ENDIF
*
*-----------Read reservoir name, rule curve, zones, intial storage,
```

```
*           bottom information, and zones.
*
 5    CONTINUE
      NLZ = 0
      NUZ = 0
      READ (IU, '(A)', END = 100) CTERM
      IF (CTERM .EQ. ' ') GOTO 5
      CALL STR_DIVD(CTERM, NCOL, COLSTR, LDCOL, 0, ' ,')
      IF (CN(COLSTR(1), 'FINISH', 1)) GOTO 100
*
*-------------1. Reservoir node name and its nodal number
*
      CALL NAMNUM(LDND, NDNAM, COLSTR(1), ND, 0, ERR)
      IF (ERR)  THEN
              PRINT *, '***Error*** There is no Z-V-A table for pond node:',
     &            COLSTR(1)
                   PRINT *, 'which is specified in the pond zoning file.'
                                                 STOP
      ENDIF
      NRES = NRES + 1
      PTRES(NRES) = ND
      NDSEQ(ND) = NRES
*
      IF (SAVOPT .EQ. 0 .OR. SAVOPT .EQ. 1) THEN
                                 WRITE (OU, 801) NDNAM(ND), ND
 801     FORMAT(/,A, T15, I10, T40,
     &       ':Pond name and corresponding node number')
      ENDIF
*
*-------------2. Stage unit code
*
      READ (COLSTR(2), '(I2)') UNITCD
*
*-------------3. Reservoir initial stroage, bottom elvation, poolbed
*                  hydraulic conductivity, and poolbed thickness
*
      READ (COLSTR(3), '(F10.0)') INEL
      IF (UNITCD .EQ. 0) THEN
                                         ZVA(1) = INEL
                             IF (.NOT. GETZVA(ND, 1, ZVA, OU)) THEN
                                           ERR = .TRUE.
                                      WRITE (OU, 1) NDNAM(ND)
 1          FORMAT ('****ERROR***',
     &          'IN TRANSFORMING INITIAL POOR ELEVATION INTO STORAGE'
     &        /,'    FOR RESERVOIR: ', A, ' AT NETWK_1')
                                                   STOP
                                                END IF
                                     INST(NRES) = ZVA(2)
      ELSE IF (UNITCD .EQ. 1) THEN
                                        INST(NRES) = INEL
      ENDIF
*
*-------------4. Reservoir bottom elevation and hydraulic properties.
*
      DO I = 0, 2
                            READ(COLSTR(I+4), '(F10.0)') RESDAT(NRES, I)
      ENDDO
      IF (SAVOPT .EQ. 0 .OR. SAVOPT .EQ. 1) THEN
                            WRITE (OU, 803) (RESDAT(NRES, I), I = 0, 2)
 803     FORMAT(5X, 3F10.2, T40,
     &       ':Bottom elevation, hydraulic conductivity, and thickness')
      ENDIF
*
*
*-------------5. Reservoir rule curve
*
      READ (COLSTR(7), '(F10.0)') RCEL
      IF (UNITCD .EQ. 0) THEN
                                         ZVA(1) = RCEL
                             IF (.NOT. GETZVA(ND, 1, ZVA, OU)) THEN
                                           ERR = .TRUE.
                                      WRITE (OU, 2) NDNAM(ND)
 2          FORMAT ('****ERROR*** IN TRANSFORMING RULE CURVE ELEVATION',
     &          /,  '   INTO CORREPONDING STORAGE FOR RESERVOIR: ', A)
                                           STOP !CALL EXIT
                                                END IF
                                     RC(NRES) = ZVA(2)
      ELSE
                                         RC(NRES) = RCEL
      ENDIF
      IF (INST(NRES) .LE. EPS) THEN
                                    INST(NRES) = RC(NRES)
      ENDIF
      IF (SAVOPT .EQ. 0 .OR. SAVOPT .EQ. 1) THEN
                                     IF (UNITCD .EQ. 0) THEN
                                 WRITE (OU, 805) INST(NRES), RCEL,
     &                      ':Rule-curve elevation, in feet'
                                                 ELSE
                               WRITE (OU, 805) INST(NRES), RC(NRES),
     &                      ':Rule-curve storage, in acre-feet'
                                                ENDIF
 805     FORMAT(3X, F12.2, T40, ':Initial storage, in acre-feet',
     &     /,3X, F12.2, T40, A)
      ENDIF
```

```
*
*------------6 reservoir zoning
*
      LVL1 = RC(NRES)
      LVL2 = RC(NRES)
      NZONE = (NCOL - 7) / 2
      IF (NZONE .EQ. 0) THEN
              PRINT *, '***ERROR*** RESERVOIR STORAGE ZONE IS NOT DEFINED',
     &              ' FOR NODE: ', NDNAM(ND)
                                  STOP !CALL EXIT
      ENDIF
      DO 25 N = 1, NZONE
                                    NZ = NZ + 1
                         READ (COLSTR(N*2+6), '(F10.0)') BON
                          READ (COLSTR(N*2+7), '(I10)') CST
*
*                 TRANSFORM THE BOND FROM STAGE TO A VOLUME
*
                               IF (UNITCD .EQ. 0) THEN
                       IF (SAVOPT .EQ. 0 .OR. SAVOPT .EQ. 1) THEN
                              WRITE (OU, 806) N, BON, CST,
     &             ':Zone number, elevation, and penalty coefficient'
                                       ENDIF
                                    ZVA(1) = BON
                         IF (.NOT. GETZVA(ND, 1, ZVA, OU)) THEN
                                     ERR = .TRUE.
                                  WRITE (OU, 910) ND
                                        STOP
                                      END IF
                                    BON = ZVA(2)
                                      ELSE
                       IF (SAVOPT .EQ. 0 .OR. SAVOPT .EQ. 1) THEN
                              WRITE (OU, 806) N, BON, CST,
     &                       ':ZONE NUMBER, STORAGE, AND PENALTY COST'
                                       ENDIF
                                      ENDIF
 806     FORMAT (I5, F10.2, I7, T40, A)
                               DIF = BON - RC(NRES)
                               IF (DIF .GE. 0) THEN
                                   NUZ = NUZ + 1
                              U = NINT((BON - LVL1)*XF)
                                    LVL1 = BON
                     CALL ARCVAL(NARCS, II, JJ, LO, HI, COST, ARTYP,
     &                LDARC, ND, SKSC, 0, U, CST, 200+NUZ)
                          PTREAR(NRES+1) = PTREAR(NRES+1) + 1
                                 REAR(NZ) = NARCS
                                      ELSE
                                   NLZ = NLZ + 1
                              U = NINT((LVL2 - BON)*XF)
                                    LVL2 = BON
                     CALL ARCVAL(NARCS, II, JJ, LO, HI, COST, ARTYP,
     &                LDARC, SKSC, ND, 0, U, CST, -200-NLZ)
                          PTREAR(NRES+1) = PTREAR(NRES+1) + 1
                                 REAR(NZ) = -NARCS
                                    END IF
*
                                 REZN(NZ) = BON
 25    CONTINUE
       GOTO 5
 100   CONTINUE
       PTREAR(1) = 1
       DO 50 L = 2, NRES + 1
                        PTREAR (L) = PTREAR(L) + PTREAR(L-1)
 50    CONTINUE
       IF (FLAG) THEN
                                    CLOSE (IU)
       ENDIF
       RETURN
 910   FORMAT ('*** ERROR IN TRANSFORMING RESERVOIR BONDS OF POND',
     &         ' NO. ', I2, ' ***')
       END
*=====================================================================
* Name:        NETWK2
* Purpose:     READ IN CHANNEL ZONES FOR WATER NODES, CREATE THE
*              CORRESPONDING ARCS TO/FROM SINK/SOURCE NODE OR RIVERS
*=====================================================================
       SUBROUTINE NETWK_2(NNODS, NDNAM, NDTYP, LDND,
     &                    NARCS, II, JJ, HI, LO, COST, ARTYP, LDARC,
     &                    NARCND, NSTRM, STRMCF, STRMAR, LDSTRM,
     &                    PTDWAR, LDPT, NDWAR, DWAR, LDDWAR,
     &                    NCTAR, CTARFW, LDCTAR,
     &                    PERD, C, XF, FILNAM, NTLS,
     &                    IN, OU, PARTNO)
       IMPLICIT   NONE
*
       INTEGER    NNODS, LDND, NDTYP(LDND)
       CHARACTER  NDNAM(LDND)*(*)
*
       INTEGER    NARCS, LDARC, II(LDARC), JJ(LDARC), HI(LDARC),
     &            LO(LDARC), COST(LDARC), ARTYP(LDARC)
       INTEGER    NARCND, LDSTRM, NSTRM, STRMAR(LDSTRM,0:6)
       REAL       STRMCF(LDSTRM, 0:4)
       INTEGER    LDPT, PTDWAR(LDPT, 2), LDDWAR, NDWAR, DWAR(LDDWAR)
       INTEGER    LDCTAR, NCTAR, CTARFW(LDCTAR, 3)
```

```
      INTEGER     IU, NTLS, IN, OU, PARTNO
      REAL        PERD, C, XF
*
      INTEGER     I, J, L, N, U, ZONE, UPND, OUPND, MDND, DWND, ODWND,
     &            NCOL, LDCOL, NRND, CST, LZONE, UZONE, TYP
      REAL        LVL1, LBND, UBND, RTERM, EPS
      PARAMETER   (LDCOL = 15)
      CHARACTER   CTERM*150, COLSTR(LDCOL)*20, FILNAM*(*)
      LOGICAL     CN, ERR, STRM, ITOJ, FLAG
      INTEGER     SAVOPT
      COMMON      /SAVOPT/ SAVOPT
      DATA        EPS /0.00001/
*
*
*----------Open flow network file
*
      IF (FILNAM .NE. ' ') THEN
                                      FLAG = .TRUE.
                                         IU = 9
                  CALL IO_OPFIL(IU, 1, FILNAM, 'Enter network file: ')
                                     DO I = 1, NTLS
                                        READ (IU, *)
                                           ENDDO
      ELSE
                                     FLAG = .FALSE.
                                        IU = IN
                                       CTERM = ' '
                                DO WHILE (CTERM .EQ. ' ')
                             READ (IU, '(A)', END = 100) CTERM
                                           ENDDO
                   CALL STR_DIVD(CTERM, I, COLSTR, LDCOL, 0, ' ,')
                               READ (COLSTR(2), '(I2)') N
                IF (CN(COLSTR(1), 'PART', 1) .AND. N .EQ. PARTNO) THEN
                            READ (IU, *) !SKIP THE VARIABLE LIST
                                            ELSE
                 IF (CN(COLSTR(1), 'PART', 1) .AND. N .GT. PARTNO )THEN
                                         BACKSPACE(1)
                                           GOTO 100
                                            ENDIF
                                      PRINT *, CHAR(7)
                       PRINT *, '***ERROR***ERROR IN FLOW NETWORK DATA'
                                       STOP !CALL EXIT
                                           ENDIF
      ENDIF
*
*--------------------Initialize some array
*
      DO I = 1, LDSTRM
                                       DO J = 0, 4
                                     STRMCF(I, J) = 0.0
                                           ENDDO
                                       DO J = 0, 6
                                      STRMAR(I, J) = 0
                                           ENDDO
      ENDDO
*
      IF (SAVOPT .EQ. 0 .OR. SAVOPT .EQ. 1) THEN
                                     WRITE (OU, 800)
 800     FORMAT (//, 'Part 2: Network flow configuration',
     &         /, '==================================', /)
                                     WRITE (OU, 801)
 801     FORMAT( T26, '      Lower       Upper',
     &     T58, '   Initial                                   Evaporation',
     &     /, T26, '  boundary  boundary                 ',
     &     T58, '   storage  Travel-                          coefficient',
     &     /, T25,'(cubic feet (cubic feet Penalty  ',
     &     T60, '  (acre-      time  Weighting Seepage       (inches  ',
     &     /, T25,'per second) per second) coefficient',
     &     T60, '   feet)     (day)    factor  coefficient   per day)',
     &     /, '---------', T12, '-------',
     &        T25,'----------  ----------- -----------',
     &     T60, ' --------  --------  -------- -----------   ----------')
      ENDIF
      NARCND = 0
      OUPND = 0
      ODWND = 0
      NDWAR = 0
      NRND = 0
 5    CONTINUE
      READ (IU, '(A)', END = 100) CTERM
      IF (CTERM .EQ. ' ') GOTO 5
      CALL STR_DIVD(CTERM, NCOL, COLSTR, LDCOL, 0, ' ,')
      IF (CN(COLSTR(1), 'FINISH', 1)) GOTO 100
      STRM = .FALSE.
      IF (NCOL .GT. 5) THEN
                                     DO I = 6, NCOL
                             READ (COLSTR(I), '(F10.0)') RTERM
                               IF (ABS(RTERM) .GT. EPS) THEN
                                         STRM = .TRUE.
                                            ENDIF
                                           ENDDO
      ENDIF
*
*---------------Upstream node and downstream node
```

```
*
      CALL NAMNUM(LDND, NDNAM, COLSTR(1), UPND, 0, ERR)
      IF (ERR) THEN
                                       NNODS = NNODS + 1
                                  NDNAM(NNODS) = COLSTR(1)
                              CALL STR_CORS(NDNAM(NNODS), 1)
                                     NDTYP(NNODS) = 2
                                       UPND = NNODS
      ENDIF
      CALL NAMNUM(LDND, NDNAM, COLSTR(2), DWND, 0, ERR)
      IF (ERR) THEN
                                       NNODS = NNODS + 1
                                  NDNAM(NNODS) = COLSTR(2)
                              CALL STR_CORS(NDNAM(NNODS), 1)
                                     NDTYP(NNODS) = 2
                                       DWND = NNODS
      ENDIF
      IF (UPND .NE. OUPND) THEN
                                        NRND = NRND + 1
                                 PTDWAR(UPND, 1) = NDWAR + 1
                                       OUPND = UPND
                                         ODWND = 0
      ENDIF
*
      IF (DWND .NE. ODWND ) THEN
                                          ZONE = 1
                                         UZONE = 0
                                         LZONE = 0
                                       ODWND = DWND
      ELSE
                                        ZONE = ZONE + 1
      ENDIF
*
*------------------Determine flow bounds and flow direction.
*
      READ (COLSTR(3), '(F10.0)') LBND
      READ (COLSTR(4), '(F10.0)') UBND
      READ (COLSTR(5), '(I10)') CST
      IF (ZONE .EQ. 1) THEN
                                         ITOJ = .TRUE.
                                          LVL1 = UBND
                                    IF (LBND .GE. 0) THEN
                                 L = NINT(LBND * C * PERD * XF)
                                              ELSE
                                               L = 0
                                              ENDIF
                              U = NINT(UBND * C * PERD * XF)
                                         IF (STRM) THEN
                                       NARCND = NARCND + 1
                                       MDND = LDND - NARCND
                                       NDNAM(MDND) = 'STRM'
                      WRITE(NDNAM(MDND)(5:10), '(I3.3,I3.3)')UPND, DWND
                                              ENDIF
      ELSE
                                  IF (UBND .GE. LVL1) THEN
                                          ITOJ = .TRUE.
                                        UZONE = UZONE + 1
                                              ELSE
                                         ITOJ = .FALSE.
                                        LZONE = LZONE + 1
                                              ENDIF
                                              L = 0
                         U = NINT((UBND - LBND) * C * PERD * XF)
      ENDIF
*
*----------------Create flow arcs.
*
      IF (ITOJ) THEN
                                        IF (STRM) THEN
                                        TYP = 110 + UZONE
                                        NSTRM = NSTRM + 1
                                     STRMAR(NSTRM, 0) = MDND
                      CALL ARCVAL(NARCS, II, JJ, LO, HI, COST, ARTYP,
     &                 LDARC, UPND, MDND, L, U, CST, TYP)
                                        NDWAR = NDWAR + 1
                                       DWAR(NDWAR) = NARCS
                                     PTDWAR(UPND, 2) = NDWAR
                                    STRMAR(NSTRM, 1) = NARCS
                      CALL ARCVAL(NARCS, II, JJ, LO, HI, COST, ARTYP,
     &                 LDARC, MDND, DWND, L, U, CST, TYP)
                                    STRMAR(NSTRM, 6) = NARCS
                                          DO I = 6, NCOL
                                              J = I - 6
                          READ (COLSTR(I), '(F15.0)') STRMCF(NSTRM, J)
                                              ENDDO
                                              ELSE
                                        TYP = 100 + UZONE
                      CALL ARCVAL(NARCS, II, JJ, LO, HI, COST, ARTYP,
     &                 LDARC, UPND, DWND, L, U, CST, TYP)
                                        NDWAR = NDWAR + 1
                                       DWAR(NDWAR) = NARCS
                                     PTDWAR(UPND, 2) = NDWAR
                                              ENDIF
      ELSE
```

```
                                          IF (STRM) THEN
                                          TYP = -110 - LZONE
                                          NSTRM = NSTRM + 1
                                        STRMAR(NSTRM, 0) = MDND
                         CALL ARCVAL(NARCS, II, JJ, LO, HI, COST, ARTYP,
     &                    LDARC, MDND, UPND, L, U, CST, TYP)
                                          NDWAR = NDWAR + 1
                                         DWAR(NDWAR) = -NARCS
                                        PTDWAR(UPND, 2) = NDWAR
                                       STRMAR(NSTRM, 1) = -NARCS
                         CALL ARCVAL(NARCS, II, JJ, LO, HI, COST, ARTYP,
     &                    LDARC, DWND, MDND, L, U, CST, TYP)
                                       STRMAR(NSTRM, 6) = -NARCS
                                            DO I = 6, NCOL
                                              J = I - 6
                           READ (COLSTR(I), '(F15.0)') STRMCF(NSTRM, J)
                                              ENDDO
                                              ELSE
                                          TYP = -110 - LZONE
                         CALL ARCVAL(NARCS, II, JJ, LO, HI, COST, ARTYP,
     &                    LDARC, DWND, UPND, L, U, CST, TYP)
                                          NDWAR = NDWAR + 1
                                         DWAR(NDWAR) = -NARCS
                                        PTDWAR(UPND, 2) = NDWAR
                                              ENDIF
      ENDIF
*
*             if lower bound of the current flow range is less than zero,
*             then the current arc is control arc and the lower flow bound
*             will be determined.
*
      IF (LBND .LT. 0) THEN
                                         NCTAR = NCTAR + 1
                                      CTARFW(NCTAR, 1) = NARCS
                                        CTARFW(NCTAR, 2) = 0
                    CTARFW(NCTAR, 3) = - NINT(LBND * C * REAL(PERD) * XF)
      ENDIF
*
      IF (SAVOPT .EQ. 0 .OR. SAVOPT .EQ. 1) THEN
                                          IF (STRM) THEN
                   WRITE(OU, 805) NDNAM(UPND), NDNAM(DWND), LBND, UBND, CST,
     &                    (STRMCF(NSTRM, I), I = 0, 4)
 805        FORMAT(2A12, T26, 2F10.2, I10, 5F10.2)
                                              ELSE
                    WRITE(OU, 806) NDNAM(UPND), NDNAM(DWND), LBND, UBND, CST
 806        FORMAT(2A12, T26, 2F10.2, I10)
                                              ENDIF
      ENDIF
*
*             next arc
      GOTO 5
 100  CONTINUE
      IF (FLAG) THEN
                                           CLOSE (IU)
      ENDIF
      RETURN
      END
*=================================================================
* Name:       strm_dat
* Purpose:    Read stream geometry data.
* Author:     Xiaodong Jian
* Date:       12/05/95
*=================================================================
      SUBROUTINE STRM_DAT(NDNAM, LDND, II, JJ, LDARC,
     &                    NSTRM, STRMAR, LDSTRM,
     &                    NSTR, STRDIR, STRDAT, LDSTR,
     &                    FILNAM, NTLS, IN, OU, PARTNO)
      IMPLICIT   NONE
      INTEGER    LDND
      CHARACTER  NDNAM(LDND)*(*), FILNAM*(*)
      INTEGER    LDARC, II(LDARC), JJ(LDARC), NTLS
      INTEGER    NSTRM, LDSTRM, STRMAR(LDSTRM, 0:6)
      INTEGER    NSTR, LDSTR, STRDIR(LDSTR, 3)
      REAL       STRDAT(LDSTR, 9)
*
      INTEGER    LDCOL
      PARAMETER  (LDCOL = 15)
      CHARACTER  CTERM*150, COLSTR(LDCOL)*15
      INTEGER    ND, UPND, DWND
      INTEGER    I, J, K, N, IN, IU, OU, PARTNO
      LOGICAL    ERR, CN, FLAG
      INTEGER    SAVOPT
      COMMON     /SAVOPT/ SAVOPT
*
      NSTR = 0
*
*------------open data file and read title lines
*
      IF (FILNAM .NE. ' ') THEN
                                           FLAG = .TRUE.
                                              IU = 9
             CALL IO_OPFIL(IU,1,FILNAM,'ENTER THE STREAM GEOMETRIC FILE: ')
                                           DO I = 1, NTLS
                                       READ (IU, *, END = 100)
```

```
                                  ENDDO
      ELSE
                              FLAG = .FALSE.
                                 IU = IN
                               CTERM = ' '
                        DO WHILE (CTERM .EQ. ' ')
                      READ (IU, '(A)', END = 999) CTERM
                                  ENDDO
                CALL STR_DIVD(CTERM, I, COLSTR, LDCOL, 0, ' ,')
                         READ (COLSTR(2), '(I2)') N
             IF (CN(COLSTR(1), 'PART', 1) .AND. N .EQ. PARTNO) THEN
                     READ (IU, *) !SKIP THE VARIABLE LIST
                                  ELSE
              IF (CN(COLSTR(1), 'PART', 1) .AND. N .GT. PARTNO )THEN
                               BACKSPACE(1)
                                 GOTO 100
                                 ENDIF
                              PRINT *, CHAR(7)
                   PRINT *, '***ERROR***ERROR IN STREAM DATA'
                              STOP !CALL EXIT
                                  ENDIF
      ENDIF
*
      IF (SAVOPT .EQ. 0 .OR. SAVOPT .EQ. 1) THEN
                              WRITE(OU, 800)
 800     FORMAT(//, 'PART 3: Canal geometry data',
     &           /, '===========================',
     & /,                        T25, '                                     ',
     & '                               Riverbed                          ',
     & /,                        T25, '                                     ',
     & '                               hydraulic                         ',
     & /,                        T25, '                      Canal          ',
     &'     Bottom              Canal  conductivity  Riverbed   Entry   ',
     & /,'F-node', T13, 'T-node', T25, ' Roughness    lendth    Canal',
     & '      width      Side    depth        (feet  thickness elevation',
     &    /, 'name', T13, 'name', T24,'coefficient   (feet)     slope',
     & '      (feet)     slope    (feet)      per day)  (feet)    (feet)',
     &    /, '----', T13, '----', T24,'----------- --------- ---------',
     & ' --------- --------- ---------  --------- --------- ---------')
      ENDIF
*
*-------------Read data
*
 5    CONTINUE
      READ (IU, '(A)', END = 100) CTERM
      IF (CTERM .EQ. ' ') GOTO 5
      CALL STR_DIVD(CTERM, K, COLSTR, LDCOL, 0, ' ,')
      IF (CN(COLSTR(1), 'FINISH', 1)) GOTO 100
      IF (K .LT. 7) THEN
                             PRINT *, CHAR(7)
             PRINT *, '***ERROR*** INCOMPLETE DATA SET IN GEOMETRIC FILE'
                             STOP !CALL EXIT
      ENDIF
      NSTR = NSTR + 1
      DO J = 1, K
                            IF (J .LE. 2) THEN
                 CALL NAMNUM(LDND, NDNAM, COLSTR(J), ND, 0, ERR)
                            STRDIR(NSTR, J) = ND
                                  ELSE
                   READ(COLSTR(J), '(F15.0)') STRDAT(NSTR, J-2)
                                  ENDIF
      ENDDO
*
      IF (SAVOPT .EQ. 0 .OR. SAVOPT .EQ. 1) THEN
                 WRITE (OU, 805) (NDNAM(STRDIR(NSTR,J)),J=1,2),
     &           (STRDAT(NSTR,J), J=1,9)
 805     FORMAT(A12, A12, F10.4, F10.2, F10.6, F10.2, F10.6, 4F10.2)
      ENDIF
      GOTO 5
 100  CONTINUE
      IF (FLAG) CLOSE(IU)
*
*           Find the middle stream node between the upstream node
*           and downstream node.
*
      DO 20 I = 1, NSTR
                             DO J = 1, NSTRM
                        UPND = II(IABS(STRMAR(J, 1)))
                        DWND = JJ(IABS(STRMAR(J, 6)))
            IF(UPND .EQ. STRDIR(I, 1) .AND. DWND .EQ. STRDIR(I, 2))THEN
                          STRDIR(I, 3) = STRMAR(J, 0)
                                 GOTO 20
                                 ENDIF
                                 ENDDO
                            STRDIR(I, 3) = 0
 20   CONTINUE
 999  RETURN
      END
*=====================================================================
* Name:       strm_rout
* Purpose:    Channel Routing.
*=====================================================================
      SUBROUTINE STRM_ROUT(
     &           NARCS, II, JJ, HI, LO, COST, FLOW, ARTYP, MXARC,
```

```
     &           NSTRM, STRMCF, STRMAR, LDSTRM,
     &           NSTR, STRDIR, STRDAT, LDSTR,
     &           NGWND, GWND, GWLVL, LDGWND,
     &           NEV, EVND, LDEV, EVIDX, EVTB, LDEVTB,
     &           SKSC, PERD, XF)
      IMPLICIT   NONE
      INTEGER    NARCS, MXARC, NSTRM, LDSTRM, SKSC
      INTEGER    II(MXARC), JJ(MXARC), HI(MXARC), LO(MXARC),
     &           COST(MXARC), ARTYP(MXARC), FLOW(MXARC)
      REAL       STRMCF(LDSTRM, 0:4)
      INTEGER    STRMAR(LDSTRM, 0:6)
      INTEGER    NSTR, LDSTR, STRDIR(LDSTR, 3)
      REAL       STRDAT(LDSTR, 9)
      INTEGER    NGWND, LDGWND, GWND(LDGWND)
      REAL       GWLVL(LDGWND)
*
      INTEGER    NEV, LDEV, EVND(LDEV,3), EVIDX, LDEVTB
      REAL       EVTB(0:LDEVTB, LDEV), EV, AREA
*
      REAL       PERD, XF
*
      INTEGER    I, J, ND, ND1, ND2, ARC, ISGN, S, VAL, ARC1, ARC2
      INTEGER    OND1, OND2
*
      REAL       K, X, IFLW, OFLW, ROUGH, LENGTH, SLOPE, WIDTH,
     &           ML, MR, HMAX, DISCH
      INTEGER    IERR
      REAL       H, HGW, KY, DY, ZB, FAC
*
      LOGICAL    FLAG, EVFLAG
      REAL       EPS
      DATA       EPS /0.0001/
*
      OND1 = 0
      OND2 = 0
      DO 100 S = 1, NSTRM
*
*-----------------arc and node infomation
*
                                    ND  = STRMAR(S, 0)
                                   ARC1 = STRMAR(S, 1)
                                   ARC2 = STRMAR(S, 6)
                                  IF (ARC1 .GT. 0) THEN
                                       ND1 = II(ARC1)
                                       ND2 = JJ(ARC2)
                                          ELSE
                                      ND1 = JJ(-ARC1)
                                      ND2 = II(-ARC2)
                                         ENDIF
*
*-----------The channel surface evaporation occurs only  in the normal flow
*           range because flow are supposed in the normal flow range.
*
                          IF (ND1 .NE. OND1 .OR. ND2 .NE. OND2) THEN
                                        OND1 = ND1
                                        OND2 = ND2
                                     EVFLAG = .TRUE.
                                          ELSE
                                    EVFLAG = .FALSE.
                                         ENDIF
*
*----------------channel geometric data
*
                                     FLAG = .FALSE.
                                     DO I = 1, NSTR
                                        AREA = 0.0
                                IF (ND .EQ. STRDIR(I, 3)) THEN
                                    ROUGH  = STRDAT(I, 1)
                                    LENGTH = STRDAT(I, 2)
                                    SLOPE  = STRDAT(I, 3)
                                    WIDTH  = STRDAT(I, 4)
                                    ML     = STRDAT(I, 5)
                                         MR     = ML
                                    HMAX   = STRDAT(I, 6)
                                     KY    = STRDAT(I, 7)
                                     DY    = STRDAT(I, 8)
                         ZB    = STRDAT(I, 9) - 0.5 * LENGTH * SLOPE
                                        FLAG = .TRUE.
*
*----------------Using Manning's equation to estimate water depth and width
*
                            IFLW = FLOW(ABS(ARC1)) / PERD * 43560 / XF
                            OFLW = FLOW(ABS(ARC2)) / PERD * 43560 / XF
                              DISCH = 0.5 * (IFLW + OFLW) / 86400.0
                       CALL CHDEP(ROUGH, DISCH, SLOPE, WIDTH, ML, MR, HMAX, H,
     &                    1.486, 0.001, 200, IERR)
                                  WIDTH = WIDTH + ML*H + MR*H
                                     IF (H .GT. EPS) THEN
                                    AREA = WIDTH * LENGTH
                                           ENDIF
                                    IF (IERR .EQ. 0) THEN
                                        CONTINUE
                                 ELSE IF (IERR .EQ. 1) THEN
                       PRINT *, '***WARNING*** H > HMAX FOR CHANNEL: ', S
```

```
                                        CONTINUE
                                            ENDIF
                                           GOTO 10
                                          ENDIF
                                         ENDDO
 10      CONTINUE
*
*--------A. Channel seepage
*
                                        VAL = 0
                            IF (STRMCF(S, 3) .GT. EPS) THEN
                                   ARC = STRMAR(S, 1)
                   VAL = NINT(ISGN(ARC) * STRMCF(S, 3) * FLOW (IABS(ARC)))
                     ELSE IF (STRMCF(S, 3) .LT. -EPS .AND. FLAG) THEN
*
*-------------2. Estimate Groundwater level if available
*
                                  IF (NGWND .GT. 0) THEN
                                         HGW = 0.0
                                           J = 0
                                      DO I = 1, NGWND
                      IF (ND1 .EQ. GWND(I) .OR. ND2 .EQ. GWND(I)) THEN
                                         J = J + 1
                                   HGW = HGW + GWLVL(I)
                                          ENDIF
                                            ENDDO
                                     IF (J .GT. 0) THEN
                                  HGW = HGW / REAL(J)
                                            ENDIF
                                          ENDIF
*
*-------------3. calculate seepages
*
                          IF (ZB .GT. EPS .AND. HGW .GT. ZB) THEN
                                      H = H + ZB - HGW
                                          ENDIF
                 VAL = NINT(KY * H / DY * LENGTH * WIDTH * PERD/43560. * XF)
                                         ENDIF
 20      CONTINUE
                               IF (ABS(VAL) .GT. EPS) THEN
                                   IF (VAL .GT. 0) THEN
                        CALL ARCVAL(NARCS, II, JJ, LO, HI, COST, ARTYP,
     &                    MXARC, ND, SKSC, VAL, VAL, 0, 10)
                                ELSE IF (VAL .LT. 0) THEN
                        CALL ARCVAL(NARCS, II, JJ, LO, HI, COST, ARTYP,
     &                    MXARC, SKSC, ND, -VAL, -VAL, 0, -10)
                                          ENDIF
                             STRMAR(S, 3) = ISGN(VAL) * NARCS
                                         ELSE
                                    STRMAR(S, 3) = 0
                                         ENDIF
*
*--------B. Channel water surface evaporation. Surface evaporation coeffiecints
*          are obtained in two ways: (1) If the channel water surface
*          evaporation coefficient is specified, this value will be used
*          for whole simulation period, and (2) If the ev value is not specified
*          the values in the upstream and downstream nodes will be averaged.
*
                     IF (ABS(STRMCF(S, 4)) .GT. EPS .AND. EVFLAG .AND.
     &       AREA .GT. EPS) THEN
                                         EV = 0.0
                              IF (STRMCF(S, 4) .GT. 0) THEN
                                     EV = STRMCF(S, 4)
                                ELSE IF (NEV .GT. 0) THEN
                                           J = 0
                                        DO I = 1, NEV
                  IF (ND1 .EQ. EVND(I, 1) .OR. ND2 .EQ. EVND(I,1)) THEN
                                         J = J + 1
                                IF (EVND(I,2) .EQ. 0) THEN
                                  FAC = 1.0 / 25.4
                             ELSE IF (EVND(I,2) .EQ. 2) THEN
                                     FAC = 12.0
                                         ELSE
                                      FAC = 1.0
                                         ENDIF
                                IF (EVND(I,3) .NE. 0) THEN
                             EV = EV + EVTB(0, I) * FAC
                                         ELSE
                           EV = EV + EVTB(EVIDX, I) * FAC
                                         ENDIF
                                       ENDIF
                                           ENDDO
                                    IF (J .GT. 0) THEN
                                     EV = EV / J
                                           ENDIF
                                         ENDIF
                                  IF (EV .GT. EPS) THEN
                      VAL = NINT(EV * PERD / 12.0 * AREA / 43560.0 * XF)
                        CALL ARCVAL(NARCS, II, JJ, LO, HI, COST, ARTYP,
     &                MXARC, ND, SKSC, VAL, VAL, 0, 11)
                               STRMAR(S, 4) = ISGN(VAL) * NARCS
                                         ELSE
                                    STRMAR(S, 4) = 0
                                         ENDIF
```

```
                                ENDIF
*
*--------C. Channel Routing
*
                                IF (ABS(STRMCF(S,1)) .GT. EPS)  THEN
*
*-----------Initial storage arc
*
                                    VAL = NINT(STRMCF(S, 0) * XF)
                                        IF (VAL .GT. 0) THEN
                                CALL ARCVAL(NARCS, II, JJ, LO, HI, COST, ARTYP,
     &                           MXARC, SKSC, ND, VAL, VAL, 0, 8)
                                    STRMAR(S, 2) = ISGN(VAL) * NARCS
                                      ELSE IF (VAL .LT. 0) THEN
                                CALL ARCVAL(NARCS, II, JJ, LO, HI, COST, ARTYP,
     &                           MXARC, ND, SKSC, -VAL, -VAL, 0, -8)
                                    STRMAR(S, 2) = ISGN(VAL) * NARCS
                                              ELSE
                                          STRMAR(S, 2) = 0
                                              ENDIF
*
*-----------Final Storage arc
*
                                        ARC  = STRMAR(S, 1)
                IFLW = FLOW(ABS(ARC1)) / PERD * 43560 / XF      ! FLOW RATE IN CFD
                            OFLW = FLOW(ABS(ARC2)) / PERD * 43560 / XF
                                           K = STRMCF(S,1)
                                           X = STRMCF(S,2)
                   VAL = NINT(K * (X * IFLW + (1.0 - X) * OFLW)/43560.0 * XF)
                                       VAL = ISGN(ARC) * VAL
*
                                        IF (VAL .GT. 0) THEN
                                CALL ARCVAL(NARCS, II, JJ, LO, HI, COST, ARTYP,
     &                           MXARC, ND, SKSC, VAL, VAL, 0, 9)
                                    STRMAR(S, 5) = ISGN(VAL) * NARCS
                                      ELSE IF (VAL .LT. 0) THEN
                                CALL ARCVAL(NARCS, II, JJ, LO, HI, COST, ARTYP,
     &                           MXARC, SKSC, ND, -VAL, -VAL, 0, -9)
                                    STRMAR(S, 5) = ISGN(VAL) * NARCS
                                              ELSE
                                          STRMAR(S, 5) = 0
                                              ENDIF
                                            ENDIF
 100  CONTINUE
*
 99   RETURN
      END
*=====================================================================
* Name:         strm_kx
* Purpose:      Estimate routing coeffients: travel time K and weighting
*               factor x if these coefficient are not specified.
*=====================================================================
      SUBROUTINE STRM_KX(NSTRM, STRMAR, STRMCF, LDSTRM,
     &                   NSTR,  STRDIR, STRDAT, LDSTR)
      IMPLICIT   NONE
      INTEGER    NSTRM, LDSTRM, NSTR, LDSTR, METHOD
      INTEGER    STRMAR(LDSTRM,0:6),  STRDIR(LDSTR, 3)
      REAL       STRMCF(LDSTRM, 0:3), STRDAT(LDSTR, 9)
*
      REAL       N, L, S, W, M, D, R, P
      REAL       K, X, EPS
      INTEGER    I, J, ND
      DATA       EPS/0.00001/
*
      DO 100 J = 1, NSTRM
                                       ND = STRMAR(J, 0)
                              IF (STRMCF(J, 1) .LT. EPS) THEN
                                          DO I = 1, NSTR
                                IF (ND .EQ. STRDIR(I, 3)) GOTO 50
                                              ENDDO
                                            GOTO 100
 50         CONTINUE
                                  METHOD = -NINT(STRMCF(J, 1))
                                         N = STRDAT(I, 1)
                                         L = STRDAT(I, 2)
                                         S = STRDAT(I, 3)
                                         W = STRDAT(I, 4)
                                         M = STRDAT(I, 5)
                                         D = STRDAT(I, 6)
*
                                     IF (METHOD .EQ. 1) THEN
                       K = 0.7872 * L  * W**0.4 * N**0.6 / S**0.3 / 86400.0
                                              X = 0.0
                                  ELSE IF (METHOD .EQ. 2) THEN
                               P = W + 2.0 * D * SQRT(1.0 + M * M)
                                       R = (W + M * D) * D / P
                       K = L * N / (1.49 * (R**0.6667) * SQRT(S)) / 86400.0
                              X = 1.49 * SQRT(S) / (N * P**0.6667)
                                              ENDIF
                                        STRMCF(J, 1) = K
                                        STRMCF(J, 2) = X
                                            ENDIF
 100  CONTINUE
      RETURN
```

```
      END
*=================================================================
* Name:       tws_dat
* Purpose:    Get seasonal target water demands.
*=================================================================
      SUBROUTINE TWS_DAT(NNODS, NDNAM, LDND, NWSND, WSND, LDWS, NPER,
     &                   WSTB, LDWSTB, WSUNIT, WSFLAG, PN, FILNAM, NTLS,
     &                   IN, OU, UNITNM, LDUNIT, CTERM, COLSTR, LDCOL)
      IMPLICIT   NONE
      INTEGER    NNODS, LDND, LDCOL
      CHARACTER*(*) NDNAM(LDND), FILNAM, CTERM, COLSTR(LDCOL)
      INTEGER    NWSND, LDWS, WSND(LDWS, 3), WSUNIT, NPER, LDWSTB
      REAL       WSTB(0:LDWSTB, LDWS)
      INTEGER    NTLS, IN, PN
      LOGICAL    WSFLAG
*
      INTEGER    I, NREC, IU, OU
      LOGICAL    FLAG, ENDFIL
      INTEGER    LDUNIT
      CHARACTER  UNITNM(0:LDUNIT)*(*)
*
*--------------Open data file and skip title lines
*
      IF (FILNAM .NE. ' ') THEN
                                  FLAG = .TRUE.
                                    IU = 9
            CALL IO_OPFIL(IU, 1, FILNAM, 'Enter water demand file: ')
                  DO I = 1, NTLS      !SKIP TITLE LINES
                        READ (IU, *, END = 99)
                                    ENDDO
      ELSE
                                  FLAG = .FALSE.
                                    IU = IN
               CALL PNCK(PN, IU, ENDFIL, CTERM, COLSTR, LDCOL)
                        IF (ENDFIL) GOTO 99
      ENDIF
*
*--------------Read units, nonal names, and data from a file and
*              print these information into general output file.
*
      CALL DATTB(NNODS, NDNAM, LDND, NWSND, WSND, LDWS, WSUNIT,
     &           NREC, WSTB, LDWSTB, WSFLAG,
     &           UNITNM, LDUNIT, FILNAM, IU, OU, PN, ENDFIL,
     &           'target water demands',
     &           CTERM, COLSTR, LDCOL)
      IF (NREC .LT. NPER) THEN
                          WRITE (OU, 805) NPER, NREC
 805     FORMAT('***WARNING*** NO. OF SEASONAL RECORDS ARE LESS',
     &          ' THAN NO. OF SEASONS.',
     &          '    NO. OF RECORDS: ', I3,
     &          '    NO. OF SEASONS: ', I3)
      ENDIF
*
 99   CONTINUE
      IF (FLAG) CLOSE(IU)
      RETURN
      END
*=================================================================
* Name:       tws_arc
* Purpose:    Read current target water demand and create
*             corresponding TWS arcs.
*=================================================================
      SUBROUTINE TWS_ARC(NARC, II, JJ, LO, HI, COST, ARTYP, LDARC,
     &                  NWSND, WSND, LDWS, WSTB, LDWSTB, WSFIL,
     &                  NDXAR, LDXAR, MTH, SKSC, PERD, MXCST, XF, IU,
     &                  CTERM, COLSTR, LDCOL)
      IMPLICIT   NONE
      INTEGER    NARC, LDARC,
     &           II(LDARC), JJ(LDARC), LO(LDARC), HI(LDARC),
     &           COST(LDARC), ARTYP(LDARC)
      INTEGER    NWSND, LDWS, WSND(LDWS, 3), LDWSTB
      REAL       WSTB(0:LDWSTB, LDWS)
      LOGICAL    WSFIL
      INTEGER    LDXAR, NDXAR(LDXAR, 6)
*
      INTEGER    IU, MTH, SKSC, MXCST
      REAL       PERD, XF
*
      INTEGER    LDCOL
      CHARACTER  CTERM*(*), COLSTR(LDCOL)*(*)
*
      INTEGER    I, J, N, ND, NNDS, NV
      REAL       UC(0:2)
      LOGICAL    TDFLAG
*
*--------------Data unit conversion factor (ac-ft)
*
      UC(0) = 1.0
      UC(1) = PERD * 86400.0 / 43560.0
      UC(2) = PERD / 43560.0
*
*----------------Read time-dependent target water demands from a file.
*
      TDFLAG = .FALSE.
```

```
      IF (WSFIL) THEN
                               READ (IU, '(A)', END = 50) CTERM
                     CALL  STR_DIVD(CTERM, NNDS, COLSTR, LDCOL, 0, ' ,')
                                        TDFLAG = .TRUE.
      ENDIF
 50   CONTINUE
*
*---------------Create the target water demand arcs
*
      DO I = 1, NWSND
                                        ND = WSND(I, 1)
                                  IF (WSND(I, 3) .EQ. 0) THEN
                                              N = MTH
                                              ELSE
                                               N = 0
                                         IF (TDFLAG) THEN
                                     IF (WSND(I, 3) .LT. 0) THEN
                            READ(COLSTR(NNDS), '(F10.0)') WSTB(0,I)
                                                ELSE
                                       J = WSND(I, 3) + 1
                             READ (COLSTR(J), '(F10.0)') WSTB(0, I)
                                                ENDIF
                                               ELSE
                                          WSTB(0, I) = 0.0
                                               ENDIF
                                              ENDIF
                          NV = NINT(WSTB(N,I) * UC(WSND(I,2)) *  XF)
                        CALL ARCVAL(NARC, II, JJ, LO, HI, COST, ARTYP,
     &                LDARC, ND, SKSC, NV, NV, 0, 6)
                                      NDXAR(ND, 5) = NARC
                        CALL ARCVAL(NARC, II, JJ, LO, HI, COST, ARTYP,
     &                LDARC, SKSC, ND,  0, NV, MXCST*10, -6)
                                      NDXAR(ND, 6) = NARC
      ENDDO
 99   RETURN
      END
*=====================================================================
* Name:         NVAR
* Purpose:      GET THE NET INFLOW TO THE ALL NODES AND CREATE THE NET
*               FLOW ARCS
*=====================================================================
      SUBROUTINE NVAR(NNODS, NDNAM, NDTYP, NDSEQ, LDND,
     &                NARCS, II, JJ, HI, LO, COST, ARTYP, LDARC,
     &                MTH, IN_IFW, LAST, MXR,
     &                INST, RC, SKSC,
     &                NDXAR, PERD, XF,
     &                NIFW, IFWND, LDIFW,
     *                PTRE, LDRES, REAR, REZN, LDREAR,
     *                NRCND, RCND, LDRC, RCTB, LDRCTB, RCFLAG, RCFIL,
     *                IN_RC,
     &                APRX, IOUT1,
     &                RPOOL, LDPL, CTERM, COLSTR, LDCOL)
      IMPLICIT   NONE
      INTEGER    LDARC, LDND, MXR, LDRCTB
      INTEGER    II(LDARC), JJ(LDARC), HI(LDARC), LO(LDARC),
     &           COST(LDARC), ARTYP(LDARC)
      INTEGER    NNODS, SKSC, NARCS, IOUT1
      REAL       RC(MXR)
      REAL       APRX,  INST(MXR)
      LOGICAL    LAST
      CHARACTER  NDNAM(LDND)*(*)
      INTEGER    NDTYP(LDND), NDSEQ(LDND), NDXAR(LDND, 6)
      INTEGER    MTH, IN_IFW, RES
      INTEGER    NIFW, LDIFW, IFWND(LDIFW, 3)
      INTEGER    LDRES, PTRE(LDRES), LDREAR, REAR(LDREAR)
      REAL       REZN(LDREAR)
      INTEGER    NRCND, LDRC, RCND(LDRC),IN_RC
      REAL       RCTB(0:LDRCTB, LDRC)
      LOGICAL    RCFLAG, RCFIL
*
      INTEGER    LDPL, LDCOL
      REAL       RPOOL(LDPL)
      CHARACTER  CTERM*(*), COLSTR(LDCOL)*(*)
      LOGICAL    DEBUG
*
      REAL       PERD, XF
*
      REAL       NVS
      INTEGER    NV
      INTEGER    I, J, K, L, NNDS
      REAL       UC(0:2)
*
      DEBUG = .FALSE.
      LAST = .FALSE.
*
*           read Incremental inflow to each node for current time step.
*
      DO I = 1, NNODS
                                        RPOOL(I) = 0.0
      ENDDO
*
      UC(0) = 1.0
      UC(1) = PERD * 86400.0 / 43560.0
      UC(2) = PERD / 43560.0
```

```
*
      READ (IN_IFW, '(A)', END = 50) CTERM
      CALL  STR_DIVD(CTERM, NNDS, COLSTR, LDCOL, 0, ' ,')
*
      DO I = 1, NIFW
                          IF (IFWND(I, 3) .GT. 0) THEN
                                J = IFWND(I, 3) + 1
                   READ (COLSTR(J), '(F10.0)') RPOOL(IFWND(I,1))
                       ELSE IF (IFWND(I, 3) .LT. 0) THEN
                  READ (COLSTR(NNDS), '(F10.0)') RPOOL(IFWND(I,1))
                                    ENDIF
              RPOOL(IFWND(I,1)) = RPOOL(IFWND(I,1)) * UC(IFWND(I, 2))
      ENDDO
*
*-------------Modify the rule cure
*
 50   CONTINUE
      IF (RCFLAG) THEN
                   CALL RC_ARC(NDNAM, NDSEQ, LDND, HI, LDARC,
     &                PTRE, RC, LDRES, REAR, REZN, LDREAR,
     &                NRCND, RCND, LDRC, RCTB, LDRCTB, RCFIL,
     &                MTH, XF, IN_RC, CTERM, COLSTR, LDCOL)
      ENDIF
*
*---------------Create the NV arcs
*
      DO 60 K = 1, NNODS
      IF (NDTYP(K) .EQ. 1) THEN
                                RES = NDSEQ(K)
                 NVS = (INST(RES) + RPOOL(K) - RC(RES)) * XF
                             IF (NVS .GT. 0.) THEN
                                NV = NVS + APRX
                 CALL ARCVAL(NARCS, II, JJ, LO, HI, COST, ARTYP,
     &                  LDARC, SKSC, K, NV, NV, 0, 3)
                               NDXAR(K, 1) = NARCS
                         ELSE IF (NVS .LT. 0.) THEN
                                NV = NVS - APRX
                 CALL ARCVAL(NARCS, II, JJ, LO, HI, COST, ARTYP,
     &                  LDARC, K, SKSC, -NV, -NV, 0, -3)
                              NDXAR(K, 1) = -NARCS
                                    ENDIF
      ELSE IF (NDTYP(K) .EQ. 2) THEN
                              NVS = RPOOL(K) * XF
                             IF (NVS .GT. 0.) THEN
                                NV = NVS + APRX
                 CALL ARCVAL(NARCS, II, JJ, LO, HI, COST, ARTYP,
     &                  LDARC, SKSC, K, NV, NV, 0, 7)
                               NDXAR(K, 1) = NARCS
                         ELSE IF (NVS .LT. 0.) THEN
                                NV = NVS - APRX
                 CALL ARCVAL(NARCS, II, JJ, LO, HI, COST, ARTYP,
     &                  LDARC, K, SKSC, -NV, -NV, 0, -7)
                               NDXAR(K,1) = -NARCS
                                    ENDIF
      ENDIF
 60   CONTINUE
      IF (.NOT. DEBUG) GO TO 70
      WRITE (IOUT1, 850)
      WRITE (IOUT1, 860)
      DO 63 L = 1, NARCS
               WRITE (IOUT1, 870) L, II(L), JJ(L), LO(L), HI(L), COST(L)
 63   CONTINUE
 65   LAST = .TRUE.
 70   RETURN
 850  FORMAT (5X, '--DEBUG FOR NVAR--', ///)
 860  FORMAT (10X, 'ARCS', 4X, 'II', 5X, 'JJ', 5X, 'LO BOND',
     &        5X, 'HI BOND', 10X, 'COST', 5X, 'FLOW', //)
 870  FORMAT (11X, I2, 5X, I2, 5X, I2, 4X, I10, 3X, I10, 5X, I6, 8X, I2)
      END
*=====================================================================
* Name:        fx_fil
* Purpose:     Open a fixed flow data file for some arcs.
*=====================================================================
      SUBROUTINE FX_FIL(NDNAM, LDND, II, JJ, ARTYP, LDARC,
     &                  PTDWAR, LDPTDW, DWAR, LDDWAR,
     &                  NFXAR, FXAR, LDFXAR,
     &                  FXUNIT, UNITNM, LDUNIT, FXFLAG,
     &                  FILNAM, NTLS, IU, OU,
     &                  CTERM, COLSTR, LDCOL)
      IMPLICIT    NONE
      INTEGER     LDND, LDARC, II(LDARC), JJ(LDARC), ARTYP(LDARC)
      INTEGER     LDPTDW, LDDWAR, PTDWAR(LDPTDW, 2), DWAR(LDDWAR)
      INTEGER     NFXAR, LDFXAR, FXAR(0:2, LDFXAR), FXUNIT, IU, OU
      INTEGER     LDUNIT, NTLS
      CHARACTER   NDNAM(LDND)*(*), UNITNM(0:LDUNIT)*(*), FILNAM*(*)
      LOGICAL     FXFLAG
*
      INTEGER     LDCOL
      CHARACTER   CTERM*(*), COLSTR(LDCOL)*(*)
*
      INTEGER     I, J, K, L, ND, ND1, ND2, I1, I2, ISGN
      INTEGER     ZONE, ARC, ARC1, TYP, NREC, SL
      LOGICAL     ERR, STRM
*
```

```
      INTEGER    SAVOPT
      COMMON     /SAVOPT/ SAVOPT
*
      FXFLAG = .FALSE.
      IF (FILNAM .EQ. ' ') GOTO 99
      CALL IO_OPFIL(IU, 1, FILNAM, 'ENTER FIXED FLOW FILE: ')
*
*------------------Skip title lines
*
      NFXAR = 0
      DO I = 1, NTLS
                                   READ (IU, *, END = 99)
      ENDDO
*
*-------------Flow units
*
      READ (IU, *, END = 99) FXUNIT
*
*-------------Read upstream and downstream nodal names
*
      DO I = 1, 2
                              READ (IU, '(A)', END = 99) CTERM
                       CALL STR_DIVD(CTERM, L, COLSTR, LDCOL, 0, ' ,')
                                    DO J = I, L
                        CALL NAMNUM(LDND, NDNAM, COLSTR(J), ND, 0, ERR)
                                    IF (ERR) THEN
                                 WRITE(*, 901) COLSTR(J), FILNAM
 901            FORMAT( '***ERROR***NODAL NAME: ', A12,
     &                 ' IN THE FILE: ',A,
     &                 /, ' NOT FOUND IN THE NETWORK CONFIGURATION.')
                                     STOP !CALL EXIT
                                         ELSE
                                   FXAR(I, J-I+1) = ND
                                        ENDIF
                                       ENDDO
                                   NFXAR = L - 1
      ENDDO
*
*------------Find corresponding arc number.
*
      DO 50 J = 1, NFXAR
                                  ND1 = FXAR(1, J)
                                  ND2 = FXAR(2, J)
                                 I1 = PTDWAR(ND1, 1)
                                 I2 = PTDWAR(ND1, 2)
                                  DO 20 I = I1, I2
                                    ARC = DWAR(I)
*
*--------------------Arc type
*
                                 TYP = ARTYP(ABS(ARC))
                                WRITE(CTERM, '(I4)') TYP
                              IF (CTERM(2:2) .NE. '1') THEN
                           PRINT *, '** ERROR ** INVALID ARC TYPE'
                                        GOTO 99
                                        ENDIF
                              IF (CTERM(3:3) .EQ. '1') THEN
                                     STRM = .TRUE.
                                         ELSE
                                     STRM = .FALSE.
                                        ENDIF
                             READ (CTERM(4:4), '(I1)') ZONE
                                ZONE = ISGN(TYP) * ZONE
*
*-------------------Downstream node
*
                                     ARC1 = ARC
                                   IF (STRM) THEN
                                ARC = ARC + ISGN(ARC)
                                        ENDIF
                                IF (ARC .GT. 0) THEN
                                     ND = JJ(ARC)
                                         ELSE
                                     ND = II(-ARC)
                                        ENDIF
*
*-------------------check the current downstream node
*
                                IF (ND .EQ. ND2) THEN
                                   FXAR(0, J) = ARC1
                                        GOTO 50
                                        ENDIF
 20      CONTINUE
 50   CONTINUE
      FXFLAG = .TRUE.
*
*------------output file information
*
      IF (SAVOPT .EQ. 0 .OR. SAVOPT .EQ. 1) THEN
                               CALL NORECS(IU, NREC)
                    WRITE (OU, 800) FILNAM, UNITNM(FXUNIT), NFXAR, NREC
 800     FORMAT (
     &    //, 'SUMMARY FOR FIXED-FLOW DATA FILE: ',
     &     /, '==================================',
```

```
     &     /, 12X, '          FILE NAME: ', A
     &     /, 12X, '              UNITS: ', A
     &     /, 12X, '    NUMBER OF NODES: ', I4,
     &     /, 12X, ' NUMBER OF RECORDS: ', I4)
                                       DO J = 1, NFXAR, 5
                                             K = J + 4
                                    IF ((J+4) .GT. NFXAR) THEN
                                              K = NFXAR
                                               ELSE
                                              K = J + 4
                                              ENDIF
                 WRITE (OU, 805) (NDNAM(FXAR(1, I))(1:SL(NDNAM(FXAR(1, I)))),
     &                       I = J, K)
                 WRITE (OU, 806) (NDNAM(FXAR(2, I))(1:SL(NDNAM(FXAR(2, I)))),
     &                       I = J, K)
                            WRITE (OU, 808) (FXAR(0, I), I = J, K)
 805        FORMAT ('            UPSTREAM NODAL NAME: ', 5A10)
 806        FORMAT ('          DOWNSTREAM NODAL NAME: ', 5A10)
 808        FORMAT ('                     ARC NUMBER: ', 5I10)
                                             ENDDO
      ENDIF
 99   CONTINUE
      RETURN
      END
*=====================================================================
* Name:        fixflw
* Purpose:     Read fixed flows and assign these flows into arc flow
*              bounds.
*=====================================================================
      SUBROUTINE FX_ARC(HI, LO, COST, LDARC, NFXAR, FXAR, LDFXAR,
     &                  FXUNIT, XF, PERD, IU,
     &                  RPOOL, LDPL, CTERM, COLSTR, LDCOL)
      IMPLICIT   NONE
      INTEGER    LDARC, HI(LDARC), LO(LDARC), COST(LDARC)
      INTEGER    NFXAR, LDFXAR, FXAR(0:2, LDFXAR), FXUNIT, IU
      REAL       XF, PERD
*
      INTEGER    LDCOL, LDPL
      REAL       RPOOL(LDPL)
      CHARACTER  CTERM*(*), COLSTR(LDCOL)*(*)
*
      REAL       UC
      INTEGER    I, J, ARC
*
      DO I = 1, NFXAR
                                       RPOOL(I) = 0.0
      ENDDO
      READ (IU, '(A)', END = 99) CTERM
      CALL  STR_DIVD(CTERM, I, COLSTR, LDCOL, 0, ' ,')
      IF (I .NE. (NFXAR + 1) ) THEN
                                      PRINT *, CHAR(7)
               PRINT *, '***ERROR*** FIXED FLOW FILE FOR TIME = ', COLSTR(1)
                                       STOP !CALL EXIT
      ENDIF
      IF (FXUNIT .EQ. 0) THEN
                                          UC = 1.0
      ELSE IF (FXUNIT .EQ. 1) THEN
                               UC = PERD * 86400.0 / 43560.0
      ELSE IF (FXUNIT .EQ. 2) THEN
                                    UC = PERD / 43560.0
      ENDIF
*
      DO I = 1, NFXAR
                         READ (COLSTR(I+1), '(F15.0)') RPOOL(I)
      ENDDO
*
      DO J = 1, NFXAR
                                      ARC = FXAR(0, J)
                                        COST(ARC) = 0
                           HI(ARC) = NINT(RPOOL(J) * UC * XF)
                           LO(ARC) = NINT(RPOOL(J) * UC * XF)
      ENDDO
 99   RETURN
      END
*=====================================================================
* Name:        sabl
* Purpose:     Open files for single arc budget list.
*=====================================================================
      SUBROUTINE SABL(NDNAM, LDND, PTDWAR, II, JJ, NDDWAR, LDARC,
     &                NSTRM, STRMAR, LDSTRM,
     &                NSABL, SABLND, LDSABL, SABLFG,
     &                FILNAM)
      IMPLICIT   NONE
      INTEGER    LDND, LDARC, LDSTRM, LDSABL, NSABL
      INTEGER    PTDWAR(LDND, 2), II(LDARC), JJ(LDARC),
     &           NDDWAR(LDARC), NSTRM, STRMAR(LDSTRM, 0:6),
     &           SABLND(LDSABL, 3)
      CHARACTER  NDNAM(LDND)*(*), FILNAM*(*)
      LOGICAL    SABLFG                  ! SABL -- SINGLE ARC BUDGET LIST.
*
      CHARACTER  CTERM*50, UPDW(2)*12, OUTFIL*12
      INTEGER    I, J, NL(2), L1, L2, ND, ARC, OU, IU
      LOGICAL    ERR
*
```

```
      SABLFG = .FALSE.
      IF (FILNAM .EQ. ' ') GOTO 99
      SABLFG = .TRUE.
      OUTFIL = 'arbud000.out'
      IU = 9
      OU = 100
      NSABL = 0
      CALL IO_OPFIL(IU, 1, FILNAM, 'ENTER SABL FILE: ')
 5    CONTINUE
      READ (IU, '(A)', END = 99) CTERM
      NSABL = NSABL + 1
      CALL STR_DIVD(CTERM, I, UPDW, 2, 0, ' ,')
      DO I = 1, 2
                       CALL NAMNUM(LDND, NDNAM, UPDW(I), ND, 0, ERR)
                                      IF (ERR) THEN
                                          CALL BEEP
                         PRINT *, '***ERROR*** NODAL NAME: ', UPDW(I),
     &                 ' IN FILE: ', FILNAM
                                  PRINT *, ' IS NOT DEFINED.'
                                        STOP !CALL EXIT
                                            ENDIF
                                   SABLND(NSABL, I) = ND
      ENDDO
*
*------------check whether the arc exists.
*
      L1 = PTDWAR(SABLND(NSABL, 1), 1)
      L2 = PTDWAR(SABLND(NSABL, 1), 2)
      DO I = L1, L2
                                         ARC = NDDWAR(I)
                                    IF (ARC .GT. 0) THEN
                   CALL FDWND(ARC, II, JJ, LDARC, NSTRM, STRMAR, LDSTRM, J)
                                     IF (J .EQ. ND) GOTO 10
                                            ENDIF
      ENDDO
      CALL BEEP
      PRINT *, '***ERROR*** ARC FROM ', UPDW(1), ' TO ', UPDW(2)
      PRINT *, '    IS NOT FOUND IN THE NETWORK CONFIGURATION.'
      PRINT *, 'CHECK FILE: ', FILNAM
      STOP !CALL EXIT
 10   CONTINUE
*
      OU = OU - 1
      SABLND(NSABL, 3) = OU
      WRITE(OUTFIL(6:8), '(I3.3)') NSABL
      CALL IO_OPFIL(OU, 3, OUTFIL, ' ')
*
      CALL STR_LEN(UPDW(1), NL(1))
      CALL STR_LEN(UPDW(2), NL(2))
      WRITE (OU, 900)  (NDNAM(SABLND(NSABL,I))(1:NL(I)), I = 1, 2),
     &                 ('=', I = 1, 31+NL(1)+NL(2))
      WRITE (OU, 901)
      GOTO 5
 99   CONTINUE
 900  FORMAT(/,T6, 'Canal water budgets from ', A, ' to ', A,
     &       /,T6, 100A)
 901  FORMAT(
     & /,T6,'                                    Water-          ',
     & /,T6,'                                   surface          ',
     & /,T6,'                 Initial    Canal    evapo-     Final',
     &      '                 Outflow',
     & /,T6,'    Inflow     storage  seepage    ration    storage',
     &      '    Outflow     (cubic',
     & /,T6,'    (acre-      (acre-     (acre-     (acre-     (acre-',
     &      '     (acre-   feet per',
     & /,' No.',
     &   T6,'     feet)      feet)      feet)      feet)      feet)',
     &      '      feet)   second)',
     & /,' ---',
     &   T6,' ---------  --------- -------- --------- ---------',
     &      ' ---------  --------')
      RETURN
      END
*======================================================================
* Name:       snbl
* Purpose:    Open files for selected nodal budget lists.
*======================================================================
      SUBROUTINE SNBL(NNODS, NDNAM, LDND, II, JJ, LDARC,
     &                PTDWAR, LDPTDW, NDDWAR, LDDWAR,
     &                NSTRM, STRMAR, LDSTRM,
     &                NSNBL, SNBLND, LDSNBL, SNBLFG, FILNAM)
      IMPLICIT    NONE
      INTEGER     NNODS, LDND, LDPTDW, LDDWAR, LDSTRM, LDSNBL
      CHARACTER   NDNAM(LDND)*(*), FILNAM*(*)
      INTEGER     LDARC, II(LDARC), JJ(LDARC), PTDWAR(LDPTDW, 2),
     &            NDDWAR(LDDWAR), NSTRM, STRMAR(LDSTRM, 0:6)
      INTEGER     NSNBL, SNBLND(LDSNBL, 3)
      LOGICAL     SNBLFG                  ! SNBL -- SINGLE NODAL BUDGET LIST.
*
      CHARACTER   NAME*12, OUTFIL*30
      INTEGER     DWNDS(15), NDWNDS
      INTEGER     I, J, L, ND, OU, IU, SL
      LOGICAL     ERR
*
```

```
      SNBLFG = .FALSE.
      IF (FILNAM .EQ. ' ') GOTO 99
      SNBLFG = .TRUE.
      IU = 9
      OU = 40
      NSNBL = 0
      CALL IO_OPFIL(IU, 1, FILNAM, 'ENTER SNBL FILE: ')
 5    CONTINUE
      READ (IU, '(A)', END = 99) NAME
      CALL NAMNUM(NNODS, NDNAM, NAME, ND, 0, ERR)
      IF (.NOT. ERR) THEN
                                          NSNBL = NSNBL + 1
*
*---------------Open budget output file
*
                                              OU = OU + 1
                                         OUTFIL = 'b'//NAME
                                     CALL STR_LEN(OUTFIL, L)
                                OUTFIL = OUTFIL(1:L) // '.out'
                                    CALL STR_CORS(OUTFIL, 0)
                                      SNBLND(NSNBL, 1) = ND
                                      SNBLND(NSNBL, 2) = OU
                              CALL IO_OPFIL(OU, 3, OUTFIL, ' ')
                                   WRITE (OU, 900) NDNAM(ND)
*
*---------------Open outflow file for a given node.
*
                                              OU = OU + 1
                                      SNBLND(NSNBL, 3) = OU
                                        OUTFIL = 'r'//NAME
                                    CALL STR_CORS(OUTFIL, 0)
                                     CALL STR_LEN(OUTFIL, L)
                                OUTFIL = OUTFIL(1:L) // '.out'
                              CALL IO_OPFIL(OU, 3, OUTFIL, ' ')
                                   CALL STR_LEN(NDNAM(ND), L)
                        WRITE(OU, 905) NDNAM(ND), ('=', I = 1, 14+L)
              CALL FDWNDS(ND, II, JJ, LDARC, PTDWAR, LDND, NDDWAR, LDDWAR,
     &                 NSTRM, STRMAR, LDSTRM,  NDWNDS, DWNDS)
                  WRITE(OU, 906) (NDNAM(DWNDS(I))(1:SL(NDNAM(DWNDS(I)))),
     &                      I = 1, NDWNDS)
 906       FORMAT(/, 5X, 20(4X, A12, 4X))
                WRITE(OU, '(5x, 10(1x, 19a1))') (('-', J = 1, 19), I=1,NDWNDS)
                  WRITE(OU, '(5x, 10(10x, a10))') ('(cubic', I = 1, NDWNDS)
                               WRITE(OU, '(5x, 10(a10, a10))')
     &                  ('(acre-', 'feet per', I = 1, NDWNDS)
                               WRITE(OU, '(a5, 20a10)') 'No.',
     &                            ('feet)', 'second)', I = 1, NDWNDS)
                               WRITE(OU, '(a5, 20a10)') '---',
     &                       ('---------', '--------', I = 1, NDWNDS)
      ENDIF
      GOTO 5
 99   CONTINUE
 900  FORMAT(/,T30,  'Water Budgets for ', A
     &       /,T30, '=========================',
     & /,       T11, '   Initial  Upstream Local net',
     & '    Evapo-                                Downstream     Final',
     & '     Final',
     & /,       T11, '   storage    inflow    inflow',
     & '    ration    Runoff   Seepage Withdrawal  release   storage',
     & /,          T11, '     (acre-     (acre-     (acre-',
     & '    (acre-     (acre-     (acre-     (acre-     (acre-     (acre-',
     & '     stage     depth',
     & /,' No.', T11, '      feet)      feet)      feet)',
     & '     feet)     feet)      feet)      feet)      feet)      feet)',
     & '    (feet)    (feet)',
     & /,'----', T11,  ' --------- --------- ---------',
     & ' --------- --------- --------- --------- --------- ---------',
     & ' --------- ---------')
 905  FORMAT(/, T5, 'Outflows from ', A, /, T5, 100A)
      RETURN
      END
*=====================================================================
* Name:       fdwnds
* Purpose:    Find all downstream nodes.
*=====================================================================
      SUBROUTINE FDWNDS(ND, II, JJ, LDARC, PTDWAR, LDND, NDDWAR, LDDWAR,
     &                  NSTRM, STRMAR, LDSTRM, NDWNDS, DWNDS)
      IMPLICIT   NONE
      INTEGER    ND,LDARC, LDND, LDDWAR, LDSTRM,
     &           II(LDARC), JJ(LDARC),
     &           PTDWAR(LDND, 2), NDDWAR(LDDWAR),
     &           NSTRM, STRMAR(LDSTRM,0:6)
      INTEGER    NDWNDS, DWNDS(15)
*
      INTEGER    I, J, K, I1, I2, ARC
      LOGICAL    CNINT
*
      I1 = PTDWAR(ND, 1)
      I2 = PTDWAR(ND, 2)
      NDWNDS = 0
      DO I = 1, 15
                                         DWNDS(I) = 0
      ENDDO
      DO I = I1, I2
```

```
                                   ARC = NDDWAR(I)
                                IF (ARC .GT. 0) THEN
                  CALL FDWND(ARC, II, JJ, LDARC, NSTRM, STRMAR, LDSTRM, J)
                          IF (.NOT. CNINT(J, 15, DWNDS, K)) THEN
                                    NDWNDS = NDWNDS + 1
                                     DWNDS(NDWNDS) = J
                                          ENDIF
                                        ENDIF
      ENDDO
      RETURN
      END
*=====================================================================
* Name:        svinfo
* Purpose:     Print the file save information on screen.
*=====================================================================
      SUBROUTINE SVINFO(FILNAM, LDFIL, NHY, NSNBL, NSABL,
     &                  NDBFLG, ARBFLG, HYBFLG)
      IMPLICIT    NONE
      INTEGER     LDFIL, NHY, NSNBL, NSABL
      CHARACTER*(*) FILNAM(0:LDFIL)
      LOGICAL     NDBFLG, ARBFLG, HYBFLG
      PRINT '(A)', CHAR(7)
      WRITE(*, 901) FILNAM(26)
      IF (NDBFLG) THEN
                          WRITE (*, '(5X, A20, A)') FILNAM(27),
     &                   ':Nodal budget summary for each time step.'
      ENDIF
      IF (ARBFLG) THEN
                          WRITE (*, '(5X, A20, A)') FILNAM(28),
     &                        ':Channel routing results.'
      ENDIF
      IF (NHY .GT. 0 .AND. HYBFLG) THEN
                                 WRITE(*, 902) FILNAM(29)
      ENDIF
      IF (NSNBL .GE. 1) THEN
                                WRITE(*, 904) NSNBL, NSNBL
      ENDIF
      IF (NSABL .EQ. 1) THEN
                                    WRITE(*, 905)
      ELSE IF (NSABL .GT. 1) THEN
                                  WRITE(*, 906) NSABL
      ENDIF
      RETURN
 901  FORMAT(' Summary of output files: '
     & /,    ' ========================'
     & /, 5X, 'File name', T26, 'Description',
     & /, 5X, '---------', T26, '-----------',
     1 /, 5X, A20, ':General data file summary and network ',
     1            'configuration.')
 902  FORMAT(
     1     5X, A20, ':Flows through outlet structure')
 904  FORMAT(' There are ', I3, ' nodal budget files in time series ',
     &   /,  '  format with name convention bNDNAME.out.',
     &   /,  ' There are ', I3, ' outflow files in time series',
     &   /,  '  format with the name convention rNDNAME.out.')
 905  FORMAT(
     &     5X, 'arbud001.out', T26, ':Channel routing in time series ',
     1            'format for a selected arc')
 906  FORMAT(
     &     5X, 'arbud001.out', T26, ':Channel routing in time series ',
     1            'format for selected arcs',
     1 /, 5X, ' through',
     2 /, 5X, 'arbud', I3.3, '.out')
      END
*=====================================================================
      SUBROUTINE BEEP
      PRINT '(A)', CHAR(7)
      RETURN
      END
*=====================================================================
* Name:        io_opfil
* Purpose:     Open a file
*=====================================================================
      SUBROUTINE IO_OPFIL(UNIT, INOUT, FILNAM, STRING)
      IMPLICIT NONE
      INTEGER UNIT, INOUT, IOS, I, J, LENGTH
      CHARACTER *(*) FILNAM, STRING
      CHARACTER   STATS*7, ACCES*10, FIL_NAM2*100
      LOGICAL YES
      IF (LEN(FILNAM) .LE. 100) THEN
                                   FIL_NAM2 = FILNAM
                                     LENGTH = 100
      END IF
*
*          open data file
*
      STATS = 'UNKNOWN'
      ACCES = 'SEQUENTIAL'
      IF (INOUT .EQ. 1) THEN
                                    STATS = 'OLD'
      ELSE IF (INOUT .EQ. 2) THEN
                                    STATS = 'NEW'
      ELSE IF (INOUT .EQ. 4) THEN
                                  STATS = 'SCRATCH'
```

```
      ELSE IF (INOUT .EQ. 6) THEN
                                              ACCES = 'APPEND'
      END IF
      IF (FIL_NAM2(1:1) .NE. ' ') GO TO 10
 2    IF (STRING .NE. ' ') THEN
                                              PRINT *, STRING
      ELSE
                                          IF (INOUT .EQ. 1) THEN
                                  PRINT *, 'ENTER INPUT DATA FILE NAME: '
                                       ELSE IF (INOUT .EQ. 2) THEN
                                    PRINT *, 'ENTER OUTPUT FILE NAME: '
                                                   ELSE
                                    PRINT *, 'ENTER SCRATCH FILE NAME: '
                                                  END IF
      END IF
      READ '(A)', FIL_NAM2
      IF (FIL_NAM2(1:1) .EQ. CHAR(27)) THEN
                                                   STOP
      ENDIF
      DO 3 I = 1, LENGTH
                                  IF (FIL_NAM2(I:I) .NE. ' ') GO TO 4
 3    CONTINUE
      PRINT *, '***** ILLEGAL FILE NAME ***** ZERO LENGTH FILE NAME'
      GO TO 2
 4    CONTINUE
      IF ((I-1) .EQ. 0) THEN
                                                 GO TO 10
      ELSE
                                            DO 5 J = 1, LENGTH
                                       IF (J .LE. (LENGTH-I+1)) THEN
                                    FIL_NAM2(J:J) = FIL_NAM2(J+I-1:J+I-1)
                                                    ELSE
                                              FIL_NAM2(J:J) = ' '
                                                   END IF
 5       CONTINUE
      END IF
10    OPEN (UNIT, FILE = FIL_NAM2, STATUS = STATS, IOSTAT = IOS,
     &      ACCESS = ACCES)
      IF (IOS .NE. 0) THEN
                                              PRINT *, CHAR(7)
                                          IF (INOUT .EQ. 1) THEN
                   PRINT *, '******ERROR****** FILE DOES NOT EXIST. FILE: ',
     &                  FIL_NAM2
                  PRINT *, ' PLEASE TYPE ANY KEY TO TRY AGAIN OR TYPE CTR-C',
     &            ' QUIT'
                                                  READ '(A)'
                                                   GO TO 2
                                       ELSE IF (INOUT .EQ. 2) THEN
                  PRINT *, '*****WARNING***** THERE ALREADY EXISTS FILE:' ,
     &                  FIL_NAM2
                         PRINT *, ' DO YOU WANT TO OVERWRITE IT? (Y) '
                                              IF (YES()) THEN
                                              STATS = 'UNKNOWN'
                                                   GO TO 10
                                                   ELSE
                                                   GO TO 2
                                                  END IF
                                                END IF
      END IF
      IF (LEN(FILNAM) .GT. 6) THEN
                                             FILNAM = FIL_NAM2
      ENDIF
      RETURN
      END
*======================================================================
* Name:        Yes
* Purpose:     Response yes or no from keyboard.
*======================================================================
      LOGICAL FUNCTION YES()
      IMPLICIT NONE
      CHARACTER YESNO*1
      YES = .FALSE.
*
      YESNO = ' '
 5    READ '(A)', YESNO
      IF (YESNO .EQ. 'Y' .OR. YESNO .EQ. 'y' .OR. YESNO .EQ. ' ') THEN
                                                YES = .TRUE.
                                                  GO TO 10
      ELSE IF (YESNO .EQ. 'N' .OR. YESNO .EQ. 'n') THEN
                                                YES = .FALSE.
                                                  GO TO 10
      ELSE
                                             PRINT *, CHAR(7)
                                                  GO TO 5
      END IF
 10   RETURN
      END
*======================================================================
* Name:        No
* Purpose:     Response no or yes from keyboard.
*======================================================================
      LOGICAL FUNCTION NO()
      IMPLICIT NONE
      CHARACTER YESNO*1
```

```
      NO = .FALSE.
*
      YESNO = ' '
 5    READ '(A)', YESNO
      IF (YESNO .EQ. 'N' .OR. YESNO .EQ. 'n' .OR. YESNO .EQ. ' ') THEN
                                  NO = .TRUE.
                                  GO TO 10
      ELSE IF (YESNO .EQ. 'Y'..OR. YESNO .EQ. 'y') THEN
                                  NO = .FALSE.
                                  GO TO 10
      ELSE
                               PRINT *, CHAR(7)
                                  GO TO 5
      END IF
 10   RETURN
      END
*==================================================================
* Name:       io_rdint
* Purpose:    assign a default value or read an integer value
*==================================================================
      SUBROUTINE IO_RDINT(STRING, DEFVAL, VAR)
      IMPLICIT   NONE
      INTEGER    DEFVAL, VAR, W, L
      CHARACTER  STRING*(*), TERM*10, FMT * 30
      REAL       TINY
      DATA       TINY/1E-3/
*
      FMT = '(1X,A,A,I    , A)'
      GOTO 10
 5    CONTINUE
      PRINT *, '***ERROR*** ONLY INTERGER NUMBER IS VALID. TRY AGAIN.'
 10   IF (ABS(DEFVAL) .GT. 0) THEN
                       W = INT( LOG10(ABS(DEFVAL)+TINY) )+ 1
      ELSE
                                    W = 1
      ENDIF
      IF (W .GE. 10) THEN
                         WRITE(FMT(10:11), '(I2)') W
      ELSE
                         WRITE(FMT(10:10), '(I1)') W
      ENDIF
*
      CALL STR_LEN(STRING, L)
      WRITE (*, FMT) STRING(1:L), '(', DEFVAL, ')'
      READ '(A)', TERM
      IF (TERM .EQ. ' ') THEN
                                  VAR = DEFVAL
      ELSE
                        READ (TERM, '(I10)', ERR = 5) VAR
      ENDIF
      RETURN
      END
*==================================================================
* Name:       io_rdnum
* purpose:    Assign a default value or read an real value
*==================================================================
      SUBROUTINE IO_RDNUM(STRING, DEFVAL, VAR, D)
      IMPLICIT    NONE
      REAL        DEFVAL, VAR
      INTEGER     W, D
      CHARACTER   STRING*(*), TERM*20, FMT * 30
      REAL        TINY
      DATA        TINY/1E-10/
*
      FMT = '(1X,A,A,F       , A)'
      IF (ABS(DEFVAL) .GT. TINY) THEN
                          W = INT( LOG10(ABS(DEFVAL)) )
                               IF (W .GT. 0) THEN
                                  W = W + D + 3
                                   ELSE
                                   W = 3 + D
                                   ENDIF
                               IF (W .LT. 10) THEN
                    WRITE(FMT(10:12), '(I1, A1, I1)') W, '.', D
                                   ELSE
                    WRITE(FMT(10:13), '(I2, A1, I1)') W, '.', D
                                   ENDIF
      ELSE
                 WRITE(FMT(10:12), '(I1, A1, I1)') D+3, '.', D
      ENDIF
 5    CONTINUE
      WRITE (*, FMT) STRING, '(', DEFVAL, ')'
      READ '(A)', TERM
      IF (TERM .EQ. ' ') THEN
                                  VAR = DEFVAL
      ELSE
                       READ (TERM, '(F15.0)', ERR = 5) VAR
      ENDIF
      RETURN
      END
*==================================================================
* Name:       str_len
* Purpose:    Find the ending position of a character string
*==================================================================
```

```
      SUBROUTINE STR_LEN(STRING, STRLEN)
      IMPLICIT   NONE
      CHARACTER  STRING*(*)
      INTEGER    STRLEN
*
      IF (STRING .EQ. ' ') THEN
                                          STRLEN = 0
      ELSE
                                     STRLEN = LEN(STRING)
                        DO WHILE (STRING(STRLEN:STRLEN) .EQ. ' ')
                                       STRLEN = STRLEN - 1
                                            ENDDO
      ENDIF
      RETURN
      END
*======================================================================
* Name:        str_divd
* purpose:     Divide a string into substrings with multiple
*              delimitors.
*======================================================================
      SUBROUTINE STR_DIVD(STRING, NOCOL, COLSTR, LDCOL, IALIGN, DELIM)
      IMPLICIT   NONE
      INTEGER    LDCOL, NOCOL, IALIGN
      CHARACTER  STRING*(*), COLSTR(LDCOL)*(*), DELIM*(*)
*
      CHARACTER  DLIM(15)*1
      INTEGER    STRING_LEN, COLSTR_LEN, I, J, K, L, NODLIM
      LOGICAL    ISDLIM
*
*          Find the no. of delimitors
*
      J = LEN(DELIM)
      DO I = J, 1, -1
                             IF (DELIM(I:I) .NE. ' ') GOTO 5
      ENDDO
      I = 1
    5 NODLIM = I
      DO I = 1, NODLIM
                                  DLIM(I) = DELIM(I:I)
      ENDDO
*
*         Find the main string defined length and output substring length
*
      DO I = 1, LDCOL
                                    COLSTR(I) = ' '
      END DO
      STRING_LEN = LEN(STRING)
      COLSTR_LEN = LEN(COLSTR(1))
      NOCOL = 0
*
*         Find the end position of the input main string
*
      I = STRING_LEN
      DO WHILE(STRING(I:I) .EQ. ' ')
                                          I = I - 1
      END DO
      IF (I .EQ. 1) THEN
                              IF (STRING(1:1) .NE. ' ')THEN
                                         NOCOL = 1
                                  COLSTR(1) = STRING(1:1)
                                          ENDIF
                                         GO TO 99
      ELSE
                                        STRING_LEN = I
      END IF
      J = 0
*
*          Find the beginning position of a substring
*
   10 CONTINUE
      I = J+1
 11   CONTINUE
      IF (I .GT. STRING_LEN) GO TO 99
      DO L = 1, NODLIM
                           IF (STRING(I:I) .EQ. DLIM(L)) THEN
                                         I = I + 1
                                          GOTO 11
                                          ENDIF
      ENDDO
*      do while(string(i:i) .eq. ' ')
*         i = i + 1
*      enddo
*
*             Find end position of a substring
*
      J = I + 1
      NOCOL = NOCOL + 1
      DO WHILE (J .LE. STRING_LEN)
                       IF (ISDLIM(STRING(J:J), NODLIM, DLIM)) THEN
                                          GOTO 20
                                          ENDIF
                                        J = J + 1
      ENDDO
   20 J = J - 1
```

```
*
*            assign the substring to an associated array
*
      COLSTR(NOCOL) = ' '
      IF (IALIGN .EQ. 1) THEN
                                            K = J - I + 1
                                       K = (COLSTR_LEN - K) / 2
                                          IF (K .EQ. 0) K = 1
                                 COLSTR(NOCOL)(K:(J-I)+K) = STRING(I:J)
      ELSE IF (IALIGN .EQ. 2) THEN
                                       K = COLSTR_LEN - (J - I)
                               COLSTR(NOCOL)(K:COLSTR_LEN) = STRING(I:J)
      ELSE
                                      COLSTR(NOCOL) = STRING(I:J)
      ENDIF
      J = J + 1
      GO TO 10
   99 CONTINUE
      RETURN
      END
      LOGICAL FUNCTION ISDLIM(CH, NDLIM, DLIM)
      IMPLICIT NONE
      INTEGER  NDLIM
      CHARACTER*1 CH, DLIM(NDLIM)
*
      INTEGER  I
*
      ISDLIM = .FALSE.
      DO I = 1, NDLIM
                                       IF (CH .EQ. DLIM(I)) THEN
                                             ISDLIM = .TRUE.
                                                 RETURN
                                                ENDIF
      ENDDO
      RETURN
      END
*=====================================================================
* Name:        io_rdstr
* Purpose:     Assign a default string or read an new string
*=====================================================================
      SUBROUTINE IO_RDSTR(STRING, VAR)
      IMPLICIT    NONE
      CHARACTER   STRING*(*), VAR*(*), TERM*150
      INTEGER     SL, VL, TL
*
      CALL STR_LEN(STRING, SL)
      IF (VAR .EQ. ' ') THEN
                                         PRINT *, STRING(1:SL)
      ELSE
                                         CALL STR_LEN(VAR, VL)
                              PRINT *,  STRING(1:SL), ' (', VAR(1:VL), ')'
      ENDIF
      READ '(A)', TERM
      IF (TERM .NE. ' ') THEN
                                        CALL STR_LEN(TERM, TL)
                                           VAR = TERM(1:TL)
      ENDIF
      RETURN
      END
*=====================================================================
* Name:        cn
* Purpose:     Check whether a substring is contained in a string
*=====================================================================
      LOGICAL FUNCTION CN(STRING, SUBSTR, CODE)
      IMPLICIT    NONE
      CHARACTER   STRING*(*), SUBSTR*(*)
      INTEGER     CODE
      INTEGER     J1, J2, K1, K2
*
      CHARACTER   RTERM * 150, KEY * 150
*
      CN = .FALSE.
      RTERM = STRING
      KEY = SUBSTR
      IF (CODE .NE. 0) THEN
                                       CALL STR_CORS(RTERM, 0)
                                        CALL STR_CORS(KEY, 0)
      ENDIF
*
      CALL STR_POS(RTERM, J1, J2)
      CALL STR_POS(KEY, K1, K2)
      IF (KEY(K1:K2) .EQ. RTERM(J1:J2)) THEN
                                              CN = .TRUE.
      ENDIF
      RETURN
      END
*=====================================================================
* Nname:       str_pos
* Purpose:     Find the beginning and end postion of a string
*=====================================================================
      SUBROUTINE STR_POS(STRING, IPT1, IPT2)
      IMPLICIT NONE
      CHARACTER * (*) STRING
      INTEGER IPT1, IPT2
```

```
      INTEGER I
      IPT2 = LEN(STRING)
      DO I = IPT2, 1, -1
                                IF (STRING(I:I) .NE. ' ') GO TO 5
      END DO
  5   IPT2 = I
      DO I = 1, IPT2
                               IF (STRING(I:I) .NE. ' ') GO TO 10
      END DO
 10   IPT1 = I
      RETURN
      END
*====================================================================
* Name:        str_cors
* Purpose:     convert a string into capital latter or small latters.
*====================================================================
      SUBROUTINE STR_CORS(CLINE, ICTR)
      IMPLICIT   NONE
      CHARACTER  CLINE*(*)
      INTEGER    ICTR, CODE, L, I
*
      CALL STR_LEN(CLINE, L)
      DO I = 1, L
                                     CODE = ICHAR(CLINE(I:I))
                                       IF (ICTR .EQ. 0) THEN
                        IF ( (CODE .GE. 65) .AND. (CODE .LE. 90) ) THEN
                                        CLINE(I:I) = CHAR(CODE+32)
                                                   ENDIF
                                    ELSE IF (ICTR. EQ. 1) THEN
                       IF ( (CODE .GE. 97) .AND. (CODE .LE. 122) ) THEN
                                        CLINE(I:I) = CHAR(CODE-32)
                                                   ENDIF
                                                 ENDIF
      ENDDO
      RETURN
      END
*====================================================================
* Name:        str_no
* purpose:     Reture the number of substrings with multiple deliminators
*====================================================================
      SUBROUTINE STR_NO(STRING, NOSUB, DELIM)
      IMPLICIT NONE
      INTEGER  NOSUB
      CHARACTER*(*) STRING, DELIM
      CHARACTER * 1 DLIM(15)
      INTEGER STRING_LEN, I, J, L, NODLIM
*
*          Initialize
*
      NOSUB = 0
*
*          Find the no. of delimitors
*
      J = LEN(DELIM)
      DO I = J, 1, -1
                                IF (DELIM(I:I) .NE. ' ') GOTO 5
      ENDDO
      I = 1
   5  NODLIM = I
      DO I = 1, NODLIM
                                      DLIM(I) = DELIM(I:I)
      ENDDO
*
*         Find the defined length for main string
*
      STRING_LEN = LEN(STRING)
*
*         Find the true length of the input main string
*
      I = STRING_LEN
      DO WHILE(STRING(I:I) .EQ. ' ')
                                            I = I - 1
      END DO
      IF (I .EQ. 1) THEN
                                 IF (STRING(1:1) .NE. ' ')THEN
                                             NOSUB = 1
                                              ELSE
                                             NOSUB = 0
                                              ENDIF
                                            GO TO 99
      ELSE
                                         STRING_LEN = I
      END IF
      J = 0
*
   10 CONTINUE
*
*          find the beginning position of a substring
*
      I = J+1
 15   CONTINUE
      IF (I .GT. STRING_LEN) GO TO 99
      IF (STRING(I:I) .EQ. ' ') THEN
                                            I = I + 1
```

```
                                            GOTO 15
      ELSE
                                        DO L = 1, NODLIM
                                 IF (STRING(I:I) .EQ. DLIM(L)) THEN
                                              I = I + 1
                                               GOTO 15
                                              ENDIF
                                             ENDDO
      ENDIF
*
*             find end position of a substring
*
      J = I + 1
      NOSUB = NOSUB + 1
      DO WHILE (J .LE. STRING_LEN)
                                        DO L = 1, NODLIM
                               IF (STRING(J:J) .EQ. DLIM(L)) GOTO 10
                                             ENDDO
                                          J = J + 1
      ENDDO
   99 RETURN
      END
*=====================================================================
      FUNCTION ISGN (NUM)
      IMPLICIT NONE
      INTEGER  ISGN, NUM
*
      IF (NUM .GT. 0) THEN
                                           ISGN = 1
      ELSE IF (NUM .LT. 0) THEN
                                           ISGN = -1
      ELSE
                                           ISGN = 0
      ENDIF
      RETURN
      END
*====================================================================*
* Name:         exist
* Purpose:      check file status.
*====================================================================*
      LOGICAL FUNCTION EXIST(FILNAM)
      IMPLICIT  NONE
      CHARACTER FILNAM*(*)
      INQUIRE (FILE = FILNAM, EXIST = EXIST)
      RETURN
      END
*====================================================================*
      FUNCTION SL(STRING)
      IMPLICIT   NONE
      CHARACTER  STRING*(*)
      INTEGER    SL
*
      IF (STRING .EQ. ' ') THEN
                                            SL = 0
      ELSE
                                        SL = LEN(STRING)
                               DO WHILE (STRING(SL:SL) .EQ. ' ')
                                           SL = SL - 1
                                             ENDDO
      ENDIF
      RETURN
      END
*=====================================================================
* Name:         hydr_dat
* Purpose:      Read hydraulic structure data.
*=====================================================================
      SUBROUTINE HYDR_DAT(NNODS, NDNAM, LDND, JJ, LDARC,
     &                    PTDWAR, LDPTDW, DWAR, LDDWAR,
     &                    NSTRM, STRMAR, LDSTRM,
     &                    NHYTP, HYTP, LDHYTP,
     &                    NHY, HYDIR, HYTPCD, HYDAT, LDHY,
     &                    FILNAM, NTLS, IN, OU, PARTNO,
     &                    CTERM, COLSTR, LDCOL)
      IMPLICIT    NONE
      INTEGER     NNODS, LDND, LDARC, JJ(LDARC)
      CHARACTER   NDNAM(LDND)*(*)
      INTEGER     LDPTDW, PTDWAR(LDPTDW, 2), LDDWAR, DWAR(LDDWAR)
      INTEGER     NSTRM, LDSTRM, STRMAR(LDSTRM, 0:6)
      INTEGER     NHY, LDHY
      CHARACTER   HYDIR(LDHY, 0:2)*(*)
      INTEGER     HYTPCD(LDHY, 0:1)
      REAL        HYDAT(LDHY, 5)
      CHARACTER*(*) FILNAM
      INTEGER     NTLS, IN, OU, PARTNO
*
      INTEGER     LDCOL
      CHARACTER   CTERM*(*), COLSTR(LDCOL)*(*)
*
      INTEGER     I, J, N, K, IU, LIM1, LIM2, ND, ND2, UPND, DWND, ARC
      LOGICAL     ERR, CN, FLAG
*
      INTEGER     NHYTP, LDHYTP
      CHARACTER   HYTP(0:LDHYTP)*(*)
      INTEGER     SAVOPT
```

```
      COMMON     /SAVOPT/ SAVOPT
*
*--------------------Assign hydraulic types
*
      NHYTP = 6
      HYTP(0) = ' '
      HYTP(1) = 'Sharp-crested weir'
      HYTP(2) = 'Gate spillway'
      HYTP(3) = 'Sluice gate'
      HYTP(6) = 'Pipe'
*
      IF (FILNAM .NE. ' ') THEN
                                  FLAG = .TRUE.
                                     IU = 9
               CALL IO_OPFIL(IU, 1, FILNAM,'ENTER STRUCTURE FILE: ')
                                  DO I = 1, NTLS
                               READ (IU, *, END = 100)
                                     ENDDO
      ELSE
                                  FLAG = .FALSE.
                                     IU = IN
                                   CTERM = ' '
                            DO WHILE (CTERM .EQ. ' ')
                          READ (IU, '(A)', END = 100) CTERM
                                     ENDDO
                  CALL STR_DIVD(CTERM, I, COLSTR, LDCOL, 0, ' ,')
                             READ (COLSTR(2), '(I2)') N
               IF (CN(COLSTR(1), 'PART', 1) .AND. N .EQ. PARTNO) THEN
                        READ (IU, *) !SKIP THE VARIABLE LIST
                                      ELSE
                IF (CN(COLSTR(1), 'PART', 1) .AND. N .GT. PARTNO )THEN
                                   BACKSPACE(1)
                                     GOTO 100
                                      ENDIF
                                  PRINT *, CHAR(7)
                    PRINT *, '***ERROR***ERROR IN STRUCTURE DATA'
                                  STOP !CALL EXIT
                                     ENDIF
      ENDIF
*
*-----------------Output title
*
      IF (SAVOPT .EQ. 0 .OR. SAVOPT .EQ. 1) THEN
                                  WRITE(OU, 800)
 800     FORMAT(//, 'Part 4: Hydraulic-structure data',
     &           /, '================================',
     & /, T57,
     & '          Weir      Weir      ',
     & /, T57,
     & '          width     height    ',
     &   /, '                                                  ', T57,
     & '    Base    or pipe  or pipe    Pipe-      Pipe-  ',
     &   /, 'Structure   F-node       T-node      Structure', T57,
     & ' elevation  diameter length    friction entry loss',
     &   /, 'name        name         name           type', T57,
     & '   (feet)   (feet)    (feet)     factor     factor',
     &   /, '----        ----         ----           ----', T57,
     & ' --------- --------- --------- --------- ---------')
      ENDIF
*
*---------------Read data
*
      NHY = 0
 5    CONTINUE
      READ (IU, '(A)', END = 100) CTERM
      CALL STR_DIVD(CTERM, K, COLSTR, LDCOL, 0, ' ,')
      IF (CN(COLSTR(1), 'FINISH', 1)) GOTO 100
      IF (K .LT. 6) GOTO 999
      NHY = NHY + 1
      DO J = 1, K
                                IF (J .LE. 3) THEN
                        READ(COLSTR(J), '(A)') HYDIR(NHY, J-1)
                              ELSE IF (J .EQ. 4) THEN
                       READ (COLSTR(J), '(I15)') HYTPCD(NHY, 0)
                                     ELSE
                     READ(COLSTR(J), '(F15.0)') HYDAT(NHY, J-4)
                                     ENDIF
      ENDDO
      IF (SAVOPT .EQ. 0 .OR. SAVOPT .EQ. 1) THEN
                    WRITE (OU, 805) (HYDIR(NHY, J), J = 0, 2),
     &              HYTP(HYTPCD(NHY, 0)),
     &             (HYDAT(NHY, J), J = 1, 5)
 805     FORMAT(3A12, A20, 3F10.2, 2F10.5)
      ENDIF
      GOTO 5
 100  CONTINUE
      IF (FLAG) CLOSE(IU)
*
*           Search the downstream arc for normal flow range.
*
      DO 30 I = 1, NHY
                    CALL NAMNUM(LDND, NDNAM, HYDIR(I,1), UPND, 0, ERR)
                    CALL NAMNUM(LDND, NDNAM, HYDIR(I,2), DWND, 0, ERR)
                                 LIM1 = PTDWAR(UPND, 1)
```

```
                              LIM2 = PTDWAR(UPND, 2)
                                DO J = LIM1, LIM2
                                 ARC = IABS(DWAR(J))
                                    ND = JJ(ARC)
              IF (ND .GT. NNODS) THEN        !THE CURRENT ARC IS A STREAM ARC.
                                    DO K = 1, NSTRM
                              IF (ND .EQ. STRMAR(K, 0)) THEN
                               ND2 = JJ( IABS(STRMAR(K, 6)))
                                 IF (ND2 .EQ. DWND) THEN
                                 HYTPCD(I, 1) = ARC
                                     GOTO 30
                                       ENDIF
                                      ENDIF
                                       ENDDO
                              ELSE IF (ND .EQ. DWND) THEN
                                  HYTPCD(I, 1) = ARC
                                       GOTO 30
                                      ENDIF
                                     ENDDO
 30   CONTINUE
 999  RETURN
      END
*=================================================================
* Name:       hydr_hite
* Purpose:    Calculate gate opening height or sharp-crested weir
*             height for a given discharge.
*==================================================================
      SUBROUTINE HYDR_HITE(
     &                   ARCBUD, LDARC,
     &
     &                   NHY, HYTPCD, HYDAT, HYOUT, LDHY,
     &                   PERD, CONST, XF)
      IMPLICIT    NONE
      INTEGER     LDARC, ARCBUD(LDARC, 0:5)
      INTEGER     NHY, LDHY, HYTPCD(LDHY, 0:1)
      REAL        HYDAT(LDHY, 2), HYOUT(LDHY, 3)
      REAL        PERD, CONST, XF
*
      REAL        FLW, W, B, P, H1, HD, E, EPS
      INTEGER     N, TYP, ARC
      INTEGER     IFAULT
      DATA        EPS/1.0E-10/
*
      LOGICAL     CHECK
      COMMON      /CHECK/ CHECK
*
      DO 100 N = 1, NHY
                                 TYP = HYTPCD(N,0)
                                 ARC = HYTPCD(N,1)
*
                                IF (TYP .LE. 0) THEN
                                     GOTO 100
*
*----------------Sharp-crested weir with fixed weir height,
*                It is no needed to calculate the weir height.
*
                   ELSE IF (TYP .EQ. 1 .AND. HYDAT(N, 3) .GT. 0) THEN
                                     GOTO 100
*
*----------------Fixed gate opening height, no need to calculate
*                the gate opeing height.
*
                      ELSE IF ((TYP .GE. 2 .AND. TYP .LE. 3) .AND.
     &           HYDAT(N,5) .GT. 0) THEN
                                     GOTO 100
                                     ENDIF
*
*----------------Find downstream flows
*
*
             FLW = ARCBUD(ARC, 1) / PERD / CONST / XF !RELEASE RATE IN CFS
                                HYOUT(N,1) = FLW
*
*----------------Calculate gate opening height or sharp-crested weir height
*
                           H1 = HYOUT(N, 2) - HYDAT(N, 1)
                                 B = HYDAT(N, 2)
                                 W = HYDAT(N, 4)
*
                            IF (ABS(FLW) .LT. EPS) THEN
                                     CONTINUE
                             ELSE IF (TYP .EQ. 1) THEN
                  CALL WEIRHITE(FLW, H1, 0.0, B, P, W, 0.0, TYP, IFAULT)
                                   HYOUT(N,3) = P
                       ELSE IF (TYP .EQ. 2 .OR. TYP .EQ. 3) THEN
                                   HD = HYDAT(N, 3)
                    CALL GATEHITE(FLW, H1, 0.0, B, E, HD, TYP, IFAULT)
                                   HYOUT(N,3) = E
                                     ENDIF
 100  CONTINUE
      RETURN
      END
*==================================================================
* Name:       hydr_prn
```

```
* Purpose:    Print the results for flow through structure.
*=====================================================================
      SUBROUTINE HYDR_PRN(HYTP, LDHYTP,
     &                    NHY, HYDIR, HYTPCD, HYDAT, HYOUT, LDHY, NOP,
     &                    HYBFLG, IOUT)
      IMPLICIT    NONE
      INTEGER     LDHYTP, NHY, LDHY
      CHARACTER   HYTP(0:LDHYTP)*(*), HYDIR(LDHY, 0:2)*(*)
      INTEGER     HYTPCD(LDHY, 0:1)
      REAL        HYDAT(LDHY, 2), HYOUT(LDHY, 3)
      INTEGER     NOP, IOUT
      LOGICAL     HYBFLG
*
      INTEGER     J, N
*
      IF (HYBFLG) THEN
                             WRITE (IOUT, 900) NOP
                                 DO N = 1, NHY
                    WRITE (IOUT, 901) (HYDIR(N, J), J = 0, 2),
     &                  HYTP(HYTPCD(N, 0)),
     &                  (HYDAT(N, J), J = 1, 2),
     &                  (HYOUT(N, J), J = 1, 3)
                                 ENDDO
      ENDIF
      RETURN
 900  FORMAT(/, 20X, 'Parameters for hydraulic structures: ', I3,
     &       /, 20X, '=======================================',
     &       /, 20X, '  [-999.99, not flow under gate]',
     &/,T61,'                 Discharge  Upstream Gate-opening',
     & /,'           Upstream    Downstream',
     & T61, ' Base       Weir     (cubic   Water    or weir  ',
     & /,'Structure   node        node',     T41,'Structure',
     & T61, ' elevation   length feet per  elevation height   ',
     & /,'name         name        name',     T41,'type',
     & T61, '   (feet)    (feet)  second)    (feet)    (feet)',
     & /,'-----        ------      -----',    T41,'----',
     & T61, '  -------- --------- ---------- --------- --------')
 901  FORMAT(3A12, T41, A20, T61, 5F10.2)
      END
*=====================================================================
* Name:       hydr_ofw
* Purpose:    Calculate overflow on weir or through pipe. Flows are dependent
*             on ponds stages, and structure types and materials.
*             In other words, flows through structures can not be
*             controlled.
*=====================================================================
      SUBROUTINE HYDR_OFW(NDSEQ, LDND,
     &                    II, HI, LO, OHI, LDARC,
     &                    NHY, HYTPCD, HYDAT, HYOUT, LDHY,
     &                    OINST, LDRES,
     &                    PERD, CONST, XF, OU)
      IMPLICIT    NONE
      INTEGER     LDND, NDSEQ(LDND),
     &            LDARC, II(LDARC), HI(LDARC), LO(LDARC)
      INTEGER     OHI(LDARC)
      INTEGER     NHY, LDHY, HYTPCD(LDHY, 0:1)
      REAL        HYDAT(LDHY, 5), HYOUT(LDHY, 3)
      INTEGER     LDRES
      REAL        OINST(LDRES), PERD, CONST, XF
      INTEGER     OU
*
      REAL        B, W, P, H1, ZVA(3), D, L, F, E, LOFLW, HIFLW, HD
      INTEGER     N, TYP, ARC, ND, RES, IFAULT
      LOGICAL     GETZVA, CHECK
      CHARACTER   NDNAM(300)*12
      COMMON      /NDNAME/NDNAM
*
      DO 100 N = 1, NHY
                               TYP = HYTPCD(N, 0)
                               ARC = HYTPCD(N, 1)
                                  ND = II(ARC)
*
*---------------Calculate upstream water depth
*
                                 RES = NDSEQ(ND)
                               ZVA(2) = OINST(RES)
                         IF (GETZVA(ND, 2, ZVA, OU)) THEN
                                     H1 = ZVA(1)
                                       ELSE
                                 PRINT *, CHAR(7)
                   PRINT *, '***ERROR*** CONVERTING STORAGE: ', ZVA(2)
                   PRINT *,'  INTO ITS ELEVATION FOR RESERVOIR: ', RES
                                  STOP !CALL EXIT
                                      ENDIF
                                HYOUT(N, 2) = H1
                          H1 = HYOUT(N, 2) - HYDAT(N, 1)
*
*------------Calculate the flow through the structure or the
*            maximum flow through the structures.
*
                              IF (H1 .LE. 0.0) THEN
*
*            The water head is lower than the base elevation
*
```

```
                              LOFLW = 0.0
                              HIFLW = 0.0
                        ELSE IF (TYP .EQ. 0) THEN
                              LOFLW = 0.0
                              HIFLW = OHI(ARC)
                        ELSE IF (TYP .EQ. 1) THEN
*
*-------------1. Flow over sharp-crested weir
*
                        B = HYDAT(N, 2)         !WEIR LENGTH.
                        P = HYDAT(N, 3)         !WEIR HEIGHT.
                            IF (P .LT. 0.0) THEN
*
*             The weir height is to be determined. It is assumed that
*             for given upstream water depth, the maximum flow over weir
*             occurs when there is no weir, i.e. p = 0.
*
                            IF (ABS(P) .GE. H1) THEN
                              LOFLW = 0.0
                                  ELSE
                CALL WEIRFLW(LOFLW, H1+P, 0.0, B, -P, 0.0, 0.0, TYP,
     &                       IFAULT)
                                  ENDIF
                   CALL WEIRFLW(HIFLW, H1, 0.0, B, 0.0, 0.0, 0.0, TYP,
     &                  IFAULT)
                          ELSE IF (H1 .LT. P) THEN
*
*              If the water level is lower than the top of the weir, no
*              flow is over the weir.
*
                               LOFLW = 0.0
                               HIFLW = 0.0
                                 ELSE
*
*              The weir height is fixed. The flow over the weir is
*              determined from the upstream level and weir height.
*              Free outfall flow is assumed.
*
                   CALL WEIRFLW(LOFLW, H1-P, 0.0, B, P, 0.0, 0.0, TYP,
     &                   IFAULT)
                               HIFLW = LOFLW
                            IF (IFAULT .NE. 0) THEN
              PRINT *, '***Error*** Calculation the flow over the'//
     &                 ' sharp-crested weir: ', N
                                  STOP
                                 ENDIF
                              HYOUT(N, 3) = P
                                 ENDIF
                     ELSE IF (TYP .EQ. 2 .OR. TYP .EQ. 3) THEN
*
*-------------2. Flow under gates on spillway or broad-crested weir
*
                          B = HYDAT(N, 2)       !WEIR LENGTH
                          P = HYDAT(N, 3)       !WEIR HEIGHT
                        W = HYDAT(N, 4)       !WEIR THICKNESS
                     E = HYDAT(N, 5)       !GATE OPENING HEIGHT
                            IF (TYP .EQ. 2) THEN
                    HD = HYDAT(N,3)    !DESIGN WATER HEAD FOR SPILLWAY
                                 ENDIF
                            IF (E .GT. 0) THEN
                   CALL GATEFLW(LOFLW, H1, 0.0, B, E, HD, TYP, IFAULT)
              IF (IFAULT .EQ. 1) THEN !GATE OPENING TOO HIGH, WEIR FLOW.
              CALL WEIRFLW(LOFLW, H1, 0.0, B, P, W, HD, TYP, IFAULT)
                          ELSE IF (IFAULT .EQ. 2) THEN
              PRINT *, '***Error*** Invalid gate type for structure'
     &                //' no: ', N
                                  STOP
                                 ENDIF
                               HIFLW = LOFLW
                              HYOUT(N, 3) = E
                                 ELSE
*
*-------------The maximum flow under gate on spillway or broad-crested
*             weir is the same as the flow over weir without gates under
*             the same water head.
*
                               LOFLW = 0.0
                 CALL WEIRFLW(HIFLW, H1, 0.0, B, P, W, HD, TYP, IFAULT)
                                 ENDIF
                        ELSE IF (TYP .EQ. 6) THEN
*
*--------------Flow through a short pipe
*
                             D = HYDAT(N, 2)
                             L = HYDAT(N, 3)
                             F = HYDAT(N, 4)
                             E = HYDAT(N, 5)
                     CALL PIPE_FLW(0, LOFLW, H1, D, L, F, E, 0.0)
                              HIFLW = LOFLW
                                 ENDIF
                  HYOUT(N,1) = LOFLW            !IN CFS
                 LO(ARC) = NINT(LOFLW * PERD * CONST * XF)
                 HI(ARC) = NINT(HIFLW * PERD * CONST * XF)
*
```

```
                                   CHECK = .FALSE.
 100  CONTINUE
      RETURN
      END
*=================================================================
* Name:        resev_dat
* Purpose:     Get seasonal reservoir surface water evaporation coefficient.
*=================================================================
      SUBROUTINE RESEV_DAT(NNODS, NDNAM, LDND, NEV, EVND, LDEV,
     &                     NPER, EVTB, LDEVTB, EVUNIT, EVFLAG,
     &                     FILNAM, NTLS, IN, OU, PN,
     &                     UNITNM, LDUNIT, CTERM, COLSTR, LDCOL)
      IMPLICIT   NONE
      INTEGER    NNODS, LDND, LDEVTB
      CHARACTER*(*) NDNAM(LDND), FILNAM
      INTEGER    NEV, LDEV, EVND(LDEV, 3), NPER, EVUNIT
      INTEGER    NTLS, IN, IU, OU, PN
      REAL       EVTB(0:LDEVTB, LDEV)
      LOGICAL    EVFLAG
      INTEGER    LDUNIT, LDCOL
      CHARACTER  UNITNM(0:LDUNIT)*(*), CTERM*(*), COLSTR(LDCOL)*(*)
*
      INTEGER    I, NREC
      LOGICAL    FLAG, ENDFIL
*
      EVFLAG = .FALSE.
*
*-------------Open data file and skip title lines
*
      IF (FILNAM .NE. ' ') THEN
                                    FLAG = .TRUE.
                                       IU = 9
               CALL IO_OPFIL(IU, 1, FILNAM, 'ENTER SEASONAL EV DATA FILE: ')
                        DO I = 1, NTLS      !SKIP TITLE LINES
                                 READ (IU, *, END = 99)
                                        ENDDO
      ELSE
                                    FLAG = .FALSE.
                                       IU = IN
                    CALL PNCK(PN, IU, ENDFIL, CTERM, COLSTR, LDCOL)
                                 IF (ENDFIL) GOTO 99
       ENDIF
*
*-------------Read units, nonal names, and data from a file and
*             print these information into general output file.
*
      CALL DATTB(NNODS, NDNAM, LDND, NEV, EVND, LDEV, EVUNIT,
     &           NREC, EVTB, LDEVTB, EVFLAG,
     &           UNITNM, LDUNIT, FILNAM, IU, OU, PN, ENDFIL,
     &           'water-surface evaporation coefficeints',
     &           CTERM, COLSTR, LDCOL)
      IF (NREC .LT. NPER) THEN
                              WRITE (OU, 805) NPER, NREC
 805     FORMAT('***WARNING*** NO. OF SEASONAL RECORDS ARE LESS',
     &          ' THAN NO. OF SEASONS.',
     &          '    NO. OF RECORDS: ', I3,
     &          '    NO. OF SEASONS: ', I3)
      ENDIF
 99   CONTINUE
      IF (FLAG) CLOSE(IU)
      RETURN
      END
*=================================================================
* Name:        resev_arc
* Purpose:     Read current evaporation coefficients and create
*              corresponding EV arcs.
*=================================================================
      SUBROUTINE RESEV_ARC(NDTYP, NDSEQ, LDND,
     &                   NARC, II, JJ, LO, HI, COST, ARTYP, LDARC,
     &                   INST, LDRES,
     &                   NDXAR, LDXAR, NEV, EVND, LDEV, EVTB, LDEVTB,
     &                   MTH, SKSC, PERD, EVTIM, XF, IU,
     &                   CTERM, COLSTR, LDCOL)
      IMPLICIT   NONE
      INTEGER    LDND, NDTYP(LDND), NDSEQ(LDND)
      INTEGER    NARC, LDARC,
     &           II(LDARC), JJ(LDARC), LO(LDARC), HI(LDARC),
     &           COST(LDARC), ARTYP(LDARC)
      INTEGER    LDRES
      REAL       INST(LDRES)
      INTEGER    LDXAR, NDXAR(LDXAR, 6)
      INTEGER    NEV, LDEV, EVND(LDEV, 3), LDEVTB
      REAL       EVTB(0:LDEVTB, LDEV), PERD, XF
      INTEGER    IU, MTH, SKSC
      LOGICAL    EVTIM
*
      INTEGER    LDCOL
      CHARACTER  CTERM*(*), COLSTR(LDCOL)*(*)
*
      INTEGER    I, J, N, ND, NNDS, RES, NV
      REAL       ZVA(3), UC(0:2), RTERM
      LOGICAL    GETZVA, TDFLAG
*
*--------------Unit conversion factor (day * ft)
```

```
*
      UC(0) = PERD / 304.8
      UC(1) = PERD / 12.0
      UC(2) = PERD
*
*----------------Read current surface water evaporation coefficient
*
      TDFLAG = .FALSE.
      IF (EVTIM) THEN
                          READ (IU, '(A)', END = 50) CTERM
                CALL  STR_DIVD(CTERM, NNDS, COLSTR, LDCOL, 0, ' ,')
                                TDFLAG = .TRUE.
      ENDIF
 50   CONTINUE
*
*---------------Calculate the reservoir surface evaporation
*
      DO I = 1, NEV
                                ND = EVND(I,1)
                            IF (EVND(I,3) .EQ. 0) THEN
                                   N = MTH
                                  ELSE
                                   N = 0
                               IF (TDFLAG) THEN
                             IF (EVND(I, 3) .LT. 0) THEN
                      READ (COLSTR(NNDS), '(F10.0)') EVTB(0,I)
                                     ELSE
                               J = EVND(I,3) + 1
                       READ (COLSTR(J), '(F10.0)') EVTB(0,I)
                                    ENDIF
                                   ELSE
                               EVTB(0,I) = 0.0
                                   ENDIF
                                  ENDIF
                            IF (NDTYP(ND) .EQ. 1) THEN
                                   RES = NDSEQ(ND)
                                 ZVA(2) = INST(RES)
                           IF (GETZVA(ND, 2, ZVA, 26)) THEN
                       RTERM = EVTB(N,I) * UC(EVND(I,2)) * ZVA(3)
*
*            If the available water is less than the calculated evaporation,
*            the evaporation is set to equal to one half of the available water
*
                             IF (RTERM .GT. ZVA(2)) THEN
                           NV = NINT(ZVA(2) * 0.5 * XF)
                                     ELSE
                              NV = NINT(RTERM * XF)
                                    ENDIF
*
                     CALL ARCVAL(NARC, II, JJ, LO, HI, COST, ARTYP,
     &                  LDARC, ND, SKSC, NV, NV, 0, 4)
                                NDXAR(ND, 2) = NARC
                                   ELSE
                PRINT *, '***Error*** Error in transforming water volume'
                  PRINT *, ' to its corresponding elevation and area.'
                   PRINT *, ' Pond: ', ND, ' Water volume: ', ZVA(2)
                                     STOP
                                   ENDIF
                                  ENDIF
      ENDDO
 99   RETURN
      END
*=====================================================================
* Name:       rnof_dat
* Purpose:    Read data for surface runoff.
*=====================================================================
      SUBROUTINE RNOF_DAT(NDNAM, LDND,
     &           NRNOF, RNOFND, LDRNOF, RNOFTB, LDRFTB, A5DR,
     &           FILNAM, NTLS, IN, OU, PN,
     &           CTERM, COLSTR, LDCOL)
      IMPLICIT   NONE
      INTEGER    LDND
      CHARACTER  NDNAM(LDND)*(*)
      INTEGER    NRNOF, LDRNOF, RNOFND(LDRNOF), LDRFTB
      REAL       RNOFTB(LDRNOF, 0:LDRFTB), A5DR(LDRNOF, 5)
      CHARACTER  FILNAM*(*)
      INTEGER    NTLS, IN, OU, PN
*
      INTEGER    LDCOL
      CHARACTER  CTERM*(*), COLSTR(LDCOL)*(*)
*
      INTEGER    I, J, K, IU
      LOGICAL    ERR, CN, FLAG, ENDFIL
      INTEGER    SAVOPT
      COMMON     /SAVOPT/ SAVOPT
*
      IF (FILNAM .NE. ' ') THEN
                               FLAG = .TRUE.
                                  IU = 9
              CALL IO_OPFIL(IU, 1, FILNAM,'ENTER RUNOFF DATA FILE: ')
                               DO I = 1, NTLS
                           READ (IU, *, END = 100)
                                  ENDDO
      ELSE
```

```
                              FLAG = .FALSE.
                              IU = IN
                  CALL PNCK(PN, IU, ENDFIL, CTERM, COLSTR, LDCOL)
                              IF (ENDFIL) GOTO 99
                    READ (IU, *) ! SKIP THE VARIABLE LIST LINE
      ENDIF
*
*-----------------Output title
*
      IF (SAVOPT .EQ. 0 .OR. SAVOPT .EQ. 1) THEN
                              WRITE(OU, 800) PN
 800     FORMAT(//, 'Part', I2, ': Surface-runoff data',
     &        /, '==========================',
     & /, T13, '     Initial criterion 5-day rainfall',
     & /, T11,'---------------------------------------',
     & /, T13, ' Antecedent       Dry         Wet           SCS',
     &        '    Drainage',
     & /, T11,'5-day rainfall condition   condition         curve',
     &        '        area',
     & /, 'Name',
     &    T13, '   (inches)   (inches)     (inches)       number',
     &        '    (acres)',
     & /, '------- ',
     &    T13, ' ---------   ---------   ---------    ---------',
     &    ' ---------')
      ENDIF
*
*---------------Read data
*
      NRNOF = 0
 5    CONTINUE
      READ (IU, '(A)', END = 100) CTERM
      CALL STR_DIVD(CTERM, K, COLSTR, LDCOL, 0, ' ,')
      IF ((K-2)  .GT. LDRFTB) THEN
                              PRINT '(A)', CHAR(7)
            PRINT *, '***ERROR*** THE DIMENSION OF RUNOFF TABLE IS NOT BIG'
     &            // ' ENOUGH'
                              STOP !CALL EXIT
      ENDIF
      IF (CN(COLSTR(1), 'FINISH', 1)) GOTO 100
      NRNOF = NRNOF + 1
      DO J = 1, K
                              IF (J .EQ. 1) THEN
                CALL NAMNUM(LDND, NDNAM, COLSTR(1), RNOFND(NRNOF), 0, ERR)
                              ELSE
                     READ (COLSTR(J), '(F15.0)') RNOFTB(NRNOF, J-2)
                              ENDIF
      ENDDO
*
*-------------Assign intial 5-day antecedent rainfall (daily).
*
      DO J = 1, 5
                     A5DR(NRNOF, J) = RNOFTB(NRNOF, 0) / 5.0
      ENDDO
      GOTO 5
 100  CONTINUE
      IF (FLAG) CLOSE(IU)
*
*----------------Print the data
*
      IF (SAVOPT .EQ. 0 .OR. SAVOPT .EQ. 1) THEN
                              DO I = 1, NRNOF
                WRITE(OU, 901) NDNAM(RNOFND(I)), (RNOFTB(I,J), J=0,LDRFTB)
 901        FORMAT(A12, 10F12.2)
                              ENDDO
      ENDIF
 99   RETURN
      END
*===============================================================
* Name:       rnof_arc
* Purpose:    Generate surface-runoff arcs.
*===============================================================
      SUBROUTINE RNOF_ARC(NDTYP, NDSEQ, LDND,
     &                   NARC, II, JJ, LO, HI, COST, ARTYP, LDARC,
     &                   INST, LDRES,
     &                   NDXAR, LDXAR, NRAIN, RAINND, LDRAIN, RAINTY,
     &                   NRNOF, RNOFND, LDRNOF, RNOFTB, LDRFTB,
     &                   A5DR, PERD,
     &                   SKSC, XF, IU,
     &                   CTERM, COLSTR, LDCOL)
      IMPLICIT   NONE
      INTEGER    LDND, NDTYP(LDND), NDSEQ(LDND)
      INTEGER    NARC, LDARC,
     &           II(LDARC), JJ(LDARC), LO(LDARC), HI(LDARC),
     &           COST(LDARC), ARTYP(LDARC)
      INTEGER    LDRES
      REAL       INST(LDRES)
      INTEGER    LDXAR, NDXAR(LDXAR, 6)
      INTEGER    NRAIN, LDRAIN, RAINND(LDRAIN, 3), RAINTY
      INTEGER    NRNOF, LDRNOF, RNOFND(LDRNOF), LDRFTB
      REAL       RNOFTB(LDRNOF, 0:LDRFTB), A5DR(LDRNOF, 5), PERD
      INTEGER    IU, SKSC
      REAL       XF
*
```

```
      INTEGER    LDCOL
      CHARACTER  CTERM*(*), COLSTR(LDCOL)*(*)
*
      INTEGER    I, NRN, ND, NNDS, RES, NV, RFND
      REAL       ZVA(3), UC(0:2,2), RNOF, RNOF2, RAIN, AREA, SCSCN,
     &           MXRET
      LOGICAL    GETZVA
*
      UC(0,1) = 12.0           ! FT      --> INCHES
      UC(1,1) = 1.0            ! INCHES --> INCHES
      UC(2,1) = 0.03937        ! MM      --> INCHES
      UC(0,2) = 1.0            ! AC-FT/D--> AC-FT/D
      UC(1,2) = 86400./43560.  ! CFS     --> AC-FT/D
      UC(2,2) = 1.0/43560      ! CFD     --> AC-FT/D
*
      DO I = 1, NRAIN
                                  NDXAR(ND, 3) = 0
      ENDDO
*
*---------------Read the rainfall data
*
      READ (IU, '(A)', END = 999) CTERM
      CALL STR_DIVD(CTERM, NNDS, COLSTR, LDCOL, 0, ' ,')
      DO NRN = 1, NRAIN
                                 ND = RAINND(NRN, 1)
                           IF (RAINND(NRN, 3) .LT. 0) THEN
                           READ (COLSTR(NNDS), '(F15.0)') RAIN
                                        ELSE
                          READ (COLSTR(NRN+1), '(F15.0)') RAIN
                                       ENDIF
                             IF (RAINTY .EQ. 1) THEN
                RAIN = RAIN * UC(RAINND(NRN, 2), 1)          ! CONVERT TO INCHES
*
*---------------Find the drainge area and SCS curve number
*
                                       RFND = 1
                                      AREA = 0.0
                    DO WHILE (RFND .LE. NRNOF .AND. RNOFND(RFND) .NE. ND)
                                     RFND = RFND + 1
                                         ENDDO
                             IF (RFND .GT. NRNOF) THEN
                                        AREA = 0.0
                                       SCSCN = 0.0
                                          ELSE
                                 AREA = RNOFTB(RFND, 4)
                    CALL SCS_CN(SCSCN, RFND, RNOFTB, LDRNOF, LDRFTB, A5DR,
     &                   RAIN, PERD)
                                         ENDIF
*
*---------------Find the reservoir water surface area
*
                                     ZVA(3) = 0.0
                                      RNOF2 = 0.0
                             IF (NDTYP(ND) .EQ. 1) THEN
                                     RES = NDSEQ(ND)
                                   ZVA(2) = INST(RES)
                           IF (GETZVA(ND, 2, ZVA, 26)) THEN
                          RNOF2 = RAIN / 12.0 * ZVA(3) * XF
                                          ENDIF
                                         ENDIF
*
*-----------------surface runoff
*
                                AREA = AREA - ZVA(3)
                                      RNOF = 0.0
                           IF (AREA .GT. 0.0000001) THEN
                             MXRET = 1000.0 / SCSCN - 10.0
                            IF (RAIN .GT. 0.2 * MXRET) THEN
                    RNOF = (RAIN - 0.2 * MXRET) * (RAIN - 0.2 * MXRET)
     &                / (RAIN + 0.8 * MXRET) / 12.0 * AREA * XF
                                          ELSE
                                     RNOF = 0.0
                                          ENDIF
                                         ENDIF
*
*-------------Create runoff arc
*
                                 NV = NINT (RNOF + RNOF2)
                          ELSE IF (RAINTY .EQ. 2) THEN
                   RAIN = RAIN * UC(RAINND(NRN, 2), 2)          !TO AC-FT/D
                             NV = NINT(RAIN * PERD * XF)
                                        ENDIF
                                 IF (NV .GT. 0) THEN
                       CALL ARCVAL(NARC, II, JJ, LO, HI, COST, ARTYP,
     &                LDARC, SKSC, ND, NV, NV, 0, 5)
                                  NDXAR(ND, 3) = NARC
                                        ENDIF
      ENDDO
 999  RETURN
      END
*=================================================================
* Name:       SCS_CN
* Purpose:    Calculate the SCS curve number based on antecedent 5-day
*             rainfall conditions.
```

```
*=====================================================================
      SUBROUTINE SCS_CN(CN, RFND, RNOFTB, LDRNOF, LDRFTB, A5DR, RAIN,
     &                  PERD)
      IMPLICIT   NONE
      INTEGER    RFND, LDRNOF, LDRFTB
      REAL       CN, RNOFTB(LDRNOF, 0:LDRFTB), A5DR(LDRNOF, 5), RAIN
      REAL       PERD
*
      INTEGER    I
      REAL       A5DR0
*
      CN = RNOFTB(RFND, 3)
      A5DR0 = RNOFTB(RFND, 0)
      IF (A5DR0 .LT. RNOFTB(RFND, 1)) THEN
                            CN = 4.2 * CN / (10.0 - 0.058 * CN)
      ELSE IF (A5DR0 .GT. RNOFTB(RFND, 2)) THEN
                            CN = 23.0 * CN / (10.0 + 0.13 * CN)
      ENDIF
*
*----------Update the 5-day antecedent rainfalls
*
      IF (INT(PERD) .GT. 4) THEN
                                      DO I = 1, 5
                               A5DR(RFND, I) = RAIN / PERD
                                          ENDDO
      ELSE
                               DO I = 5, INT(PERD)+1, -1
                             A5DR(RFND, I) = A5DR(RFND, I-1)
                                          ENDDO
                                   DO I = 1, INT(PERD)
                               A5DR(RFND, I) = RAIN / PERD
                                          ENDDO
      ENDIF
      A5DR0 = 0.0
      DO I = 1, 5
                             A5DR0 = A5DR0 + A5DR(RFND, I)
      ENDDO
      RNOFTB(RFND, 0) = A5DR0
      RETURN
      END
*=====================================================================
* Name:       weirflw
* Purpose:    Calculate flow over a sharp-crested weir.
*=====================================================================
      SUBROUTINE WEIRFLW(Q, H, HT, B, P, W, HD, WEIRTYP, IFAULT)
      IMPLICIT   NONE
      INTEGER    WEIRTYP, IFAULT
      REAL       Q, H, HT, B, P, W, HD
*
      REAL       CD
*
*-----------Determine the coefficients of discharge
*
      IFAULT = 0
      IF (WEIRTYP .EQ. 1) THEN
                                  CALL CDSCW(CD, H, P)
      ELSE IF (WEIRTYP .EQ. 2) THEN
                                  CALL CDRCW(CD, H, HD)
      ELSE IF (WEIRTYP .EQ. 3) THEN
                               CALL CDBCW(CD, H, HT, P, W)
      ELSE
                                        IFAULT = 1
      ENDIF
      Q = CD * B * SQRT(2.0 * 32.17) * H**1.5
      RETURN
      END
*=====================================================================
* Name:       weirhite
* Purpose:    Determine the weir height for a given flwo
*=====================================================================
      SUBROUTINE WEIRHITE(Q, H, HT, B, P, W, HD, WEIRTYP, IFAULT)
      IMPLICIT   NONE
      REAL       Q, H, HT, B, P, W, HD
      INTEGER    WEIRTYP, IFAULT
*
      REAL       X1, X2, TOL, QQ
      DATA       TOL / 0.0001/
*
      IFAULT = 0
      IF (WEIRTYP .NE. 1) THEN ! ONLY SHARP-CRESED WEIR IS SUPPORTED.
                                        IFAULT = 2
                                          RETURN
      ENDIF
*
*     Note: h is the water depth approaching the weir, referring to
*          the base of weir, i.e. river bed.
*
*
*----------Find the weir height.
*
      X1 = 0.0
      X2 = H
      QQ = 0.0
      DO WHILE(ABS(QQ - Q) / Q .GT. TOL)
```

```
                                P = 0.5 * (X1 + X2)
                 CALL WEIRFLW(QQ, H-P, HT-P, B, P, W, HD, WEIRTYP, IFAULT)
           IF (QQ .GT. Q) THEN        !INCREASE THE HEIGHT OF WEIR TO REDUCE FLOW
                                       X1 = P
                  ELSE IF (QQ .LT. Q) THEN   !DECREASE THE HEIGHT OF WEIR
                                       X2 = P
                                      ENDIF
      ENDDO
*
      RETURN
      END
*==================================================================
*     Calculation of coefficient of discharge for sharp-crested weir
*==================================================================
      SUBROUTINE CDSCW(CD, H, P)
      IMPLICIT   NONE
      REAL       CD, H, P
*
*------------Discharge coefficient for free outflow
*
      CD = 1.06*((14.14*P/(8.15*P+H))**10.0+(H/(H+P))**15.0)**(-0.1)
      CD = 2.0 / 3.0 * CD
      RETURN
      END
*==================================================================
*     Calculation of coefficient of discharge for round-crested weir
*==================================================================
      SUBROUTINE CDRCW(CD, H, HD)
      IMPLICIT   NONE
      REAL       CD, H, HD
*
      REAL       HHD
*
*------------Discharge coefficient for free outflow
*
      CD = 0.502
      IF (HD .GT. 0.0001 .AND. H .GT. 0.0001) THEN
                                    HHD = H / HD
                   CD = 0.3849 + 0.3849 * 4.0 * HHD / (9.0 + 5.0 * HHD)
      ENDIF
      RETURN
      END
*==================================================================
*     Calculation of coefficient of discharge for broad-crested weir
*==================================================================
      SUBROUTINE CDBCW(CD, H, HT, P, W)
      IMPLICIT   NONE
      REAL       CD, H, HT, P, W
*
      INTEGER    ISHAPE, IFWTYP
      REAL       CS, CC, PH, HTH, HW
*
*
*--------------Coefficient of discharge for free outflow
*
      IF (W .GT. 0.00001) THEN
                                      HW = H / W
              CD = 0.5 + 0.1 * ((HW**5+1500.0*HW**13)/(1.0+1000*HW**3))**0.1
                                  CD = 2.0 / 3.0 * CD
      ELSE
                                       PH = P/H
                                      ISHAPE = 0
             IF (ISHAPE .EQ. 0) THEN   ! VERTICAL FACE WITH ROUNDED ENTRANCE.
                                    IF (PH .LE. 3) THEN
                       CD = 0.36 + 0.01 * (3.0 - PH) / (1.2 + 1.5 * PH)
                                         ELSE
                                       CD = 0.36
                                        ENDIF
                   ELSE IF (ISHAPE .EQ. 1) THEN ! VERTICAL FACE.
                                    IF (PH .LE. 3) THEN
                      CD = 0.32 + 0.01 * (3.0 - PH) / (0.46 + 0.75 * PH)
                                         ELSE
                                       CD = 0.32
                                        ENDIF
                                      ENDIF
      ENDIF
*
*----------Estimate the submerged coefficient
*          The equation is the regressed equation obtained out of
*          data from Chengdu, 1977.
*
      HTH = HT / H
      CS = 1.0
      IF (HTH .GE. 0.8) THEN
                                    IFWTYP = 2
                               IF (HTH .LT. 0.86) THEN
                               CS = 1.6892 - 0.8571 * HTH
                                        ELSE
                  CS = 1.2504 - 0.06897 * HTH / (1.0 - 0.9388 * HTH)
                                       ENDIF
                                IF (CS .GT. 1.0) THEN
                                      CS = 1.0
                                      ENDIF
      ENDIF
```

```
*
      CC = 1.0
      CD = CD * CS * CC
      RETURN
      END
*==================================================================
* Name:        gateflw
* Purpose:     Calculate dicharge under gates.
* Author:      Xiaodong Jian
* Date:        12/10/96
*==================================================================
      SUBROUTINE GATEFLW(Q, H, HT, B, E, HD, GATETY, IFAULT)
      IMPLICIT   NONE
      REAL       Q, H, HT, B, E, HD
      INTEGER    GATETY, IFAULT
*
      REAL       CD, EH, EPS
      INTEGER    IFWTYP
      DATA       EPS /1.0E-10/
*
*--------------If e = 0, find the maximum flow under gate
*
      IFAULT = 0
      IF (ABS(E) .LT. EPS) THEN
                         IF (GATETY .EQ. 2) THEN          !SPILLWAY
                                        E = 0.75 * H
                        ELSE IF (GATETY .EQ. 3) THEN    !BROAD WERI
                                        E = 0.65 * H
                                          ENDIF
      ENDIF
*
*-------------Check gate opeing height.
*
      EH = E / H
      IF (GATETY .EQ. 2 .AND. EH .GT. 0.75+EPS) THEN
                                         IFAULT = 1
      ELSE IF (GATETY .EQ. 3 .AND. EH .GT. 0.65+EPS) THEN
                                         IFAULT = 1
      ENDIF
*
*--------Calculate the discharge coefficient.
*
      IF (GATETY .EQ. 2) THEN ! VERTICAL FLAT GATE ON SPILLWAY.
                                   CALL CDGTSP(CD, H, E, HD)
      ELSE IF (GATETY .EQ. 3) THEN ! VERTICAL FLAT GATE ON BROAD-CRESTED WEIR.
                               CALL CDGTBW(CD, H, HT, E, IFWTYP)
      ELSE
                                         IFAULT = 2
                                           RETURN
      ENDIF
*
*---------Calculate the discharge under gate
*
      Q = CD * B * E * SQRT(2.0 * 32.17 * H)
*
      RETURN
      END
*==================================================================
* Name:        gatehite
* Purpose:     Calculate the gate opening height for a given dicharge
*              under gates.
* Author:      Xiaodong Jian
* Date:        01/24/97
*==================================================================
      SUBROUTINE GATEHITE(Q, H, HT, B, E, HD, GATETY, IFAULT)
      IMPLICIT   NONE
      REAL       Q, H, HT, B, E, HD
      INTEGER    GATETY, IFAULT
*
      REAL       X1, X2, TOL, QQ
      DATA       TOL / 0.0001/
*
*---------maximum flow under gate.
*
      E = 0.0
      CALL GATEFLW(QQ, H, HT, B, E, HD, GATETY, IFAULT)
      IF (QQ .LT. Q) THEN
                                        E = -999.99
                                         IFAULT = 1
                                           RETURN
      ENDIF
*
*---------Find the gate opening height.
*
      X1 = 0
      X2 = H
      DO WHILE(ABS(QQ - Q) / Q .GT. TOL)
                                    E = 0.5 * (X1 + X2)
                    CALL GATEFLW(QQ, H, HT, B, E, HD, GATETY, IFAULT)
                                    IF (QQ .GT. Q) THEN
                                           X2 = E
                                  ELSE IF(QQ .LT. Q) THEN
                                           X1 = E
                                          ENDIF
```

```
      ENDDO
*
*
      RETURN
      END
*=================================================================
* Name:        cdgtbw
* Purpose:     Compute the discharge coefficien for flow under sluice-
*              gate on the broad weir.
* Author:      Xiaodong Jian
* Date:        12/17/96
*=================================================================
      SUBROUTINE CDGTBW(CD, H, HT, E, IFWTYP)
      IMPLICIT   NONE
      REAL       CD, H, HT, E
      INTEGER    IFWTYP
*
      REAL       HMAX
*
      IFWTYP = 1
      CD = 0.611 * ((H - E) / (H + 15.0*E))**0.072
*
*-----------Check flow type
*
      IF (E .NE. 0) THEN
                       HMAX = 0.81 * HT * (HT / E)**0.72
               IF (H .GT. HT .AND. H .LT. HMAX) THEN ! SUBMERGED FLOW
                                     IFWTYP = 2
              CD = CD * (H - HT)**0.7 / (0.32 * (0.81 * HT * (HT/E)**0.72
     &        - H)**0.7 + (H - HT)**0.7)
                                  ENDIF
      ELSE
                               IFWTYP = 0
      ENDIF
      RETURN
      END
*==============================================================
* cdgtsp -- Calculate the discharge coefficient for plane gate on
*           spillway
*==============================================================
      SUBROUTINE CDGTSP(CD, H, E, HD)
      IMPLICIT   NONE
      REAL       CD, H, E, HD
*
      REAL       EH, EHD, EPS
      DATA       EPS /0.0001/
*
      EH = E / H
*
      IF (HD .GT. EPS) THEN
                                EHD = E / HD
             CD = 0.495 / EH * (1.0-(1.0-EH)**1.5) * (0.1667+EHD)**0.1111
      ELSE
                            CD = 0.65 - 0.186 * EH
      ENDIF
      RETURN
      END
*=================================================================
* Name:        pipe_flw
* Purpose:     Calculate flows through a pipe.
*=================================================================
      SUBROUTINE PIPE_FLW(PIPTYP, FLW, H, DIAM, LENG, FRIC, ENLOS,
     &                    ROUGH)
      IMPLICIT   NONE
      INTEGER    PIPTYP
      REAL       FLW, H, DIAM, LENG, FRIC, ENLOS, ROUGH
*
      REAL       DISCH, R, J, A
*
      IF (PIPTYP .EQ. 0) THEN  !SHORT PIPE
                DISCH = 1.0 / SQRT(1.0 + FRIC * LENG / DIAM + ENLOS)
            FLW = DISCH * (3.14159 * DIAM**2 / 4.0) * SQRT(2.0 * 32.17 * H)
      ELSE                     !LONG PIPE
*
*-----------Long pipe: Q = K J**0.5, where K = A C R**0.5, A -- area
*                     C -- Chezy coefficient (or manning coefficient)
*                     J -- Hydraulic slope.
*               then Q = 1.486 * A / rough * R**(2/3) * J**0.5
*
                               R = DIAM / 4.0
                                J = H / LENG
                    A = 3.14159 * DIAM * DIAM / 4.0
                 FLW = 1.486 * A * R**0.6667 * SQRT(J) / ROUGH
      ENDIF
      RETURN
      END
*=================================================================
* Name:        fb_dat
* Purpose:     Open a seasonal flow bound data file for selected arcs.
*=================================================================
      SUBROUTINE FB_DAT(II, JJ, ARTYP, LDARC, NDNAM, LDND,
     &                  NFBAR, FBAR, LDFBAR, FBTB,  LDFBTB,
     &                  FBUNIT, FBFLAG, FILNAM, NTLS, IN, PN, OU,
     &                  PTDWAR, LDPTDW, NDDWAR, LDDWAR,
```

```
     &                   UNITNM, LDUNIT, CTERM, COLSTR, LDCOL)
      IMPLICIT   NONE
      INTEGER    LDARC, II(LDARC), JJ(LDARC), ARTYP(LDARC), LDND
      CHARACTER  NDNAM(LDND)*(*), FILNAM*(*)
      INTEGER    NFBAR, LDFBAR, FBAR(0:7, LDFBAR), FBUNIT
      INTEGER    LDFBTB, LDPTDW, LDDWAR
      REAL       FBTB(0:LDFBTB, LDFBAR)
      LOGICAL    FBFLAG
      INTEGER    PTDWAR(LDPTDW, 2), NDDWAR(LDDWAR)
      INTEGER    NTLS, IN, PN, OU, LDUNIT, LDCOL
      CHARACTER*(*)  UNITNM(0:LDUNIT), CTERM, COLSTR(LDCOL)
*
      INTEGER    IU
      INTEGER    NREC, SL
      INTEGER    I, J, K, L, M, N
      LOGICAL    ERR, FLAG, ENDFIL
      LOGICAL    CN
      INTEGER    SAVOPT
      COMMON     /SAVOPT/ SAVOPT
*
      FBFLAG = .FALSE.
      NFBAR = 0
*
*-------------Open data file and skip title lines
*
      IF (FILNAM .NE. ' ') THEN
                                      FLAG = .TRUE.
                                         IU = 9
              CALL IO_OPFIL(IU, 1, FILNAM,'ENTER SEASONAL FLOW BOUND FILE: ')
                                      DO I = 1, NTLS
                                   READ (IU, *, END = 99)
                                          ENDDO
      ELSE
                                      FLAG = .FALSE.
                                        IU = IN
                      CALL PNCK(PN, IU, ENDFIL, CTERM, COLSTR, LDCOL)
                                   IF (ENDFIL) GOTO 99
      ENDIF
      CALL FB_HED(NDNAM, LDND, II, JJ, ARTYP, LDARC,
     &            NFBAR, FBAR, LDFBAR, FBUNIT,
     &            ENDFIL,
     &            PTDWAR, LDPTDW, NDDWAR, LDDWAR, FILNAM, IU,
     &            CTERM, COLSTR, LDCOL, ERR)
      IF (ERR) STOP !CALL EXIT
      IF (ENDFIL) GOTO 99
      FBFLAG = .TRUE.
*
*-----------------Read data
*
      NREC = 0
 30   CONTINUE
      READ (IU, '(A)', END = 99) CTERM
      IF (CN(CTERM, 'FINISH', 1)) GOTO 99
      CALL  STR_DIVD(CTERM, J, COLSTR, LDCOL, 0, ' ,')
      IF (J .NE. (NFBAR + 1) ) THEN
                            PRINT *, '***ERROR*** FILE = ', FILNAM
                           PRINT *, '        FOR TIME = ',COLSTR(1)
                                      STOP !CALL EXIT
      ENDIF
*
      NREC = NREC + 1
      DO J = 1, NFBAR
                        READ (COLSTR(J+1), '(F10.0)') FBTB(NREC, J)
      ENDDO
      GOTO 30
 99   CONTINUE
      IF (FLAG) CLOSE(IU)
*
*---------------Print FB information into the general output file
*
      IF (NREC .GT. 0) THEN
                                    ENDFIL = .FALSE.
                                     FBFLAG = .TRUE.
                           IF (SAVOPT .EQ. 0 .OR. SAVOPT .EQ. 1) THEN
                                  CALL STR_LEN(UNITNM(FBUNIT), K)
                              WRITE(OU, 801) PN, UNITNM(FBUNIT)(1:K),
     &                        ('=', I = 1, K)
 801         FORMAT(//, 'Part ', I2, ': Seasonal flow bounds, in ', A,
     &            /, '=================================', 10A)
*
                                      DO K = 1, NFBAR, 8
                                    IF ((K+7) .GT. NFBAR) THEN
                                          L = NFBAR
                                               ELSE
                                          L = K + 7
                                               ENDIF
*
*-------------1. Upstream and downstream nodal names
*
                                 WRITE(OU, 805) '   Upstream node:',
     &                 (NDNAM(FBAR(1,J))(1:SL(NDNAM(FBAR(1,J)))), J = K, L)
                                 WRITE(OU, 805) ' Downstream node:',
     &                 (NDNAM(FBAR(2,J))(1:SL(NDNAM(FBAR(2,J)))), J = K, L)
 805            FORMAT(A, T20, 10A12)
```

```
*
*--------------2. Flow bounds index
*
                             WRITE (OU, 806) (FBAR(3, J), J = K, L)
 806          FORMAT(' Flow zone index:', T20, 10I12)
*
*--------------3. Arc number
*
                             WRITE (OU, 807) (FBAR(0, J), J = K, L)
 807          FORMAT('           Arcs:', T20, 10I12)
*
                                        DO N = 4, 5
                                         M = 1
                                       CTERM = ' '
                                       DO J = K, L
                                 IF (FBAR(N,J) .NE. 0) THEN
                           WRITE(CTERM(M:M+11), '(I12)') FBAR(N,J)
                                           ENDIF
                                        M = M + 12
                                         ENDDO
                                IF (CTERM .NE. ' ') THEN
                                WRITE (OU, 809) CTERM(1:50)
 809                FORMAT ('     EXTENDED ARC:', T20, A)
                                         ENDIF
                                           ENDDO
*
*--------------4. Data matrix
*
                                      DO I = 1, NREC
                          WRITE (OU, 810) I, (FBTB(I, J), J = K, L)
 810             FORMAT (5X, I5, T20, 10F12.2)
                                           ENDDO
                                          ENDDO
                                        ENDIF
      ENDIF
*
      RETURN
      END
*=====================================================================
* Name:       fb_fil
* Purpose:    Open a time series file header.
*=====================================================================
      SUBROUTINE FB_FIL(NDNAM, LDND, II, JJ, ARTYP, LDARC,
     &                  PTDWAR, LDPTDW, NDDWAR, LDDWAR,
     &                  NFBAR, FBAR, LDFBAR, FBUNIT,
     &                  UNITNM, LDUNIT, FBFLAG, FBFIL, FILNAM,
     &                  NTLS, IU, OU, CTERM, COLSTR, LDCOL)
      IMPLICIT   NONE
      INTEGER    LDND, LDARC, II(LDARC), JJ(LDARC), ARTYP(LDARC)
      INTEGER    LDPTDW, LDDWAR, PTDWAR(LDPTDW, 2), NDDWAR(LDDWAR)
      INTEGER    NFBAR, LDFBAR, FBAR(0:7, LDFBAR), FBUNIT, LDUNIT
      CHARACTER*(*)  NDNAM(LDND), UNITNM(0:LDUNIT), FILNAM
      INTEGER    NTLS, IU, OU
      LOGICAL    FBFLAG, FBFIL
      INTEGER    LDCOL
      CHARACTER  CTERM*(*), COLSTR(LDCOL)*(*)
*
      INTEGER    I, J, K, L, M, NREC, SL, N, FBIDX(100)
      LOGICAL    ERR, ENDFIL
      INTEGER    SAVOPT
      COMMON     /SAVOPT/ SAVOPT
*
      FBFIL = .FALSE.
      IF (FILNAM .EQ. ' ') THEN
                                      GOTO 99
      ENDIF
      CALL IO_OPFIL(IU, 1, FILNAM, 'ENTER FB DATA FILE: ')
*
*-----Read title lines, flow unit, upstream and downstream nodal names,
*     and flow zone index. And find the corresponding arc and nodal
*     numbers.
*
      DO I = 1, NTLS
                                    READ (IU, *)
      ENDDO
      CALL FB_HED(NDNAM, LDND, II, JJ, ARTYP, LDARC,
     &            NFBAR, FBAR, LDFBAR, FBUNIT,
     &            ENDFIL,
     &            PTDWAR, LDPTDW, NDDWAR, LDDWAR, FILNAM, IU,
     &            CTERM, COLSTR, LDCOL, ERR)
      IF (ERR) STOP !CALL EXIT
      IF (ENDFIL) GOTO 99
      FBFIL = .TRUE.
      FBFLAG = .TRUE.
*
      IF (FBFIL) THEN
*
*--------No. of time-dependent arcs
*
                                   DO I = 1, 100
                                     FBIDX(I) = 0
                                      ENDDO
                                      N = 0
                                  DO I = 1, NFBAR
```

```
                                J = FBAR(7,I)
                             IF (J .NE. 0) THEN
                                N = N + 1
                                FBIDX(J) = I
                             ENDIF
                          ENDDO
*
*------------Save file information
*
                   IF (SAVOPT .EQ. 0 .OR. SAVOPT .EQ. 1) THEN
                      CALL NORECS(IU, NREC)
                      WRITE (OU, 800) FILNAM, UNITNM(FBUNIT), N, NREC
 800        FORMAT (
     &       //, 'Summary for flow-bound data file: ',
     &        /, '=================================',
     &        /, 12X, '          File name: ', A
     &        /, 12X, '          Data unit: ', A
     &        /, 12X, '     Number of arcs: ', I4,
     &        /, 12X, '  Number of records: ', I4)
*
                      DO J = 1, N, 5
                         K = J + 4
                         IF ((J+4) .GT. N) THEN
                            K = N
                         ELSE
                            K = J + 4
                         ENDIF
                         WRITE (OU, 805)
     &        (NDNAM(FBAR(1,FBIDX(I)))(1:SL(NDNAM(FBAR(1,FBIDX(I))))),
     &                     I = J, K)
                         WRITE (OU, 806)
     &        (NDNAM(FBAR(2,FBIDX(I)))(1:SL(NDNAM(FBAR(2,FBIDX(I))))),
     &                     I = J, K)
                      WRITE (OU, 807) (FBAR(3, FBIDX(I)), I = J, K)
                      WRITE (OU, 808) (FBAR(0, FBIDX(I)), I = J, K)
 805        FORMAT ('  List of upstream nodal names: ', 5A10)
 806        FORMAT ('List of downstream nodal names: ', 5A10)
 807        FORMAT ('     List of flow-zone indexes: ', 5I10)
 808        FORMAT ('           List of arc numbers: ', 5I10)
*
                         DO L = 4, 5
                            M = 1
                            CTERM = ' '
                            DO I = J, K
                               IF (FBAR(L,FBIDX(I)) .NE. 0) THEN
                     WRITE(CTERM(M:M+9), '(I10)') FBAR(L,FBIDX(I))
                               ENDIF
                               M = M + 10
                            ENDDO
                            IF (CTERM .NE. ' ') THEN
                            WRITE (OU, 811) CTERM(1:50)
 811              FORMAT ('         EXTENDED ARC NUMBERS: ', A)
                            ENDIF
                         ENDDO
                      ENDDO
                   ENDIF
      ENDIF
*
 99   RETURN
      END
*=================================================================
* Name:       fb_arc
* Purpose:    Read current flow bounds and modify the flow bounds in the
*             arcs.
*=================================================================
      SUBROUTINE FB_ARC(LO, HI, ARTYP, LDARC,
     &                  NFBAR, FBAR, LDFBAR, FBTB, LDFBTB, FBFIL,
     &                  MTH, PERD, XF, IU,
     &                  CTERM, COLSTR, LDCOL)
      IMPLICIT   NONE
      INTEGER    LDARC, LO(LDARC), HI(LDARC), ARTYP(LDARC)
      INTEGER    NFBAR, LDFBAR, FBAR(0:7,LDFBAR), LDFBTB
      REAL       FBTB(0:LDFBTB, LDFBAR)
      LOGICAL    FBFIL
*
      INTEGER    IU, MTH
      REAL       PERD, XF
*
      INTEGER    LDCOL
      CHARACTER  CTERM*(*), COLSTR(LDCOL)*(*)
*
      INTEGER    I, J, N, ARC, ARC2, NV, ZNIDX
      REAL       UC(0:2)
      LOGICAL    STRM, TDFLAG
*
*------------Unit conversion factor: 0 -- ac-ft --> ac-ft
*                                    1 -- cfs   --> ac-ft
*                                    2 -- cfd   --> ac-ft
*
      UC(0) = 1.0
      UC(1) = PERD * 86400.0 / 43560.0
      UC(2) = PERD / 43560.0
*
*----------------Read current flow bounds
```

```
*
      TDFLAG = .FALSE.
      IF (FBFIL) THEN
                     READ (IU, '(A)', END = 50) CTERM
            CALL  STR_DIVD(CTERM, I, COLSTR, LDCOL, 0, ' ,')
                     TDFLAG = .TRUE.
      ENDIF
 50   CONTINUE
*
*--------------Change flow bounds for arcs
*
      DO 100 J = 1, NFBAR
*
*--------Get a flow bound
*
         IF (FBAR(7, J) .EQ. 0) THEN
            N = MTH
         ELSE
            N = 0
            IF (TDFLAG) THEN
               I = FBAR(7, J) + 1
               READ (COLSTR(I), '(F10.0)') FBTB(0,J)
            ELSE
               FBTB(0,J) = 0.0
               GOTO 100
            ENDIF
         ENDIF
*
         ARC = FBAR(0, J)
         ZNIDX = FBAR(3, J)
         WRITE (CTERM, '(I4)') ARTYP(ABS(ARC))
         IF (CTERM(3:3) .EQ. '1') THEN
            STRM = .TRUE.
            ARC2 = IABS(ARC) + 1
         ELSE
            ARC2 = 0
         ENDIF
         NV = NINT(FBTB(N,J) * UC(FBAR(6,J)) *  XF)
         IF (ARC .GT. 0 .AND. ZNIDX .EQ. -1) THEN
            LO(ARC) = NV
            IF (ARC2 .GT. 0) LO(ARC2) = NV
         ELSE
            HI(ABS(ARC)) = NV
            IF (ARC2 .GT. 0) HI(ABS(ARC2)) = NV
         ENDIF
 100  CONTINUE
 99   RETURN
      END
*=====================================================================
* Name:        fb_hed
* Purpose:     Read header of the flow bound file.
*=====================================================================
      SUBROUTINE FB_HED(NDNAM, LDND, II, JJ, ARTYP, LDARC,
     &                  NFBAR, FBAR, LDFBAR, FBUNIT,
     &                  ENDFIL,
     &                  PTDWAR, LDPTDW, NDDWAR, LDDWAR, FILNAM, IU,
     &                  CTERM, COLSTR, LDCOL, ERR)
      IMPLICIT   NONE
      INTEGER    LDND, LDARC, II(LDARC), JJ(LDARC), ARTYP(LDARC)
      INTEGER    NFBAR, LDFBAR, FBAR(0:7, LDFBAR), FBUNIT, IU
      INTEGER    LDPTDW, PTDWAR(LDPTDW, 2), LDDWAR, NDDWAR(LDDWAR)
      CHARACTER  NDNAM(LDND)*(*), FILNAM*(*)
      LOGICAL    ENDFIL
      INTEGER    LDCOL
      CHARACTER  CTERM*(*), COLSTR(LDCOL)*(*)
*
      INTEGER    ND1, ND2, ND, FBIDX, TYP, ZONE, ARC, ARC1, ISGN
      INTEGER    I, J, K, L, N, I1, I2
      LOGICAL    ERR, STRM, ARFIND
      INTEGER    LDFB
      PARAMETER  (LDFB = 100)
      INTEGER    NFB, FBDAT(3, LDFB)
*
      ENDFIL = .TRUE.
*
*------------Flow units
*
      READ (IU, *, END = 99) FBUNIT
*
*------------Read upstream and downstream nodal names and zone index.
*
      DO I = 1, 3
         READ (IU, '(A)', END = 99) CTERM
         CALL STR_DIVD(CTERM, L, COLSTR, LDCOL, 0, ' ,')
         IF (L .GT. LDFB) THEN
            PRINT *, '***ERROR*** ARRAY SIZE IS NOT BIG ENOUGH.'
            PRINT *, ' CHANGE LDFB IN FB_HED AT LEAST ', L
            STOP !CALL EXIT
         ENDIF
         IF (I .EQ. 3) THEN
            DO J = 2, L
               READ (COLSTR(J), '(I2)') FBDAT(I, J-1)
            ENDDO
         ELSE
```

```
                                  NFB = L
                                DO J = 1, L
              CALL NAMNUM(LDND, NDNAM, COLSTR(J), ND, 0, ERR)
                                 IF (ERR) THEN
                       WRITE(*, 901) COLSTR(J), FILNAM
 901            FORMAT( '***ERROR***NODAL NAME: ', A12,
     &                  ' IN THE FILE: ', A)
           PRINT *, ' NOT FOUND IN THE NETWORK CONFIGURATION.'
                             STOP !CALL EXIT
                                   ELSE
                            FBDAT(I, J) = ND
                                  ENDIF
                                 ENDDO
                                ENDIF
      ENDDO
      IF (NFBAR .EQ. 0) THEN
*
*--------seasonal data
*
                              NFBAR = NFB
                             DO J = 1, NFB
                               DO I = 1, 3
                          FBAR(I, J) = FBDAT(I,J)
                                 ENDDO
                           FBAR(6, J) = FBUNIT
                              FBAR(7, J) = 0
                                ENDDO
      ELSE
*
*--------Time-dependent data
*
                                  N = 0
                            DO 10 K = 1, NFB
                              DO J = 1, NFBAR
                 IF (FBAR(1,J) .EQ. FBDAT(1, K) .AND.
     &           FBAR(2,J) .EQ. FBDAT(2, K) .AND.
     &           FBAR(3,J) .EQ. FBDAT(3, K)) THEN
                           FBAR(6, J) = FBUNIT
                             FBAR(7, J) = K
                                GOTO 10
                                  ENDIF
                                 ENDDO
                               N = N + 1
                             J = NFBAR + N
                               DO I = 1, 3
                          FBAR(I,J) = FBDAT(I, K)
                                 ENDDO
                           FBAR(6, J) = FBUNIT
                             FBAR(7, J) = K
 10     CONTINUE
*
*-----------total number of seasonal & time-dependent flow bound arcs
*
                          NFBAR = NFBAR + N
      ENDIF
*
*---------------------Find corresponding arc number.
*
      DO 50 J = 1, NFBAR
                            ND1 = FBAR(1, J)
                            ND2 = FBAR(2, J)
                           FBIDX = FBAR(3, J)
                          I1 = PTDWAR(ND1, 1)
                          I2 = PTDWAR(ND1, 2)
                           ARFIND = .FALSE.
                                  K = 3
                            DO 20 I = I1, I2
                              ARC = NDDWAR(I)
*
*--------------------Arc type
*
                         TYP = ARTYP(ABS(ARC))
                   WRITE(CTERM, '(I4)') ARTYP(ABS(ARC))
                    IF (CTERM(2:2) .NE. '1') THEN
                   PRINT *, '** ERROR ** INVALID ARC TYPE'
                                 GOTO 99
                                  ENDIF
                    IF (CTERM(3:3) .EQ. '1') THEN
                              STRM = .TRUE.
                                  ELSE
                              STRM = .FALSE.
                                  ENDIF
                     READ (CTERM(4:4), '(I1)') ZONE
                        ZONE = ISGN(TYP) * ZONE
*
*-------------------Downstream node
*
                              ARC1 = ARC
                            IF (STRM) THEN
                          ARC = ARC + ISGN(ARC)
                                  ENDIF
                         IF (ARC .GT. 0) THEN
                              ND = JJ(ARC)
                                  ELSE
```

```
                                          ND = II(-ARC)
                                             ENDIF
*
*-------------------check the current downstream node
*
                                      IF (ND .EQ. ND2) THEN
                              IF ( (ABS(ZONE)+1) .EQ. IABS(FBIDX) .AND.
     &                ZONE * FBIDX .GE. 0) THEN
                                       IF (ARFIND) THEN
                                             K = K + 1
                                        FBAR(K, J) = ARC1
                                             ELSE
                                        FBAR(0, J) = ARC1
                                         ARFIND = .TRUE.
                                            ENDIF
                                    IF (FBIDX .LT. 0) THEN
                                        FBIDX = FBIDX - 1
                                             ELSE
                                        FBIDX = FBIDX + 1
                                            ENDIF
                                              ENDIF
                                            ENDIF
 20      CONTINUE
                                   IF (.NOT. ARFIND) THEN
                                       PRINT *, CHAR(7)
                                     WRITE (*, 810) FILNAM
 810        FORMAT('***ERROR*** THE FILE NAME IS ', A)
                           WRITE (*, 811) NDNAM(ND1), NDNAM(ND2), FBIDX
 811        FORMAT(' THERE DOES NOT AN ARC FROM ', A,
     &          'TO ', A, ' WITH ZONE INDEX = ', I3)
                                         ERR = .TRUE.
                                           ENDIF
 50   CONTINUE
      ENDFIL = .FALSE.
 99   RETURN
      END
*=====================================================================
* Name:         rc_dat
* Purpose:      Get seasonal rule curve.
*=====================================================================
      SUBROUTINE RC_DAT(NNODS, NDNAM, LDND,
     &                  NRCND, RCND, LDRC, NPER, RCTB, LDRCTB, RCUNIT,
     &                  RCFLAG, FILNAM, NTLS, IN, OU, PN,
     &                  UNITNM, LDUNIT, CTERM, COLSTR, LDCOL)
      IMPLICIT   NONE
      INTEGER    NNODS, LDND, LDUNIT
      CHARACTER*(*) NDNAM(LDND), FILNAM, UNITNM(0:LDUNIT)*(*)
      INTEGER    NRCND, LDRC, RCND(LDRC, 3), NPER, LDRCTB, RCUNIT
      REAL       RCTB(0:LDRCTB, LDRC)
      INTEGER    NTLS, IN, OU, PN
      LOGICAL    RCFLAG
      INTEGER    LDCOL
      CHARACTER  CTERM*(*), COLSTR(LDCOL)*(*)
*
      INTEGER    I, NREC, IU
      LOGICAL    FLAG, ENDFIL
*
*--------------Open data file and skip title lines
*
      IF (FILNAM .NE. ' ') THEN
                                       FLAG = .TRUE.
                                           IU = 9
                  CALL IO_OPFIL(IU, 1, FILNAM, 'ENTER RC FILE: ')
                     DO I = 1, NTLS        !SKIP NTLS TITLE LINES
                                     READ (IU, *, END = 99)
                                            ENDDO
      ELSE
                                      FLAG = .FALSE.
                                          IU = IN
                  CALL PNCK(PN, IU, ENDFIL, CTERM, COLSTR, LDCOL)
                                    IF (ENDFIL) GOTO 99
      ENDIF
*
*--------------Read units, nonal names, and data from a file and
*              print these information into general output file.
*
      CALL DATTB(NNODS, NDNAM, LDND, NRCND, RCND, LDRC, RCUNIT,
     &           NREC, RCTB, LDRCTB, RCFLAG,
     &           UNITNM, LDUNIT, FILNAM, IU, OU, PN, ENDFIL,
     &           'rule-curve elevations',
     &           CTERM, COLSTR, LDCOL)
      IF (NREC .LT. NPER) THEN
                                 WRITE (OU, 805) NPER, NREC
 805     FORMAT('***WARNING*** NO. OF SEASONAL RECORDS ARE LESS',
     &          ' THAN NO. OF SEASONS IN RC DATA SET.',
     &          '     NO. OF RECORDS: ', I3,
     &          '     NO. OF SEASONS: ', I3)
      ENDIF
 99   CONTINUE
      IF (FLAG) CLOSE(IU)
      RETURN
      END
*=====================================================================
* Name:         rc_arcs
```

```
* Purpose:    Read current target water demand and create
*             corresponding TWS arcs.
*=========================================================================
      SUBROUTINE RC_ARC(NDNAM, NDSEQ, LDND, HI, LDARC,
     &                  PTRE, RC, LDRES, REAR, REZN, LDREAR,
     &                  NRCND, RCND, LDRC, RCTB, LDRCTB, RCFIL,
     &                  MTH, XF, IU, CTERM, COLSTR, LDCOL)
      IMPLICIT   NONE
      INTEGER    LDND, NDSEQ(LDND), LDARC, HI(LDARC)
      CHARACTER  NDNAM(LDND)*(*)
      INTEGER    LDRES, PTRE(LDRES), LDREAR, REAR(LDREAR)
      REAL       RC(LDRES), REZN(LDREAR)
      INTEGER    NRCND, LDRC, RCND(LDRC, 3), LDRCTB
      REAL       RCTB(0:LDRCTB, LDRC)
      LOGICAL    RCFIL
*
      INTEGER    IU, MTH
      REAL       XF
*
      INTEGER    LDCOL
      CHARACTER  CTERM*(*), COLSTR(LDCOL)*(*)
*
      INTEGER    I, J, J1, J2, N, ND, NNDS, RES, ARC
      REAL       ZVA(3), UC(0:2), UPZN, LOZN, ZN
      LOGICAL    GETZVA, FIRSTL, FIRSTU, TDFLAG
*
*
      UC(0) = 1.0
      UC(1) = 1.0 / 12.0
      UC(2) = 1.0 / 304.8
*
*---------------Read current rule curve elevation for selected reservoirs
*
      TDFLAG = .FALSE.
      IF (RCFIL) THEN
         READ (IU, '(A)', END = 50) CTERM
         CALL  STR_DIVD(CTERM, NNDS, COLSTR, LDCOL, 0, ' ,')
         TDFLAG = .TRUE.
      ENDIF
 50   CONTINUE
*
*--------------Change the rule curve elevation.
*
      DO I = 1, NRCND
         ND = RCND(I, 1)
         IF (RCND(I, 3) .EQ. 0) THEN
            N = MTH
         ELSE
            N = 0
            IF (TDFLAG) THEN
               IF (RCND(I, 3) .LT. 0) THEN
                  READ(COLSTR(NNDS), '(F15.0)') RCTB(0, I)
               ELSE
                  J = RCND(I, 3) + 1
                  READ (COLSTR(J), '(F15.0)') RCTB(0, I)
               ENDIF
            ELSE
               RCTB(0, I) = 0.0
            ENDIF
         ENDIF
         ZVA(1) = RCTB(N, I) * UC(RCND(I, 2))
         RES = NDSEQ(ND)
         IF (.NOT. GETZVA(ND, 1, ZVA, 26)) THEN
            PRINT *, CHAR(7)
            PRINT *, '***ERROR*** TRANSFORM Z-V-A:'
            PRINT *, '         POND = ', NDNAM(ND)
            PRINT *, '           RC = ', ZVA(1)
         ENDIF
         J1 = PTRE(RES)
         J2 = PTRE(RES+1) - 1
         FIRSTL = .TRUE.
         FIRSTU = .TRUE.
*
         UPZN = ZVA(2)
         LOZN = ZVA(2)
         DO J = J1, J2
            ARC = REAR(J)
*
            ZN =  REZN(J)
            IF (ARC .LT. 0) THEN
               HI(-ARC) = NINT((LOZN - ZN)*XF)
               IF (HI(-ARC) .LT. 0) THEN
                  HI(-ARC) = 0
               ELSE
                  LOZN = ZN
               ENDIF
            ELSE IF (ARC .GT. 0) THEN
               HI(ARC) = NINT((ZN - UPZN)*XF)
               IF (HI(ARC) .LT. 0) THEN
                  HI(ARC) = 0
               ELSE
                  UPZN = ZN
               ENDIF
            ENDIF
```

```
         ENDDO
         RC(RES) = ZVA(2)
      ENDDO
 99   RETURN
      END
*=====================================================================
* Name:        savbud
* Purpose:     Save system water budget into file.
* Author:      Xiaodong Jian
* Date:        04/02/97
*=====================================================================
      SUBROUTINE SAVBUD(NNDS, NDNAM, NDTYP, NODBUD, PTDWAR, LDND,
     &                  II, JJ, ARCBUD, NDDWAR, LDARC,
     &                  NOP, NSPS, XF, XP, OU,
     &                  CNDBT, CARBT, CSNDBT, LDSND, CSARBT, LDSAR)
      IMPLICIT   NONE
      INTEGER    LDND, LDARC, NNDS
      INTEGER    NDTYP(LDND), NODBUD(LDND, 0:10), PTDWAR(LDND, 2)
      CHARACTER  NDNAM(LDND)*(*)
      INTEGER    II(LDARC), JJ(LDARC), NDDWAR(LDARC), ARCBUD(LDARC, 0:6)
      INTEGER    NOP, NSPS, XP, OU
      REAL       XF
*
      INTEGER    LDSND, LDSAR
      REAL       CNDBT(0:10), CARBT(6)
      REAL       CSNDBT(LDSND, 0:10), CSARBT(LDSAR, 0:6)
*
      REAL       NDBT(0:10), ARBT(0:6), ALLBUD(8), ERRBD, BUD(6)
      INTEGER    I, J, K, L, N, LIM1, LIM2, ARC, NARC, OJ
      CHARACTER  FMT*60, FMT1*60, FMT2*50
      LOGICAL    SWBFLG
*
      SWBFLG = .FALSE.
      FMT1 = '(10x, a, t35, 3f15.0)'
      FMT2 = '(10x, 69(''-''), /, t35, a30, f15.0)'
      IF (XP .GT. 0) THEN
         WRITE (FMT1(20:20), '(i1)') XP - 1
         WRITE (FMT2(33:33), '(i1)') XP - 1
      ENDIF
*
*            Initialize arrays for output budget
*
      DO I = 0, 10
         NDBT(I) = 0.0
         IF (NOP .EQ. 1) THEN
            CNDBT(I) = 0.0
         ENDIF
      ENDDO
      DO I = 1, 6
         ARBT(I) = 0.0
         IF (NOP .EQ. 1) THEN
            CARBT(I) = 0.0
         ENDIF
      ENDDO
*
*
      IF (NOP .EQ. 1) THEN
         DO N = 1, LDSND
            DO J = 0, 10
               CSNDBT(N, J) = 0.0
            ENDDO
         ENDDO
         DO N = 1, 500
            DO J = 0, 6
              CSARBT(N, J) = 0.0
            ENDDO
         ENDDO
      ENDIF
*
*----------------Water budget of current time step
*
*
*------------Calculate the system water budget for current time step
*
*           1. Nodal budget for current time step
*
      DO N = 1, NNDS
         DO I = 0, 8
            NDBT(I) = NDBT(I) + NODBUD(N,I)/XF
*------
            IF (NOP .EQ. 1 .AND. I .EQ. 0) THEN
               CSNDBT(N, I) = NODBUD(N,I)/XF
            ELSE IF (I .GE. 1 .AND. I .LE. 7) THEN
               CSNDBT(N, I) = CSNDBT(N,I) + NODBUD(N,I)/XF
            ELSE IF (I .EQ. 8 .AND. NOP .EQ. NSPS) THEN
               CSNDBT(N, I) = NODBUD(N,I)/XF
            ENDIF
*-------
         ENDDO
      ENDDO
*
*           2. Arc budget
*
      DO 100 N = 1, NNDS
```

```
         LIM1 = PTDWAR(N, 1)
         LIM2 = PTDWAR(N, 2)
         DO 50 K = LIM1, LIM2
            ARC = NDDWAR(K)
            I = IABS(ARC)
            DO L = 1, 6
               IF (L .GE. 1 .AND. L .LE. 5) THEN
                  ARBT(L) = ARBT(L) + ARCBUD(IABS(ARC), L)/ XF
               ENDIF
*------------
               IF (NOP .EQ. 1 .AND. L .EQ. 2 ) THEN !INITIAL STORAGE
                  CSARBT(I, L) = ARCBUD(I, L)/ XF
               ELSE IF (NOP .EQ. NSPS .AND. L .EQ. 5) THEN !FINAL STORAGE
                  CSARBT(I, L) = ARCBUD(I, L)/XF
               ELSE
                  CSARBT(I, L) = CSARBT(I, L) + ARCBUD(I, L)/XF
               ENDIF
*------------
            ENDDO
*
*-------------Outflow from the system
*
 9          IF (ARC .GT. 0) THEN
               J = JJ(ARC)
            ELSE
               J = II(-ARC)
            ENDIF
            IF (NDNAM(J) .EQ. 'SKSC') THEN
               ARBT(6) = ARBT(6) + ARCBUD(IABS(ARC), 6)/ XF
            ENDIF
 50      CONTINUE
 100  CONTINUE
*
*           Save system water budgets of current time step into the file
*
      IF (SWBFLG) THEN   !SWBFLG -- SYSTEM WATER BUDEGT FLAG.
         WRITE(OU, 980) NOP
 980     FORMAT (/,'System water budgets for time step:', I3,
     &          /,'=====================================')
         WRITE (OU, 955) 'Budget', 'Pond', 'Canal', 'Total'
         WRITE (OU, 955) '------', '----', '-----', '-----'
         ALLBUD(1) = NDBT(0)+ARBT(2)
         ALLBUD(2) = NDBT(2)
         ALLBUD(3) = NDBT(4)
         ALLBUD(4) = NDBT(3)+ARBT(4)
         ALLBUD(5) = NDBT(5)+ARBT(3)
         ALLBUD(6) = NDBT(6)
         ALLBUD(7) = ARBT(6)
         ALLBUD(8) = NDBT(8)+ARBT(5)
         WRITE(OU, FMT1) 'Initial storage:',
     &                NDBT(0), ARBT(2), ALLBUD(1)
         WRITE(OU, FMT1) 'Total net inflow:',
     &                NDBT(2), 0.0,     ALLBUD(2)
         WRITE(OU, FMT1) 'Runoff:',
     &                NDBT(4), 0.0,     ALLBUD(3)
         WRITE(OU, FMT1) 'Evaporation:',
     &                NDBT(3), ARBT(4), ALLBUD(4)
         WRITE(OU, FMT1) 'Groundwater:',
     &                NDBT(5), ARBT(3), ALLBUD(5)
         IF (NDBT(6) .GT. 0) THEN
            WRITE(OU, FMT1) 'Water Withdraw:',
     &                NDBT(6),0.0,      ALLBUD(6)
         ENDIF
         WRITE(OU, FMT1) 'Outflow:',
     &                0.0,     ARBT(6), ALLBUD(7)
         WRITE(OU, FMT1) 'Final storage:',
     &                NDBT(8), ARBT(5), ALLBUD(8)
         ERRBD = ALLBUD(1) + ALLBUD(2) + ALLBUD(3) - ALLBUD(4)
     &         - ALLBUD(5) - ALLBUD(6) - ALLBUD(7) - ALLBUD(8)
         WRITE (OU, FMT2) 'In - Out = ', ERRBD
      ENDIF
*
*           Calculate the cumulative water budgets
*
      IF (NOP .EQ. 1) THEN
         CNDBT(0) = NDBT(0)
         CARBT(2) = ARBT(2)
      ENDIF
      DO L = 2, 6
         CNDBT(L) = CNDBT(L) + NDBT(L)
      ENDDO
*
*-----------Final pond storage
*
      CNDBT(8) = NDBT(8)
*
*-----------Channel initial storage, loss, and final storage
*
      IF (NOP .EQ. 1) THEN             !INITIAL CANAL STORAGE
         CARBT(2) = ARBT(2)
      ENDIF
      DO L = 3, 4
         CARBT(L) = CARBT(L) + ARBT(L) !SEEPAGE AND EVAPORATE
      ENDDO
```

```
      CARBT(6) = CARBT(6) + ARBT(6)      !OUTFLOW
      CARBT(5) = ARBT(5)                 !FINAL CANAL STORAGE.
*
*             Save water budgets
*
      IF (NOP .EQ. NSPS) THEN
*
*------------1. Save cumulative water budget of a single node
*
         WRITE (OU, 990)
         FMT = '(4X, a12, t16, i3, t21, 9f10.0)'
         WRITE (FMT(30:30), '(i1)') XP-1
         DO N = 1, NNDS
            WRITE (OU, FMT) NDNAM(N), NDTYP(N), (CSNDBT(N,I), I = 0,8)
         ENDDO
*
*------------2. Save cumulative water budget of a single canal
*
         WRITE (OU, 995)
         FMT = '(i4, 1X, 2A12, T30, 6F10.0)'
         WRITE(FMT(26:26), '(i1)') XP - 1
         NARC = 0
         DO 200 N = 1, NNDS
            LIM1 = PTDWAR(N, 1)
            LIM2 = PTDWAR(N, 2)
            OJ = 0
            DO 150 K = LIM1, LIM2
               ARC = NDDWAR(K)
               IF (ARC .GT. 0) THEN
                  I = II(ARC)
               ELSE
                  I = JJ(-ARC)
               ENDIF
               J = ARCBUD(IABS(ARC), 0)
*
*              New Canal reach:
*
               IF (J .NE. OJ) THEN
                  IF (K .NE. LIM1) THEN
                     NARC = NARC + 1
                     WRITE(OU, FMT) NARC, NDNAM(I), NDNAM(OJ),
     &                              (BUD(L), L = 1, 6)
                  ENDIF
                  OJ = J
                  DO L = 1, 6
                     BUD(L) = 0
                  ENDDO
               ENDIF
*
*                add up the water budget for current canal reach
*
               DO L = 1, 6
                  BUD(L) = BUD(L) + CSARBT(IABS(ARC),L)
               ENDDO
               IF (K .EQ. LIM2) THEN
                  NARC = NARC + 1
                  WRITE (OU, FMT) NARC, NDNAM(I), NDNAM(OJ),
     &                            (BUD(L), L = 1, 6)
               ENDIF
 150        CONTINUE
 200     CONTINUE
*
*------------3. Save the final system water budget to the file
*
         WRITE (OU, 950)
 950     FORMAT(/, 'System water budgets',
     &          /, '====================')
*
         ALLBUD(1) = CNDBT(0)+CARBT(2)
         ALLBUD(2) = CNDBT(2)
         ALLBUD(3) = CNDBT(4)
         ALLBUD(4) = CNDBT(3)+CARBT(4)
         ALLBUD(5) = CNDBT(5)+CARBT(3)
         ALLBUD(6) = CNDBT(6)
         ALLBUD(7) = CARBT(6)
         ALLBUD(8) = CNDBT(8)+CARBT(5)
         WRITE (OU, 955) 'Budget', 'Pond', 'Canal', 'Total'
         WRITE (OU, 955) '------', '----', '-----', '-----'
         WRITE(OU, FMT1) 'Initial storage:',
     &                   CNDBT(0), CARBT(2), ALLBUD(1)
         WRITE(OU, FMT1) 'Total net inflow:',
     &                   CNDBT(2), 0.0,      ALLBUD(2)
         WRITE(OU, FMT1) 'Runoff:',
     &                   CNDBT(4), 0.0,      ALLBUD(3)
         WRITE(OU, FMT1) 'Evaporation:',
     &                   CNDBT(3), CARBT(4), ALLBUD(4)
         WRITE(OU, FMT1) 'Ground-water seepage:',
     &                   CNDBT(5), CARBT(3), ALLBUD(5)
         IF (CNDBT(6) .GT. 0) THEN
            WRITE(OU, FMT1) 'Water withdrawal:',
     &                   CNDBT(6),0.0,       ALLBUD(6)
         ENDIF
         WRITE(OU, FMT1) 'Outflow:',
     &                   0.0,      CARBT(6), ALLBUD(7)
```

```
      WRITE(OU, FMT1) 'Final storage:',
    &                  CNDBT(8), CARBT(5), ALLBUD(8)
      ERRBD = ALLBUD(1) + ALLBUD(2) + ALLBUD(3) - ALLBUD(4)
    &        - ALLBUD(5) - ALLBUD(6) - ALLBUD(7) - ALLBUD(8)
      WRITE (OU, FMT2) 'In - Out = ', ERRBD
    ENDIF
    RETURN
955 FORMAT(10X, A, T35, 3A15)
990 FORMAT(/,'Nodal water budgets for whole simulation',
    &     /,'========================================',
    &  /,                T21, '   Initial  Upstream Local net',
    &'    Evap-                            Downstream    Final',
    & /,                  T21, '   storage    inflow    inflow',
    &'   ration  Rainfall   Seepage Withdrawal  release   storage',
    & /,'    Node',T16, 'Node', T21, '   (acre-    (acre-    (acre-',
    &'   (acre-    (acre-    (acre-    (acre-    (acre-    (acre-',
    & /,'    name',T16, 'type', T21, '     feet)     feet)     feet)',
    &'     feet)     feet)     feet)     feet)     feet)     feet)',
    & /,'    ----',T16, '----', T21, ' --------- --------- ---------',
    &' --------- --------- --------- --------- --------- ---------')
995 FORMAT(/,'Canal water budgets for whole simulation',
    &     /,'========================================',
    & /,T30,'                                 Surface',
    & /,T30,'             Initial    Canal    evapo-     Final',
    & /,T30,'   Inflow   storage   seepage    ration   storage',
    &       '   Outflow',
    & /,T30,'    (acre-    (acre-    (acre-    (acre-    (acre-',
    &       '    (acre-',
    & /,' No. From', T18, 'To',
    &   T30,'     feet)     feet)     feet)     feet)     feet)',
    &       '     feet)',
    & /,' --- ----', T18, '----',
    &   T30,' --------- --------- --------- --------- ---------',
    &       ' ---------')
    END
*======================================================================
* Name:       namnum
* Purpose:    Search the node number with the node name, or vice versa.
* Author:     Xiaodong Jian
* Date:       9/13
*======================================================================
    SUBROUTINE NAMNUM(NNOD, NDNAMS, NDNAM, NDNUM, IW, ERR)
    IMPLICIT   NONE
    INTEGER    NNOD, NDNUM, IW
    CHARACTER  NDNAMS(NNOD)*(*), NDNAM*(*)
    LOGICAL    ERR, CN
*
    INTEGER    I
*
    ERR = .FALSE.
    IF (IW .EQ. 0) THEN
       DO I = 0, NNOD
          IF (CN(NDNAMS(I),NDNAM,1)) THEN
             NDNUM = I
             GOTO 99
          ENDIF
       ENDDO
    ELSE
       IF (NDNUM .LE. NNOD .AND. NDNUM .GE. 1) THEN
          NDNAM = NDNAMS(NDNUM)
          GOTO 99
       ENDIF
    ENDIF
    ERR = .TRUE.
 99 RETURN
    END
```



☆U.S. GOVERNMENT PRINTING OFFICE: 1998-657-084