MODIFICATIONS TO THE MODULAR THREE-DIMENSIONAL FINITE-DIFFERENCE GROUND-WATER FLOW MODEL USED FOR THE COLUMBIA PLATEAU REGIONAL AQUIFER-SYSTEM ANALYSIS, WASHINGTON, OREGON, AND IDAHO

By A. J. Hansen, Jr.

A Contribution of the
Regional Aquifer-System
Analysis Program

U.S. GEOLOGICAL SURVEY

Open-File Report 91-532

Tacoma, Washington
1993

U.S. DEPARTMENT OF THE INTERIOR

BRUCE BABBITT, Secretary

U.S. GEOLOGICAL SURVEY

Robert M. Hirsch, Acting Director

For additional information
write to:

District Chief
U.S. Geological Survey
1201 Pacific Avenue - Suite 600
Tacoma, Washington 98402

Copies of this report can be
purchased from:

U.S. Geological Survey
Earth Science Information Center
Open-File Reports Section
Box 25286, MS 517
Denver Federal Center
Denver, CO 80225

CONTENTS

ILLUSTRATIONS

CONVERSION FACTORS AND VERTICAL DATUM

Multiply	By	To obtain
foot (ft)	0.3048	meter
mile (mi)	1.609	kilometer
square mile (mi^2)	2.590	square kilometer
cubic foot per second (ft^3/s)	0.0283	cubic meter per second

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

MODIFICATIONS TO THE MODULAR THREE-DIMENSIONAL FINITE-DIFFERENCE GROUND-WATER FLOW MODEL USED FOR THE COLUMBIA PLATEAU REGIONAL AQUIFER-SYSTEM ANALYSIS, WASHINGTON, OREGON, AND IDAHO

By A. J. Hansen, Jr.

ABSTRACT

This report documents modifications to the U.S. Geological Survey's modular three-dimensional finite-difference ground-water flow model used for a regional aquifer-system analysis of the Columbia Plateau. The changes were needed because the aquifer system includes unconformities, deep canyons, and steeply dipping structural barriers that could not be represented realistically by the existing model. The modifications permit flow from a cell in any layer to a cell in any other adjacent layer; allow a cell that has been cut completely through by a canyon to remain in the model by setting the branch conductance to zero only on specified cell walls; simulate barriers to lateral ground-water flow by reducing the branch conductance only on specified cell walls; and allow use of a convergent grid. The concepts and mathematical basis for these modifications are described. The changes in the data-input items for this modified model, compared with those of the existing modular model, are described.

INTRODUCTION

A study of the Columbia Plateau aquifer system (Vaccaro, 1986) was completed as one of 28 studies of the U.S. Geological Survey's Regional Aquifer-System Analysis (RASA) program. The Columbia Plateau aquifer system underlies about 51,000 square miles of the Columbia Plateau (fig. 1) in central and eastern Washington, north-central and eastern Oregon, and northwestern Idaho. The aquifer system consists of part of the Columbia River Basalt Group (Miocene age), the intercalated sediments collectively assigned to the Ellensburg Formation (Miocene age), and the unconsolidated sediments (Miocene to Holocene age) overlying the basalts.

As part of the study, a three-dimensional model simulating ground-water flow in the Columbia Plateau aquifer system was constructed (Hansen and others, 1993, Hansen, 1993). The U.S. Geological Survey's modular three-dimensional finite-difference ground-water flow model (McDonald and Harbaugh, 1988), hereafter called Modular Model, was selected as the model to be used in the study. Several hydrogeologic characteristics of the study area prevented use of the Modular Model without modification. The modifications were necessary in order to more realistically represent the physical system. The hydrogeologic characteristics and the reasons for the necessary modifications are:

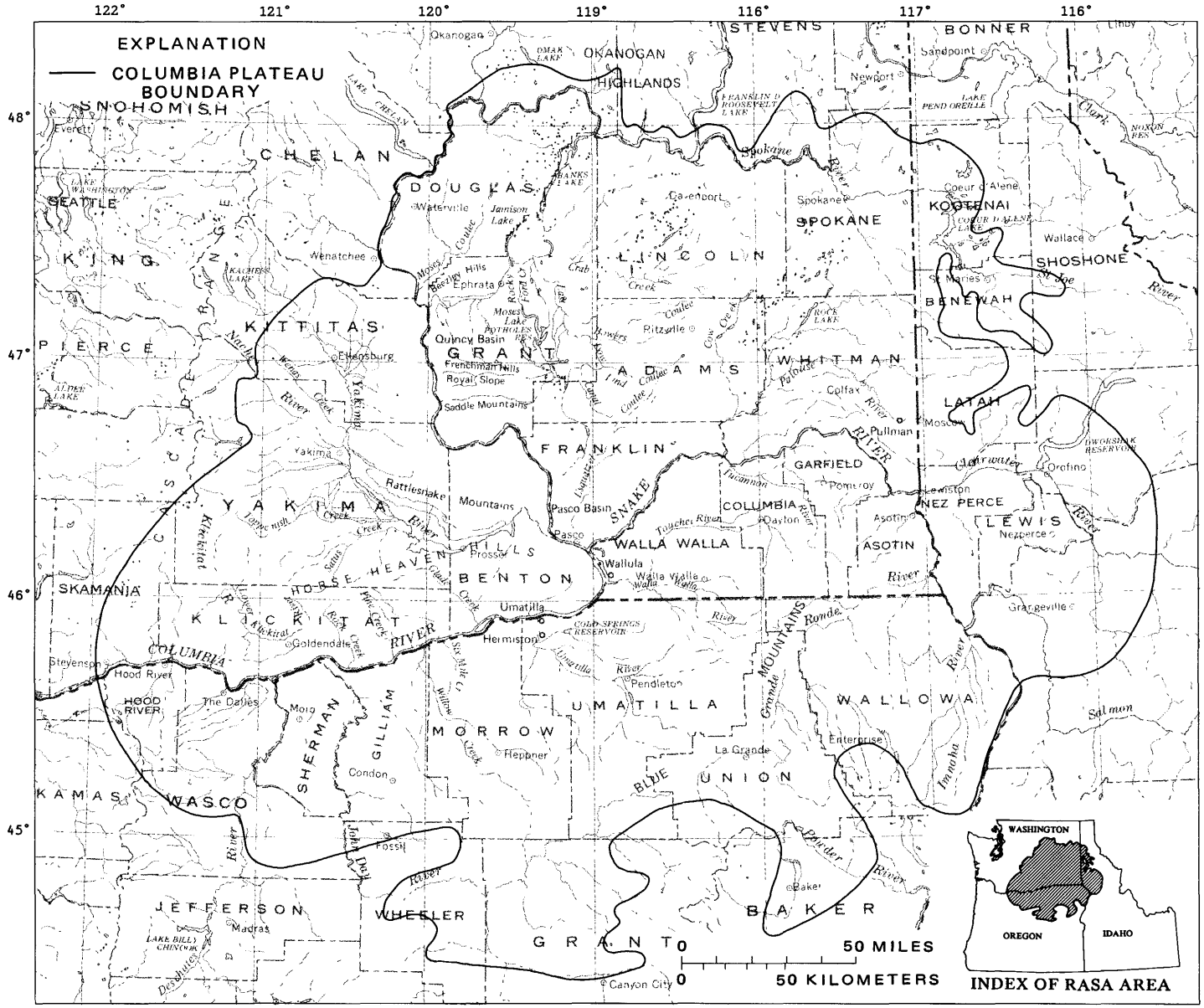
1. unconformities (areas where hydrogeologic units are absent) cannot be represented realistically using the Modular Model;

2. deep canyons that cut completely through a unit cannot be represented realistically using the Modular Model (for most spatial scales); and

3. structural features that create a narrow linear barrier to lateral ground-water flow cannot be simulated realistically using the Modular Model.

Therefore, modifications were made that permit flow from a cell in any model layer to a cell in any other adjacent layer, allow a cell that has been cut completely through by a canyon to remain in the model by setting the branch conductance to zero only on specified cell walls, and simulate barriers that impede lateral ground-water flow by reducing the branch conductance only on specified cell walls. Two new packages have been added to the Modular Model in order to incorporate the latter two modifications. Additionally, another modification made, which does not affect computational procedures or represent modifications to published approaches, allows use (a special version) of a convergent grid.

The purpose of this report is to describe these modifications to the Columbia Plateau regional aquifer-system flow model; the report is intended as a supplement to the Modular Model documentation and describes only the changes made to the Modular Model for this RASA project. Although these modifications affect many subroutines in many packages, with a few changes, the input for any other model constructed for another aquifer system could be used directly with this modified model, which then would calculate the same results as that other model. Thus, the changes described here might be used to apply the Modular Model to modeling studies of similar aquifer systems.



Base from U.S. Geological Survey
State base maps, 1:500 000

Figure 1.--Location of Columbia Plateau regional aquifer-system study.

The first part of the report describes the conceptual and mathematical basis for the major modifications to the Modular Model. The changes allow alternative conceptualizations of an aquifer system, so its flow system may be simulated using a three-dimensional ground-water flow model. The next part of the report documents the modified and new subroutines by presenting both a narrative of these subroutines and a list of the new variables in the model. The last part of the report presents the few changed input instructions, organized by package, necessary to use the model as modified for the Columbia Plateau RASA study. The report includes three appendixes; the first two present sample problems and the third lists the source code for the model.

Partial Layer

The Modular Model permits vertical flow from layer K only to layers K-1 or K+1; thus, if part of layer K is missing, as shown in figure 2, the layer still must be represented in the model. Generally, to represent this missing layer, each cell in the missing part of layer K would have a small lateral hydraulic conductivity value and a large vertical conductance value assigned to it. This would dampen interference between the real and missing parts of layer K and transmit ground water, with the least distortion, through the missing part of layer K, between cells of the actual hydrogeologic units represented by the layers above and below it.

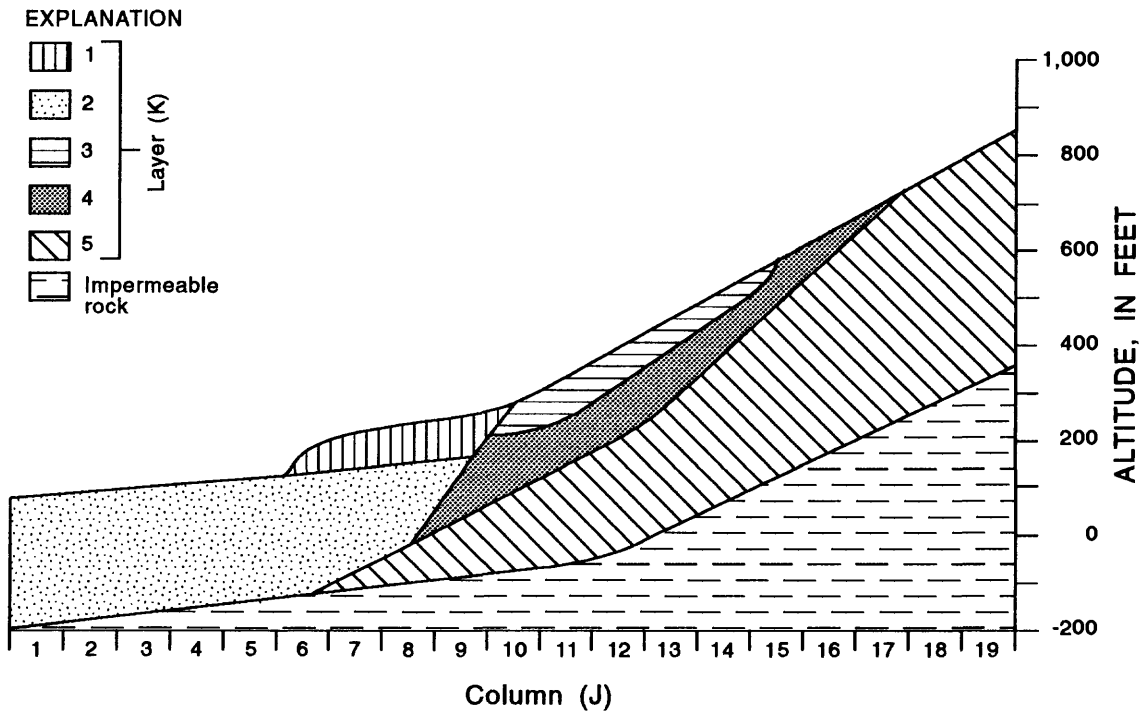


Figure 2.--Hypothetical cross-section showing layers with parts missing.

The modified model permits flow from layer K to the first active or constant-head cell in the layers above and below it, thus making all cells in the missing part of a layer inactive. Computationally, at each cell in layer K, a search is completed to check whether the cells in layers K-1 and K+1 are inactive. If they are, the search proceeds to the cell in the next layer beyond and continues until either an active or constant-head cell is found, or until the top layer for search above layer K or the bottom layer for search below it is reached. For example, in figure 2, the search downward from layer 2 will find an active cell in layer 4 in column 9, and in layer 5 in columns 7 and 8, but only inactive cells in layers 3 through 5 before column 7. Similarly, the search upward from layer 4 will find an active cell in layer 2 in column 9, and in layer 1 in column 10, but only inactive cells in layers 3 through 1 beyond column 15 (fig. 2). This requires a layer index to be advanced, layer-by-layer, to keep track of the layer search. This provision for missing layers has been added to both the block-centered-flow and the strongly implicit procedure packages of the Modular Model. The changes to the strongly implicit procedure are described later, in the section "Solvers."

Canyon Cutter

The Modular Model provides but one way to simulate the situation of a canyon cut deep enough to penetrate fully through an upper layer, so that water cannot flow in that layer from one side of the canyon to the other; each cell penetrated fully must be inactive. However, a canyon commonly is much narrower than a cell; an improved physical representation of the physical system results if each such cell has only its cell walls nearest the canyon unable to transmit water. A new package "canyon cutter" can prevent lateral flow across a specified cell wall by setting its branch conductance to zero.

Figure 3 presents three map views of a canyon. Figure 3a shows both the transmissivity and branch conductances for each cell unaffected by any canyon, and figure 3b illustrates a canyon (the solid line) extending from (3,1) to (2,2); the cell wall nearest the cutter is shown by a striped band. On that band, the branch conductance (parenthesized) of each cell wall is set to zero. Figure 3c shows the result of placing the effect of the canyon on the cell itself, by making the cell inactive, giving the same effect as setting the values in cells (2,1) and (2,2) to zero. Then, the values of five branch conductances become zero rather than only three. The latter results in a better representation of the physical system. For example, if the dot appearing in the southwest corner of (2,2) in figures 3b and 3c were the location of a well, the results of a simulation that placed the canyon's effect on the nearest cell wall (fig. 3b) would differ markedly from one that represented the canyon by making the cell inactive (fig. 3c).

Figure 4 presents three cross-section views of the canyon shown in map view in figure 3; the section extends along column 2, from row 3 to 1. As in figure 3, figure 4a represents the system without a canyon, figure 4b depicts the effect of a canyon applied to the nearest cell wall, and figure 4c shows the canyon's effect as an inactive cell. The well at the left of row 2 in figures 4b and 4c is that shown in the previous example (figs. 3b and 3c).

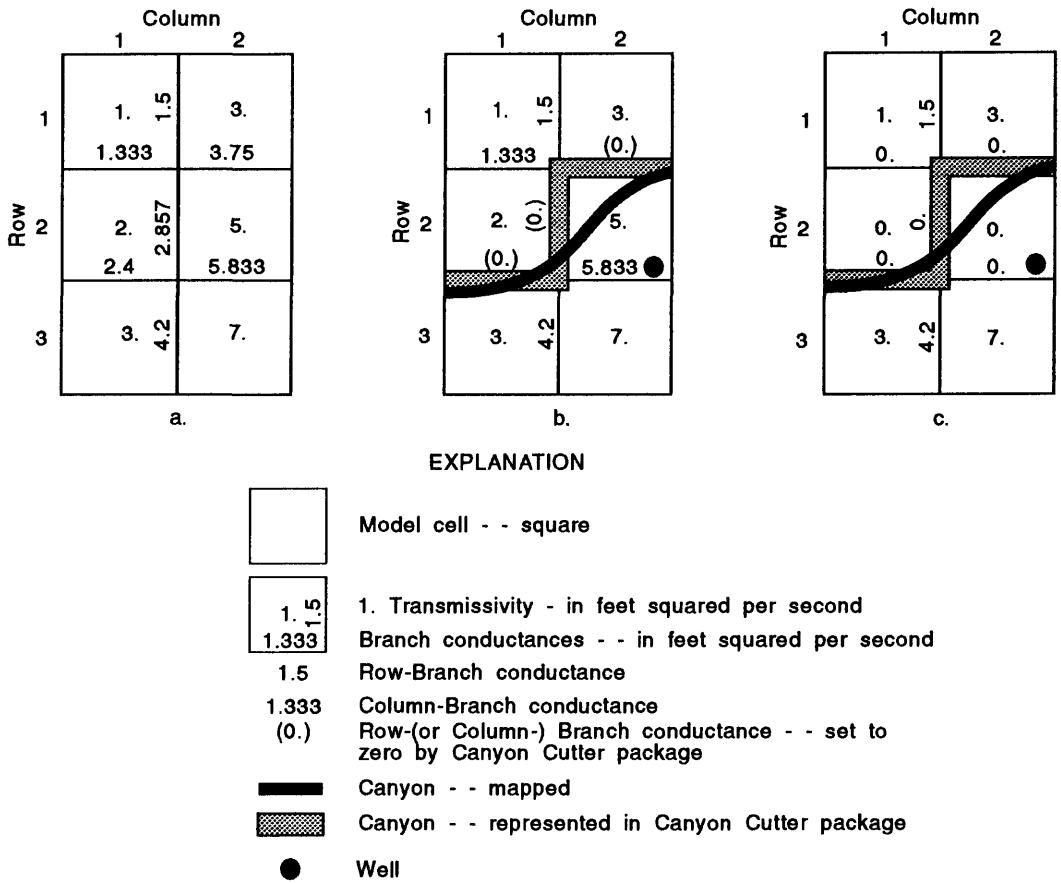


Figure 3.--Hypothetical canyon that cuts through an aquifer layer: map view of representation by Canyon Cutter and Modular Model.

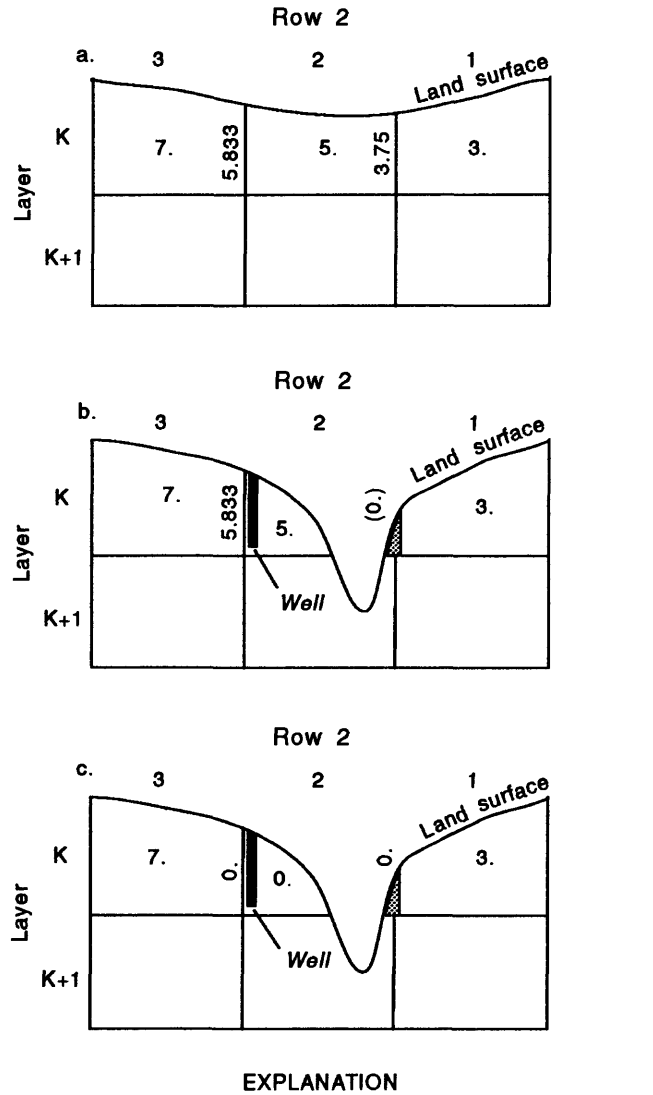


Figure 4.--Hypothetical canyon that cuts through an aquifer layer: cross-section view of representation by Canyon Cutter and Modular Model.

The equations (equations 10 and 12, McDonald and Harbaugh, 1988) that express flow through the left and rear faces of cell i,j,k (fig.4, pg.2-9, McDonald and Harbaugh, 1988) can be written:

$$q_{i,j-1/2,k} = CR_{i,j-1/2,k} (h_{i,j-1,k} - h_{i,j,k}) \quad , \tag{1}$$

$$q_{i-1/2,j,k} = CC_{i-1/2,j,k} (h_{i-1,j,k} - h_{i,j,k}) \quad , \tag{2}$$

where i is the row index,
j is the column index,
k is the layer index,
q is the volumetric fluid discharge through the face (L^3T^{-1}),
CR is the hydraulic conductance in row i and layer k between nodes i,j-1,k and i,j,k ($L^2 T^{-1}$), and
CC is the hydraulic conductance in column j and layer k between nodes i-1,j,k and i,j,k ($L^2 T^{-1}$).

The canyon cutter thus can prevent flow through either the left or rear face of this cell by setting CR in equation 1 or CC in equation 2 to zero.

This package first reads in the maximum number of canyon cutters and three flags used to identify a canyon cutter in the boundary array. The number of cutters in each layer and their addresses of cell and forward cell are next read. This package then writes a signal (the flag or identifier) into the boundary array for each branch conductance that it sets to zero. The block-centered-flow package then reads the boundary array to identify where the branch conductance has been set to zero in order to account for zero branch conductances. For example, zero conductances need to be accounted for in the cell-cancelling check (described later) in the block-centered-flow package.

Flow Barrier

In the Columbia Plateau aquifer system and similar hydrogeologic environments, lateral ground-water movement is affected by linear features such as faults, fractures, or dikes. The features generally are much thinner than the size of a model cell, commonly have a smaller hydraulic conductivity than that of the surrounding rock, and often dip steeply. Commonly, these features act as "flow barriers" that impede the lateral movement of ground water. The Modular Model can represent a cell containing such a barrier only by reducing its hydraulic conductivity (or transmissivity, if the cell's layer is either confined or partly convertible); an improved simulation of the physical system results if such a cell has the branch conductance reduced only on its cell walls nearest the barrier. In order to model the effects of these features, a new package, "flow barrier," has been added. The Columbia Plateau RASA's flow barrier package expresses a barrier's hydraulic conductivity as a fraction of that of the rock. (For the Columbia Plateau RASA study, this fraction was found to be about inversely proportional to the ratio of the head gradient between the cell centers on opposite sides of the barrier to the average regional head gradient; see Hansen and others, 1993.) Thus, the modified model can simulate a barrier by reducing the branch conductance on specified cell walls.

Three map views of a flow barrier (fig. 5) are presented and described below in order to illustrate the effects of reducing branch conductance to account for a flow barrier when using the standard Modular Model. Figure 5a presents the transmissivity and calculated branch conductances for each cell in a three-by-two model of a system without flow barriers. Figure 5b illustrates a mapped flow barrier (for example, a fault) that extends from (3,1) to (2,2); for each cell containing the barrier, the cell wall nearest the barrier is delineated. Assuming that the barrier has a hydraulic conductivity half that of the rock, each cell wall nearest the barrier in figure 5b would have a branch conductance half the value shown in figure 5a. Next, figure 5c shows the result of assigning the barrier's effect to either the hydraulic conductivity or transmissivity of the cell as would be done with the standard Modular Model. Thus, five branch conductances would be decreased, even for cells (1,1) and (3,2). The values of five branch conductances in figure 5c differ from those in figure 5a, and figure 5b shows that only three of them should be reduced. The row-branch conductance of (2,1) has the correct value; the other two branch conductances on the barrier are too large. Therefore, two branch conductances that should not be affected by the barrier would be reduced, and two other branch conductances are not reduced enough. If the computed results for cells (1,1) or (3,2) were important, as they would be if a well were in the cells' southeast or northwest corner, respectively, a comparison of the branch conductances shown on figures 5b and c indicates there generally would be a marked difference between placing the barrier's effect on the nearest cell wall and assigning it to the cell itself.

Figure 6 presents three cross-section views of the flow barrier shown in map view in figure 5; this cross section extends along column 2, from row 3 to 1. As in figure 5a, figure 6a represents the system unaffected by any flow barrier, figure 6b shows the effect of the barrier placed on the nearest cell wall, and figure 6c shows the effect of the barrier when assigning reduced transmissivity to the cells with a barrier present in it. It is seen (fig.6) that when branch conductance is reduced only at the nearest cell wall, groundwater flow is not impeded from cell (2,2) to cell (3,2). Without the flow barrier modification, flow would be impeded also between these cells.

The new flow barrier package thus allows the simulation of the effects of a barrier by altering only the cell walls nearest to it, rather than having to change either the hydraulic conductivity or the transmissivity of the cell itself. This package is similar to that of P. A. Hsieh (U.S. Geological Survey, written commun., 1986). The package first reads in the number of cell walls that will have their branch conductances reduced. The number of barriers for each layer, a master or global factor for reducing conductances for all barriers on that layer, and the cell and forward cell and factor are then read. Branch conductances, calculated by the standard method, then are reduced by the factor that is read in for each cell wall. The package is like the canyon cutter package except, rather than setting a branch conductance to zero, it multiplies the branch conductance by a specified factor. (Note, this factor also can be greater than 1.0 so that branch conductance can be increased at a specified cell wall.) The flow barrier and canyon cutter are separate packages because the canyon cutter must write a signal (an identifier) into the boundary array for each branch conductance that it sets to zero for the block-centered-flow package to read in order to account for zero conductances in the cell-cancelling check (described later).

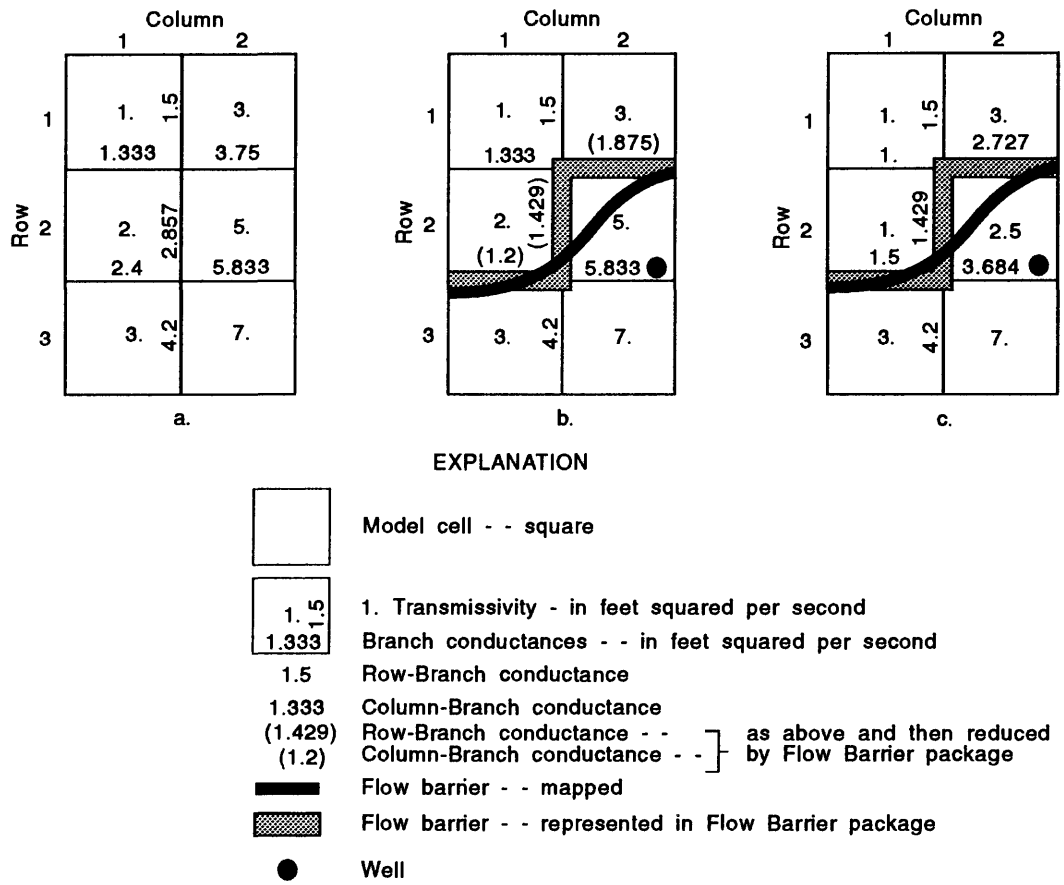


Figure 5.--Hypothetical flow barrier: map view of representation by Flow Barrier and Modular Model.

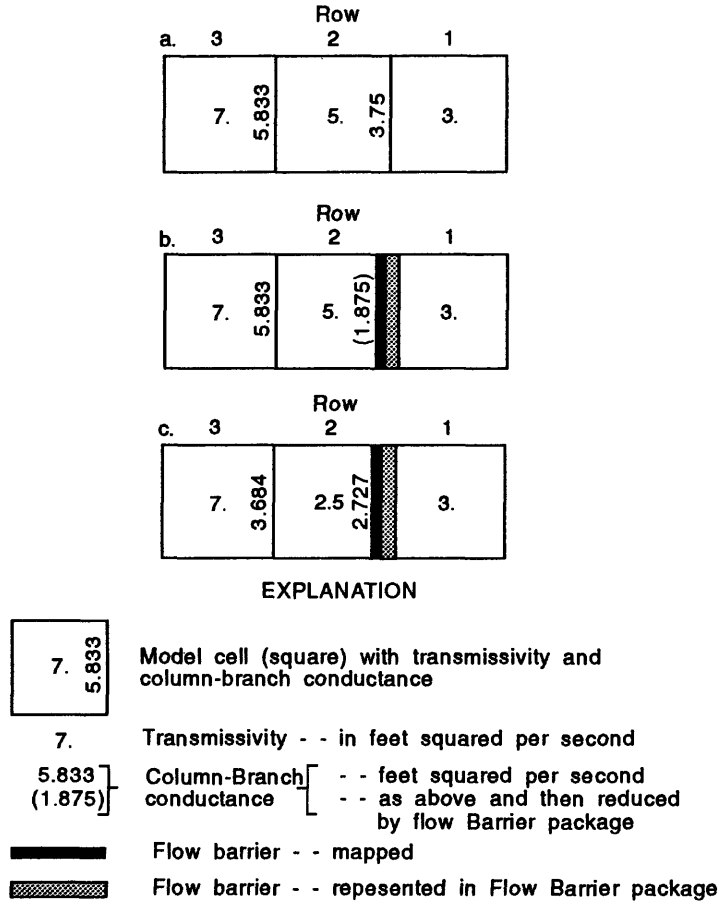


Figure 6.--Hypothetical flow barrier: cross-section view of representation by Flow Barrier and Modular Model.

Isolation of Cells

The Modular Model checks whether each cell has at least one adjacent lateral cell that is either an active or a constant-head cell; if it finds a cell that has none, it cancels this cell from the model calculations to avoid abnormal termination of the simulation. However, it was found that, for the Columbia Plateau RASA flow model, the condition "zerodivide" often occurred (and stopped the solver: strongly implicit procedure) during the early stages of the calibration process when using initial estimates of geometry and hydraulic properties. This condition occurs if two active cells become isolated laterally from the rest of their layer by inactive cells, as the diagram below shows.

I	IN	IN	IN	IN
I+1	IN	AC	AC	IN
I+2	IN	IN	IN	IN
	J	J+1	J+2	J+3

IN Inactive cell
AC Active cell

This condition sometimes occurred, even when the model was almost calibrated, whenever the standard check for no adjacent cell was used. (A further change that prevents this condition is discussed later in the final paragraph of this section.)

To alleviate this numerical problem, the block-centered-flow package was modified to permit the user to specify cancelling each cell if it has either no or only one adjacent lateral cell that is either a constant-head or an active cell. This allows the user to choose only one cell during the early part of model calibration, when numerical errors are most likely to occur because initial estimates of the hydraulic characteristics and geometry of the system (and iteration parameters) are being used. This does not imply necessarily that the initial estimates are poor. The model simply may not be able to numerically deal with large variations in hydraulic characteristics or geometry. The variable defining the requisite number of adjacent cells is constrained to be either zero or one, preventing erroneous specification of an impossible condition. (If a negative number of cells were chosen, no cells, not even one fully isolated, would be cancelled; if more than one adjacent cell were specified, every cell in the model would be cancelled.) This modification also needs to check the boundary array to find whether the canyon cutter has set a branch conductance to zero, so that lateral flow cannot occur; not only must the adjacent cell be a constant-head or an active cell, but the branch conductance to it must be able to transmit flow.

Last, the modified solver, strongly implicit procedure, checks each cell to find whether the condition "zerodivide" will occur; if it will, the solver cancels the cell, writes its address, and stops.

Budget

The Modular Model has been modified to calculate the hydrologic budget between layers and to write the boundary array (where dry cells have been set to zero) and other necessary information that could be used in post-processing programs. The flow at each constant-head, drain, river, well, and general-head-boundary cell always is printed, to a separate file. These changes affect all packages of the Modular Model, except the strongly implicit procedure package.

Convergent Grid

The Modular Model requires a grid of rectangular coordinates defined by rows and columns; it cannot account for a latitude-longitude-based grid system that converges along meridional lines. The Columbia Plateau RASA ground-water model uses such a convergent grid; it has a row-spacing of 2 minutes of longitude and a column-spacing of 2.5 minutes of latitude. Between the limits of 44° 52.5′ and 48° latitude, the maximum rate of convergence of the row-spacing per row is 0.082 percent; the column-spacing also converges, but at a much smaller rate (maximum rate of 0.0033 percent). The convergence at 46° 25′, near the center of the model, is shown in figure 7.

In the Modular Model, the arrays that describe the row-spacing and column-spacing are single-subscripted. The row-spacing is dimensioned at the maximum number of columns--the column-spacing at the maximum number of rows. Thus, in order to account for the convergent grid used in the model, the Modular Model was modified so that both the row- and column-spacings have double subscripts; both spacings are dimensioned at the maximum number of columns and of rows. This modification is not recommended for other users.

The following describes the potential errors in using the convergent grid in the Columbia Plateau RASA ground-water model. The rate of convergence of the grid system was 4.4×10^{-4} ft/ft, or about 6.7 ft per cell length of about 15,195 ft. The spacing along latitude lines also varied due to map projection errors in going from geographic to Lambert x,y coordinates. However, the maximum rate of this variation was only 5.8×10^{-5} ft/ft, or about 0.5 ft per cell width of about 8,550 ft, and will not be discussed further.

Given a row of cells (fig. 4 and fig. 25 in McDonald and Harbaugh, 1988), such that the flow $q_{i,j-1/2,k}$ into cell i,j,k from the left equals the flow $q_{i,j+1/2,k}$ out of the cell to the right. In this case, the finite-difference equation can be written using McDonald and Harbaugh's (1988) equations 32 and 41 as follows, suppressing the subscripts i and k,

$$\frac{2\Delta c TR_{j-1} TR_j}{TR_{j-1}\Delta r_j + TR_j \Delta r_{j-1}} (h_{j-1} - h_j) = \frac{2\Delta c' TR_j TR_{j+1}}{TR_j \Delta r_{j+1} + TR_{j+1}\Delta r_j} (h_{j+1} - h_j) \qquad (3)$$

where i is the row index;
j is the column index;
TR is the transmissivity in the row direction;
Δc is the column length at the left face i,j-1/2,k of cell i;
$\Delta c'$ is the column length at the right face i,j+1/2,k of cell i;
Δr is the row length; and
h is the head at the appropriate cell.

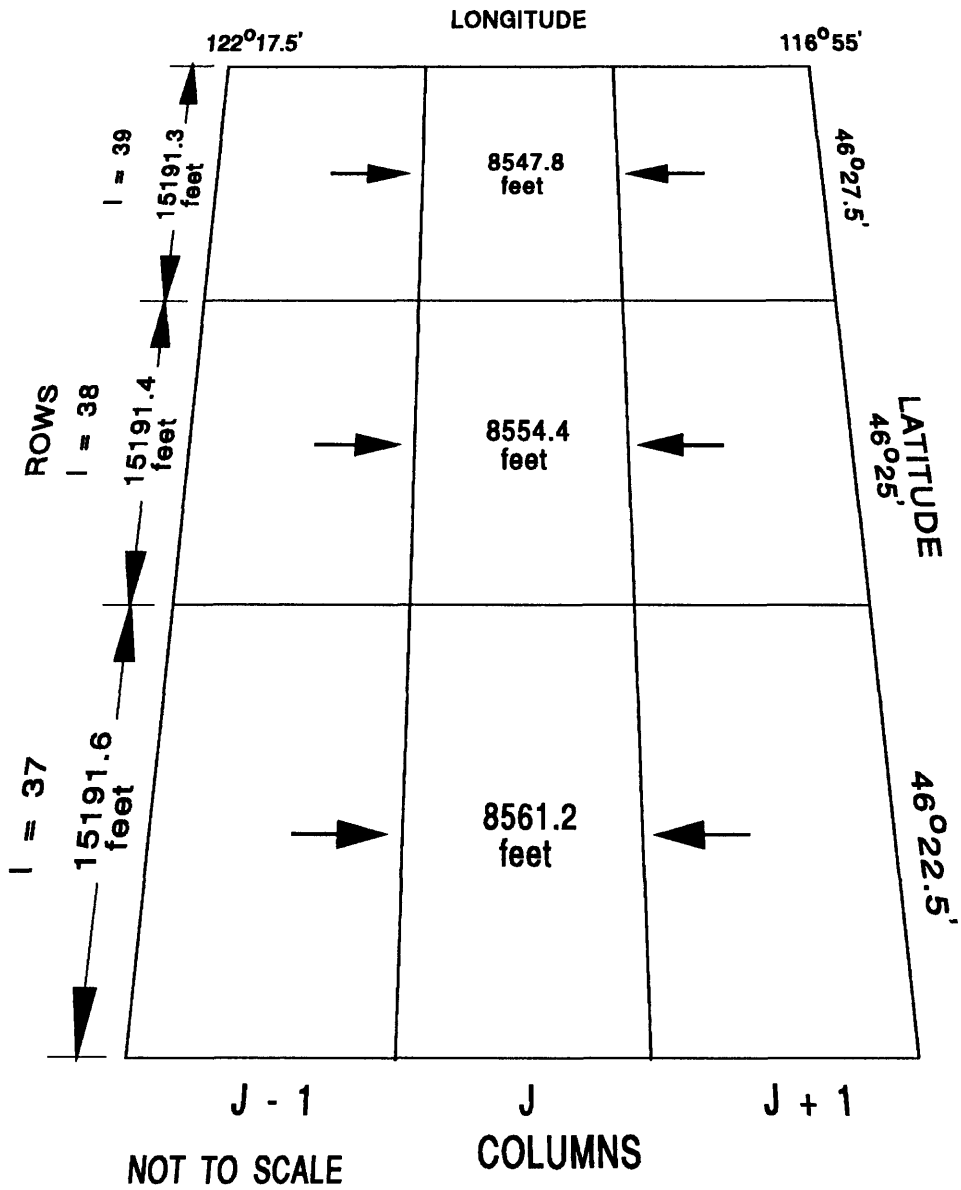


Figure 7.--Grid spacing near center of Columbia Plateau groundwater flow model, with convergent grid.

Simplifying assumptions used in writing the above equation are: (1) the row of cells lie on or nearly on a flow path, such that lateral flow into the cells along the column direction is small, (2) the vertical hydraulic conductivity and gradient both are small enough that flow through the bottom of the cells can be neglected, and (3) flow through the top of the cells (either recharge or case (2) above) is small enough that it can be neglected. These assumptions, which generally will maximize errors, will be discussed later. One then can isolate the Δc and $\Delta c'$ term on the left-hand side of equation 3, yielding,

$$\frac{\Delta c}{\Delta c'} = \frac{(h_{j+1} - h)\ TR_{j+1}\ (TR_{j-1}\Delta r_j + TR_j\Delta r_{j-1})}{(h_{j-1} - h)\ TR_{j-1}\ (TR_j\Delta r_{j+1} + TR_{j+1}\Delta r_j)} \qquad (4)$$

where $\Delta c/\Delta c'$ equals either 1.0008, or 0.9992, depending on whether the meridians converge to the right or the left. For the limiting case, where $TR_j = TR_{j-1} = TR_{j+1}$, and noting that $\Delta r_j \simeq \Delta r_{j-1} \simeq \Delta r_{j+1}$, the simple expression that the ratio of lateral gradients along the row of cells is equal to $\Delta c/\Delta c'$ is arrived at,

$$\frac{\Delta c}{\Delta c'} = \frac{(h_{j+1} - h_j)}{(h_{j-1} - h_j)} \ . \qquad (5)$$

For a non-convergent grid ($\Delta c = \Delta c'$), the gradients into and out of a cell must be equal when transmissivity is a constant. Thus, for the limiting case of equation 5, the gradient (and flux) into a cell for the Columbia Plateau would be about 0.08 percent greater or less than that under using a non-convergent grid. The total convergence from the first to the last column in the Columbia Plateau RASA model is about 500 ft over about 216 miles, or about a 6-percent total convergence or total change in Δc (potential error over such a flow path).

The 0.08-percent error can be related alternatively to a flux error for a typical cell. Defining $\Delta c_j = \Delta c' + e$, and noting that $\Delta r_j \simeq \Delta r_{j-1} \simeq \Delta r_{j+1}$, allows equation 3 to be written as follows, again suppressing the subscripts i and k,

$$q_{j+1/2} = \left\{ 2\Delta c \frac{TR_j TR_{j+1}}{(TR_j + TR_{j+1})\Delta r}(h_{j+1} - h_j) \right\} \left\{ 2e \frac{TR_j TR_{j+1}}{(TR_j + TR_{j+1})\Delta r}(h_{j+1} - h_j) \right\}. \qquad (6)$$

Let $TR_j = TR_{j+1} = 3.85 \times 10^{-2}$ ft^2/s (the calculated median transmissivity of all model cells, as reported in Hansen and others, 1993), $\Delta c = 8{,}550$ ft, $e = 6.7$ ft, and $\Delta r = 15{,}195$ ft. Consider two cases in which $\Delta h = h_{j+1} - h_j$ equals either 200 ft or 10 ft. The terms on the right-hand side of equation 6 are

$$q_{j+1/2} = 4.3333 \text{ ft}^3\text{/s} + 3.4 \times 10^{-3} \text{ ft}^3\text{/s}, \text{ for } \Delta h = 200 \text{ ft},$$
$$q_{j+1/2} = 0.2167 \text{ ft}^3\text{/s} + 1.7 \times 10^{-4} \text{ ft}^3\text{/s}, \text{ for } \Delta h = 10 \text{ ft}.$$

The e (convergent grid error) term is thus three orders of magnitude smaller than the estimated flux using a median transmissivity value under observed ranges in hydraulic gradients. The information presented above for the row of cells under the three assumptions indicates that model error would be small for the rate of convergence in the Columbia Plateau grid system.

McDonald and Harbaugh's (1988) equation 23 includes all terms for writing the finite-difference equation for a cell without the previously described assumptions. As in the case considered above, the convergence of the meridians would result in about a 0.08-percent error in some the conductances CR. It is apparent that the relative potential errors in gridded thickness values (which range from about 30 ft to more than 12,000 ft), in estimates of recharge (1 to 26 ft^3/s per cell), and in describing hydraulic head are much larger than the error due to the convergent grid. Indeed, the convergent grid errors are several orders of magnitude smaller than the other natural errors.

Basic and Block-Centered Flow Packages

In the basic package of the Modular Model, a value is read that will be placed at each inactive cell; in the modified model, the read statement has been deleted (all head values at inactive cells are set to zero) because, if a large value were read:

1. Graphics produced by selected software might be distorted, and

2. Selected programs would be affected:
   a. subroutine USCMP would scale the head at active cells too small to be discerned, and
   b. post-processors would not work.

Because of numerical instability, the hydrologic system of the RASA project could not be modeled unless a constant was added to the observed starting head at each active cell. For each layer: a constant is read and added to each active cell, the head is set to zero at each inactive cell, and the input head is retained at each constant-head cell.

In the block-centered flow package of the Modular Model, when a cell is cancelled, the head at the cell is set to a large value; in the modified model, it is set to zero because of the two reasons listed above in the discussion of the basic package.

Solvers

The slice-successive-overrelaxation procedure has been eliminated from the modified model. In the strongly implicit procedure package, a constant to test for zerodivide or underflow is read; the head change, and the address thereof, is written at each iteration.

Additionally, the modification to account for aquifer pinchouts (the partial-layer modifications described previously) required changes in the SIP package. These changes do not affect the actual mathematics of the numerical procedure. The effects of missing layers on matrix formulation are described below by first discussing row and column effects and then layer effects. It will be shown that most effects are similar to defining a cell as inactive.

First, for row and column effects, if an internal cell is identified as inactive (and missing), then the conductances between it and the four adjacent cells in that layer are set to zero so that no lateral flow is allowed to the inactive-missing cell. For the case of a cell with the standard definition of inactive, no lateral flow implies that it is impermeable, and for the case of a cell defined as inactive-missing, the underlying assumption is based on the concept that, given typical cell sizes (especially that used for the Columbia Plateau model), most flow in the vicinity of a pinchout moves vertically. With appropriate row-column conductances set to zero, the solution matrix is identical to that defining an intervening cell as inactive. For example, if cell 1,3,2 in McDonald and Harbaugh's (1988) figure 48 is set as inactive-missing, all terms along the matrix row for that cell are zero. Similarly, the term H 1,3,2 for cell 2,3,2 would be zero, as would F 1,2,2 for cell 1,2,2. As another example of row-column effects, given a grid of two rows, two columns, and three layers (fig. 8), one can write the matrix for solution of the ground-water flow equations on the basis of McDonald and Harbaugh's (1988) notation and equations 23, 79, and 80; this matrix is shown in figure 9a. If cell 1,1,2 is inactive, then all terms along the matrix row for cell 1,1,2 are set to zero, and the corresponding row-column terms (D 1,2,2 and B 2,1,2) for the adjacent cells also are set to zero (fig. 9b). The numerical implementation of row-column effects for partial layers in SIP thus is the same as that for cells with the standard definition of inactive.

Layer effects of inactive cells also are shown in figure 9b, where S 1,1,1 and Z 1,1,3 would be set to zero if cell 1,1,2 were defined as inactive. For the case where cell 1,1,2 is both inactive and missing, vertical flow is allowed between cells 1,1,1 and 1,1,3. In this case, the S and Z terms are nonzero and would be set as off-diagonal symmetric terms, as shown in figure 9c. These terms reside on diagonals that normally are zero. In order to preserve the banded structure of the matrix with seven nonzero diagonals, these terms are put in the standard S and Z diagonal locations; the true locations are transparent to the standard SIP. However, in order to know the proper (true) location of the S and Z terms for calculating the elements in the matrices [L] and [U] (McDonald and Harbaugh, 1988, equations 97a-m and equation 100), the subroutine SIP1AP of the SIP package has been modified in two locations. These modifications identify the one-dimensional subscripts NLL and NLN, of the index of two adjacent cells; NLL refers to the last layer in the same vertical column; NLN refers to the next layer. Because neither the last nor the next layer is required to be only one layer from the current cell, the changes search for the first active cell, behind and forward, respectively, in the same vertical column in order to establish the values of NLL and NLN needed to place the correct conductances in the S and Z terms.

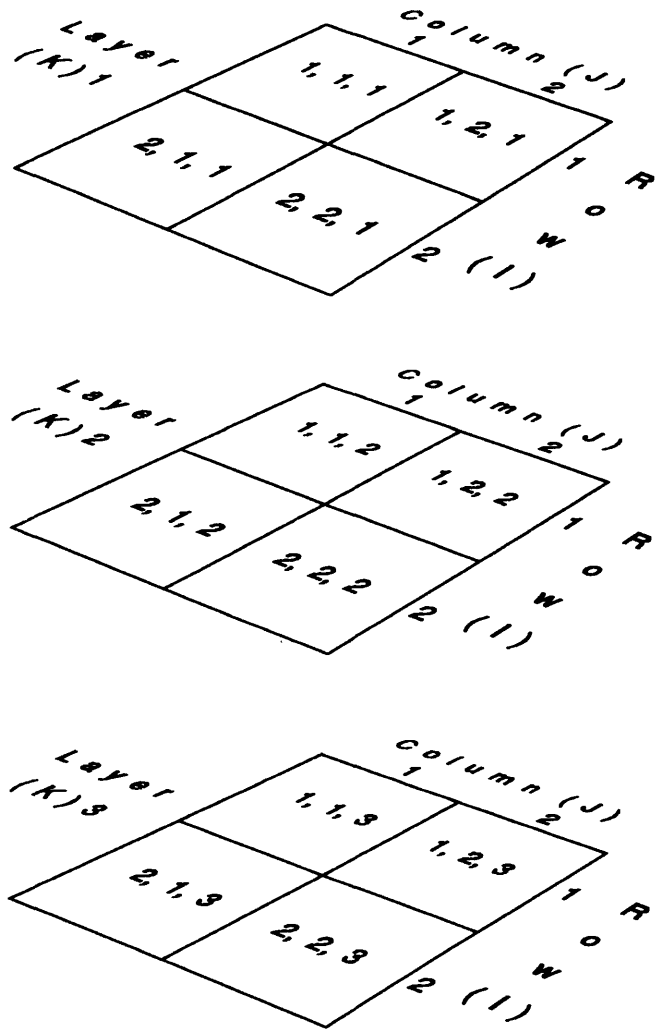


Figure 8.--Grid with two rows, two columns, and three layers, showing cell-numbering scheme using three (i,j,k) indices.

$E_{1,1,1}$	$F_{1,1,1}$	$H_{1,1,1}$		$S_{1,1,1}$							
$D_{1,2,1}$	$E_{1,2,1}$		$H_{1,2,1}$		$S_{1,2,1}$						
$B_{2,1,1}$		$E_{2,1,1}$	$F_{2,1,1}$			$S_{2,1,1}$					
	$B_{2,2,1}$	$D_{2,2,1}$	$E_{2,2,1}$				$S_{2,2,1}$				
$Z_{1,1,2}$				$E_{1,1,2}$	$F_{1,1,2}$	$H_{1,1,2}$		$S_{1,1,2}$			
	$Z_{1,2,2}$			$D_{1,2,2}$	$E_{1,2,2}$		$H_{1,2,2}$		$S_{1,2,2}$		
		$Z_{2,1,2}$		$B_{2,1,2}$		$E_{2,1,2}$	$F_{2,1,2}$			$S_{2,1,2}$	
			$Z_{2,2,2}$		$B_{2,2,2}$	$D_{2,2,2}$	$E_{2,2,2}$				$S_{2,2,2}$
				$Z_{1,1,3}$				$E_{1,1,3}$	$F_{1,1,3}$	$H_{1,1,3}$	
					$Z_{1,2,3}$			$D_{1,2,3}$	$E_{1,2,3}$		$H_{1,2,3}$
						$Z_{2,1,3}$		$B_{2,1,3}$		$E_{2,1,3}$	$F_{2,1,3}$
							$Z_{2,2,3}$		$B_{2,2,3}$	$D_{2,2,3}$	$E_{2,2,3}$

a. Full matrix

$E_{1,1,1}$	$F_{1,1,1}$	$H_{1,1,1}$									
$D_{1,2,1}$	$E_{1,2,1}$		$H_{1,2,1}$		$S_{1,2,1}$						
$B_{2,1,1}$		$E_{2,1,1}$	$F_{2,1,1}$			$S_{2,1,1}$					
	$B_{2,2,1}$	$D_{2,2,1}$	$E_{2,2,1}$				$S_{2,2,1}$				
	$Z_{1,2,2}$				$E_{1,2,2}$		$H_{1,2,2}$		$S_{1,2,2}$		
		$Z_{2,1,2}$				$E_{2,1,2}$	$F_{2,1,2}$			$S_{2,1,2}$	
			$Z_{2,2,2}$		$B_{2,2,2}$	$D_{2,2,2}$	$E_{2,2,2}$				$S_{2,2,2}$
								$E_{1,1,3}$	$F_{1,1,3}$	$H_{1,1,3}$	
					$Z_{1,2,3}$			$D_{1,2,3}$	$E_{1,2,3}$		$H_{1,2,3}$
						$Z_{2,1,3}$		$B_{2,1,3}$		$E_{2,1,3}$	$F_{2,1,3}$
							$Z_{2,2,3}$		$B_{2,2,3}$	$D_{2,2,3}$	$E_{2,2,3}$

b. Matrix with cell 1,1,2 inactive

$E_{1,1,1}$	$F_{1,1,1}$	$H_{1,1,1}$		$S_{1,1,1}$	←			$S_{1,1,1}$			
$D_{1,2,1}$	$E_{1,2,1}$		$H_{1,2,1}$		$S_{1,2,1}$						
$B_{2,1,1}$		$E_{2,1,1}$	$F_{2,1,1}$			$S_{2,1,1}$					
	$B_{2,2,1}$	$D_{2,2,1}$	$E_{2,2,1}$				$S_{2,2,1}$				
	$Z_{1,2,2}$				$E_{1,2,2}$		$H_{1,2,2}$		$S_{1,2,2}$		
		$Z_{2,1,2}$				$E_{2,1,2}$	$F_{2,1,2}$			$S_{2,1,2}$	
			$Z_{2,2,2}$		$B_{2,2,2}$	$D_{2,2,2}$	$E_{2,2,2}$				$S_{2,2,2}$
$Z_{1,1,3}$	→			$Z_{1,1,3}$				$E_{1,1,3}$	$F_{1,1,3}$	$H_{1,1,3}$	
					$Z_{1,2,3}$			$D_{1,2,3}$	$E_{1,2,3}$		$H_{1,2,3}$
						$Z_{2,1,3}$		$B_{2,1,3}$		$E_{2,1,3}$	$F_{2,1,3}$
							$Z_{2,2,3}$		$B_{2,2,3}$	$D_{2,2,3}$	$E_{2,2,3}$

c. Matrix with cell 1,1,2 inactive and with partial-layer code: flow between cells 1,1,1 and 1,1,3

EXPLANATION

- Matrix location
- $E_{i,j,k}$ Diagonal term for cell i,j,k
- B,D F,H Off-diagonal symmetric matrix terms for row-column coefficents
- S,Z Off-diagonal symmetric matrix terms for layer coefficients
- Term set to zero
- Location of symmetric layer coefficients, from partial-layer code
- ← Symmetric layer coefficients, from partial-layer code, put in standard diagonal locations

Figure 9.--Representations of coefficient matrix for a grid with two rows, two columns, and three layres.

Model Output

Output of the modified model differs from that of the Modular Model; most of the changes are additions. The budget output has been changed to an E format in order to allow for easier assessment of output and for more efficient sorting using external programs, statistical software, and operating system commands. The scaled-output mapper is a standard tool developed in the late 1970's by the Water Resources Division's Washington District; the mapper allows for an improved visual assessment of the model input and output. The mapper has been added, as one more option, to the subroutine that prints a one-layer array.

In a model run, the results of successive iterations may oscillate or diverge, or a block of cells may go dry due to numerical instability. In such cases, printing the head change at the end of each iteration lets the investigator consider terminating the run prematurely. Listing of dry-or-cancelled cells to a separate file, rather than to the main printout, allows the investigator to see them more easily for analysis. For the same reason, the flow at individual cells (as for the drain, general-head-boundary, river, and well packages) is listed in a separate file.

DOCUMENTATION OF MODIFIED MAIN PROGRAM AND PACKAGE SUBROUTINES

The following section documents the main program and the subroutines that have been modified and the new packages, flow barrier and canyon cutter, in the Columbia Plateau RASA flow model.

Main Program Package

When calling subroutines in the Allocate Procedure, BAR1AL and CUT1AL are called after BCF2AL. In the Read and Prepare Procedure, BAR1RP and CUT1RP are called after BCF2RP. In the Formulate Procedure, BAR1MD and CUT1MD are called after BCF2FM. The hydrologic budget is printed at the end of the iteration equal to the difference between the maximum number of iterations and the number of iteration parameters.

Narrative for Main Program

1. The packages' input and output unit numbers have been changed to discard the slice-successive overrelaxation and to add the flow barrier and canyon cutter packages.

Unit	Package	Action
11	Slice-successive overrelaxation	Deleted
13	Flow barrier	Added
14	Canyon cutter	Added

2. Set length of master array (equal to its dimension) to store data arrays and lists.
3. Define problem: grid, stress periods, packages, and options.
4. Allocate space in master array for data arrays and lists.
5. If size of problem is too large for master array, stop.
6. Read and prepare information for entire simulation.
7. Simulate each stress period:
    Read time of each period.
    Read and prepare information that might change each period.
    Simulate each time step:
        Read output for each step.
        Calculate step length, advance time, and store last step's heads.
        Iteratively solve equations:
            Formulate equations and approximate solution.
            If converged to solution, out of iterative loop.
            Test whether to calculate and write budget.
    Calculate and write budget at each specified period.
    Print results of each specified period.
    If not converged to solution, out of stress period loop.

Flow Barrier Package

The flow barrier package contains three subroutines: BAR1AL, BAR1RP, and BAR1MD. BAR1AL allocates space for the flow barrier package; it is grouped in the Allocate Procedures of McDonald and Harbaugh (1988). BAR1RP is part of the Read and Prepare Procedures; it reads the number of flow barriers for each layer, a master factor by which to multiply all barriers, the row- and column-address of the cells on each side of each barrier, and the factor by which to multiply the branch conductance of that cell wall.

Narrative for Subroutine BAR1AL

1. Read maximum number of flow barriers.
2. Allocate space for flow barriers.
3. Calculate number of elements in X array allocated by this package.
4. Calculate number of elements in X array allocated.
5. Write number of elements in X array allocated and size of X array.
6. If number of elements in X array allocated exceeds size of X array, write error message.

Narrative for Subroutine BAR1RP

1. Read and initialize.
2. For each layer, read number of flow barriers.
3. Read master factor (for all flow barriers):
   If not positive, set to default: 1.
4. Initialize counter for flow-barrier array.
5. For each layer:
   If there are flow barriers, process.
   For each flow barrier:
   Advance counter for flow-barrier array.
   Read row- and column-address of cells on opposite sides of flow barrier and factor.
   (Either row-addresses are equal and second column-address equals (first + 1), or second row-address equals (first + 1) and column-addresses are equal.)
   Multiply factor by master factor.
   Store addresses and factor in flow-barrier array.
6. Set flags opposite to layer-type code for constant-transmissivity (to 1 and 3) and call subroutine BAR1MD to modify branch conductance for constant-transmissivity layers before iterative loop.
7. Set flags opposite to layer-type code for variable-transmissivity (to 0 and 2) to modify branch conductance for variable-transmissivity layers during iterations.

Narrative for Subroutine BAR1MD

1. Modify branch conductance to account for flow barrier.
2. Initialize counter for flow-barrier array.
3. For each layer:
   If layer-type code is not equal to either flag, process.
   (Constant-transmissivity layers will be processed before iterative loop; variable-transmissivity layers during iterations.)
   If there are flow barriers, process.
   For each flow barrier:
   Advance counter for flow-barrier array.
   Get cell addresses and factor from flow-barrier array.
   If column-addresses are unequal, multiply row-branch conductance by factor; otherwise, multiply column-branch conductance by factor.

Block-Centered-Flow Package

Seven subroutines in the block-centered-flow package have been modified: BCF2AL, SBCF2B, BCF2FM, SBCF2N, SBCF2F, BCF2RP, and SBCF2H; two subroutines have been added: SBCANC and SBCF1K.

BCF2AL reads, and constrains to either zero or one, the required number of active adjacent cells to cancel a cell. SBCF2B finds the first active cell below, when calculating the flow in the vertical direction through the lower face of each cell. BCF2FM finds the first active cell both above and below, when correcting for vertical flow downward into a partially saturated cell. SBCF2N finds the first active cell above, when it checks whether each active cell can flow to at least one cell. SBCF2F finds the first active cell above, when calculating flow through the upper face of each constant-head cell; to calculate flow through the lower face, it finds the first active cell below. BCF2RP calls subroutine SBCF2N. SBCF2H finds the first active cell above, to set its vertical conductance to zero; it calls subroutine SBCANC.

SBCANC checks that each active cell can flow to enough adjacent cells that the solver (strongly implicit procedure) neither zerodivides nor underflows. SBCF1K calculates vertical flow through the bottom and top cell walls; it sums this flow by layer.

Narrative for Subroutine BCF2AL

1. Read flags and parameters.
2. Constrain required number of active neighbors to either 0 or 1.
3. Stop if maximum number of layers exceeds 80.
4. Read layer type.
5. Initialize top and bottom counters.
6. For each layer:
   Stop if any other than top is unconfined.
   If unconfined or fully convertible, advance bottom counter.
   If convertible (partly or fully), advance top counter.

7. Allocate space:
    Store number of elements allocated previously.
    If transient: for primary and secondary storage capacity.
    For anisotropy, bottom, hydraulic conductivity, and top.
8. Calculate number of elements in X array allocated by this package.
9. Calculate number of elements in X array allocated.
10. Write number of elements in X array allocated and size of X array.
11. If number of elements in X array allocated exceeds size of X array, write error message.

Narrative for Subroutine SBCF2B

1. Calculate flow through each forward cell wall.
2. If more than one column, calculate flow through right wall.
3. Clear array.
4. If cell-by-cell flow option chosen, call subroutine UBUDSV.
5. If more than one row, calculate flow through front wall.
6. Clear array.
7. If cell-by-cell flow option chosen, call subroutine UBUDSV.
8. If more than one layer, calculate flow through lower wall.
9. Clear array.
10. Initialize top counter.
11. Except for bottom layer:
    If convertible, advance top counter.
    If variable-head:
        Find first active cell below.
        If convertible, advance top counter.
        If convertible, use maximum of head and top.
12. If cell-by-cell flow option chosen, call subroutine UBUDSV.

Narrative for Subroutine BCF2FM

1. Formulate equations.
2. Initialize top and bottom counters.
3. For each layer:
    If convertible, advance top counter.
    If unconfined or fully convertible, advance bottom counter and calculate transmissivity from hydraulic conductivity.
4. If transient, add storage to equations:
    Initialize top counter.
    For each layer:
        If non-convertible, use primary storage coefficient.
        If convertible, check both previous and current head versus top to find whether to use primary or secondary storage coefficient (primary if head > top; secondary if not).
        Advance top counter.
5. Initialize top counter.
6. Find whether to correct for vertical flow down into partly saturated layer.

```
7. For each layer:
       If convertible, find whether to correct for leakage from above.
       Advance top counter.
       Except for top layer:
           If head below top:
               Find first active cell above:
                   Add correction term to RHS.
       Except for bottom layer:
           Find whether to correct for leakage downward.
           Find first active cell below:
               If its head below its top, add correction term to RHS.
```

Narrative for Subroutine SBCF2N

```
1. Check data.
2. Initialize bottom counter.
3. For each layer:
       If unconfined or fully convertible, advance bottom counter.
       If cell is out of model, set vertical leakance, transmissivity, and
           hydraulic conductivity to zero.
4. Assure that each active cell can flow to at least one cell.
5. Initialize bottom counter.
6. For each layer:
       If confined or fully convertible, check vertical leakance and
           transmissivity.
       Except for top layer, find first active cell above.
       If active cell cannot flow to at least one cell, convert it to
           inactive and write its address.
       If unconfined or partly convertible, check vertical leakance and
           hydraulic conductivity.
       Advance bottom counter.
       Except for top layer, find first active cell above.
       If active cell cannot flow to at least one cell, convert it to
           inactive and write its address.
7. For each layer:
       If confined or partly convertible, calculate branch conductances.
8. If more than one layer:
       Except for bottom layer:
           Calculate vertical conductance from vertical leakance and area.
9. If transient:
       Initialize top counter.
           For each layer:
               Calculate primary storage capacity from primary storage
                   coefficient and area.
               If convertible:
                   Advance top counter.
                   Calculate secondary storage capacity from secondary
                       storage coefficient and area.
```

Narrative for Subroutine SBCF2F

1. Calculate flow from constant-head cells.
2. Initialize accumulators for flow into and out of model.
3. Clear array.
4. Initialize top counter.
5. For each layer:
    If convertible, advance top counter.
    For each constant-head cell:
        Calculate flow through each cell wall with variable-head cell on other side.
        Set index for constant-head budget sets.
        Initialize fields for flow.
        Except for first column:
            Calculate flow through left wall and accumulate flow into or out of model.
        Except for last column:
            Calculate flow through right wall and accumulate flow into or out of model.
        Except for first row:
            Calculate flow through back wall and accumulate flow into or out of model.
        Except for last row:
            Calculate flow through front wall and accumulate flow into or out of model.
        Except for top layer, find first active cell above:
            If layer convertible, use maximum of head and top.
            Calculate flow through upper wall and accumulate flow into or out of model.
        Except for bottom layer, find first active cell below:
            If this layer convertible, use maximum of head and top.
            Calculate flow through lower wall and accumulate flow into or out of model.
        Sum flow through cell walls.
        Accumulate flow into and out of model.
        If cell-budget option, write flow.
        Store flow in array.
6. Write array.
7. Save flow and labels in table.
8. Advance label counter.

Narrative for Subroutine BCF2RP

1. Read and prepare.
2. Read anisotropy by layers, row spacing, and column spacing.
3. Initialize top and bottom counters.
4. For each layer:
   Find storage address of each array.
   If unconfined or fully convertible, advance bottom counter.
   If convertible, advance top counter.
   Calculate pointer to part of each array except bottom, hydraulic conductivity, top, and secondary storage coefficient.
   Calculate pointer to part of bottom and hydraulic conductivity.
   Calculate pointer to part of top and secondary storage coefficient.
   If transient, read primary storage coefficient.
   If confined or partly convertible, read transmissivity.
   If unconfined or fully convertible, read hydraulic conductivity and bottom.
   Except for bottom layer, read vertical leakance.
   If convertible:
      If transient, read secondary storage coefficient.
      Read top.
5. Call subroutine SBCF2N to check data.

Narrative for Subroutine SBCF2H

1. Calculate transmissivity from saturated thickness and hydraulic conductivity.
2. For each cell:
   If layer convertible and head > top, head = top.
   Calculate saturated thickness from head and bottom.
   If saturated thickness not positive:
      Write cell address.
      Set head, transmissivity, and boundary indicator to zero.
      Except for bottom layer, set vertical conductivity to zero.
      Except for top layer, find first active cell above:
         Set that vertical conductivity to zero.
   Calculate transmissivity from saturated thickness and hydraulic conductivity.
3. Call subroutine SBCANC to check that each active cell can flow to enough adjacent cells that the solver, strongly implicit procedure, neither zerodivides nor underflows.
4. Calculate branch conductances.

Narrative for Subroutine SBCANC

1. Initialize counter.
2. For each cell:
    For each lateral direction, within limits of array dimension:
        If neighbor cell inactive, proceed to next direction.
        Set variable to absolute value of value in boundary array:
            If direction to rear, set to value for neighbor; if to front, set to value for cell.
        If direction to either back or front, compare variable to both column-branch and both-branches flags; if to either left or right, compare it to both row-branch and both-branches flags. If variable is equal to either, proceed to next direction.
        Advance counter.
    If number of active neighbors too few to prevent cell from being cancelled:
        Write cell address, iteration counter, time step counter, and stress period counter.
        Set head, transmissivity, and boundary indicator to zero.
        Except for bottom layer, set vertical conductivity to zero.

Narrative for Subroutine SBCF1K

1. Calculates flow through bottom and top cell walls and sums it by layer.
2. If there is more than one layer, process.
3. Initialize top counter.
4. For each layer:
    Initialize accumulators of flow in, and out, of top and bottom.
    For each other layer, initialize accumulators of flow in, and out, of top and bottom.
    If convertible, advance top counter.
    For each cell:
        If active:
            Except for top layer, find first active cell above.
            Accumulate positive flow as flow in; accumulate negative flow as flow out.
            Except for bottom layer, find first active cell below.
                Initialize top counter for layers below.
                    For each layer below, if convertible, advance top counter.
            Accumulate positive flow as flow in; accumulate negative flow as flow out.
            Calculate net flow for top surface, bottom surface, and entire layer.
    For each other layer, write flow in and out.
    Write flow in and out.

Canyon Cutter Package

The canyon cutter package contains four subroutines: CUT1AL, CUT1RP, CUT1IB, and CUT1MD. CUT1AL allocates space for the canyon cutter package; it is grouped in the Allocate Procedures of McDonald and Harbaugh (1988). CUT1RP is part of the Read and Prepare Procedures; it reads the number of canyon cutters for each layer, and the row- and column-address of the cells on each side of each cutter. CUT1IB signals the boundary array to account for a canyon cut through a layer. CUT1MD sets a branch conductance to zero to account for each canyon cut through each layer.

Narrative for Subroutine CUT1AL

1. Read maximum number of canyon cutters and status-array flags (row-branch, column-branch, and both branches).
2. Allocate space for canyon cutters.
3. Calculate number of elements in X array allocated by this package.
4. Calculate number of elements in X array allocated.
5. Write number of elements in X array allocated and size of X array.
6. If number of elements in X array allocated exceeds size of X array, write error message.

Narrative for Subroutine CUT1IB

1. Sets status-array flags to indicate which branch conductances will be set to zero to account for canyon cut thru layer.
2. Initialize counter for canyon-cutter array.
3. For each layer:
   If there are canyon cutters, process.
   For each canyon cutter:
   Advance counter for canyon-cutter array.
   Get cell addresses from canyon-cutter array.
   If cell active or constant-head, process.
   Set constant-head-status flag to 1.
   If constant head, set constant-head-status flag to -1 to preserve minus sign.
   Set signed-row-branch flag to product of row-branch flag and constant-head-status flag.
   If column-addresses are unequal, row-branch conductance will be set to zero; set signed-row-branch flag. Otherwise, column-branch conductance will be set to zero. If signed-row-branch flag is not set, set column-branch flag (with proper sign); if signed-row-branch flag has been set, write both-branches flag (with proper sign) over it.

Narrative for Subroutine CUT1RP

1. Read and initialize.
2. For each layer, read number of canyon cutters.
3. Initialize counter for canyon-cutter array.
4. For each layer:
   If there are canyon cutters, process.
   For each canyon cutter:
   Advance counter for canyon-cutter array.
   Read row- and column-address of cells on opposite sides of canyon cutter.
   (Either row-addresses are equal and second column-address equals (first + 1), or second row-address equals (first + 1) and column-addresses are equal.)
   (If cell has two cutters, read that for row-branch (column-branch addresses are unequal) first so that subroutine CUT1IB will see them in correct order.)
   Store addresses in canyon-cutter array.
5. Call subroutine CUT1IB to set status-array flags to indicate which branch conductances will be set to zero to account for canyon cut thru layer.
6. Set flags opposite to layer-type code for constant-transmissivity (to 1 and 3) and call subroutine CUT1MD to set branch conductance to zero for constant-transmissivity layers before iterative loop.
7. Set flags opposite to layer-type code for variable-transmissivity (to 0 and 2) to set branch conductance to zero for variable-transmissivity layers during iterations.

Narrative for Subroutine CUT1MD

1. Set branch conductance to zero to account for canyon cut thru layer.
2. Initialize counter for canyon-cutter array.
3. For each layer:
   If layer-type code is not equal to either flag, process.
   (Constant-transmissivity layers will be processed before iterative loop; variable-transmissivity layers during iterations.)
   If there are canyon cutters, process.
   For each canyon cutter:
   Advance counter for canyon-cutter array.
   Get cell addresses from canyon-cutter array.
   If column-addresses are unequal, set row-branch conductance to zero; otherwise, set column-branch conductance to zero.

Strongly Implicit Procedure Package

Two subroutines of the strongly implicit procedure package have changes: SIP2AP and SSIP2I. SIP2AP has many changes. To assign variables to the adjacent cell in layers behind, the first active cell behind must be found; to assign variables to the adjacent cell in layers ahead, the first active cell ahead must be found. After calculating the components of the upper and lower triangular matrices, if the absolute value of the diagonal from the lower factor is less than the test for zerodivide or underflow, its address is written, and the program stops. Calculating the subscripts for neighbors and conductance to them requires finding the first active cell in layers behind. Before returning to the main program, the largest value of head change, its cell address, and the sum of the absolute values of head change are written. SSIP2I has one change: when finding the conductance from each cell to the cell in layers behind, the first active cell behind must be found.

Narrative for Subroutine SIP2AP

1. Solve by strongly implicit procedure.
2. At start of run, find whether to calculate seed and iteration parameters.
3. Assign constants for each iteration.
4. Initialize head-change trackers and work arrays.
5. Set normal-or-reverse equation orderer and calculate indices dependent on ordering.
6. Calculate intermediate vector with forward substitution.
7. For each cell:
    - Set cell-location indices, calculate subscript, and skip calculations if inactive or constant head.
    - Calculate subscripts (dependent on ordering) for neighbors and conductance to them.
    - Except for first row, assign variables to matrices one row behind.
    - Except for last row, assign variables to matrices one row ahead.
    - Except for first column, assign variables to matrices one column behind.
    - Except for last column, assign variables to matrices one column ahead.
    - Except for top layer, assign variables to first active cell in layers behind.
    - Except for bottom layer, assign variables to first active cell in layers ahead.
    - Calculate negative sum of conductances to all neighbors.
    - Calculate components of upper and lower matrices.
        - If absolute value of diagonal from lower factor is less than test for zerodivide or underflow, write its address, and stop.
    - Calculate residual.
    - Calculate intermediate vector.
8. Solve head change by back substitution.
9. For each cell:
    - Set cell-location indices, calculate subscript, and skip calculations if inactive or constant head.
    - Calculate subscripts (dependent on ordering) for neighbors and conductance to them (requires finding first active cell in layers behind).

Back substitute; store head change in place of intermediate vector
from forward substitution.
If largest absolute value of head change in grid so far,
save head change, its absolute value, and its cell address.
Get new estimate of head by adding this head change to head from
previous iteration.
10. Write largest value of head change, its cell address, and sum of
absolute value of head changes.

Narrative for Subroutine SSIP2I

1. Calculates seed and iteration parameters.
2. Calculate constants and initialize variables.
3. For each cell:
If active, calculate seed:
Get conductance to neighbors:
Except for first column, one column behind.
Except for last column, one column ahead.
Except for first row, one row behind.
Except for last row, one row ahead.
Except for top layer, find first active cell in layers behind.
Except for bottom layer, one layer ahead.
Find maximum and minimum of conductances in each of the coordinate
directions.
Calculate seed in each coordinate direction.
Cell seed is minimum of these; grid seed is minimum cell seed.
Accumulate cell seed.
4. Calculate average cell seed.
5. Calculate iteration parameters from average cell seed.

Utility Module

UL2PRW is the only modified subroutine in the utility package; it has two changes. If the format code is less than one, or greater than thirteen, it is set to thirteen. If the format code is thirteen, UL2PRW calls USCMP, and returns control to the package that called it. There are two new subroutines: UWRIB and USCMP. UWRIB writes the boundary array, where value at each dry cell has been set to zero, to a separate file. USCMP writes a map of a real array.

Narrative for Subroutine UL2PRW

1. Print header.
2. If format code is less than 1, or greater than 13, set it to 13.
3. If format code is 13, call USCMP, and return.
4. Call UCOLNO to print column numbers.
5. Print each row with appropriate format code.

Narrative for Subroutine USCMP

1. Write map of real array.
2. Set two constants: first to ten thousand, second to one thousand.
3. Initialize largest absolute value in array.
4. Find largest absolute value in array.
5. Scale array so largest absolute value has four digits.
6. Set scaler to one.
7. If largest absolute value equals zero, write that and return.
8. Scale array until largest absolute value has four digits.
   - If largest absolute value is less than one thousand:
     - Multiply both scaler and largest absolute value by ten; return to test.
   - If largest absolute value equals one thousand, it has four digits; continue.
   - If largest absolute value is greater than one thousand and less than ten thousand, it has four digits; continue.
   - If largest absolute value is greater than or equals ten thousand:
     - Divide both scaler and largest absolute value by ten; return to test.
   - Largest absolute value in array has four digits.
9. Calculate reciprocal of scaler, and set left column to one and right column to twenty-five.
10. Write map, less than twenty-six columns wide.
    - Write title, layer number, and reciprocal.
    - Set right column to smaller of right column and number of model columns.
    - Write column numbers across top of map.
    - For each row:
      - Initialize counter for integer array of scaled numbers.
      - For each column to be written:
        - Advance counter for integer array.
        - Set signed scaled number into integer array.
      - Write integer array, with row number to both left and right.
    - Write column numbers across bottom of map.
    - Advance left column by twenty-five.
    - If left column is less than or equals number of model columns, write map of next set of columns.

Narrative for Subroutine UWRIB

1. Writes formatted boundary array to separate file, where value at each dry cell has been set to zero.
2. For each layer:
   - For each row:
     - For each column, write value in boundary array, on format (40I3).

List of New Variable Names in Old Packages

For each new variable, the definition is given. Unless stated otherwise, each new variable is in both the block-centered-flow and strongly implicit procedure packages. Some of these variables also are listed in the following section, "List of variable names in new packages."

Name	Definition
CNSTNT	In basic package, constant added to each input head of layer, at each variable-head cell. Head is set to zero at each inactive cell; input head is retained at each constant-head cell.
DLAB	Absolute value of DL, in strongly implicit procedure package.
FALL	Master factor for several packages: for each yield in well package; and for each conductance in drain, general head boundary, and river packages.
KAH	Index for each layer in search for first active layer ahead, in strongly implicit procedure package.
KBE	Index for each layer in search for first active layer behind, in strongly implicit procedure package.
KL	Index for layers in search for first active cell vertically.
KLM1	NLAY-1 (number of layers minus one).
KLO	Index for each layer in search for first active cell below.
KTLO	Index for top of each layer in search for first active cell below.
KUP	Index for each layer in search for first active cell above.
MBH	Same as MBUDHD, in block-centered-flow, drain, general head boundary, river, and well packages.
MBTH	In block-centered-flow package, flag that both branch conductances have been set to zero.
MBUD1	MXITER-NPARM (flag to write budget at this iteration).
MBUDHD	Flag to test whether to write budget at this iteration.
MCANC	In block-centered-flow package, maximum number of active neighbors that a cell needs to be cancelled (set as inactive).
MCCL	In block-centered-flow package, flag that column-branch conductance has been set to zero.
MCRW	In block-centered-flow package, flag that row-branch conductance has been set to zero.
MFIR	In basic package, initial index to add CNSTNT to input heads.

MLAS In basic package, final index to add CNSTNT to input heads.

NCNR NCOL*NROW (number of cells in layer).

NDSM1 NCOL*NROW*KLM1 (number of cells in (number of layers minus one)).

SMCHK In strongly implicit procedure package, accumulates head change.

TINY In strongly implicit procedure package, test for zerodivide or underflow.

List of New Variable Names in New Packages

BARCOM Labelled common to hold number of flow barriers in each layer.

BRRR Array to hold node, forward node, and factor for each flow barrier.

CTTR Array to hold node and forward node for each canyon cutter.

CUTCOM Labelled common to hold number of canyon cutters in each layer.

FALL Master factor for every flow barrier.

FCTR Factor for each flow barrier.

I In flow barrier package, row-address of node behind barrier; in canyon cutter package, row-address of node behind cutter.

IP1 In flow barrier package, row-address of node ahead of barrier; in canyon cutter package, row-address of node ahead of cutter.

J In flow barrier package, column-address of node behind barrier; in canyon cutter package, column-address of node behind cutter.

JP1 In flow barrier package, column-address of node ahead of barrier; in canyon cutter package, column-address of node behind cutter.

KBR1 In flow barrier package, first of two signals set opposite to layer-type code.

KBR2 In flow barrier package, second of two signals set opposite to layer-type code.

KCT1 In canyon cutter package, first of two signals set opposite to layer-type code.

KCT2 In canyon cutter package, second of two signals set opposite to layer-type code.

LAYBAR Number of flow barriers in each layer.

LAYCUT Number of canyon cutters in each layer.

LBAR Location in X array of list of flow barrier data.

LCUT Location in X array of list of canyon cutter data.

LMNTS In flow barrier package, number of words in X array allocated by package; in canyon cutter package, number of words in X array allocated by package.

MBTH Flag that both branch conductances have been set to zero.

MCCL Flag that column-branch conductance has been set to zero.

MCRW Flag that row-branch conductance has been set to zero.

MXBAR Maximum number of flow barriers.

MXCUT Maximum number of canyon cutters.

NMBR In flow barrier package, index for barriers; in canyon cutter package, index for cutters.

NP Flag to preserve minus sign for constant-head cells.

NPCRW MCRW*NP (signed-row-branch flag).

In BAS2DF, do not read input number for slice-successive overrelaxation package; read input numbers for flow-barrier and canyon-cutter packages.

BAS2DF

4. Data: IUNIT(24)
   Format: 24I3
      (BCF WEL DRN RIV EVT XXX GHB RCH SIP XXX XXX OC BAR CUT)

In BAS2RP, do not read value to be printed at no-flow cells between boundary array and starting heads; after starting heads: for each layer, read a constant (CNSTNT) to add to the head at each variable-head cell.

BAS2RP

6. Data: IBOUND(NCOL,NROW)
   Module: U2DINT
   (One array for each layer in grid)

7. Data: Shead(NCOL,NROW)
   Module: U2DREL
   (One array for each layer in grid)

8. Data: CNSTNT(NLAY) (Maximum of 80 layers)
   Format: 10F10.0

In BCF2AL, read maximum number of active neighbors (MCANC) that a cell needs to be cancelled (set as inactive).

1. Data:	ISS	IBCFCB	MCANC
Format:	I10	I10	I10

In DRN1RP, GHB1RP, RIV1RP, and WEL1RP, read a master factor (FALL) after (number of drains, head bounds, reaches, or wells)(or flag to reuse data).

2. Data: ITMP
   Format: I10

3. Data: FALL
   Format: F10.0

4. Data:	Layer	Row	Col	...
Format:	I10	I10	I10	...

In SIP1RP, read test (TINY) for zerodivide or underflow.

2. Data:	ACCL	HCLOSE	IPCALC	WSEED	IPRSIP	TINY
Format:	F10.0	F10.0	I10	F10.0	I10	F10.0

In U2DINT and U2DREL, do not read an unformatted dummy record before an unformatted array.

In BAR1AL, read maximum number (MXBAR) of barriers.

In BAR1RP, read number of barriers (LAYBAR) in each layer, a master factor (FALL) for reducing branch conductances, and for each barrier: addresses of cell (I & J) and forward cell (IP1 & JP1), and factor (FCTR).

BAR1AL

1. Data: MXBAR
   Format: I10

BAR1RP

2. Data: LAYBAR(NLAY) (Maximum of 80 layers)
   Format: 10F10.0

3. Data: FALL
   Format: F10.0

4. Data: I J IP1 JP1 FCTR
   Format: I10 I10 I10 I10 F10.0

In CUT1AL, read maximum number (MXCUT) of cutters, and three flags that branch conductances have been set to zero: row-branch (MCRW), column-branch (MCCL), and both branches (MBTH). These flags are unique integers, such as 98, 120, etc., that are written into the boundary array for later checking by the block-centered-flow package.

In CUT1RP, read number of cutters (LAYCUT) in each layer, and for each cutter: addresses of cell (I & J) and forward cell (IP1 & JP1).

CUT1AL

1. Data: MXCUT MCRW MCCL MBTH
   Format: I10 I10 I10 I10

CUT1RP

2. Data: LAYCUT(NLAY) (Maximum of 80 layers)
   Format: 10I10

3. Data: I J IP1 JP1
   Format: I10 I10 I10 I10

REFERENCES

McDonald, M.B., and Harbaugh, A.W., 1988, A modular three-dimensional finite-difference ground-water flow model: U.S. Geological Survey, Techniques of Water-Resources Investigations, Book 6, Chapter A1, 14 chapters.

Hansen, A.J., Jr., Vaccaro, J.J., and Bauer, H.H., 1993, Ground-water flow simulation of the Columbia Plateau Regional Aquifer System, Washington, Oregon, and Idaho: U.S. Geological Survey, Water Resources Investigations Report 91-4187, 15 plates, 101 p.

Hansen, A.J., Jr., 1993, Archiving of source code for the finite-difference flow model and the post-processors and input and output files for the Columbia Plateau regional aquifer system, Washington, Oregon, and Idaho: U.S. Geological Survey Open-File Report 90-364, 9 p.

Vaccaro, J.J., 1986, Plan of study for the Regional Aquifer-System Analysis, Columbia Plateau, Washington, northern Oregon, and northwestern Idaho: U.S. Geological Survey Water-Resources Investigations Report 85-4151, 25 p.

– PAGE 42 FOLLOWS –

main print

```
U.S. GEOLOGICAL SURVEY MODULAR FINITE-DIFFERENCE GROUND-WATER MODEL
SAMPLE----3 LAYERS, 15 ROWS, 15 COLUMNS; STEADY STATE; CONSTANT HEADS COLUMN 1, LAYERS 1 AND 2; RECHARGE, WELLS AND DRAINS
   3 LAYERS   15 ROWS   15 COLUMNS
  1 STRESS PERIOD(S) IN SIMULATION
MODEL TIME UNIT IS SECONDS
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24
 29  30  31   0   0   0   0  36  37   0   0  40   0   0   0   0   0   0   0   0   0   0   0   0
BCF WEL DRN RIV EVT     GHB RCH SIP         OUT BAR CUT
BAS, 30SEP1991, READ ON     5
ARRAYS RHS AND BUFF WILL SHARE MEMORY.
START HEAD WILL NOT BE SAVED -- DRAWDOWN CANNOT BE CALCULATED
    5892 ELEMENTS IN X ARRAY ARE USED BY BAS
    5892 ELEMENTS OF X ARRAY USED OUT OF   26000
BCF, 30SEP1991, READ ON    29
STEADY-STATE SIMULATION
NUMBER: ACTIVE NEIGHBORS INADEQUATE (MCANC):    0
CONSTANT HEAD CELL-BY-CELL FLOWS WILL BE PRINTED
LAYER  AQUIFER TYPE
        1         1
        2         0
        3         0
     453 ELEMENTS IN X ARRAY ARE USED BY BCF
    6345 ELEMENTS OF X ARRAY USED OUT OF   26000
WEL, 30SEP1991, READ ON    30
MAXIMUM OF   15 WELLS
CELL-BY-CELL FLOWS WILL BE PRINTED WHEN ICBCFL NOT 0
      60 ELEMENTS IN X ARRAY ARE USED FOR WELLS
    6405 ELEMENTS OF X ARRAY USED OUT OF   26000
DRAIN, 30SEP1991, READ ON    31
MAXIMUM OF    9 DRAINS
CELL-BY-CELL FLOWS WILL BE PRINTED WHEN ICBCFL NOT 0
      45 ELEMENTS IN X ARRAY ARE USED FOR DRAINS
```

```
    6450 ELEMENTS OF X ARRAY USED OUT OF   26000
 RCH, 30SEP1991, READ ON    36
 OPTION 1 -- RECHARGE TO TOP LAYER
     225 ELEMENTS OF X ARRAY USED FOR RECHARGE
    6675 ELEMENTS OF X ARRAY USED OUT OF   26000
 SIP, 30SEP1991, READ ON    37
 MAXIMUM OF  50 ITERATIONS ALLOWED FOR CLOSURE
  5 ITERATION PARAMETERS
    2905 ELEMENTS IN X ARRAY ARE USED BY SIP
    9580 ELEMENTS OF X ARRAY USED OUT OF   26000
 SAMPLE----3 LAYERS, 15 ROWS, 15 COLUMNS; STEADY STATE; CONSTANT HEADS COLUMN 1, LAYERS 1 AND 2; RECHARGE, WELLS AND DRAINS
          BOUNDARY ARRAY FOR LAYER  1 WILL BE READ ON UNIT  5 USING FORMAT:      (4013)

        1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
 ------------------------------------------------------------------
0  1   -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
0  2   -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
0  3   -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
0  4   -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
0  5   -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
0  6   -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
0  7   -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
0  8   -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
0  9   -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
0 10   -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
0 11   -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
0 12   -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
0 13   -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
0 14   -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
0 15   -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
          BOUNDARY ARRAY FOR LAYER  2 WILL BE READ ON UNIT  5 USING FORMAT:      (4013)

        1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
 ------------------------------------------------------------------
0  1   -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
0  2   -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
0  3   -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
0  4   -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
```

```
0  5   -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
0  6   -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
0  7   -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
0  8   -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
0  9   -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
0 10   -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
0 11   -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
0 12   -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
0 13   -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
0 14   -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
0 15   -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
          BOUNDARY ARRAY =              1 FOR LAYER  3
            INITIAL HEAD =  0.0000000     FOR LAYER  1
            INITIAL HEAD =  0.0000000     FOR LAYER  2
            INITIAL HEAD =  0.0000000     FOR LAYER  3
CONSTANT ADDED TO INPUT HEADS:
   K    CONSTANT
   1   0.000E-01
   2   0.000E-01
   3   0.000E-01
HEAD PRINT FORMAT IS FORMAT NUMBER  13    DRAWDOWN PRINT FORMAT IS FORMAT NUMBER   0
HEADS WILL BE SAVED ON UNIT 58    DRAWDOWNS WILL BE SAVED ON UNIT  0
OUTPUT CONTROL IS SPECIFIED EVERY TIME STEP
COLUMN TO ROW ANISOTROPY =   1.000000
                    DELR =   5000.000
                    DELC =   5000.000
   HYD. COND. ALONG ROWS =  0.9999999E-03 FOR LAYER  1
                  BOTTOM =  -150.0000     FOR LAYER  1
VERT HYD COND /THICKNESS =  0.2000000E-07 FOR LAYER  1
    TRANSMIS. ALONG ROWS =  0.1000000E-01 FOR LAYER  2
VERT HYD COND /THICKNESS =  0.1000000E-07 FOR LAYER  2
    TRANSMIS. ALONG ROWS =  0.2000000E-01 FOR LAYER  3
SOLUTION BY THE STRONGLY IMPLICIT PROCEDURE
MAXIMUM ITERATIONS ALLOWED FOR CLOSURE =       50
ACCELERATION PARAMETER =   1.0000E+00
HEAD CHANGE CRITERION FOR CLOSURE =   1.00000E-03
SIP HEAD CHANGE PRINTOUT INTERVAL =        1
TEST FOR ZERO-DIVIDE OR UNDERFLOW (TINY):   1.000E-19
```

```
   5 ITERATION PARAMETERS CALCULATED FROM SPECIFIED WSEED =   1.00000E-03
  0.00000E-01   8.22172E-01   9.68377E-01   9.94377E-01   9.99000E-01
STRESS PERIOD NO.   1, LENGTH =   86400.00
NUMBER OF TIME STEPS =     1
MULTIPLIER FOR DELT =     1.000
INITIAL TIME STEP SIZE =   86400.00
FCWEL   1.0000E+00
  15 WELLS
FCDRN   1.0000E+00
    9 DRAINS
              RECHARGE =  0.3000000E-07
HEAD/DRAWDOWN PRINTOUT FLAG = 1     TOTAL BUDGET PRINTOUT FLAG = 1     CELL-BY-CELL FLOW TERM FLAG = 1
OUTPUT FLAGS FOR ALL LAYERS ARE THE SAME:
  HEAD     DRAWDOWN  HEAD  DRAWDOWN
PRINTOUT  PRINTOUT  SAVE    SAVE
    1         0       0       0
NMIT   K   I   J     MAX CHNG   CMLTV CHNG
   1   3   5  11  -2.2411E+01   1.2492E+03
   2   1   1  15   1.2475E+01   3.4689E+03
   3   3   1  14   1.3390E+01   3.3355E+03
   4   1   1  15   4.8207E+01   1.6983E+04
   5   3   1  13   3.5906E+01   1.0772E+04
   6   1   9  14   2.4821E+00   3.6973E+02
   7   3  10  13   1.4300E+00   3.2226E+02
   8   1  12  14   6.2141E+00   1.2815E+03
   9   3  11  14   7.4112E+00   2.3475E+03
  10   1  15  15   1.3659E+01   2.9825E+03
  11   3   8   7   5.5031E-01   4.1935E+01
  12   2   6   9   4.8208E-01   1.3045E+02
  13   3   5  10   4.7106E-01   1.6464E+02
  14   1  11  14   2.0185E+00   7.6159E+02
  15   3   5  13   2.3018E+00   8.3331E+02
  16   1  13  12   1.1083E-01   1.7321E+01
  17   3  12  11   7.0583E-02   1.4858E+01
  18   1  14  14   2.8186E-01   4.8599E+01
  19   3  13  14   3.1406E-01   7.2854E+01
  20   1  15  15   3.3199E-01   4.5477E+01
  21   1  13  12   7.8525E-03   1.0383E+00
```

```
  22   2  11  11   1.5862E-02   4.4744E+00
  23   3  11  10   1.7771E-02   6.1095E+00
  24   1  14  14   7.9098E-02   2.5610E+01
  25   3   7  14   8.4992E-02   3.2257E+01
  26   1  13  14   4.1683E-03   6.2666E-01
  27   3  14  15   2.5554E-03   5.3316E-01
  28   1  14  14   9.7682E-03   1.7284E+00
  29   3  13  14   1.0823E-02   2.4309E+00
  30   1  15  15   1.0297E-02   1.4126E+00
  31   1  13  12   2.4289E-04   3.4628E-02
  31 ITERATIONS FOR TIME STEP   1 IN STRESS PERIOD  1
MAXIMUM HEAD CHANGE FOR EACH ITERATION:
HEAD CHANGE LAYER,ROW,COL HEAD CHANGE LAYER,ROW,COL HEAD CHANGE LAYER,ROW,COL HEAD CHANGE LAYER,ROW,COL HEAD CHANGE LAYER,ROW,COL
  -22.41     ( 3,  5, 11)   12.48     ( 1,  1, 15)   13.39     ( 3,  1, 14)   48.21     ( 1,  1, 15)   35.91     ( 3,  1, 13)
   2.482     ( 1,  9, 14)   1.430     ( 3, 10, 13)   6.214     ( 1, 12, 14)   7.411     ( 3, 11, 14)   13.66     ( 1, 15, 15)
  0.5503     ( 3,  8,  7)  0.4821     ( 2,  6,  9)  0.4711     ( 3,  5, 10)   2.018     ( 1, 11, 14)   2.302     ( 3,  5, 13)
  0.1108     ( 1, 13, 12)  0.7058E-01 ( 3, 12, 11)  0.2819     ( 1, 14, 14)  0.3141     ( 3, 13, 14)  0.3320     ( 1, 15, 15)
  0.7853E-02 ( 1, 13, 12)  0.1586E-01 ( 2, 11, 11)  0.1777E-01 ( 3, 11, 10)  0.7910E-01 ( 1, 14, 14)  0.8499E-01 ( 3,  7, 14)
  0.4168E-02 ( 1, 13, 14)  0.2555E-02 ( 3, 14, 15)  0.9768E-02 ( 1, 14, 14)  0.1082E-01 ( 3, 13, 14)  0.1030E-01 ( 1, 15, 15)
  0.2429E-03 ( 1, 13, 12)
FLOW BETWEEN LAYERS
  K          TOPIN          TPOUT         TOTOPIN          BOTIN          BTOUT         TOBOTIN         TOTALIN
      1  0.00000E-01    0.00000E-01                    0.00000E-01    0.00000E-01
      2  0.00000E-01    0.00000E-01                    7.58970E+00   -3.00900E+01
      3  0.00000E-01    0.00000E-01                    0.00000E-01    0.00000E-01
  1       0.00000E-01    0.00000E-01   0.00000E-01      7.58970E+00   -3.00900E+01   -2.25003E+01   -2.25003E+01
      1   3.00900E+01   -7.58970E+00                    0.00000E-01    0.00000E-01
      2   0.00000E-01    0.00000E-01                    0.00000E-01    0.00000E-01
      3   0.00000E-01    0.00000E-01                    5.84375E+00   -1.56937E+01
  2       3.00900E+01   -7.58970E+00   2.25003E+01      5.84375E+00   -1.56937E+01   -9.84991E+00    1.26504E+01
      1   0.00000E-01    0.00000E-01                    0.00000E-01    0.00000E-01
      2   1.56937E+01   -1.06935E+01                    0.00000E-01    0.00000E-01
      3   0.00000E-01    0.00000E-01                    0.00000E-01    0.00000E-01
  3       1.56937E+01   -1.06935E+01   5.00012E+00      0.00000E-01    0.00000E-01    0.00000E-01    5.00012E+00
          HEAD IN LAYER  1 AT END OF TIME STEP  1 IN STRESS PERIOD  1
          HEAD    1 MULTIPLY BY     1.0E-01
     1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
   .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .
```

1	0	249	440	593	718	825	919	1000	1069	1126	1174	1213	1243	1264	1274	1
2	0	244	431	580	702	806	901	984	1053	1110	1157	1196	1227	1249	1261	2
3	0	234	413	554	668	762	865	952	1022	1076	1120	1161	1196	1221	1234	3
4	0	219	386	518	618	680	813	908	976	1025	1061	1107	1149	1179	1194	4
5	0	197	349	473	577	667	771	858	922	961	973	1031	1088	1125	1143	5
6	0	165	295	409	513	612	712	798	865	908	930	942	1021	1064	1084	6
7	0	116	211	312	414	518	631	727	800	849	886	917	964	998	1018	7
8	0	35	68	163	263	370	526	643	725	773	820	850	893	917	943	8
9	0	105	191	281	369	453	530	554	652	661	739	738	808	802	865	9
10	0	146	259	354	435	501	549	575	629	655	704	724	767	783	818	10
11	0	171	300	400	478	532	558	533	603	593	664	655	722	710	776	11
12	0	187	326	431	508	559	583	585	619	632	671	685	723	735	769	12
13	0	197	342	451	530	580	599	568	626	609	672	658	719	703	765	13

```
14    0  203  353  465  546  601  632  645  672  688  716  732  758  770  791   14
  .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .

15    0  206  358  472  555  613  650  675  699  720  743  762  782  797  808   15
  .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .
      1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
           HEAD IN LAYER  2 AT END OF TIME STEP  1 IN STRESS PERIOD  1
           HEAD    2 MULTIPLY BY     1.0E-01
      1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
  .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .

 1    0  247  437  590  716  823  917  999 1067 1125 1172 1211 1241 1262 1273    1
  .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .

 2    0  242  428  577  700  804  899  982 1051 1108 1155 1194 1226 1248 1259    2
  .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .

 3    0  232  410  552  665  758  863  950 1020 1074 1118 1160 1195 1219 1232    3
  .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .

 4    0  217  383  515  614  602  809  906  975 1023 1054 1104 1148 1177 1192    4
  .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .

 5    0  195  347  471  574  663  769  856  920  954  911 1021 1086 1124 1142    5
  .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .

 6    0  163  292  407  511  610  710  796  863  905  921  862 1017 1062 1083    6
  .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .

 7    0  114  209  310  412  517  629  725  798  847  883  912  962  996 1016    7
  .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .

 8    0   42   83  176  276  382  529  642  723  771  818  849  891  916  942    8
  .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .

 9    0  104  190  280  368  452  529  561  651  668  739  745  808  808  864    9
  .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .
```

```
10   0  144  256  352  433  499  548  575  628  655  702  724  766  782  816   10
 .   .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .

11   0  169  297  398  476  530  557  541  602  600  664  662  722  718  775   11
 .   .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .

12   0  184  323  428  506  557  582  584  618  631  670  684  721  734  767   12
 .   .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .

13   0  194  340  449  528  578  598  575  625  616  672  665  718  711  764   13
 .   .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .

14   0  200  350  463  544  599  630  644  671  687  715  731  757  769  789   14
 .   .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .

15   0  203  355  469  553  611  648  673  698  718  741  760  780  795  806   15
 .   .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .

     1    2    3    4    5    6    7    8    9   10   11   12   13   14   15

          HEAD IN LAYER  3 AT END OF TIME STEP  1 IN STRESS PERIOD  1
          HEAD    3 MULTIPLY BY     1.0E-01

     1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
 .   .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .

 1  18  243  434  587  713  821  915  996 1065 1123 1170 1209 1239 1260 1271    1
 .   .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .

 2  18  239  425  574  697  801  897  980 1049 1106 1153 1192 1224 1246 1257    2
 .   .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .

 3  17  229  407  549  662  753  860  948 1017 1072 1115 1157 1193 1217 1230    3
 .   .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .

 4  16  213  380  512  609  627  804  903  972 1019 1041 1100 1145 1175 1190    4
 .   .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .

 5  14  192  343  467  571  658  765  853  917  942  775 1007 1082 1121 1140    5
 .   .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .
```

```
 6   12  160  289  403  508  607  707  794  860  901  906  886 1012 1060 1080    6
   .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .
 7    8  112  208  309  411  515  627  722  795  845  880  908  959  994 1014    7
   .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .
 8    4   51  102  193  292  398  534  641  721  770  816  847  889  914  939    8
   .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .
 9    8  102  188  278  367  451  528  570  650  676  738  753  807  816  862    9
   .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .
10   10  141  253  348  430  497  545  574  626  654  701  723  764  782  814   10
   .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .
11   12  166  294  395  473  528  555  550  602  609  663  671  721  726  774   11
   .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .
12   13  181  320  425  503  555  579  584  616  631  668  684  720  734  765   12
   .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .
13   14  191  336  446  525  576  596  584  625  625  671  673  718  719  762   13
   .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .
14   15  197  347  460  541  596  628  642  669  685  713  729  755  768  787   14
   .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .
15   15  200  352  466  550  608  646  671  695  716  739  758  778  793  804   15
   .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .    .
     1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
VOLUMETRIC BUDGET FOR ENTIRE MODEL AT END OF TIME STEP  1 IN STRESS PERIOD  1
CUMULATIVE VOLUMES   L**3                          RATES FOR THIS TIME STEP   L**3/T
IN
                STORAGE =   0.0000E-01                          STORAGE =   0.0000E-01
          CONSTANT HEAD =   0.0000E-01                    CONSTANT HEAD =   0.0000E-01
                  WELLS =   0.0000E-01                            WELLS =   0.0000E-01
                 DRAINS =   0.0000E-01                           DRAINS =   0.0000E-01
              RECHARGE =   1.3608E+07                         RECHARGE =   1.5750E+02
```

TOTAL IN =	1.3608E+07	TOTAL IN =	1.5750E+02
OUT			
STORAGE =	0.0000E-01	STORAGE =	0.0000E-01
CONSTANT HEAD =	4.3265E+06	CONSTANT HEAD =	5.0075E+01
WELLS =	6.4800E+06	WELLS =	7.5000E+01
DRAINS =	2.8011E+06	DRAINS =	3.2420E+01
RECHARGE =	0.0000E-01	RECHARGE =	0.0000E-01
TOTAL OUT =	1.3608E+07	TOTAL OUT =	1.5750E+02
IN - OUT =	4.1200E+02	IN - OUT =	4.7607E-03
PERCENT DISCREPANCY =	3.0277E-03	PERCENT DISCREPANCY =	3.0227E-03

TIME SUMMARY AT END OF TIME STEP 1 IN STRESS PERIOD 1

SECONDS	MINUTES	HOURS	DAYS	YEARS	
TIME STEP LENGTH	86400.0	1440.00	24.0000	1.00000	0.273785E-02
STRESS PERIOD TIME	86400.0	1440.00	24.0000	1.00000	0.273785E-02
TOTAL SIMULATION TIME	86400.0	1440.00	24.0000	1.00000	0.273785E-02

---

discharge at cells

CONSTANT HEAD PERIOD	1 STEP	1	K	1 I	1 J	1 Q	-4.0290E+00
CONSTANT HEAD PERIOD	1 STEP	1	K	1 I	1 J	2 Q	-3.9434E+00
CONSTANT HEAD PERIOD	1 STEP	1	K	1 I	1 J	3 Q	-3.7719E+00
CONSTANT HEAD PERIOD	1 STEP	1	K	1 I	1 J	4 Q	-3.5112E+00
CONSTANT HEAD PERIOD	1 STEP	1	K	1 I	1 J	5 Q	-3.1422E+00
CONSTANT HEAD PERIOD	1 STEP	1	K	1 I	1 J	6 Q	-2.6050E+00
CONSTANT HEAD PERIOD	1 STEP	1	K	1 I	1 J	7 Q	-1.7967E+00
CONSTANT HEAD PERIOD	1 STEP	1	K	1 I	1 J	8 Q	-5.2837E-01
CONSTANT HEAD PERIOD	1 STEP	1	K	1 I	1 J	9 Q	-1.6352E+00
CONSTANT HEAD PERIOD	1 STEP	1	K	1 I	1 J	10 Q	-2.2956E+00
CONSTANT HEAD PERIOD	1 STEP	1	K	1 I	1 J	11 Q	-2.7056E+00
CONSTANT HEAD PERIOD	1 STEP	1	K	1 I	1 J	12 Q	-2.9661E+00
CONSTANT HEAD PERIOD	1 STEP	1	K	1 I	1 J	13 Q	-3.1324E+00
CONSTANT HEAD PERIOD	1 STEP	1	K	1 I	1 J	14 Q	-3.2332E+00
CONSTANT HEAD PERIOD	1 STEP	1	K	1 I	1 J	15 Q	-3.2814E+00
CONSTANT HEAD PERIOD	1 STEP	1	K	2 I	1 J	1 Q	-6.9673E-01
CONSTANT HEAD PERIOD	1 STEP	1	K	2 I	1 J	2 Q	-6.8267E-01
CONSTANT HEAD PERIOD	1 STEP	1	K	2 I	1 J	3 Q	-6.5439E-01

CONSTANT HEAD PERIOD	1 STEP	1			K	2 I	1 J	4 Q	-6.1095E-01
CONSTANT HEAD PERIOD	1 STEP	1			K	2 I	1 J	5 Q	-5.4858E-01
CONSTANT HEAD PERIOD	1 STEP	1			K	2 I	1 J	6 Q	-4.5674E-01
CONSTANT HEAD PERIOD	1 STEP	1			K	2 I	1 J	7 Q	-3.2066E-01
CONSTANT HEAD PERIOD	1 STEP	1			K	2 I	1 J	8 Q	-1.5036E-01
CONSTANT HEAD PERIOD	1 STEP	1			K	2 I	1 J	9 Q	-2.9240E-01
CONSTANT HEAD PERIOD	1 STEP	1			K	2 I	1 J	10 Q	-4.0372E-01
CONSTANT HEAD PERIOD	1 STEP	1			K	2 I	1 J	11 Q	-4.7479E-01
CONSTANT HEAD PERIOD	1 STEP	1			K	2 I	1 J	12 Q	-5.1965E-01
CONSTANT HEAD PERIOD	1 STEP	1			K	2 I	1 J	13 Q	-5.4804E-01
CONSTANT HEAD PERIOD	1 STEP	1			K	2 I	1 J	14 Q	-5.6513E-01
CONSTANT HEAD PERIOD	1 STEP	1			K	2 I	1 J	15 Q	-5.7326E-01
WELLS PERIOD	1 STEP	1	WELL	1	K	3 I	5 J	11 Q	-5.0000E+00
WELLS PERIOD	1 STEP	1	WELL	2	K	2 I	4 J	6 Q	-5.0000E+00
WELLS PERIOD	1 STEP	1	WELL	3	K	2 I	6 J	12 Q	-5.0000E+00
WELLS PERIOD	1 STEP	1	WELL	4	K	1 I	9 J	8 Q	-5.0000E+00
WELLS PERIOD	1 STEP	1	WELL	5	K	1 I	9 J	10 Q	-5.0000E+00
WELLS PERIOD	1 STEP	1	WELL	6	K	1 I	9 J	12 Q	-5.0000E+00
WELLS PERIOD	1 STEP	1	WELL	7	K	1 I	9 J	14 Q	-5.0000E+00
WELLS PERIOD	1 STEP	1	WELL	8	K	1 I	11 J	8 Q	-5.0000E+00
WELLS PERIOD	1 STEP	1	WELL	9	K	1 I	11 J	10 Q	-5.0000E+00
WELLS PERIOD	1 STEP	1	WELL	10	K	1 I	11 J	12 Q	-5.0000E+00
WELLS PERIOD	1 STEP	1	WELL	11	K	1 I	11 J	14 Q	-5.0000E+00
WELLS PERIOD	1 STEP	1	WELL	12	K	1 I	13 J	8 Q	-5.0000E+00
WELLS PERIOD	1 STEP	1	WELL	13	K	1 I	13 J	10 Q	-5.0000E+00
WELLS PERIOD	1 STEP	1	WELL	14	K	1 I	13 J	12 Q	-5.0000E+00
WELLS PERIOD	1 STEP	1	WELL	15	K	1 I	13 J	14 Q	-5.0000E+00
DRAINS PERIOD	1 STEP	1	DRAIN	1	K	1 I	8 J	2 Q	-3.4825E+00
DRAINS PERIOD	1 STEP	1	DRAIN	2	K	1 I	8 J	3 Q	-6.8321E+00
DRAINS PERIOD	1 STEP	1	DRAIN	3	K	1 I	8 J	4 Q	-6.2507E+00
DRAINS PERIOD	1 STEP	1	DRAIN	4	K	1 I	8 J	5 Q	-6.3012E+00
DRAINS PERIOD	1 STEP	1	DRAIN	5	K	1 I	8 J	6 Q	-6.9668E+00
DRAINS PERIOD	1 STEP	1	DRAIN	6	K	1 I	8 J	7 Q	-2.5866E+00
DRAINS PERIOD	1 STEP	1	DRAIN	7	K	1 I	8 J	8 Q	0.0000E-01
DRAINS PERIOD	1 STEP	1	DRAIN	8	K	1 I	8 J	9 Q	0.0000E-01
DRAINS PERIOD	1 STEP	1	DRAIN	9	K	1 I	8 J	10 Q	0.0000E-01

- PAGE 54 FOLLOWS -

APPENDIX 2. -- Sample problem with all features of modified model

input

bar

```
 24
  0        0        0       12       12
 1.
  1       14        1       15   8.0E-1
  2       14        2       15   8.0E-1
  3       14        3       15   8.0E-1
  4       14        4       15   8.0E-1
  4       14        5       14   8.0E-1
  5       13        5       14   8.0E-1
  6       13        6       14   8.0E-1
  7       13        7       14   8.0E-1
  8       13        8       14   8.0E-1
  9       13        9       14   8.0E-1
 10       13       10       14   8.0E-1
 11       13       11       14   8.0E-1
  1       13        1       14   8.0E-1
  2       13        2       14   8.0E-1
  3       13        3       14   8.0E-1
  4       13        4       14   8.0E-1
  4       13        5       13   8.0E-1
  5       12        5       13   8.0E-1
  6       12        6       13   8.0E-1
  7       12        7       13   8.0E-1
  8       12        8       13   8.0E-1
  9       12        9       13   8.0E-1
 10       12       10       13   8.0E-1
 11       12       11       13   8.0E-1
```

```
        5        11         19          1         1
29 30 31 32 33  0 35 36 37  0  0 40 41 42
        0         0                                         IAPART,ISTRT
        5              (24I3)                    -1      IBOUND1
 0  0  0  0  0  1  1  1  1  1  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  1  1  1  1  1  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  1  1  1  1  1  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  1  1  1  1  1  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  1  1  1  1  1  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  1  1  1  1  1  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  1  1  1  1  1  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  1  1  1  1  1  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  1  1  1  1  1  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  1  1  1  1  1  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  1  1  1  1  1  0  0  0  0  0  0  0  0  0
        5              (24I3)                    -1      IBOUND2
 1  1  1  1  1  1  1  1  1  0  0  0  0  0  0  0  0  0  0
 1  1  1  1  1  1  1  1  1  0  0  0  0  0  0  0  0  0  0
 1  1  1  1  1  1  1  1  1  0  0  0  0  0  0  0  0  0  0
 1  1  1  1  1  1  1  1  1  0  0  0  0  0  0  0  0  0  0
 1  1  1  1  1  1  1  1  1  0  0  0  0  0  0  0  0  0  0
 1  1  1  1  1  1  1  1  1  0  0  0  0  0  0  0  0  0  0
 1  1  1  1  1  1  1  1  1  0  0  0  0  0  0  0  0  0  0
 1  1  1  1  1  1  1  1  1  0  0  0  0  0  0  0  0  0  0
 1  1  1  1  1  1  1  1  1  0  0  0  0  0  0  0  0  0  0
 1  1  1  1  1  1  1  1  1  0  0  0  0  0  0  0  0  0  0
-1  1  1  1  1  1  1  1  1  0  0  0  0  0  0  0  0  0  0
        5              (24I3)                    -1      IBOUND3
 0  0  0  0  0  0  0  0  0  0  1  1  1  1  1  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  1  1  1  1  1  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  1  1  1  1  1  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  1  1  1  1  1  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  1  1  1  1  1  0  0  0  0
```

```
  0  0  0  0  0  0  0  0  0  0  1  1  1  1  1  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  1  1  1  1  1  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  1  1  1  1  1  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  1  1  1  1  1  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  1  1  1  1  1  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  1  1  1  1  1  0  0  0  0
         5           (24I3)               -1      IBOUND4
  0  0  0  0  0  0  0  0  1  1  1  1  1  1  1  1  1  0  0
  0  0  0  0  0  0  0  0  1  1  1  1  1  1  1  1  1  0  0
  0  0  0  0  0  0  0  0  1  1  1  1  1  1  1  1  1  0  0
  0  0  0  0  0  0  0  0  1  1  1  1  1  1  1  1  1  0  0
  0  0  0  0  0  0  0  0  1  1  1  1  1  1  1  1  1  0  0
  0  0  0  0  0  0  0  0  1  1  1  1  1  1  1  1  1  0  0
  0  0  0  0  0  0  0  0  1  1  1  1  1  1  1  1  1  0  0
  0  0  0  0  0  0  0  0  1  1  1  1  1  1  1  1  1  0  0
  0  0  0  0  0  0  0  0  1  1  1  1  1  1  1  1  1  0  0
  0  0  0  0  0  0  0  0  1  1  1  1  1  1  1  1  1  0  0
  0  0  0  0  0  0  0  0  1  1  1  1  1  1  1  1  1  0  0
         5           (24I3)               -1      IBOUND5
  0  0  0  0  0  0  1  1  1  1  1  1  1  1  1  1  1  1  1
  0  0  0  0  0  0  1  1  1  1  1  1  1  1  1  1  1  1  1
  0  0  0  0  0  0  1  1  1  1  1  1  1  1  1  1  1  1  1
  0  0  0  0  0  0  1  1  1  1  1  1  1  1  1  1  1  1  1
  0  0  0  0  0  0  1  1  1  1  1  1  1  1  1  1  1  1  1
  0  0  0  0  0  0  1  1  1  1  1  1  1  1  1  1  1  1  1
  0  0  0  0  0  0  1  1  1  1  1  1  1  1  1  1  1  1  1
  0  0  0  0  0  0  1  1  1  1  1  1  1  1  1  1  1  1  1
  0  0  0  0  0  0  1  1  1  1  1  1  1  1  1  1  1  1  1
  0  0  0  0  0  0  1  1  1  1  1  1  1  1  1  1  1  1  1
  0  0  0  0  0  0  1  1  1  1  1  1  1  1  1  1  1  1  1
         5       1.  (24F5.0)             -1      STRT1
    0    0    0    0    0  180  210  240  270  310    0    0    0    0    0    0    0    0    0
    0    0    0    0    0  180  210  240  270  310    0    0    0    0    0    0    0    0    0
    0    0    0    0    0  180  210  240  270  310    0    0    0    0    0    0    0    0    0
    0    0    0    0    0  180  210  240  270  310    0    0    0    0    0    0    0    0    0
    0    0    0    0    0  180  210  240  270  310    0    0    0    0    0    0    0    0    0
    0    0    0    0    0  180  210  240  270  310    0    0    0    0    0    0    0    0    0
    0    0    0    0    0  180  210  240  270  310    0    0    0    0    0    0    0    0    0
```

```
   0    0    0    0    0  180  210  240  270  310    0    0    0    0    0    0    0    0    0
   0    0    0    0    0  180  210  240  270  310    0    0    0    0    0    0    0    0    0
   0    0    0    0    0  180  210  240  270  310    0    0    0    0    0    0    0    0    0
   0    0    0    0    0  180  210  240  270  310    0    0    0    0    0    0    0    0    0
        5         1.   (24F5.0)                  -1        STRT2
 107  110  120  130  140  170  200  230  260    0    0    0    0    0    0    0    0    0    0
 105  110  120  130  140  170  200  230  260    0    0    0    0    0    0    0    0    0    0
 103  110  120  130  140  170  200  230  260    0    0    0    0    0    0    0    0    0    0
 101  110  120  130  140  170  200  230  260    0    0    0    0    0    0    0    0    0    0
  99  110  120  130  140  170  200  230  260    0    0    0    0    0    0    0    0    0    0
  97  110  120  130  140  170  200  230  260    0    0    0    0    0    0    0    0    0    0
  95  110  120  130  140  170  200  230  260    0    0    0    0    0    0    0    0    0    0
  94  110  120  130  140  170  200  230  260    0    0    0    0    0    0    0    0    0    0
  93  110  120  130  140  170  200  230  260    0    0    0    0    0    0    0    0    0    0
  92  110  120  130  140  170  200  230  260    0    0    0    0    0    0    0    0    0    0
  90  110  120  130  140  170  200  230  260    0    0    0    0    0    0    0    0    0    0
        5         1.   (24F5.0)                  -1        STRT3
   0    0    0    0    0    0    0    0    0    0  350  400  450  510  570    0    0    0    0
   0    0    0    0    0    0    0    0    0    0  350  400  450  510  570    0    0    0    0
   0    0    0    0    0    0    0    0    0    0  350  400  450  510  570    0    0    0    0
   0    0    0    0    0    0    0    0    0    0  350  400  450  510  570    0    0    0    0
   0    0    0    0    0    0    0    0    0    0  350  400  450  510  570    0    0    0    0
   0    0    0    0    0    0    0    0    0    0  350  400  450  510  570    0    0    0    0
   0    0    0    0    0    0    0    0    0    0  350  400  450  510  570    0    0    0    0
   0    0    0    0    0    0    0    0    0    0  350  400  450  510  570    0    0    0    0
   0    0    0    0    0    0    0    0    0    0  350  400  450  510  570    0    0    0    0
   0    0    0    0    0    0    0    0    0    0  350  400  450  510  570    0    0    0    0
   0    0    0    0    0    0    0    0    0    0  350  400  450  510  570    0    0    0    0
        5         1.   (24F5.0)                  -1        STRT4
   0    0    0    0    0    0    0    0  270  300  330  380  430  490  560  630  700    0    0
   0    0    0    0    0    0    0    0  270  300  330  380  430  490  560  630  700    0    0
   0    0    0    0    0    0    0    0  270  300  330  380  430  490  560  630  700    0    0
   0    0    0    0    0    0    0    0  270  300  330  380  430  490  560  630  700    0    0
   0    0    0    0    0    0    0    0  270  300  330  380  430  490  560  630  700    0    0
   0    0    0    0    0    0    0    0  270  300  330  380  430  490  560  630  700    0    0
   0    0    0    0    0    0    0    0  270  300  330  380  430  490  560  630  700    0    0
   0    0    0    0    0    0    0    0  270  300  330  380  430  490  560  630  700    0    0
   0    0    0    0    0    0    0    0  270  300  330  380  430  490  560  630  700    0    0
```

```
   0   0   0   0   0   0   0   0 270 300 330 380 430 490 560 630 700   0   0
   0   0   0   0   0   0   0   0 270 300 330 380 430 490 560 630 700   0   0
       5       1.  (24F5.0)               -1     STRT5
   0   0   0   0   0   0 210 240 260 280 310 350 400 460 520 580 650 710 760
   0   0   0   0   0   0 210 240 260 280 310 350 400 460 520 580 650 710 760
   0   0   0   0   0   0 210 240 260 280 310 350 400 460 520 580 650 710 760
   0   0   0   0   0   0 210 240 260 280 310 350 400 460 520 580 650 710 760
   0   0   0   0   0   0 210 240 260 280 310 350 400 460 520 580 650 710 760
   0   0   0   0   0   0 210 240 260 280 310 350 400 460 520 580 650 710 760
   0   0   0   0   0   0 210 240 260 280 310 350 400 460 520 580 650 710 760
   0   0   0   0   0   0 210 240 260 280 310 350 400 460 520 580 650 710 760
   0   0   0   0   0   0 210 240 260 280 310 350 400 460 520 580 650 710 760
   0   0   0   0   0   0 210 240 260 280 310 350 400 460 520 580 650 710 760
   0   0   0   0   0   0 210 240 260 280 310 350 400 460 520 580 650 710 760
        0.        0.        0.        0.        0.               CNSTNT(K)
   8.64E4          1       1.1                           PERLEN,NSTP,TSMULT
```

---

bcf

```
        1      -61         0                             ISS,IBCFBD,MCAN
 1 3 3 3 3
                1.                                     TRPY
             5.0E3                        -1           DROW
             5.0E3                        -1           DCOL
            1.0E-5                        -1           HY1
       29       1.  (24F5.0)              -1           BOT1
   0   0   0   0   0 155 165 175 185 225   0   0   0   0   0   0   0   0   0
   0   0   0   0   0 155 165 175 185 225   0   0   0   0   0   0   0   0   0
   0   0   0   0   0 155 165 175 185 225   0   0   0   0   0   0   0   0   0
   0   0   0   0   0 155 165 175 185 225   0   0   0   0   0   0   0   0   0
   0   0   0   0   0 155 165 175 185 225   0   0   0   0   0   0   0   0   0
   0   0   0   0   0 155 165 175 185 225   0   0   0   0   0   0   0   0   0
   0   0   0   0   0 155 165 175 185 225   0   0   0   0   0   0   0   0   0
   0   0   0   0   0 155 165 175 185 225   0   0   0   0   0   0   0   0   0
   0   0   0   0   0 155 165 175 185 225   0   0   0   0   0   0   0   0   0
   0   0   0   0   0 155 165 175 185 225   0   0   0   0   0   0   0   0   0
```

```
    0    0    0    0    0  155  165  175  185  225    0    0    0    0    0    0    0    0    0
          4.0E-9                                  -1        CV1
          2.0E-4                                  -1        HY2
   29          1.    (24F5.0)                     -1        BOT2
 -180 -165 -150 -135 -120 -105  -70  -20   85    0    0    0    0    0    0    0    0    0    0
 -180 -165 -150 -135 -120 -105  -70  -20   85    0    0    0    0    0    0    0    0    0    0
 -180 -165 -150 -135 -120 -105  -70  -20   85    0    0    0    0    0    0    0    0    0    0
 -180 -165 -150 -135 -120 -105  -70  -20   85    0    0    0    0    0    0    0    0    0    0
 -180 -165 -150 -135 -120 -105  -70  -20   85    0    0    0    0    0    0    0    0    0    0
 -180 -165 -150 -135 -120 -105  -70  -20   85    0    0    0    0    0    0    0    0    0    0
 -180 -165 -150 -135 -120 -105  -70  -20   85    0    0    0    0    0    0    0    0    0    0
 -180 -165 -150 -135 -120 -105  -70  -20   85    0    0    0    0    0    0    0    0    0    0
 -180 -165 -150 -135 -120 -105  -70  -20   85    0    0    0    0    0    0    0    0    0    0
 -180 -165 -150 -135 -120 -105  -70  -20   85    0    0    0    0    0    0    0    0    0    0
 -180 -165 -150 -135 -120 -105  -70  -20   85    0    0    0    0    0    0    0    0    0    0
          1.0E-8                                  -1        CV2
   29          1.    (24F5.0)                     -1        TOP2
  105  115  125  135  145  155  165  175  185    0    0    0    0    0    0    0    0    0    0
  105  115  125  135  145  155  165  175  185    0    0    0    0    0    0    0    0    0    0
  105  114  124  134  144  155  165  175  185    0    0    0    0    0    0    0    0    0    0
  105  115  125  135  145  155  165  175  185    0    0    0    0    0    0    0    0    0    0
  105  115  125  135  145  155  165  175  185    0    0    0    0    0    0    0    0    0    0
  105  115  125  135  145  155  165  175  185    0    0    0    0    0    0    0    0    0    0
  105  115  125  135  145  155  165  175  185    0    0    0    0    0    0    0    0    0    0
  105  115  125  135  145  155  165  175  185    0    0    0    0    0    0    0    0    0    0
  105  114  124  134  144  155  165  175  185    0    0    0    0    0    0    0    0    0    0
  105  115  125  135  145  155  165  175  185    0    0    0    0    0    0    0    0    0    0
  105  115  125  135  145  155  165  175  185    0    0    0    0    0    0    0    0    0    0
          5.0E-6                                  -1        HY3
   29          1.    (24F5.0)                     -1        BOT3
    0    0    0    0    0    0    0    0    0    0  260  290  350  450  550    0    0    0    0
    0    0    0    0    0    0    0    0    0    0  260  290  350  450  550    0    0    0    0
    0    0    0    0    0    0    0    0    0    0  260  290  350  450  550    0    0    0    0
    0    0    0    0    0    0    0    0    0    0  260  290  350  450  550    0    0    0    0
    0    0    0    0    0    0    0    0    0    0  260  290  350  450  550    0    0    0    0
    0    0    0    0    0    0    0    0    0    0  260  290  350  450  550    0    0    0    0
    0    0    0    0    0    0    0    0    0    0  260  290  350  450  550    0    0    0    0
    0    0    0    0    0    0    0    0    0    0  260  290  350  450  550    0    0    0    0
```

```
0    0    0    0    0    0    0    0    0    0  260  290  350  450  550    0    0    0    0
0    0    0    0    0    0    0    0    0    0  260  290  350  450  550    0    0    0    0
0    0    0    0    0    0    0    0    0    0  260  290  350  450  550    0    0    0    0
          2.0E-9                             -1        CV3
     29        1.   (24F5.0)                 -1        TOP3
0    0    0    0    0    0    0    0    0    0  340  400  460  520  580    0    0    0    0
0    0    0    0    0    0    0    0    0    0  340  400  460  520  580    0    0    0    0
0    0    0    0    0    0    0    0    0    0  330  390  450  510  570    0    0    0    0
0    0    0    0    0    0    0    0    0    0  340  400  460  520  580    0    0    0    0
0    0    0    0    0    0    0    0    0    0  340  400  460  520  580    0    0    0    0
0    0    0    0    0    0    0    0    0    0  340  400  460  520  580    0    0    0    0
0    0    0    0    0    0    0    0    0    0  340  400  460  520  580    0    0    0    0
0    0    0    0    0    0    0    0    0    0  340  400  460  520  580    0    0    0    0
0    0    0    0    0    0    0    0    0    0  330  390  450  510  570    0    0    0    0
0    0    0    0    0    0    0    0    0    0  340  400  460  520  580    0    0    0    0
0    0    0    0    0    0    0    0    0    0  340  400  460  520  580    0    0    0    0
          6.0E-5                             -1        HY4
     29        1.   (24F5.0)                 -1        BOT4
0    0    0    0    0    0    0    0   30   80  130  180  230  310  430  550  670    0    0
0    0    0    0    0    0    0    0   30   80  130  180  230  310  430  550  670    0    0
0    0    0    0    0    0    0    0   30   80  130  180  230  310  430  550  670    0    0
0    0    0    0    0    0    0    0   30   80  130  180  230  310  430  550  670    0    0
0    0    0    0    0    0    0    0   30   80  130  180  230  310  430  550  670    0    0
0    0    0    0    0    0    0    0   30   80  130  180  230  310  430  550  670    0    0
0    0    0    0    0    0    0    0   30   80  130  180  230  310  430  550  670    0    0
0    0    0    0    0    0    0    0   30   80  130  180  230  310  430  550  670    0    0
0    0    0    0    0    0    0    0   30   80  130  180  230  310  430  550  670    0    0
0    0    0    0    0    0    0    0   30   80  130  180  230  310  430  550  670    0    0
0    0    0    0    0    0    0    0   30   80  130  180  230  310  430  550  670    0    0
          6.0E-9                             -1        CV4
     29        1.   (24F5.0)                 -1        TOP4
0    0    0    0    0    0    0    0   85  225  260  290  350  450  550  640  700    0    0
0    0    0    0    0    0    0    0   85  225  260  290  350  450  550  640  700    0    0
0    0    0    0    0    0    0    0   85  225  260  290  350  450  550  630  690    0    0
0    0    0    0    0    0    0    0   85  225  260  290  350  450  550  640  700    0    0
0    0    0    0    0    0    0    0   85  225  260  290  350  450  550  640  700    0    0
0    0    0    0    0    0    0    0   85  225  260  290  350  450  550  640  700    0    0
0    0    0    0    0    0    0    0   85  225  260  290  350  450  550  640  700    0    0
```

```
 0    0    0    0    0    0    0    0   85  225  260  290  350  450  550  640  700    0    0
 0    0    0    0    0    0    0    0   85  225  260  290  350  450  550  630  690    0    0
 0    0    0    0    0    0    0    0   85  225  260  290  350  450  550  640  700    0    0
 0    0    0    0    0    0    0    0   85  225  260  290  350  450  550  640  700    0    0
           1.0E-4                         -1        HY5
      29        1.   (24F5.0)             -1        BOT5
 0    0    0    0    0    0  -90  -75  -60  -45  -30  -15   30   80  130  180  230  280  330
 0    0    0    0    0    0  -90  -75  -60  -45  -30  -15   30   80  130  180  230  280  330
 0    0    0    0    0    0  -90  -75  -60  -45  -30  -15   30   80  130  180  230  280  330
 0    0    0    0    0    0  -90  -75  -60  -45  -30  -15   30   80  130  180  230  280  330
 0    0    0    0    0    0  -90  -75  -60  -45  -30  -15   30   80  130  180  230  280  330
 0    0    0    0    0    0  -90  -75  -60  -45  -30  -15   30   80  130  180  230  280  330
 0    0    0    0    0    0  -90  -75  -60  -45  -30  -15   30   80  130  180  230  280  330
 0    0    0    0    0    0  -90  -75  -60  -45  -30  -15   30   80  130  180  230  280  330
 0    0    0    0    0    0  -90  -75  -60  -45  -30  -15   30   80  130  180  230  280  330
 0    0    0    0    0    0  -90  -75  -60  -45  -30  -15   30   80  130  180  230  280  330
 0    0    0    0    0    0  -90  -75  -60  -45  -30  -15   30   80  130  180  230  280  330
      29        1.   (24F5.0)             -1        TOP5
 0    0    0    0    0    0  -70  -20   30   80  130  180  230  310  430  550  670  760  820
 0    0    0    0    0    0  -70  -20   30   80  130  180  230  310  430  550  670  760  820
 0    0    0    0    0    0  -70  -20   30   80  130  180  230  310  430  550  670  750  810
 0    0    0    0    0    0  -70  -20   30   80  130  180  230  310  430  550  670  760  820
 0    0    0    0    0    0  -70  -20   30   80  130  180  230  310  430  550  670  760  820
 0    0    0    0    0    0  -70  -20   30   80  130  180  230  310  430  550  670  760  820
 0    0    0    0    0    0  -70  -20   30   80  130  180  230  310  430  550  670  760  820
 0    0    0    0    0    0  -70  -20   30   80  130  180  230  310  430  550  670  760  820
 0    0    0    0    0    0  -70  -20   30   80  130  180  230  310  430  550  670  750  810
 0    0    0    0    0    0  -70  -20   30   80  130  180  230  310  430  550  670  760  820
 0    0    0    0    0    0  -70  -20   30   80  130  180  230  310  430  550  670  760  820
```

---

cut

```
      4      13      26      39
      4       0       0       0       0
      3       6       4       6
      4       6       4       7
```

```
      8       6       8       7
      8       6       9       6
```

---

drn

```
     30     -63                                    MXDRAI,IDRNBD
     30                                            ITMP (NDRAIN)
     1.
      1       3       6     155    1.0E-1
      1       3       7     168    1.0E-1
      1       3       8     185    1.0E-1
      1       3       9     210    1.0E-1
      1       3      10     260    1.0E-1
      1       9       6     155    1.0E-1
      1       9       7     165    1.0E-1
      1       9       8     185    1.0E-1
      1       9       9     210    1.0E-1
      1       9      10     260    1.0E-1
      2       3       6     152    2.0E-1
      2       9       6     142    2.0E-1
      3       3      11     310    1.0E-1
      3       3      12     370    1.0E-1
      3       3      13     430    1.0E-1
      3       3      14     490    1.0E-1
      3       3      15     550    1.0E-1
      3       9      11     310    1.0E-1
      3       9      12     370    1.0E-1
      3       9      13     430    1.0E-1
      3       9      14     490    1.0E-1
      3       9      15     550    1.0E-1
      4       3      16     610    1.0E-1
      4       3      17     680    1.0E-1
      4       9      16     610    1.0E-1
      4       9      17     680    1.0E-1
      5       3      18     730    1.0E-1
      5       3      19     810    1.0E-1
```

```
     5      9     18    730   1.0E-1
     5      9     19    810   1.0E-1
```

evt

```
     2     -65                                    NEVTOP,IEVTCB
     1      1      1      1                       INSURF,INEVTR,INEXDP,INIEVT
    33      1.   (24F5.0)                -1       SURF
 108  115  125  135  145  190  230  250  270  290  340  400  460  520  580  640  700  760  820
 107  115  125  135  145  190  230  250  270  290  340  400  460  520  580  640  700  760  820
 106  113  123  133  143  170  210  240  260  280  330  390  450  510  570  630  690  750  810
 105  115  125  135  145  190  230  250  270  290  340  400  460  520  580  640  700  760  820
 104  115  125  135  145  190  230  250  270  290  340  400  460  520  580  640  700  760  820
 103  115  125  135  145  190  230  250  270  290  340  400  460  520  580  640  700  760  820
 102  115  125  135  145  190  230  250  270  290  340  400  460  520  580  640  700  760  820
 101  110  120  130  140  180  230  250  270  290  340  400  460  520  580  640  700  760  820
 100  105  115  125  135  165  205  240  260  280  330  390  450  510  570  630  690  750  810
  99  110  120  130  140  180  230  250  270  290  340  400  460  520  580  640  700  760  820
  98  115  125  135  145  190  230  250  270  290  340  400  460  520  580  640  700  760  820
          1.58E-8                    -1       EVTR
               10                    -1       EXDP
    33           (40I3)              -1       IEVT
  2  2  2  2  2  1  1  1  1  1  3  3  3  3  3  4  4  5  5
  2  2  2  2  2  1  1  1  1  1  3  3  3  3  3  4  4  5  5
  2  2  2  2  2  1  1  1  1  1  3  3  3  3  3  4  4  5  5
  2  2  2  2  2  1  1  1  1  1  3  3  3  3  3  4  4  5  5
  2  2  2  2  2  1  1  1  1  1  3  3  3  3  3  4  4  5  5
  2  2  2  2  2  1  1  1  1  1  3  3  3  3  3  4  4  5  5
  2  2  2  2  2  1  1  1  1  1  3  3  3  3  3  4  4  5  5
  2  2  2  2  2  1  1  1  1  1  3  3  3  3  3  4  4  5  5
  2  2  2  2  2  1  1  1  1  1  3  3  3  3  3  4  4  5  5
  2  2  2  2  2  1  1  1  1  1  3  3  3  3  3  4  4  5  5
  2  2  2  2  2  1  1  1  1  1  3  3  3  3  3  4  4  5  5
```

ghb

```
     12      -67                                      MXBND,IGMBCB
     12                                               ITMP (NBOUND)
     1.
      1        1       6     197    1.0E-3
      1        1       7     225    1.0E-3
      1        1       8     249    1.0E-3
      1        1       9     284    1.0E-3
      1        1      10     368    1.0E-3
      2        1       2     126    1.0E-2
      2        1       3     142    1.0E-2
      2        1       4     157    1.0E-2
      2        1       5     174    1.0E-2
      3        1       1                0.
      4        1       1                0.
      5        1       1                0.
```

out

```
     13        0      58       0          IHEDFM,IDDNFM,IHEDUN,IDDNUN
      0        1       1       1          INCODE,IHDDFL,IBUDFL,ICBCFL
      1        0       1       0          Hdpr,Ddpr,Hdsv,Ddsv
```

rch

```
      3      -68                                      NRCHOP,IRCHCB
      1        0                                      INRECH,INIRCH
     36   2.64E-9   (24F5.0)                -1      QRE
 10  10  10  10  10  10  11  11  11  11  11  12  12  13  13  14  14  15  15
 10  10  10  10  10  10  11  11  11  11  11  12  12  13  13  14  14  15  15
 10  10  10  10  10  10  11  11  11  11  11  12  12  13  13  14  14  15  15
 10  10  10  10  10  10  11  11  11  11  11  12  12  13  13  14  14  15  15
 10  10  10  10  10  10  11  11  11  11  11  12  12  13  13  14  14  15  15
```

```
  10  10  10  10  10  10  11  11  11  11  11  12  12  13  13  14  14  15  15
  10  10  10  10  10  10  11  11  11  11  11  12  12  13  13  14  14  15  15
  10  10  10  10  10  10  11  11  11  11  11  12  12  13  13  14  14  15  15
  10  10  10  10  10  10  11  11  11  11  11  12  12  13  13  14  14  15  15
  10  10  10  10  10  10  11  11  11  11  11  12  12  13  13  14  14  15  15
  10  10  10  10  10  10  11  11  11  11  11  12  12  13  13  14  14  15  15
```

---

riv

```
   22   -64                                  MXRIVR,IRIVBD
   22                                        ITMP (NRIVER)
   1.
    1     1     1              0.
    2     1     1     106   4.0E-1    102
    2     2     1     104   4.0E-1    100
    2     3     1     102   5.0E-1     98
    2     4     1     100   5.0E-1     96
    2     5     1      98   5.0E-1     94
    2     6     1      96   5.0E-1     92
    2     7     1      94   5.0E-1     90
    2     8     1      93   6.0E-1     89
    2     9     1      92   6.0E-1     88
    2    10     1      91   6.0E-1     87
    2     3     2     112   2.0E-1    108
    2     3     3     122   2.0E-1    118
    2     3     4     132   2.0E-1    128
    2     3     5     142   2.0E-1    138
    2     9     2     104   3.0E-1    100
    2     9     3     114   3.0E-1    110
    2     9     4     124   3.0E-1    120
    2     9     5     134   3.0E-1    130
    3     1     1              0.
    4     1     1              0.
    5     1     1              0.
```

---

sip

```
      65        5                                                    MXITER,NPARM
      1.    1.0E-2        0    2.0E-3        1    1.E-19           ACCL,HCLOSE,IPCALC,WSEED,IPRSIP,TINY
```

wel

```
      13       -62                                                  MXWELL,IWELBD
      13                                                            ITMP (NWELLS)
      1.
       1        1        1        0.
       2        5        7   -1.5E+0
       2        5        8   -2.0E+0
       2        5        9   -1.5E+0
       2        6        7   -2.0E+0
       2        6        8   -2.5E+0
       2        6        9   -2.0E+0
       2        7        7   -1.5E+0
       2        7        8   -2.0E+0
       2        7        9   -1.5E+0
       3        1        1        0.
       4        1        1        0.
       5        1        1        0.
```

main print

```
U.S. GEOLOGICAL SURVEY MODULAR FINITE-DIFFERENCE GROUND-WATER MODEL

   5 LAYERS   11 ROWS   19 COLUMNS
  1 STRESS PERIOD(S) IN SIMULATION
MODEL TIME UNIT IS SECONDS
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24
 29  30  31  32  33   0  35  36  37   0   0  40  41  42   0   0   0   0   0   0   0   0   0   0
```

```
BCF WEL DRN RIV EVT     GHB RCH SIP         OUT BAR CUT
BAS, 30SEP1991, READ ON     5
ARRAYS RHS AND BUFF WILL SHARE MEMORY.
START HEAD WILL NOT BE SAVED -- DRAWDOWN CANNOT BE CALCULATED
    9246 ELEMENTS IN X ARRAY ARE USED BY BAS
    9246 ELEMENTS OF X ARRAY USED OUT OF   26000
BCF, 30SEP1991, READ ON    29
STEADY-STATE SIMULATION
NUMBER: ACTIVE NEIGHBORS INADEQUATE (MCANC):    0
CONSTANT HEAD CELL-BY-CELL FLOWS WILL BE PRINTED
LAYER  AQUIFER TYPE
       1        1
       2        3
       3        3
       4        3
       5        3
    2931 ELEMENTS IN X ARRAY ARE USED BY BCF
   12177 ELEMENTS OF X ARRAY USED OUT OF   26000
BAR, 30SEP1991, READ ON    41
MAXIMUM NUMBER: FLOW BARRIERS (MXBAR):     24
ELEMENTS IN BAR       120
ISMM1,LENX     12297     26000
CUT, 30SEP1991, READ ON    42
MAXIMUM NUMBER: CUTTERS (MXCUT) & IBOUND SIGNALS:
ROW-BRANCH (MCRW), COLUMN-BRANCH (MCCL), & BOTH BRANCHES (MBTH):
     4     13     26     39
ELEMENTS IN CUT        16
ISMM1,LENX     12313     26000
WEL, 30SEP1991, READ ON    30
MAXIMUM OF   13 WELLS
CELL-BY-CELL FLOWS WILL BE PRINTED WHEN ICBCFL NOT 0
      52 ELEMENTS IN X ARRAY ARE USED FOR WELLS
   12365 ELEMENTS OF X ARRAY USED OUT OF   26000
DRAIN, 30SEP1991, READ ON    31
MAXIMUM OF   30 DRAINS
CELL-BY-CELL FLOWS WILL BE PRINTED WHEN ICBCFL NOT 0
     150 ELEMENTS IN X ARRAY ARE USED FOR DRAINS
   12515 ELEMENTS OF X ARRAY USED OUT OF   26000
```

```
RCH, 30SEP1991, READ ON    36
OPTION 3 -- RECHARGE TO HIGHEST ACTIVE NODE IN EACH VERTICAL COLUMN
     209 ELEMENTS OF X ARRAY USED FOR RECHARGE
   12724 ELEMENTS OF X ARRAY USED OUT OF   26000
EVT, 30SEP1991, READ ON    33
OPTION 2 -- EVAPOTRANSPIRATION FROM ONE SPECIFIED NODE IN EACH VERTICAL COLUMN
     836 ELEMENTS OF X ARRAY USED FOR EVAPOTRANSPIRATION
   13560 ELEMENTS OF X ARRAY USED OUT OF   26000
RIV, 30SEP1991, READ ON    32
MAXIMUM OF   22 RIVER NODES
CELL-BY-CELL FLOWS WILL BE PRINTED
     132 ELEMENTS IN X ARRAY ARE USED FOR RIVERS
   13692 ELEMENTS OF X ARRAY USED OUT OF   26000
GHB, 30SEP1991, READ ON    35
MAXIMUM OF   12 HEAD-DEPENDENT BOUNDARY NODES
CELL-BY-CELL FLOW WILL BE PRINTED WHEN ICBCFL NOT 0
      60 ELEMENTS IN X ARRAY ARE USED FOR HEAD-DEPENDENT BOUNDARIES
   13752 ELEMENTS OF X ARRAY USED OUT OF   26000
SIP, 30SEP1991, READ ON    37
MAXIMUM OF  65 ITERATIONS ALLOWED FOR CLOSURE
 5 ITERATION PARAMETERS
    4445 ELEMENTS IN X ARRAY ARE USED BY SIP
   18197 ELEMENTS OF X ARRAY USED OUT OF   26000

         BOUNDARY ARRAY FOR LAYER  1 WILL BE READ ON UNIT  5 USING FORMAT:    (24I3)
         BOUNDARY ARRAY FOR LAYER  2 WILL BE READ ON UNIT  5 USING FORMAT:    (24I3)
         BOUNDARY ARRAY FOR LAYER  3 WILL BE READ ON UNIT  5 USING FORMAT:    (24I3)
         BOUNDARY ARRAY FOR LAYER  4 WILL BE READ ON UNIT  5 USING FORMAT:    (24I3)
         BOUNDARY ARRAY FOR LAYER  5 WILL BE READ ON UNIT  5 USING FORMAT:    (24I3)
           INITIAL HEAD FOR LAYER  1 WILL BE READ ON UNIT  5 USING FORMAT:    (24F5.0)
           INITIAL HEAD FOR LAYER  2 WILL BE READ ON UNIT  5 USING FORMAT:    (24F5.0)
           INITIAL HEAD FOR LAYER  3 WILL BE READ ON UNIT  5 USING FORMAT:    (24F5.0)
           INITIAL HEAD FOR LAYER  4 WILL BE READ ON UNIT  5 USING FORMAT:    (24F5.0)
           INITIAL HEAD FOR LAYER  5 WILL BE READ ON UNIT  5 USING FORMAT:    (24F5.0)
CONSTANT ADDED TO INPUT HEADS:
   K    CONSTANT
   1   0.000E-01
   2   0.000E-01
```

```
   3   0.000E-01
   4   0.000E-01
   5   0.000E-01
HEAD PRINT FORMAT IS FORMAT NUMBER  13    DRAWDOWN PRINT FORMAT IS FORMAT NUMBER   0
HEADS WILL BE SAVED ON UNIT 58    DRAWDOWNS WILL BE SAVED ON UNIT  0
OUTPUT CONTROL IS SPECIFIED EVERY TIME STEP
COLUMN TO ROW ANISOTROPY =   1.000000
                    DELR =   5000.000
                    DELC =   5000.000
   HYD. COND. ALONG ROWS =  0.1000000E-04 FOR LAYER  1
                  BOTTOM FOR LAYER  1 WILL BE READ ON UNIT 29 USING FORMAT:    (24F5.0)
VERT HYD COND /THICKNESS =  0.4000000E-08 FOR LAYER  1
   HYD. COND. ALONG ROWS =  0.2000000E-03 FOR LAYER  2
                  BOTTOM FOR LAYER  2 WILL BE READ ON UNIT 29 USING FORMAT:    (24F5.0)
VERT HYD COND /THICKNESS =  0.1000000E-07 FOR LAYER  2
                     TOP FOR LAYER  2 WILL BE READ ON UNIT 29 USING FORMAT:    (24F5.0)
   HYD. COND. ALONG ROWS =  0.5000000E-05 FOR LAYER  3
                  BOTTOM FOR LAYER  3 WILL BE READ ON UNIT 29 USING FORMAT:    (24F5.0)
VERT HYD COND /THICKNESS =  0.2000000E-08 FOR LAYER  3
                     TOP FOR LAYER  3 WILL BE READ ON UNIT 29 USING FORMAT:    (24F5.0)
   HYD. COND. ALONG ROWS =  0.6000000E-04 FOR LAYER  4
                  BOTTOM FOR LAYER  4 WILL BE READ ON UNIT 29 USING FORMAT:    (24F5.0)
VERT HYD COND /THICKNESS =  0.6000000E-08 FOR LAYER  4
                     TOP FOR LAYER  4 WILL BE READ ON UNIT 29 USING FORMAT:    (24F5.0)
   HYD. COND. ALONG ROWS =  0.1000000E-03 FOR LAYER  5
                  BOTTOM FOR LAYER  5 WILL BE READ ON UNIT 29 USING FORMAT:    (24F5.0)
                     TOP FOR LAYER  5 WILL BE READ ON UNIT 29 USING FORMAT:    (24F5.0)
NUMBER: FLOW BARRIERS:
       K  BARRIERS
       1         0
       2         0
       3         0
       4        12
       5        12
FCBAR   1.0000E+00
NUMBER:  CUTTERS:
       K  CUTTERS
       1         4
```

```
        2         0
        3         0
        4         0
        5         0
SOLUTION BY THE STRONGLY IMPLICIT PROCEDURE
MAXIMUM ITERATIONS ALLOWED FOR CLOSURE =       65
ACCELERATION PARAMETER =   1.0000E+00
HEAD CHANGE CRITERION FOR CLOSURE =   1.00000E-02
SIP HEAD CHANGE PRINTOUT INTERVAL =        1
TEST FOR ZERO-DIVIDE OR UNDERFLOW (TINY):   1.000E-19
   5 ITERATION PARAMETERS CALCULATED FROM SPECIFIED WSEED =   2.00000E-03
  0.00000E-01   7.88526E-01   9.55279E-01   9.90543E-01   9.98000E-01
STRESS PERIOD NO.   1, LENGTH =   86400.00
NUMBER OF TIME STEPS =     1
MULTIPLIER FOR DELT =     1.100
INITIAL TIME STEP SIZE =   86400.00
FCWEL   1.0000E+00
  13 WELLS
FCDRN   1.0000E+00
   30 DRAINS
              RECHARGE WILL BE READ ON UNIT 36 USING FORMAT:    (24F5.0)
            ET SURFACE WILL BE READ ON UNIT 33 USING FORMAT:    (24F5.0)
 EVAPOTRANSPIRATION RATE =  0.1580000E-07
       EXTINCTION DEPTH =   10.00000
         ET LAYER INDEX WILL BE READ ON UNIT 33 USING FORMAT:    (40I3)
FCRVR   1.0000E+00
   22 RIVER REACHES
FCGHB   1.0000E+00
   12 HEAD-DEPENDENT BOUNDARY NODES
HEAD/DRAWDOWN PRINTOUT FLAG = 1     TOTAL BUDGET PRINTOUT FLAG = 1     CELL-BY-CELL FLOW TERM FLAG = 1
OUTPUT FLAGS FOR ALL LAYERS ARE THE SAME:
  HEAD     DRAWDOWN  HEAD  DRAWDOWN
PRINTOUT  PRINTOUT  SAVE    SAVE
    1         0       1       0
NMIT   K   I   J     MAX CHNG   CMLTV CHNG
   1   5   1  14   5.4375E+01   6.5598E+03
   2   3   8  12   2.3949E+01   3.9262E+03
   3   3   6  12   5.3572E+01   9.5021E+03
```

```
   4   3   7  12   3.7715E+01   4.0877E+03
   5   3   6  15   2.2793E+01   3.4473E+03
   6   1   6  10   4.1027E+00   3.7360E+02
   7   4   1  16   4.0855E+00   3.2238E+02
   8   4   2  15   1.1068E+00   1.2556E+02
   9   3   6  15   1.5693E+00   1.8310E+02
  10   3   7  13   7.5716E-01   7.4532E+01
  11   4   1  16   1.8998E-01   1.7260E+01
  12   4   7  12  -7.4229E-02   6.8810E+00
  13   4   5  17   7.6272E-02   8.4032E+00
  14   3   7  11  -3.6338E-02   2.5504E+00
  15   5   8  12   2.2039E-02   1.8061E+00
  16   1   7  10  -5.0673E-03   3.1050E-01
  16 ITERATIONS FOR TIME STEP   1 IN STRESS PERIOD  1
MAXIMUM HEAD CHANGE FOR EACH ITERATION:
HEAD CHANGE LAYER,ROW,COL HEAD CHANGE LAYER,ROW,COL HEAD CHANGE LAYER,ROW,COL HEAD CHANGE LAYER,ROW,COL HEAD CHANGE LAYER,ROW,COL
   54.38     (  5,  1, 14)   23.95     (  3,  8, 12)   53.57     (  3,  6, 12)   37.72     (  3,  7, 12)   22.79     (  3,  6, 15)
   4.103     (  1,  6, 10)   4.086     (  4,  1, 16)   1.107     (  4,  2, 15)   1.569     (  3,  6, 15)  0.7572     (  3,  7, 13)
  0.1900     (  4,  1, 16) -0.7423E-01 (  4,  7, 12)  0.7627E-01 (  4,  5, 17) -0.3634E-01 (  3,  7, 11)  0.2204E-01 (  5,  8, 12)
 -0.5067E-02 (  1,  7, 10)
FLOW BETWEEN LAYERS
   K          TOPIN         TPOUT        TOTOPIN        BOTIN         BTOUT        TOBOTIN        TOTALIN
       1  0.00000E-01   0.00000E-01                  0.00000E-01   0.00000E-01
       2  0.00000E-01   0.00000E-01                  5.64497E+00  -2.19094E+01
       3  0.00000E-01   0.00000E-01                  0.00000E-01   0.00000E-01
       4  0.00000E-01   0.00000E-01                  3.95810E+00  -2.02600E+00
       5  0.00000E-01   0.00000E-01                  0.00000E-01   0.00000E-01
   1       0.00000E-01   0.00000E-01  0.00000E-01     9.60307E+00  -2.39354E+01  -1.43323E+01  -1.43323E+01
       1  2.19094E+01  -5.64497E+00                  0.00000E-01   0.00000E-01
       2  0.00000E-01   0.00000E-01                  0.00000E-01   0.00000E-01
       3  0.00000E-01   0.00000E-01                  0.00000E-01   0.00000E-01
       4  0.00000E-01   0.00000E-01                  1.14740E+01   0.00000E-01
       5  0.00000E-01   0.00000E-01                  3.09116E+00   0.00000E-01
   2       2.19094E+01  -5.64497E+00  1.62644E+01     1.45652E+01   0.00000E-01   1.45652E+01   3.08296E+01
       1  0.00000E-01   0.00000E-01                  0.00000E-01   0.00000E-01
       2  0.00000E-01   0.00000E-01                  0.00000E-01   0.00000E-01
       3  0.00000E-01   0.00000E-01                  0.00000E-01   0.00000E-01
       4  0.00000E-01   0.00000E-01                  1.86532E+01  -1.78235E+01
```

```
      5  0.00000E-01   0.00000E-01                  0.00000E-01   0.00000E-01
  3      0.00000E-01   0.00000E-01   0.00000E-01    1.86532E+01  -1.78235E+01   8.29683E-01   8.29683E-01
      1  2.02600E+00  -3.95810E+00                  0.00000E-01   0.00000E-01
      2  0.00000E-01  -1.14740E+01                  0.00000E-01   0.00000E-01
      3  1.78235E+01  -1.86532E+01                  0.00000E-01   0.00000E-01
      4  0.00000E-01   0.00000E-01                  0.00000E-01   0.00000E-01
      5  0.00000E-01   0.00000E-01                  2.96797E+01  -1.69427E+01
  4      1.98495E+01  -3.40853E+01  -1.42358E+01    2.96797E+01  -1.69427E+01   1.27370E+01  -1.49883E+00
      1  0.00000E-01   0.00000E-01                  0.00000E-01   0.00000E-01
      2  0.00000E-01  -3.09116E+00                  0.00000E-01   0.00000E-01
      3  0.00000E-01   0.00000E-01                  0.00000E-01   0.00000E-01
      4  1.69427E+01  -2.96797E+01                  0.00000E-01   0.00000E-01
      5  0.00000E-01   0.00000E-01                  0.00000E-01   0.00000E-01
  5      1.69427E+01  -3.27708E+01  -1.58281E+01    0.00000E-01   0.00000E-01   0.00000E-01  -1.58281E+01
      HEAD IN LAYER  1 AT END OF TIME STEP  1 IN STRESS PERIOD  1
      HEAD    1 MULTIPLY BY     1.0E-01
     1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
 1   0   0   0   0   0 1969 2249 2489 2837 3678   0   0   0   0   0   0   0   0   0   1
   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
 2   0   0   0   0   0 1883 2181 2420 2708 3500   0   0   0   0   0   0   0   0   0   2
   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
 3   0   0   0   0   0 1621 1830 2025 2287 2839   0   0   0   0   0   0   0   0   0   3
   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
 4   0   0   0   0   0 1816 2000 2203 2559 3421   0   0   0   0   0   0   0   0   0   4
   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
 5   0   0   0   0   0 1816 1856 1980 2358 3534   0   0   0   0   0   0   0   0   0   5
   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
 6   0   0   0   0   0 1804 1779 1874 2250 3557   0   0   0   0   0   0   0   0   0   6
   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
 7   0   0   0   0   0 1801 1840 1970 2351 3530   0   0   0   0   0   0   0   0   0   7
```

```
 8   0   0   0   0   0 1750 1968 2185 2548 3414    0   0   0   0   0   0   0   0   0   8

 9   0   0   0   0   0 1582 1795 2016 2283 2837    0   0   0   0   0   0   0   0   0   9

10   0   0   0   0   0 1808 2140 2405 2696 3489    0   0   0   0   0   0   0   0   0  10

11   0   0   0   0   0 1907 2224 2474 2820 3664    0   0   0   0   0   0   0   0   0  11

     1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19

          HEAD IN LAYER  2 AT END OF TIME STEP  1 IN STRESS PERIOD  1
          HEAD    2 MULTIPLY BY     1.0E-01

     1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19

 1 1088 1256 1412 1569 1739 1941 2196 2450 2798    0   0   0   0   0   0   0   0   0   1

 2 1068 1223 1366 1508 1662 1849 2109 2356 2671    0   0   0   0   0   0   0   0   0   2

 3 1041 1159 1274 1388 1504 1634 1906 2125 2396    0   0   0   0   0   0   0   0   0   3

 4 1026 1191 1335 1469 1603 1757 1927 2130 2483    0   0   0   0   0   0   0   0   0   4

 5 1009 1199 1360 1504 1636 1757 1783 1906 2277    0   0   0   0   0   0   0   0   0   5

 6  991 1189 1357 1504 1634 1739 1706 1801 2170    0   0   0   0   0   0   0   0   0   6

 7  971 1166 1328 1474 1609 1735 1767 1896 2270    0   0   0   0   0   0   0   0   0   7
```

```
 8  954 1127 1272 1407 1544 1703 1895 2112 2472    0   0   0   0   0   0   0   0   0   0   8

 9  940 1066 1180 1292 1408 1561 1867 2107 2387    0   0   0   0   0   0   0   0   0   0   9

10  932 1109 1263 1413 1577 1780 2068 2335 2657    0   0   0   0   0   0   0   0   0   0  10

11  900 1126 1303 1474 1659 1879 2161 2430 2781    0   0   0   0   0   0   0   0   0   0  11

     1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19

          HEAD IN LAYER  3 AT END OF TIME STEP  1 IN STRESS PERIOD  1

          HEAD    3 MULTIPLY BY     1.0E-01

     1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19

 1   0   0   0   0   0   0   0   0   0   0 4348 4952 5523 6151 6645   0   0   0   0   1

 2   0   0   0   0   0   0   0   0   0   0 4164 4775 5357 6001 6515   0   0   0   0   2

 3   0   0   0   0   0   0   0   0   0   0 3320 3932 4533 5151 5728   0   0   0   0   3

 4   0   0   0   0   0   0   0   0   0   0 4145 4792 5404 6047 6559   0   0   0   0   4

 5   0   0   0   0   0   0   0   0   0   0 4328 4995 5667 6270 6732   0   0   0   0   5

 6   0   0   0   0   0   0   0   0   0   0 4380 5058 5741 6334 6788   0   0   0   0   6

 7   0   0   0   0   0   0   0   0   0   0 4323 4991 5674 6270 6730   0   0   0   0   7
```

```
 8   0   0   0   0   0   0   0   0   0   0   0 4135 4779 5456 6066 6550   0   0   0   0   8

 9   0   0   0   0   0   0   0   0   0   0   0 3317 3927 4548 5155 5726   0   0   0   0   9

10   0   0   0   0   0   0   0   0   0   0   0 4150 4760 5417 6020 6505   0   0   0   0  10

11   0   0   0   0   0   0   0   0   0   0   0 4332 4936 5587 6172 6636   0   0   0   0  11

     1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
          HEAD IN LAYER  4 AT END OF TIME STEP  1 IN STRESS PERIOD  1
          HEAD    4 MULTIPLY BY     1.0E-01
     1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19

 1   0   0   0   0   0   0   0   0 2829 3657 4278 4874 5446 6062 6555 6957 7320   0   0   1

 2   0   0   0   0   0   0   0   0 2701 3478 4097 4702 5285 5916 6426 6840 7242   0   0   2

 3   0   0   0   0   0   0   0   0 2439 3039 3677 4297 4904 5553 6091 6430 6979   0   0   3

 4   0   0   0   0   0   0   0   0 2514 3399 4077 4719 5332 5962 6470 6878 7272   0   0   4

 5   0   0   0   0   0   0   0   0 2336 3511 4256 4916 5592 6182 6641 7037 7383   0   0   5

 6   0   0   0   0   0   0   0   0 2240 3533 4308 4978 5666 6246 6698 7086 7420   0   0   6

 7   0   0   0   0   0   0   0   0 2329 3506 4252 4912 5599 6181 6639 7036 7383   0   0   7
```

8	0	0	0	0	0	0	0	0	2503	3391	4068	4705	5386	5981	6461	6875	7271	0	0	8
9	0	0	0	0	0	0	0	0	2430	3033	3669	4284	4949	5565	6082	6428	6978	0	0	9
10	0	0	0	0	0	0	0	0	2687	3466	4083	4685	5347	5935	6416	6835	7239	0	0	10
11	0	0	0	0	0	0	0	0	2813	3643	4262	4857	5511	6084	6545	6952	7317	0	0	11
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	

HEAD IN LAYER 5 AT END OF TIME STEP 1 IN STRESS PERIOD 1

HEAD 5 MULTIPLY BY 1.0E-01

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	
1	0	0	0	0	0	0	2198	2456	2858	3646	4270	4858	5423	6053	6535	6936	7292	7586	7779	1
2	0	0	0	0	0	0	2111	2360	2729	3469	4092	4690	5266	5911	6411	6824	7215	7545	7757	2
3	0	0	0	0	0	0	1910	2137	2485	3109	3759	4384	4983	5654	6186	6591	7064	7444	7732	3
4	0	0	0	0	0	0	1928	2134	2542	3389	4073	4707	5311	5959	6454	6862	7246	7565	7777	4
5	0	0	0	0	0	0	1785	1920	2395	3497	4249	4896	5583	6160	6624	7016	7355	7631	7819	5
6	0	0	0	0	0	0	1709	1817	2311	3518	4300	4958	5655	6224	6681	7065	7391	7655	7835	6
7	0	0	0	0	0	0	1769	1910	2388	3493	4244	4892	5588	6160	6623	7014	7355	7631	7819	7

```
 8   0   0   0   0   0   0 1897 2116 2532 3382 4063 4689 5380 5964 6449 6859 7244 7564 7776    8

 9   0   0   0   0   0   0 1871 2119 2476 3103 3749 4365 5047 5657 6180 6588 7063 7444 7731    9

10   0   0   0   0   0   0 2070 2340 2715 3458 4078 4669 5341 5919 6404 6820 7213 7544 7756   10

11   0   0   0   0   0   0 2164 2436 2842 3631 4253 4837 5500 6062 6528 6932 7289 7585 7778   11

     1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
HEAD WILL BE SAVED ON UNIT 58 AT END OF TIME STEP  1, STRESS PERIOD  1
VOLUMETRIC BUDGET FOR ENTIRE MODEL AT END OF TIME STEP  1 IN STRESS PERIOD  1
CUMULATIVE VOLUMES   L**3                          RATES FOR THIS TIME STEP   L**3/T
IN
           STORAGE =   0.0000E-01                           STORAGE =   0.0000E-01
     CONSTANT HEAD =   0.0000E-01                     CONSTANT HEAD =   0.0000E-01
             WELLS =   0.0000E-01                             WELLS =   0.0000E-01
            DRAINS =   0.0000E-01                            DRAINS =   0.0000E-01
          RECHARGE =   1.3931E+07                          RECHARGE =   1.6124E+02
                ET =   0.0000E-01                                ET =   0.0000E-01
     RIVER LEAKAGE =   0.0000E-01                     RIVER LEAKAGE =   0.0000E-01
   HEAD DEP BOUNDS =   1.2385E+03                   HEAD DEP BOUNDS =   1.4334E-02
          TOTAL IN =   1.3932E+07                          TOTAL IN =   1.6125E+02
OUT
           STORAGE =   0.0000E-01                           STORAGE =   0.0000E-01
     CONSTANT HEAD =   1.2220E+05                     CONSTANT HEAD =   1.4143E+00
             WELLS =   1.4256E+06                             WELLS =   1.6500E+01
            DRAINS =   4.9566E+06                            DRAINS =   5.7369E+01
          RECHARGE =   0.0000E-01                          RECHARGE =   0.0000E-01
                ET =   5.3914E+06                                ET =   6.2401E+01
     RIVER LEAKAGE =   2.0363E+06                     RIVER LEAKAGE =   2.3568E+01
   HEAD DEP BOUNDS =   0.0000E-01                   HEAD DEP BOUNDS =   0.0000E-01
         TOTAL OUT =   1.3932E+07                         TOTAL OUT =   1.6125E+02
          IN - OUT =  -9.2000E+01                          IN - OUT =  -1.0681E-03
```

PERCENT DISCREPANCY = -6.6035E-04 PERCENT DISCREPANCY = -6.6239E-04

TIME SUMMARY AT END OF TIME STEP 1 IN STRESS PERIOD 1

SECONDS	MINUTES	HOURS	DAYS	YEARS	
TIME STEP LENGTH	86400.0	1440.00	24.0000	1.00000	0.273785E-02
STRESS PERIOD TIME	86400.0	1440.00	24.0000	1.00000	0.273785E-02
TOTAL SIMULATION TIME	86400.0	1440.00	24.0000	1.00000	0.273785E-02

---

discharge at cells

CONSTANT HEAD PERIOD	1 STEP	1		K	2 I	1 J	11 Q	-1.4143E+00
WELLS PERIOD	1 STEP	1	WELL	1 K	1 I	1 J	1 Q	0.0000E-01
WELLS PERIOD	1 STEP	1	WELL	2 K	2 I	5 J	7 Q	-1.5000E+00
WELLS PERIOD	1 STEP	1	WELL	3 K	2 I	5 J	8 Q	-2.0000E+00
WELLS PERIOD	1 STEP	1	WELL	4 K	2 I	5 J	9 Q	-1.5000E+00
WELLS PERIOD	1 STEP	1	WELL	5 K	2 I	6 J	7 Q	-2.0000E+00
WELLS PERIOD	1 STEP	1	WELL	6 K	2 I	6 J	8 Q	-2.5000E+00
WELLS PERIOD	1 STEP	1	WELL	7 K	2 I	6 J	9 Q	-2.0000E+00
WELLS PERIOD	1 STEP	1	WELL	8 K	2 I	7 J	7 Q	-1.5000E+00
WELLS PERIOD	1 STEP	1	WELL	9 K	2 I	7 J	8 Q	-2.0000E+00
WELLS PERIOD	1 STEP	1	WELL	10 K	2 I	7 J	9 Q	-1.5000E+00
WELLS PERIOD	1 STEP	1	WELL	11 K	3 I	1 J	1 Q	0.0000E-01
WELLS PERIOD	1 STEP	1	WELL	12 K	4 I	1 J	1 Q	0.0000E-01
WELLS PERIOD	1 STEP	1	WELL	13 K	5 I	1 J	1 Q	0.0000E-01
DRAINS PERIOD	1 STEP	1	DRAIN	1 K	1 I	3 J	6 Q	-7.1175E-01
DRAINS PERIOD	1 STEP	1	DRAIN	2 K	1 I	3 J	7 Q	-1.4995E+00
DRAINS PERIOD	1 STEP	1	DRAIN	3 K	1 I	3 J	8 Q	-1.7503E+00
DRAINS PERIOD	1 STEP	1	DRAIN	4 K	1 I	3 J	9 Q	-1.8727E+00
DRAINS PERIOD	1 STEP	1	DRAIN	5 K	1 I	3 J	10 Q	-2.3941E+00
DRAINS PERIOD	1 STEP	1	DRAIN	6 K	1 I	9 J	6 Q	-3.2139E-01
DRAINS PERIOD	1 STEP	1	DRAIN	7 K	1 I	9 J	7 Q	-1.4534E+00
DRAINS PERIOD	1 STEP	1	DRAIN	8 K	1 I	9 J	8 Q	-1.6625E+00
DRAINS PERIOD	1 STEP	1	DRAIN	9 K	1 I	9 J	9 Q	-1.8279E+00
DRAINS PERIOD	1 STEP	1	DRAIN	10 K	1 I	9 J	10 Q	-2.3662E+00
DRAINS PERIOD	1 STEP	1	DRAIN	11 K	2 I	3 J	6 Q	-2.2838E+00
DRAINS PERIOD	1 STEP	1	DRAIN	12 K	2 I	9 J	6 Q	-2.8146E+00
DRAINS PERIOD	1 STEP	1	DRAIN	13 K	3 I	3 J	11 Q	-2.2019E+00

```
      DRAINS PERIOD   1 STEP   1 DRAIN   14 K   3 I   3 J   12 Q  -2.3183E+00
      DRAINS PERIOD   1 STEP   1 DRAIN   15 K   3 I   3 J   13 Q  -2.3332E+00
      DRAINS PERIOD   1 STEP   1 DRAIN   16 K   3 I   3 J   14 Q  -2.5119E+00
      DRAINS PERIOD   1 STEP   1 DRAIN   17 K   3 I   3 J   15 Q  -2.2845E+00
      DRAINS PERIOD   1 STEP   1 DRAIN   18 K   3 I   9 J   11 Q  -2.1738E+00
      DRAINS PERIOD   1 STEP   1 DRAIN   19 K   3 I   9 J   12 Q  -2.2728E+00
      DRAINS PERIOD   1 STEP   1 DRAIN   20 K   3 I   9 J   13 Q  -2.4841E+00
      DRAINS PERIOD   1 STEP   1 DRAIN   21 K   3 I   9 J   14 Q  -2.5548E+00
      DRAINS PERIOD   1 STEP   1 DRAIN   22 K   3 I   9 J   15 Q  -2.2554E+00
      DRAINS PERIOD   1 STEP   1 DRAIN   23 K   4 I   3 J   16 Q  -3.2969E+00
      DRAINS PERIOD   1 STEP   1 DRAIN   24 K   4 I   3 J   17 Q  -1.7858E+00
      DRAINS PERIOD   1 STEP   1 DRAIN   25 K   4 I   9 J   16 Q  -3.2788E+00
      DRAINS PERIOD   1 STEP   1 DRAIN   26 K   4 I   9 J   17 Q  -1.7782E+00
      DRAINS PERIOD   1 STEP   1 DRAIN   27 K   5 I   3 J   18 Q  -1.4425E+00
      DRAINS PERIOD   1 STEP   1 DRAIN   28 K   5 I   3 J   19 Q   0.0000E-01
      DRAINS PERIOD   1 STEP   1 DRAIN   29 K   5 I   9 J   18 Q  -1.4377E+00
      DRAINS PERIOD   1 STEP   1 DRAIN   30 K   5 I   9 J   19 Q   0.0000E-01
RIVER LEAKAGE PERIOD   1 STEP   1 REACH    1 K   1 I   1 J   1 Q   0.0000E-01
RIVER LEAKAGE PERIOD   1 STEP   1 REACH    2 K   2 I   1 J   1 Q  -1.1047E+00
RIVER LEAKAGE PERIOD   1 STEP   1 REACH    3 K   2 I   2 J   1 Q  -1.1138E+00
RIVER LEAKAGE PERIOD   1 STEP   1 REACH    4 K   2 I   3 J   1 Q  -1.0642E+00
RIVER LEAKAGE PERIOD   1 STEP   1 REACH    5 K   2 I   4 J   1 Q  -1.2843E+00
RIVER LEAKAGE PERIOD   1 STEP   1 REACH    6 K   2 I   5 J   1 Q  -1.4444E+00
RIVER LEAKAGE PERIOD   1 STEP   1 REACH    7 K   2 I   6 J   1 Q  -1.5275E+00
RIVER LEAKAGE PERIOD   1 STEP   1 REACH    8 K   2 I   7 J   1 Q  -1.5566E+00
RIVER LEAKAGE PERIOD   1 STEP   1 REACH    9 K   2 I   8 J   1 Q  -1.4553E+00
RIVER LEAKAGE PERIOD   1 STEP   1 REACH   10 K   2 I   9 J   1 Q  -1.2175E+00
RIVER LEAKAGE PERIOD   1 STEP   1 REACH   11 K   2 I  10 J   1 Q  -1.3336E+00
RIVER LEAKAGE PERIOD   1 STEP   1 REACH   12 K   2 I   3 J   2 Q  -7.7862E-01
RIVER LEAKAGE PERIOD   1 STEP   1 REACH   13 K   2 I   3 J   3 Q  -1.0825E+00
RIVER LEAKAGE PERIOD   1 STEP   1 REACH   14 K   2 I   3 J   4 Q  -1.3546E+00
RIVER LEAKAGE PERIOD   1 STEP   1 REACH   15 K   2 I   3 J   5 Q  -1.6843E+00
RIVER LEAKAGE PERIOD   1 STEP   1 REACH   16 K   2 I   9 J   2 Q  -7.7020E-01
RIVER LEAKAGE PERIOD   1 STEP   1 REACH   17 K   2 I   9 J   3 Q  -1.1979E+00
RIVER LEAKAGE PERIOD   1 STEP   1 REACH   18 K   2 I   9 J   4 Q  -1.5501E+00
RIVER LEAKAGE PERIOD   1 STEP   1 REACH   19 K   2 I   9 J   5 Q  -2.0482E+00
RIVER LEAKAGE PERIOD   1 STEP   1 REACH   20 K   3 I   1 J   1 Q   0.0000E-01
RIVER LEAKAGE PERIOD   1 STEP   1 REACH   21 K   4 I   1 J   1 Q   0.0000E-01
```

RIVER LEAKAGE PERIOD	1	STEP	1	REACH	22	K	5	I	1 J	1 Q	0.0000E-01
HEAD DEP BOUNDS PERIOD	1	STEP	1	GHB	1	K	1	I	1 J	6 Q	1.4255E-04
HEAD DEP BOUNDS PERIOD	1	STEP	1	GHB	2	K	1	I	1 J	7 Q	8.2031E-05
HEAD DEP BOUNDS PERIOD	1	STEP	1	GHB	3	K	1	I	1 J	8 Q	1.3745E-04
HEAD DEP BOUNDS PERIOD	1	STEP	1	GHB	4	K	1	I	1 J	9 Q	3.2013E-04
HEAD DEP BOUNDS PERIOD	1	STEP	1	GHB	5	K	1	I	1 J	10 Q	2.1552E-04
HEAD DEP BOUNDS PERIOD	1	STEP	1	GHB	6	K	2	I	1 J	2 Q	3.8142E-03
HEAD DEP BOUNDS PERIOD	1	STEP	1	GHB	7	K	2	I	1 J	3 Q	7.5574E-03
HEAD DEP BOUNDS PERIOD	1	STEP	1	GHB	8	K	2	I	1 J	4 Q	1.4172E-03
HEAD DEP BOUNDS PERIOD	1	STEP	1	GHB	9	K	2	I	1 J	5 Q	6.4789E-04
HEAD DEP BOUNDS PERIOD	1	STEP	1	GHB	10	K	3	I	1 J	1 Q	0.0000E-01
HEAD DEP BOUNDS PERIOD	1	STEP	1	GHB	11	K	4	I	1 J	1 Q	0.0000E-01
HEAD DEP BOUNDS PERIOD	1	STEP	1	GHB	12	K	5	I	1 J	1 Q	0.0000E-01

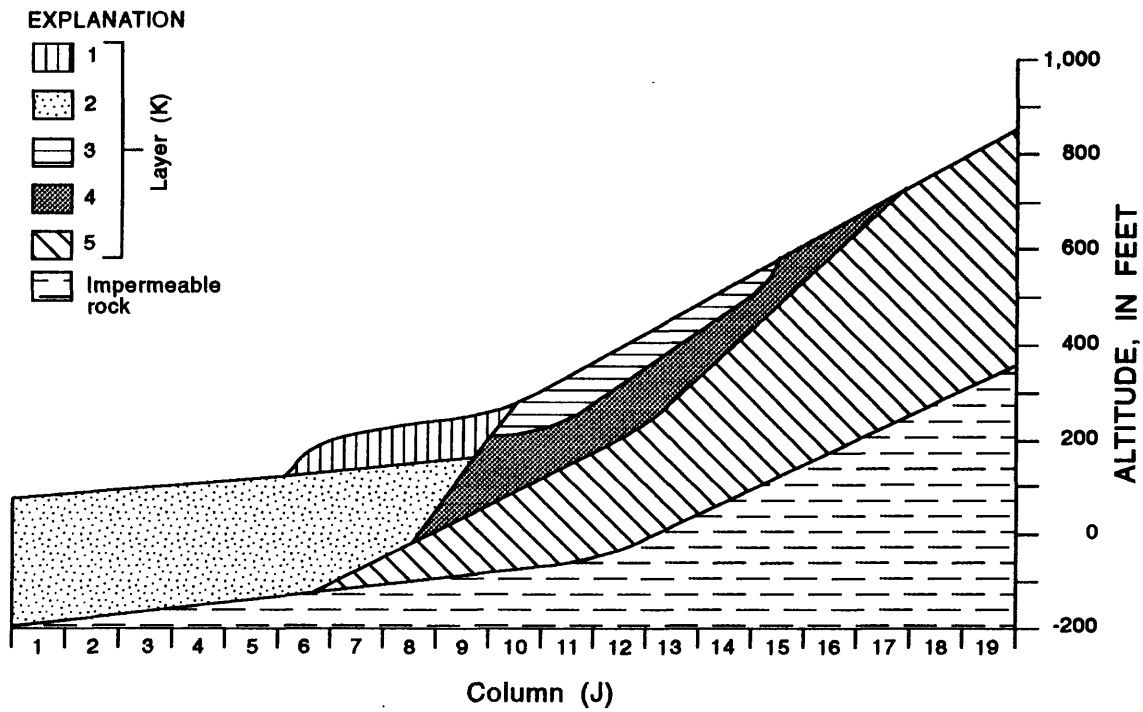


Figure A2a.--Cross-section view of the sample problem in Appendix 2.

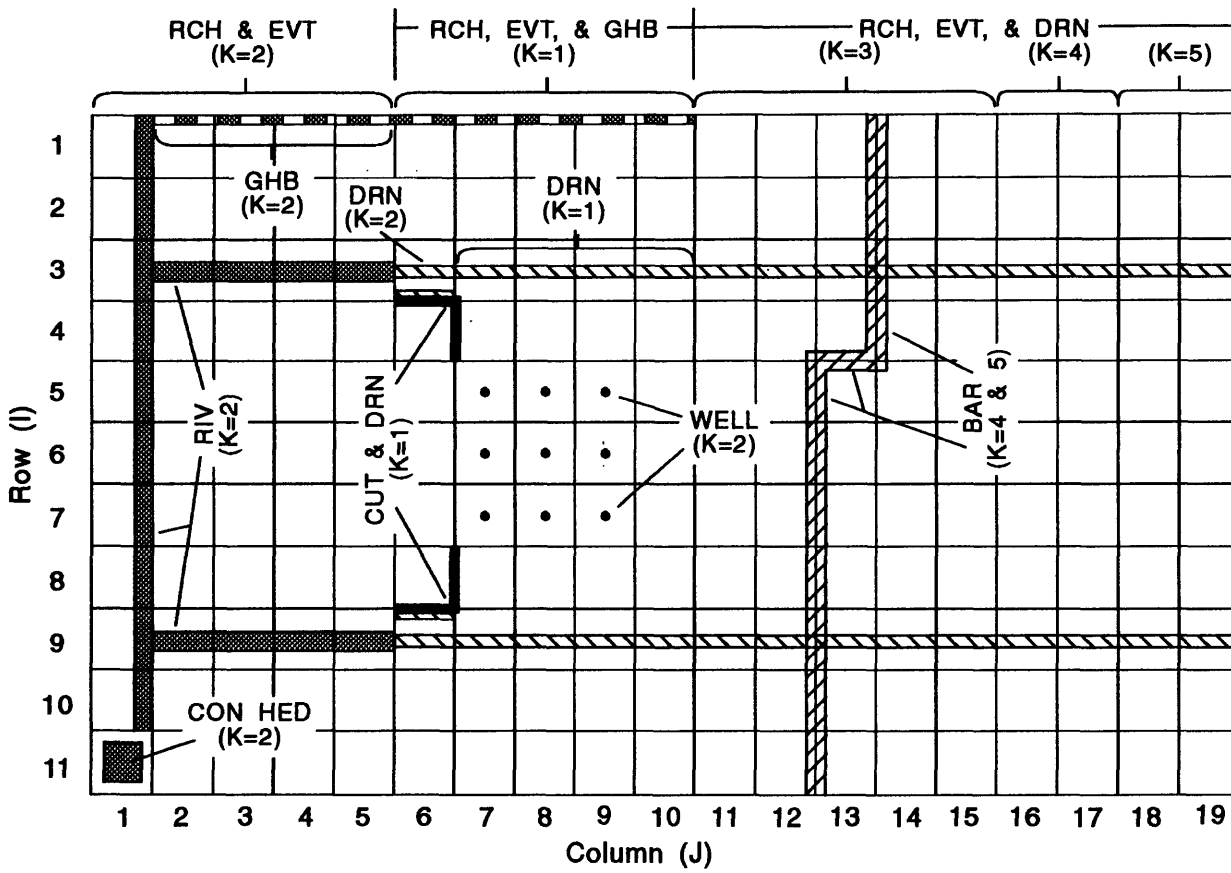


Figure A2b.--Map view of the sample problem in Appendix 2.

APPENDIX 3.-- Source code of modified model

```
c     main code for modular model
c     by michael g. mcdonald and arlen w. harbaugh
c     30sep1991 main2
c     specifications:
      common x(26000)
      common/barcom/laybar(80)
      common/cutcom/laycut(80)
      common/flwcom/laycon(80)
      character*4 headng,vbnm
      dimension headng(32),vbnm(4,20),vbvl(4,20),iunit(24)
      double precision dummy
      equivalence (dummy,x(1))
c     set size of x array. remember to redimension x.
      lenx=26000
c     assign basic input unit and printer unit.
      inbas=5
      iout=6
c     define problem rows,columns,layers,stress periods,packages
      call bas2df(isum,headng,nper,itmuni,totim,ncol,nrow,nlay,
     . klml,nodes,ndsml,inbas,iout,iunit)
c     allocate space in "x" array.
      call bas2al(isum,lenx,lchnew,lchold,lcibou,lccr,lccc,lccv,
     . lchcof,lcrhs,lcdelr,lcdelc,lcstrt,lcbuff,lciofl,
     . inbas,istrt,ncol,nrow,nlay,nodes,ndsml,iout)
      if(iunit(1).gt.0)call bcf2al(isum,lenx,lcscl,lchy,lcbot,
     . lctop,lcsc2,lctrpy,iunit(1),iss,ncol,nrow,nlay,ncnr,iout,
     . ibcfcb,mcanc)
      if(iunit(13).gt.0)call barlal(isum,lenx,lbar,mxbar,
     . iunit(13),iout)
      if(iunit(14).gt.0)call cutlal(isum,lenx,lcut,mxzer,
     . mcrw,mccl,mbth,iunit(14),iout)
      if(iunit(2).gt.0)call wellal(isum,lenx,lcwell,mxwell,nwells,
     . iunit(2),iout,iwelcb)
      if(iunit(3).gt.0)call drnlal(isum,lenx,lcdrai,ndrain,mxdrn,
     . iunit(3),iout,idrncb)
      if(iunit(8).gt.0)call rchlal(isum,lenx,lcirch,lcrech,nrchop,
     . ncol,nrow,iunit(8),iout,irchcb)
      if(iunit(5).gt.0)call evtlal(isum,lenx,lcievt,lcevtr,lcexdp,
     . lcsurf,ncol,nrow,nevtop,iunit(5),iout,ievtcb)
      if(iunit(4).gt.0)call rivlal(isum,lenx,lcrivr,mxrivr,nriver,
     . iunit(4),iout,irivcb)
      if(iunit(7).gt.0)call ghblal(isum,lenx,lcbnds,nbound,mxbnd,
     . iunit(7),iout,ighbcb)
      if(iunit(9).gt.0)call siplal(isum,lenx,lcel,lcfl,lcgl,lcv,
     . lchdcg,lclrch,lcw,mxiter,nparm,ncol,nrow,nlay,
     . iunit(9),iout)
c     if(iunit(11).gt.0)call sorlal(isum,lenx,lca,lcres,lchdcg,lclrch,
c     . lcieqp,mxiter,ncol,nlay,nslice,mbw,iunit(11),iout)
      if(iunit(9).gt.0)mbudl=mxiter-nparm
c     if the "x" array is not big enough then stop.
      if(isum-1.gt.lenx)stop
```

```
c     read and prepare information for entire simulation.
      call bas2rp(x(lcibou),x(lchnew),x(lcstrt),x(lchold),
     . istrt,inbas,headng,ncol,nrow,nlay,nodes,vbvl,x(lciofl),
     . iunit(12),ihedfm,iddnfm,ihedun,iddnun,iout)
      if(iunit(1).gt.0)call bcf2rp(x(lcibou),x(lchnew),x(lcscl),
     . x(lchy),x(lccr),x(lccc),x(lccv),x(lcdelr),
     . x(lcdelc),x(lcbot),x(lctop),x(lcsc2),x(lctrpy),
     . iunit(1),iss,ncol,nrow,nlay,klml,ncnr,nodes,ndsml,iout)
      if(iunit(13).gt.0)call barlrp(x(lccr),x(lccc),x(lbar),
     . mxbar,ncol,nrow,nlay,nodes,iunit(13),iout,kbrl,kbr2)
      if(iunit(14).gt.0)call cutlrp(x(lccr),x(lcp2),x(lcibou),
     . x(lcut),mxcut,ncol,nrow,nlay,nodes,iunit(14),iout,mcrw,
     . mccl,mbth,kctl,kct2)
      if(iunit(9).gt.0)call siplrp(nparm,mxiter,accl,hclose,x(lcw),
     . iunit(9),ipcalc,iprsip,iout,tiny)
c     if(iunit(11).gt.0)call sorlrp(mxiter,accl,hclose,iunit(11),
c     . iprsor,iout)
c     simulate each stress period.
      do 40 kper=1,nper
      kkper=kper
c     read stress period timing information.
      call baslst(nstp,delt,tsmult,pertim,kkper,inbas,iout)
c     read and prepare information for stress period.
      if(iunit(2).gt.0)call wellrp(x(lcwell),nwells,mxwell,iunit(2),
     . iout)
      if(iunit(3).gt.0)call drnlrp(x(lcdrai),ndrain,mxdrn,iunit(3),
     . iout)
      if(iunit(8).gt.0)call rchlrp(nrchop,x(lcirch),x(lcrech),
     . x(lcdelr),x(lcdelc),nrow,ncol,iunit(8),iout)
      if(iunit(5).gt.0)call evtlrp(nevtop,x(lcievt),x(lcevtr),
     . x(lcexdp),x(lcsurf),x(lcdelr),x(lcdelc),ncol,nrow,
     . iunit(5),iout)
      if(iunit(4).gt.0)call rivlrp(x(lcrivr),nriver,mxrivr,iunit(4),
     . iout)
      if(iunit(7).gt.0)call ghblrp(x(lcbnds),nbound,mxbnd,iunit(7),
     . iout)
c     simulate each time step.
      do 30 kstp=1,nstp
      kkstp=kstp
c     determine which output is needed.
      call basloc(nstp,kkstp,icnvg,x(lciofl),nlay,
     . ibudfl,icbcfl,ihddfl,iunit(12),iout)
c     calculate time step length. set hold=hnew..
      call baslad(delt,tsmult,totim,pertim,x(lchnew),x(lchold),kkstp,
     . ncol,nrow,nlay)
c     iteratively formulate and solve the equations.
      do 10 kiter=1,mxiter
      kkiter=kiter
c     formulate the finite difference equations.
      call baslfm(x(lchcof),x(lcrhs),nodes)
      if(iunit(1).gt.0)call bcf2fm(x(lchcof),x(lcrhs),x(lchold),
     . x(lcscl),x(lchnew),x(lcibou),x(lccr),x(lccc),x(lccv),
     . x(lchy),x(lctrpy),x(lcbot),x(lctop),x(lcsc2),
     . x(lcdelr),x(lcdelc),delt,iss,kkiter,kkstp,kkper,ncol,nrow,
```

```
     . nlay,klml,iout,mcanc,mcrw,mccl,mbth)
      if(iunit(13).gt.0)call barlmd(x(lccr),x(lccc),x(lbar),
     . mxbar,ncol,nrow,nlay,kbr1,kbr2)
      if(iunit(14).gt.0)call cutlmd(x(lccr),x(lccc),x(lcut),
     . mxcut,ncol,nrow,nlay,kct1,kct2)
      if(iunit(2).gt.0)call wellfm(nwells,mxwell,x(lcrhs),x(lcwell),
     . x(lcibou),ncol,nrow,nlay)
      if(iunit(3).gt.0)call drnlfm(ndrain,mxdrn,x(lcdrai),x(lchnew),
     . x(lchcof),x(lcrhs),x(lcibou),ncol,nrow,nlay)
      if(iunit(8).gt.0)call rchlfm(nrchop,x(lcirch),x(lcrech),
     . x(lcrhs),x(lcibou),ncol,nrow,nlay)
      if(iunit(5).gt.0)call evtlfm(nevtop,x(lcievt),x(lcevtr),
     . x(lcexdp),x(lcsurf),x(lcrhs),x(lchcof),x(lcibou),
     . x(lchnew),ncol,nrow,nlay)
      if(iunit(4).gt.0)call rivlfm(nriver,mxrivr,x(lcrivr),x(lchnew),
     . x(lchcof),x(lcrhs),x(lcibou),ncol,nrow,nlay)
      if(iunit(7).gt.0)call ghblfm(nbound,mxbnd,x(lcbnds),x(lchcof),
     . x(lcrhs),x(lcibou),ncol,nrow,nlay)
c     make one cut at an approximate solution.
      if(iunit(9).gt.0)call sip2ap(x(lchnew),x(lcibou),x(lccr),x(lccc),
     . x(lccv),x(lchcof),x(lcrhs),x(lcel),x(lcfl),x(lcgl),x(lcv),
     . x(lcw),x(lchdcg),x(lclrch),nparm,kkiter,hclose,accl,icnvg,
     . kkstp,kkper,ipcalc,iprsip,mxiter,nstp,ncol,nrow,nlay,klml,
     . nodes,ndsml,iout,tiny)
c     if(iunit(11).gt.0)call sorlap(x(lchnew),x(lcibou),x(lccr),
c     . x(lccc),x(lccv),x(lchcof),x(lcrhs),x(lca),x(lcres),x(lcieqp),
c     . x(lchdcg),x(lclrch),kkiter,hclose,accl,icnvg,kkstp,kkper,
c     . iprsor,mxiter,nstp,ncol,nrow,nlay,nslice,mbw,iout)
c     if convergence criterion has been met stop iterating.
      if(icnvg.eq.1)go to 20
      if(iunit(9).gt.0)then
       if(kiter.eq.mbudl)then
        mbudhd=0
c       calculate budget terms. save cell-by-cell flow terms.
        msum=1
        if(iunit(1).gt.0)call bcf2bd(vbnm,vbvl,msum,x(lchnew),
     .   x(lcibou),x(lchold),x(lcscl),x(lccr),x(lccc),x(lccv),
     .   x(lctop),x(lcsc2),delt,iss,ncol,nrow,nlay,klml,kkstp,
     .   kkper,ibcfcb,icbcfl,x(lcbuff),iout,mbudhd)
        if(iunit(2).gt.0)call wel2bd(nwells,mxwell,vbnm,vbvl,msum,
     .   x(lcwell),x(lcibou),delt,ncol,nrow,nlay,kkstp,kkper,iwelcb,
     .   icbcfl,x(lcbuff),iout,mbudhd)
        if(iunit(3).gt.0)call drn2bd(ndrain,mxdrn,vbnm,vbvl,msum,
     .   x(lcdrai),delt,x(lchnew),ncol,nrow,nlay,x(lcibou),kkstp,
     .   kkper,idrncb,icbcfl,x(lcbuff),iout,mbudhd)
        if(iunit(8).gt.0)call rchlbd(nrchop,x(lcirch),x(lcrech),
     .   x(lcibou),nrow,ncol,nlay,delt,vbvl,vbnm,msum,kkstp,kkper,
     .   irchcb,icbcfl,x(lcbuff),iout)
        if(iunit(5).gt.0)call evtlbd(nevtop,x(lcievt),x(lcevtr),
     .   x(lcexdp),x(lcsurf),x(lcibou),x(lchnew),ncol,nrow,nlay,
     .   delt,vbvl,vbnm,msum,kkstp,kkper,ievtcb,icbcfl,x(lcbuff),iout)
        if(iunit(4).gt.0)call riv2bd(nriver,mxrivr,x(lcrivr),x(lcibou),
     .   x(lchnew),ncol,nrow,nlay,delt,vbvl,vbnm,msum,
     .   kkstp,kkper,irivcb,icbcfl,x(lcbuff),iout,mbudhd)
```

```
        if(iunit(7).gt.0)call ghb2bd(nbound,mxbnd,vbnm,vbvl,msum,
     .   x(lcbnds),delt,x(lchnew),ncol,nrow,nlay,x(lcibou),kkstp,
     .   kkper,ighbcb,icbcfl,x(lcbuff),iout,mbudhd)
c       print and or save heads and drawdowns. print overall budget.
        call bas2ot(x(lchnew),x(lcstrt),istrt,x(lcbuff),x(lciofl),
     .   msum,x(lcibou),vbnm,vbvl,kkstp,kkper,delt,pertim,totim,
     .   itmuni,ncol,nrow,nlay,icnvg,ihddfl,ibudfl,
     .   ihedfm,ihedun,iddnfm,iddnun,iout,mbudhd)
        mbudhd=1
       end if
      end if
   10 continue
      kiter=mxiter
   20 mbudhd=1
c     calculate budget terms. save cell-by-cell flow terms.
      msum=1
      if(iunit(1).gt.0)call bcf2bd(vbnm,vbvl,msum,x(lchnew),
     . x(lcibou),x(lchold),x(lcscl),x(lccr),x(lccc),x(lccv),
     . x(lctop),x(lcsc2),delt,iss,ncol,nrow,nlay,klml,kkstp,
     . kkper,ibcfcb,icbcfl,x(lcbuff),iout,mbudhd)
      if(iunit(2).gt.0)call wel2bd(nwells,mxwell,vbnm,vbvl,msum,
     . x(lcwell),x(lcibou),delt,ncol,nrow,nlay,kkstp,kkper,iwelcb,
     . icbcfl,x(lcbuff),iout,mbudhd)
      if(iunit(3).gt.0)call drn2bd(ndrain,mxdrn,vbnm,vbvl,msum,
     . x(lcdrai),delt,x(lchnew),ncol,nrow,nlay,x(lcibou),kkstp,
     . kkper,idrncb,icbcfl,x(lcbuff),iout,mbudhd)
      if(iunit(8).gt.0)call rch1bd(nrchop,x(lcirch),x(lcrech),
     . x(lcibou),nrow,ncol,nlay,delt,vbvl,vbnm,msum,kkstp,kkper,
     . irchcb,icbcfl,x(lcbuff),iout)
      if(iunit(5).gt.0)call evt1bd(nevtop,x(lcievt),x(lcevtr),
     . x(lcexdp),x(lcsurf),x(lcibou),x(lchnew),ncol,nrow,nlay,
     . delt,vbvl,vbnm,msum,kkstp,kkper,ievtcb,icbcfl,x(lcbuff),iout)
      if(iunit(4).gt.0)call riv2bd(nriver,mxrivr,x(lcrivr),x(lcibou),
     . x(lchnew),ncol,nrow,nlay,delt,vbvl,vbnm,msum,
     . kkstp,kkper,irivcb,icbcfl,x(lcbuff),iout,mbudhd)
      if(iunit(7).gt.0)call ghb2bd(nbound,mxbnd,vbnm,vbvl,msum,
     . x(lcbnds),delt,x(lchnew),ncol,nrow,nlay,x(lcibou),kkstp,
     . kkper,ighbcb,icbcfl,x(lcbuff),iout,mbudhd)
c     print and or save heads and drawdowns. print overall budget.
      call bas2ot(x(lchnew),x(lcstrt),istrt,x(lcbuff),x(lciofl),
     . msum,x(lcibou),vbnm,vbvl,kkstp,kkper,delt,pertim,totim,
     . itmuni,ncol,nrow,nlay,icnvg,ihddfl,ibudfl,
     . ihedfm,ihedun,iddnfm,iddnun,iout,mbudhd)
c     if iteration failed to converge then stop.
      if(icnvg.eq.0)stop
   30 continue
   40 continue
      stop
      end
      subroutine barlal(isum,lenx,lbar,mxbar,in,iout)
c     30sep1991
c     allocates space
      write(iout,2)in
c     read number of flow barriers
```

```
      read(in,4)mxbar
      write(iout,6)mxbar
c     allocate space
      lbar=isum
      lmnts=5*mxbar
      isum=isum+lmnts
      write(iout,8)lmnts
      ismml=isum-1
      write(iout,12)ismml,lenx
      if(ismml.gt.lenx)write(iout,14)
    2 format(' bar, 30sep1991, read on',10i6)
    4 format(10i10)
    6 format(' maximum number: flow barriers (mxbar):',9i7)
    8 format(' elements in bar',10i10)
   12 format(' ismml,lenx',10i10)
   14 format(' x too small')
      return
      end
      subroutine barlmd(cr,cc,brrr,mxbar,ncol,nrow,nlay,kbr1,kbr2)
c     30sep1991
c     modifies branch conductance to account for flow barrier
      dimension cr(ncol,nrow,nlay),cc(ncol,nrow,nlay),brrr(5,mxbar)
      common/barcom/laybar(80)
      common/flwcom/laycon(80)
      nmbr=0
c     for each layer, modify branch conductance
      do 20 k=1,nlay
      if(laycon(k).ne.kbr1.and.laycon(k).ne.kbr2)then
c      signals have been set opposite to laycon
c      process layer, if: before iterations, constant transmissivity
c                         during iterations, variable transmissivity
       if(laybar(k).gt.0)then
c       if barriers, process
        do 10 m=1,laybar(k)
        nmbr=nmbr+1
        i=brrr(1,nmbr)
        j=brrr(2,nmbr)
        ipl=brrr(3,nmbr)
        jpl=brrr(4,nmbr)
        fctr=brrr(5,nmbr)
        if(j.ne.jpl)then
c        j.ne.jpl so barrier is on row-branch conductance
         if(cr(j,i,k).ne.0.)cr(j,i,k)=cr(j,i,k)*fctr
         go to 10
        end if
c       j.eq.jpl so barrier is on column-branch conductance
        if(cc(j,i,k).ne.0.)cc(j,i,k)=cc(j,i,k)*fctr
   10   continue
       end if
      end if
   20 continue
      return
      end
      subroutine barlrp(cr,cc,brrr,mxbar,ncol,nrow,nlay,nodes,in,
```

```
      iout,kbr1,kbr2)
c     30sep1991
c     reads & initializes
      dimension cr(nodes),cc(nodes),brrr(5,mxbar)
      common/barcom/laybar(80)
      common/flwcom/laycon(80)
c     for each layer, read number of flow barriers
      read(in,2)(laybar(k),k=1,nlay)
      write(iout,4)
      do 10 k=1,nlay
   10 write(iout,2)k,laybar(k)
      read(in,6)fall
      if(fall.le.0.)fall=1.
      write(6,8)fall
      write(60,12)
      nmbr=0
c     for each layer, read data
      do 30 k=1,nlay
      if(laybar(k).gt.0)then
c      if layer has barriers, process
       do 20 m=1,laybar(k)
       nmbr=nmbr+1
c      for each barrier, read node, forward node, & factor
c      either (ipl=i & jpl=(j+1)) or (ipl=(i+1) & jpl=j)
       read(in,14)i,j,ipl,jpl,fctr
       fctr=fctr*fall
       write(60,16)nmbr,i,j,ipl,jpl,fctr
       brrr(1,nmbr)=i
       brrr(2,nmbr)=j
       brrr(3,nmbr)=ipl
       brrr(4,nmbr)=jpl
       brrr(5,nmbr)=fctr
   20  continue
      end if
   30 continue
c     for constant-transmissivity, modify branch conductance &
c     set signals opposite to laycon
      kbr1=1
      kbr2=3
      call barlmd(cr,cc,brrr,mxbar,ncol,nrow,nlay,kbr1,kbr2)
c     for variable-transmissivity, set signals opposite to laycon
      kbr1=0
      kbr2=2
    2 format(10i10)
    4 format(' number: flow barriers:',/,'         k  barriers')
    6 format(10f10.0)
    8 format(' fcbar',1p8e13.4)
   12 format(' nmbr    i    j  ipl  jpl     factor')
   14 format(4i10,8f10.0)
   16 format(5i5,1p7e13.4)
      return
      end
      subroutine bas2df(isum,headng,nper,itmuni,totim,ncol,nrow,
     . nlay,klml,nodes,ndsml,inbas,iout,iunit)
```

```
c     30sep1991 bas2df
c     define key model parameters
c        specifications:
      character*4 headng
      dimension headng(32),iunit(24)
c1------print the name of the program.
      write(iout,1)
    1 format(' u.s. geological survey modular',
     1       ' finite-difference ground-water model')
c2------read and print a heading.
      read(inbas,2) headng
    2 format(20a4)
      write(iout,3) headng
    3 format(1x,32a4)
c3------read number of layers,rows,columns,stress periods and
c3------units of time code.
      read(inbas,4)nlay,nrow,ncol,nper,itmuni
      klm1=nlay-1
      nodes=ncol*nrow*nlay
      ndsm1=ncol*nrow*klm1
    4 format(8i10)
c4------print # of layers, rows, columns and stress periods.
      write(iout,5)nlay,nrow,ncol
    5 format(i5,' layers',i5,' rows',i5,' columns')
      write(iout,6) nper
    6 format(i4,' stress period(s) in simulation')
c5------select and print a message showing time units.
      if(itmuni.lt.0 .or. itmuni.gt.5) itmuni=0
      go to (10,20,30,40,50),itmuni
      write(iout,9)
    9 format(' model time units are undefined')
      go to 100
   10 write(iout,11)
   11 format(' model time unit is seconds')
      go to 100
   20 write(iout,21)
   21 format(' model time unit is minutes')
      go to 100
   30 write(iout,31)
   31 format(' model time unit is hours')
      go to 100
   40 write(iout,41)
   41 format(' model time unit is days')
      go to 100
   50 write(iout,51)
   51 format(' model time unit is years')
c6------read & print input unit numbers (iunit) for major options.
  100 read(inbas,101) iunit
      if(iunit(11).gt.0)then
       write(iout,12)
       stop
      end if
  101 format(24i3)
      write(iout,102)(i,i=1,24),iunit
```

```
  102 format(24i4,/,24i4,/,
     . ' bcf wel drn riv evt     ghb rch sip         out bar cut')
c7------initialize toal elapsed time counter storage array counter
c7------and calculate number of cells.
      totim=0.
      isum=1
   12 format(' you must use sip rather than sor!',/,' code iunit',
     . '(11) to zero, & set iunit(9) to input-unit number for sip')
      return
      end
      subroutine bas2al(isum,lenx,lchnew,lchold,lcibou,lccr,lccc,lccv,
     . lchcof,lcrhs,lcdelr,lcdelc,lcstrt,lcbuff,lciofl,inbas,istrt,
     . ncol,nrow,nlay,nodes,ndsml,iout)
c     30sep1991 bas2al
c     allocate space for basic model arrays
c        specifications:
c1------print a message identifying the package.
      write(iout,1)inbas
    1 format(' bas, 30sep1991, read on',1016)
c2------read & print flag iapart (rhs & buffer share space?) and
c2------flag istrt (should starting heads be saved for drawdown?)
      read(inbas,2) iapart,istrt
    2 format(2i10)
      if(iapart.eq.0) write(iout,3)
    3 format(' arrays rhs and buff will share memory.')
      if(istrt.ne.0) write(iout,4)
    4 format(' start head will be saved')
      if(istrt.eq.0) write(iout,5)
    5 format(' start head will not be saved',
     1       ' -- drawdown cannot be calculated')
c3------store,in isold, location of first unallocated space in x.
      isold=isum
c     nrc1=nrow*ncol*nlay
c4------allocate space for arrays.
      lchnew=isum
      isum=isum+2*nodes
      lchold=isum
      isum=isum+nodes
      lcibou=isum
      isum=isum+nodes
      lccr=isum
      isum=isum+nodes
      lccc=isum
      isum=isum+nodes
      lccv=isum
      isum=isum+ndsml
      lchcof=isum
      isum=isum+nodes
      lcrhs=isum
      isum=isum+nodes
      lcdelr=isum
      isum=isum+ncol
      lcdelc=isum
      isum=isum+nrow
```

```
      lciofl=isum
      isum=isum+nlay*4
c5------if buffer and rhs share space then lcbuff=lcrhs.
      lcbuff=lcrhs
      if(iapart.eq.0) go to 50
      lcbuff=isum
      isum=isum+nodes
c6------if strt will be saved then allocate space.
   50 lcstrt=isum
      if(istrt.ne.0)isum=isum+nodes
      isp=isum-isold
c7------print amount of space used.
      write(iout,6) isp
    6 format(i9,' elements in x array are used by bas')
      isuml=isum-1
      write(iout,7) isuml,lenx
    7 format(i9,' elements of x array used out of',i8)
      if(isuml.gt.lenx) write(iout,8)
    8 format(' ***x array must be dimensioned larger***')
      return
      end
      subroutine baslad(delt,tsmult,totim,pertim,hnew,hold,kstp,
     1                  ncol,nrow,nlay)
c-----version 1412 22feb1982 baslad
c     advance to next time step
c        specifications:
      double precision hnew
      dimension hnew(ncol,nrow,nlay), hold(ncol,nrow,nlay)
c1------if not first time step then calculate time step length.
      if(kstp.ne.1) delt=tsmult*delt
c2------accumulate elapsed time in simulation(totim) and in this
c2------stress period(pertim).
      totim=totim+delt
      pertim=pertim+delt
c3------copy hnew to hold.
      do 10 k=1,nlay
      do 10 i=1,nrow
      do 10 j=1,ncol
   10 hold(j,i,k)=hnew(j,i,k)
      return
      end
      subroutine baslfm(hcof,rhs,nodes)
c-----version 1632 24jul1987 baslfm
c     set hcof=rhs=0.
c        specifications:
      dimension hcof(nodes),rhs(nodes)
c1------for each cell initialize hcof and rhs accumulators.
      do 100 i=1,nodes
      hcof(i)=0.
      rhs(i)=0.
  100 continue
      return
      end
      subroutine basloc(nstp,kstp,icnvg,ioflg,nlay,
```

```
     .    ibudfl,icbcfl,ihddfl,inoc,iout)
c     30sep1991
c     output controller for head, drawdown, and budget
c        specifications:
      dimension ioflg(nlay,4)
c1------test unit number (inoc (inoc=iunit(12))) to see if
c1------output control is active.
      if(inoc.ne.0)go to 500
c2------if output control is inactive then set defaults and return.
      ihddfl=0
      if(icnvg.eq.0 .or. kstp.eq.nstp)ihddfl=1
      ibudfl=0
      if(icnvg.eq.0 .or. kstp.eq.nstp)ibudfl=1
      icbcfl=0
      return
c3------read and print output flags and code for defining ioflg.
  500 read(inoc,1) incode,ihddfl,ibudfl,icbcfl
    1 format(4i10)
      write(iout,3) ihddfl,ibudfl,icbcfl
    3 format(' head/drawdown printout flag =',i2,
     1    5x,'total budget printout flag =',i2,
     2    5x,'cell-by-cell flow term flag =',i2)
c4------decode incode to determine how to set flags in ioflg.
      if(incode) 100,200,300
c5------use ioflg from last time step.
  100 write(iout,101)
  101 format(' reusing previous values of ioflg')
      go to 600
c6------read ioflg for layer 1 and assign same to all layers
  200 read(inoc,201) (ioflg(1,m),m=1,4)
  201 format(4i10)
      do 210 k=1,nlay
      ioflg(k,1)=ioflg(1,1)
      ioflg(k,2)=ioflg(1,2)
      ioflg(k,3)=ioflg(1,3)
      ioflg(k,4)=ioflg(1,4)
  210 continue
      write(iout,211) (ioflg(1,m),m=1,4)
  211 format(' output flags for all layers are the same:',/,
     . '   head    drawdown  head  drawdown',/,
     . ' printout  printout  save    save',/,i6,i10,4i8)
      go to 600
c7------read ioflg in entirety
  300 read(inoc,301) ((ioflg(k,i),i=1,4),k=1,nlay)
  301 format(4i10)
      write(iout,302)
  302 format(' output flags for each layer:',/,
     . '          head    drawdown  head  drawdown',/,
     . ' layer  printout  printout  save    save')
      write(iout,303)(k,(ioflg(k,i),i=1,4),k=1,nlay)
  303 format(i4,i8,i10,4i8)
c8------the last step in a stress period and steps where iterative
c8------procedure failed to converge get a volumetric budget.
  600 if(icnvg.eq.0 .or. kstp.eq.nstp) ibudfl=1
```

```
      return
      end
      subroutine bas2ot(hnew,strt,istrt,buff,ioflg,msum,ibound,vbnm,
     . vbvl,kstp,kper,delt,pertim,totim,itmuni,ncol,nrow,nlay,icnvg,
     . ihddfl,ibudfl,ihedfm,ihedun,iddnfm,iddnun,iout,mbudhd)
c     30sepl991 bas2ot
c     output ibound, time, volumetric budget, head, & drawdown
c        specifications:
      character*4 vbnm
      double precision hnew
      dimension hnew(ncol,nrow,nlay),strt(ncol,nrow,nlay),
     . ibound(ncol,nrow,nlay),buff(ncol,nrow,nlay),
     . vbnm(4,20),vbvl(4,20),ioflg(nlay,4)
c1------clear printout flag (ipflg)
      ipflg=0
c2------if iterative procedure failed to converge print message
c     if(icnvg.eq.0) write(iout,1) kstp,kper
      if(mbudhd.eq.1.and.icnvg.eq.0)write(iout,1)kstp,kper
    1 format(' failed to converge in time step',i3,
     . ' of stress period',i3)
c3------if head and drawdown flag (ihddfl) is set write head and
c3------drawdown in accordance with flags in ioflg.
      if(mbudhd.eq.0)go to 100
c     write formatted ibound array, where dry cells set to zero
      call uwrib(ncol,nrow,nlay,ibound)
      if(ihddfl.eq.0)go to 100
      call sbaslh(hnew,buff,ioflg,kstp,kper,ncol,nrow,
     1    nlay,iout,ihedfm,ihedun,ipflg,pertim,totim)
      call sbasld(hnew,buff,ioflg,kstp,kper,ncol,nrow,nlay,iout,
     1 iddnfm,iddnun,strt,istrt,ibound,ipflg,pertim,totim)
c4------print total budget if requested
  100 if(ibudfl.eq.0) go to 120
      call sbaslv(msum,vbnm,vbvl,kstp,kper,iout)
      ipflg=1
c5------end printout with time summary and form feed if any printout
c5------will be produced.
  120 if(ipflg.eq.0) return
      call sbaslt(kstp,kper,delt,pertim,totim,itmuni,iout)
      return
      end
      subroutine bas2rp(ibound,hnew,strt,hold,istrt,inbas,
     .   headng,ncol,nrow,nlay,nodes,vbvl,ioflg,inoc,ihedfm,
     .   iddnfm,ihedun,iddnun,iout)
c     30sepl991 bas2rp
c     read and initialize basic model arrays
c        specifications:
      character*4 headng,aname
      double precision hnew
      dimension hnew(nodes),ibound(nodes),strt(nodes),hold(nodes),
     . aname(6,2),vbvl(4,20),ioflg(nlay,4),headng(32),cnstnt(80)
      data aname(1,1),aname(2,1),aname(3,1),aname(4,1),aname(5,1),
     1   aname(6,1) /'    ','    ','  bo','unda','ry a','rray'/
      data aname(1,2),aname(2,2),aname(3,2),aname(4,2),aname(5,2),
     1   aname(6,2) /'    ','    ','    ','init','ial ','head'/
```

```
c1------print simulation title, calculate # of cells in a layer.
      write(iout,1) headng
    1 format(1x,32a4)
      ncr=ncol*nrow
c2------read boundary array(ibound) one layer at a time.
      do 100 k=1,nlay
      kk=k
      loc=1+(k-1)*ncr
      call u2dint(ibound(loc),aname(1,1),nrow,ncol,kk,inbas,iout)
  100 continue
    2 format(10f10.0)
c4------read starting heads.
      do 300 k=1,nlay
      kk=k
      loc=1+(k-1)*ncr
      call u2drel(hold(loc),aname(1,2),nrow,ncol,kk,inbas,iout)
  300 continue
      read(inbas,2)(cnstnt(k),k=1,nlay)
      write(6,4)
      do 30 k=1,nlay
   30 write(6,6)k,cnstnt(k)
      mfir=1
      mlas=ncr
      do 50 k=1,nlay
      do 40 m=mfir,mlas
      if(ibound(m).le.0)then
       hnew(m)=0.
       if(ibound(m).lt.0)hnew(m)=hold(m)
       go to 40
      end if
      hnew(m)=hold(m)+cnstnt(k)
   40 continue
      mfir=mfir+ncr
      mlas=mlas+ncr
   50 continue
c6------if starting heads are to be saved then copy hold to strt.
      if(istrt.eq.0) go to 590
      do 500 i=1,nodes
      strt(i)=hold(i)
  500 continue
c7------initialize volumetric budget accumulators to zero.
  590 do 600 i=1,20
      do 600 j=1,4
      vbvl(j,i)=0.
  600 continue
c8------set up output control.
      call sbasli(nlay,istrt,ioflg,inoc,iout,ihedfm,
     1         iddnfm,ihedun,iddnun)
    4 format(' constant added to input heads:',/,'    k    constant')
    6 format(i5,1p10e12.3)
      return
      end
      subroutine baslst(nstp,delt,tsmult,pertim,kper,inbas,iout)
c     30sep1991
```

```
c     setup time parameters for new time period
c        specifications:
c1------read length of stress period, number of time steps and.
c1------time step multiplier.
      read (inbas,1) perlen,nstp,tsmult
    1 format(f10.0,i10,f10.0)
c2------calculate the length of the first time step.
c2a-----assume time step multiplier is equal to one.
      delt=perlen/float(nstp)
c2b-----if time step multiplier is not one then calculate first
c2b-----term of geometric progression.
      if(tsmult.ne.1.) delt=perlen*(1.-tsmult)/(1.-tsmult**nstp)
c3------print timing information.
      write (iout,2) kper,perlen,nstp,tsmult,delt
    2 format(' stress period no.',i4,', length =',g15.7,/,
     . ' number of time steps =',i6,/,
     . ' multiplier for delt =',f10.3,/,
     . ' initial time step size =',g15.7)
c4------initialize pertim (elapsed time within stress period).
      pertim=0.
      return
      end
      subroutine sbas1d(hnew,buff,ioflg,kstp,kper,ncol,nrow,
     1    nlay,iout,iddnfm,iddnun,strt,istrt,ibound,ipflg,
     2    pertim,totim)
c-----version 1630 15may1987 sbas1d
c     calculate print and record drawdowns
c        specifications
      character*4 text
      double precision hnew
      dimension hnew(ncol,nrow,nlay),ioflg(nlay,4),text(4),
     1     buff(ncol,nrow,nlay),strt(ncol,nrow,nlay),
     2     ibound(ncol,nrow,nlay)
      data text(1),text(2),text(3),text(4) /'    ','    ','draw',
     1       'down'/
c1------for each layer calculate drawdown if print or record
c1------is requested
      do 59 k=1,nlay
c2------is drawdown needed fro this layer?
      if(ioflg(k,2).eq.0 .and. ioflg(k,4).eq.0) go to 59
c3------drawdown is needed. were starting heads saved?
      if(istrt.ne.0) go to 53
c4------starting heads were not saved. print message and stop.
      write(iout,52)
   52 format(' cannot calculate drawdown because start',
     1   ' heads were not saved')
      stop
c5------calculate drawdown for the layer.
   53 do 58 i=1,nrow
      do 58 j=1,ncol
      hsing=hnew(j,i,k)
      buff(j,i,k)=hsing
      if(ibound(j,i,k).ne.0) buff(j,i,k)=strt(j,i,k)-hsing
   58 continue
```

```
   59 continue
c6------for each layer: determine if drawdown should be printed.
c6------if so then call ulaprs or ul2prw to print drawdown.
      do 69 k=1,nlay
      kk=k
      if(ioflg(k,2).eq.0) go to 69
      if(iddnfm.lt.0) call ulaprs(buff(1,1,k),text(1),kstp,kper,
     1             ncol,nrow,kk,-iddnfm,iout)
c     if(iddnfm.ge.0) call ulaprw(buff(1,1,k),text(1),kstp,kper,
      if(iddnfm.ge.0)call ul2prw(buff(1,1,k),text(1),kstp,kper,
     . ncol,nrow,kk,iddnfm,iout)
c    1             ncol,nrow,kk,iddnfm,iout)
      ipflg=1
   69 continue
c7------for each layer: determine if drawdown should be recorded.
c7------if so then call ulasav to record drawdown.
      ifirst=1
      if(iddnun.le.0) go to 80
      do 79 k=1,nlay
      kk=k
      if(ioflg(k,4).le.0) go to 79
      if(ifirst.eq.1) write(iout,74) iddnun,kstp,kper
   74 format(' drawdown will be saved on unit',i3,
     1    ' at end of time step',i3,', stress period',i3)
      ifirst=0
      call ulasav(buff(1,1,k),text(1),kstp,kper,pertim,totim,ncol,
     1              nrow,kk,iddnun)
   79 continue
   80 return
      end
      subroutine sbaslh(hnew,buff,ioflg,kstp,kper,ncol,nrow,
     .  nlay,iout,ihedfm,ihedun,ipflg,pertim,totim)
c     30sep1991
c     print and record heads
c        specifications
      character*4 text
      double precision hnew
      dimension hnew(ncol,nrow,nlay),ioflg(nlay,4),text(4),
     1     buff(ncol,nrow,nlay)
      data text(1),text(2),text(3),text(4) /'    ','    ','    ',
     1       'head'/
c1------for each layer: print head if requested.
      do 39 k=1,nlay
      kk=k
c2------test ioflg to see if head should be printed.
      if(ioflg(k,1).eq.0) go to 39
      ipflg=1
c3------copy heads for this layer into buffer.
      do 32 i=1,nrow
      do 32 j=1,ncol
      buff(j,i,1)=hnew(j,i,k)
   32 continue
c4------call utility module to print contents of buffer.
      if(ihedfm.lt.0) call ulaprs(buff,text(1),kstp,kper,ncol,nrow,kk,
```

```
     1          -ihedfm,iout)
c     if(ihedfm.ge.0) call ulaprw(buff,text(1),kstp,kper,ncol,nrow,kk,
      if(ihedfm.ge.0)call ul2prw(buff,text(1),kstp,kper,ncol,nrow,kk,
     . ihedfm,iout)
c    1            ihedfm,iout)
   39 continue
c5------if unit for recording heads <= 0: then return.
      if(ihedun.le.0)go to 50
      ifirst=1
c6------for each layer: record head if requested.
      do 49 k=1,nlay
      kk=k
c7------check ioflg to see if head for this layer should be recorded.
      if(ioflg(k,3).le.0) go to 49
      if(ifirst.eq.1) write(iout,41) ihedun,kstp,kper
   41 format(' head will be saved on unit',i3,' at end of time step',
     1    i3,', stress period',i3)
      ifirst=0
c8------copy heads for this layer into buffer.
      do 44 i=1,nrow
      do 44 j=1,ncol
      buff(j,i,1)=hnew(j,i,k)
   44 continue
c9------record contents of buffer on unit=ihedun
      call ulasav(buff,text(1),kstp,kper,pertim,totim,ncol,nrow,kk,
     1            ihedun)
   49 continue
   50 return
      end
      subroutine sbasli(nlay,istrt,ioflg,inoc,iout,ihedfm,
           iddnfm,ihedun,iddnun)
c     30sep1991
c     set up output control
c        specifications:
      dimension ioflg(nlay,4)
c1------test unit number from iunit (inoc) to see if output
c1------control is active.
      if(inoc.le.0) go to 600
c2------read and print formats for printing and unit numbers for
c2------recording heads and drawdown. then return.
  500 read (inoc,1)ihedfm,iddnfm,ihedun,iddnun
    1 format (4i10)
      write (iout,3)ihedfm,iddnfm
    3 format(' head print format is format number',i4,
     1       '    drawdown print format is format number',i4)
      write (iout,4)ihedun,iddnun
    4 format (' heads will be saved on unit',i3,
     1       '    drawdowns will be saved on unit',i3)
      write(iout,561)
  561 format(' output control is specified every time step')
      return
c3------output control is inactive. print a message listing defaults.
  600 write(iout,641)
  641 format(' default output control -- the following output',
```

```
     1      ' comes at the end of each stress period:')
       write(iout,642)
  642 format(' total volumetric budget')
       write(iout,643)
  643 format(' head')
       if(istrt.ne.0)write(iout,644)
  644 format(' drawdown')
c4------set the format codes equal to the default format.
       ihedfm=0
       iddnfm=0
c5------set default flags in ioflg so that head (and drawdown) is
c5------printed for every layer.
       id=0
       if(istrt.ne.0) id=1
  670 do 680 k=1,nlay
       ioflg(k,1)=1
       ioflg(k,2)=id
       ioflg(k,3)=0
       ioflg(k,4)=0
  680 continue
       return
       end
       subroutine sbaslt(kstp,kper,delt,pertim,totim,itmuni,iout)
c-----version 0837 09apr1982 sbaslt
c      print simulation time
c         specifications:
       write(iout,199) kstp,kper
  199 format(' time summary at end of time step',i3,
     1      ' in stress period',i3)
c1------use time unit indicator to get factor to convert to seconds.
       cnv=0.
       if(itmuni.eq.1) cnv=1.
       if(itmuni.eq.2) cnv=60.
       if(itmuni.eq.3) cnv=3600.
       if(itmuni.eq.4) cnv=86400.
       if(itmuni.eq.5) cnv=31557600.
c2------if factor=0 then time units are non-standard.
       if(cnv.ne.0.) go to 100
c2a-----print times in non-standard time units.
       write(iout,301) delt,pertim,totim
  301 format(' time step length =',g15.6,/,
     . ' stress period time =',g15.6,/,
     . ' total simulation time =',g15.6)
c2b-----return
       return
c3------calculate length of time step & elapsed times in seconds.
  100 delsec=cnv*delt
       totsec=cnv*totim
       persec=cnv*pertim
c4------calculate times in minutes,hours,days and years.
       delmn=delsec/60.
       delhr=delmn/60.
       deldy=delhr/24.
       delyr=deldy/365.25
```

```
      totmn=totsec/60.
      tothr=totmn/60.
      totdy=tothr/24.
      totyr=totdy/365.25
      permn=persec/60.
      perhr=permn/60.
      perdy=perhr/24.
      peryr=perdy/365.25
c5------print time step length and elapsed times in all time units.
      write(iout,200)
  200 format(' seconds        minutes         hours',
     . '          days           years')
      write (iout,201) delsec,delmn,delhr,deldy,delyr
  201 format('      time step length ',5g15.6)
      write(iout,202) persec,permn,perhr,perdy,peryr
  202 format('    stress period time ',5g15.6)
      write(iout,203) totsec,totmn,tothr,totdy,totyr
  203 format(' total simulation time ',5g15.6)
c6------return
      return
      end
      subroutine sbaslv(msum,vbnm,vbvl,kstp,kper,iout)
c     30sep1991
c     print volumetric budget
c     specifications:
      character*4 vbnm
      dimension vbnm(4,20),vbvl(4,20)
c1------determine number of individual budget entries.
      msum1=msum-1
      if(msum1.le.0) return
c2------clear rate and volume accumulators.
      totrin=0.
      totrot=0.
      totvin=0.
      totvot=0.
c3------add rates and volumes (in and out) to accumulators.
      do 100 l=1,msum1
      totrin=totrin+vbvl(3,1)
      totrot=totrot+vbvl(4,1)
      totvin=totvin+vbvl(1,1)
      totvot=totvot+vbvl(2,1)
  100 continue
c4------print time step number and stress period number.
      write(iout,260) kstp,kper
      write(iout,265)
c5------print individual inflow rates and volumes and their totals.
      do 200 l=1,msum1
      write(iout,275) (vbnm(i,1),i=1,4),vbvl(1,1),(vbnm(i,1),i=1,4)
     1,vbvl(3,1)
  200 continue
      write(iout,286) totvin,totrin
c6------print individual outflow rates and volumes and their totals.
      write(iout,287)
      do 250 l=1,msum1
```

```
      write(iout,275) (vbnm(i,1),i=1,4),vbvl(2,1),(vbnm(i,1),i=1,4)
     1,vbvl(4,1)
  250 continue
      write(iout,298) totvot,totrot
c7------calculate the difference between inflow and outflow.
c7a-----calculate difference between rate in and rate out.
      diffr=totrin-totrot
c7b-----calculate percent difference between rate in and rate out.
      pdiffr=100.*diffr/((totrin+totrot)/2)
c7c-----calculate difference between volume in and volume out.
      diffv=totvin-totvot
c7d-----get percent difference between volume in and volume out.
      pdiffv=100.*diffv/((totvin+totvot)/2)
c8------print differences and percent differences between input
c8------and output rates and volumes.
      write(iout,299) diffv,diffr
      write(iout,300) pdiffv,pdiffr
c     ---formats
  260 format(' volumetric budget for entire model at end of',
     . ' time step',i3,' in stress period',i3)
  265 format(' cumulative volumes',' l**3',23x,
     . 'rates for this time step',' l**3/t',/,' in')
  275 format(4x,4a4,' =',1pe13.4,16x,4a4,' =',1pe13.4)
  286 format(12x,'total in =',1pe13.4,24x,'total in =',1pe13.4)
  287 format(' out')
  298 format(11x,'total out =',1pe13.4,23x,'total out =',1pe13.4)
  299 format(12x,'in - out =',1pe13.4,24x,'in - out =',1pe13.4)
  300 format(' percent discrepancy =',1pe13.4,
     . 13x,'percent discrepancy =',1pe13.4)
      return
      end
      subroutine bcf2fm(hcof,rhs,hold,scl,hnew,ibound,cr,cc,cv,hy,
     . trpy,bot,top,sc2,delr,delc,delt,iss,kiter,kstp,kper,ncol,nrow,
     . nlay,klml,iout,mcanc,mcrw,mccl,mbth)
c     30sep1991 bcf2fm
c     add leakage correction and storage to hcof and rhs, and calculate
c     conductance as required
c        specifications:
      double precision hnew
      dimension hcof(ncol,nrow,nlay),rhs(ncol,nrow,nlay),
     . hold(ncol,nrow,nlay),scl(ncol,nrow,nlay),hnew(ncol,nrow,nlay),
     . ibound(ncol,nrow,nlay),cr(ncol,nrow,nlay),delr(ncol),
     . cc(ncol,nrow,nlay),cv(ncol,nrow,klml),hy(ncol,nrow,nlay),
     . trpy(nlay),bot(ncol,nrow,nlay),top(ncol,nrow,nlay),
     . delc(nrow),sc2(ncol,nrow,nlay)
      common/flwcom/laycon(80)
      kb=0
      kt=0
c1------for each layer: if t varies calculate horizontal conductances
      do 100 k=1,nlay
      kk=k
      if(laycon(k).eq.3 .or. laycon(k).eq.2) kt=kt+1
cla-----if layer type is not 1 or 3 then skip this layer.
      if(laycon(k).ne.3 .and. laycon(k).ne.1) go to 100
```

```
      kb=kb+1
c1b-----for layer types 1 & 3 call sbcf2h to calculate
c1b-----horizontal conductances
      call sbcf2h(hnew,ibound,cr,cc,cv,hy,trpy,delr,delc,bot,top,
     . kk,kb,kt,kiter,kstp,kper,ncol,nrow,nlay,klm1,iout,mcanc,
     . mcrw,mccl,mbth)
  100 continue
c2------if the simulation is transient add storage to hcof and rhs
      if(iss.ne.0) go to 201
      tled=1./delt
      kt=0
      do 200 k=1,nlay
c3------see if this layer is convertible or non-convertible.
      if(laycon(k).eq.3 .or. laycon(k).eq.2) go to 150
c4------non-convertible layer, so use primary storage
      do 140 i=1,nrow
      do 140 j=1,ncol
      if(ibound(j,i,k).le.0) go to 140
      rho=scl(j,i,k)*tled
      hcof(j,i,k)=hcof(j,i,k)-rho
      rhs(j,i,k)=rhs(j,i,k)-rho*hold(j,i,k)
  140 continue
      go to 200
c5------a convertible layer, so check old and new heads to determine
c5------when to use primary and secondary storage
  150 kt=kt+1
      do 180 i=1,nrow
      do 180 j=1,ncol
c5a-----if the cell is external then skip it.
      if(ibound(j,i,k).le.0) go to 180
      tp=top(j,i,kt)
      rho2=sc2(j,i,kt)*tled
      rho1=scl(j,i,k)*tled
c5b-----find storage factor at start of time step.
      sold=rho2
      if(hold(j,i,k).gt.tp) sold=rho1
c5c-----find storage factor at end of time step.
      htmp=hnew(j,i,k)
      snew=rho2
      if(htmp.gt.tp) snew=rho1
c5d-----add storage terms to rhs and hcof.
      hcof(j,i,k)=hcof(j,i,k)-snew
      rhs(j,i,k)=rhs(j,i,k) - sold*(hold(j,i,k)-tp) - snew*tp
  180 continue
  200 continue
c6------flow down into partially saturated layers.
  201 kt=0
      do 300 k=1,nlay
c7------see if correction is needed for leakage from above.
      if(laycon(k).ne.3 .and. laycon(k).ne.2) go to 250
      kt=kt+1
      if(k.eq.1) go to 250
c7a-----for each cell make the correction if needed.
      do 220 i=1,nrow
```

```
      do 220 j=1,ncol
c7b-----if the cell is external(ibound<=0) then skip it.
c7d-----with head below top add correction terms to rhs and hcof.
c     if cell is external, skip to next cell
      if(ibound(j,i,k).gt.0)then
       htmp=hnew(j,i,k)
c      if head below top, correction needed
       if(htmp.lt.top(j,i,kt))then
c       find first active cell above
        do 10 kl=1,nlay
        kup=k-kl
        if(kup.gt.0)then
         if(ibound(j,i,kup).ne.0)then
c         add correction to rhs
          rhs(j,i,k)=rhs(j,i,k)+cv(j,i,kup)*(top(j,i,kt)-htmp)
c         mcdonald (1989) replaced two lines below with one above
c         rhs(j,i,k)=rhs(j,i,k)+cv(j,i,kup)*top(j,i,kt)
c         hcof(j,i,k)=hcof(j,i,k)+cv(j,i,kup)
          go to 220
         end if
        end if
   10   continue
       end if
      end if
  220 continue
c8------see if this layer may need correction for leakage to below.
  250 if(k.eq.nlay) go to 300
c8a-----for each cell make the correction if needed.
      do 280 i=1,nrow
      do 280 j=1,ncol
c8b-----if cell is external (ibound<=0) then skip it.
c8c-----if head in the lower cell is less than top add correction
c8c-----term to rhs.
      if(ibound(j,i,k).gt.0)then
       ktlo=kt
c      find first active cell below
       do 20 kl=1,nlay
       klo=k+kl
       if(klo.le.nlay)then
        if(laycon(klo).gt.1)ktlo=ktlo+1
        if(ibound(j,i,klo).ne.0)then
         if(laycon(klo).gt.1)then
          htmp=hnew(j,i,klo)
          if(htmp.lt.top(j,i,ktlo))rhs(j,i,k)=rhs(j,i,k)-
     .     cv(j,i,k)*(top(j,i,ktlo)-htmp)
         end if
         go to 280
        end if
       end if
   20  continue
      end if
  280 continue
  300 continue
      return
```

```
      end
      subroutine sbcf1c(cr,cc,trpy,delr,delc,k,ncol,nrow,nlay)
c-----version 1334 22aug1987 sbcf1c
c     compute branch conductance using harmonic mean of block
c     conductances -- block transmissivity is in cc upon entry
c      specifications:
      dimension cr(ncol,nrow,nlay), cc(ncol,nrow,nlay)
     2   , trpy(nlay), delr(ncol), delc(nrow)
      yx=trpy(k)*2.
c1------for each cell calculate branch conductances from that cell
c1------to the one on the right and the one in front.
      do 40 i=1,nrow
      do 40 j=1,ncol
      t1=cc(j,i,k)
c2------if t=0 then set conductance equal to 0. go on to next cell.
      if(t1.ne.0.) go to 10
      cr(j,i,k)=0.
      go to 40
c3------if this is not the last column(rightmost) then calculate
c3------branch conductance in the row direction (cr) to the right.
   10 if(j.eq.ncol) go to 30
      t2=cc(j+1,i,k)
      cr(j,i,k)=2.*t2*t1*delc(i)/(t1*delr(j+1)+t2*delr(j))
c4------if this is not the last row(frontmost) then calculate
c4------branch conductance in the column direction (cc) to the front.
   30 if(i.eq.nrow) go to 40
      t2=cc(j,i+1,k)
      cc(j,i,k)=yx*t2*t1*delr(j)/(t1*delc(i+1)+t2*delc(i))
   40 continue
      return
      end
      subroutine sbcf2h(hnew,ibound,cr,cc,cv,hy,trpy,delr,delc,
     . bot,top,k,kb,kt,kiter,kstp,kper,ncol,nrow,nlay,klml,iout,
     . mcanc,mcrw,mccl,mbth)
c     30sep1991 sbcf2h
c     compute conductance from saturated thickness and hydraulic
c     conductivity
c      specifications:
      double precision hnew
      dimension hnew(ncol,nrow,nlay),ibound(ncol,nrow,nlay),
     . cr(ncol,nrow,nlay),cc(ncol,nrow,nlay),cv(ncol,nrow,klml),
     . hy(ncol,nrow,nlay),delr(ncol),delc(nrow),
     . bot(ncol,nrow,nlay),top(ncol,nrow,nlay),trpy(nlay)
      common/flwcom/laycon(80)
c1------calculate transmissivity at each active cell. transmissivity
c1------will be stored temporarily in the cc array.
      do 200 i=1,nrow
      do 200 j=1,ncol
c2------if cell is inactive then set t=0 & move on to next cell.
      if(ibound(j,i,k).ne.0) go to 10
      cc(j,i,k)=0.
      go to 200
c3------calculate saturated thickness.
   10 hd=hnew(j,i,k)
```

```
      if(laycon(k).lt.2)go to 50
      if(hd.gt.top(j,i,kt)) hd=top(j,i,kt)
   50 thck=hd-bot(j,i,kb)
c4------check to see if saturated thickness is greater than zero.
      if(thck.le.0.) go to 100
c5------if saturated thickness>0 then t=k*thickness.
      cc(j,i,k)=thck*hy(j,i,kb)
      go to 200
c6------when saturated thickness < 0, print a message and set
c6------transmissivity, ibound, and vertical conductance =0
  100 write(59,2)k,i,j,kiter,kstp,kper
      hnew(j,i,k)=0.
      cc(j,i,k)=0.
      ibound(j,i,k)=0
      if(k.lt.nlay) cv(j,i,k)=0.
      if(k.gt.1)then
c      find first active cell above
       do 20 kl=1,nlay
       kup=k-kl
       if(kup.gt.0)then
        if(ibound(j,i,kup).ne.0)then
         cv(j,i,kup)=0.
         go to 200
        end if
       end if
   20  continue
      end if
  200 continue
c     check that each active cell can flow to enough adjacent cells
c     that the solver neither zerodivides nor underflows
      do 30 m=1,2
      do 30 i=1,nrow
      do 30 j=1,ncol
      if(ibound(j,i,k).ne.0)call sbcanc(hnew,ibound,cc,cv,ncol,nrow,
     . nlay,klml,i,j,k,kb,kt,mcanc,kiter,kstp,kper,mcrw,mccl,mbth)
   30 continue
c7------compute horizontal branch conductances from transmissivity
      call sbcflc(cr,cc,trpy,delr,delc,k,ncol,nrow,nlay)
    2 format(' cell (k,i,j)',3i4,' dry @ iteration',i4,
     . ' time step',i4,' period',i4)
      return
      end
      subroutine sbcanc(hnew,ibound,cc,cv,ncol,nrow,nlay,klml,i,j,k,
     . kb,kt,mcanc,kiter,kstp,kper,mcrw,mccl,mbth)
c     30sep1991
      double precision hnew
      dimension hnew(ncol,nrow,nlay),ibound(ncol,nrow,nlay),
     . cc(ncol,nrow,nlay),cv(ncol,nrow,klml)
      mt=0
      do 30 m=1,4
      if(m.eq.1)then
       if(i.eq.1)go to 30
       im=i-1
       jm=j
```

```
       go to 10
      end if
      if(m.eq.2)then
       if(j.eq.1)go to 30
       im=i
       jm=j-1
       go to 10
      end if
      if(m.eq.3)then
       if(j.eq.ncol)go to 30
       im=i
       jm=j+1
       go to 10
      end if
      if(i.lt.nrow)then
       im=i+1
       jm=j
c      if cell cannot flow to neighbor, do not advance counter
   10  if(ibound(jm,im,k).ne.0)then
        ibab=iabs(ibound(jm,im,k))
        if(m.eq.1)then
         if(ibab.eq.mcc1.or.ibab.eq.mbth)go to 30
         go to 20
        end if
        if(m.eq.2)then
         if(ibab.eq.mcrw.or.ibab.eq.mbth)go to 30
         go to 20
        end if
        ibab=iabs(ibound(j,i,k))
        if(m.eq.3)then
         if(ibab.eq.mcrw.or.ibab.eq.mbth)go to 30
         go to 20
        end if
        if(ibab.eq.mcc1.or.ibab.eq.mbth)go to 30
   20   mt=mt+1
       end if
      end if
   30 continue
c     if counter does not exceed required number of neighbors you
c     read, cancel cell
      if(mt.le.mcanc)then
       write(59,2)k,i,j,kiter,kstp,kper
       hnew(j,i,k)=0.
       cc(j,i,k)=0.
       ibound(j,i,k)=0
       if(k.lt.nlay)cv(j,i,k)=0.
      end if
    2 format(' cell (k,i,j)',3i4,' cancelled @ iteration',i4,
     . ' time step',i4,' period',i4)
      return
      end
      subroutine bcf2al(isum,lenx,lcsc1,lchy,lcbot,lctop,lcsc2,
     . lctrpy,in,iss,ncol,nrow,nlay,ncnr,iout,ibcfcb,mcanc)
      30sep1991 bcf2al
```

```
c     allocate array storage for block-centered flow package
c        specifications:
      common/flwcom/laycon(80)
      ncnr=ncol*nrow
c1------identify package
      write(iout,1)in
    1 format(' bcf, 30sepl991, read on',10i6)
c2------read and print iss (steady-state flag) and ibcfcb (flag for
c2------printing or unit# for recording cell-by-cell flow terms)
      read(in,2)iss,ibcfcb,mcanc
      if(mcanc.lt.0)mcanc=0
      if(mcanc.gt.1)mcanc=1
    2 format(10i10)
      if(iss.eq.0) write(iout,3)
    3 format(' transient simulation')
      if(iss.ne.0) write(iout,4)
    4 format(' steady-state simulation')
      write(iout,12)mcanc
      if(ibcfcb.gt.0) write(iout,9) ibcfcb
    9 format(' cell-by-cell flows will be recorded on unit',i3)
      if(ibcfcb.lt.0) write(iout,88)
   88 format(' constant head cell-by-cell flows will be printed')
c3------read type code for each layer and count tops and bottoms
      if(nlay.le.80) go to 50
      write(iout,11)
   11 format(' you have specified more than 80 model layers',/,
     . ' space is reserved for maximum of 80 layers in array laycon')
      stop
c3a-----read layer type codes.
   50 read(in,51) (laycon(i),i=1,nlay)
   51 format(40i2)
c        bottom is read for types 1,3    top is read for types 2,3
      write(iout,52)
   52 format(' layer  aquifer type')
c3b-----initialize top and bottom counters.
      nbot=0
      ntop=0
c3c------print layer type and count tops and bottoms needed.
      do 100 i=1,nlay
c3c1----print layer number and layer type code.
      l=laycon(i)
      write(iout,7) i,l
    7 format(12i10)
c3c2----only the top layer can be unconfined(laycon=1).
      if(l.ne.1 .or. i.eq.1) go to 70
      write(iout,8)
    8 format(' aquifer type 1 is only allowed in top layer')
      stop
c3c3----layer types 1 and 3 need a bottom. add 1 to kb.
   70 if(l.eq.1 .or. l.eq.3) nbot=nbot+1
c3c4----layer types 2 and 3 need a top. add 1 to kt.
      if(l.eq.2 .or. l.eq.3) ntop=ntop+1
  100 continue
c4------compute dimensions for arrays.
```

```
      nrc=nrow*ncol
      isiz=nrc*nlay
c5------allocate space for arrays. if run is transient(iss=0)
c5------then space must be allocated for storage.
      isold=isum
      lcscl=isum
      if(iss.eq.0) isum=isum+isiz
      lcsc2=isum
      if(iss.eq.0) isum=isum+nrc*ntop
      lctrpy=isum
      isum=isum+nlay
      lcbot=isum
      isum=isum+nrc*nbot
      lchy=isum
      isum=isum+nrc*nbot
      lctop=isum
      isum=isum+nrc*ntop
c6------print the amount of space used by the bcf package.
      isp=isum-isold
      write(iout,101) isp
  101 format(i9,' elements in x array are used by bcf')
      isum1=isum-1
      write(iout,102) isum1,lenx
  102 format(i9,' elements of x array used out of',i8)
      if(isum1.gt.lenx) write(iout,103)
  103 format(' ***x array must be dimensioned larger***')
   12 format(' number: active neighbors inadequate (mcanc):',14i5)
      return
      end
      subroutine bcf2rp(ibound,hnew,scl,hy,cr,cc,cv,delr,delc,bot,
     . top,sc2,trpy,in,iss,ncol,nrow,nlay,klml,ncnr,nodes,ndsml,
     . iout)
c     30sep1991 bcf2rp
c     read and initialize data for block-centered flow package
c        specifications:
      character*4 aname
      double precision hnew
      dimension hnew(nodes),scl(nodes),hy(nodes),cr(nodes),cc(nodes),
     . cv(ndsml),aname(6,10),delr(ncol),delc(nrow),bot(nodes),
     . top(nodes),sc2(nodes),trpy(nlay),ibound(nodes)
      common/flwcom/laycon(80)
      data aname(1,1),aname(2,1),aname(3,1),aname(4,1),aname(5,1),
     1  aname(6,1) /'    ','prim','ary ','stor','age ','coef'/
      data aname(1,2),aname(2,2),aname(3,2),aname(4,2),aname(5,2),
     1  aname(6,2) /'    ','tran','smis','. al','ong ','rows'/
      data aname(1,3),aname(2,3),aname(3,3),aname(4,3),aname(5,3),
     1  aname(6,3) /'   h','yd. ','cond','. al','ong ','rows'/
      data aname(1,4),aname(2,4),aname(3,4),aname(4,4),aname(5,4),
     1  aname(6,4) /'vert',' hyd',' con','d /t','hick','ness'/
      data aname(1,5),aname(2,5),aname(3,5),aname(4,5),aname(5,5),
     1  aname(6,5) /'    ','    ','    ','    ',' bo','ttom'/
      data aname(1,6),aname(2,6),aname(3,6),aname(4,6),aname(5,6),
     1  aname(6,6) /'    ','    ','    ','    ','    ',' top'/
      data aname(1,7),aname(2,7),aname(3,7),aname(4,7),aname(5,7),
```

```
     1  aname(6,7) /'  se','cond','ary ','stor','age ','coef'/
      data aname(1,8),aname(2,8),aname(3,8),aname(4,8),aname(5,8),
     1  aname(6,8) /'colu','mn t','o ro','w an','isot','ropy'/
      data aname(1,9),aname(2,9),aname(3,9),aname(4,9),aname(5,9),
     1  aname(6,9) /'    ','    ','    ','    ','    ','delr'/
      data aname(1,10),aname(2,10),aname(3,10),aname(4,10),aname(5,10),
     1  aname(6,10) /'    ','    ','    ','    ','    ','delc'/
c1------calculate number of nodes in a layer and read trpy,delr,delc
c     nij=ncol*nrow
      call uldrel(trpy,aname(1,8),nlay,in,iout)
      call uldrel(delr,aname(1,9),ncol,in,iout)
      call uldrel(delc,aname(1,10),nrow,in,iout)
c2------read all parameters for each layer
      kt=0
      kb=0
      do 200 k=1,nlay
      kk=k
c2a-----find address of each layer in three dimension arrays.
      if(laycon(k).eq.1 .or. laycon(k).eq.3) kb=kb+1
      if(laycon(k).eq.2 .or. laycon(k).eq.3) kt=kt+1
      loc=1+(k-1)*ncnr
      locb=1+(kb-1)*ncnr
      loct=1+(kt-1)*ncnr
c2b-----read primary storage coefficient into array scl if transient
      if(iss.eq.0)call u2drel(scl(loc),aname(1,1),nrow,ncol,kk,in,iout)
c2c-----read transmissivity into array cc if layer type is 0 or 2
      if(laycon(k).eq.3 .or. laycon(k).eq.1) go to 100
      call u2drel(cc(loc),aname(1,2),nrow,ncol,kk,in,iout)
      go to 110
c2d-----read hydraulic conductivity(hy) and bottom elevation(bot)
c2d-----if layer type is 1 or 3
  100 call u2drel(hy(locb),aname(1,3),nrow,ncol,kk,in,iout)
      call u2drel(bot(locb),aname(1,5),nrow,ncol,kk,in,iout)
c2e-----read vertical hycond/thick into array cv if not bottom layer
c2e----- read as hycond/thickness -- converted to conductance later
  110 if(k.eq.nlay) go to 120
      call u2drel(cv(loc),aname(1,4),nrow,ncol,kk,in,iout)
c2f-----read secondary storage coefficient into array sc2 if transient
c2f-----and layer type is 2 or 3
  120 if(laycon(k).ne.3 .and. laycon(k).ne.2) go to 200
      if(iss.eq.0)call u2drel(sc2(loct),aname(1,7),nrow,ncol,kk,in,iout)
c2g-----read top elevation(top) if layer type is 2 or 3
      call u2drel(top(loct),aname(1,6),nrow,ncol,kk,in,iout)
  200 continue
c3------prepare and check bcf data
      call sbcf2n(hnew,ibound,scl,sc2,cr,cc,cv,hy,trpy,delr,delc,iss,
     . ncol,nrow,nlay,klml,iout)
      return
      end
      subroutine bcf2bd(vbnm,vbvl,msum,hnew,ibound,hold,scl,cr,cc,cv,
     . top,sc2,delt,iss,ncol,nrow,nlay,klml,kstp,kper,ibcfcb,
     . icbcfl,buff,iout,mbh)
c     xxaugl991 bcf2bd
c     compute budget flow terms for bcf -- storage, constant head,
```

```
c     & flow across cell walls & between layers
c     specifications:
      character*4 vbnm,text
      double precision hnew
      dimension hnew(ncol,nrow,nlay),ibound(ncol,nrow,nlay),
     . hold(ncol,nrow,nlay),scl(ncol,nrow,nlay),sc2(ncol,nrow,nlay),
     . cr(ncol,nrow,nlay),cc(ncol,nrow,nlay),cv(ncol,nrow,klml),
     . top(ncol,nrow,nlay),buff(ncol,nrow,nlay),vbnm(4,20),vbvl(4,20)
      common/flwcom/laycon(80)
      dimension text(4)
      data text(1),text(2),text(3),text(4) /'    ','    ',' sto','rage'/
c1------initialize budget accumulators
      stoin=0.
      stout=0.
c2------if cell-by-cell flows are needed then set flag ibd.
      ibd=0
      if(icbcfl.ne.0 .and. ibcfcb.gt.0) ibd=1
c3------if steady state then skip all storage calculations
      if(iss.ne.0) go to 305
c4------if cell-by-cell flows are needed (ibd is set) clear buffer
      if(ibd.eq.0) go to 220
      do 210 k=1,nlay
      do 210 i=1,nrow
      do 210 j=1,ncol
      buff(j,i,k)=0.
  210 continue
c5------run through every cell in the grid
  220 kt=0
      do 300 k=1,nlay
      lc=laycon(k)
      if(lc.eq.3 .or. lc.eq.2) kt=kt+1
      do 300 i=1,nrow
      do 300 j=1,ncol
c6------calculate flow from storage (variable head cells only)
      if(ibound(j,i,k).le.0) go to 300
      hsing=hnew(j,i,k)
c6a-----check layer type to see if one storage capacity or two
      if(lc.ne.3 .and. lc.ne.2) go to 285
c6b-----two storage capacities
      tp=top(j,i,kt)
      sya=sc2(j,i,kt)
      scfa=scl(j,i,k)
      sold=sya
      if(hold(j,i,k).gt.tp) sold=scfa
      snew=sya
      if(hsing.gt.tp) snew=scfa
      strg=sold*(hold(j,i,k)-tp) + snew*tp - snew*hsing
      go to 288
c6c-----one storage capacity
  285 sc=scl(j,i,k)
      strg=sc*hold(j,i,k) - sc*hsing
c7------store cell-by-cell flow in buffer and add to accumulators
  288 if(ibd.eq.1) buff(j,i,k)=strg/delt
      if(strg) 292,300,294
```

```
  292 stout=stout-strg
      go to 300
  294 stoin=stoin+strg
  300 continue
c8-----if ibd flag is set record the contents of the buffer
      if(ibd.eq.1) call ubudsv(kstp,kper,text,
     .       ibcfcb,buff,ncol,nrow,nlay,iout)
c9------add total rates and volumes to vbvl & put titles in vbnm
  305 vbvl(1,msum)=vbvl(1,msum)+stoin
      vbvl(2,msum)=vbvl(2,msum)+stout
      vbvl(3,msum)=stoin/delt
      vbvl(4,msum)=stout/delt
      vbnm(1,msum)=text(1)
      vbnm(2,msum)=text(2)
      vbnm(3,msum)=text(3)
      vbnm(4,msum)=text(4)
      msum=msum+1
c10-----calculate flow from constant head nodes
      call sbcf2f(vbnm,vbvl,msum,hnew,ibound,cr,cc,cv,top,delt,
     . ncol,nrow,nlay,klml,kstp,kper,ibd,ibcfcb,icbcfl,buff,iout,mbh)
c11-----calculate and save flow across cell boundaries if c-b-c
c11-----flow terms are requested.
      if(ibd.ne.0)call sbcf2b(hnew,ibound,cr,cc,cv,top,ncol,nrow,nlay,
     . klml,kstp,kper,ibcfcb,buff,iout)
      call sbcflk(ncol,nrow,nlay,klml,hnew,ibound,cv,top)
      return
      end
      subroutine sbcf2b(hnew,ibound,cr,cc,cv,top,ncol,nrow,nlay,klml,
     . kstp,kper,ibcfcb,buff,iout)
c     30sep1991 sbcf2b
c     compute flow across each cell wall
c     specifications:
      character*4 text
      double precision hnew,hd
      dimension hnew(ncol,nrow,nlay),ibound(ncol,nrow,nlay),
     . cr(ncol,nrow,nlay),cc(ncol,nrow,nlay),cv(ncol,nrow,klml),
     . top(ncol,nrow,nlay),buff(ncol,nrow,nlay)
      common/flwcom/laycon(80)
      dimension text(12)
      data text(1),text(2),text(3),text(4),text(5),text(6),text(7),
     . text(8),text(9),text(10),text(11),text(12)
     . /'flow',' rig','ht f','ace ',
     . 'flow',' fro','nt f','ace ','flow',' low','er f','ace '/
      ncml=ncol-1
      if(ncml.lt.1)go to 405
c1-----clear the buffer
      do 310 k=1,nlay
      do 310 i=1,nrow
      do 310 j=1,ncol
      buff(j,i,k)=0.
  310 continue
c2-----for each cell calculate flow thru right face & store in buffer
      do 400 k=1,nlay
      do 400 i=1,nrow
```

```
      do 400 j=1,ncm1
      if((ibound(j,i,k).le.0) .and. (ibound(j+1,i,k).le.0)) go to 400
      hdiff=hnew(j,i,k)-hnew(j+1,i,k)
      buff(j,i,k)=hdiff*cr(j,i,k)
  400 continue
c3-----record contents of buffer
      call ubudsv(kstp,kper,text(1),ibcfcb,buff,ncol,nrow,nlay,iout)
c4-----clear the buffer
  405 nrm1=nrow-1
      if(nrm1.lt.1) go to 505
      do 410 k=1,nlay
      do 410 i=1,nrow
      do 410 j=1,ncol
      buff(j,i,k)=0.
  410 continue
c5-----for each cell calculate flow thru front face & store in buffer
      do 500 k=1,nlay
      do 500 i=1,nrm1
      do 500 j=1,ncol
      if((ibound(j,i,k).le.0) .and. (ibound(j,i+1,k).le.0)) go to 500
      hdiff=hnew(j,i,k)-hnew(j,i+1,k)
      buff(j,i,k)=hdiff*cc(j,i,k)
  500 continue
c6-----record contents of buffer.
      call ubudsv(kstp,kper,text(5),ibcfcb,buff,ncol,nrow,nlay,iout)
  505 if(klm1.lt.1)return
c7-----clear the buffer
      do 510 k=1,nlay
      do 510 i=1,nrow
      do 510 j=1,ncol
      buff(j,i,k)=0.
  510 continue
c8-----for each cell calculate flow thru lower face & store in buffer
      kt=0
      do 600 k=1,klm1
      if(laycon(k).eq.3 .or. laycon(k).eq.2) kt=kt+1
      do 600 i=1,nrow
      do 600 j=1,ncol
      if(ibound(j,i,k).ne.0)then
c      find first active cell below
       ktlo=kt
       do 20 kl=1,nlay
       klo=k+kl
       if(klo.le.nlay)then
        if((ibound(j,i,k).ge.0).or.(ibound(j,i,klo).ge.0))then
         if(laycon(klo).gt.1)ktlo=ktlo+1
         if(ibound(j,i,klo).eq.0)go to 20
         hd=hnew(j,i,klo)
         if(laycon(klo).lt.2)go to 10
         tmp=hd
         if(tmp.lt.top(j,i,ktlo))hd=top(j,i,ktlo)
   10    hdiff=hnew(j,i,k)-hd
         buff(j,i,k)=hdiff*cv(j,i,k)
         go to 600
```

```
        end if
       end if
   20  continue
      end if
  600 continue
c9-----record contents of buffer.
      call ubudsv(kstp,kper,text(9),ibcfcb,buff,ncol,nrow,nlay,iout)
      return
      end
      subroutine sbcf2f(vbnm,vbvl,msum,hnew,ibound,cr,cc,cv,top,delt,
     . ncol,nrow,nlay,klml,kstp,kper,ibd,ibcfcb,icbcfl,buff,iout,mbh)
c     30sepl991 sbcf2f
c     compute flow from constant head nodes
c     specifications:
      character*4 vbnm,text
      double precision hnew,hd
      dimension hnew(ncol,nrow,nlay),ibound(ncol,nrow,nlay),
     . cr(ncol,nrow,nlay),cc(ncol,nrow,nlay),cv(ncol,nrow,klml),
     . top(ncol,nrow,nlay),buff(ncol,nrow,nlay),vbnm(4,20),vbvl(4,20)
      common/flwcom/laycon(80)
      dimension text(4)
      data text(1),text(2),text(3),text(4) /'   c','onst','ant ','head'/
c1------clear budget accumulators
      chin=0.
      chout=0.
c2------clear buffer if cell-by-cell flow term flag(ibd) is set
      if(ibd.eq.0) go to 8
      do 5 k=1,nlay
      do 5 i=1,nrow
      do 5 j=1,ncol
      buff(j,i,k)=0.
    5 continue
c3------for each cell if it is constant head compute flow across 6
c3-----faces.
    8 kt=0
      do 200 k=1,nlay
      lc=laycon(k)
      if(lc.eq.3 .or. lc.eq.2) kt=kt+1
      do 200 i=1,nrow
      do 200 j=1,ncol
c4-----if cell is not constant head skip it & go on to next cell.
      if (ibound(j,i,k).ge.0)go to 200
c5-----clear fields for six flow rates.
      x1=0.
      x2=0.
      x3=0.
      x4=0.
      x5=0.
      x6=0.
c6-----for each face of the cell calculate flow through that face
c6-----out of the constant head cell and into the flow domain.
c6-----comments 7-11 appear only in the section headed by comment 6a
c6-----but they apply in a similar manner to the sections headed
c6-----by comments 6b-6f.
```

```
c6a----calculate flow through the left face
c7-----if there is not a variable head cell on the other side of this
c7-----face then go on to the next face.
      if(j.eq.1) go to 30
      if(ibound(j-1,i,k).le.0)go to 30
      hdiff=hnew(j,i,k)-hnew(j-1,i,k)
c8-----calculate flow through this face into the adjacent cell.
      xl=hdiff*cr(j-1,i,k)
c9-----test to see if flow is positive or negative
      if (xl) 10,30,20
c10----if negative add to chout(flow out of domain to constant head).
   10 chout=chout-xl
      go to 30
c11----if positive add to chin(flow into domain from constant head).
   20 chin=chin+xl
c6b----calculate flow through the right face
   30 if(j.eq.ncol) go to 60
      if(ibound(j+1,i,k).le.0) go to 60
      hdiff=hnew(j,i,k)-hnew(j+1,i,k)
      x2=hdiff*cr(j,i,k)
      if(x2)40,60,50
   40 chout=chout-x2
      go to 60
   50 chin=chin+x2
c6c----calculate flow through the back face.
   60 if(i.eq.1) go to 90
      if (ibound(j,i-1,k).le.0) go to 90
      hdiff=hnew(j,i,k)-hnew(j,i-1,k)
      x3=hdiff*cc(j,i-1,k)
      if(x3) 70,90,80
   70 chout=chout-x3
      go to 90
   80 chin=chin+x3
c6d----calculate flow through the front face.
   90 if(i.eq.nrow) go to 120
      if(ibound(j,i+1,k).le.0) go to 120
      hdiff=hnew(j,i,k)-hnew(j,i+1,k)
      x4=hdiff*cc(j,i,k)
      if (x4) 100,120,110
  100 chout=chout-x4
      go to 120
  110 chin=chin+x4
  120 if(k.gt.1)then
c      flow thru upper wall
c      find first active cell above
       do 130 kl=1,nlay
       kup=k-kl
       if(kup.lt.1)go to 150
       if(ibound(j,i,kup).lt.0)go to 150
       if(ibound(j,i,kup).ne.0)then
        hd=hnew(j,i,k)
        if(laycon(k).gt.1)then
         tmp=hd
         if(tmp.lt.top(j,i,kt))hd=top(j,i,kt)
```

```
        end if
        hdiff=hd-hnew(j,i,kup)
        x5=hdiff*cv(j,i,kup)
        go to 140
       end if
  130  continue
  140  if(x5.lt.0.)then
        chout=chout-x5
        go to 150
       end if
       chin=chin+x5
      end if
  150 if(k.lt.nlay)then
c      flow thru lower wall
c      find first active cell below
       ktlo=kt
       do 160 kl=1,nlay
       klo=k+kl
       if(klo.gt.nlay)go to 180
       if(ibound(j,i,klo).lt.0)go to 180
       if(laycon(klo).gt.1)ktlo=ktlo+1
       if(ibound(j,i,klo).ne.0)then
        hd=hnew(j,i,klo)
        if(laycon(klo).gt.1)then
         tmp=hd
         if(tmp.lt.top(j,i,ktlo))hd=top(j,i,ktlo)
        end if
        hdiff=hnew(j,i,k)-hd
        x6=hdiff*cv(j,i,k)
        go to 170
       end if
  160  continue
  170  if(x6.lt.0.)then
        chout=chout-x6
        go to 180
       end if
       chin=chin+x6
      end if
c12-----sum up flows through six sides of constant head cell.
  180 rate=x1+x2+x3+x4+x5+x6
c13-----print the individual rates if requested(ibcfcb<0).
      if(mbh.eq.1)write(57,900)(text(n),n=1,4),
     .  kper,kstp,k,j,i,rate
  900 format(1x,4a4,' period',i4,' step',i4,12x,'k',
     . i4,' i',i4,' j',i4,' q',1p4e13.4)
c14----if cell-by-cell flag set store sum of flows for cell in buffer
      if(ibd.eq.1) buff(j,i,k)=rate
  200 continue
c15----if cell-by-cell flag set then record contents of buffer
      if(ibd.eq.1) call ubudsv(kstp,kper,text(1),
     .      ibcfcb,buff,ncol,nrow,nlay,iout)
c16----save total constant head flows and volumes in vbvl table
c16----for inclusion in budget. put labels in vbnm table.
      vbvl(1,msum)=vbvl(1,msum)+chin*delt
```

```
      vbvl(2,msum)=vbvl(2,msum)+chout*delt
      vbvl(3,msum)=chin
      vbvl(4,msum)=chout
c     ---setup volumetric budget names
      vbnm(1,msum)=text(1)
      vbnm(2,msum)=text(2)
      vbnm(3,msum)=text(3)
      vbnm(4,msum)=text(4)
      msum=msum+1
      return
      end
      subroutine sbcf2n(hnew,ibound,sc1,sc2,cr,cc,cv,hy,trpy,delr,
     . delc,iss,ncol,nrow,nlay,klml,iout)
c     30sep1991 sbcf2n
c     initialize and check bcf data
c        specifications:
c     double precision hnew,hcnv
      double precision hnew
      dimension hnew(ncol,nrow,nlay),ibound(ncol,nrow,nlay),
     . cr(ncol,nrow,nlay),cc(ncol,nrow,nlay),cv(ncol,nrow,klml),
     . hy(ncol,nrow,nlay),sc1(ncol,nrow,nlay),sc2(ncol,nrow,nlay),
     . trpy(nlay),delr(ncol),delc(nrow)
      common/flwcom/laycon(80)
c1------if ibound=0, set cv=0., cc=0., and hy=0.
      kb=0
      do 30 k=1,nlay
      if(laycon(k).eq.3 .or. laycon(k).eq.1) kb=kb+1
      do 30 i=1,nrow
      do 30 j=1,ncol
      if(ibound(j,i,k).ne.0) go to 30
      if(k.ne.nlay) cv(j,i,k)=0.
      cc(j,i,k)=0.
      if(laycon(k).eq.3 .or. laycon(k).eq.1) hy(j,i,kb)=0.
   30 continue
c2------insure that each active cell has at least one non-zero
c2------transmissive parameter. if not, convert cell to noflow.
      kb=0
      do 60 k=1,nlay
      if(laycon(k).eq.0.or.laycon(k).eq.2)then
c      if layer confined or fully convertible, check cv & cc
       do 130 i=1,nrow
       do 130 j=1,ncol
       if(ibound(j,i,k).ne.0)then
        if(cc(j,i,k).eq.0.)then
         if(k.lt.nlay)then
          if(cv(j,i,k).gt.0.)go to 130
         end if
         if(k.gt.1)then
c         find first active cell above
          do 110 kl=1,nlay
          kup=k-kl
          if(kup.gt.0)then
           if(ibound(j,i,kup).ne.0)then
            if(cv(j,i,kup).gt.0.)go to 130
```

```
            go to 120
           end if
          end if
  110     continue
         end if
  120    ibound(j,i,k)=0
         hnew(j,i,k)=0.
         write(59,2)k,i,j
        end if
       end if
  130  continue
       go to 60
      end if
c     if layer unconfined or partly convertible, check cv & hy
      kb=kb+1
      do 160 i=1,nrow
      do 160 j=1,ncol
      if(ibound(j,i,k).ne.0)then
       if(hy(j,i,kb).eq.0.)then
        if(k.lt.nlay)then
         if(cv(j,i,k).gt.0.)go to 160
        end if
        if(k.gt.1)then
c        find first active cell above
         do 140 kl=1,nlay
         kup=k-kl
         if(kup.gt.0)then
          if(ibound(j,i,kup).ne.0)then
           if(cv(j,i,kup).gt.0.)go to 160
           go to 150
          end if
         end if
  140    continue
        end if
  150   ibound(j,i,k)=0
        hnew(j,i,k)=0.
        cc(j,i,k)=0.
        write(59,2)k,i,j
       end if
      end if
  160 continue
   60 continue
c3------calculate hor. conductance(cr and cc) for constant t layers
      do 65 k=1,nlay
      kk=k
      if(laycon(k).eq.3 .or. laycon(k).eq.1) go to 65
      call sbcflc(cr,cc,trpy,delr,delc,kk,ncol,nrow,nlay)
   65 continue
c4------multiply vertical leakance by area to make conductance
      if(nlay.eq.1) go to 69
      do 68 k=1,klml
      do 68 i=1,nrow
      do 68 j=1,ncol
      cv(j,i,k)=cv(j,i,k)*delr(j)*delc(i)
```

```
   68 continue
c5------if transient multiply primary storage coefficient by delr &
c5------delc to get primary storage capacity(scl).
   69 if(iss.ne.0)return
      kt=0
      do 80 k=1,nlay
      do 70 i=1,nrow
      do 70 j=1,ncol
      scl(j,i,k)=scl(j,i,k)*delr(j)*delc(i)
   70 continue
c6------if layer is conf/unconf multiply secondary storage coefficient
c6------by delr and delc to get secondary storage capacity(sc2).
      if(laycon(k).ne.3 .and. laycon(k).ne.2) go to 80
      kt=kt+1
      do 75 i=1,nrow
      do 75 j=1,ncol
      sc2(j,i,kt)=sc2(j,i,kt)*delr(j)*delc(i)
   75 continue
   80 continue
    2 format(' cell (k,i,j)',3i4,
     . ' cancelled: all conductances to cell = 0')
      return
      end
      subroutine sbcflk(ncol,nrow,nlay,klml,hnew,ibound,cv,top)
c     31aug1991 sbcflk
c     calculates flow thru bottom & top cell walls & sums it by layer
      double precision hnew,hd,tpin,tpot,btin,btot,topin,tpout,
     . botin,btout,tptt,bttt,tttb,hdff,xup,xlo
      dimension hnew(ncol,nrow,nlay),ibound(ncol,nrow,nlay),
     . top(ncol,nrow,nlay),cv(ncol,nrow,klml),tpin(9,9),tpot(9,9),
     . btin(9,9),btot(9,9)
      common/flwcom/laycon(80)
      if(nlay.eq.1)return
      write(6,2)
      kt=0
      do 90 k=1,nlay
      topin=0.
      tpout=0.
      botin=0.
      btout=0.
      do 10 kl=1,nlay
      tpin(k,kl)=0.
      tpot(k,kl)=0.
      btin(k,kl)=0.
   10 btot(k,kl)=0.
      if(laycon(k).gt.1)kt=kt+1
      do 70 i=1,nrow
      do 70 j=1,ncol
      if(ibound(j,i,k).le.0)go to 70
      if(k.eq.1)go to 40
c     flow thru top wall
c     find first active cell above
      do 20 kl=1,nlay
      kup=k-kl
```

```
      if(kup.lt.1)go to 40
      if(ibound(j,i,kup).eq.0)go to 20
      hd=hnew(j,i,k)
      hdsng=hd
      if(laycon(k).gt.1)hd=amax1(hdsng,top(j,i,kt))
      hdff=hnew(j,i,kup)-hd
      xup=hdff*cv(j,i,kup)
      go to 30
   20 continue
   30 if(xup.gt.0.)then
       topin=topin+xup
       tpin(k,kup)=tpin(k,kup)+xup
       go to 40
      end if
      tpout=tpout+xup
      tpot(k,kup)=tpot(k,kup)+xup
   40 if(k.eq.nlay)go to 70
c     flow thru bottom wall
c     find first active cell below
      ktlo=kt
      do 50 kl=1,nlay
      klo=k+kl
      if(klo.gt.nlay)go to 70
      if(laycon(klo).gt.1)ktlo=ktlo+1
      if(ibound(j,i,klo).eq.0)go to 50
      hd=hnew(j,i,klo)
      hdsng=hd
      if(laycon(klo).gt.1)hd=amax1(hdsng,top(j,i,ktlo))
      hdff=hd-hnew(j,i,k)
      xlo=hdff*cv(j,i,k)
      go to 60
   50 continue
   60 if(xlo.gt.0.)then
       botin=botin+xlo
       btin(k,klo)=btin(k,klo)+xlo
      go to 70
      end if
      btout=btout+xlo
      btot(k,klo)=btot(k,klo)+xlo
   70 continue
      tptt=topin+tpout
      bttt=botin+btout
      tttb=tptt+bttt
      do 80 kl=1,nlay
   80 write(6,4)kl,tpin(k,kl),tpot(k,kl),btin(k,kl),btot(k,kl)
   90 write(6,6)k,topin,tpout,tptt,botin,btout,bttt,tttb
    2 format(' flow between layers',/,4x,'k',11x,'topin',9x,'tpout',
     . 8x,'totopin',8x,'botin',9x,'btout',8x,'tobotin',7x,'totalin')
    4 format(i10,1p2e14.5,14x,1p5e14.5)
    6 format(i5,5x,1p8e14.5)
      return
      end
      subroutine cutla1(isum,lenx,lcut,mxcut,mcrw,mccl,mbth,
     . in,iout)
```

```
c     30sep1991
c     allocates space
      write(iout,2)in
c     read number of cutters & ibound signals
      read(in,4)mxcut,mcrw,mccl,mbth
      write(iout,6)mxcut,mcrw,mccl,mbth
c     allocate space
      lcut=isum
      lmnts=4*mxcut
      isum=isum+lmnts
      write(iout,8)lmnts
      ismml=isum-1
      write(iout,12)ismml,lenx
      if(ismml.gt.lenx)write(iout,14)
    2 format(' cut, 30sep1991, read on',10i6)
    4 format(10i10)
    6 format(' maximum number: cutters (mxcut) & ibound signals:',/,
     . ' row-branch (mcrw), column-branch (mccl), & both branches ',
     . '(mbth):',/,18i7)
    8 format(' elements in cut',10i10)
   12 format(' ismml,lenx',10i10)
   14 format(' x too small')
      return
      end
      subroutine cutlib(ibound,cttr,mxcut,ncol,nrow,nlay,mcrw,mccl,
     . mbth)
c     30sep1991
c     signals ibound to account for canyon cut thru layer
      dimension ibound(ncol,nrow,nlay),cttr(4,mxcut)
      common/cutcom/laycut(80)
      nmbr=0
c     for each layer, signal ibound
      do 20 k=1,nlay
      if(laycut(k).gt.0)then
c      if cutters, process
       do 10 m=1,laycut(k)
       nmbr=nmbr+1
       i=cttr(1,nmbr)
       j=cttr(2,nmbr)
       ip1=cttr(3,nmbr)
       jp1=cttr(4,nmbr)
       if(ibound(j,i,k).ne.0)then
c       if not inactive, process
        np=1
c       if constant head, preserve minus sign
        if(ibound(j,i,k).lt.0)np=-1
        npcrw=mcrw*np
        if(j.ne.jp1)then
c        j.ne.jp1 so cutter is on row-branch conductance
         ibound(j,i,k)=npcrw
         go to 10
        end if
c       j.eq.jp1 so cutter is on column-branch conductance
        if(ibound(j,i,k).ne.npcrw)then
```

```
          ibound(j,i,k)=mccl*np
          go to 10
         end if
         ibound(j,i,k)=mbth*np
c        row-branch signal is set; write over it that both are cut
        end if
   10   continue
       end if
   20  continue
       return
       end
       subroutine cutlmd(cr,cc,cttr,mxcut,ncol,nrow,nlay,kctl,kct2)
c      30sep1991
c      sets branch conductance to zero to account for canyon cut thru layer
       dimension cr(ncol,nrow,nlay),cc(ncol,nrow,nlay),cttr(4,mxcut)
       common/cutcom/laycut(80)
       common/flwcom/laycon(80)
       nmbr=0
c      for each layer, set branch conductance to zero
       do 20 k=1,nlay
       if(laycon(k).ne.kctl.and.laycon(k).ne.kct2)then
c       signals have been set opposite to laycon
c       process layer, if: before iterations, constant transmissivity
c                          during iterations, variable transmissivity
        if(laycut(k).gt.0)then
c        if cutters, process
         do 10 m=1,laycut(k)
         nmbr=nmbr+1
         i=cttr(1,nmbr)
         j=cttr(2,nmbr)
         ipl=cttr(3,nmbr)
         jpl=cttr(4,nmbr)
         if(j.ne.jpl)then
c         j.ne.jpl so cutter is on row-branch conductance
          if(cr(j,i,k).ne.0.)cr(j,i,k)=0.
          go to 10
         end if
c        j.eq.jpl so cutter is on column-branch conductance
         if(cc(j,i,k).ne.0.)cc(j,i,k)=0.
   10    continue
        end if
       end if
   20  continue
       return
       end
       subroutine cutlrp(cr,cc,ibound,cttr,mxcut,ncol,nrow,nlay,nodes,
      . in,iout,mcrw,mccl,mbth,kctl,kct2)
c      30sep1991
c      reads & initializes
       dimension cr(nodes),cc(nodes),ibound(nodes),cttr(4,mxcut)
       common/cutcom/laycut(80)
       common/flwcom/laycon(80)
c      for each layer, read number of cutters
       read(in,2)(laycut(k),k=1,nlay)
```

```
      write(iout,4)
      do 10 k=1,nlay
   10 write(iout,2)k,laycut(k)
      write(60,6)
      nmbr=0
c     for each layer, read data
      do 30 k=1,nlay
      if(laycut(k).gt.0)then
c      if layer has cutters, process
       do 20 m=1,laycut(k)
       nmbr=nmbr+1
c      for each cutter, read node & forward node
c      either (ipl=i & jpl=(j+1)) or (ipl=(i+1) & jpl=j)
c      if cell has two cutters, read j.ne.jpl (row-branch) first
c      so that ctmbn will see them in correct order
       read(in,2)i,j,ipl,jpl
       write(60,8)nmbr,i,j,ipl,jpl
       cttr(1,nmbr)=i
       cttr(2,nmbr)=j
       cttr(3,nmbr)=ipl
       cttr(4,nmbr)=jpl
   20  continue
      end if
   30 continue
      call cutlib(ibound,cttr,mxcut,ncol,nrow,nlay,mcrw,mccl,mbth)
c     for constant-transmissivity, set branch conductance to zero &
c     set signals opposite to laycon
      kct1=1
      kct2=3
      call cutlmd(cr,cc,cttr,mxcut,ncol,nrow,nlay,kct1,kct2)
c     for variable-transmissivity, set signals opposite to laycon
      kct1=0
      kct2=2
    2 format(10i10)
    4 format(' number:  cutters:',/,'          k  cutters')
    6 format(' nmbr    i    j  ipl  jpl')
    8 format(20i5)
      return
      end
      subroutine drnlal(isum,lenx,lcdrai,ndrain,mxdrn,in,iout,
     .     idrncb)
c     30sep1991
c     allocate array storage for drain package
c        specifications:
c1------identify package and initialize ndrain.
      write(iout,1)in
    1 format(' drain, 30sep1991, read on',10i6)
      ndrain=0
c2------read & print mxdrn & idrncb(unit & flag for cell-by-cell flow)
      read(in,2) mxdrn,idrncb
    2 format(2i10)
      write(iout,3) mxdrn
    3 format(' maximum of',i5,' drains')
      if(idrncb.gt.0)write(iout,9) idrncb
```

```
    9 format(' cell-by-cell flows will be recorded on unit',i3)
      if(idrncb.lt.0)write(iout,8)
    8 format(' cell-by-cell flows will be printed when icbcfl not 0')
c3------set lcdrai equal to address of first unused space in x.
      lcdrai=isum
c4------calculate amount of space used by the drain package.
      isp=5*mxdrn
      isum=isum+isp
c5------print amount of space used by drain package.
      write(iout,4) isp
    4 format(i9,' elements in x array are used for drains')
      isum1=isum-1
      write(iout,5) isum1,lenx
    5 format(i9,' elements of x array used out of',i8)
      if(isum1.gt.lenx)write(iout,6)
    6 format(' ***x array must be dimensioned larger***')
      return
      end
      subroutine drn1rp(drai,ndrain,mxdrn,in,iout)
c     30sep1991
c     read drain locations, elevations, and conductances
c        specifications:
      dimension drai(5,mxdrn)
c1------read itmp(number of drain cells or flag to reuse data)
      read(in,8) itmp
    8 format(i10)
      read(in,22)fall
      if(fall.le.0.)fall=1.
      write(6,24)fall
   22 format(10f10.0)
   24 format(' fcdrn',1p8e13.4)
c2------test itmp
      if(itmp.ge.0) go to 50
c2a-----if itmp<0 then reuse data from last stress period.
      write(iout,7)
    7 format(' reusing drains from last stress period')
      return
c3------if itmp=>0 then it is the number of drains.
   50 ndrain=itmp
      if(ndrain.le.mxdrn) go to 100
c4------if ndrain>mxdrn then stop
      write(iout,99) ndrain,mxdrn
   99 format(' ndrain(',i4,') is greater than mxdrn(',i4,')')
      stop
c5------print number of drains in this stress period.
  100 write(iout,1) ndrain
    1 format(i6,' drains')
c6------if there are no drains then return.
      if(ndrain.eq.0)return
c7------read data for each drain.
c     write(iout,3)
    3 format(' layer',5x,'row',5x,
     . 'col   elevation   conductance   drain no.')
      do 250 ii=1,ndrain
```

```
      read(in,4)k,i,j,drai(4,ii),drai(5,ii)
      drai(5,ii)=drai(5,ii)*fall
    4 format(3i10,2f10.0)
c     write(57,5)k,j,i,drai(4,ii),drai(5,ii),ii
    5 format(3i5,f10.0,1pe12.3,3i7)
      drai(1,ii)=k
      drai(2,ii)=i
      drai(3,ii)=j
  250 continue
      return
      end
      subroutine drn1fm(ndrain,mxdrn,drai,hnew,hcof,rhs,ibound,
     . ncol,nrow,nlay)
c     30sep1991
c     add drain flow to source term
c        specifications:
      double precision hnew
      dimension drai(5,mxdrn),hnew(ncol,nrow,nlay),
     .      rhs(ncol,nrow,nlay),ibound(ncol,nrow,nlay),
     .      hcof(ncol,nrow,nlay)
c1------if ndrain<=0 there are no drains. return
      if(ndrain.le.0) return
c2------process each cell in the drain list
      do 100 l=1,ndrain
c3------get column, row and layer of cell containing drain.
      il=drai(1,l)
      ir=drai(2,l)
      ic=drai(3,l)
c4-------if the cell is external skip it.
      if(ibound(ic,ir,il).le.0) go to 100
c5-------if the cell is internal get the drain data.
      el=drai(4,l)
      hhnew=hnew(ic,ir,il)
c6------if head is lower than drain then skip this cell.
      if(hhnew.le.el) go to 100
c7------head is higher than drain. add terms to rhs and hcof.
      c=drai(5,l)
      hcof(ic,ir,il)=hcof(ic,ir,il)-c
      rhs(ic,ir,il)=rhs(ic,ir,il)-c*el
  100 continue
      return
      end
      subroutine drn2bd(ndrain,mxdrn,vbnm,vbvl,msum,drai,delt,hnew,
     . ncol,nrow,nlay,ibound,kstp,kper,idrncb,icbcfl,buff,iout,mbh)
c     30sep1991 drn2bd
c     calculate volumetric budget for drains
c        specifications:
      character*4 vbnm,text
      double precision hnew
      dimension vbnm(4,msum),vbvl(4,msum),drai(5,mxdrn),
     . hnew(ncol,nrow,nlay),ibound(ncol,nrow,nlay),
     . buff(ncol,nrow,nlay)
      dimension text(4)
      data text(1),text(2),text(3),text(4) /'    ','    ','  dr','ains'/
```

```
c1------initialize cell-by-cell flow term flag (ibd) and
c1------accumulators (ratin and ratout).
      ratout=0.
      ibd=0
c2------if there are no drains then do not accumulate drain flow
      if(ndrain.le.0) go to 200
c3------test to see if cell-by-cell flow terms are needed.
      if(icbcfl.eq.0 .or. idrncb.le.0) go to 60
c3b-----cell-by-cell flow terms are needed set ibd and clear buffer.
      ibd=1
      do 50 il=1,nlay
      do 50 ir=1,nrow
      do 50 ic=1,ncol
      buff(ic,ir,il)=0.
   50 continue
c4------for each drain accumulate drain flow
   60 do 100 l=1,ndrain
c5------get layer, row & column of cell containing reach.
      il=drai(1,l)
      ir=drai(2,l)
      ic=drai(3,l)
c     if cell is external, set q=0.
      if(ibound(ic,ir,il).le.0)then
       q=0.
       go to 10
      end if
c7------get drain parameters from drain list.
      el=drai(4,l)
      c=drai(5,l)
      hhnew=hnew(ic,ir,il)
c     if head lower than drain, set q=0.
      if(hhnew.le.el)then
       q=0.
       go to 10
      end if
c9------head higher than drain.  calculate q=c*(el-hhnew).
c9------subtract q from ratout.
      q=c*(el-hhnew)
      ratout=ratout-q
c10-----print the individual rates if requested(idrncb<0).
   10 if(mbh.eq.1)write(57,900)(text(n),n=1,4),
     . kper,kstp,1,il,ir,ic,q
  900 format(1x,4a4,' period',i4,' step',i4,' drain',i5,
     . ' k',i4,' i',i4,' j',i4,' q',1p4el3.4)
c11-----if c-b-c flow terms are to be saved then add q to buffer.
      if(ibd.eq.1) buff(ic,ir,il)=buff(ic,ir,il)+q
  100 continue
c12-----if c-b-c flow terms will be saved call ubudsv to record them.
      if(ibd.eq.1) call ubudsv(kstp,kper,text,idrncb,buff,ncol,nrow,
     .    nlay,iout)
c13-----move rates,volumes & labels into arrays for printing.
  200 vbvl(3,msum)=0.
      vbvl(4,msum)=ratout
      vbvl(2,msum)=vbvl(2,msum)+ratout*delt
```

```
      vbnm(1,msum)=text(1)
      vbnm(2,msum)=text(2)
      vbnm(3,msum)=text(3)
      vbnm(4,msum)=text(4)
c14-----increment budget term counter
      msum=msum+1
      return
      end
      subroutine ghblal(isum,lenx,lcbnds,nbound,mxbnd,in,iout,
     .   ighbcb)
c     30sep1991
c     allocate array storage for head-dependent boundaries
c     specifications:
c1------identify package and initialize # of general head bounds
      write(iout,1)in
    1 format(' ghb, 30sep1991, read on',10i6)
      nbound=0
c2------read and print mxbnd and ighbcb (max # of bounds and unit
c2------for cell-by-cell flow terms for ghb)
      read(in,2) mxbnd,ighbcb
    2 format(2i10)
      write(iout,3) mxbnd
    3 format(' maximum of',i5,' head-dependent boundary nodes')
      if(ighbcb.gt.0)write(iout,9) ighbcb
    9 format(' cell-by-cell flow will be recorded on unit',i3)
      if(ighbcb.lt.0)write(iout,8)
    8 format(' cell-by-cell flow will be printed when icbcfl not 0')
c3------set lcbnds equal to address of first unused space in x.
      lcbnds=isum
c4------calculate amount of space used by the general head list.
      isp=5*mxbnd
      isum=isum+isp
c5------print amount of space used by the ghb package
      write(iout,4) isp
    4 format(i9,' elements in x array are used for head-',
     . 'dependent boundaries')
      isum1=isum-1
      write(iout,5) isum1,lenx
    5 format(i9,' elements of x array used out of',i8)
      if(isum1.gt.lenx)write(iout,6)
    6 format(' ***x array must be dimensioned larger***')
      return
      end
      subroutine ghb1rp(bnds,nbound,mxbnd,in,iout)
c     30sep1991
c     read data for ghb
c     specifications:
      dimension bnds(5,mxbnd)
c1------read itmp(# of general head bounds or flag to reuse data.)
      read(in,8) itmp
    8 format(i10)
      read(in,22)fall
      if(fall.le.0.)fall=1.
      write(6,24)fall
```

```
   22 format(10f10.0)
   24 format(' fcghb',1p8e13.4)
c2------test itmp
      if(itmp.ge.0) go to 50
c2a-----if itmp<0 then reuse data from last stress period
      write(iout,7)
    7 format(' reusing head-dependent bounds from last stress',
     1      ' period')
      return
c3------if itmp=>0 then it is the # of general head bounds.
   50 nbound=itmp
c4------if max number of bounds is exceeded then stop
      if(nbound.le.mxbnd) go to 100
      write(iout,99) nbound,mxbnd
   99 format(' nbound(',i4,') is greater than mxbnd(',i4,')')
c4a-----abnormal stop
      stop
c5------print # of general head bounds this stress period
  100 write(iout,1) nbound
    1 format(i6,' head-dependent boundary nodes')
c6------if there are no general head bounds then return.
      if(nbound.eq.0)return
c7------read data for each general head boundary.
c     write(iout,3)
    3 format(' layer',5x,'row',5x,
     . 'col   elevation   conductance   bound no.')
      do 250 ii=1,nbound
      read(in,4)k,i,j,bnds(4,ii),bnds(5,ii)
      bnds(5,ii)=bnds(5,ii)*fall
    4 format(3i10,2f10.0)
c     write(57,5)k,j,i,bnds(4,ii),bnds(5,ii),ii
    5 format(3i5,f10.0,1pe12.3,3i7)
      bnds(1,ii)=k
      bnds(2,ii)=i
      bnds(3,ii)=j
  250 continue
      return
      end
      subroutine ghblfm(nbound,mxbnd,bnds,hcof,rhs,ibound,
     .    ncol,nrow,nlay)
c     30sep1991
c     add ghb terms to rhs and hcof
c     specifications:
      dimension bnds(5,mxbnd),hcof(ncol,nrow,nlay),
     .      rhs(ncol,nrow,nlay),ibound(ncol,nrow,nlay)
c1------if nbound<=0 then there are no general head bounds. return.
      if(nbound.le.0) return
c2------process each entry in the general head bound list (bnds)
      do 100 l=1,nbound
c3------get column, row and layer of cell containing boundary
      il=bnds(1,l)
      ir=bnds(2,l)
      ic=bnds(3,l)
c4------if the cell is external then skip it.
```

```
      if(ibound(ic,ir,il).le.0) go to 100
c5------since the cell is internal get the boundary data.
      hb=bnds(4,1)
      c=bnds(5,1)
c6------add terms to rhs and hcof
      hcof(ic,ir,il)=hcof(ic,ir,il)-c
      rhs(ic,ir,il)=rhs(ic,ir,il)-c*hb
  100 continue
      return
      end
      subroutine ghb2bd(nbound,mxbnd,vbnm,vbvl,msum,bnds,delt,hnew,
     . ncol,nrow,nlay,ibound,kstp,kper,ighbcb,icbcfl,buff,iout,mbh)
c     30sep1991 ghb2bd
c     calculate volumetric budget for ghb
c     specifications:
      character*4 vbnm,text
      double precision hnew
      dimension vbnm(4,msum),vbvl(4,msum),bnds(5,mxbnd),
     . hnew(ncol,nrow,nlay),ibound(ncol,nrow,nlay),
     . buff(ncol,nrow,nlay)
      dimension text(4)
      data text(1),text(2),text(3),text(4) /' hea','d de','p bo','unds'/
c1------initialize cell-by-cell flow term flag (ibd) and
c1------accumulators (ratin and ratout)
      ibd=0
      ratout=0.
      ratin=0.
c2------if no boundaries then keep zeroes in accumulators.
      if(nbound.eq.0) go to 200
c3------test to see if cell-by-cell flow terms are needed.
      if(icbcfl.eq.0 .or. ighbcb.le.0) go to 10
c3a-----since cell-by-cell flow terms are needed clear buffer & set
c3a-----the flag ibd.
      ibd=1
      do 5 il=1,nlay
      do 5 ir=1,nrow
      do 5 ic=1,ncol
      buff(ic,ir,il)=0.
    5 continue
c4------for each general head bound accumulate flow into aquifer
   10 do 100 l=1,nbound
c5------get layer, row and column of each general head boundary.
      il=bnds(1,1)
      ir=bnds(2,1)
      ic=bnds(3,1)
c     if cell is external, set rate=0.
c     if(ibound(ic,ir,il).le.0) go to 100
      if(ibound(ic,ir,il).le.0)then
       rate=0.
       go to 20
      end if
c7------get parameters from boundary list.
      hhnew=hnew(ic,ir,il)
      hb=bnds(4,1)
```

```
      c=bnds(5,1)
c8------calculate the fow rate into the cell
      rate=c*(hb-hhnew)
c9------print the individual rates if requested(ighbcb<0).
   20 if(mbh.eq.1)write(57,900)(text(n),n=1,4),
     . kper,kstp,1,il,ir,ic,rate
  900 format(1x,4a4,' period',i4,' step',i4,'   ghb',i5,
     . ' k',i4,' i',i4,' j',i4,' q',1p4e13.4)
c10-----if cell-by-cell terms are to be saved then put rate in buffer
      if(ibd.eq.1) buff(ic,ir,il)=buff(ic,ir,il)+rate
c11-----see if flow is into aquifer or out of aquifer.
      if(rate)94,100,96
c12------flow is out of aquifer subtract rate from ratout
   94 ratout=ratout-rate
      go to 100
c13-----flow is into aqifer add rate to ratin
   96 ratin=ratin+rate
  100 continue
c14-----if cell-by-cell terms are to be saved then call
c14-----utility module ubudsv
      if(ibd.eq.1) call ubudsv(kstp,kper,text,ighbcb,buff,ncol,nrow,
     .     nlay,iout)
c15-----move rates, volumes and labels into arrays for printing
  200 vbvl(3,msum)=ratin
      vbvl(1,msum)=vbvl(1,msum)+ratin*delt
      vbvl(4,msum)=ratout
      vbvl(2,msum)=vbvl(2,msum)+ratout*delt
      vbnm(1,msum)=text(1)
      vbnm(2,msum)=text(2)
      vbnm(3,msum)=text(3)
      vbnm(4,msum)=text(4)
c16-----increment the budget term counter
      msum=msum+1
      return
      end
      subroutine rivlal(isum,lenx,lcrivr,mxrivr,nriver,in,iout,
     .                  irivcb)
c     30sep1991
c     allocate array storage for rivers
c     specifications:
c1------identify package and initialize nriver.
      write(iout,1)in
    1 format(' riv, 30sep1991, read on',10i6)
      nriver=0
c2------read & print mxrivr & irivcb(unit or flag for c-b-c flows)
      read(in,2)mxrivr,irivcb
    2 format(2i10)
      write(iout,3)mxrivr
    3 format(' maximum of',i5,' river nodes')
      if(irivcb.gt.0)write(iout,9) irivcb
    9 format(' cell-by-cell flows will be recorded on unit',i3)
      if(irivcb.lt.0)write(iout,8)
    8 format(' cell-by-cell flows will be printed')
c3------set lcrivr equal to address of first unused space in x.
```

```
      lcrivr=isum
c4------calculate amount of space used by river list.
      isp=6*mxrivr
      isum=isum+isp
c5------print amount of space used by river package.
      write (iout,4)isp
    4 format(i9,' elements in x array are used for rivers')
      isuml=isum-1
      write(iout,5)isuml,lenx
    5 format(i9,' elements of x array used out of',i8)
      if(isuml.gt.lenx)write(iout,6)
    6 format(' ***x array must be dimensioned larger***')
      return
      end
      subroutine rivlrp(rivr,nriver,mxrivr,in,iout)
c     30sep1991
c     read river head, conductance and bottom elevation
c     specifications:
      dimension rivr(6,mxrivr)
c1------read itmp(number of river reaches or flag to reuse data)
      read(in,8)itmp
    8 format(i10)
      read(in,22)fall
      if(fall.le.0.)fall=1.
      write(6,24)fall
   22 format(10f10.0)
   24 format(' fcrvr',1p8e13.4)
c2------test itmp.
      if(itmp.ge.0)go to 50
c2a-----if itmp <0 then reuse data from last stress period.
      write(iout,7)
    7 format(' reusing river reaches from last stress period')
      return
c3------if itmp=> zero then it is the number of river reaches
   50 nriver=itmp
c4------if nriver>mxrivr then stop.
      if(nriver.le.mxrivr)go to 100
      write(iout,99)nriver,mxrivr
   99 format(' nriver(',i4,') is greater than mxrivr(',i4,')')
c4a-----abnormal stop.
      stop
c5------print number of river reaches in this stress period.
  100 write(iout,1)nriver
    1 format(i6,' river reaches')
c6------if there are no river reaches then return.
      if(nriver.eq.0)return
c7------read data for each river reach.
c     write(iout,3)
    3 format(' layer',5x,'row',5x,'col   ',
     . '  stage     conductance   bottom elevation   river reach')
      do 250 ii=1,nriver
      read(in,4)k,i,j,rivr(4,ii),rivr(5,ii),rivr(6,ii)
      rivr(5,ii)=rivr(5,ii)*fall
c     write(57,5)k,j,i,rivr(4,ii),rivr(5,ii),rivr(6,ii),ii
```

```
    4 format(3i10,3f10.0)
    5 format(3i5,f10.0,1pe12.3,f10.0,2i7)
      rivr(1,ii)=k
      rivr(2,ii)=i
      rivr(3,ii)=j
  250 continue
      return
      end
      subroutine riv1fm(nriver,mxrivr,rivr,hnew,hcof,rhs,ibound,
     .             ncol,nrow,nlay)
c     30sep1991
c     add river terms to rhs and hcof
c     specifications:
      double precision hnew
      dimension rivr(6,mxrivr),hnew(ncol,nrow,nlay),
     .      hcof(ncol,nrow,nlay),rhs(ncol,nrow,nlay),
     2          ibound(ncol,nrow,nlay)
c1------if nriver<=0 there are no rivers. return.
      if(nriver.le.0)return
c2------process each cell in the river list.
      do 100 l=1,nriver
c3------get column, row, and layer of cell containing reach
      il=rivr(1,l)
      ir=rivr(2,l)
      ic=rivr(3,l)
c4------if the cell is external skip it.
      if(ibound(ic,ir,il).le.0)go to 100
c5------since the cell is internal get the river data.
      hriv=rivr(4,l)
      criv=rivr(5,l)
      rbot=rivr(6,l)
      hhnew=hnew(ic,ir,il)
c6------compare aquifer head to bottom of stream bed.
      if(hhnew.le.rbot)go to 96
c7------since head>bottom add terms to rhs and hcof.
      rhs(ic,ir,il)=rhs(ic,ir,il)-criv*hriv
      hcof(ic,ir,il)=hcof(ic,ir,il)-criv
      go to 100
c8------since head<bottom add term only to rhs.
   96 rhs(ic,ir,il)=rhs(ic,ir,il)-criv*(hriv-rbot)
  100 continue
      return
      end
      subroutine riv2bd(nriver,mxrivr,rivr,ibound,hnew,
     . ncol,nrow,nlay,delt,vbvl,vbnm,msum,kstp,kper,irivcb,
     . icbcfl,buff,iout,mbh)
c     30sep1991 riv2bd
c     calculate volumetric budget for rivers
c     specifications:
      character*4 vbnm,text
      double precision hnew
      dimension rivr(6,mxrivr),ibound(ncol,nrow,nlay),
     . hnew(ncol,nrow,nlay),vbvl(4,20),vbnm(4,20),
     . buff(ncol,nrow,nlay)
```

```
      dimension text(4)
      data text(1),text(2),text(3),text(4) /'   r','iver',' lea','kage'/
c1------initialize cell-by-cell flow term flag (ibd) and
c1------accumulators (ratin and ratout).
      ibd=0
      ratin=0.
      ratout=0.
c2------if no reaches keep zeroes in accumulators.
      if(nriver.eq.0)go to 200
c3------test to see if cell-by-cell flow terms are needed.
      if(icbcfl.eq.0  .or. irivcb.le.0 ) go to 10
c3a-----cell-by-cell flow terms are needed set ibd and clear buffer.
      ibd=1
      do 5 il=1,nlay
      do 5 ir=1,nrow
      do 5 ic=1,ncol
      buff(ic,ir,il)=0.
    5 continue
c4------for each river reach accumulate river flow (steps 5-15)
   10 do 100 l=1,nriver
c5------get layer, row & column of cell containing reach.
      il=rivr(1,l)
      ir=rivr(2,l)
      ic=rivr(3,l)
c     if cell is external, set rate=0.
      if(ibound(ic,ir,il).le.0)then
       rate=0.
       go to 20
      end if
c7------get river parameters from river list.
      hriv=rivr(4,l)
      criv=rivr(5,l)
      rbot=rivr(6,l)
      hhnew=hnew(ic,ir,il)
c8------compare head in aquifer to bottom of riverbed.
c9------aquifer head > bottom then rate=criv*(hriv-hnew).
      if(hhnew.gt.rbot)rate=criv*(hriv-hhnew)
c10-----aquifer head < bottom then rate=criv*(hriv-rbot)
      if(hhnew.le.rbot)rate=criv*(hriv-rbot)
c11-----print the individual rates if requested(irivcb<0).
   20 if(mbh.eq.1)write(57,900)(text(n),n=1,4),
     . kper,kstp,1,il,ir,ic,rate
  900 format(1x,4a4,' period',i4,' step',i4,' reach',i5,
     . ' k',i4,' i',i4,' j',i4,' q',1p4e13.4)
c12-----if c-b-c flow terms are to be saved then add rate to buffer.
      if(ibd.eq.1) buff(ic,ir,il)=buff(ic,ir,il)+rate
c13-----see if flow is into aquifer or into river.
      if(rate)94,100,96
c14-----aquifer is discharging to river subtract rate from ratout.
   94 ratout=ratout-rate
      go to 100
c15-----aquifer is recharged from river add rate to ratin.
   96 ratin=ratin+rate
  100 continue
```

```
c16-----if c-b-c flow terms will be saved call ubudsv to record them.
      if(ibd.eq.1) call ubudsv(kstp,kper,text,irivcb,buff,ncol,nrow,
     .                         nlay,iout)
c17-----move rates,volumes & labels into arrays for printing.
  200 vbvl(3,msum)=ratin
      vbvl(4,msum)=ratout
      vbvl(1,msum)=vbvl(1,msum)+ratin*delt
      vbvl(2,msum)=vbvl(2,msum)+ratout*delt
      vbnm(1,msum)=text(1)
      vbnm(2,msum)=text(2)
      vbnm(3,msum)=text(3)
      vbnm(4,msum)=text(4)
c18-----increment budget term counter
      msum=msum+1
      return
      end
      subroutine wellal(isum,lenx,lcwell,mxwell,nwells,in,iout,
     .     iwelcb)
c     30sep1991
c     allocate array storage for well package
c        specifications:
c1------identify package and initialize nwells
      write(iout,1)in
    1 format(' wel, 30sep1991, read on',10i6)
      nwells=0
c2------read max number of wells and
c2------unit or flag for cell-by-cell flow terms.
      read(in,2) mxwell,iwelcb
    2 format(2i10)
      write(iout,3) mxwell
    3 format(' maximum of',i5,' wells')
      if(iwelcb.gt.0)write(iout,9) iwelcb
    9 format(' cell-by-cell flows will be recorded on unit',i3)
      if(iwelcb.lt.0)write(iout,8)
    8 format(' cell-by-cell flows will be printed when icbcfl not 0')
c3------set lcwell equal to location of well list in x array.
      lcwell=isum
c4------add amount of space used by well list to isum.
      isp=4*mxwell
      isum=isum+isp
c5------print number of spaces in x array used by well package.
      write(iout,4) isp
    4 format(i9,' elements in x array are used for wells')
      isum1=isum-1
      write(iout,5) isum1,lenx
    5 format(i9,' elements of x array used out of',i8)
c6------if there isn't enough space in the x array then print
c6------a warning message.
      if(isum1.gt.lenx)write(iout,6)
    6 format(' ***x array must be dimensioned larger***')
      return
      end
      subroutine wellrp(well,nwells,mxwell,in,iout)
c     30sep1991
```

```
c     read new well locations and stress rates
c        specifications:
      dimension well(4,mxwell)
c1------read itmp(number of wells or flag saying reuse well data)
      read(in,1)itmp
    1 format(i10)
      read(in,22)fall
      if(fall.le.0.)fall=1.
      write(6,24)fall
   22 format(10f10.0)
   24 format(' fcwel',1p8e13.4)
      if(itmp.ge.0) go to 50
c1a-----if itmp less than zero reuse data. print message and return.
      write(iout,6)
    6 format(' reusing wells from last stress period')
      return
c1b-----itmp=>0.  set nwells equal to itmp.
   50 nwells=itmp
      if(nwells.le.mxwell) go to 100
c2------nwells>mxwell.  print message. stop.
      write(iout,99) nwells,mxwell
   99 format(' nwells(',i4,') is greater than mxwell(',i4,')')
      stop
c3------print number of wells in current stress period.
  100 write (iout,2) nwells
    2 format(i5,' wells')
c4------if there are no active wells in this stress period then return
      if(nwells.eq.0)return
c5------read layer,row,column and recharge rate.
c     write(iout,3)
    3 format(' layer    row    col    stress rate   well no.')
      do 250 ii=1,nwells
      read(in,4)k,i,j,q
      q=q*fall
    4 format(3i10,f10.0)
c     write(57,5)k,j,i,q,ii
    5 format(3i5,1pe12.3,4i7)
      well(1,ii)=k
      well(2,ii)=i
      well(3,ii)=j
      well(4,ii)=q
  250 continue
      return
      end
      subroutine wellfm(nwells,mxwell,rhs,well,ibound,
     .      ncol,nrow,nlay)
c     30sep1991
c     subtract q from rhs
c        specifications:
      dimension rhs(ncol,nrow,nlay),well(4,mxwell),
     . ibound(ncol,nrow,nlay)
c1------if number of wells <= 0 then return.
      if(nwells.le.0) return
c2------process each well in the well list.
```

```
      do 100 l=1,nwells
      ir=well(2,l)
      ic=well(3,l)
      il=well(1,l)
      q=well(4,l)
c2a-----if the cell is inactive then bypass processing.
      if(ibound(ic,ir,il).le.0) go to 100
c2b-----if the cell is variable head then subtract q from
c       the rhs accumulator.
      rhs(ic,ir,il)=rhs(ic,ir,il)-q
  100 continue
      return
      end
      subroutine wel2bd(nwells,mxwell,vbnm,vbvl,msum,well,ibound,delt,
     . ncol,nrow,nlay,kstp,kper,iwelcb,icbcfl,buff,iout,mbh)
c     30sep1991 wel2bd
c     calculate volumetric budget for wells
c        specifications:
      character*4 vbnm,text
      dimension vbnm(4,msum),vbvl(4,msum),well(4,mxwell),
     .       ibound(ncol,nrow,nlay),buff(ncol,nrow,nlay)
      dimension text(4)
      data text(1),text(2),text(3),text(4) /'    ','    ','   w','ells'/
c1------clear ratin and ratout accumulators.
      ratin=0.
      ratout=0.
      ibd=0
c2------if there are no wells do not accumulate flow
      if(nwells.eq.0) go to 200
c3------test to see if cell-by-cell flow terms will be recorded.
      if(icbcfl.eq.0 .or. iwelcb.le.0) go to 60
c4------if cell-by-cell flows will be saved then clear the buffer.
      ibd=1
      do 50 il=1,nlay
      do 50 ir=1,nrow
      do 50 ic=1,ncol
      buff(ic,ir,il)=0.
   50 continue
c5------process wells one at a time.
   60 do 100 l=1,nwells
      ir=well(2,l)
      ic=well(3,l)
      il=well(1,l)
      q=well(4,l)
c     if cell is external, set q=0.
      if(ibound(ic,ir,il).le.0)q=0.
c5b-----print the individual rates if requested(iwelcb<0).
      if(mbh.eq.1)write(57,900)(text(n),n=1,4),
     . kper,kstp,1,il,ir,ic,q
  900 format(1x,4a4,' period',i4,' step',i4,'  well',i5,
     . ' k',i4,' i',i4,' j',i4,' q',1p4e13.4)
c5c-----if cell-by-cell flows are to be saved then add them to buffer.
      if(ibd.eq.1) buff(ic,ir,il)=buff(ic,ir,il)+q
      if(q) 90,100,80
```

```
c5d-----pumping rate is positive(recharge). add it to ratin.
   80 ratin=ratin+q
      go to 100
c5e-----pumping rate is negative(discharge). add it to ratout.
   90 ratout=ratout-q
  100 continue
c6------if cell-by-cell flows will be saved call ubudsv to record them
      if(ibd.eq.1) call ubudsv(kstp,kper,text,iwelcb,buff,ncol,nrow,
     .      nlay,iout)
c7------move rates into vbvl for printing by module baslot.
  200 vbvl(3,msum)=ratin
      vbvl(4,msum)=ratout
c8------move rates times time step length into vbvl accumulators.
      vbvl(1,msum)=vbvl(1,msum)+ratin*delt
      vbvl(2,msum)=vbvl(2,msum)+ratout*delt
c9------move budget term labels into vbnm for printing.
      vbnm(1,msum)=text(1)
      vbnm(2,msum)=text(2)
      vbnm(3,msum)=text(3)
      vbnm(4,msum)=text(4)
c10-----increment budget term counter(msum).
      msum=msum+1
      return
      end
      subroutine sip1a1(isum,lenx,lce1,lcf1,lcg1,lcv,lchdcg,lc1rch,
     .    lcw,mxiter,nparm,ncol,nrow,nlay,in,iout)
c     30sep1991
c     allocate storage in the x array for sip arrays
c        specifications:
c1------print a message identifying sip package
      write(iout,1)in
    1 format(' sip, 30sep1991, read on',10i6)
c2------read and print mxiter and nparm
      read(in,2) mxiter,nparm
    2 format(2i10)
      write(iout,3) mxiter,nparm
    3 format(' maximum of',i4,' iterations allowed for closure'/
     1       1x,i2,' iteration parameters')
c3------allocate space for the sip arrays
      isold=isum
      nrc=nrow*ncol
      isiz=nrc*nlay
      lce1=isum
      isum=isum+isiz
      lcf1=isum
      isum=isum+isiz
      lcg1=isum
      isum=isum+isiz
      lcv=isum
      isum=isum+isiz
      lchdcg=isum
      isum=isum+mxiter
      lc1rch=isum
      isum=isum+3*mxiter
```

```
      lcw=isum
      isum=isum+nparm
c4------calculate and print the space used in the x array
      isp=isum-isold
      write(iout,4) isp
    4 format(i9,' elements in x array are used by sip')
      isum1=isum-1
      write(iout,5) isum1,lenx
    5 format(i9,' elements of x array used out of',i8)
      if(isum1.gt.lenx) write(iout,6)
    6 format(' ***x array must be dimensioned larger***')
      return
      end
      subroutine siplrp(nparm,mxiter,accl,hclose,w,in,ipcalc,iprsip,
     . iout,tiny)
c     30sep1991
c     read data for sip
c        specifications:
      dimension w(nparm)
c     read accl,hclose,wseed,ipcalc,iprsip,tiny
      read(in,1) accl,hclose,ipcalc,wseed,iprsip,tiny
    1 format(2f10.0,i10,f10.0,i10,7f10.0)
      if(accl.eq.0.) accl=1.
c     if field for tiny be negative or blank, set it to a reasonable value
      if(tiny.le.0.)tiny=1.e-19
c2------print data values just read
      write(iout,100)
  100 format(' solution by the strongly implicit procedure')
      write(iout,115) mxiter
  115 format(' maximum iterations allowed for closure =',i9)
      write(iout,120) accl
  120 format(' acceleration parameter =',1p5e13.4)
      write(iout,125) hclose
  125 format(' head change criterion for closure =',1p3e14.5)
      if(iprsip.le.0)iprsip=999
      write(iout,130) iprsip
  130 format(' sip head change printout interval =',i9)
      write(iout,2)tiny
c3------check if specified value of wseed should be used or if
c3------seed should be calculated
      if(ipcalc.eq.0) go to 150
c3a-----calculate seed & iteration parameters prior to 1st iteration
      write(iout,140)
  140 format(' calculate iteration parameters from model',
     1' calculated wseed')
      return
c     go to 1000
c3b-----use specified value of wseed
c3b-----calculate and print iteration parameters
  150 p1=-1.
      p2=nparm-1
      do 160 i=1,nparm
      p1=p1+1.
  160 w(i)=1.-wseed**(p1/p2)
```

```
      write(iout,161) nparm,wseed,(w(j),j=1,nparm)
  161 format(i5,' iteration parameters calculated from',
     . ' specified wseed =',1pe14.5,/,(1p8e14.5))
    2 format(' test for zero-divide or underflow (tiny):',1p6e12.3)
      return
      end
      subroutine sip2ap(hnew,ibound,cr,cc,cv,hcof,rhs,e1,f1,g1,v,
     . w,hdcg,lrch,nparm,kiter,hclose,acc1,icnvg,kstp,kper,ipcalc,
     . iprsip,mxiter,nstp,ncol,nrow,nlay,klm1,nodes,ndsm1,iout,tiny)
c     30sep1991 sip2ap
c     solution by the strongly implicit procedure -- 1 iteration
c        specifications:
      double precision hnew,ditpar,ac,hhcof,rrhs,xi,dzero,done,res
      double precision z,b,d,e,f,h,s,ap,tp,cp,gp,up,rp
      double precision zhnew,bhnew,dhnew,fhnew,hhnew,shnew
      double precision a1,b1,c1,d1,elnc1,flnc1,glnc1
      double precision elnr1,flnr1,glnr1,eln11,fln11,gln11
      double precision vnr1,vnc1,vn11,elxi,flxi,glxi,vn,hcfhnw
      dimension hnew(nodes),ibound(nodes),cr(nodes),cc(nodes),
     . cv(ndsm1),hcof(nodes),rhs(nodes),e1(nodes),f1(nodes),
     . g1(nodes),v(nodes),w(nparm),hdcg(mxiter),lrch(3,mxiter)
c1------calculate iteration parameters if flag is set.  then
c1------clear the flag so that calculation is done only once.
      if(ipcalc.ne.0)call
     . ssip2i(cr,cc,cv,ibound,nparm,w,ncol,nrow,nlay,klm1,iout)
      ipcalc=0
c2------assign values to fields that are constant during an iteration
      dzero=0.
      done=1.
      ac=acc1
      nrc=nrow*ncol
      nth=mod(kiter-1,nparm)+1
      ditpar=w(nth)
c3------initialize variable that tracks maximum head change during
c3------the iteration
      bigg=0.
c4------clear sip work arrays.
      do 100 i=1,nodes
      e1(i)=0.
      f1(i)=0.
      g1(i)=0.
  100 v(i)=0.
c5------set normal/reverse equation ordering flag (1 or -1) and
c5------calculate indexes dependent on ordering
      idir=1
      if(mod(kiter,2).eq.0)idir=-1
      idnrc=idir*nrc
      idncol=idir*ncol
      if(kiter.eq.1)write(6,2)
c6------step through cells calculating intermediate vector v
c6------using forward substitution
      do 150 k=1,nlay
      do 150 i=1,nrow
      do 150 j=1,ncol
```

```
c6a-----set up current cell location indexes.  these are dependent
c6a-----on the direction of equation ordering.
      if(idir.le.0)go to 120
      ii=i
      jj=j
      kk=k
      go to 122
  120 ii=nrow-i+1
      jj=j
      kk=nlay-k+1
c6b-----calculate 1 dimensional subscript of current cell and
c6b-----skip calculations if cell is noflow or constant head
  122 n=jj+(ii-1)*ncol+(kk-1)*nrc
      if(ibound(n).le.0)go to 150
c6c-----calculate 1 dimensional subscripts for locating the 6
c6c-----surrounding cells
      nrn=n+idncol
      nrl=n-idncol
      ncn=n+1
      ncl=n-1
      nln=n+idnrc
      nll=n-idnrc
c6d-----calculate 1 dimensional subscripts for conductance to the 6
c6d-----surrounding cells.  these depend on ordering of equations.
      if(idir.le.0)go to 124
      ncf=n
      ncd=ncl
      nrb=nrl
      nrh=n
      nls=n
      nlz=nll
      go to 126
  124 ncf=n
      ncd=ncl
      nrb=n
      nrh=nrn
      nls=nln
      nlz=n
c6e-----assign variables in matrices a & u involving adjacent cells
c6e1----neighbor is 1 row back
  126 b=dzero
      elnrl=dzero
      flnrl=dzero
      glnrl=dzero
      bhnew=dzero
      vnrl=dzero
      if(i.eq.1) go to 128
      b=cc(nrb)
      elnrl=el(nrl)
      flnrl=fl(nrl)
      glnrl=gl(nrl)
      bhnew=b*hnew(nrl)
      vnrl=v(nrl)
c6e2----neighbor is 1 row ahead
```

```
  128 h=dzero
      hhnew=dzero
      if(i.eq.nrow) go to 130
      h=cc(nrh)
      hhnew=h*hnew(nrn)
c6e3----neighbor is 1 column back
  130 d=dzero
      elncl=dzero
      flncl=dzero
      glncl=dzero
      dhnew=dzero
      vncl=dzero
      if(j.eq.1) go to 132
      d=cr(ncd)
      elncl=el(ncl)
      flncl=fl(ncl)
      glncl=gl(ncl)
      dhnew=d*hnew(ncl)
      vncl=v(ncl)
c6e4----neighbor is 1 column ahead
  132 f=dzero
      fhnew=dzero
      if(j.eq.ncol) go to 134
      f=cr(ncf)
      fhnew=f*hnew(ncn)
c     first active layer behind
  134 z=dzero
      elnll=dzero
      flnll=dzero
      glnll=dzero
      zhnew=dzero
      vnll=dzero
      if(k.gt.1)then
c      find first active cell in layers behind
       do 10 kl=1,nlay
       kbe=k-kl
       if(kbe.lt.1)go to 30
       nll=n-idnrc*kl
       if(ibound(nll).ne.0)then
        nlz=nll
        if(idir.lt.0)nlz=n
        go to 20
       end if
   10  continue
   20  z=cv(nlz)
       elnll=el(nll)
       flnll=fl(nll)
       glnll=gl(nll)
       zhnew=z*hnew(nll)
       vnll=v(nll)
      end if
c     first active layer ahead
   30 s=dzero
      shnew=dzero
```

```
      if(k.lt.nlay)then
c      find first active cell in layers ahead
       do 40 kl=1,nlay
       kah=k+kl
       if(kah.gt.nlay)go to 60
       nln=n+idnrc*kl
       if(ibound(nln).ne.0)then
        nls=n
        if(idir.lt.0)nls=nln
        go to 50
       end if
   40  continue
   50  s=cv(nls)
       shnew=s*hnew(nln)
      end if
c6e7----calculate the negative sum of all conductances to neighboring
c6e7----cells
   60 e=-z-b-d-f-h-s
c6f-----calculate components of the upper and lower matrices, which
c6f-----are the factors of matrix (a+b)
      al=z/(done+ditpar*(elnll+flnll))
      bl=b/(done+ditpar*(elnrl+glnrl))
      cl=d/(done+ditpar*(flncl+glncl))
      ap=al*elnll
      cp=bl*elnrl
      gp=cl*flncl
      rp=cl*glncl
      tp=al*flnll
      up=bl*glnrl
      hhcof=hcof(n)
      dl=e+hhcof+ditpar*(ap+tp+cp+gp+up+rp)-al*glnll-bl*flnrl-cl*elncl
c     test for zero-divide or underflow
      dlab=dabs(dl)
      if(dlab.gt.tiny)then
       el(n)=(f-ditpar*(ap+cp))/dl
       fl(n)=(h-ditpar*(tp+gp))/dl
       gl(n)=(s-ditpar*(rp+up))/dl
c6g    calculate the residual
       rrhs=rhs(n)
       hnw=hnew(n)
       hcfhnw=hnw*hcof(n)
       res=rrhs-zhnew-bhnew-dhnew-e*hnew(n)-hcfhnw-fhnew-hhnew-shnew
c6h    calculate the intermediate vector v
       v(n)=(ac*res-al*vnll-bl*vnrl-cl*vncl)/dl
       go to 150
      end if
c     write & stop before zero-divide or underflow
      write(iout,4)kk,ii,j,kiter,dl,dlab
      stop
  150 continue
      smchk=0.
c7------step through each cell and solve for head change by back
c7------substitution
      do 160 k=1,nlay
```

```
      do 160 i=1,nrow
      do 160 j=1,ncol
c7a-----set up current cell location indexes.  these are dependent
c7a-----on the direction of equation ordering.
      if(idir.lt.0) go to 152
      kk=nlay-k+1
      ii=nrow-i+1
      jj=ncol-j+1
      go to 154
  152 kk=k
      ii=i
      jj=ncol-j+1
c7b-----calculate 1 dimensional subscript of current cell and
c7b-----skip calculations if cell is noflow or constant head
  154 n=jj+(ii-1)*ncol+(kk-1)*nrc
      if(ibound(n).le.0)go to 160
c7c-----calculate 1 dimensional subscripts for the 3 neighboring cells
c7c-----behind (relative to the direction of the back substitution
c7c-----ordering) the currrent cell.
      nc=n+1
      nr=n+idncol
c     find first active cell in layers behind
      do 70 kl=1,nlay
      kbe=k-kl
      if(kbe.lt.1)go to 80
      nl=n+idnrc*kl
      if(ibound(nl).ne.0)go to 80
   70 continue
c7d-----back substitute, storing head change in array v in place of
c7d-----intermediate forward substitution values.
   80 elxi=dzero
      flxi=dzero
      glxi=dzero
      if(jj.ne.ncol) elxi=el(n)*v(nc)
      if(i.ne.1) flxi=fl(n)*v(nr)
      if(k.ne.1) glxi=gl(n)*v(nl)
      vn=v(n)
      v(n)=vn-elxi-flxi-glxi
c7e-----get the absolute head change. if it is max over grid so far.
c7e-----then save it along with cell indices and head change.
      tchk=abs(v(n))
      smchk=smchk+tchk
      if (tchk.le.bigg) go to 155
      bigg=tchk
      big=v(n)
      ib=ii
      jb=jj
      kb=kk
c7f-----add head change this iteration to head from the previous
c7f-----iteration to get a new estimate of head.
  155 xi=v(n)
      hnew(n)=hnew(n)+xi
  160 continue
      write(6,6)kiter,kb,ib,jb,big,smchk
```

```
c8------store the largest absolute head change (this iteration) and
c8------and its location.
      hdcg(kiter)=big
      lrch(1,kiter)=kb
      lrch(2,kiter)=ib
      lrch(3,kiter)=jb
      icnvg=0
      if(bigg.le.hclose) icnvg=1
c9------if end of time step, print # of iterations this step
      if(icnvg.eq.0.and.kiter.ne.mxiter)return
c     if(kstp.eq.1) write(iout,500)
c 500 format(1h0)
      write(iout,501) kiter,kstp,kper
  501 format(i5,' iterations for time step',i4,' in stress period',
     1        i3)
c10-----print head change each iteration if printout interval is reached
      if(icnvg.eq.0 .or. kstp.eq.nstp .or. mod(kstp,iprsip).eq.0)
     1      call ssiplp(hdcg,lrch,kiter,mxiter,iout)
    2 format(' nmit   k   i   j     max chng   cmltv chng')
    4 format(' dlab < tiny',4i4,1p8e12.3)
    6 format(i5,3i4,1p8e13.4)
      return
      end
      subroutine ssip2i(cr,cc,cv,ibound,nparm,w,ncol,nrow,nlay,
     . klml,iout)
c     30sep1991 ssip2i
c     calculate an iteration parameter seed and use it to calculate sip
c     iteration parameters
c        specifications:
      dimension cr(ncol,nrow,nlay),cc(ncol,nrow,nlay),
     . cv(ncol,nrow,klml),ibound(ncol,nrow,nlay),w(nparm)
      double precision dwmin,avgsum
c1------calculate constants and initialize variables
      piepie=9.869604
      r=nrow
      c=ncol
      zl=nlay
      ccol=piepie/(2.*c*c)
      crow=piepie/(2.*r*r)
      clay=piepie/(2.*zl*zl)
      wminmn=1.
      avgsum=0.
      nodes=0
c2------loop through all cells, calculating a seed for each cell
c2------that is active
      do 100 k=1,nlay
      do 100 i=1,nrow
      do 100 j=1,ncol
      if(ibound(j,i,k).le.0) go to 100
c2a-----conductance from this cell
c2a-----to each of the 6 adjacent cells
      d=0.
      if(j.ne.1) d=cr(j-1,i,k)
      f=0.
```

```
      if(j.ne.ncol) f=cr(j,i,k)
      b=0.
      if(i.ne.1) b=cc(j,i-1,k)
      h=0.
      if(i.ne.nrow) h=cc(j,i,k)
      z=0.
      if(k.gt.1)then
c      find first active cell above
       do 10 kl=1,nlay
       kup=k-kl
       if(kup.gt.0)then
        if(ibound(j,i,kup).ne.0)then
         z=cv(j,i,kup)
         go to 20
        end if
       end if
   10  continue
      end if
   20 s=0.
      if(k.ne.nlay) s=cv(j,i,k)
c2b-----find the maximum and minimum of the 2 conductance coefficients
c2b-----in each principal coordinate direction
      dfmx=amaxl(d,f)
      bhmx=amaxl(b,h)
      zsmx=amaxl(z,s)
      dfmn=aminl(d,f)
      bhmn=aminl(b,h)
      zsmn=aminl(z,s)
      if(dfmn.eq.0.) dfmn=dfmx
      if(bhmn.eq.0.) bhmn=bhmx
      if(zsmn.eq.0.) zsmn=zsmx
c2c-----calculate a seed in each principal coordinate direction
      wcol=1.
      if(dfmn.ne.0.) wcol=ccol/(1.+(bhmx+zsmx)/dfmn)
      wrow=1.
      if(bhmn.ne.0.) wrow=crow/(1.+(dfmx+zsmx)/bhmn)
      wlay=1.
      if(zsmn.ne.0.) wlay=clay/(1.+(dfmx+bhmx)/zsmn)
c2d-----select the cell seed, which is the minimum seed of the 3.
c2d-----select the minimum seed over the whole grid.
      wmin=aminl(wcol,wrow,wlay)
      wminmn=aminl(wminmn,wmin)
c2e-----add the cell seed to the accumulator avgsum for use
c2e-----in getting the average seed.
      dwmin=wmin
      avgsum=avgsum+dwmin
      nodes=nodes+1
  100 continue
c3------calculate the average seed of the cell seeds, and print
c3------the average and minimum seeds.
      tmp=nodes
      avgmin=avgsum
      avgmin=avgmin/tmp
      write(iout,101) avgmin,wminmn
```

```
  101 format(' average seed =',1pe14.5,/,' minimum seed =',1pe14.5)
c4------calculate and print iteration parameters from the average seed
      pl=-1.
      p2=nparm-1
      do 50 i=1,nparm
      pl=pl+1.
   50 w(i)=1.-avgmin**(pl/p2)
      write(iout,150) nparm,(w(j),j=1,nparm)
  150 format(i6,' iteration parameters calculated from',
     . ' average seed:',/,(1p8e14.5))
      return
      end
      subroutine ssiplp(hdcg,lrch,kiter,mxiter,iout)
c     30sep1991
c     print maximum head change for each iteration during a time step
c        specifications:
      dimension hdcg(mxiter), lrch(3,mxiter)
      write(iout,5)
    5 format(' maximum head change for each iteration:',/,
     .   5(' head change layer,row,col'))
      write (iout,10) (hdcg(j),(lrch(i,j),i=1,3),j=1,kiter)
   10 format((1x,5(g12.4,' (',i3,',',i3,',',i3,')')))
      return
      end
      subroutine evtlal(isum,lenx,lcievt,lcevtr,lcexdp,lcsurf,
     .              ncol,nrow,nevtop,in,iout,ievtcb)
c     30sep1991
c     allocate array storage for evapotranspiration
c        specifications:
c1------identify package.
      write(iout,1)in
    1 format(' evt, 30sep1991, read on',1016)
c2------read nevtop and ievtcb.
      read(in,3)nevtop,ievtcb
    3 format(2i10)
c3------check to see that et option is legal.
      if(nevtop.ge.1.and.nevtop.le.2)go to 200
c3a-----if illegal print a message & abort simulation.
      write(iout,8)
    8 format(' illegal et option code. simulation aborting')
      stop
c4------if the option is legal then print the option code.
  200 if(nevtop.eq.1) write(iout,201)
  201 format(' option 1 -- evapotranspiration from top layer')
      if(nevtop.eq.2) write(iout,202)
  202 format(' option 2 -- evapotranspiration from one specified',
     1      ' node in each vertical column')
      irk=isum
c5------if cell-by-cell terms to be saved then print unit number.
      if(ievtcb.gt.0) write(iout,203) ievtcb
  203 format(' cell-by-cell flow terms will be saved on unit',i3)
c6------allocate space for the arrays evtr, exdp and surf.
      lcevtr=isum
      isum=isum+ncol*nrow
```

```
      lcexdp=isum
      isum=isum+ncol*nrow
      lcsurf=isum
      isum=isum+ncol*nrow
c7------if option 2 then allocate space for the indicator array(ievt)
      lcievt=isum
      if(nevtop.ne.2)go to 300
      isum=isum+ncol*nrow
c8------calculate & print amount of space used by et package.
  300 irk=isum-irk
      write(iout,4)irk
    4 format(i9,' elements of x array used for evapotranspiration')
      isum1=isum-1
      write(iout,5)isum1,lenx
    5 format(i9,' elements of x array used out of',i8)
      if(isum1.gt.lenx)write(iout,6)
    6 format(' ***x array must be made larger***')
      return
      end
      subroutine evtlrp(nevtop,ievt,evtr,exdp,surf,delr,delc,
     .                  ncol,nrow,in,iout)
c     30sep1991
c     read evapotranspiration data
c        specifications:
      character*4 aname
      dimension ievt(ncol,nrow),evtr(ncol,nrow),exdp(ncol,nrow),
     . surf(ncol,nrow),aname(6,4),delr(ncol),delc(nrow)
      data aname(1,1),aname(2,1),aname(3,1),aname(4,1),aname(5,1),
     1  aname(6,1) /'    ','    ','  et',' lay','er i','ndex'/
      data aname(1,2),aname(2,2),aname(3,2),aname(4,2),aname(5,2),
     1  aname(6,2) /'    ','    ','    ','  et',' sur','face'/
      data aname(1,3),aname(2,3),aname(3,3),aname(4,3),aname(5,3),
     1  aname(6,3) /' eva','potr','ansp','irat','ion ','rate'/
      data aname(1,4),aname(2,4),aname(3,4),aname(4,4),aname(5,4),
     1  aname(6,4) /'    ','    ','exti','ncti','on d','epth'/
c1------read flags showing whether data is to be reused.
      read(in,6)insurf,inevtr,inexdp,inievt
    6 format(4i10)
c2------test insurf to see where surface elevation comes from.
      if(insurf.ge.0)go to 32
c2a-----if insurf<0 then reuse surface array from last stress period
      write(iout,3)
    3 format(' reusing surf from last stress period')
      go to 35
c3-------if insurf=>0 then call module u2drel to read surface.
   32 call u2drel(surf,aname(1,2),nrow,ncol,0,in,iout)
c4------test inevtr to see where max et rate comes from.
   35 if(inevtr.ge.0)go to 37
c4a-----if inevtr<0 then reuse max et rate.
      write(iout,4)
    4 format(' reusing evtr from last stress period')
      go to 45
c5------if inevtr=>0 call module u2drel to read max et rate.
   37 call u2drel(evtr,aname(1,3),nrow,ncol,0,in,iout)
```

```
c6------multiply max et rate by cell area to get volumetric rate
      do 40 ir=1,nrow
      do 40 ic=1,ncol
      evtr(ic,ir)=evtr(ic,ir)*delr(ic)*delc(ir)
   40 continue
c7------test inexdp to see where extinction depth comes from
   45 if(inexdp.ge.0)go to 47
c7a-----if inexdp<0 reuse extinction depth from last stress period
      write(iout,5)
    5 format(' reusing exdp from last stress period')
      go to 48
c8-------if inexdp=>0 call module u2drel to read extinction depth
   47 call u2drel(exdp,aname(1,4),nrow,ncol,0,in,iout)
c9------if option(nevtop) is 2 then we need an indicator array.
   48 if(nevtop.ne.2)return
c10-----if inievt<0 then reuse layer indicator array.
      if(inievt.ge.0)go to 49
      write(iout,2)
    2 format(' reusing ievt from last stress period')
      return
c11-----if inievt=>0 then call module u2dint to read indicator array.
   49 call u2dint(ievt,aname(1,1),nrow,ncol,0,in,iout)
      return
      end
      subroutine evt1fm(nevtop,ievt,evtr,exdp,surf,rhs,hcof,
     .                  ibound,hnew,ncol,nrow,nlay)
c     30sep1991
c        add evapotranspiration to rhs and hcof
c        specifications:
      double precision hnew
      dimension ievt(ncol,nrow),evtr(ncol,nrow),exdp(ncol,nrow),
     .         surf(ncol,nrow),rhs(ncol,nrow,nlay),
     2           hcof(ncol,nrow,nlay),ibound(ncol,nrow,nlay),
     3           hnew(ncol,nrow,nlay)
c1------process each horizontal cell location
      do 10 ir=1,nrow
      do 10 ic=1,ncol
c2------set the layer index equal to 1
      il=1
c3------if option 2 is specified then get layer index from ievt array
      if(nevtop.eq.2)il=ievt(ic,ir)
c4------if the cell is external ignore it.
      if(ibound(ic,ir,il).le.0)go to 10
      c=evtr(ic,ir)
      s=surf(ic,ir)
      h=hnew(ic,ir,il)
c5------if aquifer head is greater than or equal to surf, et is constant
      if(h.lt.s) go to 5
c5a-----subtract -evtr from rhs
      rhs(ic,ir,il)=rhs(ic,ir,il) + c
      go to 10
c6------if depth to water>=extinction depth then et is 0
    5 d=s-h
      x=exdp(ic,ir)
```

```
      if(d.ge.x)go to 10
c7------linear range. add et terms to both rhs and hcof.
      rhs(ic,ir,il)=rhs(ic,ir,il)+c-c*s/x
      hcof(ic,ir,il)=hcof(ic,ir,il)-c/x
   10 continue
      return
      end
      subroutine evtlbd(nevtop,ievt,evtr,exdp,surf,ibound,hnew,
     .          ncol,nrow,nlay,delt,vbvl,vbnm,msum,kstp,kper,
     .          ievtcb,icbcfl,buff,iout)
c     30sep1991
c     calculate volumetric budget for evapotranspiration
c        specifications:
      character*4 vbnm,text
      double precision hnew
      dimension ievt(ncol,nrow),evtr(ncol,nrow),exdp(ncol,nrow),
     .         surf(ncol,nrow),ibound(ncol,nrow,nlay),
     2          vbvl(4,20),vbnm(4,20),hnew(ncol,nrow,nlay),
     3          buff(ncol,nrow,nlay)
      dimension text(4)
      data text(1),text(2),text(3),text(4) /'    ','    ','    ','  et'/
c1------clear the rate accumulator.
      ratout=0
c2------if cell-by-cell flow terms will be saved then clear the buffer.
      ibd=0
      if(ievtcb.le.0 .or. icbcfl.eq.0) go to 5
      ibd=1
      do 4 il=1,nlay
      do 4 ir=1,nrow
      do 4 ic=1,ncol
      buff(ic,ir,il)=0.
    4 continue
c3------process each horizontal cell location
    5 do 10 ir=1,nrow
      do 10 ic=1,ncol
c4------set the layer index equal to 1
      il=1
c5------if option 2 is specified then get layer index from ievt array
      if(nevtop.eq.2)il=ievt(ic,ir)
c6------if cell is external then ignore it.
      if(ibound(ic,ir,il).le.0)go to 10
      c=evtr(ic,ir)
      s=surf(ic,ir)
      h=hnew(ic,ir,il)
c7------if aquifer head => surf,set q=max et rate
      if(h.lt.s) go to 7
      q=-c
      go to 9
c8------if depth=>extinction depth, et is 0
    7 x=exdp(ic,ir)
      d=s-h
      if(d.ge.x)go to 10
c9------linear range . q=-evtr(h-exel)/exdp
      q=c*d/x-c
```

```
c10-----accumulate total flow rate
    9 ratout=ratout-q
c11-----if cell-by-cell flow terms to be saved the add q to buffer.
      if(ibd.eq.1) buff(ic,ir,il)=q
   10 continue
c12-----if c-b-c to be saved call module ubudsv to record them.
      if(ibd.eq.1) call ubudsv(kstp,kper,text,ievtcb,buff,ncol,nrow,
     .                         nlay,iout)
c13-----move total et rate into vbvl for printing by baslot.
      vbvl(3,msum)=0.
      vbvl(4,msum)=ratout
c14-----add et(et_rate times step length) to vbvl
      vbvl(1,msum)=0.
      vbvl(2,msum)=vbvl(2,msum)+ratout*delt
c15-----move budget term labels to vbnm for print by module baslot
      vbnm(1,msum)=text(1)
      vbnm(2,msum)=text(2)
      vbnm(3,msum)=text(3)
      vbnm(4,msum)=text(4)
c16-----increment budget term counter
      msum=msum+1
      return
      end
      subroutine rchlal(isum,lenx,lcirch,lcrech,nrchop,
     .                  ncol,nrow,in,iout,irchcb)
c     30sep1991
c     allocate array storage for recharge
c        specifications:
c1------identify package.
      write(iout,1)in
    1 format(' rch, 30sep1991, read on',1016)
c2------read nrchop and irchcb.
      read(in,2)nrchop,irchcb
    2 format(2i10)
c3------check to see that option is legal.
      if(nrchop.ge.1.and.nrchop.le.3)go to 200
c3a-----if illegal print a message and abort simulation
      write(iout,8)
    8 format(' illegal option code. simulation aborting')
      stop
c4------if option is legal print option code.
  200 irk=isum
      if(nrchop.eq.1) write(iout,201)
  201 format(' option 1 -- recharge to top layer')
      if(nrchop.eq.2) write(iout,202)
  202 format(' option 2 -- recharge to one specified node in each',
     1       ' vertical column')
      if(nrchop.eq.3) write(iout,203)
  203 format(' option 3 -- recharge to highest active node in each',
     1       ' vertical column')
c5------if cell-by-cell flow  terms to be saved then print unit #
      if(irchcb.gt.0) write(iout,204) irchcb
  204 format(' cell-by-cell flow terms will be recorded on unit',i3)
c6------allocate space for the recharge array(rech).
```

```
      lcrech=isum
      isum=isum+ncol*nrow
c7------if option 2 then allocate space for indicator array(irch)
      lcirch=isum
      if(nrchop.ne.2)go to 300
      isum=isum+ncol*nrow
c8------calculate and print amount of space used by recharge.
  300 irk=isum-irk
      write(iout,4)irk
    4 format(i9,' elements of x array used for recharge')
      isum1=isum-1
      write(iout,5)isum1,lenx
    5 format(i9,' elements of x array used out of',i8)
      if(isum1.gt.lenx)write(iout,6)
    6 format(' ***x array must be made larger***')
      return
      end
      subroutine rch1rp(nrchop,irch,rech,delr,delc,nrow,ncol,
     .                  in,iout)
c     30sep1991
c     read recharge rates
c        specifications:
      character*4 aname
      dimension irch(ncol,nrow),rech(ncol,nrow),
     . aname(6,2),delr(ncol),delc(nrow)
      data aname(1,1),aname(2,1),aname(3,1),aname(4,1),aname(5,1),
     1  aname(6,1) /'    ','rech','arge',' lay','er i','ndex'/
      data aname(1,2),aname(2,2),aname(3,2),aname(4,2),aname(5,2),
     1  aname(6,2) /'    ','    ','    ','    ','rech','arge'/
c1------read flags showing whether data is to be reused.
      read(in,4)inrech,inirch
    4 format(2i10)
c2------test inrech to see where rech is coming from.
      if(inrech.ge.0)go to 32
c2a-----if inrech<0 then reuse recharge array from last stress period
      write(iout,3)
    3 format(' reusing rech from last stress period')
      go to 55
c3------if inrech=>0 then call u2drel to read recharge rate.
   32 call u2drel(rech,aname(1,2),nrow,ncol,0,in,iout)
c4------multiply recharge rate by cell area to get volumetric rate.
      do 50 ir=1,nrow
      do 50 ic=1,ncol
      rech(ic,ir)=rech(ic,ir)*delr(ic)*delc(ir)
   50 continue
c5------if nrchop=2 then a layer indicator array is needed.
   55 if (nrchop.ne.2)return
c6------if inirch<0 then reuse layer indicator array.
      if(inirch.ge.0)go to 58
      write(iout,2)
    2 format(' reusing irch from last stress period')
      return
c7------if inirch=>0 call u2dint to read layer ind array(irch)
   58 call u2dint(irch,aname(1,1),nrow,ncol,0,in,iout)
```

```
      return
      end
      subroutine rch1fm(nrchop,irch,rech,rhs,ibound,ncol,
     .                  nrow,nlay)
c     30sep1991
c     subtract recharge from rhs
c        specifications:
      dimension irch(ncol,nrow),rech(ncol,nrow),
     .        rhs(ncol,nrow,nlay),ibound(ncol,nrow,nlay)
c1------if nrchop is 1 recharge is in top layer. layer index is 1.
      if(nrchop.ne.1) go to 15
      do 10 ir=1,nrow
      do 10 ic=1,ncol
c1a-----if cell is external there is no recharge into it.
      if(ibound(ic,ir,1).le.0)go to 10
c1b-----subtract recharge rate from right-hand-side.
      rhs(ic,ir,1)=rhs(ic,ir,1)-rech(ic,ir)
   10 continue
      go to 100
c2------if option is 2 then recharge is into layer in indicator array
   15 if(nrchop.ne.2)go to 25
      do 20 ir=1,nrow
      do 20 ic=1,ncol
c2a-----layer index is in indicator array.
      il=irch(ic,ir)
c2b-----if the cell is external there is no recharge into it.
      if(ibound(ic,ir,il).le.0)go to 20
c2c-----subtract recharge from right-hand-side.
      rhs(ic,ir,il)=rhs(ic,ir,il)-rech(ic,ir)
   20 continue
      go to 100
c3------if option is 3 recharge is into highest internal cell.
   25 if(nrchop.ne.3)go to 100
c        cannot pass through constant head node
      do 30 ir=1,nrow
      do 30 ic=1,ncol
      do 28 il=1,nlay
c3a-----if cell is constant head move on to next horizontal location.
      if(ibound(ic,ir,il).lt.0) go to 30
c3b-----if cell is inactive move down a layer.
      if (ibound(ic,ir,il).eq.0)go to 28
c3c-----subtract recharge from right-hand-side.
      rhs(ic,ir,il)=rhs(ic,ir,il)-rech(ic,ir)
      go to 30
   28 continue
   30 continue
  100 continue
      return
      end
      subroutine rch1bd(nrchop,irch,rech,ibound,nrow,ncol,nlay,
     .   delt,vbvl,vbnm,msum,kstp,kper,irchcb,icbcfl,buff,iout)
c     30sep1991
c     calculate volumetric budget for recharge
c        specifications:
```

```
      character*4 vbnm,text
      dimension irch(ncol,nrow),rech(ncol,nrow),
     .          ibound(ncol,nrow,nlay),buff(ncol,nrow,nlay),
     2          vbvl(4,20),vbnm(4,20)
      dimension text(4)
      data text(1),text(2),text(3),text(4) /'    ','    ','rech','arge'/
c1------clear the rate accumulators.
      ratin=0.
      ratout=0.
c2------if cell-by-cell flow terms will be saved then clear the buffer.
      ibd=0
      if(icbcfl.eq.0 .or. irchcb.le.0) go to 5
      ibd=1
      do 2 il=1,nlay
      do 2 ir=1,nrow
      do 2 ic=1,ncol
      buff(ic,ir,il)=0.
    2 continue
c3------if nrchop=1 rech goes into layer 1. process each horizontal
c3------cell location.
    5 if(nrchop.ne.1) go to 15
c     ---recharge is applied to top layer
      do 10 ir=1,nrow
      do 10 ic=1,ncol
c3a-----if cell is external then do not do budget for it.
      if(ibound(ic,ir,1).le.0)go to 10
      q=rech(ic,ir)
c3b-----if cell-by-cell flow terms will be saved then add rech to buff
      if(ibd.eq.1) buff(ic,ir,1)=q
c3c-----if rech positive add it to ratin else add it to ratout.
      if(q) 8,10,7
    7 ratin=ratin+q
      go to 10
    8 ratout=ratout-q
   10 continue
      go to 100
c4------if nrchop=2 rech is in layer shown in indicator array(irch).
c4------process horizontal cell locations one at a time.
   15 if(nrchop.ne.2)go to 25
      do 20 ir=1,nrow
      do 20 ic=1,ncol
c4a-----get layer index from indicator array(irch).
      il=irch(ic,ir)
c4b-----if cell is external do not calculate budget for it.
      if(ibound(ic,ir,il).le.0)go to 20
      q=rech(ic,ir)
c4c-----if c-b-c flow terms will be saved then add recharge to buffer.
      if(ibd.eq.1) buff(ic,ir,il)=q
c4d-----if recharge is positive add to ratin else add it to ratout.
      if(q) 18,20,17
   17 ratin=ratin+q
      go to 20
   18 ratout=ratout-q
   20 continue
```

```
      go to 100
c5------if option=3 recharge is into highest internal cell. it will not
c5------pass through a constant head cell. process horizontal cell
c5------locations one at a time.
   25 if(nrchop.ne.3)go to 100
      do 30 ir=1,nrow
      do 30 ic=1,ncol
      do 28 il=1,nlay
c5a-----if cell is constant head move on to next horizontal location.
      if(ibound(ic,ir,il).lt.0) go to 30
c5b-----if cell is inactive move down to next cell.
      if (ibound(ic,ir,il).eq.0)go to 28
      q=rech(ic,ir)
c5c-----if c-b-c flow terms to be saved then add recharge to buffer.
      if(ibd.eq.1) buff(ic,ir,il)=q
c5d-----if rech is positive add it to ratin else add it to ratout.
      if(q) 27,30,26
   26 ratin=ratin+q
      go to 30
   27 ratout=ratout-q
      go to 30
   28 continue
   30 continue
  100 continue
c6------if c-b-c flow terms to be saved call ubudsv to write them.
      if(ibd.eq.1) call ubudsv(kstp,kper,text,irchcb,buff,ncol,nrow,
     .                         nlay,iout)
c7------move total recharge rate into vbvl for printing by baslot.
      vbvl(4,msum)=ratout
      vbvl(3,msum)=ratin
c8------add recharge for time step to recharge accumulator in vbvl.
      vbvl(2,msum)=vbvl(2,msum)+ratout*delt
      vbvl(1,msum)=vbvl(1,msum)+ratin*delt
c9------move budget term labels to vbnm for print by module bas_ot.
      vbnm(1,msum)=text(1)
      vbnm(2,msum)=text(2)
      vbnm(3,msum)=text(3)
      vbnm(4,msum)=text(4)
c10-----increment budget term counter.
      msum=msum+1
      return
      end
      subroutine ubudsv(kstp,kper,text,ibdchn,buff,ncol,nrow,nlay,iout)
c     30sep1991
c     record cell-by-cell flow terms for one component of flow.
c        specifications:
      character*4 text
      dimension text(4),buff(ncol,nrow,nlay)
c1------write an unformatted record containing identifying
c1------information.
      write(iout,1) text,ibdchn,kstp,kper
    1 format(1x,4a4,' budget values will be saved on unit',i3,
     1     ' at end of time step',i3,', stress period',i3)
      write(ibdchn) kstp,kper,text,ncol,nrow,nlay
```

```
c2------write an unformatted record containing values for
c2------each cell in the grid. the array is dimensioned
c2------(ncol,nrow,nlay)
      write(ibdchn) buff
      return
      end
      subroutine ucolno(n1b11,n1b12,nspace,ncp1,ndig,iout)
c-----version 1638 12may1987 ucolno
c     output column numbers above a matrix printout
c        n1b11 is the start column label (number)
c        n1b12 is the stop column label (number)
c        nspace is number of blank spaces to leave at start of line
c        ncp1 is number of column numbers per line
c        ndig is number of characters in each column field
c        iout is output channel
c        specifications:
      character*4 dot,space,dg,bf
      dimension bf(130),dg(10)
      data dg(1),dg(2),dg(3),dg(4),dg(5),dg(6),dg(7),dg(8),dg(9),dg(10)/
     1         '0   ','1   ','2   ','3   ','4   ','5   ','6   ',
     2         '7   ','8   ','9   '/
      data dot,space/'.   ','    '/
c1------calculate # of columns to be printed (n1b1), width
c1------of a line (ntot), number of lines (nwrap).
      write(iout,1)
    1 format(1x)
      n1b1=n1b12-n1b11+1
      n=n1b1
      if(n1b1.gt.ncp1) n=ncp1
      ntot=nspace+n*ndig
      if(ntot.gt.130) go to 50
      nwrap=(n1b1-1)/ncp1 + 1
      j1=n1b11-ncp1
      j2=n1b11-1
c2------build and print each line
      do 40 n=1,nwrap
c3------clear the buffer (bf).
      do 20 i=1,130
      bf(i)=space
   20 continue
      nbf=nspace
c4------determine first (j1) and last (j2) column # for this line.
      j1=j1+ncp1
      j2=j2+ncp1
      if(j2.gt.n1b12) j2=n1b12
c5------load the column #'s into the buffer.
      do 30 j=j1,j2
      nbf=nbf+ndig
      i2=j/10
      i1=j-i2*10+1
      bf(nbf)=dg(i1)
      if(i2.eq.0) go to 30
      i3=i2/10
      i2=i2-i3*10+1
```

```
      bf(nbf-1)=dg(i2)
      if(i3.eq.0) go to 30
      bf(nbf-2)=dg(i3+1)
   30 continue
c6------print the contents of the buffer (i.e. print the line).
      write(iout,31) (bf(i),i=1,nbf)
   31 format(1x,130a1)
   40 continue
c7------print a line of dots (for esthetic purposes only).
   50 ntot=ntot+5
      if(ntot.gt.130) ntot=130
      write(iout,51) (dot,i=1,ntot)
   51 format(1x,130a1)
      return
      end
      subroutine ulaprs(buf,text,kstp,kper,ncol,nrow,k,iprn,iout)
c-----version 1640 12may1987 ulaprs
c     print a 1 layer array in strips
c        specifications:
      character*4 text
      dimension buf(ncol,nrow),text(4)
c     be sure the format code (ip or iprn) is between 1 & 12
      ip=iprn
      if(ip.lt.1 .or. ip.gt.12) ip=12
c2------determine the number of values (ncap) printed on one line.
      if(ip.eq.1) ncap=11
      if(ip.eq.2) ncap=9
      if(ip.gt.2 .and. ip.lt.7) ncap=15
      if(ip.gt.6 .and. ip.lt.12) ncap=20
      if(ip.eq.12) ncap=10
c3------calculate the number of strips (nstrip).
      ncpf=129/ncap
      isp=0
      if(ncap.gt.12) isp=3
      nstrip=(ncol-1)/ncap + 1
      j1=1-ncap
      j2=0
c4------loop through the strips.
      do 2000 n=1,nstrip
c5------calculate the first(j1) & the last(j2) columns for this strip
      j1=j1+ncap
      j2=j2+ncap
      if(j2.gt.ncol) j2=ncol
c6-------print title on each strip
      write(iout,1) text,k,kstp,kper
    1 format(1x,4a4,' in layer',i3,' at end of time step',i3,
     1     ' in stress period',i3)
c7------print column numbers above the strip
      call ucolno(j1,j2,isp,ncap,ncpf,iout)
c8------loop through the rows printing cols j1 thru j2 with format ip
      do 1000 i=1,nrow
      go to(10,20,30,40,50,60,70,80,90,100,110,120), ip
c-----format 10g10.3
   10 write(iout,11) i,(buf(j,i),j=j1,j2)
```

```
   11 format(1h0,i3,2x,1pg10.3,10(1x,g10.3))
      go to 1000
c-----format 8g13.6
   20 write(iout,21) i,(buf(j,i),j=j1,j2)
   21 format(1h0,i3,2x,1pg13.6,8(1x,g13.6))
      go to 1000
c-----format 15f7.1
   30 write(iout,31) i,(buf(j,i),j=j1,j2)
   31 format(1h0,i3,1x,15(1x,f7.1))
      go to 1000
c-----format 15f7.2
   40 write(iout,41) i,(buf(j,i),j=j1,j2)
   41 format(1h0,i3,1x,15(1x,f7.2))
      go to 1000
c-----format 15f7.3
   50 write(iout,51) i,(buf(j,i),j=j1,j2)
   51 format(1h0,i3,1x,15(1x,f7.3))
      go to 1000
c-----format 15f7.4
   60 write(iout,61) i,(buf(j,i),j=j1,j2)
   61 format(1h0,i3,1x,15(1x,f7.4))
      go to 1000
c-----format 20f5.0
   70 write(iout,71) i,(buf(j,i),j=j1,j2)
   71 format(1h0,i3,1x,20(1x,f5.0))
      go to 1000
c-----format 20f5.1
   80 write(iout,81) i,(buf(j,i),j=j1,j2)
   81 format(1h0,i3,1x,20(1x,f5.1))
      go to 1000
c-----format 20f5.2
   90 write(iout,91) i,(buf(j,i),j=j1,j2)
   91 format(1h0,i3,1x,20(1x,f5.2))
      go to 1000
c-----format 20f5.3
  100 write(iout,101) i,(buf(j,i),j=j1,j2)
  101 format(1h0,i3,1x,20(1x,f5.3))
      go to 1000
c-----format 20f5.4
  110 write(iout,111) i,(buf(j,i),j=j1,j2)
  111 format(1h0,i3,1x,20(1x,f5.4))
      go to 1000
c-----format 9g11.4
  120 write(iout,121) i,(buf(j,i),j=j1,j2)
  121 format(1h0,i3,2x,1pg11.4,9(1x,g11.4))
 1000 continue
 2000 continue
      return
      end
      subroutine ul2prw(buf,text,kstp,kper,ncol,nrow,k,iprn,iout)
c     30sep1991 ul2prw
c     print 1 layer array
c        specifications:
      character*4 text
```

```
      dimension buf(ncol,nrow),text(4)
c1------print a header
      if(k.le.0)go to 5
      write(iout,1)text,k,kstp,kper
    1 format(1x,4a4,' in layer',i3,' at end of time step',i3,
     1     ' in stress period',i3)
c     be sure the format code (ip or iprn) is between 1 & 13
c     set default to 13
    5 ip=iprn
c     if(ip.lt.1 .or. ip.gt.12) ip=12
      if(ip.lt.1.or.ip.gt.13)ip=13
      if(ip.eq.13)then
c      write map scaled so largest value has four digits
       call uscmp(ncol,nrow,k,buf,text)
       return
      end if
c3------call the utility module ucolno to print column numbers.
      if(ip.eq.1) call ucolno(1,ncol,0,11,11,iout)
      if(ip.eq.2) call ucolno(1,ncol,0,9,14,iout)
      if(ip.gt.2 .and. ip.lt.7) call ucolno(1,ncol,3,15,8,iout)
      if(ip.gt.6 .and. ip.lt.12) call ucolno(1,ncol,3,20,6,iout)
      if(ip.eq.12) call ucolno(1,ncol,0,10,12,iout)
c4------loop through the rows printing each one in its entirety.
      do 1000 i=1,nrow
      go to(10,20,30,40,50,60,70,80,90,100,110,120), ip
c-----format 11g10.3
   10 write(iout,11) i,(buf(j,i),j=1,ncol)
   11 format(1h0,i3,2x,1pg10.3,10(1x,g10.3)/(5x,11(1x,g10.3)))
      go to 1000
c-----format 9g13.6
   20 write(iout,21) i,(buf(j,i),j=1,ncol)
   21 format(1h0,i3,2x,1pg13.6,8(1x,g13.6)/(5x,9(1x,g13.6)))
      go to 1000
c-----format 15f7.1
   30 write(iout,31) i,(buf(j,i),j=1,ncol)
   31 format(1h0,i3,1x,15(1x,f7.1)/(5x,15(1x,f7.1)))
      go to 1000
c-----format 15f7.2
   40 write(iout,41) i,(buf(j,i),j=1,ncol)
   41 format(1h0,i3,1x,15(1x,f7.2)/(5x,15(1x,f7.2)))
      go to 1000
c-----format 15f7.3
   50 write(iout,51) i,(buf(j,i),j=1,ncol)
   51 format(1h0,i3,1x,15(1x,f7.3)/(5x,15(1x,f7.3)))
      go to 1000
c-----format 15f7.4
   60 write(iout,61) i,(buf(j,i),j=1,ncol)
   61 format(1h0,i3,1x,15(1x,f7.4)/(5x,15(1x,f7.4)))
      go to 1000
c-----format 20f5.0
   70 write(iout,71) i,(buf(j,i),j=1,ncol)
   71 format(1h0,i3,1x,20(1x,f5.0)/(5x,20(1x,f5.0)))
      go to 1000
c-----format 20f5.1
```

```
   80 write(iout,81) i,(buf(j,i),j=1,ncol)
   81 format(1h0,i3,1x,20(1x,f5.1)/(5x,20(1x,f5.1)))
      go to 1000
c-----format 20f5.2
   90 write(iout,91) i,(buf(j,i),j=1,ncol)
   91 format(1h0,i3,1x,20(1x,f5.2)/(5x,20(1x,f5.2)))
      go to 1000
c-----format 20f5.3
  100 write(iout,101) i,(buf(j,i),j=1,ncol)
  101 format(1h0,i3,1x,20(1x,f5.3)/(5x,20(1x,f5.3)))
      go to 1000
c-----format 20f5.4
  110 write(iout,111) i,(buf(j,i),j=1,ncol)
  111 format(1h0,i3,1x,20(1x,f5.4)/(5x,20(1x,f5.4)))
      go to 1000
c-----format 10g11.4
  120 write(iout,121) i,(buf(j,i),j=1,ncol)
  121 format(1h0,i3,2x,1pg11.4,9(1x,g11.4)/(5x,10(1x,g11.4)))
 1000 continue
      return
      end
      subroutine ulasav(buf,text,kstp,kper,pertim,totim,ncol,nrow,
     . nlay,ichn)
c     30sep1991 ulasav
c     save 1 layer array on disk
c     specifications:
      character*4 text
      dimension buf(ncol,nrow),text(4)
c1    write an unformatted record containing identifying information
      write(ichn)kstp,kper,pertim,totim,text,ncol,nrow,nlay
c2    write an unformatted record containing array values
c2    the array is dimensioned (ncol,nrow)
c     write(ichn)((buf(ic,ir),ic=1,ncol),ir=1,nrow)
      do 20 i=1,nrow
   20 write(ichn)(buf(j,i),j=1,ncol)
      return
      end
      subroutine uldrel(a,aname,jj,in,iout)
c     30sep1991
c     routine to input 1-d real data matrices
c       a is array to input
c       aname is 24 character description of a
c       jj is no. of elements
c       in is input unit
c       iout is output unit
c        specifications:
      character*4 aname
      character*20 fmtin
      dimension a(jj),aname(6)
c1------read array control record.
      read (in,1) locat,cnstnt,fmtin,iprn
    1 format(i10,f10.0,a20,i10)
c2------use locat to see where array values come from.
      if(locat.gt.0) go to 90
```

```
c3------if locat=0 then set all array values equal to cnstnt. return
      do 80 j=1,jj
   80 a(j)=cnstnt
      write(iout,3) aname,cnstnt
    3 format(1x,6a4,' =',g15.7)
      return
c4------if locat>0 then read formatted records using format fmtin.
   90 write(iout,5) aname,locat,fmtin
    5 format(1x,6a4,' will be read on unit',i3,
     1         ' using format: ',a20)
      read (locat,fmtin) (a(j),j=1,jj)
c5------if cnstnt not zero then multiply array values by cnstnt.
      if(cnstnt.eq.0.) go to 120
      do 100 j=1,jj
  100 a(j)=a(j)*cnstnt
c6------if print code (iprn) =>0 then print array values.
  120 if(iprn.lt.0) return
      write(iout,1001) (a(j),j=1,jj)
 1001 format((1x,1pg12.5,9(1x,g12.5)))
      return
      end
      subroutine u2dint(ia,aname,ii,jj,k,in,iout)
c     30sep1991
c     routine to input 2-d integer data matrices
c       ia is array to input
c       aname is 24 character description of ia
c       ii is no. of rows
c       jj is no. of cols
c       k is layer no. (used with name to title printout unless k is 0)
c       in is input unit
c       iout is output unit
c        specifications:
      character*4 aname
      character*20 fmtin
      dimension ia(jj,ii),aname(6)
c1------read array control record.
      read (in,1) locat,iconst,fmtin,iprn
    1 format(i10,i10,a20,i10)
c2------use locat to see where array values come from.
      if(locat) 200,50,90
c3------if locat=0 then set all array values equal to iconst. return
   50 do 80 i=1,ii
      do 80 j=1,jj
   80 ia(j,i)=iconst
      if(k.gt.0) write(iout,2) aname,iconst,k
    2 format(1x,6a4,' =',i15,' for layer',i3)
      if(k.le.0) write(iout,3) aname,iconst
    3 format(1x,6a4,' =',i15)
      return
c4------if locat>0 then read formatted records using format fmtin.
   90 if(k.gt.0) write(iout,4) aname,k,locat,fmtin
    4 format(1x,6a4,' for layer',i3,' will be read on unit',
     1       i3,' using format: ',a20)
      if(k.le.0) write(iout,5) aname,locat,fmtin
```

```
    5 format(1x,6a4,' will be read on unit',
     1        i3,' using format: ',a20)
      do 100 i=1,ii
      read (locat,fmtin) (ia(j,i),j=1,jj)
  100 continue
      go to 300
c5------locat<0 then read unformatted record containing array values
  200 locat=-locat
      if(k.gt.0) write(iout,201) aname,k,locat
  201 format(1x,6a4,', layer',i3,
     1    ' will be read unformatted on unit',i3)
      if(k.le.0) write(iout,202) aname,locat
  202 format(1x,6a4,
     1    ' will be read unformatted on unit',i3)
c5a   do not read an unformatted dummy record first.
c     read(locat)
      read(locat)ia
c6------if iconst not zero then multiply array values by iconst.
  300 if(iconst.eq.0) go to 320
      do 310 i=1,ii
      do 310 j=1,jj
      ia(j,i)=ia(j,i)*iconst
  310 continue
c7------if print code (iprn) =>0 then print array values.
  320 if(iprn.lt.0) return
      if(iprn.gt.5) iprn=0
      iprn=iprn+1
c8------print column numbers at top of page.
      if(iprn.eq.1) call ucolno(1,jj,0,10,12,iout)
      n1=125/iprn/5*5
      if(iprn.gt.1) call ucolno(1,jj,4,n1,iprn,iout)
c9------print each row in the array.
      do 110 i=1,ii
c10-----select the format
      go to(101,102,103,104,105,106), iprn
c-----format 10i11
  101 write(iout,1001) i,(ia(j,i),j=1,jj)
 1001 format(1h0,i3,2x,i11,9(1x,i11)/(5x,10(1x,i11)))
      go to 110
c-----format 60i1
  102 write(iout,1002) i,(ia(j,i),j=1,jj)
 1002 format(1h0,i3,1x,60(1x,i1)/(5x,60(1x,i1)))
      go to 110
c-----format 40i2
  103 write(iout,1003) i,(ia(j,i),j=1,jj)
 1003 format(1h0,i3,1x,40(1x,i2)/(5x,40(1x,i2)))
      go to 110
c-----format 30i3
  104 write(iout,1004) i,(ia(j,i),j=1,jj)
 1004 format(1h0,i3,1x,30(1x,i3)/(5x,30(1x,i3)))
      go to 110
c-----format 25i4
  105 write(iout,1005) i,(ia(j,i),j=1,jj)
 1005 format(1h0,i3,1x,25(1x,i4)/(5x,25(1x,i4)))
```

```
      go to 110
c-----format 20i5
  106 write(iout,1006) i,(ia(j,i),j=1,jj)
 1006 format(1h0,i3,1x,20(1x,i5)/(5x,20(1x,i5)))
  110 continue
      return
      end
      subroutine u2dre1(a,aname,ii,jj,k,in,iout)
c     30sep1991
c     routine to input 2-d real data matrices
c       a is array to input
c       aname is 24 character description of a
c       ii is no. of rows
c       jj is no. of cols
c       k is layer no. (used with name to title printout unless k is 0)
c       in is input unit
c       iout is output unit
c        specifications:
      character*4 aname
      character*20 fmtin
      dimension a(jj,ii),aname(6)
c1------read array control record.
      read (in,1) locat,cnstnt,fmtin,iprn
    1 format(i10,f10.0,a20,i10)
c2------use locat to see where array values come from.
      if(locat) 200,50,90
c3------if locat=0 then set all array values equal to cnstnt. return
   50 do 80 i=1,ii
      do 80 j=1,jj
   80 a(j,i)=cnstnt
      if(k.gt.0) write(iout,2) aname,cnstnt,k
    2 format(1x,6a4,' =',g15.7,' for layer',i3)
      if(k.le.0) write(iout,3) aname,cnstnt
    3 format(1x,6a4,' =',g15.7)
      return
c4------if locat>0 then read formatted records using format fmtin.
   90 if(k.gt.0) write(iout,4) aname,k,locat,fmtin
    4 format(1x,6a4,' for layer',i3,' will be read on unit',
     1       i3,' using format: ',a20)
      if(k.le.0) write(iout,5) aname,locat,fmtin
    5 format(1x,6a4,' will be read on unit',
     1       i3,' using format: ',a20)
      do 100 i=1,ii
      read (locat,fmtin) (a(j,i),j=1,jj)
  100 continue
      go to 300
c5------locat<0 then read unformatted record containing array values
  200 locat=-locat
      if(k.gt.0) write(iout,201) aname,k,locat
  201 format(1x,6a4,', layer',i3,
     1    ' will be read unformatted on unit',i3)
      if(k.le.0) write(iout,202) aname,locat
  202 format(1x,6a4,
     1    ' will be read unformatted on unit',i3)
```

```
c5a   do not read an unformatted dummy record first.
c     read(locat)
      read(locat)a
c6------if cnstnt not zero then multiply array values by cnstnt.
  300 if(cnstnt.eq.0.) go to 320
      do 310 i=1,ii
      do 310 j=1,jj
      a(j,i)=a(j,i)*cnstnt
  310 continue
c7------if print code (iprn) =>0 then print array values.
  320 if(iprn.lt.0) return
      call ul2prw(a,aname,0,0,jj,ii,0,iprn,iout)
      return
      end
      subroutine uscmp(ncol,nrow,k,arr,tit)
c     30sep1991
c     writes map
      dimension arr(ncol,nrow),mar(25),tit(4)
      frbg=1.e4
      frlt=1.e3
c     big=largest absolute value in array
      big=0.
      do 10 i=1,nrow
      do 10 j=1,ncol
   10 big=amax1(big,abs(arr(j,i)))
c     scale array so big has four digits
      scl=1.
      if(big.eq.0.)then
       write(6,2)tit
       return
      end if
   20 if(big-frlt)30,50,40
   30 scl=scl*10.
      big=big*10.
      go to 20
   40 if(big.ge.frbg)then
       scl=scl*.1
       big=big*.1
       go to 40
      end if
   50 continue
c     big has four digits
      rec=1./scl
      left=1
      ight=25
   60 continue
c     write map<26 columns wide
      write(6,4)tit,k,rec
      ight=min0(ncol,ight)
      write(6,6)(m,m=left,ight)
      do 80 i=1,nrow
      mr=0
      do 70 j=left,ight
      mr=mr+1
```

```
   70 mar(mr)=arr(j,i)*scl+sign(.5,arr(j,i))
   80 write(6,8)i,(mar(m),m=1,mr),i
      write(6,12)
      write(6,6)(m,m=left,ight)
      left=left+25
      if(left.le.ncol)then
c      write map of next set of columns
       ight=ight+25
       go to 60
      end if
    2 format(' every value in ',4a4,' is zero')
    4 format(1x,4a4,i5,' multiply by',1p5e12.1)
    6 format(3x,25i5)
    8 format(26(4x,1h.),//,i3,25i5,i3)
   12 format(26(4x,1h.))
      return
      end
      subroutine uwrib(ncol,nrow,nlay,ibound)
c     30sep1991
      dimension ibound(ncol,nrow,nlay)
c     write formatted ibound array, where dry cells set to zero
      do 10 k=1,nlay
      do 10 i=1,nrow
   10 write(61,2)(ibound(j,i,k),j=1,ncol)
    2 format(40i3)
      return
      end
```


*U.S. GOVERNMENT PRINTING OFFICE 1993-0-794-506/10007