Techniques of Water-Resources Investigations
of the United States Geological Survey

Chapter A1

A MODULAR THREE-DIMENSIONAL FINITE-DIFFERENCE GROUND-WATER FLOW MODEL

By Michael G. McDonald and
Arlen W. Harbaugh

This chapter supersedes U.S. Geological
Survey Open-File Report 83-875

Book 6

MODELING TECHNIQUES

CHAPTER 3

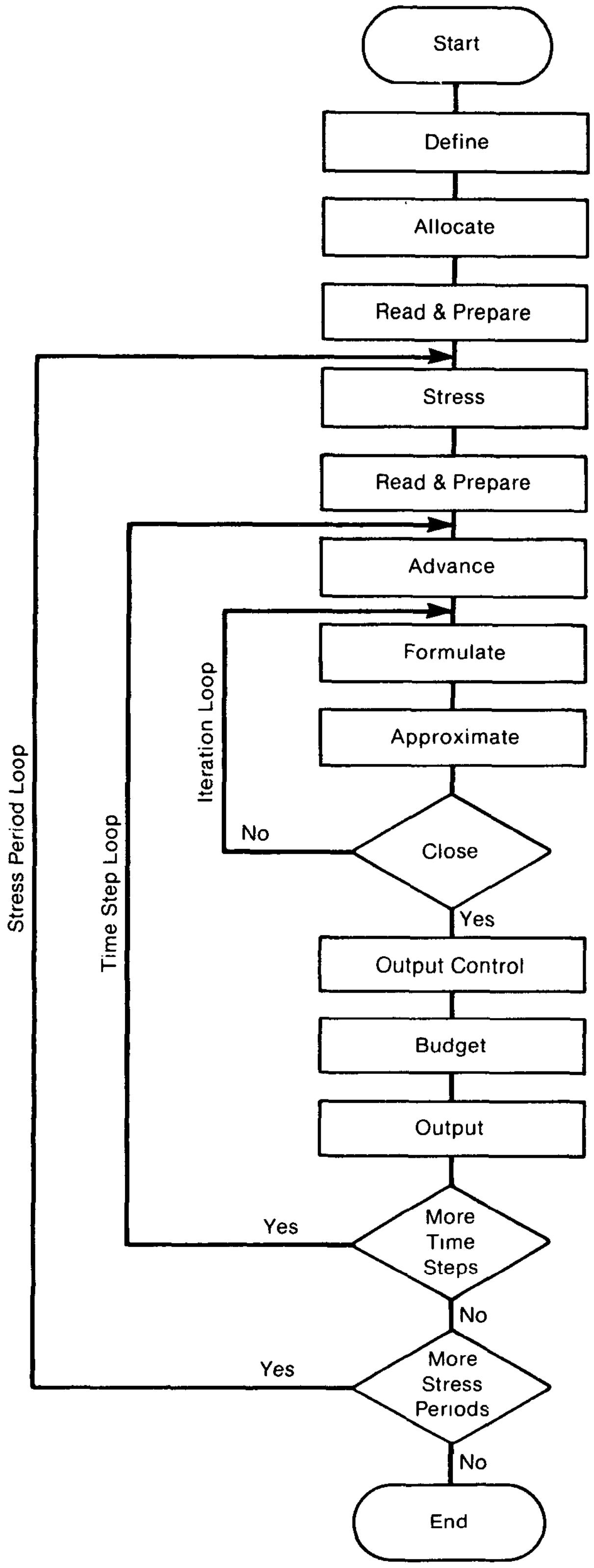
PROGRAM DESIGN

Overall Structure

This chapter describes the overall design of the model program. The program consists of a main program (MAIN) and a large number of highly independent subroutines called modules. This chapter will explain the functions of MAIN and explain how the modules can be grouped into "packages" and "procedures".

The functions which must be performed for a typical simulation are shown in figure 13. The period of simulation is divided into a series of "stress periods" within which specified stress parameters are constant. Each stress period, in turn, is divided into a series of time steps. The system of finite-difference equations of the form of equation (27) is formulated and solved to yield the head at each node at the end of each time step. Iterative solution methods are used to solve for the heads for each time step. Thus within a simulation, there are three nested loops: a stress-period loop, within which there is a time-step loop, which in turn contains an iteration loop.

Each rectangle in figure 13 is termed a "procedure". For example, prior to entering the stress loop, the program executes three procedures which pertain to the simulation as a whole. In the Define Procedure, the problem to be simulated is defined: the size of the model, the type of simulation (transient or steady-state), the number of stress periods, the hydrologic options, and the solution scheme to be used are specified. In the Allocate Procedure, memory space required by the program is allocated. In the Read and Prepare Procedure, all data that are not functions of time



DEFINE — Read data specifying number of rows, columns, layers, stress periods, and major program options.

ALLOCATE — Allocate space in the computer to store data.

READ AND PREPARE — Read data which is constant throughout the simulation. Prepare the data by performing whatever calculations can be made at this stage.

STRESS — Determine the length of a stress period and calculate terms to divide stress periods into time steps.

READ AND PREPARE — Read data which changes from one stress period to the next. Prepare the data by performing whatever calculations can be made at this stage

ADVANCE — Calculate length of time step and set heads at beginning of a new time step equal to heads calculated for the end of the previous time step.

FORMULATE — Calculate the coefficients of the finite difference equations for each cell.

APROXIMATE — Make one cut at approximating a solution to the system of finite difference equations.

OUTPUT CONTROL — Determine whether results should be written or saved on disk for this time step. Send signals to the BUDGET and OUTPUT procedures to indicate exactly what information should be put out.

BUDGET — Calculate terms for the overall volumetric budget and calculate and save cell-by-cell flow terms for each component of flow.

OUTPUT — Print and save heads, drawdown and overall volumetric budgets in accordance with signals from OUTPUT CONTROL procedure.

Figure 13.—Overall program structure.

are read. These data may include all or some of the following: boundary conditions, initial heads (starting heads), transmissivity, hydraulic conductivity, specific yield, storage coefficients, elevations of layer tops and bottoms, and parameters required by the specified solution scheme. Certain preliminary calculations are also made in this procedure to prepare data for further processing.

Within the stress period loop the first procedure is termed the Stress Procedure. In this procedure the number of time steps (NSTP) in the stress period and certain information to calculate the length of each time step are read. In a second Read and Prepare Procedure, all data that pertain to a stress period, such as pumping rates and areal recharge, are read and processed. The time-step loop is then entered (figure 13); in the Advance Procedure, the length of the time step is calculated and the heads for the start of the time step are initialized. The iteration loop contains the Formulate Procedure which determines the conductances and coefficients for each node as required by equation (27), and the Approximate Procedure which approximates a solution to the system of linear equations for head. Iteration proceeds until closure is achieved or until a specified maximum number of allowable iterations is reached. At the end of the iteration loop, the Output Control Procedure determines the disposition of the computed heads, budget terms, and cell-by-cell flow terms. In the Budget Procedure, budget entries are calculated and cell-by-cell flow terms are printed or recorded, as explained in a subsequent section. In the Output Procedure, heads, drawdown, and the volumetric budget are printed or recorded.

As shown in the preceding discussion, figure 13 provides a flow chart for the overall program structure, a list of the various procedures, and an indication of the sequence in which those procedures are implemented; it also provides a flow chart for the main program of the model. The work within the procedures--i.e., within the rectangles of figure 13--is performed by individual subroutines, or modules, called by the main program. The main program itself is simply an organized sequence of call statements, most of which are coupled to "IF" tests which determine whether a module is required. Accordingly, the main program does not itself do the work of simulation; it merely calls the various modules in the proper sequence to do that work. Modules which are called directly by the main program are termed "primary" modules; those that are called by other modules are termed "secondary" modules.

Thus the various procedures indicated in figure 13 are implemented through individual modules; and the modules can accordingly be grouped according to the procedure which they help to perform. As noted in Chapter 1, modules can also be grouped by "packages", where a package (for example, the River Package, the Well Package, or the SIP Package) includes those modules required to incorporate a particular hydrologic process or solution algorithm into the simulation. In terms of understanding the operation of the model, these two methods of grouping modules are both useful. The package classification, for example, indicates which modules will be active in a given simulation. (Modules are called by the main program only if they are part of a package which is required in the simulation; and while some packages are required in all simulations, most are needed only when the hydrologic process or solution method embodied in the package is specified by the user.) The procedure classification, on the other hand, defines the specific function of the module in relation to the functions of other modules of the package. For example, several modules whose function is to allocate space are grouped under the Allocate Procedure; each of these modules allocates the space required for the arrays used in a single package. If few options or features are specified, relatively few packages are involved in the simulation, and the Allocate Procedure is handled by a relatively small number of modules. As the options specified by the user increase, more packages enter the simulation, and more modules are called to complete the space allocation task.

Figure 14 illustrates the classification of modules by procedure and by package in terms of a matrix of primary modules (i.e., modules called by the main program). The horizontal rows in figure 14 correspond to procedures, while the vertical columns correspond to packages. An "X" is entered in each block of the matrix for which a module exists; absence of an "X" indicates that the procedure in question is not required in the indicated package. Entries marked with a subscript "S" indicate primary modules which utilize submodules in accomplishing their function; submodules are secondary modules which are utilized only in a single package. Entries marked with the subscript "U" indicate primary modules which utilize utility modules to accomplish their tasks; utility modules are secondary modules which are available to many packages.

		Flow Component Packages							Solver Packages	
			Stress Packages							
Procedures	BAS	BCF	WEL	RCH	RIV	DRN	EVT	GHB	SIP	SOR
Define (DF)	X									
Allocate (AL)	X	X	X	X	X	X	X	X	X	X
Read & Prepare (RP)	X_U	X_{US}							X	X
Stress (ST)	X									
Read & Prepare (RP)			X	X_U	X	X	X_U	X		
Advance (AD)	X									
Formulate (FM)	X	X_S	X	X	X	X	X	X		
Approximate (AP)									X_S	X_S
Output Control (OC)	X									
Budget (BD)		X_{US}	X_U	X_U	X_U	X_U	X_U	X_U		
Output (OT)	X_U									

Figure 14.—Organization of modules by procedures and packages.

The primary modules are named according to a convention which indicates both the package and the procedure to which they belong. The first three characters designate the package, the fourth is a package version number, and the last two indicate the procedure. For example, in figure 14, a module is indicated for the Well Package and Allocate Procedure. This module is designated as WEL1AL; the first three letters, WEL, indicate that the module is part of the Well Package; the last two letters, AL, indicate that it performs the Allocate Procedure in that package. Thus this module is one of those that deals with the simulation of specified withdrawal or input, as through wells, and its particular function is to allocate the space in computer memory used to store well data. The number one appearing in the fourth place of the six-character module designation is a package version number. If the package is modified to effect improvements, a different integer would be used in this place to distinguish the modified package from the original or from other modified versions.

Figure 15 shows the names of the primary modules arranged in the same matrix format that was used in figure 14. As in figure 14, a subscript "S" indicates that submodules are utilized and "U" indicates that utility modules are utilized.

Submodules are designated by a six-character name in which the first character is always the letter "S". This is followed by three characters designating the package name, a numeral indicating the package version number, and a one-character mnemonic to distinguish the module from other submodules of the same package; for example, the secondary module "SBCF1C" is a submodule in version one of the Block-Centered Flow Package. Utility

PROCEDURES	Packages									
	BAS	BCF	WEL	RCH	RIV	DRN	EVT	GHB	SIP	SOR
Define (DF)	BAS1DF									
Allocate (AL)	BAS1AL	BCF1AL	WEL1AL	RCH1AL	RIV1AL	DRN1AL	EVT1AL	GHB1AL	SIP1AL	SOR1AL
Read & Prepare (RP)	BAS1RP$_U$	BCF1RP$_{US}$							SIP1RP	SOR1RP
Stress (ST)	BAS1ST									
Read & Prepare (RP)			WEL1RP	RCH1RP$_U$	RIV1RP	DRN1RP	EVT1RP$_U$	GHB1RP		
Advance (AD)	BAS1AD									
Formulate (FM)	BAS1FM	BCF1FM$_S$	WEL1FM	RCH1FM	RIV1FM	DRN1FM	EVT1FM	GHB1FM		
Approximate (AP)									SIP1AP$_S$	SOR1AP$_S$
Output Control (OC)	BAS1OC									
Budget (BD)		BCF1BD$_{US}$	WEL1BD$_U$	RCH1BD$_U$	RIV1BD$_U$	DRN1BD$_U$	EVT1BD$_U$	GHB1BD$_U$		
Output (OT)	BAS1OT$_U$									

Figure 15.—Primary modules organized by procedure and package.

modules are designated by the letter "U" followed by a five-character mnemonic. For example, the secondary module "U2DREL" is a utility module which reads two-dimensional real arrays.

Table 1 lists the various packages documented in this publication, gives the three-character abbreviation used in the module designation scheme, and provides a brief description of the package operation. Two major categories of package may be recognized--the flow component packages and the solver packages; within the category of flow component packages, a stress package subcategory may be recognized. The flow component packages are those which calculate the coefficients of the finite-difference equation for each cell. This category includes the Block-Centered Flow Package, which formulates the internal flow terms (describing flow between cells and flow to or from storage); and the subcategory of stress packages. Each of the stress packages formulates the coefficients describing a particular external or boundary flow; for example, the River Package calculates the coefficients describing flow between a cell and a surface stream. The solver packages are those which implement algorithms for solution of the systems of finite-difference equations. This documentation describes two packages in this category, one incorporating the Strongly Implicit Procedure of solution, and the other utilizing Slice-Successive Overrelaxation. The only package which does not fit into any of these categories is the Basic Package, which addresses a variety of tasks in support of the entire simulation.

The Block-Centered Flow Package is the only option described in this documentation for the formulation of internal flow terms in the equations. However, alternative packages, for example, utilizing a point centered approach, could certainly be developed and used in place of the Block-Centered Flow Package.

Table 1.--List of packages.

Package Name	Abbreviation	Package Description		
Basic	BAS	Handles those tasks that are part of the model as a whole. Among those tasks are specification of boundaries, determination of time-step length, establishment of initial conditions, and printing of results.		
Block-Centered Flow	BCF	Calculates terms of finite-difference equations which represent flow within porous medium; specifically, flow from cell to cell and flow into storage.		
Well	WEL	Adds terms representing flow to wells to the finite-difference equations.		
Recharge	RCH	Adds terms representing areally distributed recharge to the finite-difference equations.		
River	RIV	Adds terms representing flow to rivers to the finite-difference equations.		Flow Component Packages
Drain	DRN	Adds terms representing flow to drains to the finite-difference equations.	Stress Packages	
Evapotranspiration	EVT	Adds terms representing ET to the finite-difference equations.		
General-Head Boundaries	GHB	Adds terms representing general-head boundaries to the finite-difference equations.		
Strongly Implicit Procedure	SIP	Iteratively solves the system of finite-difference equations using the Strongly Implicit Procedure.		Solver Packages
Slice-Successive Overrelaxation	SOR	Iteratively solves the system of finite-difference equations using Slice-Successive Overrelaxation.		

Similarly, additional solver packages, incorporating different solution algorithms, could be added, as could additional stress packages. Every simulation must include the Basic Package, the Block-Centered Flow Package (or a suitable replacement) and a solver package. Beyond this, the packages to be included in a simulation are at the option of the user, and will depend on the hydrologic processes influencing the problem. The individual modules in the program have been designed in such a way that the packages are totally independent; with the exception of the three required packages noted above, addition or removal of an individual package has no effect on other packages. If an entirely new package is desired, modules can be developed for each of the procedures involved (and the main program modified to call those modules in proper sequence) without affecting other packages of the program.

Figure 16 shows a detailed flow chart of the main program, indicating all of the primary modules together with the tests which determine whether or not each module is to be called. Figure 16 may be studied in conjunction with figures 13 and 15, and table 1, for an appreciation of the overall structure and operation of the model.

The overall design of the model is such that the conductance terms for cell-to-cell flow (CC, CR, and CV of equation (26)) are formulated at the beginning of the simulation, and are reformulated if necessary at each iteration during solution. Reformulation takes place only in unconfined situations, where the conductances depend upon saturated thickness, which may change at each iteration. At present, formulation of conductances is done only by the Block-Centered Flow Package, although again, a replacement package could readily be developed. The lateral conductance terms (CC and CR

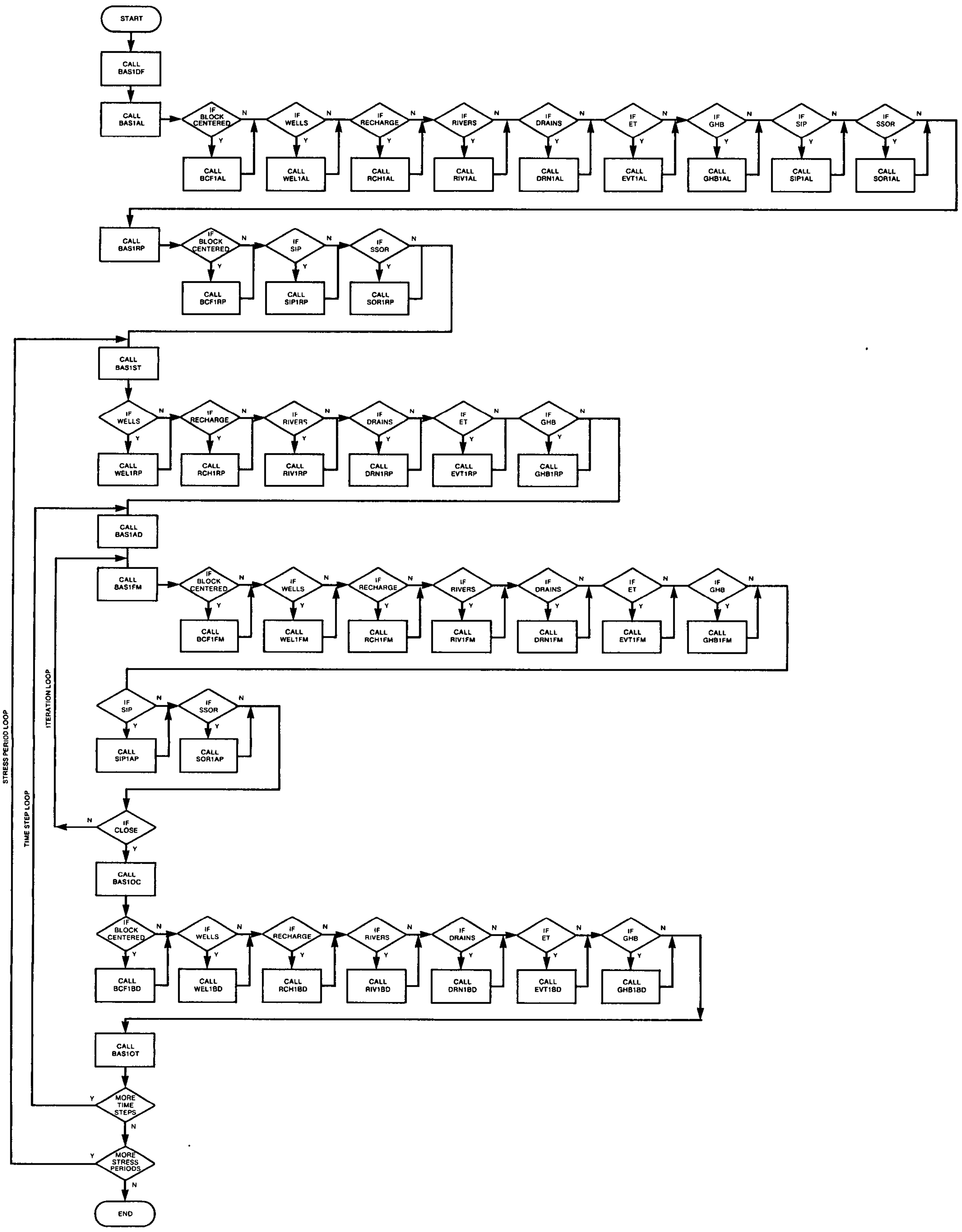


Figure 16.—Overall program structure showing all primary modules.

in equation (26)) are computed as harmonic means for the intervals between nodes, using parameters specified by the user for individual cells; the vertical conductance (CV in equation (26)) is calculated using information which is specified directly for the vertical interval between nodes. The various conductance terms are stored in arrays which are ultimately passed to the solver package, where the matrix equations (equation (27)) are solved.

The coefficient $HCOF_{i,j,k}$ and the term $RHS_{i,j,k}$ of equation (26) are formulated anew at each iteration, for all active nodes in the mesh. This formulation is done progressively, as each package calculates and adds terms for the particular process associated with that package. At the beginning of each iteration, the values of $HCOF_{i,j,k}$ and $RHS_{i,j,k}$ are set to zero throughout the mesh. The Block-Centered Flow Package then adds the term $-SS_{i,j,k}\Delta r_j \Delta c_i \Delta v_k/(t_m - t_{m-1})$ to $HCOF_{i,j,k}$ at each node, and adds the term

$$\frac{-SS_{i,j,k}\Delta r_j \Delta c_i \Delta v_k}{t_m - t_{m-1}} \; h^{m-1}{}_{i,j,k}$$

to $RHS_{i,j,k}$ at each node. For cells that are affected by flow from a stream, given by an expression of the form $P_{si,j,k}$ $(h_s - h_{i,j,k})$, where h_s is the (constant) stream head, the River package adds the term $-P_{si,j,k}$ to $HCOF_{i,j,k}$, and adds the constant term $-P_{si,j,k}$ h_s to $RHS_{i,j,k}$. This process continues until each package specified by the user has added its contribution to HCOF and RHS at each indicated node of the mesh. The HCOF and RHS arrays are then transferred to the solver package, together with the three conductance arrays (CC,CR and CV), an array containing heads at the beginning of the time step, and the IBOUND array, which identifies constand head, no flow and active nodes. The solver package sums the six conductance terms and the value of HCOF at each node to create a single coefficient of $h_{i,j,k}$

(corresponding to the term in brackets in equation (26)), and carries out one iteration of the solution procedure. The various arrays used in the solution procedure are actually stored as segments of a single one-dimensional array, the "X" array.

As noted in Chapter 1, Chapters 4 through 13 of this document discuss the program in terms of individual packages. Each of these chapters contains a detailed description of a particular package, including a listing and discussion of each module included in the package. The remainder of this chapter describes the way boundaries, water budget calculations, space allocation and input-output are handled in the model, and provides a brief description and listing of the main program.

Array Boundaries and Aquifer Boundaries

As noted in Chapter 2, the model may be visualized in terms of a three-dimensional assemblage of cells, each cell associated with a node of the model array. The size of the model array is specified by the user in terms of the number of rows (NROW), number of columns (NCOL) and number of layers (NLAY); these terms define a three-dimensional array of cells in the form of a rectangular box. In formulating the finite-difference equations, cell-to-cell conductance terms are omitted for the exterior of cells on the outer surface of this rectangular array. Thus considering flow along a row, a cell-to-cell conductance term is developed for the interval between column 1 and column 2, but not for the interval to the opposite side of column 1; similarly, a conductance term is developed for the interval between column (NCOL-1) and column (NCOL), but not for the interval beyond column (NCOL). Similar conventions are established in the other two directions, so that in effect the array is bounded externally by planes across which no cell-to-cell flow occurs. If these boundaries of the model array, which are actually embedded in the program, coincide with impermeable boundaries in the aquifer, they can be relied upon to simulate the no-flow condition along those aquifer boundaries without further intervention by the user. In general, however, the aquifer boundaries will be irregular in form, or will not be of a simple impermeable character. In these cases, the aquifer boundary must be simulated by specifying certain cells within the array as no-flow or constant-head, by using external stress terms, or by using a combination of no-flow cells and external stress terms. This was discussed in Chapter 2, and is further discussed below. It should also be noted that while no cell-to-cell conductance terms are formulated for the interval above the uppermost layer of the model array, flow into this layer from above is frequently represented in the model through external stress terms--for example, terms representing evapotranspiration or stream seepage.

A finite-difference equation of the form of (26) is formulated for each variable-head cell in the mesh. For constant-head cells, no equation is formulated; however, the equation for each variable-head cell adjacent to a constant-head cell contains a term describing flow to and from the constant-head cell. For inactive no-flow cells, no equation is formulated, and no term appears in the equation of any adjacent cell for flow to or from the inactive cell; thus no flow is simulated across the interval between an inactive cell and any adjacent cell.

As pointed out above, the model array as initially generated always has the form of a rectangular box. Where the limits of an aquifer do not coincide with this rectangular shape, inactive cells may be used to delete portions of the array which fall outside the aquifer boundaries; this was discussed through an example in Chapter 2. As noted in the same example, constant-head cells may be used to represent such features as surface water bodies of constant level which are in full contact with the aquifer. Boundaries which are characterized by a constant rate of flow into or out of the aquifer may be simulated using a no-flow boundary in conjunction with the Well Package, by assigning appropriate withdrawal or recharge rates to nodes just inside the boundary. Boundaries characterized by inflow which varies in proportion to head can be simulated using the General Head Boundary Package or the River Package, where these again are applied to nodes just interior to a no-flow boundary. Use of the River Package would involve specifying artificial streambed conductance and stream-head values at each cell along the boundary, where these values are deliberately chosen in such a way as to duplicate the required head-flow relationships.

Constant-head cells, inactive cells and variable-head cells are distinguished from one another in the model through the IBOUND array, which contains one element for each cell in the mesh. The entry in the IBOUND array for a given cell indicates the type of cell according to the following convention:

```
IBOUND (I,J,K) < 0 . . . .Cell I,J,K is constant head
IBOUND (I,J,K) = 0 . . . .Cell I,J,K is inactive
IBOUND (I,J,K) > 0 . . . .Cell I,J,K is variable head
```

The IBOUND codes are initially specified by the user. If necessary, the codes are adjusted so that they are consistent with other data specified by the user and with intermediate results. For example, cells which are specified as active but are given transmissivity and vertical-leakance values equal to zero are changed to inactive cells by the program.

Volumetric Budget

A summary of all inflows and outflows to a region is generally called a water budget. In this report, the water budget is termed a volumetric budget because it deals with volumes of water and volumetric flow rates; thus strictly speaking it is not a mass balance, although this term has been used in reference to volumetric budgets in other model reports. The model program calculates a water budget for the overall model as a check on the acceptability of the solution, and in order to provide summarized information on the flow system.

Numerical solution techniques for simultaneous equations do not always result in a correct answer; in particular, iterative solvers may stop iterating before a sufficiently close approximation to the solution is attained. A water budget provides an indication of the overall acceptability of the solution. The system of equations solved by the model actually consists of a flow continuity statement for each model cell. Continuity should also exist for the total flows into and out of the model--that is, the difference between total inflow and total outflow should equal the total change in storage. In the model program, the water budget is calculated independently of the equation solution process, and in this sense may provide independent evidence of a valid solution.

Each flow component package calculates its own contribution to the budget. The total budget as printed in the output does not include internal flows between model cells--only flows into or out of the model as a whole. For example, flow to or from rivers, flow to or from constant head cells, and flow to wells are all included in the overall budget terms. Flow into and out of storage is also considered part of the overall budget inasmuch as accumulation in storage effectively removes water from the flow system, and storage release effectively adds water to the flow--even though neither process, in itself, involves the transfer of water into or out of the ground water regime.

For every time step, the budget module of each flow component package calculates the rate of flow into and out of the system due to the process simulated by the package. The inflows and outflows for each component of flow are stored separately in the VBVL array. Most packages deal with only one such component of flow, but the Block-Centered Flow Package deals with two--flow to constant head cells and flow to storage. In addition to flow, the volumes of water entering and leaving the model during the time step are calculated as the product of flow rate and time step length. Cumulative volumes, from the beginning of the simulation, are then calculated and stored in array VBVL.

Module SBAS1V in the BAS Package uses the inflows, outflows and cumulative volumes in the VBVL array to print the budget at the times requested by the model user. When a budget is printed, the flow rates for the last time step and cumulative volumes from the beginning of simulation are printed for each component of flow. Inflows are printed separately from ouflows; following the convention indicated above, water entering storage is treated as an outflow while water released form storage is treated as an inflow. In addition, total inflow and total outflow are printed, as well as the difference between total inflow and outflow. The difference is then printed as a percent error, calculated using the formula:

$$D = \frac{100(IN-OUT)}{(IN+OUT)/2}$$

where IN is the total inflow to the system, OUT is the total outflow and D is the percent error term. If the model equations are correctly solved, the percent error should be small. In general, flow rates may be taken as an indication of solution validity for the time step to which they apply, while cumulative volumes are an indication of validity for the entire simulation up to the time of the printout. The budget is printed at the end of each stress period whether requested or not.

There are situations in which it is useful to calculate flow terms for various subregions of the model. To facilitate such calculations, provision has been made to save flow terms for individual cells on disk so they can be used in computations external to the model itself. These individual cell flows are referred to here as "cell-by-cell" flow terms, and are of four general types: (1) cell-by-cell stress flows, or flows into or from an individual cell due to one of the external stresses represented in the model, such as evapotranspiration or recharge; (2) cell-by-cell storage terms, which give the rate of accummulation or depletion of storage in an individual cell; (3) cell-by-cell constant-head flow terms, which give the net flow to or from individual constant-head cells; and (4) internal cell-by-cell flows, which are actually the flows across individual cell faces--that is, between adjacent model cells. These four kinds of cell-by-cell term are further discussed in subsequent paragraphs. To save any of these cell-by-cell terms, two flags in the model input must be set. The input to the Output Control section of the Basic Package includes a flag, ICBCFL, which must be set for each time step for which <u>any</u> cell-by-cell terms are to be saved. In addition, each flow component package includes a flag which is set if the cell-by-cell terms computed by that package are to be saved. Thus if the appropriate flag in the Evapotranspiration Package input is set, cell-by-cell evapotranspiration terms will be saved for each time step for which the ICBCFL flag in the Basic Package input is also set. Three of the four types of cell-by-cell flow terms listed above--storage, constant-head cell and internal flows--are computed in the Block-Centered Flow Package, and thus fall under the control of a single flag, IBCFCB, in the input to that package. Thus in general all three types are saved on disk if this flag is set, and ICBCFL is also set for the time step. Only flow values are saved in the cell-by-cell disk files; neither water volumes nor cumulative water volumes are included. The flow dimensions are volume per unit time, where volume and time are in the same units used for all model input data. The cell-by-cell flow values are stored in unformatted form to make the most efficient use of disk space; see the narrative for the UBUDSV module for information on how the data are written to disk.

Cell-by-cell stress flows are flow rates into or out of the model, at a particular cell, due to one particular external stress. For example, the cell-by-cell evapotranspiration term for cell i,j,k would give the flow out of the model by evapotranspiration from cell i,j,k. Cell-by-cell stress flows are considered positive if flow is into the cell, and negative if it is out of the cell. A cell-by-cell stress flow value is saved for every model cell, for each stress component for which the cell-by-cell flow is requested. That is, an array the size of the model grid is saved on disk for each requested component of flow. For many of the stress components, flow will be zero at most model cells. For example, when using the River Package, there will be nonzero cell-by-cell budget values only at those cells that are traversed by rivers. Thus the amount of disk space required for cell-by-cell flow terms can be large; a flow value is stored for each model cell even when that value is zero, and terms may be saved at many time steps.

The cell-by-cell storage term gives the net flow to or from storage in a variable-head cell. An array of these terms, one for each cell in the mesh is saved in transient simulations if the appropriate flags are set. Withdrawal from storage in the cell is considered positive, whereas accumulation in storage is considered negative.

The cell-by-cell constant-head flow term gives the flow into or out of an individual constant-head cell. This term is always associated with the constant-head cell itself, rather than with the surrounding cells which contribute or receive the flow. A constant-head cell may be surrounded by as many as six adjacent variable-head cells. The cell-by-cell calculation provides a single flow value for each constant-head cell, representing the algebraic sum of the flows between that cell and all of the adjacent variable-head cells. A positive value indicates that the net flow is away from the constant-head cell (into the variable-head portion of the mesh); a negative value indicates that the net flow is into the constant-head cell.

The internal cell-by-cell flow values represent flows across the individual faces of a model cell. Three such terms are saved by the Block-Centered Flow Package for each variable-head cell and constant-head cell in the mesh, whenever the appropriate cell-by-cell flags are set. These three terms are flow across the front cell face (between cell i,j,k and i+1,j,k), flow across the right face (between cell i,j,k and i,j+1,k), and flow across the lower face (between cell i,j,k and i,j,k+1). Each of these represents flow between a given cell and a neighboring cell. (Although each cell has six neighbors, only three flow terms are required; flow across the other three sides is accounted for in the calculations of flow for cells adjacent to those sides.) Flows are considered positive if they are in the direction of increasing row number, increasing column number or increasing layer number, and are considered negative if in the opposite directions. These internal cell-by-cell flow values are useful in calculations of the ground-water flow into various subregions of the model, or in constructing flow vectors.

In theory one could calculate a budget identical to the overall budget by using the cell-by-cell flow terms. This is not always true in practice because in some situations the budgets may be summed differently. The cell-by-cell value at a cell for a given stress or flow component is the net flow for that component, which could possibly include two or more flows of the same type, some negative and some positive. Only the net flow for the cell is saved in the cell-by-cell disk file. In the overall budget calculations as performed in the model, on the other hand, positive and negative flows are assembled separately, so that a negative flow at an individual cell would be added to the outflow term and a positive flow at the same cell would be added to the inflow term. Thus if inflow and outflow terms for the entire model are calculated by summing individual cell-by-cell values, they may differ from the corresponding terms as calculated by the model program in the overall budget. However, the difference between inflow and outflow should be the same for either calculation.

Space Allocation

Space in the central memory of the computer used by data arrays and lists is allocated at execution time in a one-dimensional array called the "X" array. The Allocate Procedure contains a module for each package of the model which allocates space needed by that package. The total number of words needed in the X array depends on the type and number of packages required in a simulation and generally will range from 10 to 20 times the number of cells in the grid. The main program contains two statements referring to the length of the X array, both of which appear in the first part of the program listing. In the listing reproduced with this documentation these statements are COMMON X(30000) and LENX = 30000. The number 30000 in the statements refers to the length of the X array; this number must be increased if the storage requirements of the problem exceed 30000 elements.

Three-Dimensional Subscripts for Model Arrays

The conceptualization and implementation sections of this report designate cell locations by row, column, and layer indices in that order (usually designated as i,j,k), as is customary in scientific literature; however, this order of indices is not the most efficient order for array subscripts in the model program. Many model parameters are declared to be three dimensional arrays and accordingly have row, column and layer subscripts. The order of array subscripts in the FORTRAN language determines how data are stored in computer memory. The design of the program is such that the model array subscripts should be in column, row, and layer order for the most efficient memory access on most computers; this order has been used throughout the program. Typically in the program, J is used for the column subscript, I is used for row, and K for layer, but the order is J,I,K rather than I,J,K. It is important to bear in mind this difference in the subscript ordering when comparing the model program to the conceptualization and implementation sections of the report.

Input Structure

The input structure of the program is designed to permit input to be gathered, as it is needed, from many different files. It is based on an element of the FORTRAN language called the unit number, which identifies the file from which the input is to be read (or to which the output is to be written). The user must provide a link between the name of each input or output file and the corresponding unit number; this is generally done externally to the program, through operating system statements.

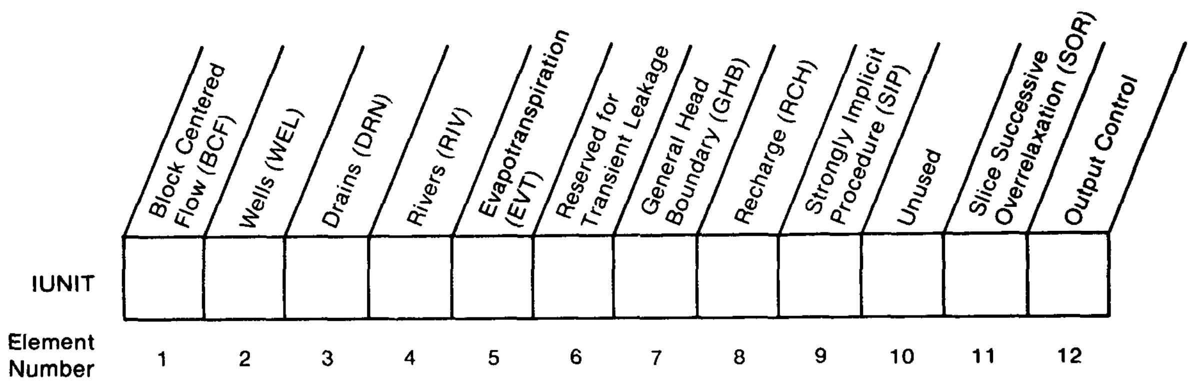
For input purposes, the program may be discussed in terms of "major options"; these are major segments of the program which are utilized only at the user's request. They correspond generally to the individual packages; in fact, all of the existing packages except the Basic Package are considered major options. Output Control, which is not an individual package but rather an optional segment of the Basic Package providing flexibility in program output, is also considered a major option. The balance of the Basic Package is not considered an option since it is always utilized and input for it must always be read. Block-Centered Flow has been treated here as an option, even though it is presently required in all simulations. This has been done to allow for the addition of replacement packages for Block-Centered Flow in the future.

One of the first steps in organizing input data is to specify which of the major options are to be used. This is done using the "IUNIT" array (figure 17) which is read in the Define Procedure by the Basic Package. An option is invoked by inserting an input unit number in the appropriate element of the IUNIT array; if an option is not desired, the value of the element is set to zero. Thus the IUNIT array serves as a flag to indicate whether an option is active, and also serves to specify the unit number containing input data required by the option. For example, if the Drain Package is not to be used, the third element of the IUNIT array (figure 18) is set to zero; if it is to be used, the third element of the array is set to the unit number of the file containing the input data for the package. In the main program, the value of IUNIT (3) is tested in several of the program procedures. If it is zero, the Drain module associated with the procedure is not called. If IUNIT (3) is greater than zero, the subroutine is called and input data is read from the file associated with the unit number.

As noted above, the Basic (BAS) Package, exclusive of the Output Control option, is used for every simulation; and input data for the Basic Package are always required. Basic Package data (figure 18) are read from unit number 1 as specified in the main program. If necessary, the unit number for BAS input can be changed to meet the requirements of a particular computer.

The first element of the IUNIT array must contain the unit number from which data for the Block-Centered Flow (BCF) Package are to be read. At present, because BCF is the only package available for the formulation of cell-to-cell flow terms, a non-zero entry in the first element of IUNIT is always required.

Assignment of Major Options to Elements in the IUNIT Array

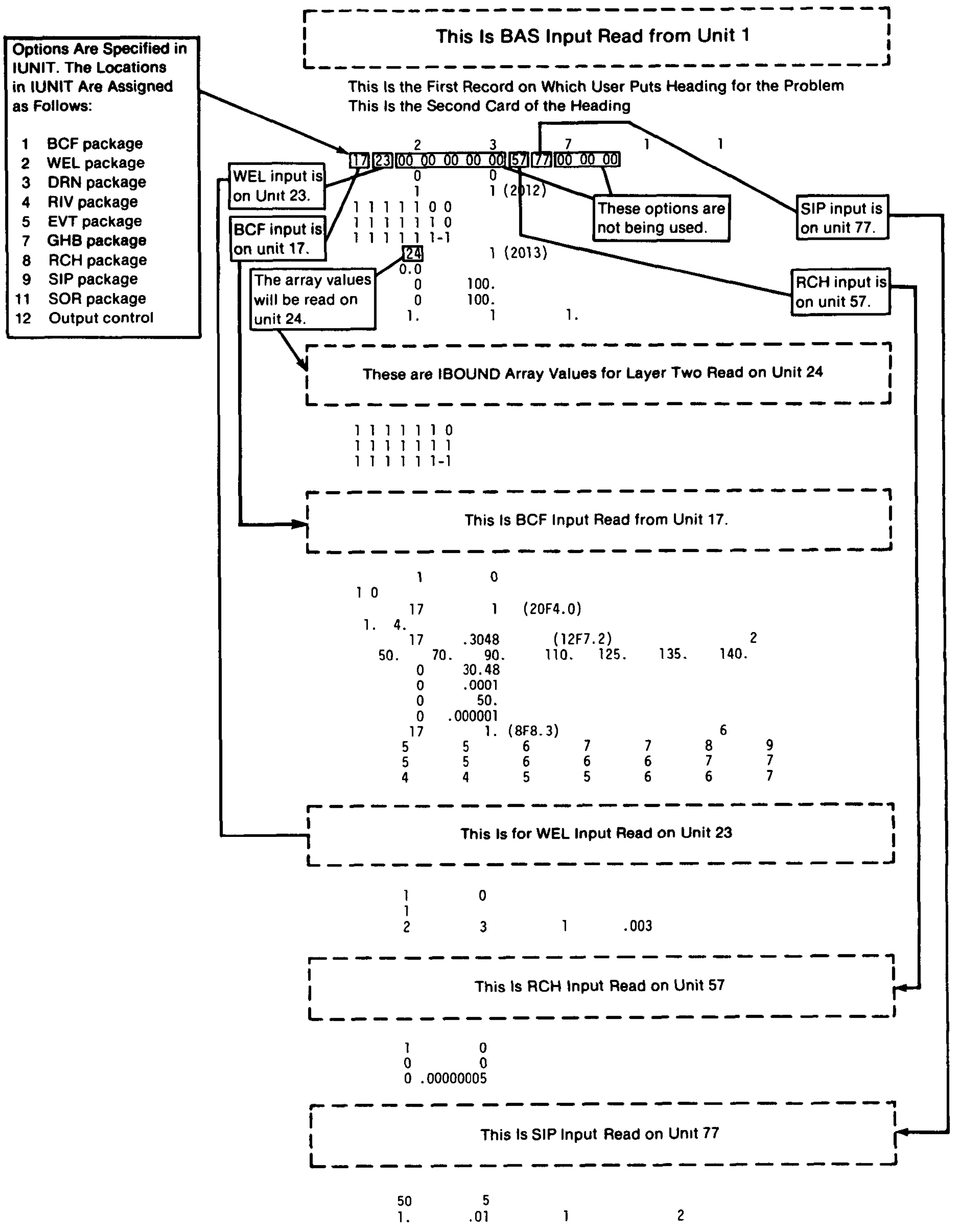

Block Centered Flow (BCF)
Wells (WEL)
Drains (DRN)
Rivers (RIV)
Evapotranspiration (EVT)
Reserved for Transient Leakage
General Head Boundary (GHB)
Recharge (RCH)
Strongly Implicit Procedure (SIP)
Unused
Slice Successive Overrelaxation (SOR)
Output Control
IUNIT
Element Number
1
2
3
4
5
6
7
8
9
10
11
12

Sample IUNIT Input Record

IUNIT	13	41	0	0	81	0	0	0	26	0	0	17
Element Number	1	2	3	4	5	6	7	8	9	10	11	12

1	BCF	Input Is on Unit 13
2	WEL	Input Is on Unit 41
3	DRN	Is Inactive
4	RIV	Is Inactive
5	EVT	Input Is on Unit 81
7	GHB	Is Inactive
8	RCH	Is Inactive
9	SIP	Input Is on Unit 26
11	SOR	Is Inactive
12	Output Control	Input Is on Unit 17

Figure 17.—Specification of major options using the IUNIT array.


Options Are Specified in IUNIT. The Locations in IUNIT Are Assigned as Follows:
1 BCF package
2 WEL package
3 DRN package
4 RIV package
5 EVT package
7 GHB package
8 RCH package
9 SIP package
11 SOR package
12 Output control
This Is BAS Input Read from Unit 1
WEL input is on Unit 23.
BCF input is on unit 17.
The array values will be read on unit 24.
These options are not being used.
SIP input is on unit 77.
RCH input is on unit 57.
These are IBOUND Array Values for Layer Two Read on Unit 24
This Is BCF Input Read from Unit 17.
This Is for WEL Input Read on Unit 23
This Is RCH Input Read on Unit 57
This Is SIP Input Read on Unit 77

```
This Is the First Record on Which User Puts Heading for the Problem
This Is the Second Card of the Heading
        2         3         7         1         1
17 23 00 00 00 00 00 57 77 00 00 00
        0         0
        1         1 (2012)
 1 1 1 1 1 0 0
 1 1 1 1 1 1 0
 1 1 1 1 1 1-1
       24         1 (2013)
      0.0
        0      100.
        0      100.
       1.         1         1.
```

```
 1 1 1 1 1 1 0
 1 1 1 1 1 1 1
 1 1 1 1 1 1-1
```

```
        1         0
 1 0
       17         1   (20F4.0)
  1.  4.
       17     .3048      (12F7.2)                 2
    50.    70.    90.    110.   125.    135.   140.
        0     30.48
        0     .0001
        0       50.
        0   .000001
       17        1. (8F8.3)                     6
       5       5       6       7       7       8       9
       5       5       6       6       6       7       7
       4       4       5       5       6       6       7
```

```
       1         0
       1
       2         3          1         .003
```

```
       1         0
       0         0
       0 .00000005
```

```
      50         5
      1.       .01          1           2
```

Figure 18.—Sample input data showing role of the IUNIT array.

Most of the data submitted by the user will consist of one-dimensional and two-dimensional arrays. Those arrays are submitted as an "array control record" plus, optionally, a series of records containing the array elements. The array control record is read from the unit number specified for the major option which calls for the array. If all the elements of an array have the same value, the value is specified on the control record and it is not necessary to read the associated array. If the elements of the array vary, records containing the array values are read from the unit specified on the array control record according to a format which is also specified in the control record. The unit number may be the same as that from which the control record is read, or it may be different. Thus there is a great deal of flexibility regarding the organization of the input data for a simulation.

Any consistent length and time units may be used for model data. This gives a certain amount of freedom to the user, but care must be exercised to avoid any mixing of units. There is no way for the program to detect the use of inconsistent units. For example, if transmissivity is entered in units of ft^2/day and pumpage as m^3/s, the program will run, but the results will be meaningless.

Output Structure

The output structure is designed to control the amount, type, and frequency of information to be printed or written on disk. It controls the printing of head and drawdown by layer and time step, and the printing of the overall volumetric budget. It also controls disk output of head,