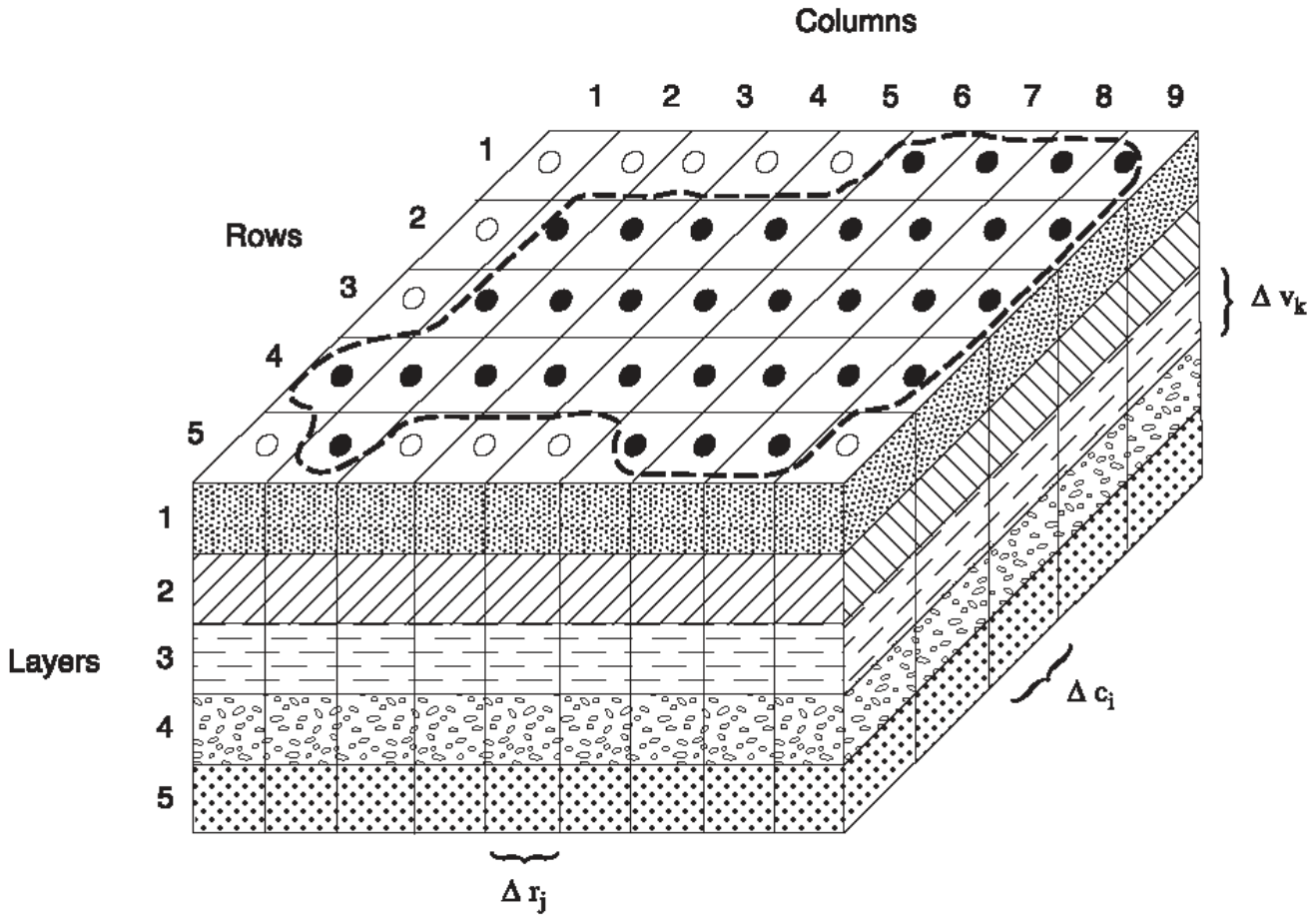
MODFLOW-2005, The U.S. Geological Survey Modular Ground-Water Model—the Ground-Water Flow Process



U.S. Geological Survey Techniques and Methods 6-A16

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

MODFLOW-2005, The U.S. Geological Survey Modular Ground-Water Model—the Ground-Water Flow Process

By Arlen W. Harbaugh

Chapter 16 of
Book 6. Modeling techniques, Section A. Ground Water

U.S. Geological Survey Techniques and Methods 6–A16

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

U.S. Department of the Interior
Gale A. Norton, Secretary

U.S. Geological Survey
P. Patrick Leahy, Acting Director

U.S. Geological Survey, Reston, Virginia: 2005

For product and ordering information:
World Wide Web: *http://www.usgs.gov/pubprod*
Telephone: 1-888-ASK-USGS

For more information on the USGS—the Federal source for science about the Earth, its natural and living resources, natural hazards, and the environment:
World Wide Web: http://www.usgs.gov
Telephone: 1-888-ASK-USGS

Suggested citation:
Harbaugh, A.W., 2005, MODFLOW-2005, The U.S. Geological Survey modular ground-water model—the Ground-Water Flow Process: U.S. Geological Survey Techniques and Methods 6-A16, variously p.

Library of Congress Cataloging-in-Publication Data

PREFACE

This report describes an enhanced version of the U.S. Geological Survey modular ground-water model, called MODFLOW-2005. The model program can be obtained using the Internet at address *http://water.usgs.gov/software/ground_water.html/*. The performance of the program has been tested in a variety of applications. Future applications, however, might reveal errors that were not detected in the test simulations. Users are requested to send notification of any errors found in this User Guide or the model program to:

Office of ground Water
U.S. Geological Survey
411 National Center
Reston, VA 20192
(703) 648-5001

Updates might be made to both the report and to the model program. Users can check for updates at the above Internet address.

Contents

Figures

Tables

MODFLOW-2005, The U.S. Geological Survey Modular Ground-Water Model—the Ground-Water Flow Process

By Arlen W. Harbaugh

Abstract

This report presents MODFLOW-2005, which is a new version of the finite-difference ground-water model commonly called MODFLOW. Ground-water flow is simulated using a block-centered finite-difference approach. Layers can be simulated as confined or unconfined. Flow associated with external stresses, such as wells, areal recharge, evapotranspiration, drains, and rivers, also can be simulated. The report includes detailed explanations of physical and mathematical concepts on which the model is based, an explanation of how those concepts are incorporated in the modular structure of the computer program, instructions for using the model, and details of the computer code.

The modular structure consists of a MAIN Program and a series of highly independent subroutines. The subroutines are grouped into "packages." Each package deals with a specific feature of the hydrologic system that is to be simulated, such as flow from rivers or flow into drains, or with a specific method of solving the set of simultaneous equations resulting from the finite-difference method. Several solution methods are incorporated, including the Preconditioned Conjugate-Gradient method. The division of the program into packages permits the user to examine specific hydrologic features of the model independently. This also facilitates development of additional capabilities because new packages can be added to the program without modifying the existing packages. The input and output systems of the computer program also are designed to permit maximum flexibility.

The program is designed to allow other capabilities, such as transport and optimization, to be incorporated, but this report is limited to describing the ground-water flow capability. The program is written in Fortran 90 and will run without modification on most computers that have a Fortran 90 compiler.

CHAPTER 1
Introduction

MODFLOW-2005 is a new version of the finite-difference ground-water model commonly called MODFLOW. This report documents the ground-water flow part of MODFLOW-2005. The primary objectives of the new version are the same as for prior versions: the program can be readily understood and modified, is simple to use and maintain, easily executed on a variety of computers with minimal changes, and is efficient with respect to computer memory and execution time.

History

Prior to the development of MODFLOW, the two- and three-dimensional finite-difference models described by Trescott (1975), Trescott and Larson (1976), and Trescott, Pinder, and Larson (1976) were used extensively by the U.S. Geological Survey (USGS) and others for the computer simulation of ground-water flow. The first version of MODFLOW (McDonald and Harbaugh, 1984) was the result of the need to consolidate all the commonly used simulation capabilities into a single code that was easy to understand, use, and modify. This first version was developed between the spring of 1981 and the winter of 1983. That model code was originally called the USGS Modular Three-Dimensional Finite-Difference Ground-Water Flow Model, but the model became known as MODFLOW several years later. This was developed using the Fortran 66 computer language.

Revised documentation was released in the report series Techniques of Water Resources Investigations (TWRI) (McDonald and Harbaugh, 1988). The program was largely the same as the 1984 version, but small changes were made to make the code conform to Fortran 77 rather than Fortran 66. This first version of MODFLOW is called MODFLOW-88.

By the early 1990s, MODFLOW had become the most widely used ground-water flow model both within and outside the USGS. Many additions had been made to expand MODFLOW's capabilities. For example, more elaborate representation of the relation between streams and an aquifer was developed (Prudic, 1989). Leake and Prudic (1991) developed a package to represent subsidence. Two preconditioned conjugate-gradient packages were developed (Kuiper, 1987; Hill, 1990). An overall update to MODFLOW, called MODFLOW-96, was published (Harbaugh and McDonald, 1996a and 1996b). MODFLOW-96 was a relatively minor update primarily to improve ease of use.

MODFLOW was originally conceived solely as a ground-water flow model. The authors viewed the solution of additional related equations as something to be done in separate programs. An example of a related equation is a transport equation that uses flows computed by the ground-water flow equation. Another example is parameter estimation, which solves an additional equation to compute optimal hydraulic parameters that result in the best match to real-world observations. By the late 1990s, there was a growing belief by many developers of modeling programs that combining such related capabilities into a single program promised to make development and use easier; therefore, the decision was made to broaden the scope of MODFLOW to allow capabilities such as transport and parameter estimation to be directly incorporated.

To facilitate the incorporation of related equations into MODFLOW, an expansion of the modular design was required. The result, which became MODFLOW-2000 (Harbaugh and others, 2000), was the addition of "Process," which is defined as parts of the code that solve a major equation or set of related equations. The part of the code that solves the ground-water flow equation became the Ground-Water Flow (GWF) Process. Three processes, Observation, Sensitivity and Parameter Estimation, aid calibration and model evaluation (Hill and others, 2000). Solution of the transport equation is the Ground-Water Transport Process (Konikow, Goode, and Hornberger, 1996) and the management of ground-water is the Ground-Water Management Process (Ahlfeld, Barlow, and Mulligan, 2005).

MODFLOW-2005 is similar in design to MODFLOW-2000. The expanded concept of processes continues as in MODFLOW-2000. The primary change in MODFLOW-2005 is the incorporation of a different approach for

managing internal data. Fortran modules are used to declare data that can be shared among subroutines. This allows data to be shared without using subroutine arguments. As a result of using Fortran modules, a change in terminology for MODFLOW has been made. MODFLOW subroutines were originally called modules in a generic sense. The generic term module has been eliminated and replaced by the term subroutine.

Overview of Design

The goals for MODFLOW-2005, as for all prior versions of MODFLOW, can be easily stated: the program is easy to understand, use, enhance, and modify. These goals interrelate to some degree; for example, it would be difficult to use the code if it could not be understood. MODFLOW uses a modular structure wherein similar program functions are grouped together, and specific computational and hydrologic options are constructed in such a manner that each option is independent of other options. Because of this structure, new options can be added without the necessity of changing existing options. The model may be used for either two- or three-dimensional applications. Input procedures have been designed so that each type of model input data may be stored and read from separate external files. User-specified formatting allows input data for the grid to be read in almost any format without modification to the program. The output of model results is also flexible; the user may select which data to output, the frequency of output, and for some data, the format of the output.

MODFLOW-2005 is written in the Fortran 90 (American National Standards Institute, 1992) programming language. Programs written in standard Fortran are highly portable. Use of non-standard features has been carefully avoided so that MODFLOW-2005 will run, without modification, on most computers. Minor modification, however, may be necessary or desirable on some computers.

The GWF Process of MODFLOW has been divided into "packages." A package is the part of the program that deals with a single aspect of simulation. For example, the Well Package simulates the effect of wells, the River Package simulates the effect of rivers, and the Strongly Implicit Procedure Package solves the system of simultaneous finite-difference equations. Many of the packages represent options that the user may or may not have occasion to use. The fundamental method in Fortran for dividing a program into pieces is subroutines, so each package consists of multiple subroutines. The MAIN Program calls the various subroutines of the packages in the proper sequence to simulate ground-water flow.

When additional processes are included, each process is similarly divided into packages. A single code for all processes combined could become quite large. Accordingly, the authors consider it acceptable to have multiple versions of MODFLOW that consist of various combinations of processes rather than one giant version containing all processes. Thus, MODFLOW is not necessarily a single program, but all MODFLOW programs include the GWF Process.

MODFLOW-2005 makes use of Fortran modules for storing and sharing data. Each package includes one or more Fortran modules that declare the shared data for that package. The modules are designed so that data for multiple model grids can be simultaneously defined. Support for multiple grids makes it possible to incorporate local grid refinement (Mehl and Hill, 2004) into MODFLOW. The details of writing code that supports multiple grids are included in Chapter 9; however, new packages added to MODFLOW are not required to support multiple grids.

Organization and Scope of This Report

MODFLOW-2005 is being released initially with only the Ground-Water Flow (GWF) Process. This report consolidates the documentation of the GWF Process. The documentation since MODFLOW-88 has been increasingly dispersed. Documentation for MODFLOW-96 and MODFLOW-2000 was not complete by itself and referred extensively to the MODFLOW-88 documentation. This report documenting MODFLOW-2005 is similar to McDonald and Harbaugh (1988); the fundamental concepts, programmer information, and user input instructions for ground-water flow are entirely contained in this one report. There will still be a need to refer to additional reports for capabilities added to MODFLOW, including additions of processes and capabilities to simulate additional hydrologic features.

The purpose of this report is to describe the mathematical concepts used in the GWF Process of MODFLOW-2005, the program design, the input needed to use it, and programming details. Two preliminary chapters describe topics relating to the overall program; Chapter 2 derives the finite-difference equation that is used in the model, and Chapter 3 describes the overall design of the program. The core packages are described in Chapters 4-7. Chapter 4 describes the Basic Package, which handles a number of administrative tasks for the program as a whole. Because the Basic Package deals with administrative tasks for the entire program, part of Chapter 4 also deals with overall program design. Chapter 5 describes details of how flow through a porous medium is simulated. Chapter 6 describes the simulation of hydrologic stresses, or external sources and sinks. The description of each stress consists of the physical and mathematical concepts and the derivation of the terms that incorporate the stress into the flow equation. For example, in the section on the River Package, an equation is derived that approximates flow through a riverbed, and a discussion is provided to show how that equation can be incorporated into the finite-difference equation. Chapter 7 describes numerical solvers. Chapter 8 describes input instructions for all of the packages, including instructions for the utility subroutines that are used by various packages to perform common tasks. Chapter 9 provides programmer documentation. A sample problem is included in an appendix.

The programmer documentation in Chapter 9 defines the shared data for the GWF Process as a whole and for each package within the GWF Process. The programmer documentation also includes a list of the subroutines in each package, followed by descriptions of the subroutines. The subroutines are not fully documented for all packages because many packages are similar. Only one package of each type is fully documented. The solvers are rarely modified by users, so these are not fully documented here. In situations where this information is needed, prior references can be used.

The packages documented in this report are:

Basic
Block-Centered Flow
Layer-Property Flow
Horizontal Flow Barrier
Well
Recharge
General-Head Boundary
River
Drain
Evapotranspiration
Strongly Implicit Procedure
Preconditioned Conjugate Gradient
Direct Solver

CHAPTER 2
DERIVATION OF THE FINITE-DIFFERENCE EQUATION

Mathematical Model

The three-dimensional movement of ground water of constant density through porous earth material may be described by the partial-differential equation

$$\frac{\partial}{\partial x}\left(K_{xx}\frac{\partial h}{\partial x}\right)+\frac{\partial}{\partial y}\left(K_{yy}\frac{\partial h}{\partial y}\right)+\frac{\partial}{\partial z}\left(K_{zz}\frac{\partial h}{\partial z}\right)+W=S_s\frac{\partial h}{\partial t}, \qquad (2\text{–}1)$$

where

K_{xx}, K_{yy}, and K_{zz} are values of hydraulic conductivity along the x, y, and z coordinate axes, which are assumed to be parallel to the major axes of hydraulic conductivity (L/T);

h is the potentiometric head (L);

W is a volumetric flux per unit volume representing sources and/or sinks of water, with $W<0.0$ for flow out of the ground-water system, and $W>0.0$ for flow into the system (T^{-1});

S_S is the specific storage of the porous material (L^{-1}); and

t is time (T).

For a derivation of equation 2–1 see for example Rushton and Redshaw (1979). In general, S_s, K_{xx}, K_{yy}, and K_{zz} may be functions of space ($S_s = S_s(x,y,z)$, $K_{xx} = K_{xx}(x,y,z)$, and so forth) and W may be a function of space and time ($W = W(x,y,z,t)$). Equation 2–1 describes ground-water flow under nonequilibrium conditions in a heterogeneous and anisotropic medium, provided the principal axes of hydraulic conductivity are aligned with the coordinate directions. Equation 2–1, together with specification of flow and/or head conditions at the boundaries of an aquifer system and specification of initial-head conditions, constitutes a mathematical representation of a ground-water flow system. A solution of equation 2–1, in an analytical sense, is an algebraic expression giving h(x,y,z,t) such that, when the derivatives of h with respect to space and time are substituted into equation 2–1, the equation and its initial and boundary conditions are satisfied. A time-varying head distribution of this nature characterizes the flow system, in that it measures both the energy of flow and the volume of water in storage, and can be used to calculate directions and rates of movement.

Except for very simple systems, analytical solutions of equation 2–1 are rarely possible, so various numerical methods must be employed to obtain approximate solutions. One such approach is the finite-difference method, wherein the continuous system described by equation 2–1 is replaced by a finite set of discrete points in space and time, and the partial derivatives are replaced by terms calculated from the differences in head values at these points. The process leads to systems of simultaneous linear algebraic difference equations; their solution yields values of head at specific points and times. These values constitute an approximation to the time-varying head distribution that would be given by an analytical solution of the partial-differential equation of flow.

The finite-difference analog of equation 2–1 may be derived by applying the rules of difference calculus; however, in the discussion presented here, an alternative approach is used with the aim of simplifying the mathematical treatment and explaining the computational procedure in terms of familiar physical concepts regarding the flow system.

Discretization Convention

Figure 2–1 shows a spatial discretization of an aquifer system with a grid of blocks called cells, the locations of which are described in terms of rows, columns, and layers. An i,j,k indexing system is used. For a system consisting of "NROW" rows, "NCOL" columns, and "NLAY" layers, i is the row index, i = 1,2,. . . NROW; j is the column index, j = 1,2,. . . NCOL; and k is the layer index, k = 1,2,. . . NLAY. For example, figure 2–1 shows a system with NROW = 5, NCOL = 9, and NLAY = 5. In formulating the equations of the model, an assumption was made that layers would generally correspond to horizontal geohydrologic units or intervals. Thus in terms of Cartesian coordinates, the k index denotes changes along the vertical, z; because the convention followed in this model is to number layers from the top down, an increment in the k index corresponds to a decrease in elevation. Similarly, rows would be considered parallel to the x axis, so that increments in the row index, i, would correspond to decreases in y; and columns would be considered parallel to the y axis, so that increments in the column index, j, would correspond to increases in x. These conventions were followed in constructing figure 2–1; however, applications of the model requires only that rows and columns fall along consistent orthogonal directions within the layers, and does not require the designation of x, y, or z coordinate axes.

Within each cell there is a point called a "node" at which head is to be calculated. Many schemes for locating nodes in cells could be used; however, the finite-difference equation developed in the following section uses the block-centered formulation in which the nodes are at the center of the cells.

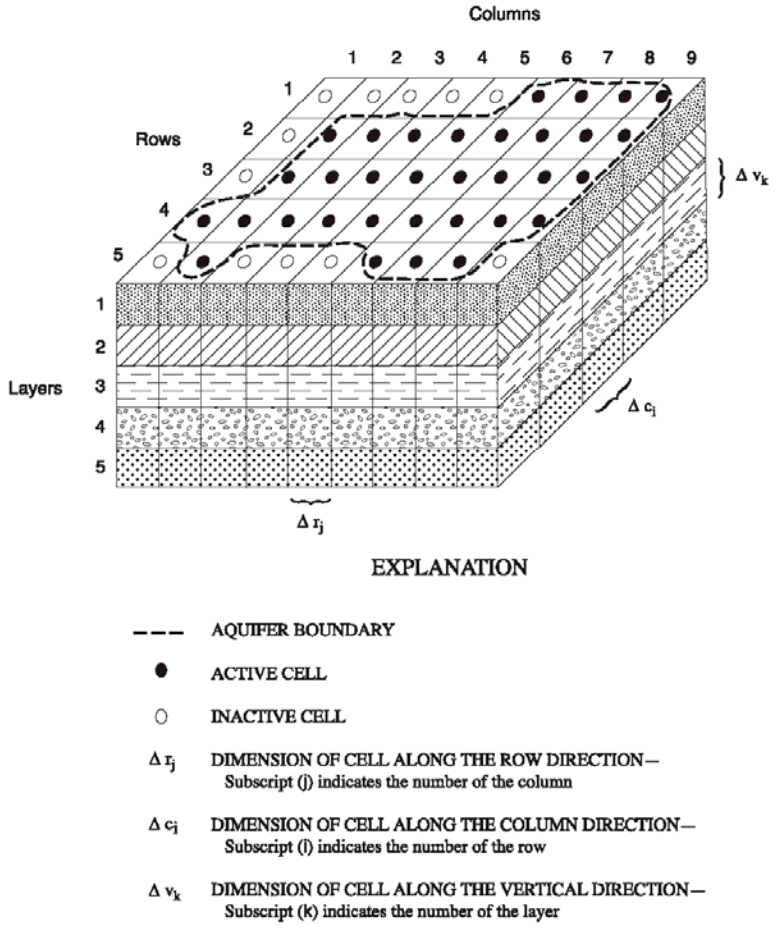


Figure 2–1. A discretized hypothetical aquifer system. (Modified from McDonald and Harbaugh, 1988.)

Following the conventions used in figure 2–1, the width of cells in the row direction, at a given column, j, is designated Δr_j; the width of cells in the column direction at a given row, i, is designated Δc_i; and the thickness of cells in a given layer, k, is designated Δv_k. Thus a cell with coordinates (i,j,k) = (4,8,3) has a volume of $\Delta r_8 \Delta c_4 \Delta v_3$. Note that in Cartesian coordinates with rows parallel to the x axis and columns parallel to the y axis, Δr_j would correspond to Δx_j, and Δc_i would correspond to Δy_i.

In the development of the finite-difference equation in the following section, the grid in MODFLOW is assumed to be rectangular horizontally and vertically. Later in this chapter (see Conceptual Aspects of Vertical Discretization) and in Chapter 5 (Internal Flow Packages), the possibility of vertical distortion of the grid is discussed.

In equation 2–1, the head, h, is a function of time as well as space so that, in the finite-difference formulation, discretization of the continuous time domain is also required. Time is broken into time steps, and head is calculated at each time step.

Finite-Difference Equation

Development of the ground-water flow equation in finite-difference form follows from the application of the continuity equation: the sum of all flows into and out of the cell must be equal to the rate of change in storage within the cell. Under the assumption that the density of ground water is constant, the continuity equation expressing the balance of flow for a cell is

$$\sum Q_i = SS \frac{\Delta h}{\Delta t} \Delta V \quad , \tag{2–2}$$

where

Q_i is a flow rate into the cell (L^3T^{-1});

SS has been introduced as the notation for specific storage in the finite-difference formulation; its definition is equivalent to that of S_s in equation 2–1—that is, SS is the volume of water that can be injected per unit volume of aquifer material per unit change in head (L^{-1});

ΔV is the volume of the cell (L^3); and

Δh is the change in head over a time interval of length Δt.

The term on the right-hand side is equivalent to the volume of water taken into storage over a time interval Δt given a change in head of Δh. Equation 2–2 is stated in terms of inflow and storage gain. Outflow and loss are represented by defining outflow as negative inflow and loss as negative gain.

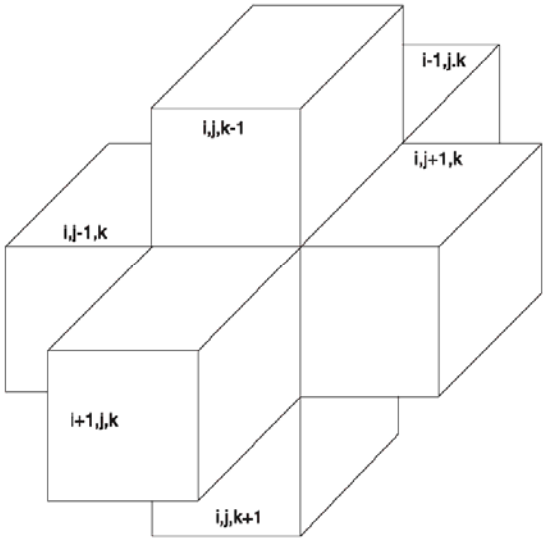


Figure 2–2. Indicies for the six adjacent cells surrounding cell i,j,k (hidden). (Modified from McDonald and Harbaugh, 1988.)

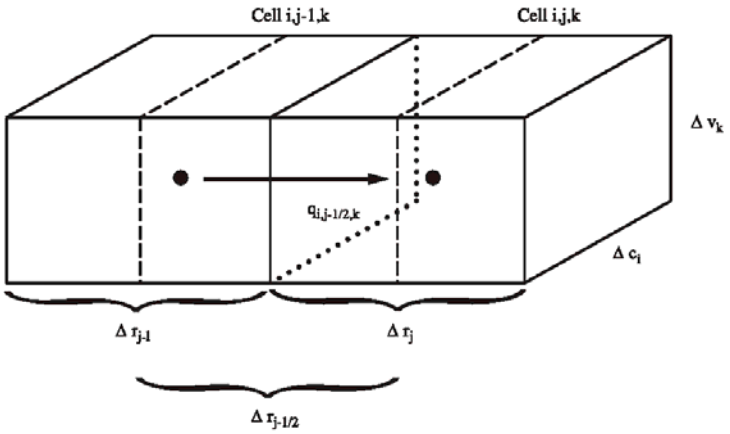


Figure 2–3. Flow into cell i,j,k from cell i,j-1,k. (Modified from McDonald and Harbaugh, 1988.)

Figure 2–2 depicts six aquifer cells adjacent to cell i,j,k — i-1,j,k; i+1,j,k; i,j-1,k; i,j+1,k; i,j,k-1; and i,j,k+1. To simplify the following development, flows are considered positive if they are entering cell i,j,k; the negative sign usually incorporated in Darcy's law has been dropped from all terms. Following these conventions, flow into cell i,j,k in the row direction from cell i,j-1,k (figure 2–3), is given by Darcy's law as

$$q_{i,j-1/2,k} = KR_{i,j-1/2,k}\Delta c_i \Delta v_k \frac{\left(h_{i,j-1,k} - h_{i,j,k}\right)}{\Delta r_{j-1/2}}, \qquad (2\text{–}3)$$

where

$h_{i,j,k}$ is the head at node i,j,k, and $h_{i,j-1,k}$ is the head at node i,j-1,k;

$q_{i,j-1/2,k}$ is the volumetric flow rate through the face between cells i,j,k and i,j-1,k (L^3T^{-1});

$KR_{i,j-1/2,k}$ is the hydraulic conductivity along the row between nodes i,j,k and i,j-1,k (LT^{-1});

$\Delta c_i \Delta v_k$ is the area of the cell faces normal to the row direction; and

$\Delta r_{j-1/2}$ is the distance between nodes i,j,k and i,j-1,k (L).

Equation 2–3 gives the exact flow for a one-dimensional steady-state case through a block of aquifer extending from node i,j-1,k to node i,j,k and having a cross-sectional area $\Delta c_i \Delta v_k$. KR i,j-1/2,k is the conductivity of the material between nodes i,j,k and i,j-1,k, which is the effective hydraulic conductivity for the entire region between the nodes, normally calculated as a harmonic mean in the sense described, for example, by Collins (1961).

The subscript i,j-1/2,k is used in equation 2–3 to designate the region between nodes i,j-1,k and i,j,k. The "1/2" does not indicate a specific point half way between nodes. Thus, $q_{i,j-1/2,k}$ indicates that the flow from node i,j-1,k to node i,j,k, $\Delta r_{j-1/2}$ is the distance between nodes i,j,k and i,j-1/2,k, and KR i,j-1/2,k is the effective hydraulic conductivity between the nodes. The term "1/2" is used in the same manner to indicate the region between nodes in many of the equations throughout this report.

Similar expressions can be written approximating the flow into the cell through the remaining five faces, for example, for flow in the row direction through the face between cells i,j,k and i,j+1,k,

$$q_{i,j+1/2,k} = KR_{i,j+1/2,k}\Delta c_i \Delta v_k \frac{\left(h_{i,j+1,k} - h_{i,j,k}\right)}{\Delta r_{j+1/2}}, \qquad (2\text{–}4)$$

while for the column direction, flow into the block through the front face is

$$q_{i+1/2,j,k} = KC_{i+1/2,j,k}\Delta r_j \Delta v_k \frac{\left(h_{i+1,j,k} - h_{i,j,k}\right)}{\Delta c_{i+1/2}} \tag{2–5}$$

and flow into the block through the rear face is

$$q_{i-1/2,j,k} = KC_{i-1/2,j,k}\Delta r_j \Delta v_k \frac{\left(h_{i-1,j,k} - h_{i,j,k}\right)}{\Delta c_{i-1/2}} . \tag{2–6}$$

For the vertical direction, inflow through the bottom face is

$$q_{i,j,k+1/2} = KV_{i,j,k+1/2}\Delta r_j \Delta c_i \frac{\left(h_{i,j,k+1} - h_{i,j,k}\right)}{\Delta v_{k+1/2}} , \tag{2–7}$$

whereas inflow through the upper face is given by

$$q_{i,j,k-1/2} = KV_{i,j,k-1/2}\Delta r_j \Delta c_i \frac{\left(h_{i,j,k-1} - h_{i,j,k}\right)}{\Delta v_{k-1/2}} . \tag{2–8}$$

Each of equations 2–3 through 2–8 expresses inflow through a face of cell i,j,k in terms of heads, grid dimensions, and hydraulic conductivity. The notation can be simplified by combining grid dimensions and hydraulic conductivity into a single constant, the "hydraulic conductance" or, more simply, the "conductance." For example,

$$CR_{i,j-1/2,k} = \frac{KR_{i,j-1/2,k}\Delta c_i \Delta v_k}{\Delta r_{j-1/2}} , \tag{2–9}$$

where

$CR_{i,j-1/2,k}$ is the conductance in row i and layer k between nodes i,j-1,k and i,j,k (L^2T^{-1}).

Thus, conductance is the product of hydraulic conductivity and cross-sectional area of flow divided by the length of the flow path (in this case, the distance between the nodes).

Substituting conductance from equation 2–9 into equation 2–3 yields

$$q_{i,j-1/2,k} = CR_{i,j-1/2,k}\left(h_{i,j-1,k} - h_{i,j,k}\right) . \tag{2–10}$$

Similarly, equations 2–4 through 2–8 can be rewritten to yield

$$q_{i,j+1/2,k} = CR_{i,j+1/2,k}\left(h_{i,j+1,k} - h_{i,j,k}\right) \tag{2–11}$$

$$q_{i-1/2,j,k} = CC_{i-1/2,j,k}\left(h_{i-1,j,k} - h_{i,j,k}\right) \tag{2–12}$$

$$q_{i+1/2,j,k} = CC_{i+1/2,j,k}\left(h_{i+1,j,k} - h_{i,j,k}\right) \tag{2–13}$$

$$q_{i,j,k-1/2} = CV_{i,j,k-1/2}\left(h_{i,j,k-1} - h_{i,j,k}\right) \tag{2–14}$$

$$q_{i,j,k+1/2} = CV_{i,j,k+1/2}\left(h_{i,j,k+1} - h_{i,j,k}\right) , \tag{2–15}$$

where the conductances are defined analogously to CR in equation (2–9).

Equations 2–10 through 2–15 account for the flow into cell i,j,k from the six adjacent cells. To account for flows into the cell from features or processes external to the aquifer, such as rivers, drains, areal recharge, evapotranspiration, or wells, additional terms are required. These flows may be dependent on the head in the receiving cell but independent of all other heads in the aquifer, or they may be entirely independent of the head in the receiving cell. Flow from outside the aquifer may be represented by the expression

$$a_{i,j,k,n} = p_{i,j,k,n} h_{i,j,k} + q_{i,j,k,n} \quad , \tag{2–16}$$

where

$a_{i,j,k,n}$ represents flow from the nth external source into cell i,j,k (L^3T^{-1}), and

$p_{i,j,k,n}$ and $q_{i,j,k,n}$ are constants ((L^2T^{-1}) and (L^3T^{-1}), respectively).

For example, suppose a cell is receiving flow from two sources, recharge from a well and seepage through a riverbed. For the first source (n=1), because the flow from the well is assumed to be independent of head, $p_{i,j,k,1}$ is zero and $q_{i,j,k,1}$ is the recharge rate for the well. In this case,

$$a_{i,j,k,1} = q_{i,j,k,1} \quad . \tag{2–17}$$

For the second source (n=2), the assumption is made that the river-aquifer interconnection can be treated as a simple conductance, so that the seepage is proportional to the head difference between the river stage and the head in cell i,j,k (figure 2–4); thus we have

$$a_{i,j,k,2} = \mathrm{CRIV}_{i,j,k}\left(R_{i,j,k} - h_{i,j,k}\right) \quad , \tag{2–18}$$

where

$R_{i,j,k}$ is the head in the river (L), and

$\mathrm{CRIV}_{i,j,k}$ is a conductance (L^2T^{-1}) controlling flow from the river into cell i,j,k.

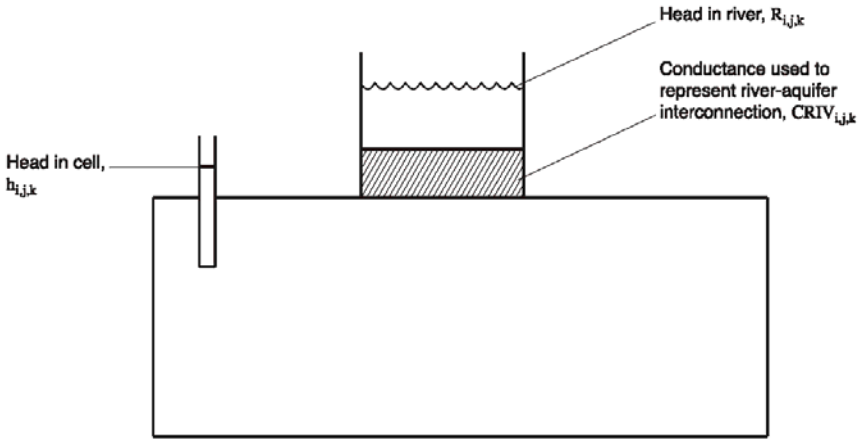


Figure 2–4. Conceptual representation of leakage through a riverbed into a cell. (Modified from McDonald and Harbaugh, 1988.)

For example, in the situation shown schematically in figure 2–4, CRIV would be given as the product of the vertical hydraulic conductivity of the riverbed material and the area of the riverbed as it crosses the cell, divided by the thickness of the riverbed material. Equation 2–18 can be rewritten as

$$a_{i,j,k,2} = -CRIV_{i,j,k} h_{i,j,k} + CRIV_{i,j,k} R_{i,j,k} \quad . \tag{2–19}$$

The negative conductance term, $-CRIV_{i,j,k}$ corresponds to $p_{i,j,k,2}$ of equation 2–16, while the term $CRIV_{i,j,k}R_{i,j,k}$ corresponds to $q_{i,j,k,2}$. Similarly, all other external sources or stresses can be represented by an expression of the form of equation 2–16. In general, if there are N external sources or stresses affecting a single cell, the combined flow is expressed by

$$\sum_{n=1}^{N} a_{i,j,k,n} = \sum_{n=1}^{N} \left(p_{i,j,k,n} h_{i,j,k} \right) + \sum_{n=1}^{N} q_{i,j,k,n} \quad . \tag{2–20}$$

Defining $P_{i,j,k}$ and $Q_{i,j,k}$ by the expressions

$$P_{i,j,k} = \sum_{n=1}^{N} p_{i,j,k,n}$$

and

$$Q_{i,j,k} = \sum_{n=1}^{N} q_{i,j,k,n}$$

the general external flow term for cell i,j,k is

$$\sum_{n=1}^{N} a_{i,j,k,n} = P_{i,j,k} h_{i,j,k} + Q_{i,j,k} \quad . \tag{2–21}$$

Applying the continuity equation 2–2 to cell i,j,k, taking into account the flows from the six adjacent cells, change in storage, and the external flow rate yields

$$\begin{aligned} & q_{i,j-1/2,k} + q_{i,j+1/2,k} + q_{i-1/2,j,k} + q_{i+1/2,j,k} \\ & + q_{i,j,k-1/2} + q_{i,j,k+1/2} + P_{i,j,k} h_{i,j,k} + Q_{i,j,k} = SS_{i,j,k} \left(\Delta r_j \Delta c_i \Delta v_k \right) \frac{\Delta h_{i,j,k}}{\Delta t} \quad , \end{aligned} \tag{2–22}$$

where

$\Delta h_{i,j,k}/\Delta t$ is a finite-difference approximation for the derivative of head with respect to time (LT^{-1});

$SS_{i,j,k}$ represents the specific storage of cell i,j,k (L^{-1}); and

$\Delta r_j \Delta c_i \Delta v_k$ is the volume of cell i,j,k (L^3).

Equations 2–10 through 2–15 may be substituted into equation 2–22 to give the finite-difference approximation for cell i,j,k as

$$\begin{aligned} & CR_{i,j-1/2,k} \left(h_{i,j-1,k} - h_{i,j,k} \right) + CR_{i,j+1/2,k} \left(h_{i,j+1,k} - h_{i,j,k} \right) \\ & + CC_{i-1/2,j,k} \left(h_{i-1,j,k} - h_{i,j,k} \right) + CC_{i+1/2,j,k} \left(h_{i+1,j,k} - h_{i,j,k} \right) \\ & + CV_{i,j,k-1/2} \left(h_{i,j,k-1} - h_{i,j,k} \right) + CV_{i,j,k+1/2} \left(h_{i,j,k+1} - h_{i,j,k} \right) \\ & + P_{i,j,k} h_{i,j,k} + Q_{i,j,k} = SS_{i,j,k} \left(\Delta r_j \Delta c_i \Delta v_k \right) \frac{\Delta h_{i,j,k}}{\Delta t} \quad . \end{aligned} \tag{2–23}$$

The finite-difference approximation for the time derivative of head, $\Delta h_{i,j,k}/\Delta t$ must next be expressed in terms of specific heads and times. Figure 2–5 shows a hydrograph of head values at node i,j,k. Two values of time are shown on the horizontal axis: t^m, the time at which the flow terms of equation (2–23) are evaluated; and t^{m-1}, a time that precedes t^m. The head values at node i,j,k associated with these times are designated by superscript as $h^m_{i,j,k}$ and $h^{m-1}_{i,j,k}$, respectively. An approximation to the time derivative of head at time t^m is obtained by dividing the head difference $h^m_{i,j,k} - h^{m-1}_{i,j,k}$ by the time interval $t^m - t^{m-1}$; that is,

$$\frac{\Delta h_{i,j,k}}{\Delta t} \cong \frac{h^m_{i,j,k} - h^{m-1}_{i,j,k}}{t^m - t^{m-1}} \quad .$$

Thus the hydrograph slope, or time derivative, is approximated using the change in head at the node over a time interval that precedes, and ends with, the time at which flow is evaluated. This is termed a backward-difference approach, in that $\Delta h/\Delta t$ is approximated over a time interval that extends backward in time from t^m, the time at which the flow terms are calculated.

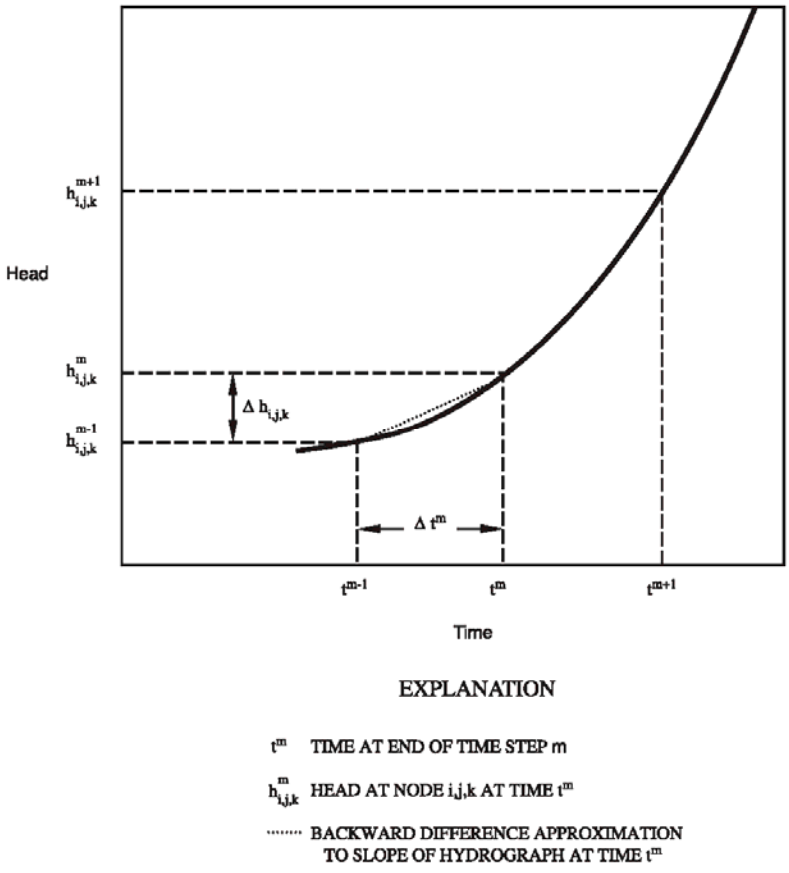


Figure 2–5. Hydrograph for cell i,j,k. (Modified from McDonald and Harbaugh, 1988.)

The time derivative of head could be approximated in other ways; for example, we could approximate the time derivative of head over a time interval that begins at the time of flow evaluation and extends to some later time; or over a time interval that is centered at the time of flow evaluation, extending both forward and backward from it. These alternatives, however, may cause numerical instability—that is, the growth or propagation of error during the calculation of heads at successive times in a simulation. In an unstable situation, errors that enter the calculation for any reason at a particular time will increase at each succeeding time as the calculation progresses, until finally the errors completely dominate the result. By contrast, the backward-difference approach is always numerically stable—that is, errors introduced at any time diminish progressively at succeeding times. For this reason, the

backward-difference approach is preferred even though this approach leads to large systems of equations that must be solved simultaneously for each time step.

Equation 2–23 can be rewritten in backward-difference form by specifying flow terms at t^m, the end of the time interval, and approximating the time derivative of head over the interval t^{m-1} to t^m; that is:

$$\begin{aligned} & CR_{i,j-1/2,k}\left(h^m_{i,j-1,k} - h^m_{i,j,k}\right) + CR_{i,j+1/2,k}\left(h^m_{i,j+1,k} - h^m_{i,j,k}\right) \\ & + CC_{i-1/2,j,k}\left(h^m_{i-1,j,k} - h^m_{i,j,k}\right) + CC_{i+1/2,j,k}\left(h^m_{i+1,j,k} - h^m_{i,j,k}\right) \\ & + CV_{i,j,k-1/2}\left(h^m_{i,j,k-1} - h^m_{i,j,k}\right) + CV_{i,j,k+1/2}\left(h^m_{i,j,k+1} - h^m_{i,j,k}\right) \\ & + P_{i,j,k}h^m_{i,j,k} + Q_{i,j,k} = SS_{i,j,k}\left(\Delta r_j \Delta c_i \Delta v_k\right)\frac{h^m_{i,j,k} - h^{m-1}_{i,j,k}}{t^m - t^{m-1}} \end{aligned} \quad . \qquad (2\text{–}24)$$

Equation 2–24 is a backward-difference equation that can be used as the basis for a simulation of the partial-differential equation of ground-water flow, equation 2–1. Like the term $Q_{i,j,k}$, the coefficients of the various head terms in equation 2–24 are all known, as is the head at the beginning of the time step, $h^{m-1}_{i,j,k}$. The seven heads at time t^m, the end of the time step, are unknown; that is, they are part of the head distribution to be predicted. Thus equation 2–24 cannot be solved independently, because it represents a single equation in seven unknowns. An equation of this type, however, can be written for each active cell in the grid; and, because only one unknown head exists for each cell, we are left with a system of "n" equations in "n" unknowns. Such a system can be solved simultaneously.

The objective of transient simulation is generally to predict head distributions at successive times, given the initial head distribution, the boundary conditions, the hydraulic parameters, and the external stresses. The initial-head distribution provides a value of $h^0_{i,j,k}$ at each point in the grid—that is, the initial head provides the values of head at the beginning of the first of the discrete time steps into which the time axis is divided in the finite-difference process. The first step in the solution process is to calculate values of $h^1_{i,j,k}$—that is, heads at time t^1, which marks the end of the first time step. In equation 2–24, therefore, the head superscript m is taken as 1, while the superscript m-1, which appears in only one head term, is taken as 0. Equation 2–24 then becomes

$$\begin{aligned} & CR_{i,j-1/2,k}\left(h^1_{i,j-1,k} - h^1_{i,j,k}\right) + CR_{i,j+1/2,k}\left(h^1_{i,j+1,k} - h^1_{i,j,k}\right) \\ & + CC_{i-1/2,j,k}\left(h^1_{i-1,j,k} - h^1_{i,j,k}\right) + CC_{i+1/2,j,k}\left(h^1_{i+1,j,k} - h^1_{i,j,k}\right) \\ & + CV_{i,j,k-1/2}\left(h^1_{i,j,k-1} - h^1_{i,j,k}\right) + CV_{i,j,k+1/2}\left(h^1_{i,j,k+1} - h^1_{i,j,k}\right) \\ & + P_{i,j,k}h^1_{i,j,k} + Q_{i,j,k} = SS_{i,j,k}\left(\Delta r_j \Delta c_i \Delta v_k\right)\frac{h^1_{i,j,k} - h^0_{i,j,k}}{t^1 - t^0} \end{aligned} \quad , \qquad (2\text{–}25)$$

where again, the superscripts 0 and 1 refer to the time at which the heads are taken and should not be interpreted as exponents.

An equation of this form is written for every cell in the grid in which head is free to vary with time (variable-head cells), and the system of equations is solved simultaneously for the heads at time t^1. When these have been obtained, the process is repeated to obtain heads at time t^2, the end of the second time step. To do this, equation 2–24 is reapplied, now using 1 as time superscript m-1 and 2 as time superscript m. Again, a system of equations is formulated, where the unknowns are now the heads at time t^2; and this set of equations is solved simultaneously to obtain the head distribution at time t^2. This process is continued for as many time steps as necessary to cover the time range of interest.

The set of finite-difference equations is reformulated at each time step; that is, at each time step there is a new system of simultaneous equations to be solved. The heads at the end of the time step make up the unknowns for which this system must be solved; the heads at the beginning of the step are among the known terms in the equations. The solution process is repeated at each time step yielding a new array of heads for the end of the step.

The finite-difference flow equation for a cell is a representation of the volumetric flow from all sources in units of L^3/T, where L is a length unit and T is a time unit. Provided that consistent length and time units are used for all terms, any specific units can be used. For example, length units could be feet and time units could be days, or length units could be meters and time units could be seconds. This gives a certain amount of freedom to the user, but care must be exercised to avoid mixing units when specifying the input data. The use of inconsistent units cannot be detected. For example, if conductance is specified in units of square feet per day and an external recharge rate is given in units of cubic meters per second, the resulting flow equation will be meaningless.

Iteration

MODFLOW utilizes iterative methods to obtain the solution to the system of finite-difference equations for each time step. In these methods, the calculation of head values for the end of a given time step is started by arbitrarily assigning a trial value, or estimate, for the head at each node at the end of that step. A procedure of calculation is then initiated that alters these estimated values, producing a new set of head values that are in closer agreement with the system of equations. These new, or interim, head values then take the place of the initially assumed heads, and the procedure of calculation is repeated, producing a third set of head values. This procedure is repeated successively at each stage, producing a new set of interim heads that more nearly satisfies the system of equations. Each repetition of the calculation is termed an "iteration." Ultimately, as the interim heads approach values that would exactly satisfy the set of equations, the changes produced by succeeding stages of calculation become very small. This behavior is utilized in determining when to stop iteration, as discussed in a subsequent paragraph.

Thus, during the calculations for a time step, arrays of interim head values are generated in succession, each array containing one interim head value for each active node in the grid. In figure 2–6, these arrays are represented as three-dimensional lattices, each identified by an array symbol, h, bearing two superscripts. The first superscript indicates the time step for which the heads in the array are calculated, whereas the second indicates the number, or level, of the iteration that produced the head array. Thus $h^{m,1}$ represents the array of values computed in the first iteration for time step m; $h^{m,2}$ would represent the array of values computed in the second iteration, and so on. The head values that were initially assumed for time step m, to begin the process of iteration, appear in the array designated $h^{m,0}$. In the example of figure 2–6, a total of n iterations is required to achieve closure for the heads at the end of time step m; thus the array of final head values for the time step is designated $h^{m,n}$. Figure 2–6 also shows the array of final head values for the end of the preceding time step $h^{m-1,n}$ (where again it is assumed that n iterations were required for closure). The head values in this array appear in the storage term of equation 2–24—that is, they are used in the term $h_{i,j,k}^{m-1}$ on the right side of equation 2–24—in the calculation of heads for time step m. Because they represent heads for the preceding time step, for which computations have already been completed, they appear as predetermined constants in the equation for time step m; thus they retain the same value in each iteration of the time step. Similarly, the final values of head for time step m are used as constants in the storage term during calculations for time step m+1.

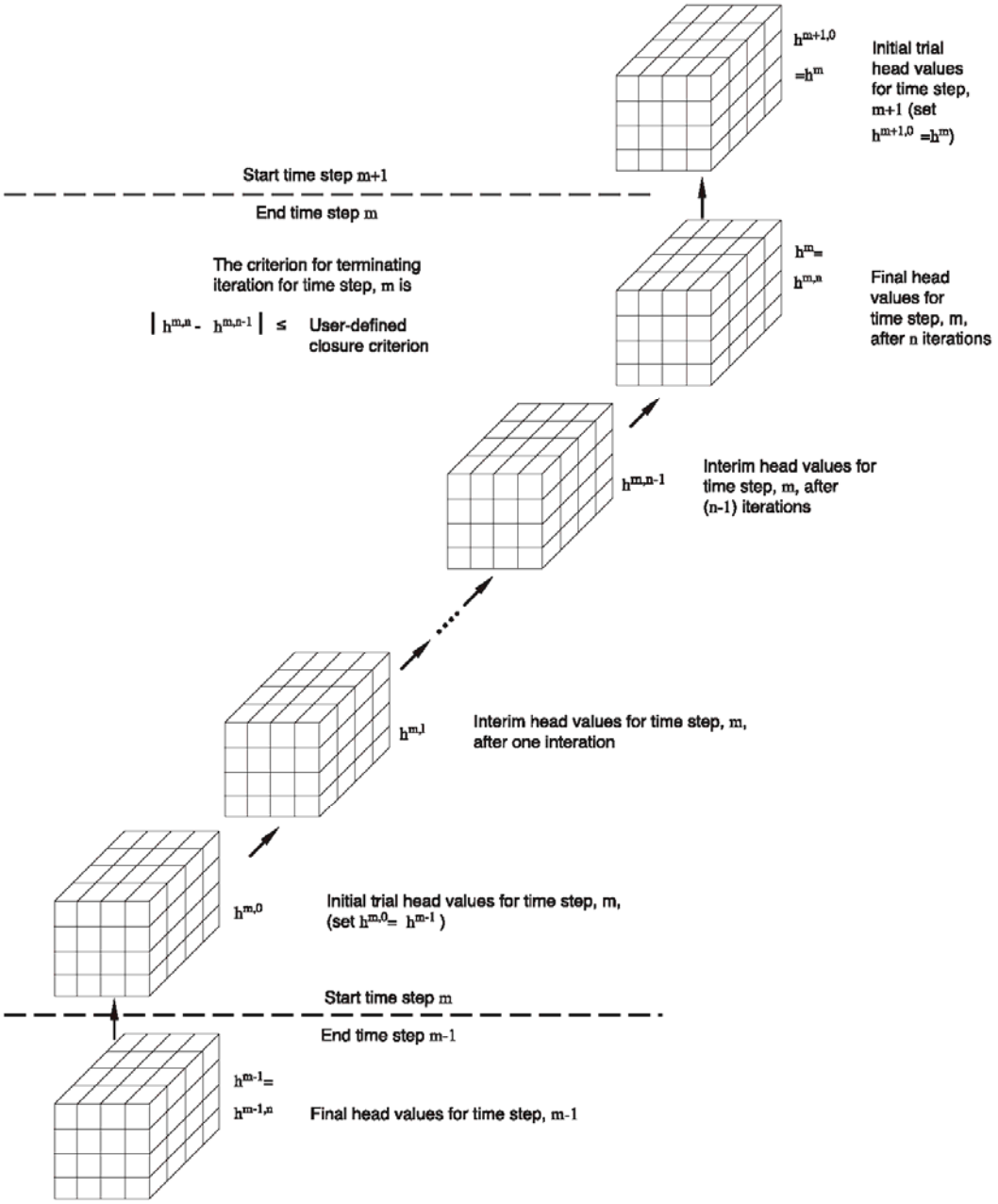


Figure 2–6. Iterative calculation of a head distribution. (From McDonald and Harbaugh, 1988.)

Ideally, one would like to specify that iteration stop when the calculated heads are suitably close to the exact solution. Because the actual solution is unknown, however, an indirect method of specifying when to stop iterating must be used. The method most commonly employed is to specify that the changes in computed heads occurring from one iteration level to the next must be less than a certain quantity, termed the "closure criterion" or "convergence criterion," which is specified by the user. After each iteration, absolute values of computed head change in that iteration are examined for all nodes in the grid. The largest of these absolute head-change values is compared with the closure criterion. If this largest value exceeds the closure criterion, iteration continues; if this value is less than the closure criterion, iteration is said to have "closed" or "converged," and the process is terminated for that time step. Normally, this method of determining when to stop iteration is adequate. Note that the closure criterion refers to change in computed head, and that values of head are not themselves necessarily calculated to a level of accuracy comparable to the closure criterion. As a rule of thumb, it is wise to use a value of closure criterion that is an order of magnitude smaller than the level of accuracy desired in the head results.

MODFLOW also incorporates a maximum permissible number of iterations per time step. If closure is not achieved within this maximum number of iterations, then the iterative process is terminated and a corresponding message is printed in the output.

The initial estimates of head for time step m, in array $h^{m,0}$ of figure 2–6, could be assigned arbitrarily, or they could be chosen according to a number of different conventions. Theoretically, the iterative process would eventually converge to the same result regardless of the choice of initial head values, although the work required would be much greater for some choices than for others. In MODFLOW, the heads computed for the end of each time step are used as the initial trial values of head for the succeeding time step. Thus in figure 2–6, the array $h^{m-1,n}$ contains the final estimates of head, obtained after n iterations, for the end of time step m-1. When the calculations for step m-1 are complete, these same values of head are transferred to the array $h^{m,0}$, and used as the initial estimates, or trial values, for the heads of time step m. Head values for the first time step in the simulation are assumed initially to be equal to the heads specified by the user for the beginning of the simulation.

Discussions of the mathematical basis of various iterative methods can be found in many standard references, including Peaceman (1977), Crichlow (1977) and Remson, Hornberger and Molz (1971). The reader is advised to review one of these discussions, both to clarify general concepts and to provide an introduction to such topics as the use of matrix notation, the role of iteration parameters, and the influence of various factors on rate of convergence. In particular, such a review is recommended prior to reading Chapter 7 of this report.

An iterative procedure yields only an approximation to the solution of the system of finite-difference equations for each time step; the accuracy of this approximation depends upon several factors, including the closure criterion that is employed. Even if exact solutions to the set of finite-difference equations were obtained at each step, these solutions would themselves be only an approximation to the solution of the differential equation of flow (eq. 2–1). The discrepancy between the head, $h^{m}_{i,j,k}$, given by the solution to the system of difference equations for a given node and time and the head h(x,y,z,t), which would be given by the formal solution of the differential equation for the corresponding point and time, is termed the truncation error. In general, this error tends to become greater as the grid spacing and time-step length are increased. The reader also should recognize that even if a formal solution of the differential equation could be obtained, it would be only an approximation to conditions in the field, in that hydraulic conductivity and specific storage are seldom known with accuracy, and uncertainties with regard to hydrologic boundaries are generally present.

Steady-State Simulations

The flow equation, equation 2–24, was developed assuming transient conditions; however, the transient flow equation becomes the steady-state flow equation when the storage term is zero. The resulting equation specifies that the sum of all inflows (where outflow is a negative inflow) from adjacent cells and external stresses must be zero for each cell in the model. A steady-state problem requires only a single solution of simultaneous equations, rather than multiple solutions for multiple time steps. Recall that an initial head was required in a transient simulation to calculate the time derivative for the first time step. For steady-state simulations, there is no direct requirement for initial head because the time derivative is removed from the flow equation. In practice, however, initial head is required for steady-state simulations because of the assumption that iterative solution is used. As already described, iterative solution works by successively improving the estimated answer; therefore, an initial estimate is required to start the iterative process. In MODFLOW, the user-specified initial head is used as this initial estimate. The initial estimate should normally have no effect on the solution to the steady-state flow equation, but it may affect the number of iterations required to obtain an acceptable approximation of the solution.

Formulation of Equations for Solution

MODFLOW incorporates several different options for iterative solution of the set of finite-difference equations, and is organized so that alternative schemes of solution may be added without disruption of the program structure. Whatever scheme of solution is employed, it is convenient to rearrange equation 2–24 so that all terms containing heads at the end of the current time step are grouped on the left-hand side of the equation, and all terms that are independent of head at the end of the current time step are on the right-hand side. All coefficients of $h^{m}_{i,j,k}$ that do not include conductance between nodes are combined into a single term, HCOF, and all right-hand-side terms are

combined into the term RHS. Further, the complexity can be reduced by assuming that the time superscript is m unless otherwise shown. The resulting equation is

$$\begin{aligned} &CV_{i,j,k-1/2}\,h_{i,j,k-1} + CC_{i-1/2,j,k}\,h_{i-1,j,k} + CR_{i,j-1/2,k}\,h_{i,j-1,k} \\ &+ (-CV_{i,j,k-1/2} - CC_{i-1/2,j,k} - CR_{i,j-1/2,k} - CR_{i,j+1/2,k} - CC_{i+1/2,j,k} - CV_{i,j,k+1/2} + HCOF_{i,j,k})\,h_{i,j,k} \\ &+ CR_{i,j+1/2,k}\,h_{i,j+1,k} + CC_{i+1/2,j,k}\,h_{i+1,j,k} + CV_{i,j,k+1/2}\,h_{i,j,k+1} = RHS_{i,j,k}\ , \end{aligned} \qquad (2\text{–}26)$$

where

$$HCOF_{i,j,k} = P_{i,j,k} - \frac{SS_{i,j,k}\,\Delta r_j\,\Delta c_i\,\Delta v_k}{t - t^{m-1}} \qquad (L^2T^{-1}) \text{ and}$$

$$RHS_{i,j,k} = -Q_{i,j,k} - SS_{i,j,k}\,\Delta r_j\,\Delta c_i\,\Delta v_k \frac{h_{i,j,k}^{m-1}}{t - t^{m-1}} \qquad (L^3T^{-1}).$$

The entire system of equations of the form of equation 2–26, which includes one equation for each variable-head cell in the grid, may be written in matrix form as

$$[A]\{h\} = \{q\}\ , \qquad (2\text{–}27)$$

where

[A] is a matrix of the coefficients of head, from the left side of equation 2–26, for all active nodes in the grid;

{h} is a vector of head values at the end of time step m for all nodes in the grid; and

{q} is a vector of the constant terms, RHS, for all nodes of the grid.

MODFLOW assembles the vector {q} and the terms that comprise [A] through a series of subroutines. The vector {q} and the terms comprising [A] are then transferred to subroutines that actually solve the matrix equations for the vector {h}.

Types of Model Cells and Simulation of Boundaries

In practice, formulating an equation of the form of equation 2–24 for every cell in a model grid is generally unnecessary, because the status of certain cells is specified in advance to simulate the boundary conditions of the problem. In MODFLOW, cells used to simulate boundary conditions are grouped into two categories—"constant-head" cells and "no-flow" cells. Constant-head cells are those for which the head is specified for each time, and the head value does not change as a result of solving the flow equations. No-flow cells are those for which no flow into or out of the cell is permitted. The remaining cells of the grid, termed "variable-head" cells in this report, are characterized by heads that are unspecified and free to vary with time. An equation of the form of equation 2–24 must be formulated for each variable-head cell in the grid, and the resulting system of equations must be solved simultaneously for each time step in the simulation.

Constant-head and no-flow cells are used in MODFLOW to represent conditions along various hydrologic boundaries inside the grid. For example, figure 2–7 shows the map of an aquifer boundary superimposed onto a grid of cells generated for the model. The aquifer is of irregular shape, whereas the model grid is always rectangular in outline. In areas where the aquifer is coincident with the outside edge of the grid, no special designation is needed to indicate no flow because MODFLOW does not compute inter-cell flow through the outside edges of the grid, including the top and bottom. Within the grid, no-flow cells have been used to delete the part of the grid beyond the aquifer boundary. The figure also shows constant-head cells along one section of the boundary; these may be used, for example, where the aquifer is in direct contact with major surface-water features. Other boundary conditions, such as areas of constant inflow or areas where inflow varies with head, can be simulated through the use of external source terms (Chapter 6) or through a combination of no-flow cells and external source terms. If flow occurs to or

from the outside of the grid where the aquifer extends to edge of the grid, then the flow can be simulated by using external source terms in the cells at the edges of the grid.

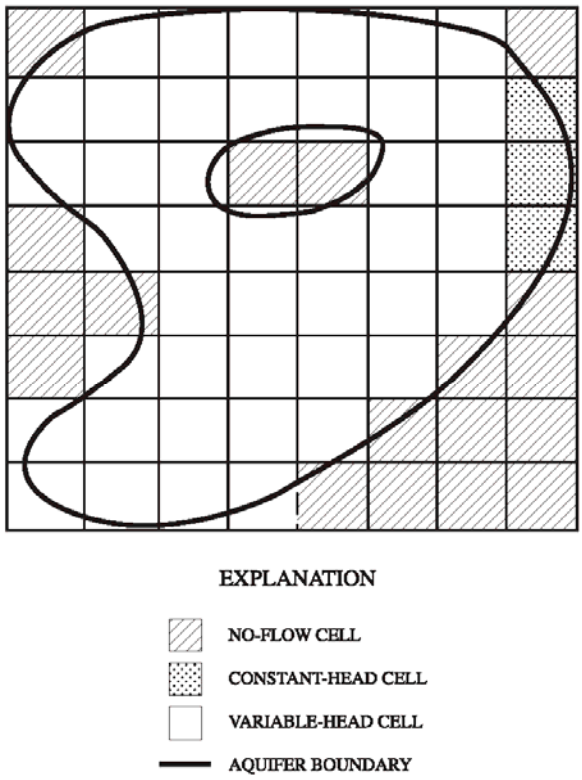


Figure 2–7. Discretized aquifer showing boundaries and cell designations. (Modified from McDonald and Harbaugh, 1988.)

Conceptual Aspects of Vertical Discretization

MODFLOW handles discretization of space in the horizontal direction by reading the number of rows, the number of columns, and the width of each row and column (that is, the width of the cells in the direction transverse to the row or column). Discretization of space in the vertical direction is handled in the model by specifying the number of layers to be used, and by specifying the top and bottom elevations of every cell in each layer.

At one extreme, vertical discretization can be visualized simply as an extension of areal discretization—a more or less arbitrary process of dividing the flow system into segments in the vertical dimension, governed in part by the vertical resolution desired in the results. At the opposite extreme, vertical discretization can be viewed as an effort to represent individual aquifers or permeable zones by individual layers of the model. Figure 2–8A shows a typical geohydrologic sequence that has been discretized in figures 2–8B and 2–8C according to both interpretations. The first viewpoint (fig. 2–8B) leads to rigid superposition of an orthogonal three-dimensional grid on the geohydrologic system; while there may be a general correspondence between geohydrologic layers and model layers, no attempt is made to make the grid conform to stratigraphic irregularities. Under the second viewpoint (fig. 2–8C), model layer thickness is considered variable, to simulate the varying thickness of geohydrologic units; this leads, in effect, to a deformed grid.

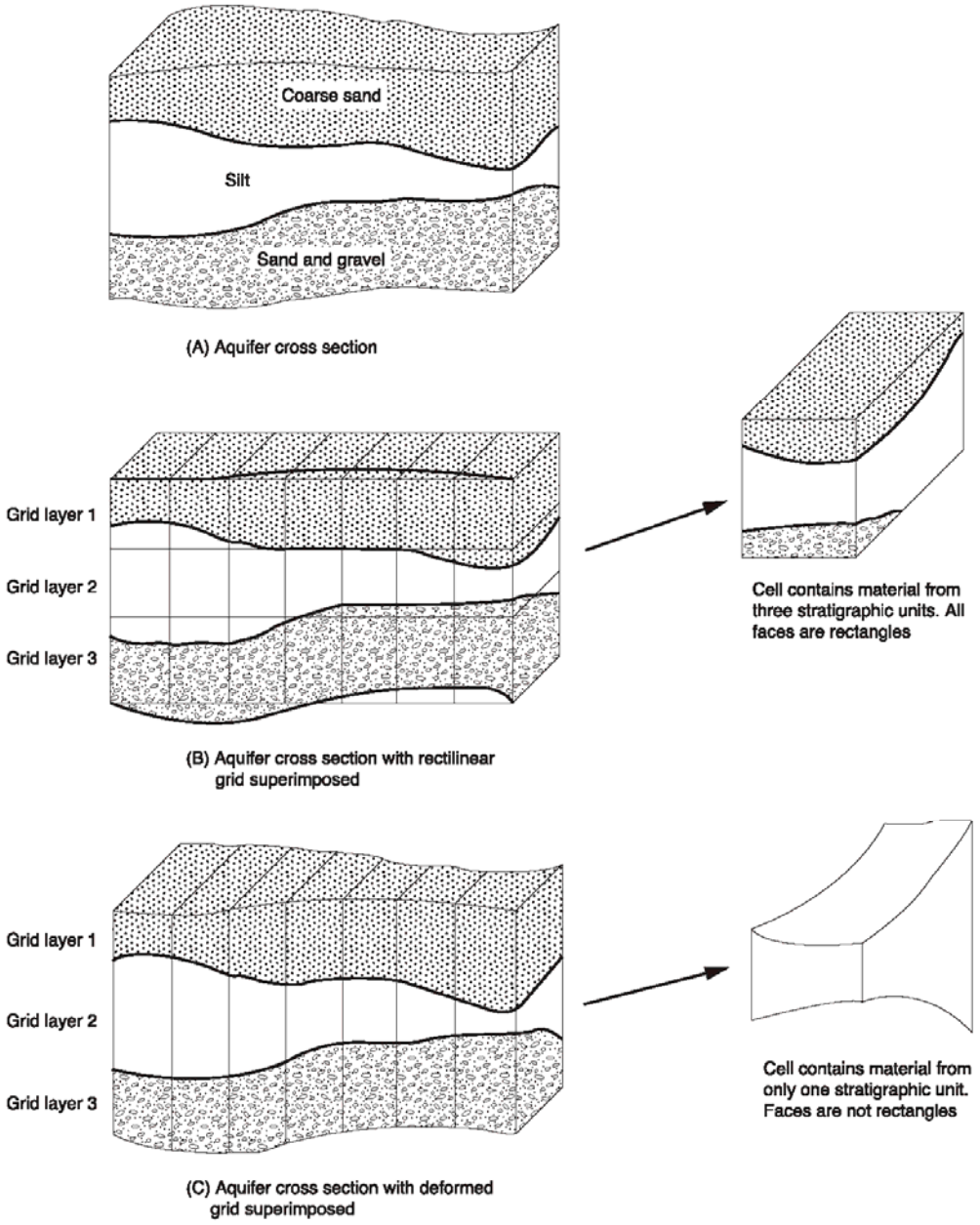


Figure 2–8. Schemes of vertical discretization. (From McDonald and Harbaugh, 1988.)

Each of these methods of viewing the vertical discretization process has advantages, and each presents difficulties. The model equations are based on the assumption that hydraulic properties are uniform within individual cells, or at least that meaningful average or integrated properties can be specified for each cell; these conditions are more likely to be met when model layers conform to geohydrologic units as in figure 2–8C. Moreover, greater accuracy can be expected if model layers correspond to intervals within which vertical head loss is negligible, and this is also more likely under the configuration of 2–8C. On the other hand, the deformed grid of 2–8C fails to conform to many of the assumptions upon which the model equations are based; for example, individual cells no longer have rectangular faces, and the major axes of hydraulic conductivity may not be aligned with the model grid. Some error is always introduced by these departures from assumed conditions.

In practice, many vertical discretization schemes turn out to be a combination of the viewpoints illustrated in figures 2–8B and 2–8C. For example, even where layer boundaries conform to geohydrologic contacts, the use of more than one layer to simulate a single geohydrologic unit may be necessary simply to achieve the required model

accuracy. Figure 2–9 shows a system consisting of two sand units separated by a clay unit; the units are of uniform thickness, and each could be represented by a single layer without deformation of the grid. However, head at each horizontal location would be represented by a single value in each unit; if greater detail in representing the vertical head distribution in the units is required, multiple layers must be used to represent each unit. Similarly, figure 2–10 shows a sand-clay system in which pumpage from the sands is sustained partially by vertical flow of water released from storage in the clay. If the objective of analysis is to determine the pattern of storage release in the clay, several model layers would be required to represent that unit, as shown in the figure. On the other hand, figure 2–11 shows a sand-clay system in which storage release occurs only in the sands, flow in the sand is essentially horizontal, and flow in the clay is essentially vertical. In this case, a single model layer may be used to represent each sand unit, whereas the clay may be represented simply by the vertical conductance between layers. This approach to vertical discretization has been termed the "Quasi Three-Dimensional" approach.

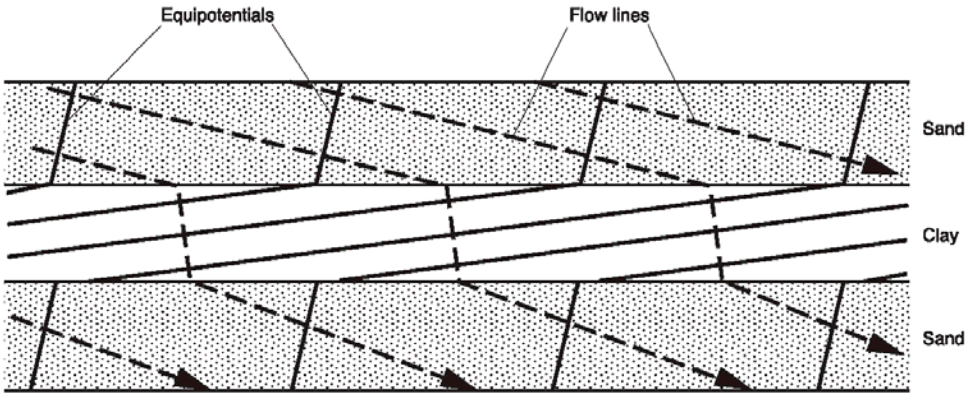


Figure 2–9. Possible pattern of flow in a cross section consisting of two high-conductivity units separated by a low-conductivity unit. (From McDonald and Harbaugh, 1988.)

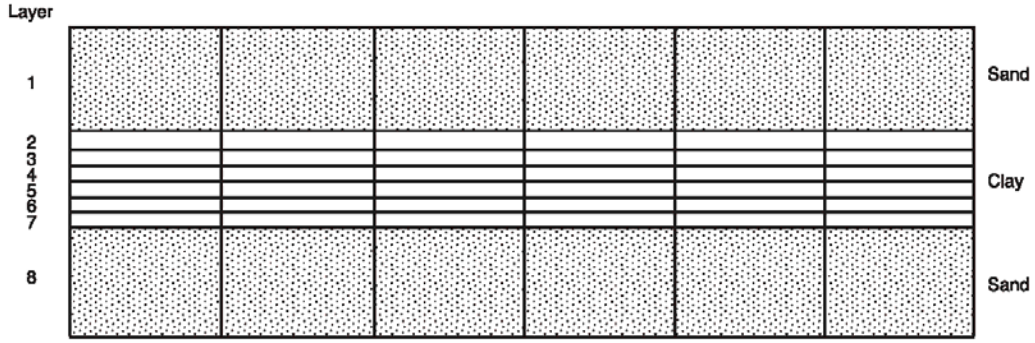


Figure 2–10. A cross section in which a low-conductivity unit is represented by six model layers. (Modified from McDonald and Harbaugh, 1988.)

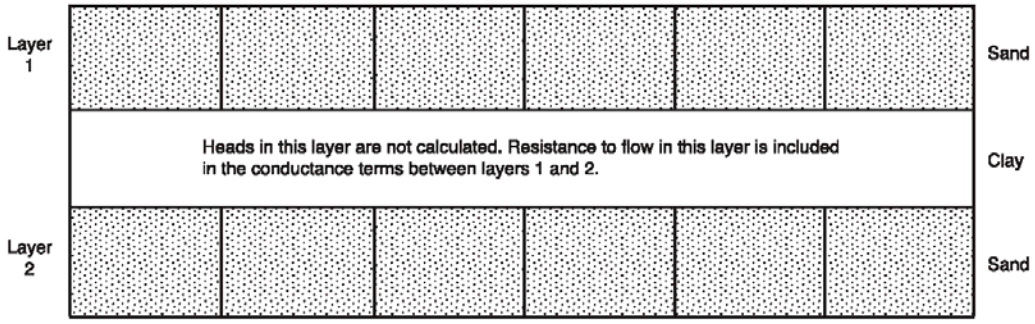


Figure 2–11. A cross section in which a low-conductivity unit is represented by the conductance between model layers. (From McDonald and Harbaugh, 1988.)

The approaches to vertical discretization described above all lead to a set of equations of the form of equation 2–26, which must be solved simultaneously at each time step. The differences among these approaches arise in the way the various conductances and storage terms are formulated and, in general, in the number of equations to be solved, the resolution of the results, and the accuracy of the results. MODFLOW is capable of implementing any of these approaches to vertical discretization in that the elevations of the individual cells in each layer can vary. Chapter 5, which describes the Block-Centered Flow and Layer-Property Flow Packages, contains a discussion of the formulation of conductance and storage terms corresponding to the various ways of conceptualizing the vertical discretization.

CHAPTER 3
Design of the Ground-Water Flow Process

Procedures

To facilitate writing an easily understood program, the Ground-Water Flow (GWF) Process is broken into pieces called “procedures.” Figure 3–1 is a flowchart that shows the order in which procedures are invoked to create a program that solves the ground-water flow equation. The procedures are shown as rectangles in figure 3–1. Other processes can be broken into procedures and combined in the flowchart in a similar manner, but the scope of this report is limited to GWF.

The period of simulation is divided into a series of "stress periods" within which specified stress data 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 2–26 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, the program includes three nested loops: a stress-period loop, within which there is a time-step loop, which in turn contains an iteration loop.

Prior to entering the stress loop, the Allocate and Read (AR) Procedure is executed. The AR Procedure pertains to the simulation as a whole and performs a number of setup functions. The number of cells in the grid for the problem to be simulated is determined as well as the hydrologic options and the solution method. Memory is allocated for all aspects of the simulation. Data that do not vary within stress periods are read. These data include the following: the cell dimensions and time information, boundary conditions, initial heads (starting heads), aquifer hydraulic properties, and control information required by the specified solution scheme. Certain preliminary calculations also are made in this procedure to prepare data for further processing.

Within the stress period loop, the first procedure is termed the Stress (ST) Procedure. This procedure advances to a new stress period. The Read and Prepare (RP) Procedure reads and processes all data that pertain to a stress period, such as pumping rates and areal recharge. The time-step loop is then entered; in the Advance (AD) Procedure, the length of the time step is calculated and the heads for the start of the time step are initialized. The AD Procedure also performs other processing that must be done every time step.

The iteration loop contains the Formulate (FM) Procedure, which determines the conductances and coefficients for each node as required by equation 2–26, and the Approximate Procedure (AP), which approximates a solution to the system of linear equations for head. The FM Procedure is called prior to each solver iteration so that the conductances and coefficients in the flow equation can be changed based on the latest approximate head solution. Iteration proceeds until closure is achieved or until a specified maximum number of allowable iterations are reached.

At the end of the iteration loop, the Output Control (OC) Procedure determines which computed information, such as heads, budget terms, and cell-by-cell flow terms, will be output. In the Water Budget (BD) Procedure, budget entries are calculated and cell-by-cell flow terms are printed or recorded, as explained in a subsequent section. In the Output (OT) Procedure, the specified computed information is printed or recorded.

After all time steps are completed for all stress periods, the Deallocate (DA) Procedure releases allocated memory.

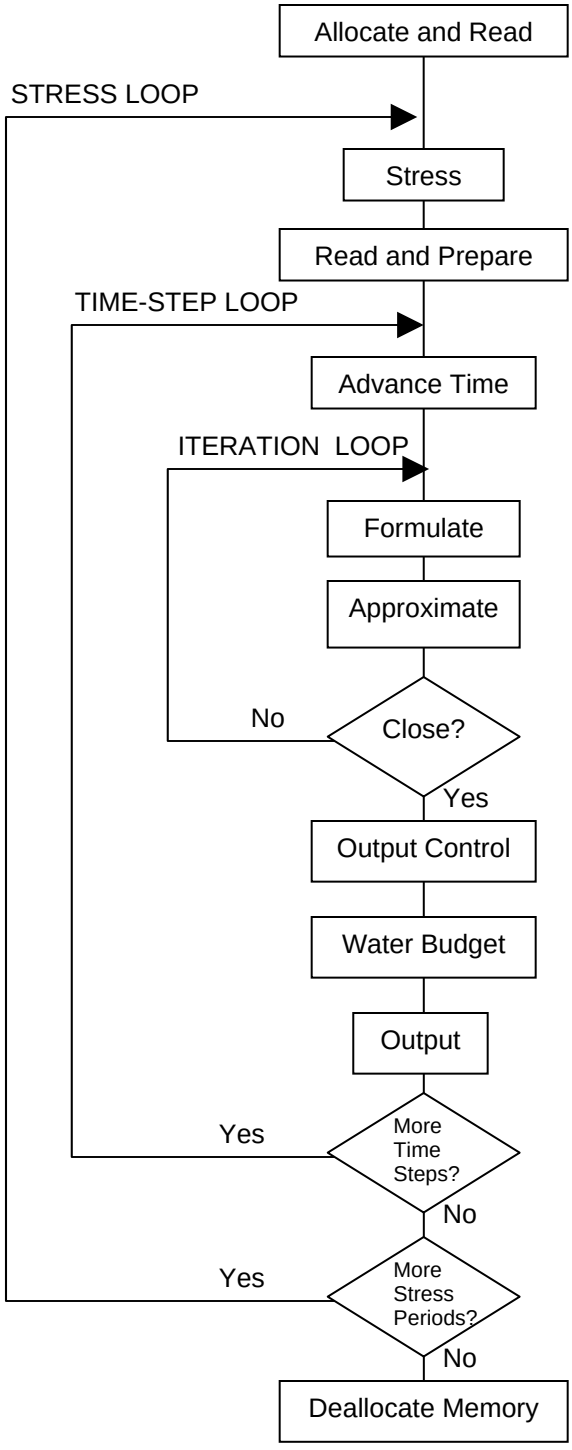


Figure 3–1. Flowchart of program to simulate ground-water flow.

Packages

Although the procedures are fundamental components of the computer program, a hydrologist using the program will likely want to think of the program in terms of its capabilities for solving hydrologic problems. For this purpose, the program is divided into packages. The hydrologist will naturally think about the way the ground-water flow equation is constructed. Accordingly, the various parts of the code that deal with defining the ground-water flow equation are divided into packages that are called hydrologic packages. There are two types of hydrologic packages. The first type is the internal flow package, which simulates flow between adjacent cells. The second type is the stress package, which simulates an individual kind of stress (such as rivers, wells, and recharge). Although hydrologists would generally prefer not to have to deal with how the simultaneous equations that result from the finite-difference approximation are solved, the reality is that equation solution is not straightforward and must be dealt with when using a ground-water model. MODFLOW incorporates multiple solution methods, and each one is termed a solver package. Logically, the hydrologic and solver packages represent the entire work of the program; however, an additional component is needed to control the program. This is called the Basic Package.

Table 1 lists the various packages of GWF that are documented in this publication, the three-character abbreviation used for each package, and the package category. Table 1 does not show anything about the programming, but shows the capabilities of the program. The hydrologic packages calculate the coefficients of the finite-difference equation for each cell. The Block-Centered Flow (BCF) and Layer-Property Flow (LPF) Packages are both internal flow packages that provide alternate approaches to formulate the internal flow terms. The Horizontal Flow Barrier (HFB) Package is a supplementary internal flow package that works with BCF and LPF to modify conductances to simulate a barrier between nodes. 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 river. The solver packages are those that implement algorithms for solution of the systems of finite-difference equations. This documentation describes three solver packages, one incorporating the Strongly Implicit Procedure of solution, another utilizing Preconditioned Conjugate Gradient method, and one using a direct solution method. The only package that does not fit into the hydrologic or solver categories is the Basic Package, which addresses a variety of tasks in support of the entire process.

Table 3–1. List of Packages for simulating ground-water flow.

Package Name	Abbreviation	Package Category
Basic	BAS	Program Control
Block-Centered Flow	BCF	Hydrologic/Internal
Layer-Property Flow	LPF	Hydrologic/Internal
Horizontal Flow Barrier	HFB	Hydrologic/Internal
Well	WEL	Hydrologic/Stress
Recharge	RCH	Hydrologic/Stress
River	RIV	Hydrologic/Stress
General-Head Boundary	GHB	Hydrologic/Stress
Drain	DRN	Hydrologic/Stress
Evapotranspiration	EVT	Hydrologic/Stress
Strongly Implicit Procedure	SIP	Solver
Preconditioned Conjugate Gradient	PCG	Solver
Direct Solution	DE4	Solver

Primary Subroutines

As described earlier in this chapter, figure 3–1 provides a flow chart for the overall structure of MODFLOW for solving the ground-water flow equation. The flow chart indicates the sequence in which procedures must be invoked to build and solve the ground-water flow equation. To construct a program following this flow chart, each procedure could be implemented as a single subroutine. From a programming perspective, this would be the most direct way to construct the program; however, the subroutines would be quite large, and there is benefit from further subdividing the work into smaller subroutines.

There are many ways the code could be subdivided. The approach used for MODFLOW is to divide the code into pieces that can be combined into both procedures and packages as desired. Accordingly, a primary subroutine is defined as the code within a procedure for one package. A procedure can be constructed by grouping all the primary subroutines that comprise the procedure. A package can be constructed by grouping all the primary subroutines that comprise the package.

Using this approach, the MAIN Program is simply an organized sequence of call statements to the primary subroutines. The MAIN Program is organized according to procedures in the flow chart (fig. 3–1). Each procedure is invoked through calls to multiple primary subroutines — one for each package represented by the procedure. Most of the packages are optional depending on the problem being simulated. Accordingly, the calls to the subroutines of optional packages are invoked through "IF" tests that determine whether a primary subroutine is required. Thus, the MAIN Program does not itself do the work of simulation, but merely calls the various primary subroutines in the proper sequence to do that work.

The classification of GWF Process primary subroutines by procedure and by package is illustrated in figure 3–2. The horizontal rows in figure 3–2 correspond to procedures, and the vertical columns correspond to packages. An "X" is entered in each block of the matrix for which a primary subroutine exists; absence of an "X" indicates that the procedure in question is not required in the indicated package.

	BAS	BCF	LPF	HFB	WEL	RCH	RIV	GHB	DRN	EVT	SIP	PCG	DE4
Allocate and Read (AR)	X	X	X	X	X	X	X	X	X	X	X	X	X
Stress (ST)	X												
Read and Prepare (RP)					X	X	X	X	X	X			
Advance (AD)	X												
Formulate (FM)	X	X	X	X	X	X	X	X	X	X			
Approximate (AP)											X	X	X
Output Control (OC)	X												
Water Budget (BD)		X	X		X	X	X	X	X	X			
Output (OT)	X												
Deallocate (DA)	X	X	X	X	X	X	X	X	X	X	X	X	X

Figure 3–2. GWF Process primary subroutines classified by procedure and package.

The primary subroutines are named according to a convention that indicates the process, package, and the procedure to which they belong. The first three characters designate the process. The fourth character is a process version number. The next three characters designate the package, the following character is a package version number, and the last two indicate the procedure. For example, in figure 3–2, a GWF primary subroutine is indicated

for the Well Package and Allocate and Read Procedure. This subroutine is designated as GWF2WEL7AR—the first three letters, GWF, indicate the Ground-Water Flow Process; the fifth through seventh letters, WEL, indicate that the subroutine is part of the Well Package; the last two letters, AR, indicate that it performs the Allocate and Read Procedure in that package. Thus, this subroutine is one that deals with the simulation of specified withdrawal or injection, as through wells, and its particular functions are to allocate the space in computer memory used to store well data and to read well data that are constant during the simulation. The number 2 in the fourth place of the name is a process version number. In this report, the GWF Process is designated as version 2. The number 7 appearing in the eighth place of the subroutine name 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 other versions. McDonald and Harbaugh (1988) designated all packages as version 1. Many model users have made modifications to various packages, so there are model packages with versions greater than 1. In this report, all packages are designated as version 7 in order to minimize the chance of duplicating a version number used by others.

When primary subroutines become so large that they are difficult to understand, they are broken into smaller secondary subroutines. All of these subroutines start with the letter “S.” Utility subroutines are further used to keep the program as easy to understand as possible. Utility subroutines implement functionality that is required by many packages. When a secondary or utility subroutine is called, it is logically viewed as being part of the calling subroutine. Specific secondary and utility subroutines are documented later in this report.

In summary, the program is broken into primary subroutines. These primary subroutines can be grouped according to the procedures indicated in figure 3–1 in order to construct a computer program. Primary subroutines can also be grouped by "packages," where a package (for example, the River Package, the Well Package, or the Strongly Implicit Procedure Package) includes those subroutines required to incorporate a particular hydrologic process or solution algorithm into the simulation. In terms of understanding the operation of the model, both of these two methods of grouping subroutines are useful. The package classification, for example, indicates which subroutines will be active in a given simulation. Subroutines are called by the MAIN Program only if they are part of a package that is required in the simulation. On the other hand, the procedure classification defines the specific function of a subroutine in relation to the structure of the computer program. For example, several subroutines whose function is to allocate space and read data are grouped under the Allocate and Read Procedure; each of these subroutines allocates the space required for use in a single package. If few options or features are specified, relatively few packages are involved in the simulation, and the Allocate and Read Procedure is handled by a relatively small number of subroutines. As the options specified by the user increase, more packages enter the simulation, and more subroutines are called to complete the space allocation task.

Computing Flow Equation Terms

For cells that are designated as always being confined, the conductance terms for cell-to-cell flow (CC, CR, and CV of equation 2–26) are computed at the beginning of the simulation in the AR subroutine of the internal flow package. These terms will remain constant throughout the simulation for confined cells. For cells that are (or may become) unconfined, the conductance terms are computed in the FM Procedure at each solution iteration because the conductances depend upon saturated thickness, which may change at each iteration. The various conductance terms are ultimately passed to a solver package, where the matrix equations (eq. 2–27) are solved.

The coefficient $HCOF_{i,j,k}$ and the term $RHS_{i,j,k}$ of equation 2–26 are formulated anew at each iteration for all active nodes in the grid. This formulation is done progressively as the FM subroutine for each package calculates and adds terms for the particular hydrologic 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 grid. The internal flow package then adds the term

$$\frac{-SS_{i,j,k}\,\Delta r_j\,\Delta c_i\,\Delta v_k}{t - t^{m-1}}$$

to $HCOF_{i,j,k}$ at each node, and adds the term

$$-SS_{i,j,k}\,\Delta r_j\,\Delta c_i\,\Delta v_k\,\frac{h_{i,j,k}^{m-1}}{t - t^{m-1}}$$

to $RHS_{i,j,k}$ at each node. Packages that specify external inflow (negative value indicates outflow) using expressions of the form $P_{i,j,k}h_{i,j,k}+Q_{i,j,k}$ add the term $P_{i,j,k}$ to $HCOF_{i,j,k}$ and the term $-Q_{i,j,k}h_s$ to $RHS_{i,j,k}$ for the affected cells. This process continues until each package specified by the user has added its contribution to HCOF and RHS at each indicated cell of the grid. The HCOF and RHS values are then transferred to the solver package, together with the conductances (CC, CR, and CV) and the heads at the beginning of the time step. The solver package sums the six conductance terms and the value of HCOF for each cell to create a single coefficient of $h_{i,j,k}$ (corresponding to the term in parentheses in equation 2–26), and carries out one iteration of the solution procedure.

Adding and Modifying Packages

The structure of GWF has been designed in such a way that the packages are as independent as possible. This facilitates making modifications of all types. New packages can be added and the subroutines of an existing package can be modified without affecting other packages. The method of incorporating stresses through the HCOF and RHS terms allows new stress packages to be used along with the original stress packages. Development of a new package involves creating the primary subroutines for each of the procedures involved and modifying the MAIN Program to call those subroutines in proper sequence. Two general constraints limit what can be done with new packages. First, only one internal flow package that formulates the internal flow terms (the BCF and LPF Packages in this report), and second, only one solver package can be in use in a single simulation. For example, a new internal flow package utilizing a point centered approach could be added, but using the new package at the same time as the BCF or LPF Package would not be possible (or make sense). Likewise, new solvers can be added as long as only one is used at a time.

Steady-State Simulations

The program structure for simulating ground-water flow illustrated in figure 3–1 was described assuming that a transient problem is being solved; however, the same structure also works for steady-state simulations. A steady-state simulation is represented by a single stress period having a single time step with the storage term set to zero. Thus a single set of simultaneous equations will be solved. All procedures within the program structure are required much as they are for a transient problem. Setting the number and length of stress periods and time steps is the responsibility of the Basic Package. The length of the stress period and time step will not affect the head solution because the time derivative is not calculated in a steady-state problem. Setting the storage term to zero is the responsibility of the internal flow package. Most other packages need not "know" that a simulation is steady state.

Simulations also can be mixed transient and steady state because each stress period can be designated transient or steady state. Thus, a simulation can start with a steady-state stress period and continue with one or more transient stress periods.

Units of Length and Time

The GWF Process formulates the ground-water flow equation without using prescribed length and time units. Any consistent units of length and time can be used when specifying the input data for a simulation. This capability gives a certain amount of freedom to the user, but care must be exercised to avoid mixing units. The program cannot detect the use of inconsistent units. For example, if hydraulic conductivity is entered in units of feet per day and pumpage as cubic meters per second, the program will run, but the results will be meaningless. Other processes

generally are expected to work with consistent length and time units; however, other processes could conceivably place restrictions on which units are supported.

The user can set flags that specify the length and time units (see the input instructions for the Discretization File), which may be useful in various parts of MODFLOW. For example, the Basic Package will label the table of simulation time with time units if the time units are specified by the time-units flag (ITMUNI). If the time units are not specified, the program still runs, but the table of simulation time does not indicate the time units. A length-unit flag (LENUNI) is also included in MODFLOW. Situations in other processes may require that the length or time units be specified. In such situations, the input instructions will state the requirements. Remember that specifying the unit flags does not enforce consistent use of units. The user must insure that consistent units are used in all input data.

Model Grid and Aquifer Boundaries

As noted in Chapter 2, the model may be visualized in terms of a three-dimensional assemblage of cells, each cell containing a point called a node at which head is to be calculated. The size of the model grid 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 grid 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 grid. 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 grid is bounded externally by planes across which no cell-to-cell flow occurs. If these boundaries of the model grid, 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 specification 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 grid as no-flow or constant-head cells, 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. While no cell-to-cell conductance terms are formulated for the interval above the uppermost layer of the model grid, flow into this layer from above is commonly represented in the model through external stress terms—for example, terms representing recharge or stream seepage.

As pointed out above, the model grid as initially generated always has the form of a rectangular box. Where the limits of an aquifer do not coincide with this rectangular shape, no-flow cells may be used to designate parts of the grid that 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 that are in full contact with the aquifer. Boundaries that 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 that 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 riverbed conductance and river stage (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 relations.

A finite-difference equation of the form of equation 2–26 is formulated for each variable-head cell in the grid. 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 no-flow cells, no equation is formulated, and no term appears in the equation of any adjacent cell for flow to or from the no-flow cell; thus no flow is simulated across the interval between a no-flow cell and any adjacent cell.

Constant-head cells, no-flow cells, and variable-head cells are distinguished from one another in the model through the IBOUND variable, which contains one value for each cell in the grid. The entry in IBOUND for a given cell indicates the type of cell according to the following convention:

If $IBOUND_{i,j,k} < 0$, then cell i,j,k is constant head.

If $IBOUND_{i,j,k} = 0$, then cell i,j,k is no flow.

If $IBOUND_{i,j,k} > 0$, then 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 that are specified as active but are given hydraulic conductivity values equal to zero are changed to no-flow cells by the program.

Model Input and Output

MODFLOW is designed to permit model input to be gathered, as it is needed, from many different files. Likewise, results from the model calculations can be written to a number of files. The Listing File is a key file to which model output is written. As MODFLOW runs, information is written to the Listing File, including much of the input data (as a record of the simulation) and calculated results. An overview of input and output is provided here. Details about the files used by each package are provided in the chapters documenting the packages. Also, the next chapter provides details about the ways data are read from files.

MODFLOW is further designed to allow the user to control the amount, type, and frequency of information to be output. Much of the output will be written to the Listing File, but some model output can be written to other files. The Listing File is designed to be printed; however, the Listing File may not need to be printed if tools (such as a word processor) are available to examine it using the computer display. The Listing File includes a summary of the input data read for all packages. In addition, the Listing File optionally contains calculated head and drawdown controlled by layer and time step, and the overall volumetric budget controlled by time step. The Listing File also contains information about solver convergence and error messages. Output to other files can include head, drawdown, and cell-by-cell flow terms for use in calculations external to the model or in user-supplied applications such as plotting programs.

The Basic Package reads a file called the Name File, which specifies most of the files that will be used in a simulation. Several files, including the Listing File, are always required whereas other files are optional depending on the simulation. Output Control, which is a major option contained within the Basic Package (and a procedure in the overall flow chart), receives instructions from the user to control the amount and frequency of output. Details about Output Control and the Name File are provided in Chapter 4 (Basic Package).

Within the program, input and output are based on an element of the Fortran language called the file unit number, which identifies the file from which the input is to be read or to which the output is to be written. A connection must be provided between the name of each input or output file and the corresponding unit number. For files included in the Name File, this link is made by directly specifying the file unit number along with the file name.

Volumetric Budget

A summary of all inflows (sources) and outflows (sinks) of water to a region is generally called a water budget. In this report, the water budget is termed a volumetric budget because volumes of water and volumetric flow rates are involved; thus strictly speaking, a volumetric budget is not a mass balance, although this term has been used 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 to provide a summary of the sources and sinks of water to 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.

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 or from 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. Each hydrologic package calculates its own contribution to the budget.

For every time step, the budget subroutine of each hydrologic 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 a program variable (VBVL). Most packages deal with only one such component of flow, but the internal flow packages (BCF and LPF) deal 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 VBVL.

The BAS Package uses the inflows, outflows, and cumulative volumes in VBVL to write the budget to the Listing File at the times requested by the model user. When a budget is written, the flow rates for the last time step and cumulative volumes from the beginning of simulation are written for each component of flow. Inflows are written separately from ouflows. Following the convention indicated above, water entering storage is treated as an outflow (that is, as a loss of water from the flow system) while water released from storage is treated as an inflow (that is, a source of water to the flow system). In addition, total inflow and total outflow are written, as well as the difference between total inflow and outflow. The difference is then written as a percentage error, calculated using the formula:

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

where

D is the percentage error term,

IN is the total inflow to the system, and

OUT is the total outflow.

If the model equations are solved correctly, the percentage 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 output. The budget is written to the Listing File at the end of each stress period whether requested or not.

In some situations, calculating flow terms for various subregions of the model is useful. To facilitate such calculations, provision has been made to save flow terms for individual cells in a separate file 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 caused by 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 accumulation 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 flow terms are discussed further 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 indicates the time steps for which cell-by-cell terms are to be saved. In addition, each hydrologic package includes a flag that 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 saving of cell-by-cell flow is requested through the Output Control Option. Only flow values are saved in

the cell-by-cell 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 budget utility subroutines in Chapter 9 for information on how the data are written to a file.

Three of the four types of cell-by-cell flow terms listed above—storage, constant-head cell, and internal flows—are computed in the internal flow package, and thus fall under the control of a single flag in the input to that package. Thus, all three types are saved in a file if this flag is set, and if the saving of cell-by-cell flow also is requested in Output Control for the time step.

The cell-by-cell storage term gives the net flow to or from storage in a variable-head cell. The net storage for each cell in the grid 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 that 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 part of the grid); 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 internal flow package for each cell in the grid 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.

Cell-by-cell stress flows are flow rates into or out of the model, at a particular cell, owing 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 out of the cell.

Three-Dimensional Model Data

Many terms in the equations presented in this report specify cell location using row, column, and layer indices in that order (usually designated as i,j,k), as is customary in scientific literature. Such terms are stored in the program as three-dimensional variables (called arrays in Fortran). Fortran incorporates the capability to use subscripts to access values (called elements) in variables, and the subscripts are indicated in parentheses after a variable name. The order of subscripts in the Fortran language determines the order of storage in computer memory, which can affect program efficiency. The design of the program is such that the subscripts should be in column, row, and layer order for the best efficiency on most computers; therefore, this order has been used throughout the program. Bear in mind this difference in the subscript ordering when comparing the model program to the equations presented in this report. Typically in the program, J is used for the column subscript, I is used for row, and K is used for layer, so the order of indices is J,I,K.

In many cases, the variable name in the program is the same as the term name used in the equations. For example, there are CR, CC, CV, HCOF, and RHS variables that store the corresponding terms. There are some variables in the model program, however, that have different names than the terms used in equations. For example, h is not used as a variable name primarily because head is not stored for every time step. Rather, two values of head are stored: the head at the end of the new time step (h^m) and the head at the end of the prior time step (h^{m-1}). These

are named HNEW and HOLD, respectively, and these names serve to indicate that these values change as the time step changes. The program moves HNEW to HOLD at the start of each time step. Another head variable in the program is STRT, which is the initial head specified by the user at the start of the simulation (h^0). The documentation for each package specifies any differences between the names of terms in equations and the variable names used in the program.

Memory Allocation and Deallocation

Most variables that are shared among subroutines are declared in Fortran modules. These variables can be accessed in subroutines through Fortran USE statements. The memory required for variables that are referenced using subscripts (arrays) is obtained using Fortran ALLOCATE statements. DEALLOCATE statements are used at the end of the model to release the allocated memory.

The MAIN Program

The MAIN Program controls the order in which the primary subroutines are executed. This occurs with CALL statements that specify, by name, the subroutines to be executed. The arrangement of CALL statements in the MAIN Program reflects the order of procedures shown in the system flow chart (fig. 3–1). The subroutines of a package are called only if the package is being used in a simulation as specified in the Name File as described previously. Details about the MAIN Program are provided in Chapter 9.

CHAPTER 4
BASIC PACKAGE

The Basic (BAS) Package of the GWF Process handles a number of administrative tasks for MODFLOW. It reads the Name File to open files and determine options that will be active. The BAS Package declares and allocates memory for variables, reads the IBOUND variable, reads initial head and tracks head throughout time, reads data defining the discretization of space and time, calculates an overall water budget, and controls model output according to user specifications. The BAS Package also reads zone and multiplier arrays that can be used by other packages to define parameters and reads the optional Parameter Value File. Much of the data declared by the GWF BAS Package is used by other GWF Process packages and other processes.

Opening Files and Activating Options Using the Name File

The BAS Package reads the Name File. Each line in the Name File specifies a file name, type, and unit number. The file unit number is used by Fortran to refer to the file. The file type specifies the purpose of a file. Many of the file types correspond to major options; a major program option is a major segment of the program that is utilized only at the user's request. Specifying a file having the corresponding file type causes the option to be activated. Likewise, an option is inactive if the Name File does not include a file having the option's file type. Most packages are themselves major options, so each package generally has at least one input file that is specified in the Name File. For example, if areal recharge is simulated, a file having a file type of RCH will be required. The Input Instructions section defines all of the file types that can be used. The Name File must include two files required by the Basic Package, the BAS Package file and the Discretization File. In addition, the Name File can include files for several options included in the Basic Package: Time-Variant Constant Head, Output Control, Zone arrays, Multiplier arrays. These options are described throughout this chapter.

File types DATA and DATA(BINARY) define input or output files whose file unit numbers are referenced by other files. The DATA and DATA(BINARY) file types can be specified for many files. An example is the input of variables containing multiple values that are accessed by grid indices. The data can be read from the appropriate package file, or the package file can indicate that the data should be read from a different file unit number. If a different file unit number is indicated, the file connected to that file unit number must be specified in the Name File. For example, the RCH Package reads the recharge rate for every cell in a model layer when recharge is being simulated. These recharge values can be defined directly in the RCH file (that is, the file whose file type is RCH), or the RCH file can indicate that the recharge values should be read from a different file. Another example is the output of computed head, which can be written to a file unit number. The file connected to that file unit number must be specified in the Name File.

The user enters the name of the Name File either through a command line argument or an interactive prompt. Once the name of the Name File is specified, the program runs to completion without any further user intervention. If all steps in the simulation work successfully, the required input files will be opened, the input data will be read from the input files, and results will be written to output files.

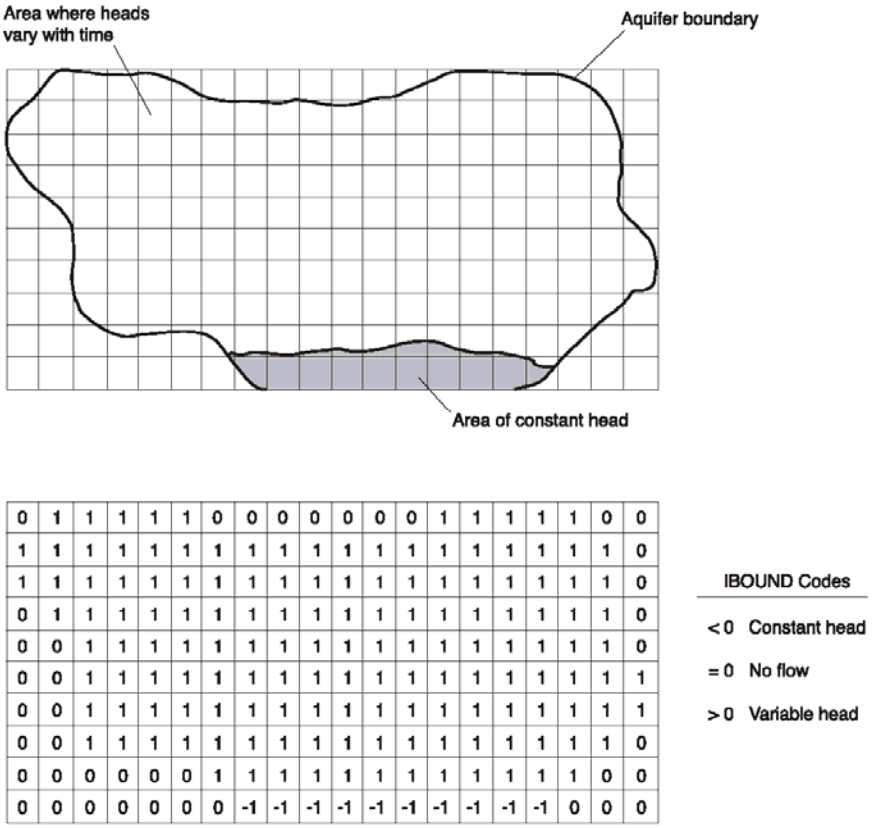
Global Data

The BAS Package declares variables, called global data, that can be shared throughout the GWF Process and other processes and allocates the memory required for the data. A user need not know the names of most of the program variables, but names of variables are essential information for those wanting to understand or modify the program. Chapter 9, Programmer Documentation, defines all of the global data.

The IBOUND Variable

Recall that in Chapter 2 the finite-difference equation (eq. 2–26) is written for each variable-head cell in the grid. The IBOUND variable contains a code for each cell that indicates whether (1) the head varies with time

(variable-head cell), (2) the head is constant (constant-head cell), or (3) no flow takes place within the cell (no-flow or inactive cell). Values of IBOUND are read from the BAS Package file. IBOUND can be modified by other packages if the state of a cell changes. Distribution of the IBOUND code entries for a typical model layer is shown in figure 4–1.



0	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1	1	1	0	0
1	1	1	1	1	1	1	1	1	1	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	1	1	1	1	0
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0
0	0	0	0	0	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	0

Figure 4–1. Example of the IBOUND variable for a single layer. (Modified from McDonald and Harbaugh, 1988.)

Constant-head cells also can be designated using the Time-Variant Specified-Head (CHD) Option of the BAS Package. This capability was originally developed by Leake and Prudic (1991) and called a package; however, the capability is viewed here as an option within the BAS Package because of its function to set values of IBOUND. When the CHD Option is used by including a file with file type CHD in the Name File, constant-head cells are read from the CHD file as a list of cells. For each cell in the list, the CHD Option sets IBOUND equal to negative one. The CHD Option is invoked after IBOUND is read from the BAS Package file, so CHD can be used to designate constant-head cells without specifying a negative value for IBOUND in the BAS Package file.

In addition to designating cells as constant head, the CHD Option specifies the head at the designated constant-head cells. As explained in the following section, the head at cells that are designated as constant head when IBOUND is read is set equal to the initial head. For cells designated as constant head using the CHD Option, the head is specified in the CHD file. If a cell that is designated as constant head in the CHD Option was already designated as constant head when IBOUND was read, the value for head specified by the CHD Option is used rather than the initial head. CHD reads data every stress period, which makes changing the head at constant-head cells throughout a simulation possible.

Initial Head and Tracking Head Throughout Time

Because equation 2–26 is in backward-difference form, a known head distribution is required at the beginning of each time step of a transient simulation in order to calculate the head distribution at the end of that time step. For each time step after the first, the head distribution at the beginning of one time step is set equal to the head

distribution at the end of the previous time step. Heads at the beginning of the first time step are called initial heads, and initial heads are read from the BAS Package input file.

Ideally, the initial heads are defined from a previous steady-state simulation. This insures that the initial heads are accurate at all locations. For example, a model might be developed to simulate the effect of a well field on a previously steady-state system. If the steady-state system without the well is simulated first, then the simulated steady-state heads can be used as initial heads for the transient simulation. Of course, errors in the steady-state heads can occur as a result of errors or misrepresentations in model data, but the steady-state heads will be exact for the data being used. Without simulated steady-state heads, initial heads have to be estimated for all model cells based upon measured heads at observation points. Undoubtedly some error in these interpolated heads may exist, which would impact the calculated heads in the transient model. Note, however, that the errors in simulated head caused by errors in initial heads will decrease with time depending on the storage and transmissive factors of the porous medium being simulated. Thus, the transient model possibly can be started far enough in advance of the time for which results are required so that any errors in head resulting from errors in initial heads will have dissipated.

Even for steady-state simulations, the initial heads must be specified in the BAS Package input file because initial heads become the initial estimate of heads for the interative solvers. In most situations, the initial heads do not affect the computed steady-state heads but may affect how many iterations are required. That is, the closer the initial head is to the final head, the fewer iterations will be needed; however, the impact on the number of iterations is not large unless the initial head is far from the final head. Thus, making a detailed estimate of the initial head for steady-state simulations is generally not warranted.

A water-table cell that is not given the option to convert from no flow to variable head is an example of a situation in which initial head can affect the computed steady-state head. (The option to allow cells to convert from no flow to variable head is discussed in Chapter 5.) If the initial head is below the bottom of the aquifer at such a cell, then the cell will convert to no flow at the start of the simulation and must stay no flow throughout the simulation; yet if the initial head is above the bottom of the aquifer, that cell will begin as a variable-head cell and may remain saturated depending on conditions in the simulation. See Chapter 5 for a further discussion of the effect of initial head in water-table cells.

At cells that are specified as constant head when IBOUND is read from the BAS Package file (IBOUND<0), the initial head becomes the constant head. The head at these cells remains the same throughout the simulation unless the head is changed using the Time-Variant Specified-Head (CHD) Option.

Discretization of Space

As described in Chapter 2, space is discretized by a grid of cells. The finite-difference grid in MODFLOW is assumed to be rectangular horizontally, while the grid can be distorted vertically (fig. 4–2). The Basic Package reads spatial discretization information from a file called the Discretization File, which is required for all model simulations.

The number of rows and columns are specified in variables NROW and NCOL, respectively. The horizontal size of cells is defined by variables DELR and DELC, which represent Δr and Δc, respectively, in the equations of Chapter 2. DELR has one value for every column in the grid, and the value of $DELR_J$ is the width of column J (fig. 4–2A). This width of a specified column is the same for all rows. Similarly $DELC_I$ is the width of row I. The width of a row is the same for all columns.

Some users of previous versions of MODFLOW have reported confusion over DELR and DELC. They interpret DELR to represent distance “between” rows rather than distance “along” rows. DELR is the cell size in the X direction if the grid for a layer is imposed on a Cartesian coordinate system. Likewise, DELC is the cell size in the Y direction.

The number of layers in the grid is specified in variable NLAY. Layers are numbered starting from the top layer and going down (fig. 4–2B). Cell elevations are specified in variable BOTM. Elevation of the top of layer 1 is specified in addition to the bottom elevation of every layer. Below each layer except the bottom layer, a confining bed also can be included through which only vertical flow is simulated. Simulating confining beds by this method often is called the Quasi-Three-Dimensional (Quasi–3D) approach (McDonald and Harbaugh, 1988, p. 5–18). For these confining beds, the elevation of the bottom of the bed is specified also. Use of the Quasi–3D approach is not required; that is, any confining bed can be simulated using one or more distinct model layers as desired. Chapter 5 explains how the Quasi–3D approach is implemented.

The elevation information is used to calculate the thickness of all cells and confining beds below cells. As described, the user must specify the top elevation of layer 1 and the bottom elevations of all model layers and confining beds. The top elevation of other model layers and confining beds need not be separately specified because the top elevation is the same as the bottom elevation of the layer or confining bed that is directly above. Use of distorted layers, as shown in figure 4–2B, is not required. A flat layer is indicated by simply specifying the same value for the bottom elevation of all the cells in the layer.

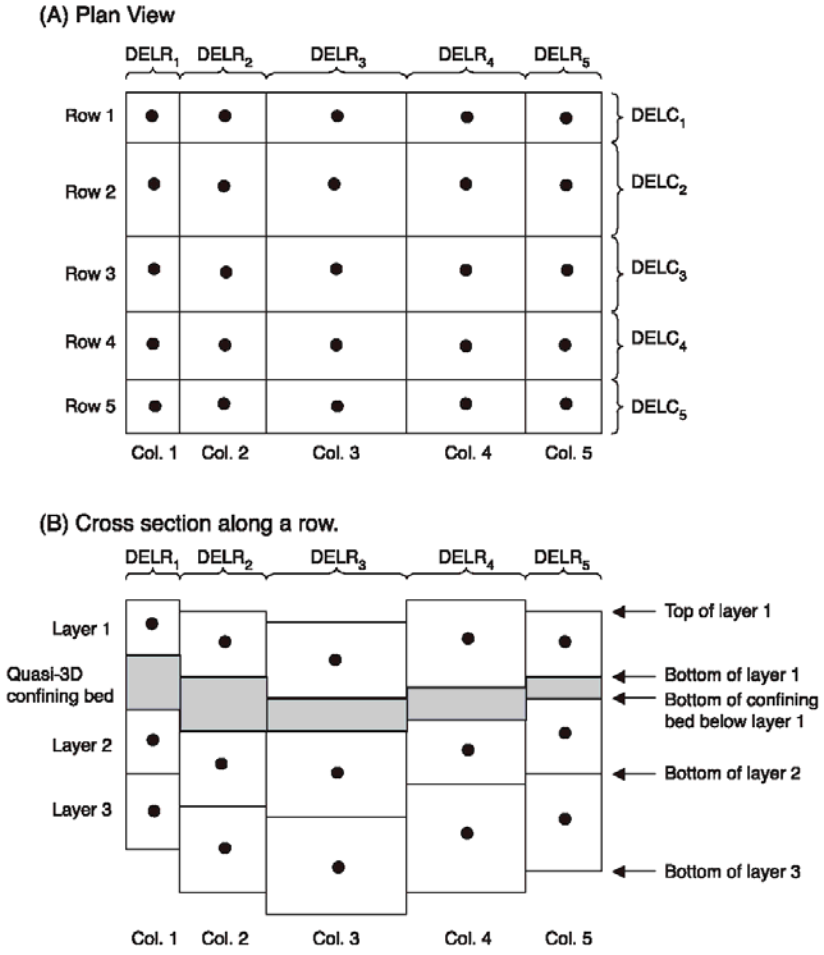


Figure 4–2. Finite-difference grid in (A) plan view and (B) cross-section view.

Discretization of Time

Simulation time is divided into stress periods—time intervals during which the input data for all external stresses are constant—which are, in turn, divided into time steps (fig. 4–3). Note that time steps are fundamental to the finite-difference method whereas stress periods have been incorporated in MODFLOW as a convenience for user input. Discretization information for time is read from the Discretization File.

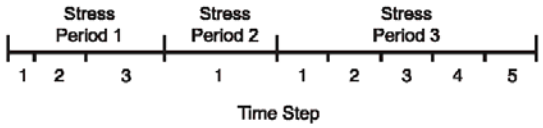


Figure 4–3. Division of simulation time into stress periods and time steps.

Within each stress period, the time steps form a geometric progression. The user specifies the length of the stress period (PERLEN), the number of time steps into which the stress period is to be divided (NSTP), and the time step multiplier (TSMULT). The time step multiplier is the ratio of the length of each time step to that of the

preceding time step. Using these values, the program calculates the length of the first time step (Δt^1) in the stress period as:

$$\Delta t^1 = \text{PERLEN}\left(\frac{\text{TSMULT}-1}{\text{TSMULT}^{\text{NSTP}}-1}\right) \quad \text{when TSMULT}\neq 1, \text{ and}$$

$$\Delta t^1 = \frac{\text{PERLEN}}{\text{NSTP}} \quad \text{when TSMULT is one.}$$

The length of each successive time step is computed as:

$$\Delta t^{m+1} = \Delta t^m \text{TSMULT} .$$

As stated previously, stress periods are implemented only as a convenience. Packages that define time-dependent stresses read input data every stress period. Stress periods facilitate the frequent need to have constant input data for stresses for multiple time steps. Situations are not unusual, however, in which a need arises to change stress data for every time step. In this situation, each stress period must consist of a single time step.

As an example of how stress periods are used, consider a simulation that uses the River and Well Packages. The simulation is for a period of 90 days, and 90 one-day time steps are used. The well and river cells are the same throughout the simulation, but the pumping rates and river stage vary. The number of stress periods depends on how frequently the river stage and pumping rates vary because a new stress period must start whenever stage or pumping for any cell changes. For example, if river stage or pumping only change every 30 days, then three 30-day stress periods can be used. Likewise, if pumping or river stage varies every 3 days, then 30 three-day stress periods would be used. When a new stress period begins, all stress data must be redefined; however, most stress packages have an option to reuse the data from the previous stress period. In this example, reuse of river data would be useful if a new stress period is started because the pumping rates change while the river stage stays the same.

Simulation of steady state requires a single stress period with one time step. The length of a steady-state stress period does not have an impact on the computed head because the storage term in the flow equation is set to zero by the internal flow package. A length of one is suggested for steady-state simulations, unless a specific stress period length is needed.

Budget Calculations in the Basic Package

The calculation of the overall volumetric budget is carried out in two parts—the calculation of individual budget entries and the summation of all entries. As explained in Chapter 3, the entries, which correspond to individual components of flow, are calculated in the hydrologic packages and stored in the variable VBVL. VBVL is passed to the Basic Package, which sums and writes the budget entries.

Figure 4–4 is an example of the overall volumetric budget for the end of a stress period. When sufficient space is available, the budget values are written with aligned decimal points and without exponents rather than using Fortran's exponential notation. The use of aligned decimal points aids in observing the relative sizes of the values; however, this can result in an inappropriate number of significant digits being written. The number of valid significant digits depends on a variety of factors such as the accuracy of the computed heads and the precision used in the computer to represent numbers. On computers in which single precision real numbers are represented with 32 bits, the maximum possible number of significant digits is usually seven, and the actual number of significant digits is likely to be less.

```
VOLUMETRIC BUDGET FOR ENTIRE MODEL AT END OF TIME STEP  1 IN STRESS PERIOD   1
-----------------------------------------------------------------------------

   CUMULATIVE VOLUMES      L**3       RATES FOR THIS TIME STEP      L**3/T
   ------------------                 ------------------------

         IN:                                  IN:
         ---                                  ---
           STORAGE =           0.0000           STORAGE =           0.0000
```

```
        CONSTANT HEAD =           0.0000          CONSTANT HEAD =           0.0000
                WELLS =           0.0000                  WELLS =           0.0000
        RIVER LEAKAGE =           0.0000          RIVER LEAKAGE =           0.0000
             RECHARGE =    13608000.0000               RECHARGE =         157.5000

             TOTAL IN =    13608000.0000               TOTAL IN =         157.5000

           OUT:                                      OUT:
           ----                                      ----
              STORAGE =           0.0000                 STORAGE =           0.0000
        CONSTANT HEAD =     4326522.0000          CONSTANT HEAD =          50.0755
                WELLS =     6480000.0000                  WELLS =          75.0000
        RIVER LEAKAGE =     2801081.2500          RIVER LEAKAGE =          32.4199
             RECHARGE =           0.0000               RECHARGE =           0.0000

            TOTAL OUT =    13607603.0000              TOTAL OUT =         157.4954

             IN - OUT =         397.0000               IN - OUT =       4.5929E-03

  PERCENT DISCREPANCY =           0.00     PERCENT DISCREPANCY =           0.00
```

Figure 4–4. Sample of an overall volumetric water budget. (Modified from McDonald and Harbaugh, 1988.)

Output

The primary output of the GWF Process is head distribution. The user may control the frequency at which heads are written to the listing or to separate files through the "Output Control" option of the Basic Package. Other output includes drawdown and budget information; the Output Control option also provides for writing these to listing or separate files. Output Control has its own input file, which is specified in the Name File by file type OC. If Output Control is not utilized, a default output option is invoked—the head distribution and the overall volumetric budget are written to the Listing File at the end of each stress period, and drawdowns are also written to the Listing File.

Zone and Multiplier Arrays for Parameters

As described in Chapter 8, special variables called zone arrays and multiplier arrays can be used to define array parameters. Each multiplier and zone array is given a name, which is used when parameters are defined. The BAS Package reads the zone and multiplier arrays from files specified in the Name File using file types ZONE and MULT, respectively.

Parameter Value File

As described in Chapter 8, some input data for various packages can be defined using parameters. Each parameter has a value that is defined in the input file in which the parameter is defined; however, flexibility has been incorporated to allow all or some parameter values to be specified in a single file, which is called the Parameter Value File. If the Parameter Value File exists, it is read by the Basic Package, and the values in the file replace the values specified when the parameter is defined.

CHAPTER 5
Internal Flow Packages

This chapter documents two packages that simulate internal flow in MODFLOW. The packages are Block-Centered Flow (BCF) and Layer-Property Flow (LPF). Only one internal flow package can be used in a particular simulation. Internally, both packages use identical conceptualization. The difference is in the input data specified by the user and the details of implementation. This chapter begins with a number of sections describing the common conceptualization used in both packages. Later sections describe each of these package in detail. The final section describes the Horizontal-Flow Barrier (HFB) Package, which can be used to supplement BCF or LPF.

An internal flow package as previously mentioned in Chapter 3 calculates the CV, CR, and CC conductance coefficients and the ground-water storage terms in the finite-difference flow (eq. 2–24):

$$\begin{aligned} &CR_{i,j-1/2,k}\left(h^m_{i,j-1,k} - h^m_{i,j,k}\right) + CR_{i,j+1/2,k}\left(h^m_{i,j+1,k} - h^m_{i,j,k}\right) \\ &+ CC_{i-1/2,j,k}\left(h^m_{i-1,j,k} - h^m_{i,j,k}\right) + CC_{i+1/2,j,k}\left(h^m_{i+1,j,k} - h^m_{i,j,k}\right) \\ &+ CV_{i,j,k-1/2}\left(h^m_{i,j,k-1} - h^m_{i,j,k}\right) + CV_{i,j,k+1/2}\left(h^m_{i,j,k+1} - h^m_{i,j,k}\right) \\ &+ P_{i,j,k}\,h^m_{i,j,k} + Q_{i,j,k} = SS_{i,j,k}\left(DELR_j\,DELC_i\,\Delta v_{i,j,k}\right)\frac{h^m_{i,j,k} - h^{m-1}_{i,j,k}}{t^m - t^{m-1}} \; . \end{aligned} \tag{5–1}$$

The m superscript indicates the time step. In equation 5–1, Δr has been replaced by DELR, and Δc has been replaced with DELC. These new names are the variables used in the model code. Also, the subscript for cell thickness, Δv, has been expanded from k to i,j,k to allow for a vertically distorted grid (also called a deformed mesh) as discussed in Chapters 2 and 4.

For solution, equation 5–1 is transformed into the following (eq. 2–26):

$$\begin{aligned} &CV_{i,j,k-1/2}\,h_{i,j,k-1} + CC_{i-1/2,j,k}\,h_{i-1,j,k} + CR_{i,j-1/2,k}\,h_{i,j-1,k} \\ &+ \left(-CV_{i,j,k-1/2} - CC_{i-1/2,j,k} - CR_{i,j-1/2,k} - CR_{i,j+1/2,k} - CC_{i+1/2,j,k} - CV_{i,j,k+1/2} + HCOF_{i,j,k}\right)h_{i,j,k} \\ &+ CR_{i,j+1/2,k}\,h_{i,j+1,k} + CC_{i+1/2,j,k}\,h_{i+1,j,k} + CV_{i,j,k+1/2}\,h_{i,j,k+1} = RHS_{i,j,k} \; . \end{aligned} \tag{5–2}$$

The storage terms are incorporated within the HCOF and RHS coefficients, and the m time-step superscript has been removed.

Basic Hydraulic Conductance Equations

The concept of hydraulic conductance (shortened to conductance in this report) is defined in Chapter 2. Conductance is reviewed here, including the calculation of equivalent conductance for elements arranged in series. Conductance is a combination of several parameters used in Darcy's law. Darcy's law defines one-dimensional flow in a prism of porous material (fig. 5–1) as

$$Q = \frac{KA(h_1 - h_2)}{L} \; , \tag{5–3}$$

where

- Q is the volumetric flow (L^3T^{-1});
- K is the hydraulic conductivity of the material in the direction of flow (LT^{-1});
- A is the cross-sectional area perpendicular to the flow (L^2);
- h_1-h_2 is the head difference across the prism parallel to flow (L); and
- L is the length of the prism parallel to the flow path (L).

Conductance, C, is defined as

$$C = \frac{KA}{L} . \tag{5–4}$$

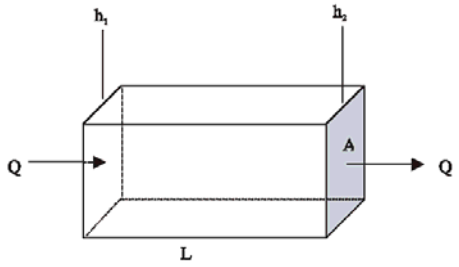


Figure 5–1. Prism of porous material illustrating Darcy's law. (From McDonald and Harbaugh, 1988.)

Therefore, Darcy's law can be written as

$$Q = C(h_1 - h_2) . \tag{5–5}$$

Another form of the conductance definition for horizontal flow in a prism is

$$C = \frac{TW}{L} , \tag{5–6}$$

where

T is transmissivity (K times thickness of the prism) in the direction of flow (L^2T^{-1}); and
W is the width of the prism (L).

Conductance is defined for a particular prism of material and for a particular direction of flow. In an anisotropic medium that is characterized by three principal directions of hydraulic conductivity, the conductances of a prism in these three principal directions will generally differ.

If a prism of porous material consists of two or more subprisms in series (aligned sequentially in the direction of flow) as shown in figure 5–2, and the conductance of each subprism is known, a conductance representing the entire prism can be calculated. The equivalent conductance for the entire prism is the rate of flow in the prism divided by the head change across the prism:

$$C = \frac{Q}{h_A - h_B} . \tag{5–7}$$

Assuming continuity of head across each subprism gives the identity

$$\sum_{i=1}^{n} \Delta h_i = h_A - h_B , \tag{5–8}$$

substituting for head change across each subprism using Darcy's law (eq. 5–5) gives

$$\sum_{i=1}^{n} \frac{q_i}{c_i} = h_A - h_B \ , \tag{5–9}$$

where

q_i is the flow across subprism i, and

c_i is the conductance of subprism i.

Because flow is one dimensional, and we assume no accumulation or depletion in storage, each q_i is equal to the total flow Q; therefore,

$$Q\sum_{i=1}^{n} \frac{1}{c_i} = h_A - h_B \qquad \text{and} \qquad \frac{h_A - h_B}{Q} = \sum_{i=1}^{n} \frac{1}{c_i} \ . \tag{5–10}$$

By comparing equation 5–10 with equation 5–7, the reader can see that

$$\frac{1}{C} = \sum_{i=1}^{n} \frac{1}{c_i} . \tag{5–11}$$

Thus for a set of conductances arranged in series, the inverse of the equivalent conductance equals the sum of the inverses of the individual conductances. When only two subprisms exist, the equivalent conductance reduces to

$$C = \frac{c_1 c_2}{c_1 + c_2} . \tag{5–12}$$

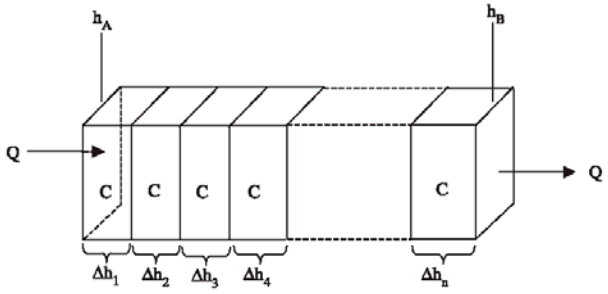


Figure 5–2. Calculation of conductance through several prisms in series. (From McDonald and Harbaugh, 1988.)

Horizontal Conductance

The finite-difference equations in MODFLOW use equivalent conductances between nodes of adjacent cells—that is, "branch conductances"—rather than conductances defined within individual cells. The horizontal conductance terms, CR and CC of equation 5–2, are calculated between adjacent horizontal nodes. CR terms are oriented along rows and thus specify conductance between two nodes in the same row. Similarly, CC terms specify conductance between two nodes in the same column. (Remember that the "1/2" subscript denotes conductance between nodes, as opposed to conductance within a cell. For example, $CR_{i,j+1/2,k}$ represents the conductance between nodes i,j,k and i,j+1,k.)

An internal flow package reads data defining the horizontal hydraulic conductivity for individual cells and calculates conductance between nodes. Four methods of calculating these conductances are supported. The methods

differ in the assumptions about the way the ground-water system varies from cell to cell. The four methods are described in the following two subsections.

Uniform Transmissivity within a Cell

The first method for computing horizontal conductance is based on the assumption that hydraulic conductivity times thickness, that is, transmissivity, is uniform within a cell. There can, however, be a discrete change in transmissivity at the boundary between any two cells. By use of this assumption and the previously mentioned assumption that nodes are in the center of cells, the conductance between the nodes is the equivalent conductance of two half cells in series. Figure 5–3 illustrates two cells along a row based on these assumptions. Substituting the conductance for each half cell (eq. 5–6) into equation 5–12 gives:

$$CR_{i,j+1/2,k} = \frac{\dfrac{TR_{i,j,k}\,DELC_i}{(½)DELR_j}\ \dfrac{TR_{i,j+1,k}\,DELC_i}{(½)DELR_{j+1}}}{\dfrac{TR_{i,j,k}\,DELC_i}{(½)DELR_j} + \dfrac{TR_{i,j+1,k}\,DELC_i}{(½)DELR_{j+1}}} \qquad (5–13)$$

where

$TR_{i,j,k}$ is transmissivity in the row direction at cell i,j,k (L^2T^{-1});
$DELR_j$ is the grid width of column j (L); and
$DELC_i$ is the grid width of row i (L).

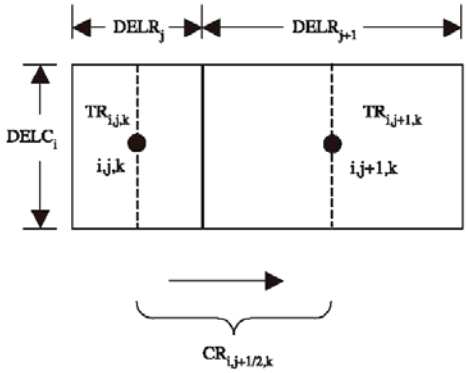


Figure 5–3. Calculation of conductance between nodes using transmissivities and dimensions of cells. (From McDonald and Harbaugh, 1988.)

Simplification of equation 5–13 gives the final equation:

$$CR_{i,j+1/2,k} = 2\,DELC_i \frac{TR_{i,j,k}\,TR_{i,j+1,k}}{TR_{i,j,k}\,DELR_{j+1} + TR_{i,j+1,k}\,DELR_j}\ . \qquad (5–14)$$

The same process can be applied to the calculation of $CC_{i+1/2,j,k}$ giving:

$$CC_{i+1/2,j,k} = 2\,DELR_j \frac{TC_{i,j,k}\,TC_{i+1,j,k}}{TC_{i,j,k}\,DELC_{i+1} + TC_{i+1,j,k}\,DELC_i}\ , \qquad (5–15)$$

where

$TC_{i,j,k}$ is the transmissivity in the column direction at cell i,j,k (L^2T^{-1}).

Equations 5–14 and 5–15 are used unless the transmissivity of both cells is zero, in which case the conductance between the nodes is set equal to zero without invoking the equations.

The above approach for calculating interblock conductance is called the "harmonic mean" method. The reason for this name can be seen by manipulating equation 5–14 into the following:

$$CR_{i,j+1/2,k} = \left(\frac{(½)DELR_j + (½)DELR_{j+1}}{\frac{(½)DELR_j}{TR_{i,j,k}} + \frac{(½)DELR_{j+1}}{TR_{i,j+1,k}}} \right) \frac{DELC_i}{(½)DELR_j + (½)DELR_{j+1}} . \quad (5–16)$$

The term in parentheses is the distance weighted harmonic mean of transmissivity of the two half cells, which by comparison with equation 5–6, can be seen to be the equivalent transmissivity between nodes i,j,k and i,j+1,k.

In order to apply the harmonic mean computation (eq. 5–14 and 5–15) for conductance between confined cells, transmissivity for each cell must be provided as input data or computed. Transmissivity is computed as

$$TR_{i,j,k} = \Delta v_{i,j,k} HK_{i,j,k} \quad (5–17A)$$

$$TC_{i,j,k} = \Delta v_{i,j,k} HK_{i,j,k} HANI_{i,j,k} \quad (5–17B)$$

where

$HK_{i,j,k}$ is the hydraulic conductivity of cell i,j,k in the row direction (LT^{-1});

$\Delta v_{i,j,k}$ is the saturated thickness of cell i,j,k (L); and

$HANI_{i,j,k}$ is the ratio of hydraulic conductivity along columns to the hydraulic conductivity along rows (dimensionless).

For confined cells, $\Delta v_{i,j,k}$ is simply the cell thickness (vertical cell width). This can be computed internally in MODFLOW using cell elevations in the discretization file:

$$\Delta v_{i,j,k} = (TOP_{i,j,k} - BOT_{i,j,k}) \quad (5–18)$$

where

$TOP_{i,j,k}$ is the elevation of the top of cell i,j,k (L); and

$BOT_{i,j,k}$ is the elevation of the bottom of cell i,j,k (L).

For water-table conditions, $\Delta v_{i,j,k}$ depends on head. Further, a cell can convert between confined and water table within a simulation. The following approach for computing saturated thickness can be applied:

$$\text{if } HNEW_{i,j,k} \geq TOP_{i,j,k}, \quad \text{then } \Delta v_{i,j,k} = (TOP_{i,j,k} - BOT_{i,j,k}) ; \quad (5–19A)$$

$$\text{if } TOP_{i,j,k} > HNEW_{i,j,k} > BOT_{i,j,k}, \quad \text{then } \Delta v_{i,j,k} = (HNEW_{i,j,k} - BOT_{i,j,k}) ; \quad (5–19B)$$

$$\text{if } HNEW_{i,j,k} \leq BOT_{i,j,k}, \quad \text{then } \Delta v_{i,j,k} = 0 . \quad (5–19C)$$

When water-table conditions are simulated, cell saturated thickness is recalculated at the start of each iteration using equation 5–19. Transmissivity values (TR and TC) are then calculated as the product of hydraulic conductivity, saturated thickness, and variable HANI (for TC) (eq. 5–17). Finally, conductance is recalculated using

equations 5–14 and 5–15. If head drops below the aquifer bottom (eq. 5–19C), the cell is considered to be fully dewatered, and is set to no flow by changing $IBOUND_{i,j,k}$ to 0. The model also has provision for the resaturation of fully dewatered cells, which is described in another section of this report.

Three Alternative Approaches for Calculating Horizontal Branch Conductances

Goode and Appel (1992) discuss three alternative approaches for calculating the horizontal branch conductances. These alternative approaches are each based on different assumptions about the flow system: (1) transmissivity varies linearly between nodes; (2) the aquifer is flat and homogeneous with a water table; or (3) the aquifer is flat with a water table in which hydraulic conductivity varies linearly between nodes. For each approach, a formulation of interblock transmissivity is developed. The interblock conductance is then computed as the interblock transmissivity times cell width divided by distance between nodes according to equation 5–6.

When transmissivity varies linearly between nodes, the interblock transmissivity derived by Goode and Appel (1992) is:

$$\frac{T_2 - T_1}{\ln\left(T_2/T_1\right)}, \tag{5–20}$$

where T represents transmissivity and subscripts 2 and 1 indicate any two adjacent cells along a row or column. Transmissivity values for the cells are computed using equation 5–17 with equation 5–18 or 5–19 used for thickness. This is called the "logarithmic-mean" interblock transmissivity method. This formulation will produce exact solutions for steady-state, one-dimensional flow if flow is uniform and aligned with the direction of changing transmissivity. To save numerical effort, this function is approximated by

$$\frac{T_1 + T_2}{2} \text{ when } 0.995 \le \frac{T_2}{T_1} \le 1.005 \ .$$

For an unconfined homogeneous aquifer with a flat bottom, the interblock transmissivity derived by Goode and Appel (1992) is:

$$\frac{T_1 + T_2}{2} \ . \tag{5–21}$$

Transmissivity values for the cells are computed using equation 5–17 with equation 5–18 or 5–19 used for thickness. This is called the "arithmetic-mean" interblock transmissivity method. This formulation will produce exact solutions for uniform, steady-state, one-dimensional flow.

For an unconfined aquifer with a flat bottom and with hydraulic conductivity varying linearly between nodes, Goode and Appel derive the interblock transmissivity as:

$$\left(\frac{\Delta v_1 + \Delta v_2}{2}\right)\left(\frac{K_1 - K_2}{\ln\left(K_2/K_1\right)}\right), \tag{5–22}$$

where K represents hydraulic conductivity. Thickness is computed with equation 5–18 or 5–19. This is called the "arithmetic-mean thickness and logarithmic-mean hydraulic conductivity" method. This formulation produces exact solutions for uniform, steady-state, one-dimensional flow when the hydraulic conductivity varies in the direction of flow. To save numerical effort, the hydraulic conductivity part of this function is approximated by $\frac{K_1 + K_2}{2}$ when $0.995 \le \frac{K_2}{K_1} \le 1.005$.

Vertical Conductance

Vertical conductance is calculated assuming that nodes are in the center of cells and that discrete changes can occur in vertical hydraulic conductivity at layer boundaries. Figure 5–4 illustrates this situation for two cells in layers k and k+1. Under these assumptions, the vertical conductance between two nodes will be the equivalent conductance of two half cells in series. Substituting the conductance for each half cell from equation 5–4 into equation 5–11 gives

$$\frac{1}{CV_{i,j,k+1/2}} = \frac{1}{\frac{DELR_j DELC_i VK_{i,j,k}}{(½)\Delta v_{i,j,k}}} + \frac{1}{\frac{DELR_j DELC_i VK_{i,j,k+1}}{(½)\Delta v_{i,j,k+1}}} \quad , \qquad (5–23)$$

where

$VK_{i,j,k}$ is vertical hydraulic conductivity of cell i,j,k and

$\Delta v_{i,j,k}$ is the saturated thickness of cell i,j,k as defined by equation 5–18 for confined cells and equation 5–19 for unconfined cells.

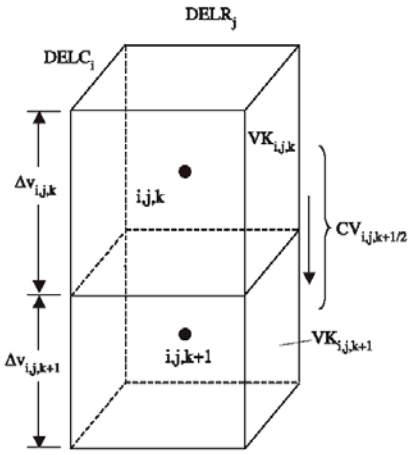


Figure 5–4. Calculation of vertical conductance between two nodes. (Modified from McDonald and Harbaugh, 1988.)

Simplification of equation 5–23 gives

$$CV_{i,j,k+1/2} = \frac{DELR_j DELC_i}{\frac{(½)\Delta v_{i,j,k}}{VK_{i,j,k}} + \frac{(½)\Delta v_{i,j,k+1}}{VK_{i,j,k+1}}} \quad . \qquad (5–24)$$

Figure 5–5 shows a second situation, in which cells i,j,k and i,j,k+1 are separated by a semiconfining unit. If the assumptions are made that the semiconfining unit makes no measurable contribution to the horizontal conductance or the storage capacity of either model layer, then the only effect of the confining bed is to restrict vertical flow between the model cells. Under these assumptions, the impact of the semiconfining unit can be simulated without using a separate layer in the finite-difference grid. This is done by including the semiconfining unit in the calculation of vertical conductance between the nodes. As mentioned previously, this is commonly referred to as the "Quasi-Three-Dimensional" (Quasi–3D) approach.

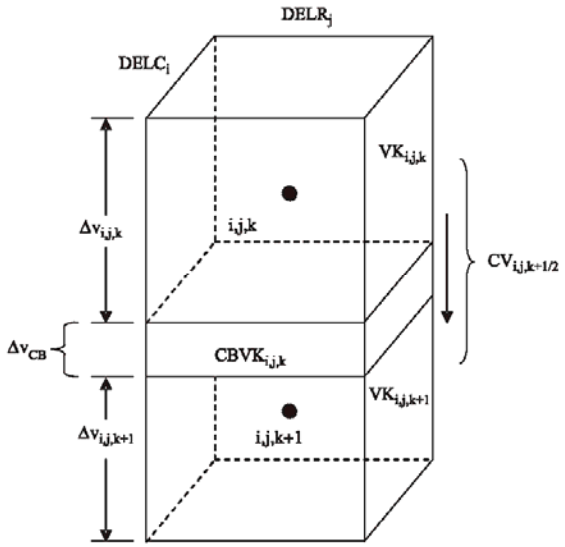


Figure 5–5. Calculation of vertical conductance between two nodes with a semiconfining unit between. (Modified from McDonald and Harbaugh, 1988.)

In the Quasi–3D approach, three intervals must be represented in the summation of conductance between the nodes—the lower half of the upper aquifer, the semiconfining unit, and the upper half of the lower aquifer. Equation 5–11 can be used to write an expression for the inverse of vertical conductance:

$$\frac{1}{CV_{i,j,k+1/2}} = \frac{1}{\dfrac{DELR_j\, DELC_i\, VK_{i,j,k}}{(½)\Delta v_{i,j,k}}} + \frac{1}{\dfrac{DELR_j\, DELC_i\, VKCB_{i,j,k}}{\Delta v_{CB}}} + \frac{1}{\dfrac{DELR_j\, DELC_i\, VK_{i,j,k+1}}{(½)\Delta v_{i,j,k+1}}} \qquad (5–25)$$

where

$VKCB_{i,j,k}$ is the hydraulic conductivity of the semiconfining unit between cells i,j,k and i,j,k+1, and

Δv_{CB} is the thickness of the semiconfining unit.

Δv_{CB} is determined from information in the discretization file, and $VKCB_{i,j,k}$ is input data.

Simplification of equation 5–25 produces

$$CV_{i,j,k+1/2} = \frac{DELR_j\, DELC_i}{\dfrac{(½)\Delta v_{i,j,k}}{VK_{i,j,k}} + \dfrac{\Delta v_{CB}}{VKCB_{i,j,k}} + \dfrac{(½)\Delta v_{i,j,k+1}}{VK_{i,j,k+1}}} \; . \qquad (5–26)$$

Vertical Flow Correction Under Dewatered Conditions

This section describes the vertical flow correction capability that can be activated in both the BCF and LPF Packages. When this capability is activated, the vertical flow calculation is modified if a cell is unconfined (head is below its top elevation) while the cell directly above is fully or partially saturated. Such a situation can occur, for example, under perched conditions. The following paragraphs describe implementation of this capability. The sections on the BCF and LPF Packages describe the situations in which this capability is active.

From equation 5–1, the term that represents flow into cell i,j,k through its lower face, which will be called $q_{i,j,k+1/2}$, is

$$q_{i,j,k+1/2} = CV_{i,j,k+1/2}\left(h_{i,j,k+1} - h_{i,j,k}\right), \tag{5–27}$$

where the m superscript that indicates the time step has been removed. Following the convention of MODFLOW, a positive value of $q_{i,j,k+1/2}$ indicates flow into cell i,j,k, and a negative value indicates flow out of the cell. Equation 5–27 is based on the assumption that cells i,j,k and i,j,k+1 are fully saturated—that is, that the hydraulic head in each cell stands higher than the elevation of the top of the cell.

Situations can occur, however, in which part of a confined aquifer may become unsaturated—for example, when drawdown due to pumpage causes water levels to fall, at least locally, below the top of an aquifer. This condition is most likely to occur when an aquifer is overlain by a lower conductivity unit. In terms of simulation, this condition is shown in figure 5–6. Cells in two model layers are represented. The upper cell (i,j,k) has lower hydraulic conductivity than the underlying cell (i,j,k+1), which represents an aquifer. Pumping from the lower layer has lowered the water level in cell i,j,k+1 below the elevation of the top of the cell; however, cell i,j,k remains fully saturated. Thus, cell i,j,k+1 is effectively unconfined even though the cell above is saturated.

Consider the calculation of flow between nodes in the upper and lower cells. In the upper cell, head is simply whatever head is at that cell—$h_{i,j,k}$. Just below the upper cell, however, unsaturated conditions prevail, so that the pressure sensed on the lower surface of the confining unit is atmospheric—taken as zero in the model formulation. Thus, the head at the bottom of the upper cell is simply the elevation at that point—that is, the elevation of the top of the lower cell. If this elevation is designated $TOP_{i,j,k+1}$, then the actual flow through the confining bed is obtained by substituting $TOP_{i,j,k+1}$ for $h_{i,j,k+1}$ in equation 5–27,

$$q_{i,j,k+1/2} = CV_{i,j,k+1/2}\left(TOP_{i,j,k+1} - h_{i,j,k}\right). \tag{5–28}$$

Thus, the flow will be downward, from cell i,j,k to cell i,j,k+1, but under this condition the flow will no longer be dependent on the water level, $h_{i,j,k+1}$, in the lower cell. The simplest approach to this problem in formulating the equation for cell i,j,k would be to substitute the flow expression of equation 5–28 into equation 5–1, in place of the expression given in equation 5–27. If we consider, however, the matrix of coefficients of the entire system of finite-difference equations, this direct substitution would render this matrix asymmetric, thus generating problems in the solution process.

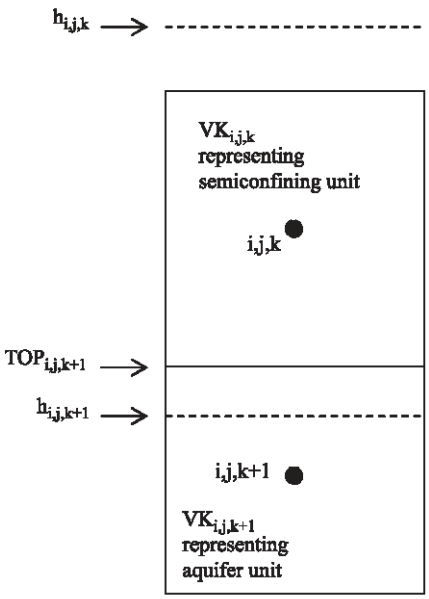


Figure 5–6. Situation in which a correction is required to limit the downward flow into cell i,j,k+1, as a result of partial desaturation of a cell. (From McDonald and Harbaugh, 1988.

To avoid this condition, an alternative approach is used. The flow term of equation 5–27 is allowed to remain on the left side of equation 5–1. The "actual" flow into cell i,j,k is given by equation 5–28, so a correction term, q_c, can be obtained by subtracting equation 5–28 from equation 5–27:

$$\begin{aligned} q_c &= (\text{computed flow into cell i,j,k}) - (\text{"actual" flow into cell i,j,k}) \\ &= CV_{i,j,k+1/2}(h_{i,j,k+1} - TOP_{i,j,k+1}) \quad . \end{aligned} \tag{5–29}$$

This correction term, q_c, should be added to the right side of equation 5–1 to compensate for allowing the computed flow to remain on the left side of equation 5–1. This again introduces a problem of matrix asymmetry because q_c contains the term $h_{i,j,k+1}$. To circumvent this difficulty, q_c is actually computed using the value of $h_{i,j,k+1}$ from the preceding iteration, rather than from the current iteration:

$$q_c^n = CV_{i,j,k+\frac{1}{2}}\left(h_{i,j,k+1}^{n-1} - TOP_{i,j,k+1}\right), \tag{5–30}$$

where

q_c^n is the value of q_c to be added to the right side of equation 5–2 in the nth iteration and

$h_{i,j,k+1}^{n-1}$ is the value of $h_{i,j,k+1}$ from the preceding iteration, n-1.

As convergence is approached, the difference between $h_{i,j,k+1}^{n-1}$ and $h_{i,j,k+1}^{n}$ becomes progressively smaller, and the approximation involved in equation 5–30 becomes more accurate. In the first iteration of each time step, the initial trial value of $h_{i,j,k+1}$ is used in computing q_c. In terms of equation 5–2, which is the form of the flow equation that is directly implemented in MODFLOW, q_c^n is added to $RHS_{i,j,k}$.

The process described above is used in formulating the equations for cell i,j,k when the underlying cell, i,j,k+1, has "dewatered"—that is, when the water level in i,j,k+1 has declined to below the top of the cell. A correction must also be applied in formulating the equations for the dewatered cell itself. To compute this correction, cell i,j,k is considered to be the dewatered cell, and we consider flow into i,j,k from the overlying cell, i,j,k-1. For this case, the computed flow into cell i,j,k from above is

$$CV_{i,j,k-1/2}(h_{i,j,k-1} - h_{i,j,k}) \quad ,$$

whereas, the "actual" flow into the cell is

$$CV_{i,j,k-1/2}(h_{i,j,k-1} - TOP_{i,j,k}) \quad .$$

The difference, computed minus "actual" flow, is thus

$$q_c' = CV_{i,j,k-1/2}\,(TOP_{i,j,k} - h_{i,j,k}), \tag{5–31}$$

where q_c' is added to the right-hand side of equation 5–1. From a programming point of view, the most efficient way to handle this correction is to add the term $CV_{i,j,k-1/2}$ to HCOF on the left side of equation 5–2, while adding the term $CV_{i,j,k-1/2}\ TOP_{i,j,k}$ to the RHS term. This approach causes difficulties for some solution methods, however, because the dominance of the diagonal of the coefficient matrix of the finite-difference equations is reduced. So the same approach is used as described previously for the vertical correction for the cell above the dewatered cell. That is, q_c' is computed on the basis of the head from the previous iteration, and this correction is added to RHS of equation 5–2.

This conceptualization of limiting vertical flow also is valid when the unsaturated region in a lower layer is caused by an overlying confining unit that is simulated using the Quasi–3D approach. Although the confining unit is not simulated as a separate model layer when using this approach, the impact of the confining unit is still simulated through the vertical conductance calculation. The above vertical flow correction is applied with the value of $TOP_{i,j,k}$ being the bottom of the confining bed.

In summary, when the vertical flow correction capability is active, the correction applies whenever dewatering of a cell occurs below a cell that is not fully dewatered. The correction involves two adjustments—one in formulating equation 5–2 as it applies to the overlying cell, and one in formulating equation 5–2 as it applies to the dewatered cell itself. These two adjustments are discussed separately above, in each case using the designation i,j,k to represent the cell for which equation 5–2 is formulated. Keep in mind, however, that both corrections are applied in any dewatering event, and that the form of the corrections has been developed to preserve the symmetry of the coefficient matrix of the finite-difference equations and its diagonal dominance.

Conversion from Dry Cell to Wet Cell

When saturated thickness is zero as defined by head being less than the bottom elevation of a cell (eq. 5–19C), it is clear that a variable-head (wet) cell should convert to dry. Unfortunately, there is not a straight-forward way to know when a dry cell should convert to wet. McDonald and others (1992) describe an approach for doing this and its application in the BCF Package of an earlier version of MODFLOW. The same approach is applied in MODFLOW–2005. The following is a brief overview of the approach. Readers should refer to McDonald and others (1992) for additional details. In particular, that report describes detailed information about numerical instabilities that can result from using this option, and several test problems are included.

A dry cell converts to wet based upon the head in an adjacent cell compared to a wetting threshold, THRESH, for the cell. If the head at the adjacent cell equals or exceeds the threshold at the start of a solution iteration, the dry cell is converted to wet. One option is to allow any of the four cells that are directly adjacent to the cell horizontally or the cell directly below to cause a dry cell to convert to wet are. Another option is to use only the cell below to determine when a dry cell becomes wet. The head below can be a better wetting indicator than the head at horizontally adjacent cells when the head variations between adjacent horizontal cells are larger than the vertical head variations, which is frequently the case.

A primary difficulty with this approach is that the value of THRESH must be determined by trial and error. If THRESH is too low, a cell may be incorrectly converted to wet. That is, a cell may convert to wet and then reconvert to dry in later solver iterations because the head does not stay above the bottom elevation. This cycle of converting between wet and dry conditions can repeat indefinitely, preventing the solver from converging. Larger values of THRESH can avoid repeated conversion between wet and dry, but they increase the non-uniqueness of the solution (McDonald and others, 1992, p. 8).

When a cell converts to wet, the initial estimate of head is established according to one of two equations:

$$h_{i,j,k} = BOT_{i,j,k} + WETFCT(h_n - BOT_{i,j,k}) \qquad (5–32A)$$

or

$$h_{i,j,k} = BOT_{i,j,k} + WETFCT(THRESH_{i,j,k}) , \qquad (5–32B)$$

where

WETFCT is a user-specified constant, generally between 0 and 1, and

h_n is the head at the neighboring cell that causes cell i,j,k to convert to wet.

The user chooses the equation that works best for the problem being simulated. The initial estimate of head at a cell that converts to wet is theoretically unimportant because the solver should calculate the correct value in subsequent iterations. In practice, however, the initial estimate is often important for solver efficiency; that is, if a poor choice is made, the solver convergence may be slower than optimum. A bad choice for initial head also can cause the cell to oscillate between wet and dry.

Another user-specified option is the solver iteration interval for attempting to wet cells. For example, if the interval is 4, then cell wetting is attempted every 4 iterations. This can prevent the normal fluctuations in heads that can occur in iterative solvers in the next few iterations after cells are converted to wet from inappropriately triggering more cell wetting. All of these options are included to help make the solution process more stable. Refer to McDonald and others (1992) for more information about solution instability and how to deal with it.

Storage Formulation

A distinction is made in MODFLOW between confined layers in which storage terms remain constant throughout the simulation, and those layers in which the storage terms may "convert" from a confined value to a water-table value, or conversely from a water-table value to a confined value, as the water level in a cell falls below or rises above the top of the cell. For a confined layer, the storage formulation is based upon a direct application of the storage expression in equation 5–1. This expression for the rate of accumulation of water into a single cell is:

$$SS_{i,j,k}\left(DELR_j\, DELC_i\, \Delta v_{i,j,k}\right)\frac{h_{i,j,k}^{m} - h_{i,j,k}^{m-1}}{t^{m} - t^{m-1}} \qquad (5\text{–}33)$$

where

$SS_{i,j,k}$ is the specific storage of the material in cell i,j,k (L^{-1});
$DELR_j$ is the width of cells in column j (L);
$DELC_i$ is the width of cells in row i (L);
$\Delta v_{i,j,k}$ is the cell thickness as defined by equation 5–18 (L);
$h_{i,j,k}^{m}$ is the head in cell i,j,k at the end of time step m (L);
$h_{i,j,k}^{m-1}$ is the head in cell i,j,k at the end of time step m-1 (L);
t^{m} is the time at the end of time step m (T); and
t^{m-1} is the time at the end of time step m-1 (T).

The notation $SC1_{i,j,k}$ is introduced where $SC1_{i,j,k} = SS_{i,j,k}(DELR_j DELC_i \Delta v_{i,j,k})$. In this report the term $SC1_{i,j,k}$ is termed the "storage capacity" or the "primary storage capacity" of cell i,j,k; the "primary" designation is used to distinguish $SC1_{i,j,k}$ from a secondary storage capacity, which is used when storage term conversion is invoked, as explained in the following section. By using the concept of storage capacity, equation 5–33 can be rewritten as

$$SC1_{i,j,k}\frac{h_{i,j,k}^{m} - h_{i,j,k}^{m-1}}{t^{m} - t^{m-1}} \; . \qquad (5\text{–}34)$$

This expression is separated into two terms for inclusion into equation 5–2:

$$\frac{SC1_{i,j,k}\, h_{i,j,k}^{m}}{t^{m} - t^{m-1}} \; ,$$

which is incorporated in the left side of equation 5–2 through the term $HCOF_{i,j,k}$, and

$$\frac{-SC1_{i,j,k}\, h_{i,j,k}^{m-1}}{t^{m} - t^{m-1}} \; ,$$

which is included in the term $RHS_{i,j,k}$ on the right side of equation 5–2.

Storage Term Conversion

The primary storage capacity described above, $SC1_{i,j,k}$ is adequate for simulations in which a cell is confined throughout the course of a simulation; however, if the water level is below the top of a cell during a simulation, then the cell is under a water-table condition, and a different formulation for the rate of accumulation of water into a single cell applies:

$$SY\left(DELR_j DELC_i\right)\frac{h_{i,j,k}^{m} - h_{i,j,k}^{m-1}}{t^{m} - t^{m-1}}\ , \tag{5–35}$$

where

SY is the specific yield (dimensionless).

A secondary storage capacity, $SC2_{i,j,k}$, is used to represent specific yield multiplied by cell area, and equation 5–35 can be written as

$$SC2_{i,j,k}\ \frac{h_{i,j,k}^{m} - h_{i,j,k}^{m-1}}{t^{m} - t^{m-1}}\ . \tag{5–36}$$

During any one time step, four storage conditions are possible for each cell:

The cell is confined for the entire time step,

The cell is unconfined for the entire time step,

The cell converts from confined to unconfined, and

The cell converts from unconfined to confined.

The following expression for rate of accumulation in storage in cell i,j,k is used to handle all four possibilities:

$$\frac{SCB\left(h_{i,j,k}^{m} - TOP_{i,j,k}\right) + SCA\left(TOP_{i,j,k} - h_{i,j,k}^{m-1}\right)}{t^{m} - t^{m-1}}\ , \tag{5–37}$$

where

TOP is the elevation of the top of the model cell,

SCA is the storage capacity in effect in cell i,j,k at the start of the time step; and

SCB is the "current" storage capacity—that is, the storage capacity in effect during the current iteration.

Equation 5–37 handles the four situations as follows. First, consider a case in which the head in cell i,j,k at the beginning of time step m ($h_{i,j,k}^{m-1}$) is above the top of the cell (fig. 5–7). Because no free surface exists in the cell at the start of the time step, the storage capacity at that time is taken as the confined storage capacity; that is, SCA is set equal to $SC1_{i,j,k}$. If, during a given iteration for time step m, the computed value of head for the end of the time step ($h_{i,j,k}^{m}$) is found to be above the top of the cell, then SCB for the following iteration also is set equal to $SC1_{i,j,k}$; equation 5–37 for that iteration then reverts to the form of equation 5–34, the equation for confined storage. If, however, the computed value of $h_{i,j,k}^{m}$ in a given iteration turns out to be below the top of the cell, as shown in figure 5–7, the value of SCB for the following iteration is set equal to SC2, the unconfined storage capacity. In this case, the computed rate of release of water from storage in the time step has two components:

$\frac{SC1_{i,j,k}\left(TOP_{i,j,k} - h_{i,j,k}^{m-1}\right)}{t^m - t^{m-1}}$, the rate of release from confined or compressive storage, and

$\frac{SC2_{i,j,k}\left(h_{i,j,k}^{m} - TOP_{i,j,k}\right)}{t^m - t^{m-1}}$, the rate of release from water-table storage.

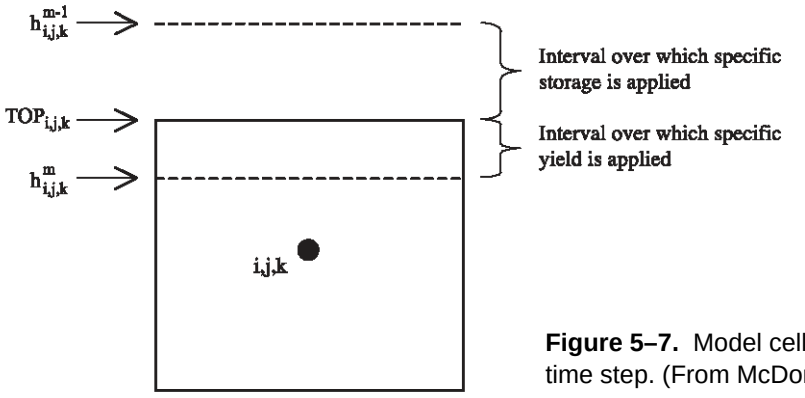


Figure 5–7. Model cell for which two storage factors are used during one time step. (From McDonald and Harbaugh, 1988.)

If the head at the beginning of the time step, $h_{i,j,k}$, is below the top of cell i,j,k, so that a free surface exists within the cell, SCA in equation 5–37 is set equal to $SC2_{i,j,k}$. If, during an iteration for time step m, the computed value of head for the end of the time step is below the top of the cell, SCB in the subsequent iteration is also set equal to $SC2_{i,j,k}$, and equation 5–37 reverts to the form of equation 5–36. If, however, the computed head for the end of the time step turns out to be above the top of the cell, SCB in the subsequent iteration is set equal to $SC1_{i,j,k}$, the confined storage capacity. This situation occurs during intervals of rising water level, and again two components are computed for the rate of accumulation of water in storage—one corresponding to unconfined or water-table storage and one corresponding to confined or compressive storage.

Equation 5–37 can be rearranged as follows:

$$\frac{SCB \times h_{i,j,k}^{m}}{t^m - t^{m-1}} + \frac{SCA\left(TOP_{i,j,k} - h_{i,j,k}^{m-1}\right) - SCB \times TOP_{i,j,k}}{t^m - t^{m-1}} \quad . \tag{5–38}$$

Equation 5–38 represents rate of accumulation in storage, which is the right side of equation 5–1. In the formulation of equation 5–2, therefore, the expression $\frac{SCB}{t^m - t^{m-1}}$ is subtracted from $HCOF_{i,j,k}$ on the left-hand side, while the term

$$\frac{SCA\left(TOP_{i,j,k} - h_{i,j,k}^{m-1}\right) - SCB \times TOP_{i,j,k}}{t^m - t^{m-1}}$$

is added to $RHS_{i,j,k}$ on the right-hand side.

Storage Formulation for Steady-State Simulations

For steady-state conditions, no storage effects exist; thus, nothing is added to RHS and HCOF coefficients to represent storage. A stress period can be specified as steady state in the discretization file. Normally, a steady-state

stress period will have a single time step. When all stress periods in a simulation are steady state, storage data are not included in the input data.

Applicability and Limitations of Optional Formulations

The options for calculation of horizontal conductance under water-table conditions, vertical flow correction, and storage term conversion were all developed on the assumption that each model layer corresponds to a distinct aquifer or permeable horizon, and that these horizons are separated by distinct units of low permeability. Use of these options where these conditions are not satisfied may lead to a variety of problems and inaccuracies in simulation. For example, if the option for horizontal conductance calculation under water-table conditions is used where a water-table aquifer is represented by several model layers, and the water table is expected to traverse more than one layer during simulation, incorrect conversion of cells to a no-flow condition may occur as iterations are carried out. If the wetting option is not used, conversion to no flow is irreversible. The wetting option allows conversion to no flow to be reversed, yet conversion back and forth can sometimes repeatedly occur so that solver convergence is not achieved. Thus, care should be exercised in the decision to use these options.

Block-Centered Flow Package

This section discusses the unique aspects of BCF, including the overall requirements for input data. The detailed input instructions are contained in Chapter 8.

BCF utilizes a layer-type code (LAYCON) to classify layers according to which of the head-dependent formulations are used. The user specifies the values of LAYCON for each layer. There are four possible values for LAYCON: 0 indicates a confined layer, 1 indicates a water-table layer, 2 indicates a limited convertible layer, and 3 indicates a fully convertible layer. Explanations for each value follow.

Layer-type 0 — Confined

In this category, all cells in the layer are assumed to have unchanging transmissivity throughout the simulation. The user must externally compute transmissivity in the row direction for cells, which is directly input into variable Tran, rather than having BCF compute transmissivity from cell thickness and hydraulic conductivity. Thus, the cell elevations in the discretization file are not used for computing transmissivity. Horizontal interblock conductance is computed from the user-specified transmissivities using the specified interblock transmissivity equation (for example equations 5–14 and 5–15 if harmonic mean is selected); however, the "arithmetic-mean thickness and logarithmic-mean hydraulic conductivity" method is not supported because that method requires cell hydraulic conductivity. Modification of transmissivity is not provided for as water level varies. If the simulation is transient, storage coefficient (the product of specific storage and cell thickness) is specified by the user in variable Sf1, and as with Tran, Sf1 must be computed outside of the model. Cell drying, cell wetting, storage term conversion, and vertical flow correction are not active. The conductance and storage terms are retained without change throughout the simulation.

This layer type normally is used to simulate confined conditions, but could also be used to simulate a layer in which unconfined conditions will always prevail, provided drawdowns are expected to be a small fraction of layer thickness and the vertical flow correction is not needed. If used to simulate an unconfined layer, specific yield should be entered in place of the confined storage coefficient.

Layer-type 1 — Unconfined

This layer type is utilized only in a single-layer model or in the uppermost layer of a model, and only where unconfined conditions are expected to persist in the layer throughout the entire period of simulation. Transmissivities are computed at each iteration as the products of hydraulic conductivity and saturated thickness values within the layer. There is no check to determine if the head exceeds the top elevation, so the thickness

calculation includes only equations 5–19B and 5–19C. All four formulations for calculating horizontal interblock conductance are supported. Hydraulic conductivity is read into variable HY. Bottom elevations are determined from the elevations in the discretization file. If the simulation is indicated as transient, specific yield values are entered in the input variable Sf1. No provision is made for storage term conversion. Because use of this layer type would be inappropriate except in the uppermost layer, a check of the layer number is made whenever LAYCON is given a value of one; if the layer number is not also equal to 1, indicating the uppermost model layer, an error message is printed. Cell drying is active. Cell wetting can be activated as described below. Vertical flow correction does not apply because this layer type is only allowed for the top model layer.

Layer-type 2 — Limited Convertible

This layer type is used where heads may alternate between confined and unconfined conditions, so that storage term conversion and limitation of flow from above under dewatered conditions are both desirable. An assumption is made, however, that the saturated thickness will remain a high fraction of the layer thickness throughout the period of simulation, and accordingly transmissivity is not recalculated as the product of hydraulic conductivity and saturated thickness. Transmissivity values are entered in the variable Tran as they are for a confined layer (LAYCON 0). The "arithmetic-mean thickness and logarithmic-mean hydraulic conductivity," but the other three methods for calculating horizontal interblock conductance are supported. For transient simulations, the storage term conversion option requires that both confined storage coefficient and specific yield be specified for each cell (variables Sf1 and Sf2, respectively). The vertical flow correction is active. The top elevation, which is required for the storage computations and vertical flow correction, is determined from elevation data in the discretization file. Cell drying and wetting are inactive.

Layer-Type 3 — Fully Convertible

This layer type incorporates all of the Block-Centered Flow options associated with water-table conditions. Transmissivities are recalculated at each iteration using hydraulic conductivity and layer top and bottom elevations, and both storage term conversion and vertical flow correction are implemented. The thickness calculation includes all three parts of equation 5–19. Confined storage coefficients are entered in the input variable Sf1, hydraulic conductivity values are then entered in the variable HY, and aquifer bottom and top elevations are determined from elevations in the discretization file. Specific yield values are entered in the variable Sf2. Cell drying is active, and wetting can be activated as described below. All four formulations for computing horizontal interblock conductance are supported.

Other BCF Conceptualizations

For all values of LAYCON, the user is required to enter vertical hydraulic conductivity divided by the flow distance (called vertical leakance) into variable VCONT. Within BCF, vertical leakance is multiplied by cell area to obtain vertical conductance. Vertical leakance for a layer represents the leakance between the nodes in that layer and the nodes in the next lower layer. For example, the VCONT values entered during the input sequence for layer 1 actually apply to the interval between the midpoint of layer 1 and the midpoint of layer 2. Accordingly, VCONT is not input for the bottom layer.

VCONT can be computed for the two conceptualizations presented in the Vertical Conductance Section by dividing the vertical conductance by cell area. When there is no Quasiv3D confining bed (eq. 5–24), the result is:

$$VCONT_{i,j,k+1/2} = \frac{1}{\frac{(½)\Delta v_{i,j,k}}{VK_{i,j,k}} + \frac{(½)\Delta v_{i,j,k+1}}{VK_{i,j,k+1}}}. \qquad (5–39)$$

When there is a Quasi–3D confining bed (eq. 5–26), the result is:

$$VCONT_{i,j,k+1/2} = \frac{1}{\frac{(½)\Delta v_{i,j,k}}{VK_{i,j,k}} + \frac{\Delta v_{CB}}{VKCB_{i,j,k}} + \frac{(½)\Delta v_{i,j,k+1}}{VK_{i,j,k+1}}} \quad . \qquad (5–40)$$

Although in many situations one of these two conceptualizations will likely be desired, the approach in BCF of having vertical leakance specified as user input allows the user to compute vertical leakance using any desired conceptualization. Because vertical distance is incorporated in vertical leakance, BCF does not use cell elevations to determine vertical conductance. Vertical conductance remains constant throughout a simulation in BCF; therefore, vertical conductance is not modified when vertical flow is limited.

To reduce the amount of input data, the interblock transmissivity method is combined with the layer-type flag (LAYCON). The combined flag is called Ltype in the input instructions. The left digit of Ltype indicates the interblock transmissivity method and the right digit is the layer type. The interblock transmissivity values are:

0 indicates harmonic mean,
1 indicates arithmetic mean,
2 indicates logarithmic mean, and
3 indicates arithmetic-mean thickness and logarithmic-mean hydraulic conductivity.

For example, an Ltype value of 13 for a layer indicates arithmetic mean for interblock transmissivity and 3 for LAYCON.

A single IWDFLG variable for the entire grid determines whether the wetting capability (conversion of dry cells to wet) is active. Even when wetting is active, however, wetting will occur only in layers where LAYCON is 1 or 3. The user must specify variable WETDRY for every layer where LAYCON is 1 or 3. The wetting threshold is determined as the absolute value of WETDRY. Negative values of WETDRY indicate that wetting is based only on the head in the cell below. Positive values indicate that wetting is based on head from the four surrounding horizontal cells and the cell below. A zero value in WETDRY deactivates wetting for that cell.

Within each layer, the required hydraulic data must be specified for every cell, including constant-head and no-flow cells. For no-flow cells, the entered values are never used in calculation, and thus any values may be specified; for constant-head cells, the storage terms are not used but the other data values are, and realistic values for those cells must be entered. BCF does not support the parameter method (Chapter 8) for defining any input data.

As indicated in equations 5–17A and 5–17B, two values of transmissivity are required at each cell—one in the row direction and one in the column direction. To reduce input effort, only a single value of transmissivity or hydraulic conductivity (depending on LAYCON) is read for each cell, giving only the value in the row direction; the row-direction values are subsequently multiplied by an anisotropic factor to obtain the corresponding column-direction values. A single value of the anisotropic factor is specified by the user for each layer, through the one-dimensional variable TRPY (NLAY).

Layer-Property Flow Package

This section discusses the unique aspects of the LPF Package, including the overall requirements for input data. The detailed input instructions are contained in Chapter 8.

The LPF Package supports two types of layers – confined and convertible. A confined layer is one in which transmissivity is constant throughout the simulation. Transmissivity is computed from hydraulic conductivity and cell elevations using equations 5–18 and 5–17. In transient simulations, the storage flow is computed from specific storage. The cell drying, cell wetting, and vertical flow correction capabilities are not active.

A convertible layer is one in which transmissivity varies based on head throughout the simulation. Transmissivity is computed during each solver iteration based on cell thickness computed from equation 5–19. The computation of transmissivity includes the possibility for a cell converting to no flow when head goes below the

bottom elevation. The vertical flow correction capability is active, and the cell wetting capability can be applied if desired.

For convertible layers, the storage contribution to the flow equation is determined from confined and/or unconfined storage, depending on head compared to the top elevation of cells. Values of specific storage for each cell in the layer are read through the variable Ss, and values of specific yield for each cell in the layer are read through the variable Sy. The specific-storage values are multiplied by cell volume to obtain confined storage capacity, and the specific-yield values are multiplied by cell area to obtain unconfined storage capacity. Thus, two storage capacities are stored, and the program uses the appropriate value, depending on head conditions at each iteration.

LPF supports the "harmonic mean," "logarithmic mean," and "arithmetic-mean and logarithmic-mean hydraulic conductivity" horizontal interblock transmissivity formulations. Because the "arithmetic-mean thickness and logarithmic-mean hydraulic conductivity" method reduces to the "arithmetic-mean" method when the aquifer is homogeneous, the "arithmetic-mean" method is not separately included in the LPF Package. That is, only equations 5–20 and 5–22 are included in the LPF Package along with the harmonic mean method.

As shown in equation 5–17A, the transmissivity in the row direction for a cell is computed as the product of the cell hydraulic conductivity and the saturated thickness. The row-direction values are subsequently multiplied by an anisotropic factor to obtain the corresponding column-direction values (eq. 5–17B). A single value of the anisotropic factor can be specified by the user for each layer through variable CHANI; or a layer can have one value of horizontal anisotropy for each cell in variable HANI.

In confined and convertible situations, vertical conductance is computed using equation 5–24 or 5–26, depending on whether a Quasi–3D confining bed exists. Vertical conductivity is required for all cells to allow vertical conductance to be computed. Vertical hydraulic conductivity can be directly specified as input data, or the ratio of horizontal to vertical hydraulic conductivity can be input. Further, the vertical conductance of the confining bed is required when the Quasi–3D approximation is used. In convertible layers, vertical conductance is recomputed every iteration. Also, vertical conductance is adjusted when the vertical flow correction is active. This adjustment removes the component of vertical conductance for the unconfined cell that is below a wet cell.

The computations for vertical conductance for convertible layers can cause numerical instabilities because the change in conductance can be quite large between successive solver iterations. Accordingly, two options have been incorporated to avoid solution instability. The first option, called CONSTANTCV, keeps vertical conductance constant throughout the simulation, even for convertible layers. Vertical conductance is calculated from the cell thickness rather than saturated thickness. The second option is called NOCVCORRECTION, which prevents adjustment of vertical conductance when the vertical flow correction is applied. Note that the CONSTANTCV option automatically invokes the NOCVCORRECTION option.

In LPF, the convertible formulation should be used if layer 1 represents a water table. Also, the elevation of the top of layer 1, which is specified in the Discretization File, should be higher than the water level will reach at any time in the simulation. This will result in the use of equation 5–19B for calculating saturated thickness. It is the user's responsibility in this situation to check to see that the simulated water level has not exceeded the top elevation, which would trigger the use of equation 5–19C for calculating saturated thickness. A logical approach for specifying the top elevation for a layer-1 water table is to set it equal to land-surface elevation. That is, in most situations the water table is not expected to exceed land-surface elevation.

The water-table formulations based on saturated thickness for convertible layers (horizontal conductance, vertical conductance, cell drying and wetting) can result in solution instability (excessive iterations or failure to converge) when saturated thickness is thin or layers have a large slope. One way to deal with such a problem is to simulate a convertible layer as confined. Although this approach results in errors to the extent that the computed head results in a saturated thickness different from the confined thickness, repeated trial and error simulations can then be made in which model data are adjusted to cause saturated thickness to be close to the confined thickness.

Two difficulties occur when approximating a convertible layer as confined. One difficulty is that for confined layers, the storage term in the flow equation is based on specific storage as input data. But if the cells are really unconfined, then specific yield should be used for the storage computations. This requires a user to compute specific storage as specific yield divided by cell thickness. This computation is conceptually simple, but inconvenient to implement if cell thickness varies throughout a layer. In this version of LPF (Version 7), an option named STORAGECOEFFICIENT has been added to resolve this problem. When STORAGECOEFFICIENT is used, then

the specific storage input values are viewed as storage coefficients, which are comparable in concept to specific yield.

The second difficulty with treating convertible layers as confined is that a user frequently wants to rerun the model with different saturated thickness values. This can be done by modifying the top elevations of layers in the Discretization File; however, the top elevation for one layer is also the bottom elevation for another. For example, the top of layer 2 is the bottom of layer 1 (or the bottom of the confining bed below layer 1 if the Quasi–3D option is being used). One generally does not want the bottom elevations adjusted when using this approach. An option named THICKSTRT has been added to LPF to provide a resolve this difficulty. The THICKSTRT option in combination with a negative LAYTYP value causes the specified layers to be confined and to have the cell thickness for conductance calculations computed from initial head (variable STRT read by the Basic Package) minus the bottom elevation rather than top elevation minus bottom elevation. The user can therefore rerun the model with different values for the initial head. The cell thickness for storage calculations is still top elevation minus bottom elevation for these layers unless the STORAGECOEFFICIENT option as described above is used.

The optional formulations described above are controlled by option variables, which have one value for each layer; that is, the options can be turned on or off on a layer by layer basis. The option variables are:

LAYCBD—Quasi–3D confining bed flag (read as part of the Discretization File),
LAYTYP—Layer-type flag,
LAYAVG—Interblock transmissivity flag,
CHANI—Horizontal anisotropic flag and value,
LAYVKA—Vertical anisotropic flag, and
LAYWET—Wetting flag.

The Quasi–3D confining bed flag (LAYCBD) indicates whether a layer is underlain by a confining bed that is simulated using the Quasi–3D approach. This determines how vertical conductance is calculated.

LAYCBD = 0—there is no Quasi–3D confining bed below the layer, so vertical conductance is calculated using equation 5–24.
LAYCBD ≠ 0—there is a Quasi–3D confining bed below the layer, so vertical conductance is calculated using equation 5–26.

The layer-type flag (LAYTYP) indicates whether a layer is confined or convertible. This controls whether (1) saturated thickness varies with head, (2) storage term conversion is used, and (3) vertical flow from above is limited under dewatered conditions:

Layer-type flag = 0 – The layer is simulated as confined. For a layer of this type, there is no provision for modification of saturated thickness as head varies, for storage term conversion, or for limitation of vertical flow from above if the calculated hydraulic head falls below the top of the cell.

Layer-type flag >0 – The layer is simulated as convertible between confined and unconfined. When the calculated hydraulic head is below the top of the cell, all of the options associated with water-table conditions are implemented. Saturated thickness and transmissivity are recalculated at each iteration on the basis of head, and both storage term conversion and limitation of flow from above under dewatered conditions are implemented.

Layer-type flag <0 – The layer is simulated as convertible unless the THICKSTRT option is in effect. When THICKSTRT is in effect, a negative value of LAYTYP indicates that the layer is confined, and its saturated thickness will be computed as the initial head minus (variable STRT) the bottom elevation.

The Interblock transmissivity flag (LAYAVG) indicates the method used for calculating horizontal interblock conductance:

LAYAVG = 0—Harmonic mean transmissivity is used. This assumes that transmissivity is uniform within a cell, and that changes occur as discrete changes at cell boundaries.

LAYAVG = 1—logarithmic mean transmissivity is used. This assumes that hydraulic conductivity varies linearly between nodes.

LAYAVG = 2—arithmetic-mean thickness and logarithmic-mean hydraulic conductivity are used. This is based on a water-table condition.

The horizontal anisotropic flag (CHANI) indicates whether horizontal anisotropy is constant for a layer, or can vary at each cell in a layer. In either case, the grid is assumed to be aligned with the principal directions of hydraulic conductivity.

CHANI ≤ 0.0—Horizontal anisotropy can vary at each cell in the layer. A separate layer-data variable of the horizontal anisotropic values (HANI) will be read in the input data.

CHANI > 0.0—Horizontal anisotropy is constant for all cells in the layer. The anisotropy is the value of the flag. For example, set CHANI=1.0 for an isotropic layer.

The vertical anisotropic flag (LAYVKA) indicates the type of values that are specified in variable VKA.

LAYVKA = 0—Values of VKA are vertical hydraulic conductivity of the model layer.

LAYVKA ≠ 0—Values of VKA are the ratio of horizontal hydraulic conductivity (along rows) to vertical hydraulic conductivity for the model layer.

The wetting flag (LAYWET) indicates whether the wetting capability is on or off.

LAYWET = 0—Wetting capability is off.

LAYWET ≠ 0—Wetting capability is on for the layer. This is valid only if the layer-type flag indicates that the layer is convertible (LAYTYP is not 0).

Most of the data values that must be specified for the LPF Package are layer data, meaning that the data must be defined for all cells in one or more of the model layers. As used in the equations in this chapter that define the LPF formulations, the layer-data variables have three-dimensional subscripts, indicating that there are values for all layers; however, some of the variables do not always exist for all layers. The layers for which these layer variables exist are determined by the values in the abovementioned option variables. The layer-data variables defined for use by LPF are:

HK—horizontal hydraulic conductivity along rows,

HANI—Ratio of horizontal hydraulic conductivity along columns to hydraulic conductivity along rows,

VKA—Vertical hydraulic conductivity or the ratio of horizontal hydraulic conductivity along rows (HK) to vertical hydraulic conductivity,

VKCB—Vertical hydraulic conductivity of a confining bed below a layer,

Ss—specific storage,

Sy—specific yield, and

WETDRY—wetting threshold combined with a wetting flag.

The discretization data—time and space—required by the LPF Package to formulate the flow and storage terms are specified in the Basic Package Discretization File. All other data required by LPF are read from the LPF data file. The input instructions (Chapter 8) describe the details of how the LPF input data are read. All of the layer variables can be directly read, and variables HK, HANI, VKA, VKCB, Ss, and Sy also can be defined by using parameters. The same method of data definition (directly reading or using parameters) must be used for all the layers for a given variable—that is, a particular variable cannot be defined by combining direct reading and using parameters.

When layer variables are read directly, a value is read for every cell even if the value will not be used. For no-flow cells, the values of variables are never used in any calculation, and thus any values may be specified; for constant-head cells, the Ss, Sy, and WETDRY variables are not used but the other variables are. When variables are

defined by using parameters, a value is required for variable-head cells (IBOUND > 0) and constant-head cells (IBOUND < 0). If a value is not assigned at one or more variable- or constant-head cells, the program will print an error message and stop.

Horizontal Flow Barrier Package

The capability to simulate thin, vertical, and low-permeability geologic features (barriers) was previously added to MODFLOW with the introduction of the Horizontal Flow Barrier (HFB) Package (Hsieh and Freckleton, 1993). This package has been incorporated into MODFLOW–2005. Figure 5–8 (from Hsieh and Freckleton, 1993) shows a conceptualization of a model layer containing a low-permeability barrier. HFB adjusts the horizontal hydraulic conductance computed by the internal flow package to account for the barriers. HFB is not a complete internal flow package, but rather acts as a supplement to the BCF and LPF Packages.

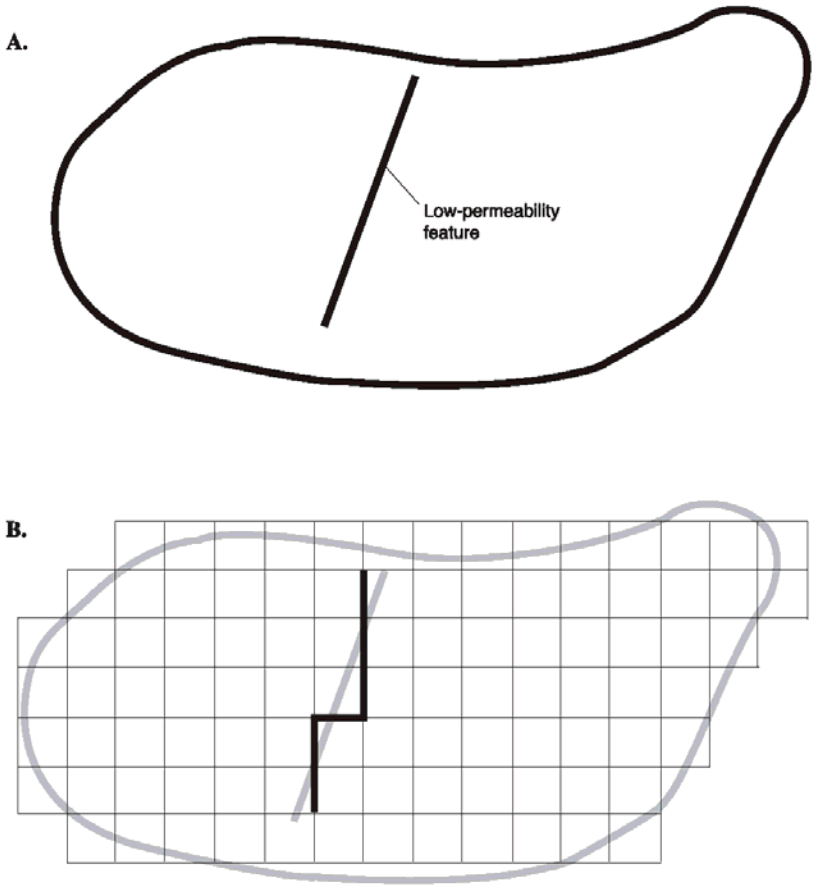


Figure 5–8. Schematic representation of (A) a model layer and (B) a grid with a low-permeability feature represented in the grid as a series of six horizontal-flow barriers. (From Hsieh and Freckleton, 1993.)

In the HFB Package, a barrier is conceptualized as being located on the boundary between two adjacent finite-difference cells in the same layer (fig. 5–9). The barrier is further conceptualized as being in series with the horizontal conductance computed by the internal flow package. Using equation 5–12, the equivalent conductance for the additional conductance of the barrier, $C_{Barrier}$, in series with the original cell conductance along a row, $CR^{Original}_{i,j+1/2,k}$, (fig.5–9A) is:

$$CR_{i,j+1/2,k} = \frac{CR^{Original}_{i,j+1/2,k} \, C_{Barrier}}{CR^{Original}_{i,j+1/2,k} + C_{Barrier}} \, . \qquad (5–41)$$

$C_{Barrier}$ is computed by use of equation 5–4:

$$C_{Barrier} = \frac{K_{Barrier} \Delta v_{Barrier} DELC_i}{L_{Barrier}}, \quad (5–42)$$

where

$K_{Barrier}$ is hydraulic conductivity of the barrier,

$\Delta v_{Barrier}$ is vertical thickness of the barrier, which is the average saturated thickness of the two cells, and

$L_{Barrier}$ is the distance across the barrier in the flow direction.

Conductance along a column is computed similarly. The distance across the barrier is assumed to be a small fraction of the distance between nodes so that the original conductance does not require correction because of the barrier. The HFB Package requires as input the "hydraulic characteristic" for a barrier, which is defined here as $K_{Barrier}/L_{Barrier}$. HFB includes the capability to use parameters to define the hydraulic characteristic. HFB does not simulate vertical barriers.

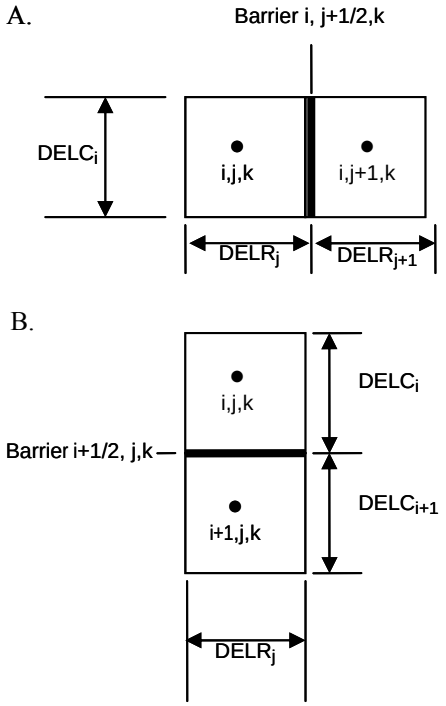


Figure 5–9. Schematic representation of a horizontal-flow barrier separating (A) two adjacent model cells in the same row and (B) two adjacent cells in the same column. (From Hsieh and Freckleton, 1993.)

CHAPTER 6
CONCEPTUALIZATION AND IMPLEMENTATION OF STRESS PACKAGES

This chapter documents the conceptualization and implementation of the packages in the Ground-Water Flow (GWF) Process that simulate hydrologic stresses to a ground-water system. The stress packages add terms to the flow equation representing inflows or outflows. Mathematically, these are boundary conditions. The six packages documented are the Well (WEL) Package, Recharge (RCH) Package, General-Head Boundary (GHB) Package, River (RIV) Package, Drain (DRN) Package, and Evapotranspiration (EVT) Package. Input instructions are contained in Chapter 8. Programming is documented in Chapter 9.

In Chapter 2, the finite-difference flow equation for MODFLOW was developed (eq. 2–26):

$$\begin{aligned} &CV_{i,j,k-1/2}\,h_{i,j,k-1} + CC_{i-1/2,j,k}\,h_{i-1,j,k} + CR_{i,j-1/2,k}\,h_{i,j-1,k} \\ &+ (-CV_{i,j,k-1/2} - CC_{i-1/2,j,k} - CR_{i,j-1/2,k} - CR_{i,j+1/2,k} - CC_{i+1/2,j,k} - CV_{i,j,k+1/2} + HCOF_{i,j,k})\,h_{i,j,k} \\ &+ CR_{i,j+1/2,k}\,h_{i,j+1,k} + CC_{i+1/2,j,k}\,h_{i+1,j,k} + CV_{i,j,k+1/2}\,h_{i,j,k+1} = RHS_{i,j,k} \quad . \end{aligned} \tag{6–1}$$

The flow equation was formulated so that inflows are added to the left side, with outflows represented as negative inflows. Stresses are incorporated to the flow equation by adding terms to HCOF and RHS. A stress term that is a coefficient of head, $h_{i,j,k}$, is added to HCOF. A stress term that is a constant is subtracted from RHS because RHS has been moved to the right side of the flow equation.

Well Package

The Well (WEL) Package is designed to simulate features such as wells that withdraw water from or add water to the aquifer at a constant rate during a stress period, where the rate is independent of both the cell area and the head in the cell. The discussion in this section is developed on the assumption that the features to be simulated are actually wells, either discharging or recharging; however, the package can be used to simulate any features for which the recharge or discharge can be directly specified.

The flow rate, Q, for a well is specified by the user as a fluid volume per unit time at which water is added to the aquifer. Negative values of Q are used to indicate well discharge (pumping), whereas positive values of Q indicate a recharging well. Four data values are required for each well—the row, column, and layer number of the cell in which the well is located, and the recharge rate, Q, of the well. The wells are redefined each stress period.

At each iteration, as the matrix equations are formulated, the value of Q for each well is subtracted from the RHS value (eq. 6–1) for the cell containing that well. Where more than one well falls within a single cell, the calculation is repeated for each well as the RHS term for that cell is assembled. Thus the user specifies the recharge associated with each individual well, and these are in effect summed within the program to obtain the total recharge to the cell.

The WEL Package does not directly accommodate wells that are open to more than one layer of the model. A well of this type, however, can be represented as a group of single-cell wells, each open to one of the layers tapped by the multilayer well, and each having an individual Q term specified for each stress period. If this approach is used, the recharge (negative for discharge) of the multilayer well must be divided or apportioned in some way among the individual layers, externally to the model program. A common method of doing this is to divide the well discharge (negative for pumping) in proportion to the layer transmissivities:

$$\frac{Q_n}{Q_W} = \frac{T_n}{\sum T} \quad , \tag{6–2}$$

where

Q_n is the recharge (negative for pumping) from layer n to a particular well in a given stress period (L^3T^{-1}),

Q_W is the total recharge (negative for pumping) for the well in that stress period (L^3T^{-1}),

T_n is the transmissivity of layer n (L^2T^{-1}), and

$\sum T$ represents the sum of the transmissivities of all layers penetrated by the well (L^2T^{-1}).

IMPORTANT: Equation 6–2, or some other method of apportioning the recharge, must be implemented by the user externally to the program for each multilayer well, and for each stress period. This approach, in which a multilayer well is represented as a group of single layer wells, fails to take into account the interconnection between various layers provided by the well itself, and is thus an incomplete representation of the problem. The Multi-Node Well Package has been created to directly simulate multilayer wells (Halford and Hanson, 2002).

Recharge Package

The Recharge (RCH) Package is designed to simulate areally distributed recharge to the ground-water system. Most commonly, areal recharge occurs as a result of precipitation that percolates to the ground-water system. Recharge applied to the model is defined as

$$QR_{i,j} = I_{i,j}DELR_jDELC_i \qquad (6–3)$$

where

$QR_{i,j}$ is the recharge flow rate applied to the model at horizontal cell location (i,j) expressed as a fluid volume per unit time (L^3T^{-1}); and

$I_{i,j}$ is the recharge flux (in units of length per time, LT^{-1}) applicable to the map area, $DELR_jDELC_i$, of the cell.

Values of recharge flux, $I_{i,j}$, are specified by the user at each stress period. These values of recharge flux are multiplied by horizontal cell areas, $DELR_jDELC_i$, to obtain values of $QR_{i,j}$. The recharge, $QR_{i,j}$, is applied to a single cell within the vertical column of cells located at (i,j). The RCH Package does not allow for recharge to occur simultaneously at multiple depths in the same vertical column because natural recharge enters the ground-water system at the top. In the simplest situation, the top of the ground-water system will occur in model layer 1; however, the vertical position of the top of the system may vary with horizontal location and with time as the water-table rises and falls. Three options for specifying the cell in each vertical column of cells that receives the recharge have been implemented as described below.

The cell within each vertical column to which the recharge is applied is specified through the recharge option code, NRCHOP, and optional two-dimensional variable IRCH. The options include: (1) application of the recharge to model layer 1; (2) application of the recharge to any cell in the vertical column as specified by layer numbers contained in two-dimensional variable $IRCH_{i,j}$; and (3) application of the recharge to the uppermost variable-head cell in the vertical column, provided no constant-head cell is above the variable-head cell in the column. Under options 1 and 2, if a cell designated to receive recharge is no flow or constant head, then no recharge is added. Under the third option, if a constant-head cell is in a vertical column of cells and no variable-head cell is above, then no recharge is applied to this column because the assumption is that any recharge would be intercepted by the constant-head source. Recharge flux values that are read into the model must be expressed in units that are consistent with the length and time units used to represent all other model data.

In the formation of the matrix equations, the recharge flow rate, $QR_{i,j}$, associated with a given horizontal cell location (i,j) and vertical location, k, that is determined by the recharge option is subtracted from the value of $RHS_{i,j,k}$ (eq. 6–1). This is done at each iteration for all cells that receive recharge. Because recharge, as defined, is independent of aquifer head, nothing is added to the coefficient of head, $HCOF_{i,j,k}$.

Careful consideration should be given to the problem under study and to the other options employed in the simulation before deciding which of the three recharge options to use in a given situation. For example, figure 6–1 shows a situation in which a cross-section model has been used to simulate a hypothetical problem involving

recharge, seepage from a river, and seepage into a river (fig. 6–1A). The river is simulated by constant-head cells. Using the provision described in Chapter 5 for horizontal conductance formulation under water-table conditions, cells for which the computed head was lower than the bottom elevation were converted to no flow so that the uppermost variable-head cell in each vertical column contains the water table. This process yields the final distribution of variable-head, constant-head, and no-flow cells shown in figure 6–1B.

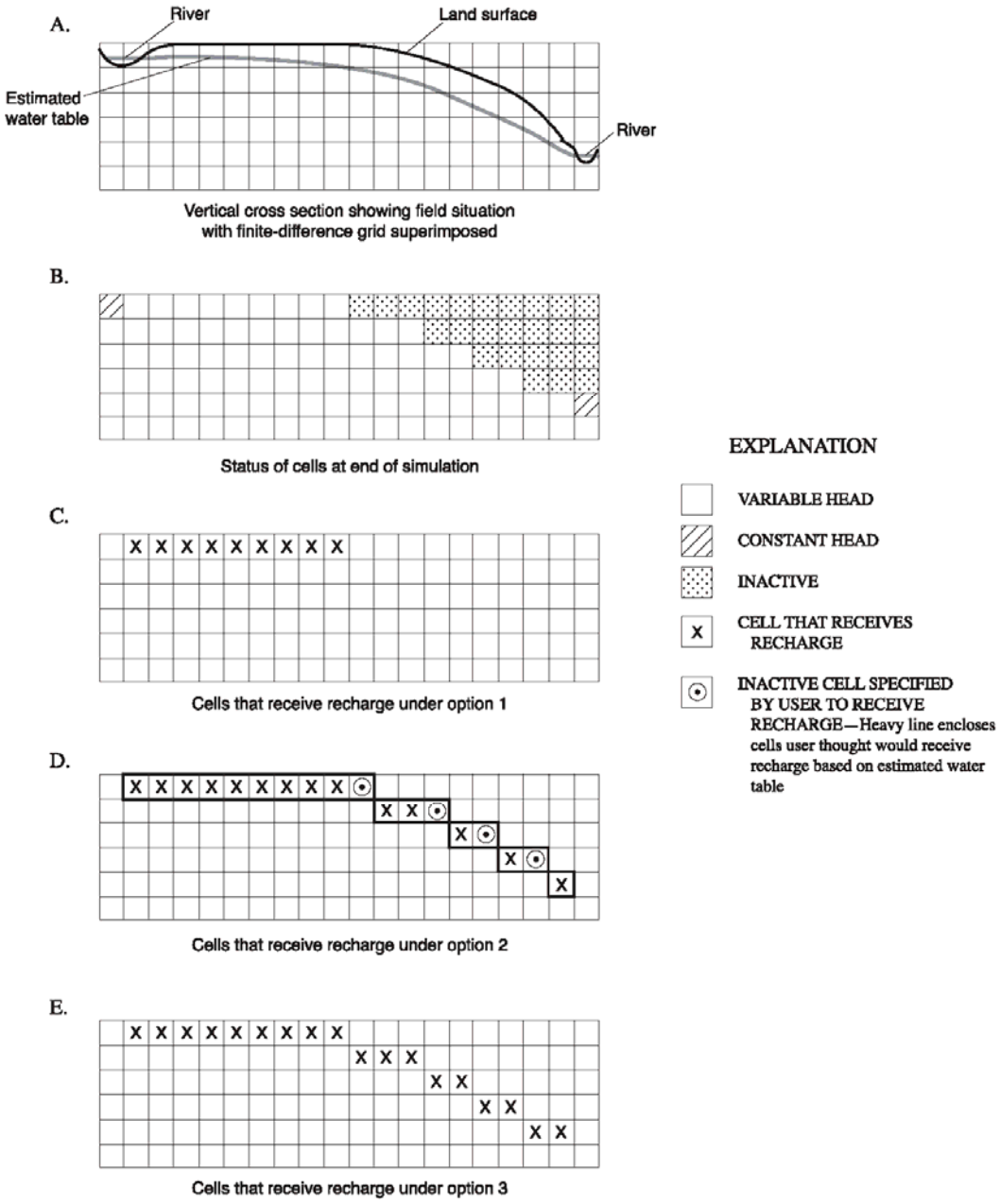


Figure 6–1. Hypothetical problem showing which cells receive recharge under the three options available in the Recharge Package. (From McDonald and Harbaugh, 1988.)

Figure 6–1C illustrates the recharge distribution to the model if option 1 above is utilized. Under this option, recharge is permitted only to the top layer of the model. Thus, once the water-table shape has been simulated by the

use of no-flow cells in the top layer, recharge to the vertical columns beneath those cells is shut off. This clearly fails to simulate the given system.

Figure 6–1D illustrates the recharge distribution if option 2 is used, assuming that the user specifies recharge cells prior to the simulation on the basis of an estimated water-table position, which differs slightly from the computed water table obtained in the simulation process. Four of the cells that the user had designated as recharge cells have converted to a no-flow condition and thus receive no recharge.

Figure 6–1E illustrates the recharge distribution under option 3, which turns out to be the option best suited for this particular situation. Under this option, recharge enters the uppermost variable-head cell in each vertical column, except where constant-head cells have been used to represent the river. Thus, a continuous distribution of recharge to the water table is simulated.

For the typical situation of recharge from precipitation, option 3 is the easiest to use. The model user does not have to be concerned about determining which is the highest variable-head cell in a vertical column because the program automatically determines this throughout the simulation. Option 1, however, can be useful in situations where recharge should not pass through the no-flow cells in layer 1. For example, some cells may be designated no-flow because they are impermeable. Any recharge specified for those cells should not pass into layer 2. Alternately, option 3 could still be used in this situation by specifying that the recharge rate is zero at the impermeable cells. Similarly, option 2 may be useful when layers other than layer 1 have outcrop areas and when recharge to the specified layers should not penetrate through no-flow cells to a lower layer.

The RCH Package can be used to simulate recharge from sources other than precipitation—for example, artificial recharge. Discharge can also be simulated using the RCH Package by specifying negative values of the recharge flux. If the ability to apply recharge to more than one cell in a vertical column of cells is required, then the Well Package, which allows recharge or discharge to be specified at any model cell, can be used.

General-Head Boundary Package

The function of the General-Head Boundary (GHB) Package is to simulate flow into or out of a cell i,j,k, from an external source in proportion to the difference between the head in the cell and the head assigned to the external source. The constant of proportionality is called the boundary conductance. Thus a linear relation between flow into the cell and head in the cell is established:

$$QB_n = CB_n\left(HB_n - h_{i,j,k}\right), \tag{6–4}$$

where

- n is a boundary number,
- QB_n is the flow into cell i,j,k from the boundary (L^3T^{-1});
- CB_n is the boundary conductance (L^2T^{-1});
- HB_n is the head assigned to the external source (L); and
- $h_{i,j,k}$ is the head in cell i,j,k (L).

The relation between cell i,j,k and the external source is shown schematically in figure 6–2. The constant-head source is represented by the apparatus on the right in figure 6–2, which holds the source head at the level HB_n regardless of other factors; the link between the source and cell i,j,k is represented by the block of porous material having conductance CB_n. Note that figure 6–2 shows no mechanism to limit flow in either direction as $h_{i,j,k}$ rises or falls.

A graph of inflow from a general-head boundary and head in the cell containing the boundary as given by equation 6–4 is shown in figure 6–3. The GHB Package provides no limiting value of flow to bound the linear function in either direction; and as the difference between the head in the cell containing the boundary and the source head increases, flow into or out of the cell continues to increase without limit. Accordingly, the GHB Package must be used with care to insure that unrealistic flows into or out of the system do not develop during the course of simulation. The GHB Package is included in MODFLOW without a specific real-world conceptualization in mind. The River, Drain, and Evapotranspiration Packages documented in the following sections of this chapter incorporate

the same head-dependent form, except that they limit the flow to simulate the conceptualization of specific hydrologic features.

QB_n of equation 6–4 is defined as an inflow to the aquifer, and according to the flow convention used in MODFLOW, inflows are added to the left side of equation 6–1. In terms of the variables HCOF and RHS, the term $-CB_n$ is added to $HCOF_{i,j,k}$ and the term CB_nHB_n is subtracted from $RHS_{i,j,k}$ as the matrix equations are assembled.

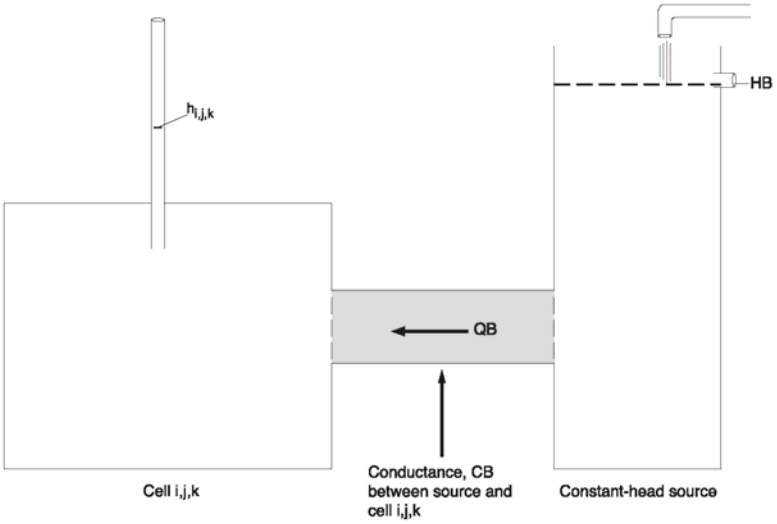


Figure 6–2. Schematic diagram illustrating principle of General-Head Boundary Package. (Modified from McDonald and Harbaugh, 1988.)

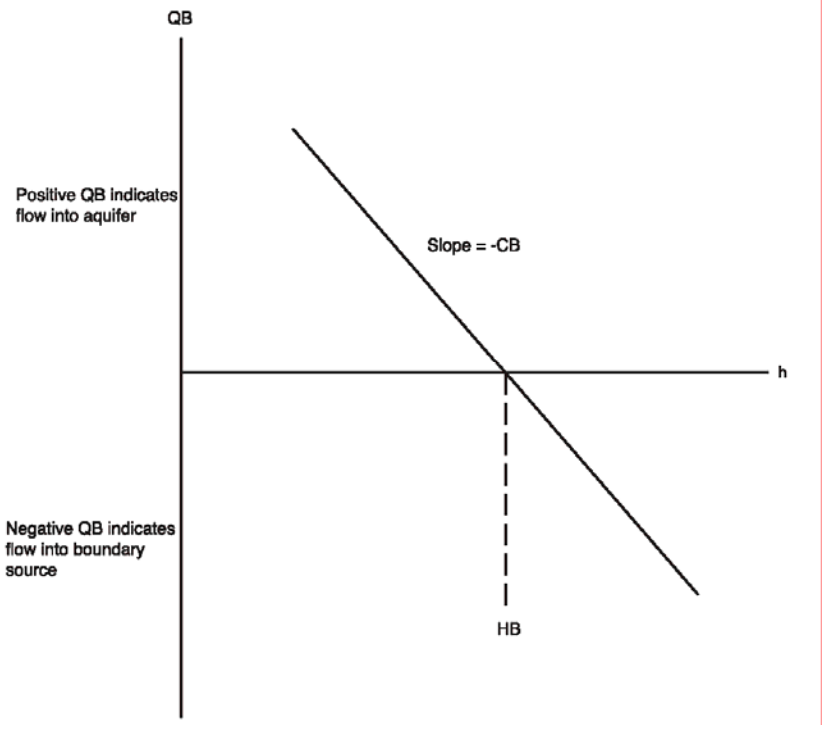


Figure 6–3. Plot of flow, QB, from a general-head boundary source into a cell as a function of head, h, in the cell where HB is the source head. (Modified from McDonald and Harbaugh, 1988.

River Package

Rivers and streams contribute water to or drain water from the ground-water system, depending on the head gradient between the river and the ground-water regime. The purpose of the River (RIV) Package is to simulate the effects of flow between surface-water features and ground-water systems. To accomplish this purpose, terms representing seepage to or from the surface features must be added to the ground-water flow equation for each cell affected by the seepage.

Figure 6–4 shows two rivers divided into reaches so that each reach is completely contained in a single cell. River-aquifer seepage is simulated between each reach and the model cell that contains that reach. RIV does not simulate surface-water flow in the river – only the river/aquifer seepage. [Other packages that also simulate surface-water flow have been developed for MODFLOW (for example, Prudic, 1989, and Prudic, Konikow, and Banta, 2004).] Accordingly, the order of numbering for reaches has no impact on calculations in the RIV Package. River seepage is independently simulated for each river reach.

The cross section in figure 6–5 shows a single cell containing a river reach. In the conceptualized system (fig. 6–5A), the open water of a river is separated from the ground-water system by a layer of low permeability riverbed material. Figure 6–5B shows an idealization of this system in which the river-aquifer interconnection is represented as a simple conductance through which one-dimensional flow occurs.

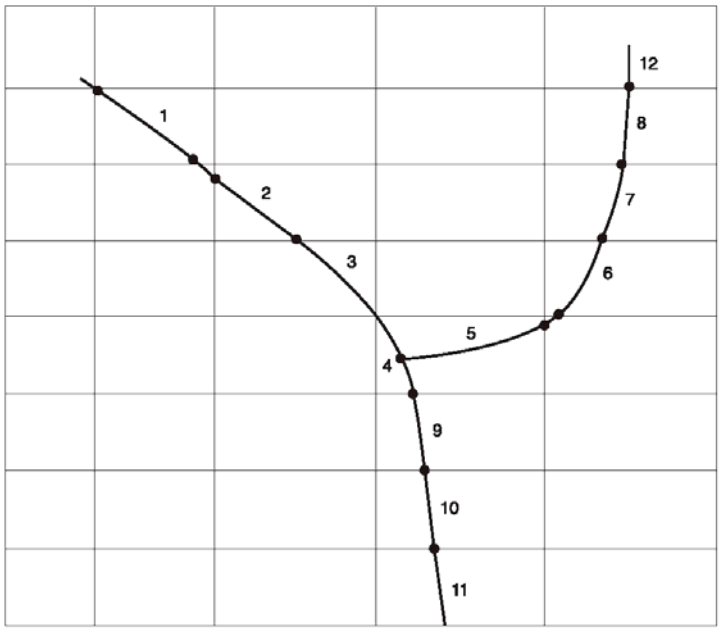


Figure 6–4. Discretization of two rivers into reaches. Some small reaches are ignored. (Modified from McDonald and Harbaugh, 1988.

The assumption is made that measurable head losses between the river and the aquifer are limited to those across the riverbed layer itself—that is, that no substantial head loss occurs between the bottom of the riverbed layer and the point represented by the underlying model node. Further, an assumption is made that the underlying model cell remains fully saturated—that is, the water level does not drop below the bottom of the riverbed layer. Under these assumptions, flow between the river and the ground-water system for reach n is given by

$$QRIV_n = CRIV_n\left(HRIV_n - h_{i,j,k}\right) \quad (6–5)$$

where

$QRIV_n$ is the flow between the river and the aquifer, taken as positive if it is directed into the aquifer (L^3T^{-1});

$HRIV_n$ is the water level (stage) in the river (L);

$CRIV_n$ is the hydraulic conductance of the river-aquifer interconnection (L^2T^{-1}); and

$h_{i,j,k}$ is the head at the node in the cell underlying the river reach (L).

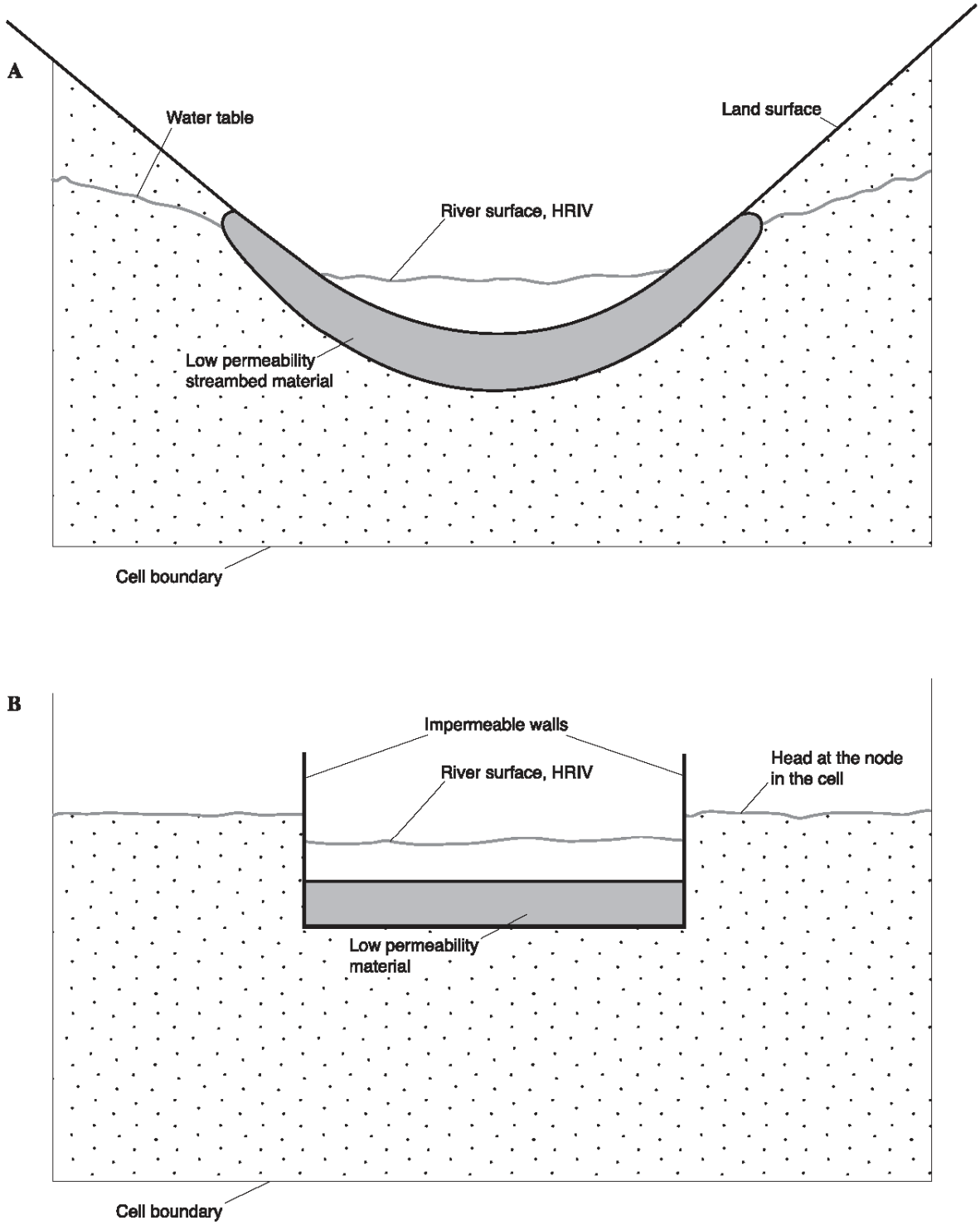


Figure 6–5. (A) Cross section of an aquifer containing a river and (B) conceptual representation of river-aquifer interconnection in a simulation. (From McDonald and Harbaugh, 1988.)

Figure 6–6 shows an isolated view of the idealized riverbed conductance of figure 6–5B, as it crosses an individual cell. The length (L_n) of the conductance block is the length of the river as it crosses the node; the width (W_n) is the river width; the distance of flow is taken as the thickness (M_n) of the riverbed layer; and K_n is the hydraulic conductivity of the riverbed material. The conductance can then be computed as

$$CRIV_n = \frac{K_n L_n W_n}{M_n} \quad . \tag{6–6}$$

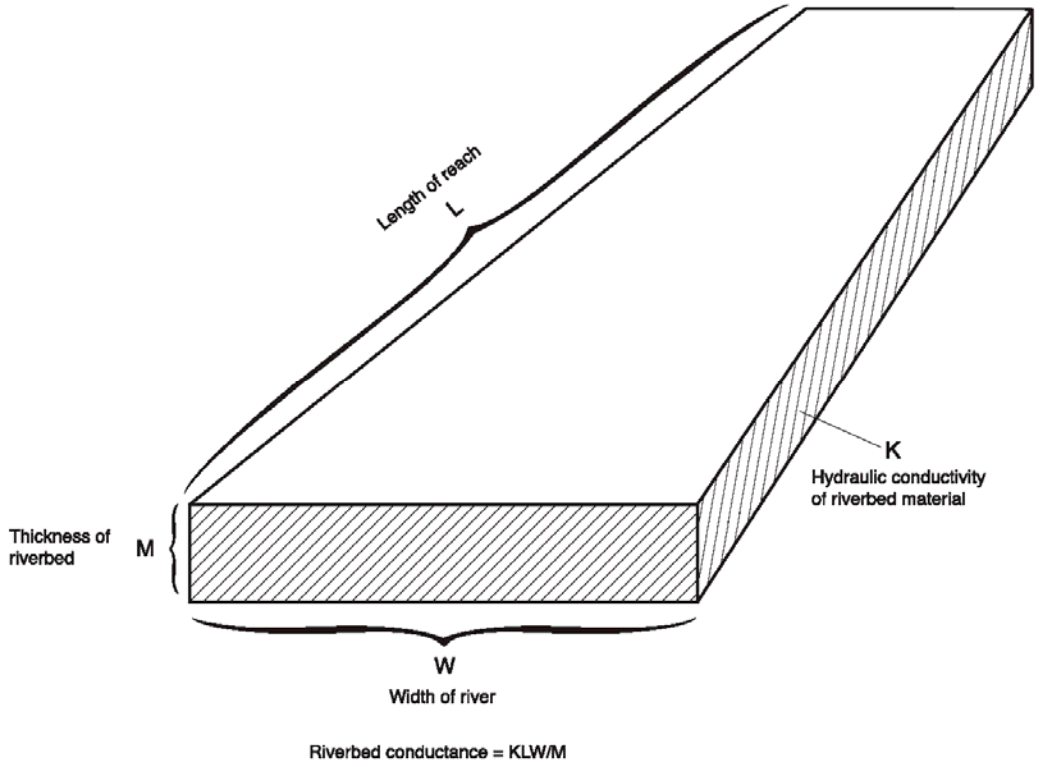


Figure 6–6. Idealization of riverbed conductance in an individual cell. (From McDonald and Harbaugh, 1988.)

Equation 6–5 normally provides an acceptable approximation of river-aquifer interaction over a certain range of aquifer head values. In most cases, however, if water levels in the aquifer fall below a certain point, seepage from the river ceases to depend on head in the aquifer. This can be visualized by returning to the concept of a discrete riverbed layer. Figure 6–7A shows the situation described by equation 6–5; the water level in the aquifer is above the bottom of the riverbed layer, and flow through that layer is proportional to the head difference between the river and the aquifer. In figure 6–7B, the water level in the aquifer has fallen below the bottom of the riverbed layer, leaving an unsaturated interval beneath that layer; if it is assumed that the riverbed layer itself remains saturated, the head at its base will simply be the elevation at that point. If this elevation is designated $RBOT_n$, the flow through the riverbed layer is given by

$$QRIV_n = CRIV_n\left(HRIV_n - RBOT_n\right) \tag{6–7}$$

where $QRIV_n$, $CRIV_n$, and $HRIV_n$ are as defined for equations 6–5 and 6–6. Obviously, further declines in head below $RBOT_n$ produce no increase in flow through the riverbed layer; the flow simply retains the constant value given by equation 6–7, as long as head remains below $RBOT_n$. MODFLOW uses these concepts in simulating river-aquifer interaction—that is, flow between a river and a node i,j,k is simulated according to the equation set

$$QRIV_n = CRIV_n\left(HRIV_n - h_{i,j,k}\right), \qquad h_{i,j,k} > RBOT_n \tag{6–8A}$$

$$QRIV_n = CRIV_n\left(HRIV_n - RBOT_n\right), \qquad h_{i,j,k} \le RBOT_n\,. \tag{6–8B}$$

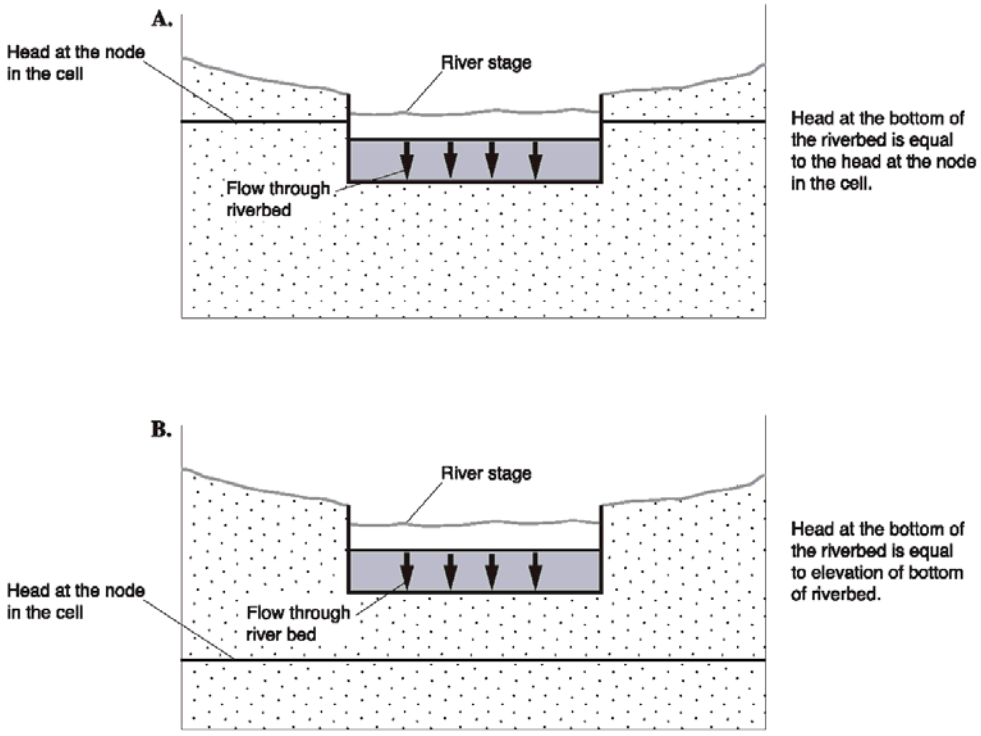


Figure 6–7. Cross sections showing the relation between head at the bottom of the riverbed layer and head in the cell. Head in the cell is equal to the water-table elevation. (Modified from McDonald and Harbaugh, 1988.)

Figure 6–8 shows a graph of flow from a river reach as a function of the head, h, in the cell containing the reach, as calculated using equation 6–8. Flow is zero when h is equal to the water level in the river, HRIV. For values of h higher than HRIV, flow is into the river, represented as a negative inflow to the aquifer. For values of h lower than HRIV, flow is positive into the aquifer. This positive flow increases linearly as h decreases, until h reaches RBOT; thereafter the flow remains constant. Thus, the RIV Package is mathematically identical to the GHB Package as long as h is greater than RBOT.

The conceptualization of river-aquifer interaction used here assumes that this interaction is independent of the location of the river reach within the cell, and that the level of water in the river is uniform over the reach and constant over each stress period. The latter assumption implies that conditions of flow in the river do not vary substantially during the stress period—for example, the river does not go dry or overflow its banks, or such events are of such short duration as to have no effect on river-aquifer interaction.

If the assumption is satisfied that all substantial head loss occurs across a discrete, rectangular riverbed layer, the application of equations 6–6 and 6–8 is straightforward. The idealized riverbed conductance can be further generalized to include a variable width and a meandering river segment. In this case, the length would be the total length of the river reach in the cell, and the width would be the average width along the length. If reliable field measurements of river seepage and associated head difference are available, they may be used to calculate an effective conductance. Otherwise, a conductance value must be chosen more or less arbitrarily and adjusted during model calibration.

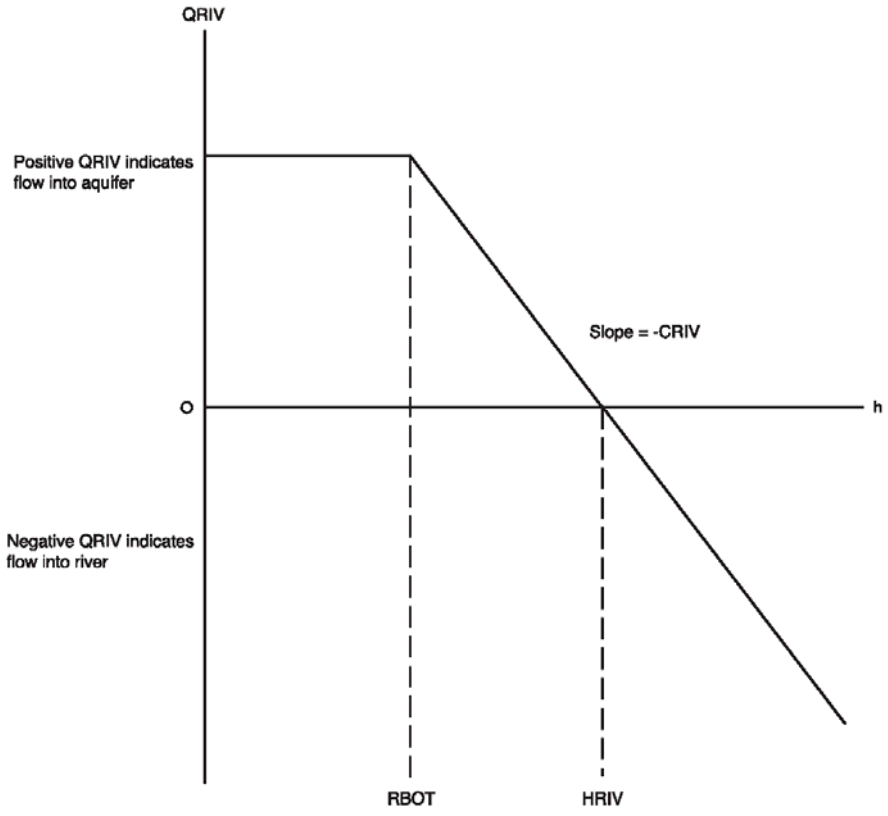


Figure 6–8. Plot of flow, QRIV, from a river into a cell as a function of head, h, in the cell where RBOT is the elevation of the bottom of the riverbed and HRIV is the head in the river. (From McDonald and Harbaugh, 1988.)

The flow between the river and aquifer is in general a three-dimensional process, and its representation through a single conductance term and riverbed elevation is only approximate in situations in which the riverbed is much different from the idealized confining layer. For example, the bed may be discontinuous, multiple zones of low conductivity may be below the river, or perhaps no discrete riverbed layer can be identified. The user should recognize that formulation of a single conductance term to account for the resulting three-dimensional flow process is inherently an empirical exercise, and that adjustment during calibration is almost always required. Certain rules can be formulated, however, to guide the initial choice of conductance in these situations. For example, the area through which flow occurs should still be viewed as the product of the total length of the river reach in the cell and the average width; the assumed distance of flow should not exceed the vertical interval between the riverbed and node i,j,k; and, if distinct layers can be recognized within this interval, these layers should normally be treated as conductances in series in formulating an equivalent conductance.

The application of equation 6–8 is the most difficult in situations where a discrete riverbed does not exist. In this case, equation 6–8 should be applied as an approximation. The following is a simple justification for applying the general relation of equation 6–8 in cases of no distinct riverbed. Although a strictly linear relation between head and seepage may not exist, the seepage into the river will increase as aquifer head increases above the river stage. Likewise, when the river is losing, seepage from the river will increase as head drops below the river stage. Thus, choosing a value for CRIV that will approximate the seepage as represented by equation 6–8A should be possible, although the value of CRIV should probably vary with aquifer head. The existence of a depth below the river for which a further decrease in water level will not cause additional seepage also seems reasonable, and thus equation 6–8B would apply. This situation is conceptualized in figure 6–9, where the water table has fallen substantially below the river and only a narrow saturated connection exists between the river and the regional water table. When the water table drops, as shown in figure 6–9, a region in the saturated column will form in which the head gradient above the

water table is nearly 1. Further lowering the water table will not increase the gradient, and therefore the seepage should not increase. A break in saturation also will occur if the water table lowers enough, but leakage will not substantially increase once the head drops low enough to cause the region of unit gradient.

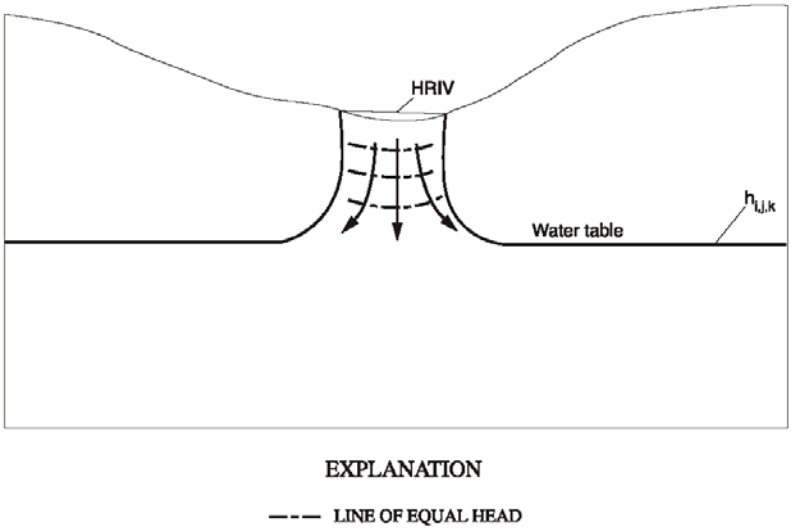


Figure 6–9. Limiting seepage from a river at unit hydraulic gradient. (Modified from McDonald and Harbaugh, 1988.)

Unfortunately, the above argument that equation 6–8 can be qualitatively applied in the situation of no distinct riverbed does not provide much guidance on how to determine appropriate values for $CRIV_n$ and $RBOT_n$. One approach is to assume that the maximum seepage from the stream is the seepage in the aquifer in a column of water in which unity head gradient occurs. Darcy's law can be applied to compute the vertical flow, Qmax, through the water column. If the water column is assumed to have the same horizontal dimensions as the stream channel, L_n and W_n., then the result is:

$$Qmax = Kaq\ L_n W_n\ dh/dl\ ,$$

where

Kaq is aquifer hydraulic conductivity (LT^{-1}), and

dh/dl is the head gradient through the water column (dimensionless).

The head gradient is 1, so $Qmax=KaqL_nW_n$.

Equation 6–5 can then be used to compute the conductance that will produce Qmax when aquifer head has a specified value of $h_{i,j,k}$. If $h_{i,j,k}$ is replaced by $RBOT_n$, then $CRIV_n$ is:

$$CRIV_n = \frac{Kaq\ L_n W_n}{HRIV_n - RBOT_n} \qquad (6–9)$$

In this situation, rather than being a distinct riverbed-bottom elevation, $RBOT_n$ is the value of aquifer head at which the seepage from the river will equal Qmax. Seepage will not increase if the aquifer head goes lower. Regardless of the approach used to determine these values initially, some evaluation or adjustment is essential during the calibration process.

Data describing each river reach are specified by the user for each stress period. Input consists of six entries for each river reach: the layer, row, and column of the cell containing the reach, and the three values needed to calculate seepage—river level or stage ($HRIV_n$), the conductance of the river-aquifer interconnection ($CRIV_n$), and the "elevation of the bottom of the riverbed," or level at which the limiting value of river seepage is attained ($RBOT_n$).

At the start of each iteration, terms representing river seepage are added to the flow equation for each cell containing a river reach. The choice of which river seepage equation to use, equation 6–8A or equation 6–8B, is made by comparing the most recent value of head at the cell to the value of $RBOT_n$ for the reach. Because this process is done at the start of each iteration, the most current value of head ($h_{i,j,k}$) is the value from the previous iteration. Thus, the check for which river seepage equation to use lags behind the seepage calculations by one iteration. If equation 6–8A is selected, then the term $-CRIV_n$ is added to the term $HCOF_{i,j,k}$ and the term $CRIV_n*HRIV_n$ is subtracted from $RHS_{i,j,k}$. If equation 6–8B is selected, then the term $CRIV_n(HRIV_n - RBOT_n)$ is subtracted from $RHS_{i,j,k}$.

Drain Package

The Drain (DRN) Package is designed to simulate the effects of features such as agricultural drains, which remove water from the aquifer at a rate proportional to the difference between the head in the aquifer and some fixed head or elevation, called the drain elevation, so long as the head in the aquifer is above that elevation. If, however, the aquifer head falls below the drain elevation, then the drain has no effect on the aquifer. The constant of proportionality is called the drain conductance. A mathematical statement of this situation is:

$$Qout = CD\left(h_{i,j,k} - HD\right), \qquad h_{i,j,k} > HD$$

$$Qout = 0, \qquad h_{i,j,k} \le HD \qquad (6\text{–}10)$$

where

Qout is the flow from the aquifer into the drain (L^3T^{-1}),
CD is the drain conductance (L^2T^{-1}),
HD is the drain elevation (L), and
$h_{i,j,k}$ is the head in the cell containing the drain (L).

Equation 6–10 is rewritten in terms of flow from the drain into the aquifer, QD, which is the flow convention used throughout this report:

$$QD_n = CD_n\left(HD_n - h_{i,j,k}\right), \qquad h_{i,j,k} > HD_n \qquad (6\text{–}11A)$$

$$QD_n = 0, \qquad h_{i,j,k} \le HD_n \qquad (6\text{–}11B)$$

where the subscript n is a drain number.

Thus, from the perspective of inflow to the model, drain flow is either negative or zero. For purposes of simulation, an assumption is made that each model drain represents the part of a physical drain that overlies a single model cell. Figure 6–10 shows a graph of flow from a drain and head in the cell containing the drain as defined by equation 6–11; the function is similar to that for flow between a surface river and the aquifer (fig. 6–8) except that flow into the aquifer is excluded. With proper selection of coefficients, the River Package could in fact be used to perform the functions of the Drain Package.

Many physical conceptualizations can be approximated by equation 6–11. Figure 6–11A shows one conceptualization—a three-dimensional view of a buried drain tile (pipe containing slots) as used for agricultural drains. To justify representation by equation 6–11, the slope of the pipe is assumed to be great enough that once water enters the slots, it is carried away without filling the pipe. Accordingly, the drain runs only partially full, and the head representing the drain, HD_n, is the average elevation of the slots.

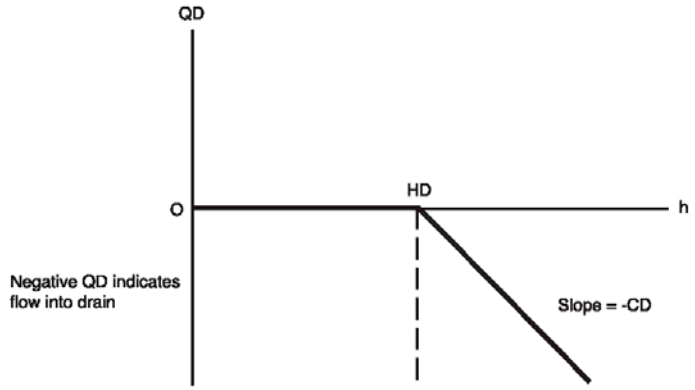


Figure 6–10. Plot of flow, QD, into a drain as a function of head, h, in a cell where elevation of the drain is HD and the conductance is CD. (Modified from McDonald and Harbaugh, 1988.

Figure 6–11B shows a cross section of cell i,j,k that is traversed by the drain. The head computed by the model for cell i,j,k ($h_{i,j,k}$) is actually an average value for the cell, and is normally assumed to prevail at some distance from the drain itself. HD_n prevails only locally at the drain pipe—it does not characterize the cell as a whole. Between the drain and the area in which head $h_{i,j,k}$ prevails, a radial or semiradial flow pattern exists in the vertical plane, normally characterized by progressively steeper head gradients as the drain is approached. The head loss within this converging flow pattern forms one part of the head difference $h_{i,j,k}$-HD_n. An additional component of head loss may occur in the immediate vicinity of the drain if the hydraulic conductivity in that region differs from the average value used for cell i,j,k owing to the presence of foreign material around the drain pipe. Finally, head losses occur through the wall of a drain pipe, depending upon the number and size of the openings in the pipe, and the degree to which those openings may be blocked by chemical precipitates, plant roots, or other obstructions.

The three processes discussed above—convergent flow toward the drain, flow through material of differing conductivity in the vicinity of the drain, and flow through the wall of the drain—each generate head losses that may be assumed proportional to the discharge, QD, into the drain from cell i,j,k. Because these head losses occur in series, the total head loss HD_n-$h_{i,j,k}$ also may be taken as proportional to QD_n. This has been done in the method of simulation embodied in the Drain Package. That is, the drain function is assumed to be described by equation 6–11. The coefficient CD_n of equation 6–11 is a lumped (or equivalent) conductance describing all of the head loss between the drain and the region of cell i,j,k in which the head $h_{i,j,k}$ can be assumed to prevail. Thus, CD_n depends on the characteristics of the convergent flow pattern toward the drain, as well as on the characteristics of the drain itself and its immediate environment.

One could attempt to calculate values for CD_n by developing approximate equations for conductance for the three flow processes, and then calculate the equivalent series conductance. The conductance for each process would be based on the formulation of a one-dimensional flow equation. The formulations vary substantially depending on the specific drain system being simulated, so no general formulation for calculating CD_n is presented here. Also, in most situations a specific formulation would require detailed information that is not usually available, such as detailed head distribution around the drain, aquifer hydraulic conductivity near the drain, distribution of the fill material, hydraulic conductivity of fill material, number and size of the drain pipe openings, the amount of clogging materials, and the hydraulic conductivity of the clogging materials. In practice, calculating CD_n from measured values of QD_n and HD_n–$h_{i,j,k}$ using equation 6–11A is more common. If HD_n–$h_{i,j,k}$ is not accurately known, CD_n is usually adjusted during model calibration in order to match measured values of QD_n to model calculated values.

Figure 6–12 shows two other real-world conceptualizations that can sometimes be simulated using equation 6–11. Figure 6–12A shows an open channel, which could be natural or man made. If the channel carries water even when aquifer head is below the channel bed, then equation 6–11 presumably would not always apply because water could seep into the aquifer. In this case, the channel would be better simulated using the RIV Package as described earlier. But if the channel is dry unless ground water is seeping into it, and there is no chance for this water to seep back into the aquifer, then the Drain Package could provide a reasonable representation. For example, the head waters of a river might be acceptably simulated with DRN whereas the lower sections of a river might be simulated using RIV.

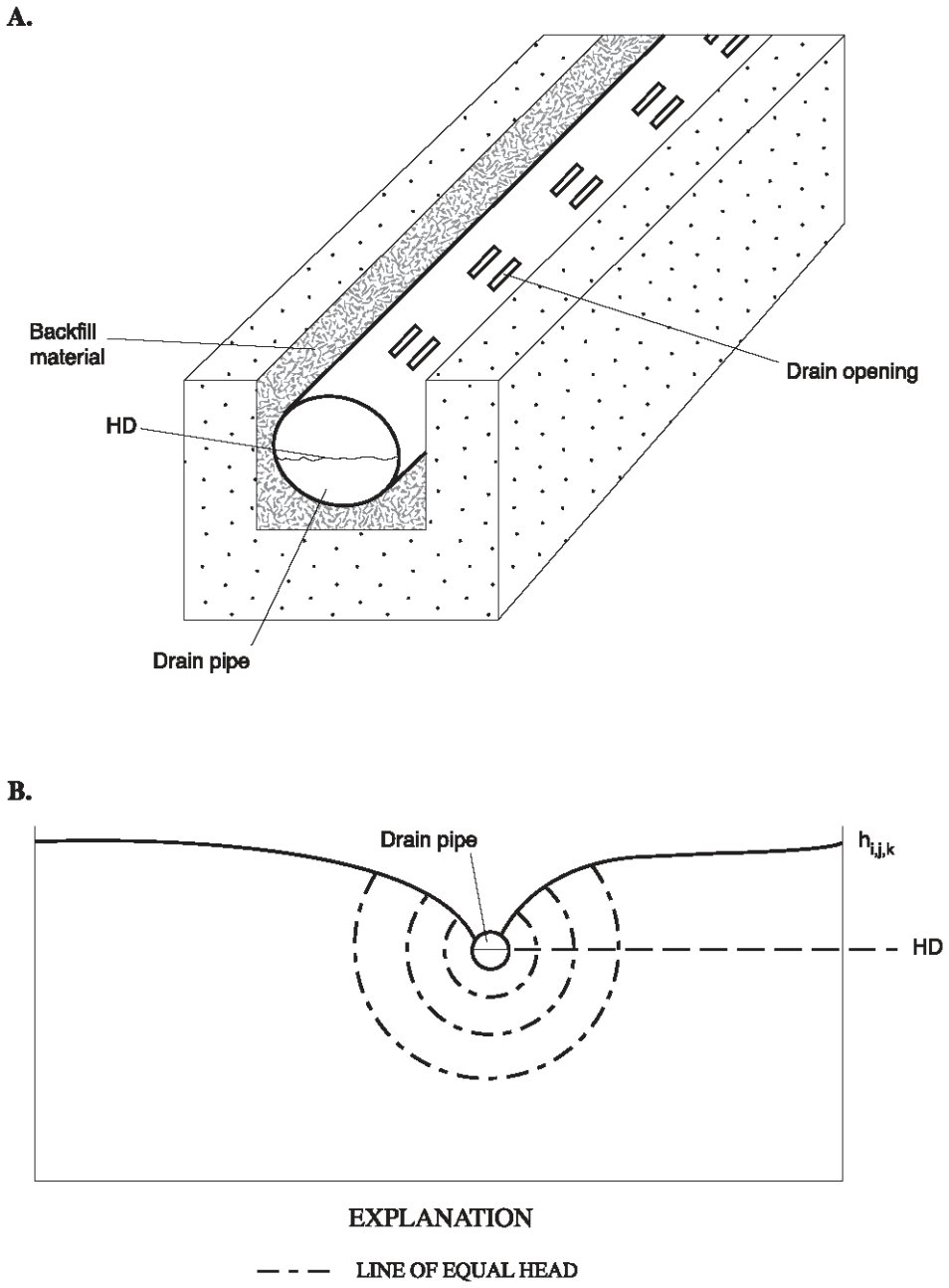


Figure 6–11. (A) Factors affecting head loss in the immediate vicinity around a buried drain pipe in a backfilled ditch, and (B) cross section through cell i,j,k illustrating head loss in convergent flow into drain. (Modified from McDonald and Harbaugh, 1988.)

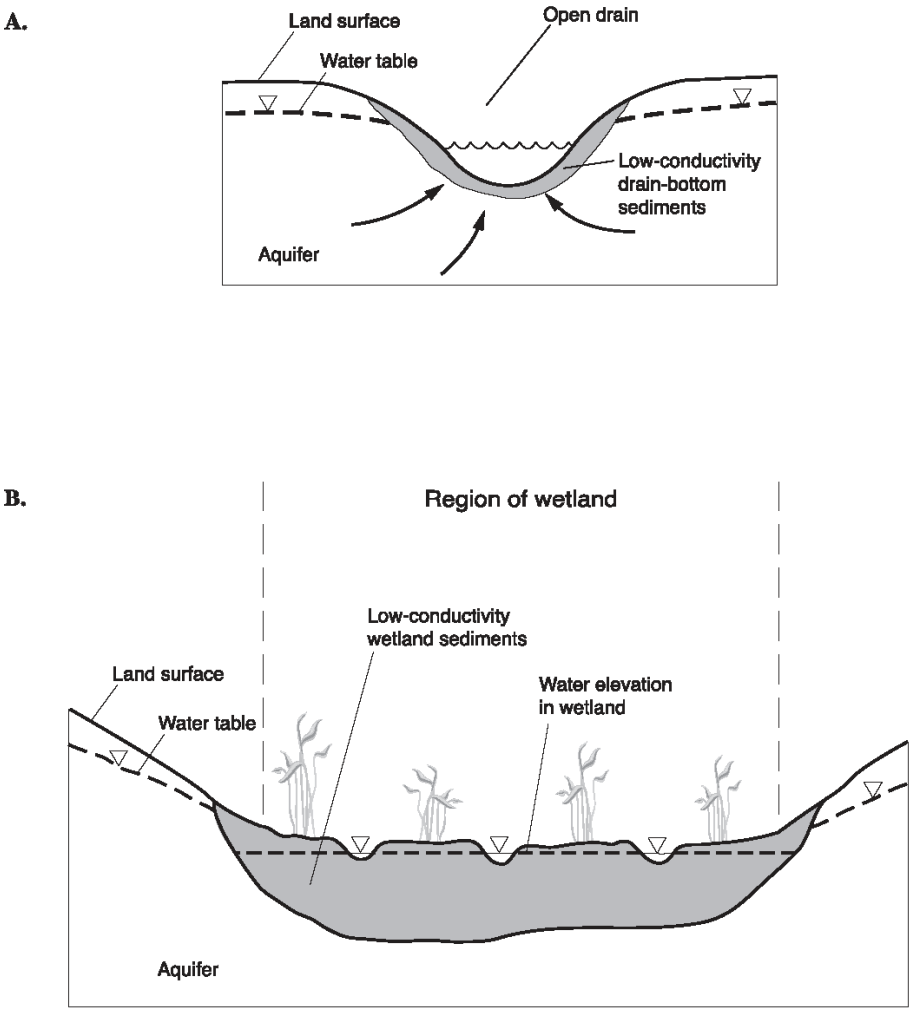


Figure 6–12. Alternate conceptualization of drains: (A) open drainage channel and (B) wetland.

Figure 6–12B shows a wetland, which can sometimes be conceptualized as functioning according to equation 6–11. If the wetland is wet primarily from ground-water seepage, then it will presumably dry up if the water table drops below the wetland. When wet, the conceptualized wetland water elevation is assumed to be fairly constant because excess ground-water seepage drains to a nearby river through a network of shallow channels or over the nearly flat surface. Seepage into the wetland is limited by low conductivity sediments in the wetland. Under these conditions, equation 6–11 would be a reasonable approximation. Although one physical wetland might cover many cells, the part covered by each cell would be viewed as a separate drain in MODFLOW. The drain conductance for each drain would depend on the area of the cell covered by the wetland and the properties of the lower conductivity sediments in the wetland.

According to the sign convention in MODFLOW, QD_n in equation 6–11 is defined as a flow into cell i,j,k and must be added to the left side of equation 6–1 for each cell affected by a drain, provided the head $h_{i,j,k}$ is above the drain elevation. This is accomplished in the Drain Package by testing to determine whether head exceeds drain elevation, and if so, by adding the term $-CD_n$ to $HCOF_{i,j,k}$ (equation 6–1) and subtracting the term CD_nHD_n from $RHS_{i,j,k}$, as the matrix equations are assembled.

Evapotranspiration Package

The Evapotranspiration (ET) Package simulates the effects of plant transpiration and direct evaporation in removing water from the saturated ground-water regime. The approach is based on the following assumptions: (1) when the water table is at or above a specified elevation, termed the "ET surface" in this report, evapotranspiration loss from the water table occurs at a maximum rate specified by the user; (2) when the depth of the water table below the ET surface elevation exceeds a specified interval, termed the "extinction depth" or "cutoff depth" in this report, evapotranspiration from the water table ceases; and (3) between these limits, evapotranspiration from the water table varies linearly with water-table elevation.

This can be expressed in equation form as

$$RET = EVTR, \qquad h_{i,j,k} > SURF \qquad (6–12A)$$

$$RET = EVTR\frac{h_{i,j,k} - (SURF - EXDP)}{EXDP}, \qquad (SURF\text{-}EXDP) \le h_{i,j,k} \le SURF \qquad (6–12B)$$

$$RET = 0, \qquad h_{i,j,k} < SURF\text{-}EXDP \qquad (6–12C)$$

where

RET is the rate of loss per unit surface area of water table due to evapotranspiration, in units of volume of water per unit area per unit time (LT^{-1});

$h_{i,j,k}$ is the head, or water-table elevation in the cell from which the evapotranspiration occurs (L);

EVTR is the maximum possible value of RET (LT^{-1});

SURF is the ET surface elevation, or the water-table elevation at which this maximum value of evapotranspiration loss occurs (L); and

EXDP is the cutoff or extinction depth (L), such that when the distance between $h_{i,j,k}$ and SURF exceeds EXDP, evapotranspiration ceases.

In implementing the finite-difference approach, the volumetric rate of evapotranspiration loss from a given cell is required. This is given as the product of the loss rate per unit area, and the horizontal surface area, $DELR_jDELC_i$ of the cell from which the loss occurs:

$$QET = RET(DELR_j DELC_i) \qquad (6–13)$$

where QET is the evapotranspiration loss, in volume of water per unit time (L^3T^{-1}), through the area $DELR_jDELC_i$.

If the maximum value of QET (corresponding to EVTR) is designated QETM, equation 6–12 can be expressed in terms of volumetric discharge as

$$QET = QETM \qquad h_{i,j,k} > SURF \qquad (6–14A)$$

$$QET = QETM\frac{h_{i,j,k} - (SURF - EXDP)}{EXDP} \qquad (SURF\text{-}EXDP) \le h_{i,j,k} \le SURF \qquad (6–14B)$$

$$QET = 0 \qquad h_{i,j,k} < SURF\text{-}EXDP \qquad (6–14C)$$

Figure 6–13 shows a graph of evapotranspiration loss, QET, from a cell and head in the cell based on equation 6–14. Comparison of the ET function with the river or drain functions shows that the three are mathematically similar, except that the linear part of the ET function is bounded at both ends by constant values, rather than only at one end. Note that QET (eq. 6–14 and figure 6–13) represents outflow from the ground-water system, which is counter to the convention in MODFLOW of representing stresses as inflow. QET was developed in terms of outflow because it can be conceptually confusing to view evapotranspiration as a negative quantity.

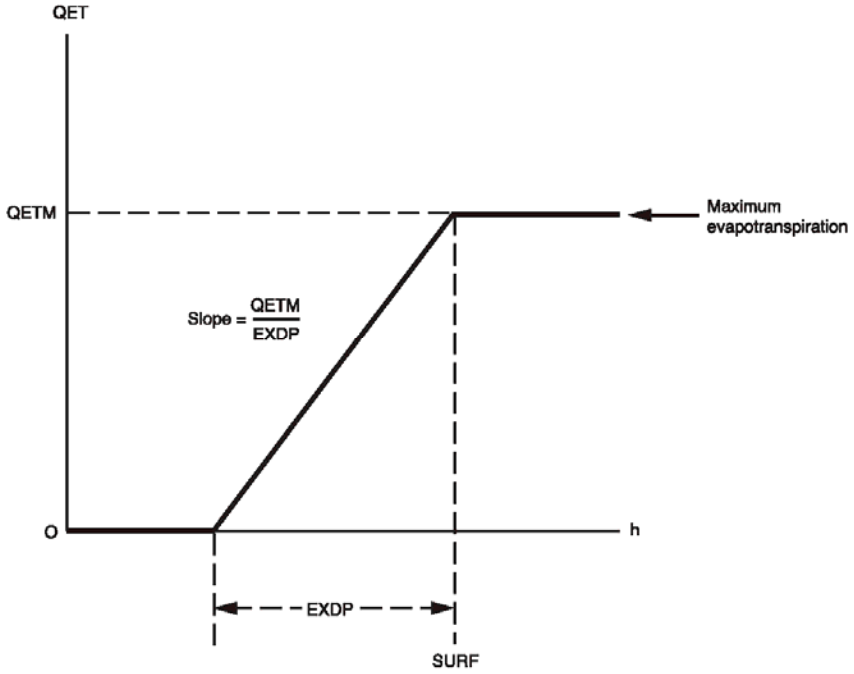


Figure 6–13. Plot of volumetric evapotranspiration, QET, as a function of head, h, in a cell where EXDP is the cutoff depth and SURF is the ET surface elevation. (Modified from McDonald and Harbaugh, 1988.)

Evapotranspiration is conceptualized as an areal phenomenon like recharge simulated by the Recharge Package. Evapotranspiration can be drawn from only one cell in each vertical column beneath the map area of each horizontal grid location. As with the Recharge Package, the user designates the layer for each horizontal location using one of three options. Under the first option, evapotranspiration is always drawn from the uppermost layer of the model; under the second option, the user specifies the cell within the vertical column at i,j from which the evapotranspiration is to be taken. In using either of these options, the computed evapotranspiration has no influence on the simulation if the designated cell is either a no-flow cell or a constant-head cell. Under the third option, evapotranspiration is taken from the uppermost variable-head cell in each vertical column, provided no constant-head cell is above it. A constant-head cell above a variable-head cell prevents evapotranspiration from occurring.

For each horizontal cell location, (i,j), and for each stress period (unless an option is exercised to use prior values) the ET package reads values of EVTR (maximum evapotranspiration loss per unit area per unit time), with dimensions LT^{-1}. These rates are immediately multiplied by cell areas, $(DELR_j)(DELC_i)$, to obtain the maximum volumetric rate of evapotranspiration from each cell. Likewise, values of SURF (the ET surface elevation or water-table elevation at which evapotranspiration is maximum) and EXDP (the cutoff depth or extinction depth) are read for each horizontal cell location for each stress period. If the second option is used, the evapotranspiration layer is read for each horizontal location in the grid.

Equation 6–14 is rewritten in terms of inflow to a model cell as is the convention in MODFLOW. Equation 6–14C is further rearranged algebraically for convenient implementation in the code. Also, i,j subscripts are added to indicate the variables that have a different value for each horizontal cell location:

$$QETin_{i,j,k} = -QETM_{i,j} \qquad \text{for } h_{i,j,k} > SURF_{i,j} \tag{6–15A}$$

$$QETin_{i,j,k} = \frac{-QETM_{i,j}}{EXDP_{i,j}} h_{i,j,k} - QETM_{i,j} + \frac{QETM_{i,j} SURF_{i,j}}{EXDP_{i,j}}$$

$$\text{for } (SURF_{i,j}\text{-}EXDP_{i,j}) \le h_{i,j,k} \le SURF_{i,j} \tag{6–15B}$$

$$QETin_{i,j,k} = 0 \qquad \text{for } h_{i,j,k} < SURF_{i,j}\text{-}EXDP_{i,j} \text{ .} \tag{6–15C}$$

In terms of the expressions HCOF and RHS of equation 6–1, evapotranspiration is added to the flow equation as follows:

1. if $h_{i,j,k} > SURF_{i,j}$, $QETM_{i,j}$ is added to $RHS_{i,j,k}$;
2. if $(SURF_{i,j}\text{-}EXDP_{i,j}) \le h_{i,j,k} \le SURF_{i,j}$, $\frac{-QETM_{i,j}}{EXDP_{i,j}}$ is added to $HCOF_{i,j,k}$, and

   $-QETM_{i,j} + \frac{QETM_{i,j} SURF_{i,j}}{EXDP_{i,j}}$ is subtracted from $RHS_{i,j,k}$; and
3. if $h_{i,j,k} < SURF_{i,j}\text{-}EXDP_{i,j}$, no changes are made in the terms $HCOF_{i,j,k}$ or $RHS_{i,j,k}$.

The value of SURF, the water-table elevation at which evapotranspiration is maximum, is sometimes assumed to be the land-surface elevation; however, the maximum evapotranspiration usually occurs at some depth below land surface because plants can generally withdraw the maximum amount from ground water when a fraction of the root zone is in contact with the water table. The cutoff or extinction depth, EXDP, is frequently assumed to be the distance from SURF down to the bottom of the deepest roots. Considerable variation can be introduced by climatic factors and plant type.

Summary of Stress Packages

Figure 6–14 schematically shows flow into a model cell containing a stress, Qin, and head in the cell, h, for all six stress packages documented in this chapter. These plots illustrate qualitatively the functional differences among the stress packages. Notice that the flows by the Well and Recharge Packages are independent of head in the model cell, whereas the flows calculated for the remaining packages are dependent on head in the model cell. The well inflow is negative, illustrating the typical situation of a pumping well, but any positive or negative value can be applied. Similarly, the recharge inflow is shown as positive, but negative values can be used as desired. The plot for evapotranspiration has been reversed to show inflow to the model rather than outflow as in figure 6–13.

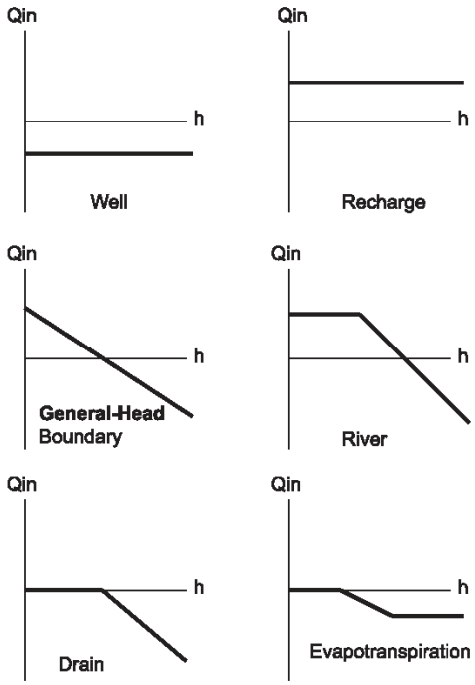


Figure 6–14. Comparison of stresses.

CHAPTER 7
SOLVER PACKAGES

This chapter documents three solvers used in MODFLOW–2005. The focus is on how to use the solvers to produce an acceptable solution to the flow equations with a minimum of computational time. Details of how the solvers work are not included here. These details can be found in the references.

The best solver to use for a problem is usually problem dependent, which is why there are multiple solvers in MODFLOW. The fundamental criterion for solver selection is the ability to solve the equations for the simulation. All of the solvers can solve the equations of typical simulations, but some very large or highly nonlinear problems cannot be solved by all of the solvers. Assuming that a solver can solve the equations, the next most important objective for solver selection is minimizing the time to solve the equations. Solution time of course varies with the power of the computer, but for a given computer, different solvers can take substantially different amounts of time. Another consideration regarding solver selection is the amount of computer memory required. Each solver uses a different amount of memory. The amount of computer memory used, although not nearly the constraint that it once was, can still be of concern for very large simulations. Insufficient memory will of course prevent the solution of the equations. Minimizing the amount of memory used for the model when other programs are running concurrently can be advantageous.

All of the solvers documented here incorporate iteration. Iteration means repeated partial solution of the flow equation resulting in each solution being more accurate than the previous one. Iteration usually is more efficient than direct solution because a single iteration requires a small fraction of the work of direct solution. Doing multiple iterations is still less work than a direct solution. Further, the Direct solver incorporates the capability of iteration to allow the solution of nonlinear problems.

Iteration can occur at two levels. The primary level, called outer iteration, is implemented in the Formulate and Approximate Procedure shown in the overall flow chart for MODFLOW (fig. 3–1). The Formulate Procedure begins by constructing the flow equation. Any head-dependent (nonlinear) terms, such as transmissivity of water-table layers, will change as a result of the head computed in the prior outer iteration. The Approximate Procedure then performs one partial solution of the flow equations. Each invocation of the Formulate and Approximate Procedures is an outer iteration. The second iteration level is called inner iteration. Inner iteration is incorporated within the Approximate Procedure of some solvers. Inner iteration is used to improve accuracy without changing the nonlinear terms. So, the overall flowchart does not directly indicate the possibility of inner iteration. The outer iterations are repeated until a satisfactory solution is obtained, which is called solver convergence, or until a user-specified maximum number of iterations is reached.

Solver convergence cannot be directly determined by the solvers because a solver does not know what the actual solution is. Accordingly, indirect measures of convergence are used. One measure is called the head-change criterion. For every iteration, a solver compares the head from the prior iteration to the new head at all cells. The maximum absolute value of the head change for the iteration is compared to a user-specified value. If the maximum change is less than or equal to the user-specified value, the head-change convergence criterion has been met. The head-change criterion is implemented in all of the solvers.

Another convergence criterion, which is used by the Preconditioned Conjugate-Gradient solver, is called the residual criterion. For every iteration, the solver computes the difference between the inflows and outflows for each cell. The maximum absolute value of the difference between inflows and outflows is compared to a user-specified value. If the maximum change is less than or equal to the user-specified value, the residual criterion has been met. The difference between inflows and outflows is commonly called the residual, which is why this criterion is called the residual criterion.

Strongly Implicit Procedure Package

The Strongly Implicit Procedure (SIP) is a method for solving a large system of simultaneous linear equations by iteration developed by Weinstein, Stone, and Kwan (1969). The implementation of SIP as a MODFLOW package is fully described in McDonald and Harbaugh (1988).

Iteration

The SIP Package uses only outer iteration. The SIP algorithm uses a user-specified number of iteration variables (described below) that are generated from an iteration seed. The iteration seed can be user specified; alternatively, the program will calculate a value. To avoid excessive computation time when the solution process is failing to converge, the user specifies the maximum number of outer iterations, MXITER. The simulation is stopped if the convergence criterion is not met within MXITER iterations. Convergence is determined using the head-change criterion described above.

User Guidance

The SIP iteration variables (sometimes called iteration parameters) mentioned above are used empirically. In practice, iteration variables are used in a repeated cycle. The user must specify the number of iteration variables, NPARM, which normally ranges from 4 to 10. The actual values of the iteration variables are computed from a single value called the seed, which is input variable WSEED. WSEED can be generated automatically from an empirical algorithm, or the user can specify WSEED, which is less than 1 and greater than 0.8.

Several points should be made regarding iteration variable selection. First, the process is essentially empirical and little is understood about why one sequence of variables performs better than another. Second, the variables chosen affect the rate of convergence but (assuming that convergence is achieved) should not influence the final solution. Third, the influence of the variables on the rate of convergence is extremely important.

The SIP solver provided in MODFLOW–2005 includes an additional variable that can influence convergence. This variable, which is designated ACCL in the program, functions as a multiplier of the head change computed by the SIP algorithm; thus, where no impact is desired, ACCL should be assigned a value of 1. Variables similar to ACCL have been used in various versions of SIP (Peaceman, 1977, p. 130) although Weinstein, Stone, and Kwan (1969) do not employ a variable of this type. ACCL is not cycled, but rather has a single user-assigned value. As a general rule, ACCL initially should be given a value of 1, and improvement in the rate of convergence should be pursued through adjustment of the seed term, as explained below. If problems with convergence persist, values of ACCL other than 1 can be tried.

Experience has shown that setting ACCL to 1 and using the seed value calculated by the program does not always produce optimum convergence—that is, the number of iterations required to achieve convergence is not minimum. Convergence rates will deviate from the optimum if the absolute value of head change in each iteration is consistently either too small or too large. When the head change is too large, the computed head overshoots the correct value, and oscillations occur as the head change repeatedly reverses to compensate for the overshoot. Severe overshoot causes divergence, while moderate overshoot can slow down convergence. When head change is too small, the opposite problem occurs; head tends to approach the correct value monotonically, but very slowly. In severe situations, the head changes at each iteration may be so small that the criterion for convergence is satisfied at all points, even though the computed heads are still far from the correct values. In such situations, a substantial volumetric budget imbalance will occur.

Weinstein, Stone, and Kwan (1969) suggest that a trial and error method can be used to improve the choice of WSEED. This can be done by making an initial run using the WSEED calculated by the program or chosen from experience, and using 1 for ACCL. A value of 5 for NPARM is usually acceptable. The trend of the maximum head change per iteration, with increasing iteration number, is observed for the iterations of a single time step. A table showing the cell at which the maximum absolute value of head change occurs for each iteration will be printed in the Listing File when variable IPRSIP is 1. The head change at these cells also is printed, including the sign. A positive sign means head increased. Normally, some variation in head change occurs from one iteration to the next caused by the cycling of iteration variables, but this variation is often superimposed on an overall trend in which head change

tends either to increase or decrease as iterations continue; this overall trend (which is often most evident in the later iterations of the test) is of interest here. Some oscillations (reversals in sign) of the computed head change are normal during convergence; however, repeated oscillation is a sign of overshoot, indicating that computed head changes are too great for optimal convergence. On the other hand, head changes that are too small are indicated by a very flat overall trend. For proper evaluation of the trend, the trial generally should be run for a number of iterations equal to 4 or 5 times the number of iteration variables, unless convergence occurs before this.

Following the initial trial, WSEED is multiplied by a number between 2 and 10 if head changes in the initial trial appear to be too large, and divided by a number between 2 and 10 if those head changes appear to be too small. If the trend in the initial trial is unclear, either multiplication or division of the seed may be tried. In any case, a second run is made using the new WSEED value, and the trend of head change versus iteration level is again examined. The results are compared with those of the initial trial to see if the rate of convergence has improved. If both runs have converged, the comparison is based on the number of iterations required for convergence; if they have failed to converge, the comparison is based on the magnitude of the head changes observed in the final iterations.

The trial runs can be continued to refine further the choice of WSEED; in general, the WSEED value will be multiplied or divided by progressively smaller numbers at each step of the procedure. Carrying the process too far usually is not worthwhile; multiplication or division of the seed by factors less than 2 is seldom warranted.

In most cases, a satisfactory WSEED value developed by this procedure will remain satisfactory even though changes in the model are introduced—for example, additional stresses, modifications in boundary conditions, or changes in the model mesh. If, however, convergence problems arise after such changes, the trial and error procedure can be repeated. It should be noted that the more strongly diagonal the coefficient matrix, the less important the choice of WSEED will be. Thus, source terms such as evapotranspiration or stream seepage, which affect only coefficients on the main diagonal, normally tend to make the choice of WSEED less critical; and the addition of such terms to a model seldom necessitates modification of WSEED.

Improvements in the rate of convergence can be obtained by also adjusting ACCL. Increases in ACCL will cause increases in the head change at each iteration, while decreases in ACCL will cause decreases in head change. The trial procedure described above can be used for this case as well; however, changes in WSEED and in ACCL should not be attempted in the same set of trial runs.

Slowing the process of convergence is sometimes necessary to prevent cells from converting to the no-flow condition as a result of head overshoot during iterations. This requires ACCL to be less than 1. In these situations, optimal convergence cannot be considered convergence in the minimum number of iterations, but rather convergence in the smallest number of iterations that does not involve head overshoot. The procedure of examining maximum head change per iteration and adjusting iteration variables can be used again to determine when this condition is being met, and to develop the required WSEED or ACCL values.

Preconditioned Conjugate-Gradient Package

The Preconditioned Conjugate-Gradient Solver (PCG) Package is fully documented in Hill (1990).

Iteration

The PCG Package uses both inner and outer iteration. To avoid excessive computation time when the solution process is failing to converge, the user specifies the maximum number of outer iterations, MXITER. The simulation is stopped if the convergence criteria are not met within MXITER outer iterations. Convergence is determined by using the head-change and residual criteria described above. Both criteria must be met.

Variable ITER1 specifies the maximum number of inner iterations for each outer iteration. The inner iteration ends when the closure criteria are met or after ITER1 iterations if the closure criteria are not met. When MXITER is 1, convergence is achieved if the closure criteria are met during inner iteration; convergence fails if the closure criteria are not met. When MXITER is greater than 1, a new outer iteration is started after inner iteration ends even if the closure criteria are met during inner iteration. Convergence is achieved only if the convergence criteria are met in the first inner iteration of an outer iteration. This approach for achieving convergence prevents convergence

of nonlinear problems until the changed flow equations resulting from a new outer iteration no longer cause additional head and residual changes.

User Guidance

For a linear problem, a single outer iteration with many inner iterations theoretically is adequate, but a few outer iterations should be used for large (greater than 10,000 cells) linear models. For numerical reasons described in Hill (1990, p. 8), the outer iterations improve the accuracy of large linear problems.

Nonlinear problems generally run most efficiently with five or more outer iterations and a large number of inner iterations (20 or more). The solution process becomes inefficient if the nonlinear terms are reformulated frequently by having a small (less than 20) value for ITER1.

The PCG solver provided in MODFLOW–2005 includes an additional variable that can influence convergence. This variable, which is designated DAMP in the program, functions as a multiplier of the head change computed by the PCG algorithm; thus, where no impact is desired, DAMP should be assigned a value of 1. As a general rule, DAMP initially should be given a value of 1. If problems with convergence occur, values of DAMP less than 1 can be tried. Slowing the process of convergence is sometimes necessary to prevent cells from converting to the no-flow condition as a result of head overshoot during an iteration. In these situations, a value of DAMP less than 1 should be used, even though additional iteration may be necessary.

The PCG solver actually incorporates two forms of the Preconditioned Conjugate-Gradient method: the Modified Incomplete Cholesky and the Polynomial methods. Knowledge of the details of these methods generally is not necessary. The most important aspect is that the Modified Incomplete Cholesky method is almost always faster. The Polynomial method is potentially faster only on parallel computers where some of the work can be performed simultaneously on multiple processors. Details can be found in Hill (1990).

When using the Modified Incomplete Cholesky method, the user-specified relaxation parameter called RELAX is provided to reduce the number of iterations for convergence in some situations. The value initially should be set to 1. If convergence fails or takes an unusually large number of iterations, values of 0.99, 0.98, and 0.97 can be used on a trial-and-error basis.

Direct Solver Package

The Direct Solver (DE4) Package is fully documented in Harbaugh (1995).

Iteration

The DE4 Package uses inner or outer iteration, but not at the same time. For a linear problem, a single inner iteration is theoretically adequate. If the budget error is unacceptably large in a linear model, a few inner iterations should be used. For numerical reasons described in Harbaugh (1995, p. 7), the inner iteration improves the accuracy of linear problems. Nonlinear problems require outer iteration; generally five or more outer iterations are needed. To avoid excessive computation time when the solution process is failing to converge, the user specifies the maximum number of iterations, ITMX. The simulation is stopped if the convergence criterion is not met within ITMX iterations. Convergence is determined using the head-change criterion described above.

User Guidance

The use of inner or outer iteration is determined according to the frequency at which the [A] matrix in the flow equation (eq. 2–27) changes. This affects the efficiency of solution; substantial work can be avoided if [A] remains constant during all or part of the simulation. Two situations can cause [A] to change during a simulation. The first is nonlinear terms in the flow equation. A nonlinear term is one in which the term varies with the head that is being computed. An example of this is dependence of conductance on water level under water-table conditions. This can happen in the Block-Centered Flow Package when LAYCON is 1 or 3 and in the Layer-Property Flow Package when LAYTYP is not 0. Similarly, storage terms in [A] vary with head when LAYCON is 2 or 3 or LAYTYP is not

0. Other nonlinear terms are the formulations that change based upon head, such as seepage from a river changing from head dependent flow to a constant flow when head drops below the bottom of the riverbed.

The second situation that causes [A] to change during a simulation is coefficients of simulated head that change with time. For example, riverbed conductance, drain conductance, maximum evapotranspiration rate, evapotranspiration extinction depth, and general-head boundary conductance are coefficients of simulated head, and new values can be read every stress period.

The user indicates the frequency of change in [A] through variable IFREQ. IFREQ = 1 indicates that the flow equations are linear and that coefficients of simulated head for all stress terms are constant for all stress periods. IFREQ = 2 indicates that the flow equations are linear as described for IFREQ=1, but coefficients of simulated head for some stress terms may change at the start of each stress period. For a simulation consisting of only one stress period, IFREQ = 2 has the same meaning as IFREQ = 1. IFREQ = 3 indicates that a nonlinear flow equation is being solved, which means that some terms in [A] depend on simulated head.

If time steps are not equal, [A] changes every time step even if the problem is linear because the storage term depends on the length of the time step. The user does not have to recognize this situation because the DE4 solver automatically recognizes that [A] will change when the time step length changes. The use of non-equal time steps, however, results in less efficiency (longer execution times) for linear problems. Users wanting to solve linear flow equations most efficiently should consider using stress periods with equal-length time steps. There is no execution-time penalty for unequal time steps in nonlinear problems.

If a nonlinear flow equation is being solved, outer iteration (ITMX > 1) is required. When nonlinearities caused by water-table calculations are part of a simulation, obvious signs may not necessarily be evident in the output from a simulation that does not use external iteration to indicate that iteration is needed. In particular, the budget error may be acceptably small without iteration even though error in head is substantial because of nonlinearity. Therefore, the user must be careful to specify a value of 3 for IFREQ when a problem is nonlinear.

To understand why signs that iteration is needed for nonlinear equations may not be obvious, consider the water-table computation of transmissivity. During each iteration, a new transmissivity is calculated based on the previous head. Then the flow equations are solved, and a budget is computed using the new head with the same transmissivities. No budget discrepancy results because heads are correct for the transmissivity being used at this point; however, the new heads may indicate a substantial change in transmissivity. The new transmissivity will not be calculated unless another iteration occurs, which causes the Formulate Procedure to be invoked. Therefore, when one or more layers are under water-table conditions, iteration should always be tried. The maximum change in head for each iteration (printed by DE4 when IPRD4 = 1 and MUTD4 = 0) provides an indication of the impact of all nonlinearities.

DE4 uses a specialized form of direct solution that is optimized for sparse, symmetric matrices. The user need not know the details of the solution method, but there is one aspect of solution for which the user must have knowledge. When the equations are solved, they are divided into two parts, an upper and lower part. Memory must be allocated for both parts. The program can attempt to determine the required memory, but in some situations the automatic approach is not optimal. Automatic allocation is used when variables MXLOW, MXUP, and MXBW are specified as 0. Unless a large number of constant-head or no-flow cells are present or the automatic method results in the message “INSUFFICIENT MEMORY FOR DE4 SOLVER“ in the Listing File, then the automatic method should be used. If a message indicating insufficient memory occurs, then the user must not use automatic allocation. The Listing File will show the required values for MXLOW, MXUP and MXBW, and the user can modify the DE4 Package input file to specify these values.

If a large number of constant-head or no-flow cells exist, then some memory can possibly be saved by specifying non-zero values for MXUP, MXLOW, and MXBW. This is generally not worth the effort, however, unless computer memory is constrained. If the user desires or needs to specify values, the best way to do this is to run the model with MXUP, MXLOW, and MXBW set to 0, and then look in the Listing File to find the optimal values. The model can then be rerun using the new values.

CHAPTER 8
INPUT INSTRUCTIONS

This chapter documents the data files used by MODFLOW-2005. An initial section describes how gridded data are input, another section describes the form of input instructions, and these are followed by sections describing the input for each package.

Input of Gridded Data

For input purposes, data for cells in the model grid can be categorized in two ways according to the density or sparseness of the data. Layer data refers to any type of data for which a value is required for every cell in one or more horizontal layers of the grid. Examples of layer data include areal recharge flux, hydraulic conductivity, and specific storage. List data refers to any type of data for which data values are required for only some of the cells in the grid. Examples include the well recharge rate as simulated by the Well Package and the riverbed conductance as simulated by the River Package.

In addition, there are two methods used for defining layer and list data: direct and parameter. All layer and list data in MODFLOW-2005 can be read using the direct method whereas only part of the layer and list data can be defined using parameters. The direct and parameter methods for layer and list data are described in the following sections.

Direct Input of Layer Data

When directly inputting layer data, the form of input is to read the values one row at a time starting at the first row. Within a row, columns are read across a line of a file starting with column one. If one line cannot hold all columns in a row, additional lines are used as required. Because the data are read in this prescribed row/column order, the row and column do not need to be explicitly indicated. When layer data for a given type are required for multiple layers, the layers are read in order. For example, STRT is the initial head and is defined by reading initial head for layer 1, followed by layer 1, and so on for all layers of the grid. Some packages read data for multiple data types for each layer, and they do this by reading all the types for one layer before advancing to the next layer. LPF is an example of a package that uses this approach.

The direct approach for reading layer data is implemented by utility subroutines, which provide a common mechanism for reading the layer data required for any package. Two utility subroutines for reading layer data are provided in MODFLOW: one for integer data (U2DINT) and one for real data (U2DREL). Both subroutines start by reading a "control line." The control line is read from the data file for the package that requires the layer data. If all the cells for the layer have the same value, the value is specified on the control line, and reading separate values for each cell is not necessary. If the values vary among cells, lines containing the values for all cells are read either from the package file or from the file indicated in the control line. The format for reading the values also is specified in the control line. Thus, a great deal of flexibility is allowed regarding the organization of the input data for a simulation. Detailed input instructions for U2DREL and U2DINT are contained in the Input Instructions for Array Reading Utility Subroutines section of this chapter.

Direct Input of List Data

When directly inputting list data, one line of a file is used for each cell for which data are specified. Each line explicitly includes layer, row, and column indices, and the data value or values follow the indices on the same line. For example, the River Package requires three types of data for each cell at which a river interacts with the ground-water system (river stage, riverbed conductance, and elevation of the bottom of the riverbed), and each line of data includes all three data types. The order of cells in the list does not matter. The direct approach of reading list data is implemented by a utility subroutine (ULSTRD), which provides a common mechanism for reading list data required for any package.

Parameters

As defined in MODFLOW-2005, a parameter is a single value that can be used to determine data values for multiple cells. Each parameter is identified by a name and type. The name is user-defined and is used to reference the parameter. The parameter type specifies the kind of data being defined. Parameters can make model data easier to adjust when calibrating a model. Even in a small model, thousands of data values are used by the Ground-Water Flow Process; yet, the available observation data typically are sufficient only to calibrate a relatively small number of values. Parameters can be defined in such a way that only the parameters are adjusted when calibrating the model. Parameters also can facilitate making a series of simulations that require groups of data values to be modified by prescribed amounts. For example, all the pumping rates for the wells determined by one parameter can be doubled by doubling the parameter. As described in the following sections, parameter definition is different for layer and list data.

The use of parameters also can be differentiated by two approaches, which are termed simple and additive. The simple approach is the most common; in this approach, the value for an individual cell is defined by one parameter. In the more complex additive approach, the data value for a single cell is determined by adding contributions from multiple parameters. The additive approach allows interpolation techniques, such as kriging (a procedure used in geostatistics to estimate unknown values from known values), and stochastic techniques, such as the pilot-point method, to be used to produce smooth variations of data values throughout a region based upon the multiple parameter values.

Simple Parameters: the Value for a Cell is Determined by One Parameter

When a simple parameter is used, the data value for a cell is calculated as the product of the parameter value, which might apply to many cells, and a cell multiplier, which applies to only that cell. The multiplier allows a single parameter to define different data values for the different cells associated with the same parameter. For example, consider a riverbed for which the riverbed conductance is to be defined using a parameter. Field data might indicate that the riverbed has a uniform hydraulic conductivity over a length that covers many cells, but the geometry of the riverbed varies from cell to cell. A single parameter can be used to represent the uniform riverbed hydraulic conductivity. The multiplier for each cell associated with the parameter would then represent the area of the riverbed in the cell divided by the thickness of the riverbed. Thus, the final data value for each cell would be the product of the riverbed hydraulic conductivity and riverbed area divided by thickness, which is the riverbed conductance.

Another example is hydraulic conductivity for model cells. Consider a situation in which the field data indicate interbedded coarse- and fine-grained sediments in one region of an aquifer. The fine-grained sediments have such low hydraulic conductivity that their contribution to the transmissivity is negligible. The coarse-grained sediments are assumed to have a generally uniform hydraulic conductivity, and the proportion of high-conductivity sediments has been mapped throughout the region. To avoid using many parameters to represent the varying average hydraulic conductivity resulting from the combination of coarse and fine materials, a single parameter representing the uniform hydraulic conductivity of the coarse-grained material could be defined. The multipliers for individual cells could then represent the fraction of the total thickness that is coarse-grained material. The product of the parameter and multiplier would then represent the average hydraulic conductivity for each cell.

Simple Parameters for List Data

Each package that incorporates list data requires one or more types of data to be defined for listed cells. The input instructions for a package indicate which data types can be defined using parameters. Each data type is given a specific name that must be included as part of the input data that defines a parameter of that type. For example, the River Package uses three types of list data (river stage, riverbed conductance, and elevation of the bottom of the riverbed), but only riverbed conductance can be defined using parameters. The parameter type for riverbed conductance is RIV.

For list data, the cells that are associated with each parameter are defined in a list in which each line is identical to the line that would normally occur in the package input file, with one exception: a multiplier is in the position where the data of the specified type would be if parameters were not being used. The value for the specified data

type will be computed as the product of this multiplier and the parameter value. All the other data values on the line are the same as they would be if parameters were not being used.

For example, consider again the use of parameters to define data for the River (RIV) Package. When parameters are not used, input data for a cell includes six values: layer, row, column, stage, riverbed conductance, and riverbed bottom elevation. When parameters are used, the list for each parameter also contains six values per line. The fifth value is the multiplier for riverbed conductance; the other five values are the same as when parameters are not used. Thus, a non-parameter list can be made into a parameter list by changing just the fifth value on each line from a conductance to a multiplier. If the parameter value is specified as 1.0, the multiplier would be the same as the conductance.

Here is an illustration of how list data are calculated using contributions from two RIV parameters, R1 and R2. Assume that the data list for parameter R1 is specified as:

Layer	Row	Column	River Stage	Riverbed Conductance Multiplier	Riverbed Bottom Elevation
1	4	5	4.5	150	4.0
1	5	6	4.8	180	4.3
1	5	7	5.0	200	4.5
1	6	7	5.1	250	4.6

Assume that the data list for parameter R2 is specified as:

Layer	Row	Column	River Stage	Riverbed Conductance Multiplier	Riverbed Bottom Elevation
1	10	22	8.9	500	8.0
1	11	21	9.5	600	8.5
1	11	20	10.2	700	9.0

If R1 = 10, and R2 = 20, the resulting list data for the river is:

Layer	Row	Column	River Stage	Riverbed Conductance	Riverbed Bottom Elevation
1	4	5	4.5	1500	4.0
1	5	6	4.8	1800	4.3
1	5	7	5.0	2000	4.5
1	6	7	5.1	2500	4.6
1	10	22	8.9	10000	8.0
1	11	21	9.5	12000	8.5
1	11	20	10.2	14000	9.0

The details of how to specify parameter information are contained in the input instructions for each package.

Simple Parameters for Layer Data

Each package that incorporates layer data may have any number of variables (types of data) to be defined. The input instructions for a package indicate which variables can be defined using parameters. When a parameter is defined, the parameter type, which indicates the variable being defined, must be specified. For example, in the Layer-Property Flow Package, six layer-data variables, such as horizontal hydraulic conductivity and specific storage, can be defined using parameters, and one, called WETDRY, cannot. The parameter type for defining horizontal hydraulic conductivity is HK, and the parameter type for defining specific storage is SS.

For layer data, parameter multipliers for cells are defined using multiplier arrays. “Array” is programming terminology for a variable that contains many values. In this case, each multiplier array contains values for every cell in a layer; thus, a multiplier array is a kind of layer data. Multiplier arrays are given names. A different multiplier array can be used for each layer to which the parameter applies, and the multiplier arrays are identified when the

parameter is defined. Thus, the multiplier for cell (i,j,k) is the (i,j) value of the multiplier array that is specified for layer "k."

To allow only some of the cells of a layer to be associated with a parameter, a capability called zonation is used. Zonation is implemented through zone arrays. Like multiplier arrays, each zone array is named and contains values for every cell in a layer. Values in a zone array are integers. When a parameter is defined, the zone array and one or more zone values are specified. The parameter applies to cells at which the value of the zone array matches any one of the specified zone values; that is, the data value at a cell is the product of the multiplier array at the cell and the parameter value only if the value of the zone array matches one of the zone values specified for the parameter. There can be a different zone array for every layer to which the parameter applies.

Multiplier and zone arrays are defined as part of the input to the Basic Package. These arrays are read by the utility subroutines the same way as other layer data are read.

As an example of how layer data are calculated from parameters, consider the following multiplier and zone arrays for a layer that has 3 rows and 4 columns.

Multiplier array:

		Column			
		1	2	3	4
Row	1	200.	30.	350.	400.
	2	200.	30.	60.	400.
	3	15.	30.	60.	425.

Zone array:

		Column			
		1	2	3	4
Row	1	1	3	1	1
	2	1	2	3	4
	3	3	3	3	5

If parameter P1 applies to values where the zone array is 2 or 3 and parameter P2 applies to values where the zone array is 1, 4, or 5; then the following input values would be calculated.

		Column			
		1	2	3	4
Row	1	200×P2	30×P1	350×P2	400×P2
	2	200×P2	30×P1	60×P1	400×P2
	3	15×P1	30×P1	60×P1	425×P2

If P1=50 and P2 =80, the final result is:

		Column			
		1	2	3	4
Row	1	16000	1500	28000	32000
	2	16000	1500	3000	32000
	3	750	1500	3000	34000

As already mentioned, a single parameter that defines layer data can specify data for multiple layers. Each layer included in a parameter can be assigned a different multiplier and zone array. The above procedure is applied to calculate the data values for the cells in each layer.

As is shown in the example, the use of a multiplier array allows layer data that are determined by one parameter to be different at each cell. Further, the relative values of the parameter-based layer data are the same as the relative values of the multipliers regardless of the value of the parameter. Consider parameter P1 in the example above. The multiplier for each successive column from left to right is twice the value of the previous column. Thus, the resulting data values that are based on P1 also differ by a factor of 2 from left to right.

Additive Parameters

There is no direct indicator to specify that additive parameters are being used. Rather, additive parameters automatically occur if two or more parameters of the same type defined as above incorporate the same cell(s). In this situation, the final data value equals the sum of the contributions from all of the parameters. As mentioned previously, additive parameters can be used to perform interpolation such as linear, distance weighting, and kriging.

Additive Parameters for List Data

Consider a hypothetical situation in which the River Package is being used. In this situation the riverbed sediments are coarse grained at the head of the river and fine grained at the mouth. The resulting riverbed conductance along the first three cells representing the river is assumed to be a constant indicative of the coarse-grained sediments, and the conductance in the last three cells is assumed to be a constant indicative of the fine-grained sediments. In the middle region of the river, which crosses five model cells, the two materials are assumed to be mixed in such a manner that the riverbed conductance varies linearly from the coarse-grained value to the fine-grained value. This riverbed conductance distribution can be represented by two additive parameters—one for the coarse-grained material and one for the fine-grained material. The following is an example of a data set for this situation in which P1 represents the riverbed conductance of the coarse-grained material and P2 represents the riverbed conductance of the fine-grained material. Assume that the data list for parameter P1 is:

Layer	Row	Column	River Stage	Riverbed Conductance Multiplier	Riverbed Bottom Elevation
1	7	9	7.5	1.0	7.0
1	8	9	6.8	1.0	6.3
1	9	9	5.9	1.0	5.4
1	10	9	5.1	.83	4.6
1	11	9	4.5	.67	4.0
1	12	9	3.8	.50	3.3
1	13	9	3.2	.33	2.7
1	14	9	2.5	.17	2.0

Assume that the data list for parameter P2 is:

Layer	Row	Column	River Stage	Riverbed Conductance Multiplier	Riverbed Bottom Elevation
1	10	9	5.1	.17	4.6
1	11	9	4.5	.33	4.0
1	12	9	3.8	.50	3.3
1	13	9	3.2	.67	2.7
1	14	9	2.5	.83	2.0
1	15	9	2.1	1.0	1.6
1	16	9	1.8	1.0	1.3
1	17	9	1.5	1.0	1.0

Parameters P1 and P2 both include the five cells where the two riverbed materials are assumed to intermix, so the riverbed conductance for those cells will be the sum of the components from both parameters. For example, the river reach at cell (1, 12, 9) is in the middle of the transition region, and the riverbed conductance multipliers are both 0.5 so that the riverbed conductance will be the sum of half of P1 and half of P2. If P=100 and P2=10, then the values for riverbed conductance will be:

Layer	Row	Column	River Stage	Riverbed Conductance	Riverbed Bottom Elevation
1	7	9	7.5	100.0	7.0
1	8	9	6.8	100.0	6.3

1	9	9	5.9	100.0	5.4
1	10	9	5.1	84.7	4.6
1	11	9	4.5	70.3	4.0
1	12	9	3.8	55.0	3.3
1	13	9	3.2	39.7	2.7
1	14	9	2.5	25.3	2.0
1	15	9	2.1	10.	1.6
1	16	9	1.8	10.	1.3
1	17	9	1.5	10.	1.0

Additive Parameters for Layer Data

As an example of using two parameters to determine the data values for the same cells, assume that the user desires to have the distribution of horizontal hydraulic conductivity, which is layer data, vary linearly from 10 to 100 from the left side to the right side of a layer. This can be done using two additive parameters, P1 and P2, that both apply to all cells in the layer. For a layer consisting of 4 rows and 5 columns, the multiplier array for P1 is:

		Column				
		1	**2**	**3**	**4**	**5**
Row	1	1.0	0.75	0.50	0.25	0.0
	2	1.0	0.75	0.50	0.25	0.0
	3	1.0	0.75	0.50	0.25	0.0
	4	1.0	0.75	0.50	0.25	0.0

The multiplier array for P2 is:

		Column				
		1	**2**	**3**	**4**	**5**
Row	1	0.0	0.25	0.50	0.75	1.0
	2	0.0	0.25	0.50	0.75	1.0
	3	0.0	0.25	0.50	0.75	1.0
	4	0.0	0.25	0.50	0.75	1.0

If P1 = 10., and P2=100., the contributions from both parameters will be added as follows.

The contribution from both parameters is:

		Column				
		1	**2**	**3**	**4**	**5**
Row	1	10+0	7.5+25	5+50	2.5+75	0+100
	2	10+0	7.5+25	5+50	2.5+75	0+100
	3	10+0	7.5+25	5+50	2.5+75	0+100
	4	10+0	7.5+25	5+50	2.5+75	0+100

The final result is the desired values:

		Column				
		1	2	3	4	5
Row	1	10.	32.5	55	77.5	100.
	2	10.	32.5	55	77.5	100.
	3	10.	32.5	55	77.5	100.
	4	10.	32.5	55	77.5	100.

This same distribution could be created in other ways; for example, (1) without using parameters by simply reading the final data values as input data, (2) using a single parameter by having the multiplier vary linearly from 1.0 on the left and 10.0 on the right and having a parameter value be 10.0. Using two parameters would be advantageous, however, if there were a need to try many distributions that have a linear variation from left to right. The two-parameter approach would allow any such linear distribution to be created by simply changing the two parameter values.

Time-Varying Parameters

A time-varying parameter is a parameter for which multiple instances can be defined. Each instance of a parameter is a different set of data to which the parameter value can be applied to construct data values for the cells associated with the parameter; however, the same parameter value applies to all instances of a parameter. Only one instance of a parameter can be in use during a stress period. The instance being used can be changed each stress period.

Time-varying parameters have been incorporated into MODFLOW-2005 because they make it possible to specify time constraints on a parameter when using parameter estimation. Consider for example a simulation in which winter ground-water recharge is known (or assumed) to be twice the summer recharge. Without time-varying parameters, two parameters would be required to simulate this – one for winter and one for summer. Parameter estimation could be used to estimate either or both parameters, but without instances there would be no way to cause parameter estimation to find the optimal values while forcing the winter value to be twice the summer value. In contrast, the time-varying parameter capability makes it possible to define one recharge parameter with two instances – one for winter and one for summer. The winter instance would consist of data that cause the recharge to be twice that of the summer instance. The use of parameter estimation to calculate the optimal value for the single time-varying parameter would always result in the winter recharge being twice the summer recharge.

As described above, a parameter for list data is defined by specifying a list of cells and associated data to which the parameter applies. Each time-varying parameter for list data is defined as having one or more instances, where each instance is a list of cells and associated data. Each instance must include the same number of cells, but the specific cells can differ among the instances.

A parameter for array data is defined using a list of parameter clusters that specify zone and multiplier arrays indicating the cells to which the parameter applies. Each time-varying parameter for array data is defined as having one or multiple instances, where each instance is a list of clusters. Each instance must be defined using the same number of clusters, but the specific clusters can differ among the instances.

Form of Input Instructions

Input instructions for the Ground-Water Flow Process of MODFLOW-2005 are separated by package. The Basic Package reads several files, and each file is described in a separate subsection. The final sections in this chapter describe the Array Reading Utility Subroutines and the List Utility Subroutine. The instructions for a file are grouped by numbered items. The headings "FOR EACH SIMULATION" and "FOR EACH STRESS PERIOD" indicate the items that are entered once and those that are repeated for every stress period, respectively.

Each item consists of input variables. If a variable is represented in the program code by a variable of the same name, then the variable is shown in all upper case. Variables that are not actual program variables are shown with the first character in uppercase and the remaining characters in lowercase. Each variable is defined after all the items are listed. Optional variables are enclosed between "[" and "]" symbols.

An input variable may include a single value or multiple values. One-dimensional variables are multi-valued variables in which the number of values is indicated by a single number in parentheses after the name. Two-dimensional variables are multi-valued variables in which the number of values is indicated by two numbers in parentheses after the name. The total number of values represented is the product of the two numbers. As described previously, a layer variable is a two-dimensional variable that includes one value for each cell in a layer. Therefore, the column and row dimensions are the numbers in parentheses after the variable. Although the program contains many three-dimensional variables that are defined through user input, these are treated as multiple layer variables for input purposes. For example, "IBOUND(NCOL,NROW)" indicates that IBOUND is a layer variable. An IBOUND layer variable will be read for each layer of the grid. Most multi-value variables are read by utility subroutines, and if so, the utility subroutine (U1DREL, U2DREL, or U2DINT) is named along with the variable. Subroutine U1DREL is used for reading one-dimensional variables that are real numbers, U2DREL is used for reading two-dimensional variables that are real numbers, and U2DINT is used for reading two-dimensional variables that are integer numbers.

Some items consist of several variables that can be repeated multiple times, and the first three variables are layer, row, and column indices. This kind of data is called list data. List items are usually read by the list utility subroutine, ULSTRD. The use of ULSTRD is mentioned as part of the note that describes the repeated item.

The input data for each item must start on a new line. All variables for an item are assumed to be contained in a single line unless the item consists of a multi-valued variable. That is, a multi-valued variable may occupy multiple lines. When reading a layer variable, the data for each row must start on a new line.

Each input variable has a data type, which can be Real, Integer, or Character. Integers are whole numbers and must not include a decimal point or exponent. Real numbers can include a decimal point and an exponent. If no decimal point is included in the entered value, then the decimal point is assumed to be at the right side of the value. Any printable character is allowed for character variables. All variables starting with the letters I-N are integers. Variables starting with the letters A-H and O-Z are either real numbers or character data, and most of these variables are real numbers; that is, there are few character variables in MODFLOW. This data-type convention is used for all input variables regardless of whether a variable is an actual program variable.

Both free and fixed formatting are used as specified throughout the input instructions. With fixed format, a data value has a specified column location, called the field, within a line. If a numeric value does not require the entire field width, the value should be right justified within the field, and blanks should fill the remaining width of the field. The justification of character data within a field does not matter unless specifically noted.

With free format, values are not required to occupy a fixed number of columns in a line. Each value can occupy one or more columns as required to represent the value; however, the values must still be included in the prescribed order. One or more spaces, or a single comma optionally combined with spaces, must separate adjacent values. Also, a numeric value of zero must be explicitly represented with 0 and not by one or more spaces when free format is used, because detecting the difference between a space that represents 0 and a space that represents a value separator is not possible. Free format is similar to Fortran's list directed input.

Two capabilities included in Fortran's list-directed input are not included in the free-format input implemented in MODFLOW-2005.

Null values in which input values are left unchanged from their previous values are not allowed. In general, MODFLOW's input values are not defined prior to their input.

A "/" cannot be used to terminate an input line without including values for all the variables; data values for all input variables must be explicitly specified on an input line.

For character data, MODFLOW's free format implementation is less stringent than the list-directed input of Fortran 77. Fortran requires character data to be delineated by apostrophes. MODFLOW does not require apostrophes unless a blank or a comma is part of a character variable.

Basic Package Input Instructions

The Basic Package reads several files: Name File, Discretization File, Basic Package file, Multiplier Array file, Zone Array file, Output Control Option file, the Time-Variant Specified-Head Option file, and the Parameter Value File. The Name File, Discretization File, and Basic Package file are required for all simulations. The other files are optional.

Name File

The Name File contains the names of most of the input and output files used in a model simulation and controls whether or not parts of the model program are active. ("OPEN/CLOSE" files, described in the Input Instructions for Array Reading Utility Subroutines section, are not included in the Name File.) The Name File is read on unit 99. The Name File is constructed as follows.

FOR EACH SIMULATION

```
1.  Ftype     Nunit     Fname    [Fstatus]
```

The Name File contains one of the above lines (item 1) for each file. All variables are free format. The length of each line must be 299 characters or less. The lines can be in any order except for the line where Ftype (file type) is "LIST" as described below.

Comment lines are indicated by the # character in column one and can be located anywhere in the file. Any text characters can follow the # character. Comment lines have no effect on the simulation; their purpose is to allow users to provide documentation about a particular simulation. All comment lines after the first item-1 line are written in the listing file.

Explanation of Variables in the Name File:

Ftype—is the file type, which must be one of the following character values. Ftype may be entered in any combination of uppercase and lowercase. (For users of MODFLOW-2000, the GLOBAL file type is no longer supported in MODFLOW-2005.)

LIST for the Listing File—This type must be present and must be the first file in the Name File.
DIS for the Discretization File
BAS6 for the Basic Package
MULT for the multiplier array file
ZONE for the zone array file
OC for the Output Control Option
CHD for the Time-Variant Specified-Head Option
PVAL for the Parameter Value File
BCF6 for the Block-Centered Flow Package
LPF for the Layer Property Flow package
HFB6 for the Horizontal Flow Barrier Package
RCH for the Recharge Package
RIV for the River Package
WEL for the Well Package
DRN for the Drain Package
GHB for the General-Head Boundary Package
EVT for the Evapotranspiration Package
SIP for the Strongly Implicit Procedure Package
PCG for the Preconditioned Conjugate-Gradient Package
DE4 for the Direct Solution Package

DATA(BINARY) for binary (unformatted) files, such as those used to save cell-by-cell budget data and binary (unformatted) head and drawdown data. Files of this type are rewound at the start of each parameter-estimation iteration.

DATA for formatted (text) files, such as those used to save formatted head and drawdown and for input of data from files that are separate from the primary package input files. Files of this type are rewound at the start of each parameter-estimation iteration.

Nunit—is the Fortran unit to be used when reading from or writing to the file. Any legal unit number on the computer being used can be specified except unit 99. Unit 99 is used for the Name File and for reading multi-valued variables using the OPEN/CLOSE option of the utility subroutines (see Input Instructions for Array Reading Utility Subroutines section). The unit number for each file must be unique.

Fname—is the name of the file, which is a character value. Pathnames may be specified as part of Fname.

Fstatus—is the optional file status, which applies only to file types Data and Data(Binary). Two values are allowed: OLD and REPLACE. "Old" indicates that the file should already exist. "Replace" indicates that if the file already exists, then it should be deleted before opening a new file. The default is to open the existing file if the file exists or create a new file if the file does not exist.

Discretization File

Discretization information is read from the file that is specified by "DIS" as the file type.

FOR EACH SIMULATION

0. [#Text]

Item 0 is optional—"#" must be in column 1. Item 0 can be repeated multiple times.

1. NLAY NROW NCOL NPER ITMUNI LENUNI

2. LAYCBD(NLAY)

3. DELR(NCOL) - U1DREL

Figure 8-1 illustrates the orientation of DELR and DELC.

4. DELC(NROW) - U1DREL

5. Top(NCOL,NROW) - U2DREL

6. BOTM(NCOL,NROW) - U2DREL

Item 6 is repeated for each model layer and Quasi-3D confining bed in the grid. Thus, the number of BOTM variables must be NLAY plus the number of Quasi-3D confining beds. The BOTM variables are read in sequence going down from the top of the system. For example, in a 3-layer model with a Quasi-3D confining bed below layer 2, there would be 4 BOTM arrays. The arrays would be the bottom of layer 1, the bottom of layer 2, the bottom of the Quasi-3D confining bed below layer 2, and the bottom of layer 3.

FOR EACH STRESS PERIOD

7. PERLEN NSTP TSMULT Ss/Tr

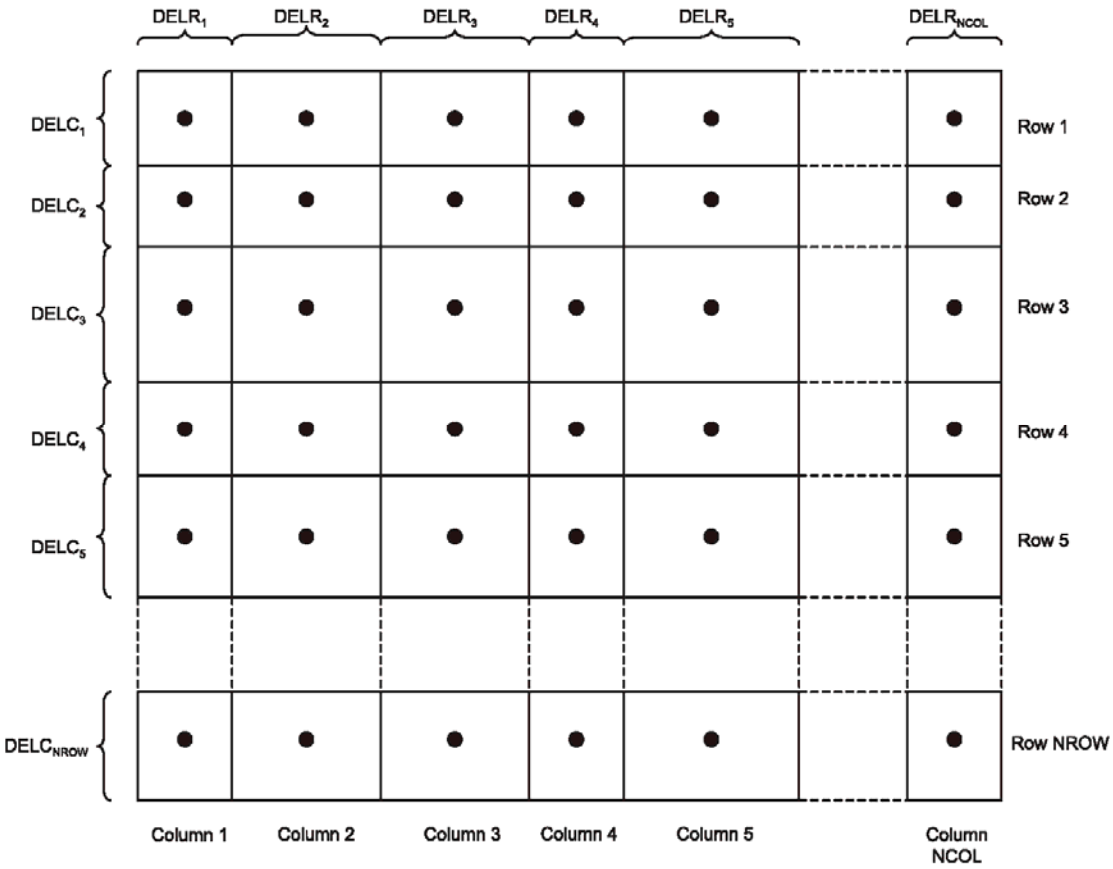


Figure 8-1. Plan view of grid showing DELR and DELC values.

Explanation of Variables Read from the Discretization File:

Text—is a character variable (199 characters) that starts in column 2. Any characters can be included in Text. The "#" character must be in column 1. Lines beginning with # are restricted to the first lines of the file. Text is written to the Listing File.

NLAY—is the number of layers in the model grid.

NROW—is the number of rows in the model grid.

NCOL—is the number of columns in the model grid.

NPER—is the number of stress periods in the simulation.

ITMUNI—indicates the time unit of model data, which must be consistent for all data values that involve time. For example, if years is the chosen time unit, then stress-period length, time-step length, transmissivity, and so on, must all be expressed using years for their time units. Note that the program will still run even if "undefined" time units are specified because the fundamental equations used in MODFLOW do not require that the time unit be identified; but be sure to use consistent units for all input data even when ITMUNI indicates an undefined time unit. When the time unit is defined, MODFLOW uses it to print a table of elapsed simulation time:

```
0 - undefined          3 - hours
1 - seconds            4 - days
2 - minutes            5 - years
```

LENUNI—indicates the length unit of model data, which must be consistent for all data values that involve length. For example, if feet is the chosen length unit, grid spacing, head, hydraulic conductivity, water volumes, and so forth, must all be expressed using feet for their length units. Note that the program will still run even if "undefined" length units are specified because the fundamental equations used in MODFLOW do not require that the length unit be identified; but be sure to use consistent units for all input data even when LENUNI indicates an undefined length unit:

```
0 - undefined
1 - feet
2 - meters
3 - centimeters
```

LAYCBD—is a flag, with one value for each model layer, that indicates whether or not a layer has a Quasi-3D confining bed below it. 0 indicates no confining bed, and not zero indicates a confining bed. LAYCBD for the bottom layer must be 0.

DELR—is the cell width along rows. Read one value for each of the NCOL columns. This is a multi-value one-dimensional variable with one value for each column. (See figure 8-1.)

DELC—is the cell width along columns. Read one value for each of the NROW rows. This is a multi-value one-dimensional variable with one value for each row. (See figure 8-1.)

Top—is the top elevation of layer 1. For the common situation in which the top layer represents a water-table aquifer, setting Top equal to land-surface elevation may be reasonable.

BOTM—is the bottom elevation of a model layer or a Quasi-3d confining bed.

PERLEN—is the length of a stress period.

NSTP—is the number of time steps in a stress period.
TSMULT—is the multiplier for the length of successive time steps. The length of a time step is calculated by multiplying the length of the previous time step by TSMULT. The length of the first time step, Δt_1, is related to PERLEN, NSTP, and TSMULT by the relation

$$\Delta t_1 = \mathrm{PERLEN}\left(\frac{\mathrm{TSMULT} - 1}{\mathrm{TSMULT}^{\mathrm{NSTP}} - 1}\right).$$

Ss/Tr—is a character variable that indicates whether the stress period is transient or steady state. The only allowed options are "SS" and "TR", but these are case insensitive.

Basic Package File

The Basic (BAS) Package file is specified with "BAS6" as the file type.

FOR EACH SIMULATION

0. [#Text]
Item 0 is optional—"#" must be in column 1. Item 0 can be repeated multiple times.

1. Options (199 text characters)
If there are no options to specify, then a blank line must be included for Item 1.

2. IBOUND(NCOL,NROW) or (NCOL,NLAY) -- U2DINT
If not a cross section, a layer variable is read for each layer in the grid.
If a cross section, NLAY rows of NCOL values are read.

```
3.  HNOFLO  (10-space field unless Item 1 contains 'FREE'.)

4.  STRT(NCOL,NROW) or (NCOL,NLAY) -- U2DREL
```
If not a cross section, a layer variable is read for each layer in the grid.
If a cross section, NLAY rows of NCOL values are read.

Explanation of Variables Read from the BAS Package File:

Text—is a character variable that starts in column 2. The first two comment lines will become variable HEADNG, which is used as a printout title throughout the program. (If there are no comment lines, then HEADNG will be blank.) HEADNG is limited to 80 columns, but subsequent Text lines can be up to 199 columns. Any characters can be included in Text. The "#" character must be in column 1. Lines beginning with # are restricted to the first lines of the file. Text is written to the Listing File.

Options—is a character variable that is scanned for words (separated by one or more spaces) that specify program options. Three options are currently recognized. Unrecognized words are ignored, and a word may be specified in either uppercase or lowercase. A blank line is acceptable and indicates no options.

XSECTION indicates that the model is a 1-row cross section for which STRT and IBOUND should each be read as single two-dimensional variables with dimensions of NCOL and NLAY. Likewise, head and drawdown should be printed and saved in disk files as single two-dimensional variables.

CHTOCH indicates that flow between adjacent constant-head cells should be calculated.

FREE indicates that free format is used for input variables throughout the Basic Package and other packages as indicated in their input instructions. Be sure that all variables read using free format have a non-blank value and that a comma or at least one blank separates all adjacent values.

IBOUND—is the boundary variable. One value is read for every model cell. Usually, these values are read one layer at a time; however, when the XSECTION option is specified, a single two-dimensional variable for the cross section is read. Note that although IBOUND is read as one or more two-dimensional variables, IBOUND is stored internally as a three-dimensional variable.

If IBOUND(J,I,K) < 0, cell J,I,K has a constant head.
If IBOUND(J,I,K) = 0, cell J,I,K is no flow.
If IBOUND(J,I,K) > 0, cell J,I,K is variable head.

HNOFLO—is the value of head to be assigned to all no-flow cells (IBOUND = 0). Because head at no-flow cells is unused in model calculations, this does not affect model results but serves to identify no-flow cells when head is printed. This value is used also as drawdown at no-flow cells if the drawdown option is used. Even if the user does not anticipate having no-flow cells, a value for HNOFLO must be entered.

STRT—is initial (starting) head—that is, head at the beginning of the simulation. STRT must be specified for all simulations, including steady-state simulations. One value is read for every model cell. Usually, these values are read a layer at a time. When the XSECTION option is specified, however, a single two-dimensional variable for the cross section is read. For simulations in which the first stress period is steady state, the values used for STRT generally do not affect the simulation [exceptions may occur if cells go dry and (or) rewet]. The execution time, however, will be less if STRT includes hydraulic heads that are close to the steady-state solution.

Multiplier Array File

Input to define multiplier arrays is read from the file that is specified with "MULT" as the file type. Multiplier arrays can be used to calculate layer variables from parameter values.

FOR EACH SIMULATION

0. [#Text]

Item 0 is optional—"#" must be in column 1. Item 0 can be repeated multiple times.

1. NML

2. MLTNAM [***FUNCTION***]

If Item 2 does not contain the optional ***FUNCTION*** keyword, read Item 3:

3. [RMLT(NCOL,NROW)] - U2DREL

Otherwise, if Item 2 contains the optional ***FUNCTION*** keyword, read Item 4:

4. [MLTNAM1 [op1 MLTNAM2] [op2 MLTNAM3] [op3 MLTNAM4] ...]

Repeat items 2 through 4 for each of the NML multiplier arrays.

Explanation of Variables Read from the Multiplier File:

Text—is a character variable (199 characters) that starts in column 2. Any characters can be included in Text. The "#" character must be in column 1. Lines beginning with # are restricted to the first lines of the file. Text is written to the Listing File.

NML—is the number of multiplier arrays to be defined.

MLTNAM—is the name of a multiplier array. This name can consist of 1 to 10 characters and is not case sensitive. That is, any combination of the same characters with different case are equivalent. The name "NONE" is a reserved word and should not be used for a multiplier array.

FUNCTION —is an optional keyword, which indicates that the multiplier array will be constructed from other multiplier arrays that have already been defined. Construction is by arithmetic combinations of the multipliers; see the explanation below for variable op1, op2, op3 ....

RMLT—is a two-dimensional (one layer) multiplier array.

MLTNAM1, MLTNAM2, MLTNAM3, ...—are the names of multiplier arrays that have already been defined.

op1, op2, op3, ...—are arithmetic operators used to define a multiplier array based on other multiplier arrays. Each operator can be either "+", "-", "*", or "/". Operations are applied from left to right to each array element. The operators must be separated from the multiplier array names by at least one space.

The following example input illustrates the use of the FUNCTION keyword to construct a multiplier array from other multiplier arrays. In this example, three multiplier arrays are defined, and accordingly the first line of the file contains "3". The first two arrays (named M1 and M2) are read using the U2DREL utility array reader (item 3), and the third array (named M3) is defined as the sum of M1 and M2. In this example, a model layer has 5 rows and 4 columns.

```
3
M1
INTERNAL 1.0  (4F6.0)  0
   1.0   1.1    1.2    1.3
   1.0   1.1    1.2    1.3
   2.0   2.2    2.4    2.6
   2.0   2.2    2.4    2.6
   1.0   1.1    1.2    1.3
M2
INTERNAL 1.0  (4F6.0)  0
   5.0   5.1    5.2    5.3
   5.0   5.1    5.2    5.3
   6.0   6.1    6.2    6.3
   6.0   6.1    6.2    6.3
   5.0   5.1    5.2    5.3
M3 FUNCTION
M1 + M2
```

The resulting values for multiplier M3 are:

```
 6.0   6.2    6.4    6.6
 6.0   6.2    6.4    6.6
 8.0   8.3    8.6    8.9
 8.0   8.3    8.6    8.9
 6.0   6.2    6.4    6.6
```

Zone Array File

Input to define zone arrays is read from the file that is specified with "ZONE" as the file type. Zone arrays can be used to specify the cells in a layer variable that are associated with a parameter.

FOR EACH SIMULATION

```
0.  [#Text]
```

Item 0 is optional—"#" must be in column 1. Item 0 can be repeated multiple times.

```
1.  NZN

2.  ZONNAM

3.  IZON(NCOL,NROW) - U2DINT
```

Items 2-3 are repeated for each of the NZN zone arrays.

Explanation of Variables Read from the Zone File:

Text—is a character variable (199 characters) that starts in column 2. Any characters can be included in Text. The "#" character must be in column 1. Lines beginning with # are restricted to the first lines of the file. Text is written to the Listing File.

NZN—is the number of zone arrays to be defined.

ZONNAM—is the name of a zone array. This name can consist of 1 to 10 characters and is not case sensitive; that is, any combination of the same characters with different case are equivalent. The name "ALL" is a reserved word and should not be used for a zone array.

IZON—is a two-dimensional (one layer) zone array.

Output Control Option

Input to the Output Control Option of the Ground-Water Flow Process is read from the file that is specified as type "OC" in the Name File. If no "OC" file is specified, default output control is used. Under the default, head and overall budget are written to the Listing File at the end of every stress period. The default printout format for head and drawdown is 10G11.4.

Output Control data may be specified as words or numeric codes. One of these methods must be used throughout any simulation.

Output Control Using Words

Recognized words are shown in bold italics; these words must be entered exactly as shown except that they may be entered in either uppercase or lowercase. Optional parts of lines are shown in brackets. One or more spaces must separate each word or variable, and the total line length must not exceed 199 characters.

FOR EACH SIMULATION

0. [#Text]
   Item 0 is optional—"#" must be in column 1. Item 0 can be repeated multiple times.

1. Any combination of the following lines:
   HEAD PRINT FORMAT IHEDFM
   Specifies the format for writing head to the Listing File.

   HEAD SAVE FORMAT CHEDFM [***LABEL***]
   Specifies the format for writing head to a file other than the Listing File. Omit this line to obtain a binary (unformatted) file. Binary files usually are smaller than text files, but they are not generally transportable among different computer operating systems or different Fortran compilers.

   HEAD SAVE UNIT IHEDUN
   Specifies the file unit for writing head to a file other than the Listing File.

   DRAWDOWN PRINT FORMAT IDDNFM
   Specifies the format for writing drwadown to the Listing File.

   DRAWDOWN SAVE FORMAT CDDNFM [***LABEL***]
   Specifies the format for writing drawdown to a file other than the Listing File. Omit this line to obtain an unformatted (binary) file. Binary files usually are smaller than text files, but they are not generally transportable among different computer operating systems or different Fortran compilers.

   DRAWDOWN SAVE UNIT IDDNUN
   Specifies the file unit for writing drawdown to a file other than the Listing File.

   IBOUND SAVE FORMAT CBOUFM [***LABEL***]
   Specifies the format for writing IBOUND to a file.

   IBOUND SAVE UNIT IBOUUN
   Specifies the file unit for writing IBOUND to a file.

COMPACT BUDGET [***AUX*** or ***AUXILIARY***]

COMPACT BUDGET indicates that the cell-by-cell budget file(s) will be written in a more compact form than is used in the 1988 version of MODFLOW (referred to as MODFLOW-88)(McDonald and Harbaugh, 1988); however, programs that read these data in the form written by MODFLOW-88 will be unable to read the new compact file. If this option is not used, MODFLOW-2005 will write the files using the MODFLOW-88 form. The optional word ***AUX*** (or ***AUXILIARY***) indicates that auxiliary data that are defined in packages (see input data for the RIV, WEL, DRN, and GHB Packages) should be saved in the budget file along with budget data.

FOR EACH TIME STEP FOR WHICH OUTPUT IS DESIRED

2. ***PERIOD*** IPEROC ***STEP*** ITSOC

3. Any combination of the following lines:

PRINT HEAD [list layers if all layers not desired]
Head is written to the Listing File.

PRINT DRAWDOWN [list layers if all layers not desired]
Drawdown is written to the Listing File.

PRINT BUDGET
Overall volumetric budget is written to the Listing File.

SAVE HEAD [list layers if all layers not desired]
Head is written to a file other than the Listing File.

SAVE DRAWDOWN [list layers if all layers not desired]
Drawdown is written to a file other than the Listing File.

SAVE IBOUND [list layers if all layers not desired]
IBOUND is written to a file other than the Listing File. This option is provided to allow changes in IBOUND to be recorded in simulations where IBOUND changes during a simulation.

SAVE BUDGET
Cell-by-cell budget data are written to the files that are designated in the packages that compute budget terms.

Item 2 and one or more Item-3 lines are specified for each time for which output is desired. These lines must be in the order of increasing simulation time.

Explanation of Variables Read by Output Control Using Words:

Text—is a character variable (199 characters) that starts in column 2. Any characters can be included in Text. The "#" character must be in column 1. Lines beginning with # are restricted to the first lines of the file. Text is written to the Listing File.

IHEDFM—is a code for the format in which heads will be printed. (Positive values indicate wrap format; negative values indicate strip format.)

```
 0 - 10G11.4      11 - 20F5.4
 1 - 11G10.3      12 - 10G11.4
 2 - 9G13.6       13 - 10F6.0
 3 - 15F7.1       14 - 10F6.1
 4 - 15F7.2       15 - 10F6.2
 5 - 15F7.3       16 - 10F6.3
 6 - 15F7.4       17 - 10F6.4
 7 - 20F5.0       18 - 10F6.5
 8 - 20F5.1       19 - 5G12.5
 9 - 20F5.2       20 - 6G11.4
10 - 20F5.3       21 - 7G9.2
```

CHEDFM—is a character value that specifies the format for saving heads, and can only be specified if the word method of output control is used. The format must contain 20 characters or less and must be a valid Fortran format that is enclosed in parentheses. The format must be enclosed in apostrophes if it contains one or more blanks or commas. The optional word ***LABEL*** after the format is used to indicate that each layer of output should be preceded with a line that defines the output (simulation time, the layer being output, and so forth). If there is no line specifying CHEDFM, then heads are written to a binary (unformatted) file. Binary files are usually more compact than text files, but they are not generally transportable among different computer operating systems or different Fortran compilers.

IHEDUN—is the unit number on which heads will be saved.

IDDNFM—is a code for the format in which drawdowns will be printed. The codes are the same as for IHEDFM.

CDDNFM—is a character value that specifies the format for saving drawdown, and can only be specified if the word method of output control is used. The format must contain 20 characters or less and must be a valid Fortran format that is enclosed in parentheses. The format must be enclosed in apostrophes if it contains one or more blanks or commas. The optional word ***LABEL*** after the format is used to indicate that each layer of output should be preceded with a line that defines the output (simulation time, the layer being output, and so forth). If there is no line specifying CDDNFM, then drawdown is written to a binary (unformatted) file. Binary files are usually more compact than text files, but they are not generally transportable among different computer operating systems or different Fortran compilers.

IDDNUN—is the unit number on which drawdowns will be saved.

CBOUFM—is a character value that specifies the format for saving IBOUND, and can only be specified if the word method of output control is used. The format must contain 20 characters or less and must be a valid Fortran format that is enclosed in parentheses. The format must be enclosed in apostrophes if it contains one or more blanks or commas. The optional word ***LABEL*** is used to indicate that each layer of output should be preceded with a line that defines the output (simulation time, the layer being output, and so forth). If there is no line specifying CBOUFM, then IBOUND is written using format (20I4). IBOUND is never written as a binary (unformatted) file.

IBOUUN—is the unit number on which IBOUND will be saved.

IPEROC—is the stress period number at which output is desired.

ITSOC—is the time step number (within a stress period) at which output is desired.

Example Output Control Input Using Words:

```
HEAD PRINT FORMAT 15
HEAD SAVE FORMAT (20F10.3) LABEL
HEAD SAVE UNIT 30
COMPACT BUDGET
DRAWDOWN PRINT FORMAT 14

PERIOD 1 STEP 1
   PRINT HEAD 2 6
   PRINT DRAWDOWN
   PRINT BUDGET
   SAVE BUDGET
   SAVE HEAD

PERIOD 1 STEP 7
   SAVE HEAD 1 3 5
   PRINT DRAWDOWN
   SAVE BUDGET

PERIOD 2 STEP 5
   PRINT HEAD
   PRINT BUDGET
   SAVE BUDGET
   SAVE HEAD
```

Note that the first line cannot be blank, but after the first line blank lines are ignored when the word method is used to specify Output Control data. Indented lines are allowed because of the use of free format input.

Output Control Using Numeric Codes

All variables are free format if the word FREE is specified in Item 1 of the Basic Package input file; otherwise, the variables all have 10-character fields.

FOR EACH SIMULATION

```
0.  [#Text]
```

Item 0 is optional—"#" must be in column 1. Item 0 can be repeated multiple times.

```
1.   IHEDFM     IDDNFM      IHEDUN     IDDNUN
```

FOR EACH TIME STEP

```
2.   INCODE     IHDDFL      IBUDFL     ICBCFL

3.   Hdpr       Ddpr        Hdsv       Ddsv
```

(Item 3 is read 0, 1, or NLAY times, depending on the value of INCODE.)

Explanation of Variables Read by Output Control Using Numeric Codes:

Text—is a character variable (199 characters) that starts in column 2. Any characters can be included in Text. The "#" character must be in column 1. Lines beginning with # are restricted to the first lines of the file. Text is written to the Listing File.

IHEDFM—is a code for the format in which heads will be printed. See the description above in the explanation of variables read by output control using words.

IDDNFM—is a code for the format in which drawdowns will be printed. The codes are the same as for IHEDFM.

IHEDUN—is the unit number on which heads will be saved.

IDDNUN—is the unit number on which drawdowns will be saved.

INCODE—is the code for reading Item 3.
If INCODE < 0, Item 3 flags are used from the last time step. Item 3 is not read.
If INCODE = 0, all layers are treated the same way. Item 3 will consist of one line.
If INCODE > 0, Item 3 will consist of one line for each layer.

IHDDFL—is a head and drawdown output flag. This flag allows Item 3 flags to be specified in an early time step and then used or not used in subsequent time steps. Thus, using IHDDFL to avoid resetting Item 3 flags every time step may be possible.
If IHDDFL = 0, no heads or drawdowns will be printed or saved regardless of which Item 3 flags are specified.
If IHDDFL ≠ 0, heads and drawdowns will be printed or saved according to the Item 3 flags.

IBUDFL—is a budget print flag.
If IBUDFL = 0, overall volumetric budget will not be printed.
If IBUDFL ≠ 0, overall volumetric budget will be printed.

ICBCFL—is a flag for writing cell-by-cell flow data.
If ICBCFL = 0, cell-by-cell flow terms are not written to any file.
If ICBCFL ≠ 0, cell-by-cell flow terms are written to the LIST file or a budget file depending on flags set in the component of flow packages, that is, IWELCB, IRCHCB, and so forth.

Hdpr—is the output flag for head printout.
If Hdpr = 0, head is not printed for the corresponding layer.
If Hdpr ≠ 0, head is printed for the corresponding layer.

Ddpr—is the output flag for drawdown printout.
If Ddpr = 0, drawdown is not printed for the corresponding layer.
If Ddpr ≠ 0, drawdown is printed for the corresponding layer.

Hdsv—is the output flag for head save.
If Hdsv = 0, head is not saved for the corresponding layer.
If Hdsv ≠ 0, head is saved for the corresponding layer.

Ddsv—is the output flag for drawdown save.
If Ddsv = 0, drawdown is not saved for the corresponding layer.
If Ddsv ≠ 0, drawdown is saved for the corresponding layer.

Time-Variant Specified-Head Option

Input to the Time-Variant Specified-Head (CHD) Option is read from the file that has file type "CHD" in the Name File. Optional variables are shown in brackets. All variables are free format if the option “FREE” is specified in the Basic Package input file; otherwise, the non-optional variables have 10-character fields and the optional variables are free format.

Once a cell is made constant head, the cell stays constant head throughout the remainder of the simulation. For example, if a cell is listed in the CHD file as constant head in stress period 1 and not listed in stress period 2, then the cell continues to be constant head in stress period 2 and throughout the remainder of the stress periods. The head is adjusted only in the stress periods in which a cell is listed. For the stress periods in which a constant-head cell is not listed, the head stays at the value that it had at the end of the previous stress period.

FOR EACH SIMULATION

0. [#Text]
Item 0 is optional—"#" must be in column 1. Item 0 can be repeated multiple times.

1. [***PARAMETER*** NPCHD MXL]
This optional item must start with the word "PARAMETER".

2. MXACTC [Option]

3. [PARNAM PARTYP Parval NLST]

4. [Layer Row Column Shdfact Ehdfact [xyz]]
Repeat Items 3 and 4 NPCHD times. Items 3 and 4 are not read if NPCHD is negative or 0.
NLST repetitions of Item 4 are required; they are read by subroutine ULSTRD. (SFAC of the ULSTRD utility subroutine applies to Shdfact and Ehdfact).

FOR EACH STRESS PERIOD

5. ITMP NP

6. Layer Row Column Shead Ehead [xyz]
ITMP repetitions of Item 6 are read by subroutine ULSTRD if ITMP > 0. (SFAC of the ULSTRD utility subroutine applies to Shead and Ehead.) Item 6 is not read if ITMP is negative or 0.

7. [Pname]
(Item 7 is repeated NP times. Item 7 is not read if NP is negative or 0.)

Explanation of Variables Read by the CHD Option:

Text—is a character variable (199 characters) that starts in column 2. Any characters can be included in Text. The "#" character must be in column 1. Lines beginning with # are restricted to the first lines of the file. Text is written to the Listing File.

NPCHD—is the number of constant-head boundary parameters.

MXL—is the maximum number of constant-head-boundary cells that will be defined using parameters.

MXACTC—is the maximum number of constant-head boundary cells in use during any stress period, including those that are defined using parameters.

Option—is an optional list of character values.
"AUXILIARY abc" or "AUX abc"—defines an auxiliary variable, named "abc", which will be read for each constant-head boundary as part of Items 4 and 6. Up to 20 variables can be specified, each of which must be preceded by "AUXILIARY" or "AUX." These variables will not be used by the Ground-Water Flow Process, but they will be available for use by other processes. The auxiliary variable values will be read after the Ehead variable.
"NOPRINT"—specifies that lists of constant-head cells will not be written to the Listing File.
PARNAM—is the name of a parameter. This name can consist of 1 to 10 characters and is not case sensitive; that is, any combination of the same characters with different case will be equivalent.

PARTYP—is the type of parameter to be defined. For the CHD Package, the only allowed parameter type is CHD, which defines values of the start and end head at the boundary.

Parval—is the parameter value. This parameter value may be overridden by a value in the Parameter Value File.

NLST—is the number of constant-head cells that are included in the parameter.

Layer—is the layer number of the constant-head boundary.

Row—is the row number of the constant-head boundary.

Column—is the column number of the constant-head boundary.

Shdfact—is the factor used to calculate the head at the boundary at the start of the stress period from the parameter value. The head is the product of Shdfact and the parameter value.

Ehdfact—is the factor used to calculate the head at the boundary at the end of the stress period from the parameter value. The head is the product of Ehdfact and the parameter value.

ITMP—is a flag and a counter.
If ITMP < 0, non-parameter CHD data from the preceding stress period will be reused.
If ITMP ≥ 0, ITMP is the number of non-parameter constant-head boundaries read for the current stress period.

NP—is the number of parameters in use in the current stress period.

Shead—is the head at the boundary at the start of the stress period.

Ehead—is the head at the boundary at the end of the stress period.

[xyz]—represents the values of the auxiliary variables for a constant-head boundary that have been defined in Item 2. The values of auxiliary variables must be present in each repetition of Items 4 and 6 if they are defined in Item 2. The values must be specified in the order used to define the variables in Item 2.

Pname—is the name of a parameter that is being used in the current stress period. NP parameter names will be read.

Parameter Value File

The Parameter Value File is the file that is specified with "PVAL" as the file type. Parameter values in this file replace parameter values specified in the files where parameters are defined.

FOR EACH SIMULATION

```
0.    [#Text]
```
Item 0 is optional—"#" must be in column 1. Item 0 can be repeated multiple times.

```
1.  NP
```

```
2.  PARNAM    Parval
```
NP repetitions of Item 2 are read.

Explanation of Variables Read from the Parameter Value File:

Text—is a character variable (199 characters) that starts in column 2. Any characters can be included in Text. The "#" character must be in column 1. Lines beginning with # are restricted to the first lines of the file. Text is written to the Listing File.

PARNAM—is the name of a parameter whose value is to be defined. This name can consist of 1 to 10 characters and is not case sensitive. That is, any combination of the same characters with different case will be equivalent.

Parval—is the parameter value. This value overrides the parameter value specified in the file where the parameter is defined.

Block-Centered Flow Package

Input for the Block-Centered Flow (BCF) Package is read from the file that is type "BCF6" in the Name File. The BCF Package is an alternative to the LPF Package. Both packages should not be used simultaneously.

FOR EACH SIMULATION

1. IBCFCB HDRY IWDFLG WETFCT IWETIT IHDWET
   These six variables are free format if the option “FREE” is specified in the Basic Package input file; otherwise, the variables all have 10-character fields.

2. Ltype(NLAY)
   Read one value for each layer. These values are free format if the word FREE is specified in Item 1 of the Basic Package input file; otherwise, the values are read using fixed format fields that are each 2 characters wide with 40 values per line. Use only as many lines as required for the number of model layers.

3. TRPY(NLAY) -- U1DREL

A subset of the following two-dimensional variables is used to describe each layer. The variables needed for each layer depend on the layer-type code (LAYCON, which is defined as part of the Item-2 Ltype), whether the simulation has any transient stress periods (at least one stress period defined in the Discretization File specifies Ss/Tr as “TR”), and if the wetting capability is active (IWDFLG not 0). Unneeded variables must be omitted. In no situation will all variables be required. The required variables (Items 4-9) for layer 1 are read first; then the variables for layer 2 and so forth.

4.	[Sf1(NCOL,NROW)] -- U2DREL	If there is at least one transient stress period.

If LAYCON is 0 or 2 (see Ltype), then read item 5.

5. [Tran(NCOL,NROW)] -- U2DREL.

Otherwise, if LAYCON is 1 or 3 (see Ltype), read item 6.

6. [HY(NCOL,NROW)] -- U2DREL

7.	[Vcont(NCOL,NROW)] -- U2DREL	If not the bottom layer.
8.	[Sf2(NCOL,NROW)] -- U2DREL	If there is at least one transient stress period and LAYCON (see Ltype) is 2 or 3.
9.	[WETDRY(NCOL,NROW)] -- U2DREL	If IWDFLG is not 0 and LAYCON is 1 or 3 (see Ltype).

Explanation of Variables Read by the BCF Packa:ge:

IBCFCB—is a flag and a unit number.
If IBCFCB > 0, cell-by-cell flow terms will be written to this unit number when "SAVE BUDGET" or a non-zero value for ICBCFL is specified in Output Control. The terms that are saved are storage, constant-head flow, and flow between adjacent cells.
If IBCFCB = 0, cell-by-cell flow terms will not be written.
If IBCFCB < 0, cell-by-cell flow for constant-head cells will be written in the listing file when "SAVE BUDGET" or a non-zero value for ICBCFL is specified in Output Control. Cell-by-cell flow to storage and between adjacent cells will not be written to any file.

HDRY—is the head that is assigned to cells that are converted to dry during a simulation. Although this value plays no role in the model calculations, HDRY values are useful as indicators when looking at the resulting heads that are output from the model. HDRY is thus similar to HNOFLO in the Basic Package, which is the value assigned to cells that are no-flow cells at the start of a model simulation.

IWDFLG—is a flag that determines if the wetting capability is active.
If IWDFLG = 0, the wetting capability is inactive.
If IWDFLG is not 0, the wetting capability is active.

WETFCT—is a factor that is included in the calculation of the head that is initially established at a cell when that cell is converted from dry to wet. (See IHDWET.)

IWETIT—is the iteration interval for attempting to wet cells. Wetting is attempted every IWETIT iteration. This applies to outer iterations and not inner iterations. If IWETIT is 0, the value is changed to 1.

IHDWET—is a flag that determines which equation is used to define the initial head at cells that become wet:
If IHDWET = 0, equation 5-32A is used: $h = BOT + WETFCT (h_n - BOT)$
If IHDWET is not 0, equation 5-32B is used: $h = BOT + WETFCT (THRESH)$

Ltype—contains a combined code for each layer that specifies both the layer type (LAYCON) and the method of computing interblock conductance. Use as many lines as needed to enter a value for each layer. Values are two-digit numbers:

The left digit defines the method of calculating interblock transmissivity. The methods are described by Goode and Appel (1992).

0 or blank—harmonic mean (the method used in MODFLOW-88).
1—arithmetic mean
2—logarithmic mean
3—arithmetic mean of saturated thickness and logarithmic-mean hydraulic conductivity.

The right digit defines the layer type (LAYCON), which is the same as in MODFLOW-88:

0—confined—Transmissivity and storage coefficient of the layer are constant for the entire simulation.
1—unconfined—Transmissivity of the layer varies and is calculated from the saturated thickness and hydraulic conductivity. The storage coefficient is constant. This type code is valid only for layer 1.
2—confined/unconfined—Transmissivity of the layer is constant. The storage coefficient may alternate between confined and unconfined values. Vertical flow from above is limited if the layer desaturates.
3—confined/unconfined—Transmissivity of the layer varies and is calculated from the saturated thickness and hydraulic conductivity. The storage coefficient may alternate between confined and unconfined values. Vertical flow from above is limited if the aquifer desaturates.

TRPY—is a one-dimensional variable containing a horizontal anisotropic factor for each layer and is the ratio of transmissivity or hydraulic conductivity (whichever is being used) along a column to transmissivity or hydraulic conductivity along a row. Set to 1.0 for isotropic conditions. This is a single variable with one value per layer. Do not read a variable for each layer—that is, include only one array control line for the entire variable.

Sf1—is the primary storage coefficient. Read only if one or more transient stress periods are specified in the Discretization File. For LAYCON equal to 1, Sf1 will always be specific yield, whereas for LAYCON equal to 2 or 3, Sf1 will always be confined storage coefficient. For LAYCON equal to 0, Sf1 would normally be confined storage coefficient; however, a LAYCON value of 0 also can be used to simulate water-table conditions where drawdowns everywhere are expected to remain a small fraction of the saturated thickness, and where there is no layer above, or flow from above is negligible. In this case, specific yield values would be entered for Sf1.

Tran—is the transmissivity along rows. Tran is multiplied by TRPY to obtain transmissivity along columns. Read only for layers where LAYCON is 0 or 2.

HY—is the hydraulic conductivity along rows. HY is multiplied by TRPY to obtain hydraulic conductivity along columns. Read only for layers where LAYCON is 1 or 3.

Vcont—is the vertical hydraulic conductivity divided by the thickness from a layer to the layer below (also called leakance). The value for a cell is the hydraulic conductivity divided by thickness for the material between the node in that cell and the node in the cell below. Because there is not a layer beneath the bottom layer, Vcont cannot be specified for the bottom layer.

Sf2—is the secondary storage coefficient. Read only for layers where LAYCON is 2 or 3 and only if there are one or more transient stress periods specified in the Discretization File. The secondary storage coefficient is always specific yield.

WETDRY—is a combination of the wetting threshold (THRESH) and a flag to indicate which neighboring cells can cause a cell to become wet. If WETDRY < 0, only the cell below a dry cell can cause the cell to become wet. If WETDRY > 0, the cell below a dry cell and the four horizontally adjacent cells can cause a cell to become wet. If WETDRY is 0, the cell cannot be wetted. The absolute value of WETDRY is the wetting threshold. When the sum of BOT and the absolute value of WETDRY at a dry cell is equaled or exceeded by the head at an adjacent cell, the cell is wetted. Read only if LAYCON is 1 or 3 and IWDFLG is not 0.

Layer-Property Flow Package

Input for the Layer-Property Flow (LPF) Package is read from the file that is type "LPF" in the Name File. Free format is used for reading all values. The LPF Package is an alternative to the BCF Package. The two packages should not be used simultaneously.

FOR EACH SIMULATION

0. [#Text]
Item 0 is optional—"#" must be in column 1. Item 0 can be repeated multiple times.

1. ILPFCB HDRY NPLPF [Options]

2. LAYTYP(NLAY)

3. LAYAVG(NLAY)

4. CHANI(NLAY)

5. LAYVKA(NLAY)

6. LAYWET(NLAY)

7. [WETFCT IWETIT IHDWET]
(Include item 7 only if LAYWET indicates at least one wettable layer.)

8. [PARNAM PARTYP Parval NCLU]

9. [Layer Mltarr Zonarr IZ]
Each repetition of Item 9 is called a parameter cluster. Repeat Item 9 NCLU times.
Repeat Items 8-9 for each parameter to be defined (that is, NPLPF times).

A subset of the following two-dimensional variables is used to describe each layer. All the variables that apply to layer 1 are read first, followed by layer 2, followed by layer 3, and so forth. A variable not required due to simulation options (for example, Ss and Sy for a completely steady-state simulation) must be omitted from the input file.

These variables are either read by the array-reading utility subroutine, U2DREL, or they are defined through parameters. If a variable is defined through parameters, then the variable itself is not read; however, a single line containing a print code is read in place of the control line. The print code determines the format for printing the values of the variable as defined by parameters. The print codes are the same as those used in a control line. If any parameters of a given type are used, parameters must be used to define the corresponding variable for all layers in the model.

10.	HK(NCOL,NROW)	If any HK parameters are included, read only a print code.
11.	[HANI(NCOL,NROW)]	Include item 11 only if CHANI is less than or equal to 0. If any HANI parameters are included, read only a print code.
12.	VKA(NCOL,NROW)	If any VK or VANI parameters are included, read only a print code.
13.	[Ss(NCOL,NROW)]	Include item 13 only if at least one stress period is transient. If there are any SS parameters, read only a print code.

14.	[Sy(NCOL,NROW)]	Include item 14 only if at least one stress period is transient and LAYTYP is not 0. If any SY parameters are included, read only a print code.
15.	[VKCB(NCOL,NROW)]	Include item 15 only if LAYCBD (in the Discretization File) is not 0. If any VKCB parameters are included, read only a print code.
16.	[WETDRY(NCOL,NROW)]	Include item 16 only if LAYWET is not 0 and LAYTYP is not 0.

Explanation of Variables Read by the LPF Package:

Text—is a character variable (199 characters) that starts in column 2. Any characters can be included in Text. The "#" character must be in column 1. Lines beginning with # are restricted to the first lines of the file. Text is written to the Listing File.

ILPFCB—is a flag and a unit number.
If ILPFCB > 0, cell-by-cell flow terms will be written to this unit number when "SAVE BUDGET" or a non-zero value for ICBCFL is specified in Output Control. The terms that are saved are storage, constant-head flow, and flow between adjacent cells.
If ILPFCB = 0, cell-by-cell flow terms will not be written.
If ILPFCB < 0, cell-by-cell flow for constant-head cells will be written in the listing file when "SAVE BUDGET" or a non-zero value for ICBCFL is specified in Output Control. Cell-by-cell flow to storage and between adjacent cells will not be written to any file.

HDRY—is the head that is assigned to cells that are converted to dry during a simulation. Although this value plays no role in the model calculations, HDRY values are useful as indicators when looking at the resulting heads that are output from the model. HDRY is thus similar to HNOFLO in the Basic Package, which is the value assigned to cells that are no-flow cells at the start of a model simulation.

NPLPF—is the number of LPF parameters.

Options—are optional key words that activate options:
STORAGECOEFFICIENT indicates that variable Ss and SS parameters are read as storage coefficient rather than specific storage.
CONSTANTCV indicates that vertical conductance for an unconfined cell is computed from the cell thickness rather than the saturated thickness.
THICKSTRT indicates that layers having a negative LAYTYP are confined, and their cell thickness for conductance calculations will be computed as STRT-BOT rather than TOP-BOT.
NOCVCORRECTION indicates that vertical conductance is not corrected when the vertical flow correction is applied.

LAYTYP—contains a flag for each layer that specifies the layer type.
0 – confined
>0 – convertible
<0 – convertible unless the THICKSTRT option is in effect. When THICKSTRT is in effect, a negative value of LAYTYP indicates that the layer is confined, and its saturated thickness will be computed as STRT-BOT.

LAYAVG—contains a flag for each layer that defines the method of calculating interblock transmissivity.
0—harmonic mean
1—logarithmic mean
2—arithmetic mean of saturated thickness and logarithmic-mean hydraulic conductivity.

CHANI—contains a value for each layer that is a flag or the horizontal anisotropy. If CHANI is less than or equal to 0, then variable HANI defines horizontal anisotropy. If CHANI is greater than 0, then CHANI is the horizontal

anisotropy for the entire layer, and HANI is not read. If any HANI parameters are used, CHANI for all layers must be less than or equal to 0.

LAYVKA—contains a flag for each layer that indicates whether variable VKA is vertical hydraulic conductivity or the ratio of horizontal to vertical hydraulic conductivity.
0—indicates VKA is vertical hydraulic conductivity
not 0—indicates VKA is the ratio of horizontal to vertical hydraulic conductivity, where the horizontal hydraulic conductivity is specified as HK in item 10.

LAYWET—contains a flag for each layer that indicates whether wetting is active.
0—indicates wetting is inactive
not 0—indicates wetting is active

WETFCT—is a factor that is included in the calculation of the head that is initially established at a cell when the cell is converted from dry to wet. (See IHDWET.)
IWETIT—is the iteration interval for attempting to wet cells. Wetting is attempted every IWETIT iteration. If using the PCG solver (Hill, 1990), this applies to outer iterations, not inner iterations. If IWETIT ≤ 0, the value is changed to 1.

IHDWET—is a flag that determines which equation is used to define the initial head at cells that become wet:
If IHDWET = 0, equation 5-32A is used: $h = BOT + WETFCT\ (h_n - BOT)$.
If IHDWET is not 0, equation 5-32B is used: $h = BOT + WETFCT(THRESH)$.

PARNAM—is the name of a parameter to be defined. This name can consist of 1 to 10 characters and is not case sensitive. That is, any combination of the same characters with different case will be equivalent.

PARTYP—is the type of parameter to be defined. For the LPF Package, the allowed parameter types are:
HK—defines variable HK, horizontal hydraulic conductivity
HANI—defines variable HANI, horizontal anisotropy
VK—defines variable VKA for layers for which VKA represents vertical hydraulic conductivity (LAYVKA=0)
VANI—defines variable VKA for layers for which VKA represents vertical anisotropy (LAYVKA≠0)
SS—defines variable Ss, the specific storage
SY—defines variable Sy, the specific yield
VKCB—defines variable VKCB, the vertical hydraulic conductivity of a Quasi-3D confining layer.

Parval—is the parameter value. This parameter value may be overridden by a value in the Parameter Value File.

NCLU—is the number of clusters required to define the parameter. Each repetition of Item 9 is a cluster (variables Layer, Mltarr, Zonarr, and IZ). Each layer that is associated with a parameter usually has only one cluster. For example, parameters which apply to cells in a single layer generally will be defined by just one cluster. However, having more than one cluster for the same layer is acceptable.

Layer—is the layer number to which a cluster definition applies.

Mltarr—is the name of the multiplier array to be used to define variable values that are associated with a parameter. The name “NONE” means that there is no multiplier array, and the variable values will be set equal to Parval.

Zonarr—is the name of the zone array to be used to define the cells that are associated with a parameter. The name “ALL” means that there is no zone array, and all cells in the specified layer are part of the parameter.

IZ—is up to 10 zone numbers (separated by spaces) that define the cells that are associated with a parameter. These values are not used if ZONARR is specified as “ALL”. Values can be positive or negative, but 0 is not allowed. The end of the line, a zero value, or a non-numeric entry terminates the list of values.

HK—is the hydraulic conductivity along rows. HK is multiplied by horizontal anisotropy (see CHANI and HANI) to obtain hydraulic conductivity along columns.

HANI—is the ratio of hydraulic conductivity along columns to hydraulic conductivity along rows, where HK of item 10 specifies the hydraulic conductivity along rows. Thus, the hydraulic conductivity along columns is the product of the values in HK and HANI. Read only if CHANI ≤ 0.

VKA—is either vertical hydraulic conductivity or the ratio of horizontal to vertical hydraulic conductivity depending on the value of LAYVKA. If LAYVKA is 0, VKA is vertical hydraulic conductivity. If LAYVKA is not 0, VKA is the ratio of horizontal to vertical hydraulic conductivity. In this case, HK is divided by VKA to obtain vertical hydraulic conductivity, and values of VKA typically are greater than or equal to 1.0.

Ss—is specific storage unless the STORAGECOEFFICIENT option is used. When STORAGECOEFFICIENT is used, Ss is confined storage coefficient. Read only for a transient simulation (at least one transient stress period).

Sy—is specific yield. Read only for a transient simulation (at least one transient stress period) and if the layer is convertible (LAYTYP is not 0).

VKCB—is the vertical hydraulic conductivity of a quasi-three-dimensional confining bed below a layer. Read only if there is a confining bed. Because the bottom layer cannot have a confining bed, VKCB cannot be specified for the bottom layer.

WETDRY—is a combination of the wetting threshold and a flag to indicate which neighboring cells can cause a cell to become wet. If WETDRY < 0, only the cell below a dry cell can cause the cell to become wet. If WETDRY > 0, the cell below a dry cell and the four horizontally adjacent cells can cause a cell to become wet. If WETDRY is 0, the cell cannot be wetted. The absolute value of WETDRY is the wetting threshold. When the sum of BOT and the absolute value of WETDRY at a dry cell is equaled or exceeded by the head at an adjacent cell, the cell is wetted. Read only if LAYTYP is not 0 and LAYWET is not 0.

Horizontal Flow Barrier Package

Input for the Horizontal Flow Barrier (HFB) Package is read from the file that has file type "HFB6" in the Name File. All variables are read in free format.

FOR EACH SIMULATION

0. [#Text]
Item 0 is optional—"#" must be in column 1. Item 0 can be repeated multiple times.

1. NPHFB MXFB NHFBNP ***[NOPRINT]***
The optional keyword "NOPRINT" specified that lists of flow barriers will not be written to the Listing File.

2. [PARNAM PARTYP Parval NLST]

3. [Layer IROW1 ICOL1 IROW2 ICOL2 Factor]
Repeat Items 2 and 3 NPHFB times. Items 2 and 3 are not read if NPHFB is negative or zero.
NLST repetitions of Item 3 are required; they are read by subroutine ULSTRD. (SFAC of the ULSTRD utility subroutine applies to Factor).

4. Layer IROW1 ICOL1 IROW2 ICOL2 Hydchr
NHFBNP repetitions of Item 4 are read. Item 4 is not read if NHFBNP is negative or zero.

5. NACTHFB

6. Pname
NACTHFB repetitions of Item 6 are read. Item 6 is not read if NACTHFB is negative or zero.

Explanation of Variables Read by the HFB Package:

Text—is a character variable (199 characters) that starts in column 2. Any characters can be included in Text. The "#" character must be in column 1. Lines beginning with # are restricted to the first lines of the file. Text is written to the Listing File.

NPHFB—is the number of horizontal flow barrier parameters to be defined in Items 2 and 3. Note: An HFB parameter must be defined in Items 2 and 3, and made active using Item 6, to have an effect in the simulation.

MXFB—is the maximum number of HFB barriers that will be defined using parameters.

NHFBNP—is the number of HFB barriers not defined by parameters.

PARNAM—is the name of a parameter. This name can consist of 1 to 10 characters and is not case sensitive. That is, any combination of the same characters with different case will be equivalent.

PARTYP—is the type of parameter. For the HFB Package, the only allowed parameter type is HFB, which defines values of the hydraulic characteristic of the barrier.

Parval—is the parameter value. This parameter value may be overridden by a value in the Parameter Value File.

NLST—is the number of horizontal flow barrier cells included in the parameter.

Layer—is the number of the model layer in which the horizontal flow barrier is located.

IROW1—is the row number of the cell on one side of the horizontal flow barrier.

ICOL1—is the column number of the cell on one side of the horizontal flow barrier.

IROW2—is the row number of the cell on the other side of the horizontal flow barrier.

ICOL2—is the column number of the cell on the other side of the horizontal flow barrier.

Factor—is the factor used to calculate hydraulic characteristic from the parameter value. The hydraulic characteristic is the product of Factor and the parameter value.

Hydchr—is the hydraulic characteristic of the horizontal flow barrier. The hydraulic characteristic is the barrier hydraulic conductivity divided by the width of the horizontal flow barrier.

NACTHFB—is the number of active HFB parameters.

Pname—is the name of a parameter to be used in the simulation. NACTHFB parameter names will be read.

River Package

Input to the River (RIV) Package is read from the file that has file type "RIV" in the Name File. Optional variables are shown in brackets. All variables are free format if the option “FREE” is specified in the Basic Package input file; otherwise, the non-optional variables have 10-character fields and the optional variables are free format.

FOR EACH SIMULATION

0. [#Text]
Item 0 is optional—“#” must be in column 1. Item 0 can be repeated multiple times.

1. [***PARAMETER*** NPRIV MXL]
This optional item must start with the word “PARAMETER”.

2. MXACTR IRIVCB [Option]

3. [PARNAM PARTYP Parval NLST [***INSTANCES*** NUMINST]]
Repeat Item 3 combined with the indicated repetitions of Item 4 NPRIV times. Items 3 and 4 are not read if NPRIV is negative or 0.
If PARNAM is to be a time-varying parameter, the keyword “INSTANCES” and a value for NUMINST must be entered.

4a. [INSTNAM]
4b. [Layer Row Column Stage Condfact Rbot [xyz]]
After each Item 3 for which the keyword “INSTANCES” is not entered, read Item 4b and not Item 4a.
After each Item 3 for which the keyword “INSTANCES” is entered, read Item 4a and Item 4b for each instance.
NLST repetitions of Item 4b are required; they are read by subroutine ULSTRD. (SFAC of the ULSTRD utility subroutine applies to Condfact). The NLST repetitions of Item 4b follow each repetition of Item 4a when PARNAM is time varying.

FOR EACH STRESS PERIOD

5. ITMP NP

6. Layer Row Column Stage Cond Rbot [xyz]
ITMP repetitions of Item 6 are read by subroutine ULSTRD if ITMP > 0. (SFAC of the ULSTRD utility subroutine applies to Cond.) Item 6 is not read if ITMP is negative or 0.

7. [Pname [Iname]]
(Item 7 is repeated NP times. Item 7 is not read if NP is negative or 0. Iname is read if Pname is a time-varying parameter.)

Explanation of Variables Read by the RIV Package:

Text—is a character variable (199 characters) that starts in column 2. Any characters can be included in Text. The “#” character must be in column 1. Lines beginning with # are restricted to the first lines of the file. Text is written to the Listing File.

NPRIV—is the number of river parameters.

MXL—is the maximum number of river reaches that will be defined using parameters.

MXACTR—is the maximum number of river reaches in use during any stress period, including those that are defined using parameters.

IRIVCB—is a flag and a unit number.
If IRIVCB > 0, cell-by-cell flow terms will be written to this unit number when "SAVE BUDGET" or a non-zero value for ICBCFL is specified in Output Control.
If IRIVCB = 0, cell-by-cell flow terms will not be written.
If IRIVCB < 0, river leakage for each reach will be written to the listing file when "SAVE BUDGET" or a non-zero value for ICBCFL is specified in Output Control.

Option—is an optional list of character values.
"AUXILIARY abc" or "AUX abc"—defines an auxiliary variable, named "abc", which will be read for each river reach as part of Items 4 and 6. Up to 20 variables can be specified, each of which must be preceded by "AUXILIARY" or "AUX." These variables will not be used by the Ground-Water Flow Process Package, but they will be available for use by other processes. The auxiliary variable values will be read after the Rbot variable.
"NOPRINT"—specifies that lists of river reaches will not be written to the Listing File.

PARNAM—is the name of a parameter. This name can consist of 1 to 10 characters and is not case sensitive. That is, any combination of the same characters with different case will be equivalent.

PARTYP—is the type of parameter. For the RIV Package, the only allowed parameter type is RIV, which defines values of riverbed hydraulic conductance.

Parval—is the parameter value. This parameter value may be overridden by a value in the Parameter Value File.

NLST—is the number of river reaches in a non-time-varying parameter. For a time-varying parameter, NLST is the number of reaches in each instance.

INSTANCES—is an optional keyword that designates a parameter as time varying. The keyword is not case sensitive; that is, any combination of the same characters with different case will be equivalent. If ***INSTANCES*** is present, it must be followed by a value for NUMINST. If ***INSTANCES*** is absent, PARNAM is non-time-varying and NUMINST should not be present.

NUMINST—is the number of instances for a time-varying parameter, where each instance is a list of river reaches and associated properties. If the keyword ***INSTANCES*** is present, it must be followed by a value for NUMINST. If ***INSTANCES*** is absent, NUMINST should not be present.

INSTNAM—is the name of an instance associated with the parameter named in the corresponding Item 3. The instance name can be 1 to 10 characters and is not case sensitive. That is, any combination of the same characters with different case will be equivalent. Instance names must be unique for a parameter, but instance names may be reused for different parameters.

Layer—is the layer number of the cell containing the river reach.

Row—is the row number of the cell containing the river reach.

Column—is the column number of the cell containing the river reach.

Stage—is the head in the river.

Condfact—is the factor used to calculate riverbed hydraulic conductance from the parameter value. The conductance is the product of Condfact and the parameter value.

Rbot—is the elevation of the bottom of the riverbed.

[xyz]—represents the values of the auxiliary variables for a river reach that have been defined in Item 2. The values of auxiliary variables must be present in each repetition of Items 4 and 6 if they are defined in Item 2. The values must be specified in the order used to define the variables in Item 2.

ITMP—is a flag and a counter.
If ITMP < 0, non-parameter river data from the last stress period will be reused.
If ITMP ≥ 0, ITMP will be the number of non-parameter reaches read for the current stress period.

NP—is the number of parameters in use in the current stress period.

Cond—is the riverbed hydraulic conductance.

Pname—is the name of a parameter that is being used in the current stress period. NP parameter names will be read.

Iname—is an instance name that is read only if Pname is a time-varying parameter. Multiple instances of the same parameter are not allowed in a stress period.

Recharge Package

Input to the Recharge (RCH) Package is read from the file that has type "RCH" in the Name File. All single-valued variables are free format if the option “FREE” is specified in the Basic Package input file; otherwise, the non-optional variables have 10-character fields and the optional variables are free format.

FOR EACH SIMULATION

0. [#Text]
Item 0 is optional—“#” must be in column 1. Item 0 can be repeated multiple times.

1. [***PARAMETER*** NPRCH]
This optional item must start with the word “PARAMETER”.

2. NRCHOP IRCHCB

3. [PARNAM PARTYP Parval NCLU [INSTANCES NUMINST]]
Repeat Item 3 combined with the indicated repetitions of Item 4 NPRCH times. Items 3 and 4 are not read if NPRCH is negative or 0.
If PARNAM is to be a time-varying parameter, the keyword “INSTANCES” and a value for NUMINST must be entered.

4a. [INSTNAM]
4b. [Mltarr Zonarr IZ]
After each Item 3 for which the keyword “INSTANCES” is not entered, read Item 4b and not Item 4a.
After each Item 3 for which the keyword “INSTANCES” is entered, read Item 4a and Item 4b for each instance.
NCLU repetitions of Item 4b are required. Each repetition of Item 4b is called a parameter cluster. The NCLU repetitions of Item 4b follow each repetition of Item 4a when PARNAM is time varying.

FOR EACH STRESS PERIOD

5. INRECH INIRCH

6. [RECH(NCOL,NROW)] -- U2DREL if NPRCH=0 and if INRECH ≥ 0

7. [Pname [Iname] [IRCHPF]] -- if NPRCH > 0 and if INRECH > 0
Either Item 6 or Item 7 may be read, but not both. Item 7, if read, is repeated INRECH times. Iname is read if Pname is a time-varying parameter.

8. [IRCH(NCOL,NROW)] -- U2DINT If NRCHOP=2 and if INIRCH >= 0

Explanation of Variables Read by the RCH Package:

Text—is a character variable (199 characters) that starts in column 2. Any characters can be included in Text. The “#” character must be in column 1. Lines beginning with # are restricted to the first lines of the file. Text is written to the Listing File.

NPRCH—is the number of recharge parameters.

NRCHOP—is the recharge option code. Recharge fluxes are defined in a layer variable, RECH, with one value for each vertical column. Accordingly, recharge is applied to one cell in each vertical column, and the option code determines which cell in the column is selected for recharge.
1—Recharge is only to the top grid layer.

2—Vertical distribution of recharge is specified in layer variable IRCH.
3—Recharge is applied to the highest active cell in each vertical column. A constant-head node intercepts recharge and prevents deeper infiltration.

IRCHCB—is a flag and a unit number.
If IRCHCB > 0, cell-by-cell flow terms will be written to this unit number when "SAVE BUDGET" or a non-zero value for ICBCFL is specified in Output Control.
If IRCHCB ≤ 0, cell-by-cell flow terms will not be written.

PARNAM—is the name of a parameter to be defined. This name can consist of 1 to 10 characters and is not case sensitive. That is, any combination of the same characters with different case will be equivalent.

PARTYP—is the type of parameter to be defined. For the RCH Package, the only allowed parameter type is RCH, which defines values of the recharge flux.

Parval—is the parameter value. This parameter value may be overridden by a value in the Parameter Value File.

NCLU—is the number of clusters required to define a non-time-varying parameter or one instance of a time varying parameter. Each repetition of Item 4b is a cluster (variables Mltarr, Zonarr, and IZ). Usually only one cluster is used to define a RCH non-time-varying parameter or an instance of a time-varying parameter; however, more than one cluster is acceptable.

INSTANCES—is an optional keyword that designates a parameter as time varying. The keyword is not case sensitive; that is, any combination of the same characters with different case can be used. If ***INSTANCES*** is present, it must be followed by a value for NUMINST. If ***INSTANCES*** is absent, PARNAM is non-time-varying and NUMINST should not be present.

NUMINST—is the number of instances for a time-varying parameter, where each instance is a list of river reaches and associated properties. If the keyword ***INSTANCES*** is present, it must be followed by a value for NUMINST. If ***INSTANCES*** is absent, NUMINST should not be present.

INSTNAM—is the name of an instance associated with the parameter named in the corresponding Item 3. The instance name can be 1 to 10 characters and is not case sensitive. That is, any combination of the same characters with different case will be equivalent. Instance names must be unique for a parameter, but instance names may be reused for different parameters.

Mltarr—is the name of the multiplier array to be used to define cell values that are determined by a parameter. The name “NONE” means that there is no multiplier array, and the cell values will be set equal to Parval.

Zonarr—is the name of the zone array to be used to define the cells that are associated with a parameter. The name “ALL” means that there is no zone array, and all cells in the layer are associated with the parameter.

IZ—is up to 10 zone numbers (separated by spaces) that define the cells that are associated with a parameter. These values are not used if Zonarr is specified as “ALL.” Values can be positive or negative, but 0 is not allowed. The end of the line, a zero value, or a non-numeric entry terminates the list of values.

INRECH—is the RECH read flag. Its function depends on whether or not parameters are being used.
If no parameters are being used (NPRCH = 0):
If INRECH ≥ 0, a layer variable of recharge fluxes, RECH, is read.
If INRECH < 0, recharge rates from the preceding stress period are used.
If parameters are being used (NPRCH > 0):
If INRECH > 0, INRECH is the number of parameters that will be used to define RECH in the current stress period. Item 7 defines the names of the parameters.

If INRECH < 0, recharge parameters from the preceding stress period are used.
INRECH = 0 is not allowed. That is, when parameters are used, at least one parameter must be specified each stress period.

INIRCH—is the IRCH read flag, which is read only if NRCHOP is two:
If INIRCH ≥ 0, a layer variable of layer numbers (IRCH) is read.
If INIRCH < 0, the variable (IRCH) used in the preceding stress period is reused.

RECH—is the recharge flux (LT^{-1}). Read only if INRECH is greater than or equal to zero and if NPRCH = 0.

Pname—is the name of a parameter that will be used to define the RECH variable in the current stress period. Read INRECH values if NPRCH > 0 and INRECH > 0.

Iname—is an instance name that is read only if Pname is a time-varying parameter. Multiple instances of the same parameter are not allowed in a stress period.

IRCHPF—is an optional format code for printing the RECH variable after it has been defined by parameters. The format codes are the same as those used in the U2DREL array reading utility subroutine.

IRCH—is the layer number variable that defines the layer in each vertical column where recharge is applied. Read only if NRCHOP is two and if INIRCH is greater than or equal to zero.

Well Package

Input to the Well (WEL) Package is read from the file that has type "WEL" in the Name File. Optional variables are shown in brackets. All variables are free format if the option “FREE” is specified in the Basic Package input file; otherwise, the non-optional variables have 10-character fields and the optional variables are free format.

FOR EACH SIMULATION

0. [#Text]
Item 0 is optional—“#” must be in column 1. Item 0 can be repeated multiple times.

1. [***PARAMETER*** NPWEL MXL]
This optional item must start with the word “PARAMETER”.

2. MXACTW IWELCB [Option]

3. [PARNAM PARTYP Parval NLST [***INSTANCES*** NUMINST]]
Repeat Item 3 combined with the indicated repetitions of Item 4 NPWEL times. Items 3 and 4 are not read if NPWEL is negative or 0.
If PARNAM is to be a time-varying parameter, the keyword “INSTANCES” and a value for NUMINST must be entered.

4a. [INSTNAM]
4b. Layer Row Column Qfact [xyz]
After each Item 3 for which the keyword “INSTANCES” is not entered, read Item 4b and not Item 4a.
After each Item 3 for which the keyword “INSTANCES” is entered, read Item 4a and Item 4b for each instance.
NLST repetitions of Item 4b are required; they are read by subroutine ULSTRD. (SFAC of the ULSTRD utility subroutine applies to Qfact). The NLST repetitions of Item 4b follow each repetition of Item 4a when PARNAM is time varying.

FOR EACH STRESS PERIOD

5. ITMP NP

6. Layer Row Column Q [xyz]
(ITMP repetitions of Item 6 are read by subroutine ULSTRD if ITMP > 0. (SFAC of the ULSTRD utility subroutine applies to Q.) Item 6 is not read if ITMP is negative or zero.

7. [Pname [Iname]]
(Item 7 is repeated NP times. It is not read if NP is negative or 0. Iname is read if Pname is a time-varying parameter.)

Explanation of Variables Read by the WEL Package:

Text—is a character variable (199 characters) that starts in column 2. Any characters can be included in Text. The “#” character must be in column 1. Lines beginning with # are restricted to the first lines of the file. Text is written to the Listing File.

NPWEL—is the number of well parameters.

MXL—is the maximum number of wells that will be defined using parameters.

MXACTW—is the maximum number of wells in use during any stress period, including those that are defined using parameters.

IWELCB—is a flag and a unit number.
If IWELCB > 0, cell-by-cell flow terms will be written to this unit number when "SAVE BUDGET" or a non-zero value for ICBCFL is specified in Output Control.
If IWELCB = 0, cell-by-cell flow terms will not be written.
If IWELCB < 0, well recharge for each well will be written to the listing file when "SAVE BUDGET" or a non-zero value for ICBCFL is specified in Output Control.

Option—is an optional list of character values.
"AUXILIARY abc" or "AUX abc"—defines an auxiliary variable, named "abc", which will be read for each well as part of Items 4 and 6. Up to 20 variables can be specified, each of which must be preceded by "AUXILIARY" or "AUX." These variables will not be used by the Ground-Water Flow Process, but they will be available for use by other processes. The auxiliary variable values will be read after the Q variable.
"NOPRINT"—specifies that lists of wells will not be written to the Listing File.

PARNAM—is the name of a parameter. This name can consist of 1 to 10 characters and is not case sensitive. That is, any combination of the same characters with different case will be equivalent.

PARTYP—is the type of parameter. For the WEL Package, the only allowed parameter type is Q, which defines values of the volumetric recharge rate.

Parval—is the parameter value. This parameter value may be overridden by a value in the Parameter Value File.

NLST—is the number of wells in a non-time-varying parameter. For a time-varying parameter, NLST is the number of wells in each instance.

INSTANCES—is an optional keyword that designates a parameter as time varying. The keyword is not case sensitive; that is, any combination of the same characters with different case can be used. If ***INSTANCES*** is present, it must be followed by a value for NUMINST. If ***INSTANCES*** is absent, PARNAM is non-time-varying and NUMINST should not be present.

NUMINST—is the number of instances for a time-varying parameter, where each instance is a list of river reaches and associated properties. If the keyword ***INSTANCES*** is present, it must be followed by a value for NUMINST. If ***INSTANCES*** is absent, NUMINST should not be present.

INSTNAM—is the name of an instance associated with the parameter named in the corresponding Item 3. The instance name can be 1 to 10 characters and is not case sensitive. That is, any combination of the same characters with different case will be equivalent. Instance names must be unique for a parameter, but instance names may be reused for different parameters.

Layer—is the layer number of the model cell that contains the well.

Row—is the row number of the model cell that contains the well.

Column—is the column number of the model cell that contains the well.

Qfact—is the factor used to calculate well recharge rate from the parameter value. The recharge rate is the product of Qfact and the parameter value.

[xyz]—represents the values of the auxiliary variables for a well that have been defined in Item 2. The values of auxiliary variables must be present in each repetition of Items 4 and 6 if they are defined in Item 2. The values must be specified in the order used to define the variables in Item 2.

ITMP—is a flag and a counter.
If ITMP < 0, non-parameter well data from the last stress period will be reused.
If ITMP ≥ 0, ITMP will be the number of non-parameter wells read for the current stress period.

NP—is the number of parameters in use in the current stress period.

Q—is the volumetric recharge rate. A positive value indicates recharge and a negative value indicates discharge (pumping).

Pname—is the name of a parameter that is being used in the current stress period. NP parameter names will be read.

Iname—is an instance name that is read only if Pname is a time-varying parameter. Multiple instances of the same parameter are not allowed in a stress period.

Drain Package

Input to the Drain (DRN) Package is read from the file that has type "DRN" in the Name File. Optional variables are shown in brackets. All variables are free format if the option “FREE” is specified in the Basic Package input file; otherwise, the non-optional variables have 10-character fields and the optional variables are free format.

FOR EACH SIMULATION

0. [#Text]
Item 0 is optional—“#” must be in column 1. Item 0 can be repeated multiple times.

1. [***PARAMETER*** NPDRN MXL]
This optional item must start with the word “PARAMETER”.

2. MXACTD IDRNCB [Option]

3. [PARNAM PARTYP Parval NLST [***INSTANCES*** NUMINST]]
Repeat Item 3 combined with the indicated repetitions of Item 4 NPDRB times. Items 3 and 4 are not read if NPDRN is negative or 0.
If PARNAM is to be a time-varying parameter, the keyword “INSTANCES” and a value for NUMINST must be entered.

4a. [INSTNAM]
4b. [Layer Row Column Elevation Condfact [xyz]]
After each Item 3 for which the keyword “INSTANCES” is not entered, read Item 4b and not Item 4a.
After each Item 3 for which the keyword “INSTANCES” is entered, read Item 4a and Item 4b for each instance.
NLST repetitions of Item 4b are required; they are read by subroutine ULSTRD. (SFAC of the ULSTRD utility subroutine applies to Condfact). The NLST repetitions of Item 4b follow each repetition of Item 4a when PARNAM is time varying.

FOR EACH STRESS PERIOD

5. ITMP NP

6. Layer Row Column Elevation Cond [xyz]
ITMP repetitions of Item 6 are read by subroutine ULSTRD if ITMP > 0. (SFAC of the ULSTRD utility subroutine applies to Cond.) Item 6 is not read if ITMP is negative or 0.

7. [Pname [Iname]]
(Item 7 is repeated NP times. Item 7 is not read if NP is negative or 0. Iname is read if Pname is a time-varying parameter.)

Explanation of Variables Read by the DRN Package:

Text—is a character variable (199 characters) that starts in column 2. Any characters can be included in Text. The “#” character must be in column 1. Lines beginning with # are restricted to the first lines of the file. Text is written to the Listing File.

NPDRN—is the number of drain parameters.

MXL—is the maximum number of drain cells that will be defined using parameters.

MXACTD—is the maximum number of drain cells in use during any stress period, including those that are defined using parameters.

IDRNCB—is a flag and a unit number.
If IDRNCB > 0, cell-by-cell flow terms will be written to this unit number when "SAVE BUDGET" or a non-zero value for ICBCFL is specified in Output Control.
If IDRNCB = 0, cell-by-cell flow terms will not be written.
If IDRNCB < 0, drain leakage for each drain cell will be written to the listing file when "SAVE BUDGET" or a non-zero value for ICBCFL is specified in Output Control.

Option—is an optional list of character values.
"AUXILIARY abc" or "AUX abc"—defines an auxiliary variable, named "abc", which will be read for each drain as part of Items 4 and 6. Up to 20 variables can be specified, each of which must be preceded by "AUXILIARY" or "AUX." These variables will not be used by the Ground-Water Flow Process, but they will be available for use by other processes. The auxiliary variable values will be read after the Cond variable.
"NOPRINT"—specifies that lists of drains will not be written to the Listing File.

PARNAM—is the name of a parameter. This name can consist of 1 to 10 characters and is not case sensitive. That is, any combination of the same characters with different case will be equivalent.

PARTYP—is the type of parameter. For the DRN Package, the only allowed parameter type is DRN, which defines values of the drain hydraulic conductance.

Parval—is the parameter value. This parameter value may be overridden by a value in the Parameter Value File.

NLST—is the number of drain in a non-time-varying parameter. For a time-varying parameter, NLST is the number of drain cells in each instance.

INSTANCES—is an optional keyword that designates a parameter as time varying. The keyword is not case sensitive; that is, any combination of the same characters with different case can be used. If ***INSTANCES*** is present, it must be followed by a value for NUMINST. If ***INSTANCES*** is absent, PARNAM is non-time-varying and NUMINST should not be present.

NUMINST—is the number of instances for a time-varying parameter, where each instance is a list of river reaches and associated properties. If the keyword ***INSTANCES*** is present, it must be followed by a value for NUMINST. If ***INSTANCES*** is absent, NUMINST should not be present.

INSTNAM—is the name of an instance associated with the parameter named in the corresponding Item 3. The instance name can be 1 to 10 characters and is not case sensitive. That is, any combination of the same characters with different case will be equivalent. Instance names must be unique for a parameter, but instance names may be reused for different parameters.

Layer—is the layer number of the cell containing the drain.

Row—is the row number of the cell containing the drain.

Column—is the column number of the cell containing the drain.

Elevation—is the elevation of the drain.

Condfact—is the factor used to calculate drain hydraulic conductance from the parameter value. The conductance is the product of Condfact and the parameter value.

[xyz]—represents the values of the auxiliary variables for a drain that have been defined in Item 2. The values of auxiliary variables must be present in each repetition of Items 4 and 6 if they are defined in Item 2. The values must be specified in the order used to define the variables in Item 2.

ITMP—is a flag and a counter.
If ITMP < 0, non-parameter drain data from the last stress period will be reused.
If ITMP ≥ 0, ITMP will be the number of non-parameter drains read for the current stress period.

NP—is the number of parameters in use in the current stress period.

Cond—is the hydraulic conductance of the interface between the aquifer and the drain.

Pname—is the name of a parameter that is being used in the current stress period. NP parameter names will be read.

Iname—is an instance name that is read only if Pname is a time-varying parameter. Multiple instances of the same parameter are not allowed in a stress period.

Evapotranspiration Package

Input to the Evapotranspiration (EVT) Package is read from the file that is type "EVT" in the Name File. All single-valued variables are free format if the option “FREE” is specified in the Basic Package input file; otherwise, the non-optional variables have 10-character fields and the optional variables are free format.

FOR EACH SIMULATION

0. [#Text]
Item 0 is optional—“#” must be in column 1. Item 0 can be repeated multiple times.

1. [***PARAMETER*** NPEVT]
This optional item must start with the word “PARAMETER”.

2. NEVTOP IEVTCB

3. [PARNAM PARTYP Parval NCLU [INSTANCES NUMINST]]
Repeat Item 3 combined with the indicated repetitions of Item 4 NPEVT times. Items 3 and 4 are not read if NPEVT is negative or 0.
If PARNAM is to be a time-varying parameter, the keyword “INSTANCES” and a value for NUMINST must be entered.

4a. [INSTNAM]
4b. [Mltarr Zonarr IZ]
After each Item 3 for which the keyword “INSTANCES” is not entered, read Item 4b and not Item 4a.
After each Item 3 for which the keyword “INSTANCES” is entered, read Item 4a and Item 4b for each instance.
NCLU repetitions of Item 4b are required. Each repetition of Item 4 is called a parameter cluster. The NCLU repetitions of Item 4b follow each repetition of Item 4a when PARNAM is time varying.

FOR EACH STRESS PERIOD

5. INSURF INEVTR INEXDP INIEVT

6. [SURF(NCOL,NROW)] -- U2DREL If INSURF ≥ 0

7. [EVTR(NCOL,NROW)] -- U2DREL If NPEVT = 0 and if INEVTR ≥ 0

8. [Pname [Iname] [IEVTPF]] -- if NPEVT > 0 and if INEVTR > 0
Either Item 7 or Item 8 may be read, but not both. Item 8, if read, is repeated INEVTR times. Iname is read if Pname is a time-varying parameter.

9. [EXDP(NCOL,NROW)] -- U2DREL If INEXDP ≥ 0

10. [IEVT(NCOL,NROW)] -- U2DINT If NEVTOP = 2 and if INIEVT ≥ 0

Explanation of Variables Read by the EVT Package:

Text—is a character variable (199 characters) that starts in column 2. Any characters can be included in Text. The “#” character must be in column 1. Lines beginning with # are restricted to the first lines of the file. Text is written to the Listing File.

NPEVT—is the number of evapotranspiration parameters.

NEVTOP—is the evapotranspiration (ET) option code. ET variables (ET surface, maximum ET rate, and extinction depth) are specified in layer variables, SURF, EVTR, and EXDP, with one value for each vertical column. Accordingly, ET is calculated for one cell in each vertical column. The option codes determine the cell within a column for which ET will be calculated.

1—ET is calculated only for cells in the top grid layer.

2—The cell for each vertical column is specified by the user in variable IEVT.

IEVTCB—is a flag and a unit number.

If IEVTCB > 0, cell-by-cell flow terms will be written to this unit number when "SAVE BUDGET" or a non-zero value for ICBCFL is specified in Output Control.

If IEVTCB ≤ 0, cell-by-cell flow terms will not be written.

PARNAM—is the name of a parameter to be defined. This name can consist of 1 to 10 characters and is not case sensitive; that is, any combination of the same characters with different case will be equivalent.

PARTYP—is the type of parameter to be defined. For the EVT Package, the only allowed parameter type is EVT, which defines values of the maximum ET flux, variable EVTR.

Parval—is the parameter value. This parameter value may be overridden by a value in the Parameter Value File.

NCLU—is the number of clusters required to define a non-time-varying parameter or one instance of a time-varying parameter. Each repetition of Item 4b is a cluster (variables Mltarr, Zonarr, and IZ). Usually only one cluster is used to define an EVT non-time-varying parameter or an instance of a time-varying parameter; however, more than one cluster is acceptable.

INSTANCES—is an optional keyword that designates a parameter as time varying. The keyword is not case sensitive; that is, any combination of the same characters with different case can be used. If ***INSTANCES*** is present, it must be followed by a value for NUMINST. If ***INSTANCES*** is absent, PARNAM is non-time-varying and NUMINST should not be present.

NUMINST—is the number of instances for a time-varying parameter, where each instance is a list of river reaches and associated properties. If the keyword ***INSTANCES*** is present, it must be followed by a value for NUMINST. If ***INSTANCES*** is absent, NUMINST should not be present.

INSTNAM—is the name of an instance associated with the parameter named in the corresponding Item 3. The instance name can be 1 to 10 characters and is not case sensitive. That is, any combination of the same characters with different case will be equivalent. Instance names must be unique for a parameter, but instance names may be reused for different parameters.

Mltarr—is the name of the multiplier array to be used to define the values that are determined by a parameter. The name “NONE” means that there is no multiplier array, and the values will be set equal to Parval.

Zonarr—is the name of the zone array to be used to define the cells that are associated with a parameter. The name “ALL” means that there is no zone array, and all cells are associated with the parameter.

IZ—is up to 10 zone numbers (separated by spaces) that define the cells that are associated with a parameter. These values are not used if Zonarr is specified as “ALL.” Values can be positive or negative, but 0 is not allowed. The end of the line, a zero value, or a non-numeric entry terminates the list of values.

INSURF—is the ET surface (SURF) read flag.

If INSURF ≥ 0, a layer variable containing the ET surface elevation (SURF) will be read.

If INSURF < 0, the ET surface from the preceding stress period will be reused.

INEVTR—is the EVTR read flag. Its function depends on whether or not parameters are being used.
If no parameters are being used (NPEVT=0):
If INEVTR ≥ 0, a layer variable containing the maximum ET rate (EVTR) will be read.
If INEVTR < 0, the maximum ET rate from the preceding stress period will be reused.
If parameters are being used (NPEVT>0):
If INEVTR > 0, INEVTR is the number of parameters that will be used to define EVTR in the current stress period. Item 8 defines the names of the parameters.
If INEVTR < 0, EVT parameters from the preceding stress period are used.
INEVTR = 0 is not allowed. That is, when parameters are used, at least one parameter must be specified each stress period

INEXDP—is the extinction depth (EXDP) read flag.
If INEXDP ≥ 0, a layer variable containing the extinction depth (EXDP) will be read.
If INEXDP < 0, the extinction depth from the preceding stress period will be reused.

INIEVT—is the layer indicator (IEVT) read flag that is read only if the ET option (NEVTOP) is equal to two.
If INIEVT ≥ 0, a layer variable containing the layer indicators (IEVT) will be read.
If INIEVT < 0, layer indicators used during the preceding stress period will be reused.

SURF—is the elevation of the ET surface. This variable is read only if INSURF ≥ 0

EVTR—is the maximum ET flux [volumetric flow rate per unit area (LT^{-1})]. This variable is read only if INEVTR ≥ 0 and if NPEVT=0. Contrary to the usual convention in MODFLOW, EVTR values should be specified as positive values even though they represent an outflow from the ground-water system.

Pname—is the name of a parameter that will be used to define the EVTR variable in the current stress period. Read INEVTR values if NPEVT > 0 and INEVTR > 0.

Iname—is an instance name that is read only if Pname is a time-varying parameter. Multiple instances of the same parameter are not allowed in a stress period.

IEVTPF—is an optional format code for printing the EVTR variable after it has been defined by parameters. The format codes are the same as those used in the U2DREL array reading utility subroutine.

EXDP—is the ET extinction depth. This variable is read only if INEXDP ≥ 0.

IEVT—is the layer indicator variable. For each horizontal location, IEVT indicates the layer from which ET is removed. IEVT is read only if the ET option is equal to two and if INIEVT ≥ 0.

General-Head Boundary Package

Input to the General-Head Boundary (GHB) Package is read from the file that has file type "GHB" in the Name File. Optional variables are shown in brackets. All variables are free format if the option “FREE” is specified in the Basic Package input file; otherwise, the non-optional variables have 10-character fields and the optional variables are free format.

FOR EACH SIMULATION

0. [#Text]
Item 0 is optional—“#” must be in column 1. Item 0 can be repeated multiple times.

1. [***PARAMETER*** NPGHB MXL]
This optional item must start with the word “PARAMETER”.

2. MXACTB IGHBCB [Option]

3. [PARNAM PARTYP Parval NLST [***INSTANCES*** NUMINST]]
Repeat Item 3 combined with the indicated repetitions of Item 4 NPGHB times. Items 3 and 4 are not read if NPGHB is negative or 0.
If PARNAM is to be a time-varying parameter, the keyword “INSTANCES” and a value for NUMINST must be entered.

4a. [INSTNAM]
4b. [Layer Row Column Bhead Condfact [xyz]]
After each Item 3 for which the keyword “INSTANCES” is not entered, read Item 4b and not Item 4a.
After each Item 3 for which the keyword “INSTANCES” is entered, read Item 4a and Item 4b for each instance.
NLST repetitions of Item 4b are required; they are read by subroutine ULSTRD. (SFAC of the ULSTRD utility subroutine applies to Condfact). The NLST repetitions of Item 4b follow each repetition of Item 4a when PARNAM is time varying.

FOR EACH STRESS PERIOD

5. ITMP NP

6. Layer Row Column Bhead Cond [xyz]
ITMP repetitions of Item 6 are read by subroutine ULSTRD if ITMP > 0. (SFAC of the ULSTRD utility subroutine applies to Cond.) Item 6 is not read if ITMP is negative or 0.

7. [Pname [Iname]]
(Item 7 is repeated NP times. Item 7 is not read if NP is negative or 0. Iname is read if Pname is a time-varying parameter.)

Explanation of Variables Read by the GHB Package:

Text—is a character variable (199 characters) that starts in column 2. Any characters can be included in Text. The “#” character must be in column 1. Lines beginning with # are restricted to the first lines of the file. Text is written to the Listing File.

NPGHB—is the number of general-head boundary parameters.

MXL—is the maximum number of general-head-boundary cells that will be defined using parameters.

MXACTB—is the maximum number of general-head boundary cells in use during any stress period, including those that are defined using parameters.

IGHBCB—is a flag and a unit number.
If IGHBCB > 0, cell-by-cell flow terms will be written to the unit number when "SAVE BUDGET" or a non-zero value for ICBCFL is specified in Output Control.
If IGHBCB = 0, cell-by-cell flow terms will not be written.
If IGHBCB < 0, boundary leakage for each GHB cell will be written to the listing file when "SAVE BUDGET" or a non-zero value for ICBCFL is specified in Output Control.

Option—is an optional list of character values.
"AUXILIARY abc" or "AUX abc"—defines an auxiliary variable, named "abc", which will be read for each general-head boundary as part of Items 4 and 6. Up to 20 variables can be specified, each of which must be preceded by "AUXILIARY" or "AUX." These variables will not be used by the Ground-Water Flow Process, but they will be available for use by other processes. The auxiliary variable values will be read after the Cond variable.
"NOPRINT"—specifies that lists of general-head boundary cells will not be written to the Listing File.

PARNAM—is the name of a parameter. This name can consist of 1 to 10 characters and is not case sensitive; that is, any combination of the same characters with different case will be equivalent.

PARTYP—is the type of parameter to be defined. For the GHB Package, the only allowed parameter type is GHB, which defines values of the general-head boundary hydraulic conductance.

Parval—is the parameter value. This parameter value may be overridden by a value in the Parameter Value File.

NLST—is the number of head-dependent boundaries in a non-time-varying parameter. For a time-varying parameter, NLST is the number of head-dependent boundaries in each instance.

INSTANCES—is an optional keyword that designates a parameter as time varying. The keyword is not case sensitive; that is, any combination of the same characters with different case can be used. If ***INSTANCES*** is present, it must be followed by a value for NUMINST. If ***INSTANCES*** is absent, PARNAM is non-time-varying and NUMINST should not be present.

NUMINST—is the number of instances for a time-varying parameter, where each instance is a list of river reaches and associated properties. If the keyword ***INSTANCES*** is present, it must be followed by a value for NUMINST. If ***INSTANCES*** is absent, NUMINST should not be present.

INSTNAM—is the name of an instance associated with the parameter named in the corresponding Item 3. The instance name can be 1 to 10 characters and is not case sensitive. That is, any combination of the same characters with different case will be equivalent. Instance names must be unique for a parameter, but instance names may be reused for different parameters.

Layer—is the layer number of the cell affected by the head-dependent boundary.

Row—is the row number of the cell affected by the head-dependent boundary.

Column—is the column number of the cell affected by the head-dependent boundary.

Bhead—is the boundary head.

Condfact—is the factor used to calculate hydraulic conductance from the parameter value. The conductance is the product of Condfact and the parameter value.

[xyz]—represents the values of the auxiliary variables for a boundary that have been defined in Item 2. The values of auxiliary variables must be present in each repetition of Items 4 and 6 if they are defined in Item 2. The values must be specified in the order used to define the variables in Item 2.

ITMP—is a flag and a counter.

If ITMP < 0, non-parameter GHB data from the preceding stress period will be reused.

If ITMP ≥ 0, ITMP is the number of non-parameter general-head boundaries read for the current stress period.

NP—is the number of parameters in use in the current stress period.

Cond—is the hydraulic conductance of the interface between the aquifer cell and the boundary.

Pname—is the name of a parameter that is being used in the current stress period. NP parameter names will be read.

Iname—is an instance name that is read only if Pname is a time-varying parameter. Multiple instances of the same parameter are not allowed in a stress period.

Strongly Implicit Procedure Package

Input to the Strongly Implicit Procedure (SIP) Package is read from the file that is type "SIP" in the Name File. All numeric variables are free format if the option “FREE” is specified in the Basic Package input file; otherwise, all the variables have 10-character fields.

FOR EACH SIMULATION

```
0.   [#Text]
```

Item 0 is optional—“#” must be in column 1. Item 0 can be repeated multiple times.

```
1. MXITER     NPARM

2. ACCL       HCLOSE      IPCALC      WSEED      IPRSIP
```

Explanation of Variables Read by the SIP Package:

Text—is a character variable (199 characters) that starts in column 2. Any characters can be included in Text. The “#” character must be in column 1. Lines beginning with # are restricted to the first lines of the file. Text is written to the Listing File.

MXITER—is the maximum number of times through the iteration loop in one time step in an attempt to solve the system of finite-difference equations.

NPARM—is the number of iteration variables to be used. Five variables are generally sufficient.

ACCL—is the acceleration variable, which must be greater than zero and is generally equal to one. If a zero is entered, it is changed to one.

HCLOSE—is the head change criterion for convergence. When the maximum absolute value of head change from all nodes during an iteration is less than or equal to HCLOSE, iteration stops.

IPCALC—is a flag indicating where the seed for calculating iteration variables will come from.
0—the seed entered by the user will be used.
1—the seed will be calculated at the start of the simulation from problem variables.

WSEED—is the seed for calculating iteration variables. WSEED is always read, but is used only if IPCALC is equal to zero.

IPRSIP—is the printout interval for SIP. IPRSIP, if equal to zero, is changed to 999. The maximum head change (positive or negative) is printed for each iteration of a time step whenever the time step is an even multiple of IPRSIP. This printout also occurs at the end of each stress period regardless of the value of IPRSIP.

Preconditioned Conjugate-Gradient Package

Input to the Preconditioned Conjugate-Gradient (PCG) Package is read from the file that is type "PCG" in the Name File. All numeric variables are free format if the option “FREE” is specified in the Basic Package input file; otherwise, all the variables have 10-character fields.

FOR EACH SIMULATION

```
0.   [#Text]
```

Item 0 is optional—“#” must be in column 1. Item 0 can be repeated multiple times.

```
1. MXITER    ITER1    NPCOND

2. HCLOSE    RCLOSE   RELAX   NBPOL   IPRPCG  MUTPCG   DAMP
```

Explanation of Variables Read by the PCG Package:

Text—is a character variable (199 characters) that starts in column 2. Any characters can be included in Text. The “#” character must be in column 1. Lines beginning with # are restricted to the first lines of the file. Text is written to the Listing File.

MXITER—is the maximum number of outer iterations. For a linear problem MXITER should be 1, unless more than 50 inner iterations are required, when MXITER could be as large as 10. A larger number (generally less than 100) is required for a nonlinear problem.

ITER1—is the number of inner iterations. For nonlinear problems, ITER1 usually ranges from 10 to 30; a value of 30 will be sufficient for most linear problems.

NPCOND—is the flag used to select the matrix conditioning method:
1—is for Modified Incomplete Cholesky (for use on scalar computers)
2—is for Polynomial (for use on vector computers or to conserve computer memory)

HCLOSE—is the head change criterion for convergence, in units of length. When the maximum absolute value of head change from all nodes during an iteration is less than or equal to HCLOSE, and the criterion for RCLOSE is also satisfied, iteration stops.

RCLOSE—is the residual criterion for convergence, in units of cubic length per time. The units for length and time are the same as established for all model data. (See LENUNI and ITMUNI input variables in the Discretization File.) When the maximum absolute value of the residual at all nodes during an iteration is less than or equal to RCLOSE, and the criterion for HCLOSE is also satisfied, iteration stops.

For nonlinear problems, convergence is achieved when the convergence criteria are satisfied for the first inner iteration.

RELAX—is the relaxation parameter used with NPCOND = 1. Usually, RELAX = 1.0, but for some problems a value of 0.99, 0.98, or 0.97 will reduce the number of iterations required for convergence. RELAX is not used if NPCOND is not 1.

NBPOL—is used when NPCOND = 2 to indicate whether the estimate of the upper bound on the maximum eigenvalue is 2.0, or whether the estimate will be calculated. NBPOL = 2 is used to specify the value is 2.0; for any other value of NBPOL, the estimate is calculated. Convergence is generally insensitive to this NBPOL. NBPOL is not used if NBPOL is not 2.

IPRPCG—is the printout interval for PCG. IPRPCG, if equal to zero, is changed to 999. The maximum head change (positive or negative) and residual change are printed for each iteration of a time step whenever the time step is an even multiple of IPRPCG. This printout also occurs at the end of each stress period regardless of the value of IPRPCG.

MUTPCG—is a flag that controls printing of convergence information from the solver:
- 0—is for printing tables of maximum head change and residual each iteration
- 1—is for printing only the total number of iterations
- 2—is for no printing
- 3—is for printing only if convergence fails

DAMP—is the damping factor. This variable typically is set equal to one, which indicates no damping. A value less than 1 and greater than 0 causes damping.

Direct Solver Package

Input to the Direct Solver (DE4) Package is read from the file that is type "DE4" in the Name File. All numeric variables are free format.

FOR EACH SIMULATION

```
0.    [#Text]
```

Item 0 is optional—"#" must be in column 1. Item 0 can be repeated multiple times.

```
1.  ITMX     MXUP     MXLOW     MXBW

2.  IFREQ    MUTD4    ACCL      HCLOSE    IPRD4
```

Explanation of Variables Read by the DE4 Package:

Text—is a character variable (199 characters) that starts in column 2. Any characters can be included in Text. The "#" character must be in column 1. Lines beginning with # are restricted to the first lines of the file. Text is written to the Listing File.

ITMX—is the maximum number of iterations each time step. Specify ITMX = 1 if iteration is not desired. Ideally, iteration would not be required for direct solution; however, iteration is necessary if the flow equation is nonlinear (see explanation for IFREQ = 3) or if computer precision limitations result in inaccurate calculations as indicated by a large water-budget error. For a nonlinear flow equation, each iteration is equally time consuming because [A] (eq. 2-27) changes each iteration and Gaussian elimination is required after each change. This is called external iteration. For a linear equation, iteration is substantially faster because [A] is changed at most once per time step; thus, Gaussian elimination is required at most once per time step. This is called internal iteration.

MXUP—is the maximum number of equations in the upper part of the equations to be solved. This value affects the amount of memory used by the DE4 Package. If specified as 0, the program will calculate MXUP as half the number of cells in the model, which is an upper limit. The actual number of equations in the upper part will be less than half the number of cells whenever no-flow and constant-head cells are indicated because flow equations are not formulated for these cells. The DE4 Package prints the actual number of equations in the upper part when it runs. The printed value can be used for MXUP in future runs to minimize memory usage.

MXLOW—is the maximum number of equations in the lower part of equations to be solved. This value affects the amount of memory used by the DE4 Package. If specified as 0, the program will calculate MXLOW as half the number of cells in the model, which is an upper limit. The actual number of equations in the lower part will be less than half the number of cells whenever no-flow and constant-head cells are included because flow equations are not formulated for these cells. The DE4 Package prints the actual number of equations in the lower part when it runs. The printed value can be used for MXLOW in future runs to minimize memory usage.

MXBW—is the maximum band width plus 1 of the [AL] matrix. This value affects the amount of memory used by the DE4 Package. If specified as 0, the program will calculate MXBW as the product of the two smallest grid dimensions plus 1, which is an upper limit. The DE4 Package prints the actual band width plus 1 when it runs. The printed value can be used for MXBW in future runs to minimize memory usage.

IFREQ—is a flag indicating the frequency at which coefficients in [A] change. This affects the efficiency of solution; significant work can be avoided if the user knows that [A] remains constant all or part of the time.

IFREQ = 1 indicates that the flow equations are linear and that coefficients of simulated head for all stress terms are constant for all stress periods. To meet the linearity requirement, all model layers must be confined (which is specified in the Block-Centered Flow Package by setting LAYCON equal to 0 for all layers or in the Layer-Property Flow Package by setting LAYTYP equal to 0 for all layers), and formulations must not change based upon head (such as seepage from a river changing from head dependent flow to a constant flow when head drops below the bottom of the riverbed). Examples of coefficients of simulated head for stress terms are riverbed conductance, drain conductance, maximum evapotranspiration rate, evapotranspiration extinction depth, and general-head boundary conductance.

IFREQ = 2 indicates that the flow equations are linear as described for IFREQ=1, but coefficients of simulated head for some stress terms may change at the start of each stress period. Examples of coefficients of simulated head for stress terms are riverbed conductance, drain conductance, maximum evapotranspiration rate, evapotranspiration extinction depth, and general-head boundary conductance. For a simulation consisting of only one stress period, IFREQ = 2 has the same meaning as IFREQ = 1.

IFREQ = 3 indicates that a nonlinear flow equation is being solved, which means that some terms in [A] depend on simulated head. Examples of head-dependent terms in [A] are transmissivity for water-table layers, which is based on saturated thickness; flow terms for rivers, drains, and evapotranspiration if they convert between head dependent flow and constant flow; and the change in storage coefficient when a cell converts between confined and unconfined. When a nonlinear flow equation is being solved, external iteration (ITMX > 1) is normally required to accurately approximate the nonlinearities. Note that when nonlinearities caused by water-table calculations are part of a simulation, obvious signs may not be present in the output from a simulation that does not use external iteration to indicate that iteration is needed. In particular, the budget error may be acceptably small without iteration even though significant error in head exists because of nonlinearity. Additional information about this issue is contained in Chapter 7.

MUTD4—is a flag that indicates the quantity of information that is printed when convergence information is printed for a time step.

MUTD4 = 0 indicates that the number of iterations in the time step and the maximum head change each iteration are printed.

MUTD4 = 1 indicates that only the number of iterations in the time step is printed.

MUTD4 = 2 indicates no information is printed.

ACCL—is a multiplier for the computed head change for each iteration. Normally this value is 1. A value greater than 1 may be useful for improving the rate of convergence when using external iteration to solve nonlinear problems (IFREQ = 3). ACCL should always be 1 for linear problems. When ITMX = 1, ACCL is changed to 1 regardless of the input value; however, a value must always be specified.

HCLOSE—is the head change closure criterion. If iterating (ITMX > 1), iteration stops when the absolute value of head change at every node is less than or equal to HCLOSE. HCLOSE is not used if not iterating, but a value must always be specified.

IPRD4—is the time step interval for printing out convergence information when iterating (ITMX > 1). For example, if IPRD4 is 2, convergence information is printed every other time step. A value must always be specified even if not iterating.

Input Instructions for Array Reading Utility Subroutines

The array reading utility subroutines provide a common way for all packages to read variables that have multiple values. The term "array" is simply a programming term for a variable that contains multiple values. There are three subroutines: U2DREL, U2DINT, and U1DREL. U2DREL reads real two-dimensional variables, U2DINT reads integer two-dimensional variables, and U1DREL reads real one-dimensional variables. All of these subroutines work similarly. They read one array-control line and, optionally, a data array in a format specified on the array-control line. Several alternate structures for the control line are provided. The original fixed-format control lines work as documented in McDonald and Harbaugh (1988), and four free-format versions have been added. The free-format versions are described first because they are easier to use.

FREE-FORMAT CONTROL LINES FOR ARRAY READERS:

Values in bold italics are keywords that can be specified as uppercase or lowercase. Each control line is limited to a length of 199 characters.

1. ***CONSTANT*** CNSTNT
   All values in the array are set equal to CNSTNT.

2. ***INTERNAL*** CNSTNT FMTIN IPRN
   The individual array elements will be read from the same file that contains the control line.

3. ***EXTERNAL*** Nunit CNSTNT FMTIN IPRN
   The individual array elements will be read from the file unit number specified by Nunit. The name of the file associated with this file unit must be contained in the Name File.

4. ***OPEN/CLOSE*** FNAME CNSTNT FMTIN IPRN
   The array will be read from the file whose name is specified by FNAME. This file will be opened on unit 99 just prior to reading the array and closed immediately after the array is read. This file should not be included in the Name File. A file that is read using this control line can contain only a single array.

FIXED-FORMAT CONTROL LINE FOR ARRAY READERS:

A fixed-format control line contains the following variables:

```
LOCAT      CNSTNT       FMTIN      IPRN
```

These variables are explained below. LOCAT, CNSTNT, and IPRN are 10-character numeric fields. For U2DREL and U1DREL, CNSTNT is a real number. For U2DINT, CNSTNT is an integer and must not include a decimal. FMTIN is a 20-character text field. All four variables are always read when the control line is fixed format; however, some of the variables are unused in some situations. For example when LOCAT = 0, FMTIN and IPRN are not used.

Explanation of Variables in the Control Lines:

CNSTNT—is a real-number constant for U2DREL and U1DREL, and an integer constant for U2DINT. If the array is being defined as a constant, CNSTNT is the constant value. If individual elements of the array are being read, the values are multiplied by CNSTNT after they are read. CNSTNT, when used as a multiplier and specified as 0, is changed to 1.

FMTIN—is the format for reading array elements. The format must contain 20 characters or less. The format must either be a standard Fortran format that is enclosed in parentheses, "(FREE)" which indicates free format, or "(BINARY)" which indicates binary (unformatted) data. When using a free-format control line, the format must be enclosed in apostrophes if it contains one or more blanks or commas. A binary file that can be read by MODFLOW

may be created in only two ways. The first way is to use MODFLOW to create the file by saving heads in a binary file. This is commonly done when the user desires to use computed heads from one simulation as initial heads for a subsequent simulation. The other way to create a binary file is to write a special program that generates a binary file, and compile this program using a Fortran compiler that is compatible with the compiler used to compile MODFLOW. "(FREE)" and "(BINARY)" can only be specified in free-format control lines. Also, "(BINARY)" can be specified only when using U2DREL or U2DINT, and only when the control line is EXTERNAL or OPEN/CLOSE. When the "(FREE)" option is used, be sure that all array elements have a non-blank value and that a comma or at least one blank separates adjacent values.

IPRN—is a flag that indicates whether the array being read should be written to the Listing File after the array has been read and a code for indicating the format that should be used when the array is written. The format codes are different for each of the three array-reading subroutines as shown below. IPRN is set to zero when the specified value exceeds those defined. If IPRN is less than zero, the array will not be printed.

IPRN	U2DREL	U2DINT	U1DREL
0	10G11.4	10I11	10G12.5
1	11G10.3	60I1	5G12.5
2	9G13.6	40I2	
3	15F7.1	30I3	
4	15F7.2	25I4	
5	15F7.3	20I5	
6	15F7.4	10I11	
7	20F5.0	25I2	
8	20F5.1	15I4	
9	20F5.2	10I6	
10	20F5.3		
11	20F5.4		
12	10G11.4		
13	10F6.0		
14	10F6.1		
15	10F6.2		
16	10F6.3		
17	10F6.4		
18	10F6.5		
19	5G12.5		
20	6G11.4		
21	7G9.2		

Nunit—is the unit for reading the array when the ***EXTERNAL*** free-format control line is used.

LOCAT—indicates the location of the array values for a fixed-format array control line. If LOCAT = 0, all elements are set equal to CNSTNT. If LOCAT > 0, it is the unit number for reading formatted lines using FMTIN as the format. If LOCAT < 0, it is the unit number for binary (unformatted) lines, and FMTIN is ignored. Also, when LOCAT is not 0, the array values are multiplied by CNSTNT after they are read.

Examples of Free-Format Control Lines

The following examples use free-format control lines for reading an array. The example array is a real array consisting of 4 rows with 7 columns per row:

```
CONSTANT  5.7      This sets an entire array to the value "5.7".

INTERNAL  1.0  (7F4.0)  3     This reads the array values from the
 1.2 3.7 9.3 4.2 2.2 9.9 1.0  file that contains the control line.
 3.3 4.9 7.3 7.5 8.2 8.7 6.6  Thus, the values immediately follow the
 4.5 5.7 2.2 1.1 1.7 6.7 6.9  control line.
 7.4 3.5 7.8 8.5 7.4 6.8 8.8

EXTERNAL 52  1.0 (7F4.0)  3   This reads the array from the formatted
                              file opened on unit 52.

EXTERNAL 47 1.0 (BINARY)  3   This reads the array from the
                              binary file opened on unit 47.


OPEN/CLOSE  test.dat  1.0  (7F4.0)  3  This reads the array from the
                                       file named "test.dat".
```

Input Instructions for List Utility Subroutine (ULSTRD)

Subroutine ULSTRD reads lists that are any number of repetitions of an input item that contains multiple variables. Examples of packages that make use of this subroutine are the General-Head Boundary, Drain, River, and Well Packages.

1. [***EXTERNAL*** IN] or [***OPEN/CLOSE*** FNAME]
   If Item 1 is not included, then the list is read from the package file. Item 1 must begin with the keyword "EXTERNAL" or the keyword "OPEN/CLOSE" (not both).

2. [***SFAC*** Scale]

3. List

Explanation of Variables Read by the List Utility Subroutine:

IN—is the unit number for a file from which the list will be read. The name of the file associated with this file unit must be contained in the Name File, and its file type must be "DATA" in the Name File.

FNAME—is the name of a file from which the list will be read. This file will be opened on unit 99 just before reading the list and closed immediately after the list is read. This file should not be included in the Name File.

Scale—is a scale factor that is multiplied times the value of one or more variables within every line of the list. The input instructions that define a list, which will be read by ULSTRD, should specify the variables to which SFAC applies. If Item 2 is not included, then Scale is 1.0. If Item 2 is included, it must begin with the keyword "SFAC." The values of the list variables that are printed to the listing file include the effect of Scale.

List—is a specified number of lines of data in which each line contains a specified number of variables. The first three variables are always layer, row, and column. The other fields vary according to which package is calling this subroutine.

CHAPTER 9
PROGRAMMER DOCUMENTATION

This chapter provides code documentation for programmers. This chapter starts with sections on overall design decisions and the use of Fortran modules for sharing data. The remaining sections describe the MAIN Program, each package, and the utility subroutines; the extent of the documentation varies substantially among the sections. The most detailed documentation is provided for the MAIN Program, the Basic (BAS) Package, the Layer-Property Flow (LPF) Package, the River (RIV) Package, Recharge (RCH) Package, and the Utility subroutines. The remaining stress packages are similar to RIV and RCH, so the subroutines are only briefly summarized. The Block-Centered Flow (BCF) Package is similar to LPF. Solvers are only briefly described because rarely are they modified, and programming details can be found in the references. Many details about the programs also are included in the code as Fortran comments, which must be examined to gain total understanding of the code.

Overall Design Decisions

The following paragraphs provide information about a variety of programming decisions or conventions that were adopted for MODFLOW. The coding decisions are not absolute, and coding style is not strictly enforced; however, anyone who examines the code is likely to notice some aspects of the coding decisions. Developers of new code are encouraged to follow the coding decisions.

MODFLOW is written as a batch program, which means that the program is designed to run without user intervention. The primary reason for this is the expectation that execution time will be lengthy. Execution time of course varies with the problem and the power of the computer, but this expectation continues to be valid. Simulations frequently take many minutes, and simulations taking hours or days are common.

The input data are read from files that must be prepared in advance. Although interactive computer programs usually are used to prepare the input data, interactive data input has been intentionally avoided within MODFLOW because this is in conflict with the view that MODFLOW is a batch program. The methods of interactive input are inherently less standard among kinds of computers so that including interactive input would result in a less portable program. The variety of user preferences would also promote the addition of numerous input options that would cause the code to be more difficult to understand. From the perspective of maintaining a compact, efficient, and easily understood code, input data read from files is best.

To maximize computational efficiency and minimize code complexity, MODFLOW includes minimal error checking. Also, when an error is found, the program stops rather than prompting the user to correct the error and resuming computations. This does not mean that the author views error checking as unimportant. Rather, the author believes that the error checking is best done as part of an interactive program that prepares the input files. Having separate data preparation and data checking makes possible the accommodation of the needs and preferences of a variety of users while keeping the model code straightforward and efficient.

Variable names generally are six characters or less, which was originally a requirement of Fortran. Short names also help to keep the code compact. Longer variable names are used sparingly in new parts of the code. Changing the original variable names to longer names at this time would be unwise because of possible confusion.

Many programmers have strong feelings about the issue of specifying the data type of variables. Default implicit typing generally is used in MODFLOW. This means that type statements are not used unless specifically needed; and therefore, the type will be Real for variables starting with letters A–H and O–Z, and the type will be Integer for variables starting with I–N. The default types allow the code to be shorter, but limit flexibility for naming variables. Requiring data types to be declared for all variables also would have the advantage that the compiler would produce an error if an untyped variable were used, which is helpful for avoiding misspelling of variables. Nevertheless, the desire for short, concise code led to the decision to use default typing.

Logical variables generally are avoided. Instead, Integer flags are used in which false is a value of 0 and true is a value of not 0 (usually 1). This is partly evolutionary because the original authors experienced errors with early

Fortran compilers regarding logical variables. Some code simplicity also is gained from avoiding the need for logical type statements and logical operators.

The precision used to represent numbers in the computer determines in large part the accuracy of the numeric solution of the flow equation. When using Fortran, precision depends on the computer hardware and the declared precision of Real numbers. Although the Fortran standard does not directly specify precision requirements, Fortran includes single and double precision Real numbers without specifying the actual precision of either type. Fortran 90 provides a further mechanism for finding out the precision of numbers and specifying what precision is desired, but still no absolute requirements for numeric precision are incorporated. MODFLOW was developed assuming that single precision Real numbers are represented with approximately 32 binary digits and double precision Real numbers are represented with 64, which was and continues (2005) to be common among computers. The default implicit Real type is single precision. When double precision is desired, the type is explicitly declared to be double precision.

Tests of the original MODFLOW code were done to determine the need or benefit from making some variables double precision. The tests showed that a wide variety of problems worked well when using a limited amount of double precision. Head and some of the solver arrays were made double precision based on this testing. Some problems benefit from the use of full double precision, and in general, the larger the problem, the more likely that full double precision is beneficial. Other reasons why double precision might be needed are for high-precision answers and for problems where the head distribution is relatively flat in large areas of the model.

The number of cells in a typical model has increased over the years as the complexity of problems and computer speed have increased; thus, the need for full double precision is greater now than when MODFLOW was originally released (1984). This would argue in favor of eliminating the mixed precision coding in favor of full double precision; however, the original mixed precision continues to be adequate for a variety of problems. There is a penalty of computer memory, disk space, and sometimes on execution time when full double precision is used. The author is reluctant to impose this penalty unnecessarily. Therefore, the original mixed precision code has been maintained in MODFLOW–2005. Modern Fortran compilers have an option to convert all Real numbers to double precision without the need to modify the code, so on most computers, making any code changes to convert the code to double precision is unnecesssary. Simply recompile by using the double precision compiler option. When necessary, code modification to convert MODFLOW to all double precision as described by McDonald and Harbaugh (1988, Appendix A) is still a relatively simple task.

To avoid inconsistent results among early Fortran compilers, mixed precision in a single arithmetic expression was avoided when MODFLOW was developed, and this practice continues. If any operand in an arithmetic expression is double precision, all of the single precision operands are replaced with temporary double precision variables that are set equal to the single precision values. These temporary variables have no affect when converted to full double precision. When constants are needed in arithmetic statements, ambiguity over the precision of the constants is generally avoided by using variables to store the constant values. Regardless of the precision of the constant, the Fortran compiler will cause the constant to be converted to the precision of the variable. The constant value is then referenced using the variable. For example, variables ONE and ZERO are typically used to store values for 1.0 and 0.0, respectively.

Explicit DO Loops are used throughout much of MODFLOW to make assignments of array elements an element at a time; however, in some places a Fortran 90 array assignment statement is used to assign values to all elements of an array without using a Do Loop. The lack of array assignment statements is primarily evolutionary in that this capability was not available until Fortran 90. Further, the author sees benefit in maintaining explicit loops for the sake of clarity. Although array assignment statements result in shorter code, they tend to make the code more obscure. When an array assignment statement is used, one must recognize that a variable is an array to understand what is going on in the code.

Experience has shown that some specifiers used in the Fortran OPEN statement are not standard among all compilers. File openspec.inc contains variable definitions used for opening files. Variables ACCESS, FORM, and ACTION are used as values for the ACCESS, FORM, and ACTION specifiers in open statements. Openspec.inc is incorporated in the code whenever the variables are needed by using a Fortran INCLUDE statement. The values of variables can be modified in openspec.inc and the code recompiled if different values for the open specifiers are required.

Output to the Listing File is mostly written using lines of 80 characters or less. Although computer displays are no longer limited to 80 characters as in the past, a width of 80 characters is judged to be a convenient size for a wide range of uses. For compatibility with older versions of MODFLOW, the utility subroutines continue to support the writing of lines that are 132 characters wide.

Each stress and internal flow package must incorporate a budget subroutine that adds budget data to the VBVL array. The VBVL array is defined as part of Fortran module GWFBASMODULE of the Basic Package. Additional information about the model budget is contained in Chapter 3 of this report. Budget subroutines can optionally support cell-by-cell budgets, and all packages documented in this report support this capability. This capability requires budget data to be written into a file using the appropriate cell-by-cell budget utility subroutine.

When the cell-by-cell budget capability is supported, the budget subroutine should store budget values in array BUFF. BUFF should contain the net inflow for each cell in the entire grid. At cells where IBOUND is 0 or where stress is not applied, BUFF should be set equal to 0. The Ground-Water Flow Process does not use the budget values stored in BUFF, but this is a mechanism for other processes to obtain the budget data. Further, for packages that specify stresses using lists of cell locations (as opposed to layer arrays), the cell-by-cell budget values should be stored in the list of data for each stress location. Again, this can be useful for other processes. Space for this budget data must be reserved in the list of data.

For internal flow packages, flow between adjacent cells is a special kind of cell-by-cell budget data. These data do not represent inflow or outflow for the system as a whole, but the data can be useful for many purposes. These data comprise three terms, one for each coordinate axis as described in Chapter 3. These data are written to a file as is done for inflows and outflows. Flow between adjacent cells is also stored in BUFF for use by the Ground Water Transport Process if the IBDRET flag is not 0. When IBDRET is not 0 and budget data are not written to disk, flow between adjacent cells is computed only for a subset of the grid, which is designated through subroutine arguments. IBDRET is a local variable in the MAIN Program.

Data Declaration and Sharing Using Fortran Modules

Shared data can be declared in Fortran modules. This was not done in the original code because the capability was not available until Fortran 90. Most data were passed to subroutines as arguments, which results in long argument lists, but makes clear which data are used by the subroutines. Shared data in modules are passed to subroutines using the Fortran USE statement. The “ONLY” specifier to USE allows the specific variables being used to be identified in each subroutine. Fortran modules eliminate the need for most subroutine arguments.

Figure 9–1 is an example Fortran module for a simple river package. This is simpler than the RIV Package in MODFLOW–2005, but illustrates the concept of sharing data using a Fortran module. Integer NRIVER is the number of river cells in use in the current time step, and MXRIVR is the maximum number of river cells that can be used in any stress period. RIVR is a two-dimensional array declared as a pointer. RIVR is allocated with the first index being the number of values needed for each river cell, and the second index being MXRIVR. The specific dimensions for RIVR will be declared when the array is allocated in the Allocate and Read Procedure subroutine (GWF2RIV7AR). All of these variables have the SAVE attribute so that they will never become unassociated.

```
MODULE GWFRIVMODULE
  INTEGER,SAVE                                  ::NRIVER,MXRIVR
  REAL,   SAVE, DIMENSION(:,:), POINTER     ::RIVR
END MODULE GWFRIVMODULE
```

Figure 9–1. Fortran module for declaring shared data.

The data in the module can be accessed in any subroutine by including the statement

```
USE GWFRIVMODULE,   ONLY:NRIVER,MXRIVR,RIVR
```

If each river cell requires six data values, the memory for RIVR can be allocated with the statement:

```
ALLOCATE (RIVR(6,MXRIVR))
```

A new complexity is introduced when the possibility of Local Grid Refinement (LGR) (Mehl and Hill, 2004) is added to MODFLOW. Although LGR is not described here, the impact on shared data is described. LGR requires data to be defined for multiple grids. At a minimum, a regional grid for the entire modeled area and a refined grid containing a subset of the modeled area is included. The regional and refined grids must both have data defined. In MODFLOW–2005, multiple grids are accommodated by using a derived data type. The derived type contains data for one grid, and an array of the derived type can then store data for multiple grids.

Figure 9–2 is the Fortran module for the RIV Package of MODFLOW–2005, which supports LGR. GWFRIVTYPE is a derived type that includes a complete set of pointers for all data required for a grid. GWFRIVDAT is an array of this type. Thus each element of GWFRIVDAT can contain data for one grid. The data for different grids can be accessed by specifying an element of GWFRIVDAT and the variable within GWFRIVTYPE. For example, the value of MXRIVR for the second grid would be GWFRIVDAT(2)%MXRIVR.

```
MODULE GWFRIVMODULE
TYPE GWFRIVTYPE
  INTEGER,POINTER  ::NRIVER,MXRIVR,NRIVVL,IRIVCB,IPRRIV
  INTEGER,POINTER  ::NPRIV,IRIVPB,NNPRIV
  CHARACTER(LEN=16), DIMENSION(:),   POINTER     ::RIVAUX
  REAL,              DIMENSION(:,:), POINTER     ::RIVR
END TYPE
TYPE(GWFRIVTYPE), SAVE:: GWFRIVDAT(10)
END MODULE GWFRIVMODULE
```

Figure 9–2. Fortran module for declaring shared data with support for multiple grids.

Several additional variables compared with those used for the simple river package (fig. 9–1) also have been included in the module shown in figure 9–2. These variables support some of the optional capabilities of the River Package. See the documentation for the River Package for definitions of all the variables.

Variables in a derived type can be referenced more simply through the use of pointers. Although some extra steps are needed to setup the pointers, the code then looks more like the code without the LGR capability. This approach is implemented by creating pointers that have the same names as the variables in the derived type as shown in figure 9–3. The 4 lines preceding the derived type declaration are the declarations for these variables. Each pointer can be made to point to a variable for a specific grid. The pointers can then be used to access the variables for that grid without repeating the grid designation. For example, the statement

MXRIVR=>GWFRIVDAT(2)%MXRIVR

causes MXRIVR to point to the value of MXRIVR in the derived type for grid 2, and accordingly, GWFRIVDAT(2)%MXRIVR can be accessed simply as MXRIVR.

```
MODULE GWFRIVMODULE
  INTEGER,SAVE,POINTER  ::NRIVER,MXRIVR,NRIVVL,IRIVCB,IPRRIV
  INTEGER,SAVE,POINTER  ::NPRIV,IRIVPB,NNPRIV
  CHARACTER(LEN=16),SAVE, DIMENSION(:),   POINTER     ::RIVAUX
  REAL,             SAVE, DIMENSION(:,:), POINTER     ::RIVR
TYPE GWFRIVTYPE
  INTEGER,POINTER  ::NRIVER,MXRIVR,NRIVVL,IRIVCB,IPRRIV
  INTEGER,POINTER  ::NPRIV,IRIVPB,NNPRIV
  CHARACTER(LEN=16), DIMENSION(:),   POINTER     ::RIVAUX
  REAL,              DIMENSION(:,:), POINTER     ::RIVR
END TYPE
TYPE(GWFRIVTYPE), SAVE:: GWFRIVDAT(10)
END MODULE GWFRIVMODULE
```

Figure 9–3. Fortran module for declaring shared data with support for multiple grids and pointers for simplified access.

To make possible the use of the pointers as simple variables, several steps are needed in addition to declaring the pointers in the module. First, the non-array pointers must be allocated much like the array pointers are allocated. Allocating a scalar pointer causes memory to be created for that scalar. This is done by ALLOCATE statements at the beginning of the Allocate and Read (AR) Procedure subroutine. When LGR is used, the AR subroutine will be called for each grid. At the end of the AR subroutine, the pointers to the variables for the current grid must be saved in the corresponding pointers of the derived type (GWFRIVDAT) for the grid. For example, the statement

```
GWFRIVDAT(2)%MXRIVR=>MXRIVR
```

causes the MXRIVR pointer for the current grid to be saved in the derived type for grid 2. The complete set of these pointer saving statements are placed in a pointer saving subroutine, named SGWF2RIV7PSV for the River Package, which is called at the end of the AR subroutine. All other subroutines that use the data need to insure that the pointers are set to the current grid by setting the simple variable pointers. This is done by calling a pointer setting subroutine, named SGWF2RIV7PNT for the River Package, at the beginning of each subroutine. The pointer setting subroutine contains statements setting the pointers for all the data for the package. Each package that supports LGR will have a pointer saving secondary subroutine and a pointer setting secondary subroutine. These pointer saving and setting subroutines are not specifically documented in the subsequent sections that document each package because they all have the same simple structure.

Fortran provides a mechanism for releasing allocated memory, which is called deallocation. Memory is used throughout a simulation in MODFLOW, so deallocation is not needed until the end. Further, a Fortran program automatically releases all memory when the program terminates, and accordingly, explicit memory deallocation generally is not required in MODFLOW. A deallocation procedure is included in the MODFLOW–2005 flowchart (fig. 3–1), however, and each package includes a deallocation subroutine that is called to release memory from the Ground-Water Flow Process (GWF). The deallocation subroutines are not specifically documented in the subsequent sections that document each package because they all have the same simple structure.

MAIN Program

The MAIN Program controls the order in which the primary subroutines are executed. This occurs with CALL statements that specify by name the subroutines to be executed. The arrangement of CALL statements in the MAIN Program reflects the order of procedures shown in the system flow chart (fig. 3–1). Within most procedures, the calls to the primary subroutines should begin with the Basic Package (if included in that procedure) followed by calls to the other packages in any order. The one exception is that the Basic Package Deallocate subroutine should be the last call in the Deallocate procedure. The reason for this is that the data allocated in the Basic Package may be needed by other deallocate subroutines.

The subroutines of a package are called only if the package is being used in a simulation. The Name File indicates which packages are being used. Each primary option has a unique file type; when the option is a package, the file type is simply the package abbreviation. The defined file types are stored in the one-dimensional CUNIT variable; a DATA statement in the MAIN Program defines these values. CUNIT contains 100 elements, and each element consists of four characters. Unused elements are filled with blanks. A corresponding IUNIT integer variable also has 100 elements. The Basic Package initializes all IUNIT values to 0. When the Name File is read, each file is tested to see if its File Type parameter matches one of the defined file types in CUNIT. If a match is indicated, the unit number for that file is placed in the element of IUNIT that corresponds to the matched element of CUNIT. IUNIT is an indicator of which options are active, and therefore is tested to determine whether or not the subroutines of a package should be called. For example, the Recharge Package corresponds to the eighth element of IUNIT. If IUNIT(8) is greater than 0; then the Recharge Package is active, and the primary subroutines of the Recharge Package are all called within the appropriate procedures.

The MAIN Program is the only place where Fortran modules are used without the “ONLY” option. The reason for using modules in the MAIN Program is to allow all data to pass to the solvers using subroutine arguments. The subroutine arguments indicate which variables are used.

File “openspec.inc” appears in an Include Statement in the MAIN Program because openspec.inc defines file attributes that can sometimes vary among compilers. Putting such attributes in a separate file allows the code to be changed in many places by modifying the single file. In the MAIN Program, the Open Statement for the Name File uses one of the file attributes (ACTION).

Character variable VERSION is defined using a Fortran Parameter Statement. VERSION is used to identify the specific version of MODFLOW.

The executable part of the code starts by writing VERSION to the standard output. Then, two unit numbers are defined. INUNIT is the unit number for the Name File, and IERRU is the output unit number for writing errors.

The MAIN Program makes use of a special secondary subroutine, GETNAMFIL, to get the name of the Name File. This subroutine is not part of any MODFLOW package. Isolating this code in a separate subroutine makes the MAIN Program easier to follow and makes modifying the way the Name File is defined easier should this be desired. GETNAMFIL first attempts to get the Name File from the command line. If the command line argument is absent, the user is prompted to enter a file name. Modifying the call to a system subroutine that retrieves the command line may be necessary. After calling GETNAMFIL, the MAIN Program opens the Name File and writes the name to standard output.

The next step in MAIN is to call the standard Fortran subroutine for obtaining the date and time. This will be used at the end of the program to compute the elapsed simulation time.

The variable IGRID is incorporated in some primary subroutines to facilitate the addition of the Local Grid Refinement (LGR) capability mentioned above. Variable IGRID is set equal to 1 in MAIN, which indicates the primary grid.

The loop indices for iteration, time steps, and stress periods are needed in many subroutines; however, these are not directly passed as arguments. Rather, duplicate variables are assigned the values and the duplicates are passed. This avoids the appearance that these loop indices could be modified, which allows some Fortran compilers to perform greater optimization.

Like subroutine GETNAMFIL, subroutine GLO1BAS7ET is a secondary subroutine of MAIN. GLO1BAS7ET is used to compute execution time for the simulation and write the execution time to the standard output. This is an unessential part of the program that users may wish to modify or delete. This routine makes a second call to the standard Fortran date and time routine and computes the elapsed time since the initial call.

The following is a summary of the MAIN Program (the numbers in the list correspond to the numbers of the comments in the MAIN Program listing):

1. USE package modules for purpose of data sharing.

2. Write banner to screen and assign 99 as the input unit number for the Name File.

3. Call GETNAMFIL to obtain the name of the Name File either from a command argument or from a user response to a prompt. This code was placed in a subroutine rather than directly in the MAIN to avoid obscuring the structure of the MAIN.

4. Open the Name File.

5. Get current date and time so execution time can be tracked.

6. Call primary subroutines in the AR Procedure. (GWF1BAS6AR reads the Name File and assigns IUNIT values.)

7. For each stress period:
   - 7A. Read stress-period timing information.
   - 7B. Read and prepare information that changes each stress period.
   - 7C. For each time step:
     - 7C1. Calculate the current time-step length and move "new" heads from the preceding time step to the variable containing "old" heads of the current time step.

7C2. For each solution iteration:
    7C2A. Formulate the finite-difference equations.
    7C2B. Calculate an approximate solution to the system of equations.
    7C2C. If convergence criterion has been met, stop iterating.
7C3. Determine the type and amount of output needed for this time step.
7C4. Calculate overall budget terms and, if specified, calculate and print or record cell-by-cell flow terms.
7C5 Print and/or record heads and/or drawdown. Print the overall volumetric budget and simulation time summary.
7C6. If convergence criterion was not met during iteration, STOP.

8. Call GLO1BAS7ET to get date and time at end of simulation.

9. Close files and deallocate memory.

10. End of program.

Basic Package

Although the Basic (BAS) Package does not contribute terms to or solve the flow equation, the BAS Package performs many logistical functions. BAS declares and allocates memory for variables that can be shared throughout the GWF and other processes, initializes variables that are used to construct the flow equation, reads the Name File and opens files, reads discretization data, and implements output control.

Basic Package Data

The shared Basic Package variables are declared in four Fortran modules: GLOBAL, PARAMMODULE, GWFBASMODULE, and GWFCHDMODULE; tables 9–1, 9–2, 9–3, and 9–4 define these variables, respectively. Some of the program variables have different names than the variables used to develop the finite-difference equation in Chapter 2, and these differences are described in the tables.

GLOBAL Module

NIUNIT, which defines the size of the IUNIT, VBVL, and VBNM arrays, is defined using a Fortran Parameter statement. These arrays cannot be readily defined on the basis of user requirements because these arrays are allocated prior to the user specifying information about the required array dimensions.

PARAMMODULE Module

Parameter definitions for all packages are stored in a group of variables contained in Fortran module PARAMMODULE. The parameter variables are somewhat complex, and it is recommended that they be accessed only through the parameter utility subroutines. These routines create parameters and access all of the associated data.

The primary parameter data are stored in one-dimensional arrays. PARNAM contains the parameter name, PARTYP contains the parameter type, B contains the parameter value, and IACTIVE is a flag indicating whether a parameter is active in the current time step. The parameters are stored in the order in which they are defined. Parameters are found when needed by a simple sequential search through PARNAM.

Parameter indexing data depend on whether a parameter is an array parameter or a list parameter. An array parameter requires a set of clusters that define the cells in each layer that are associated with the parameter. Clusters are stored in IPCLST. Pointers to the first and last cluster for a parameter are stored in IPLOC. For example, if parameter 7 is a hydraulic conductivity parameter, then IPLOC(1,7) and IPLOC(2,7) contain the location of the first and last clusters in IPCLST associated with this parameter.

A list parameter requires pointers to the first and last cells associated with the parameter in the list that stores data for the package. These pointers are stored in IPLOC. For example, if parameter 5 is a river parameter, then IPLOC(1,5) and IPLOC(2,5) contain the location in RIVR of the first and last cells associated with this parameter.

The capability to have multiple instances of the same parameter adds further complexity to parameter definition. This capability is used to allow parameter definitions to change during the simulation. Each instance has the same parameter value, but the cells associated with the parameter can be different. Thus, each array parameter instance has its own clusters, and each list parameter instance has its own cell list. IPLOC(3,n) indicates the number of instances. If there are instances for an array parameter, then IPLOC(2,n) is the last cluster of the last instance. Each instance must have the same number of clusters. Similarly, if there are instances for a list parameter, IPLOC(2,n) is the location of the last cell of the last instance. Each instance must have the same number of cells.

Several variables that specify array dimensions (MXPAR, MXCLST, and MXINST) are defined using Fortran Parameter statements. These cannot be readily computed from input data because the arrays are allocated prior to the user specifying information about the array dimensions.

Table 9–1. Variables in Fortran module GLOBAL.

Variable Name	Size	Description
NIUNIT	**Scalar**	**The dimension of the IUNIT, VBVL, and VBNM arrays. Initially set equal to 100 using a Parameter statement.**
NCOL	**Scalar**	**The number of columns in the model grid.**
NROW	**Scalar**	**The number of rows in the model grid.**
NLAY	**Scalar**	**The number of layers in the model grid.**
NPER	**Scalar**	**The number of stress periods in the simulation.**
NBOTM	**Scalar**	**The number of layers of data in the BOTM array.**
NCNFBD	**Scalar**	**The number of model layers that are underlain by Quasi–3D confining beds.**
ITMUNI	**Scalar**	**Time unit code: 0=undefined, 1=seconds, 2=minutes, 3=hours, 4=days, 5=years**
LENUNI	**Scalar**	**Length unit code: 0=undefined, 1=feet, 2=meters, 3=centimeters**
IXSEC	**Scalar**	**Cross section flag that is set to 1 if "XSECTION" is found in the option line of Basic Package file: 0 indicates not a cross section, 1 indicates a 1-row cross section.**
ITRSS	**Scalar**	**Flag: 0=steady state, 1=transient, -1=combined steady state and transient.**
INBAS	**Scalar**	**File unit number for the BAS Package of the GWF Process.**
IFREFM	**Scalar**	**Free format flag that is set to 1 if "FREE" is found in the option line of Basic Package file: 0=fixed format, 1=free format.**
NODES	**Scalar**	**The number of nodes (or cells) in the model grid.**
IOUT	**Scalar**	**The file unit number for the Listing File.**
IUNIT	**NIUNIT**	**An array of unit numbers. Each package or option that requires an input file is assigned an element in IUNIT. If that element is 0, the package or option is inactive. If the element is greater than 0, the package or option is active and the value is unit number for reading data.**
LBOTM	**NLAY**	**Array of pointers to the 3rd index of the BOTM array. For example, LBOTM(2) is the index in BOTM of the bottom of layer 2.**
LAYCBD	**NLAY**	**Quasi–3D confining bed flag for each layer: 0 indicates no confining bed under a layer, not 0 indicates a confining bed.**
LAYHDT	**NLAY**	**Head-dependent transmissivity flag: 0 indicates transmissivity for a layer is constant, 1 indicates transmissivity varies with head.**
LAYHDS	**NLAY**	**Head-dependent storage flag: 0 indicates storage for a layer is constant, 1 indicates storage varies with head.**
PERLEN	**NPER**	**The length of a stress period.**
NSTP	**NPER**	**The number of time steps in a stress period.**
TSMULT	**NPER**	**Multiplier for the length of successive time steps in a stress period.**
ISSFLG	**NPER**	**Steady State/Transient flag for a stress period – "SS" for steady state and "TR" for transient.**
DELR	**NCOL**	**The cell width along rows (Δr in equations used throughout this report). DELR(J) is the width of column J.**
DELC	**NROW**	**The cell width along columns (designated as Δc in equations throughout this report). DELC(I) is the width of row I.**
BOTM	**NCOL,NROW,0:NBOTM**	**Elevation of cell and Quasi–3D confining bed bottoms. A value of 0 for the 3rd index is used to store the system top elevation.**
HNEW	**NCOL,NROW,NLAY**	**Computed head at the current time step (designated as h^m throughout this report).**
HOLD	**NCOL,NROW,NLAY**	**Head from the end of the previous time step (designated as h^{m-1} throughout this report).**
IBOUND	**NCOL,NROW,NLAY**	**Boundary code: <0 – specified head, =0 – no flow, >0 – variable head.**
CR	**NCOL,NROW,NLAY**	**Conductance along rows. CR(J,I,K) is the conductance between cell (J,I,K) and (J+1,I,K), where J is the column index.**
CC	**NCOL,NROW,NLAY**	**Conductance along columns. CC(J,I,K) is the conductance between cell (J,I,K) and (J,I+1,K), where I is the row index.**
CV	**NCOL,NROW,NLAY-1**	**Vertical conductance. CV(J,I,K) is the conductance between cell (J,I,K) and (J,I,K+1), where K is the column index.**
HCOF	**NCOL,NROW,NLAY**	**Coefficient of head in ground-water flow equation.**
RHS	**NCOL,NROW,NLAY**	**Right-hand side of ground-water flow equation.**
BUFF	**NCOL,NROW,NLAY**	**Temporary buffer for use within a subroutine.**
STRT	**NCOL,NROW,NLAY**	**Initial head.**

Table 9–2. Variables in Fortran module PARAMMODULE.

["C*n" in size column indicates a character variable of n characters]

Variable Name	Size	Description
MXPAR	Scalar	Maximum number of parameters allowed. Initially set equal to 500 using a Parameter statement.
MXCLST	Scalar	Maximum number of clusters that can be used for defining array parameters. Initially set equal to 1000 using a Parameter statement.
MXINST	Scalar	Maximum number of zone arrays. Initially set equal to 1000 using a Parameter statement.
ICLSUM	Scalar	The number of defined array parameter clusters.
IPSUM	Scalar	The number of defined parameters.
INAMLOC	Scalar	Pointer to the next unused instance name.
NMLTAR	Scalar	The number of multiplier arrays
NZONAR	Scalar	The number of zone arrays.
NPVAL	Scalar	The number of parameters for which values are defined in the Parameter Value File.
B	MXPAR	Parameter values.
IACTIVE	MXPAR	Active parameter flags: -1 – Active in all time steps. 0 – Not active in current time step. >0 – Active in current time step – if using instances, IACTIVE is the instance number.
IPLOC	4,MXPAR	Parameter index. Values of IPLOC(n,p), where p is the parameter number: n=1 – First cluster or list location of first instance. n=2 – Last cluster or list location of last instance. n=3 – Number of instances. n=4 – Location in INAME of first instance name.
IPCLST	14,MXCLST	IPCLST(n,c) – c=cluster number: n=1 – Layer number. n=2 – Multiplier array number (0 indicates none). n=3 – Zone array number (0 indicates all cells). n=4 – Index of last zone number for this cluster. n=5-14 – Zone numbers.
PARNAM	C*10,MXPAR	Parameter names.
PARTYP	C*4,MXPAR	Parameter types.
ZONNAM	C*10,MXZON	Zone array names.
MLTNAM	C*10,MXMLT	Multiplier array names.
INAME	C*10,MXINST	Instance names.
RMLT	NCOL,NROW,NMLTAR	Multiplier arrays for defining parameters.
IZON	NCOL,NROW,NZONAR	Zone arrays for defining parameters.

GWFBASMODULE Module

Table 9–3. Variables in Fortran module GWFBASMODULE.

["C*n" in size column indicates a character variable of n characters]

Variable Name	Size	Description
MSUM	**Scalar**	**Budget term counter.**
IHEDFM	**Scalar**	**Format code for writing head in the Listing File.**
IHEDUN	**Scalar**	**File unit number for saving head in a file.**
IDDNFM	**Scalar**	**Format code for writing drawdown in the Listing File.**
IDDNUN	**Scalar**	**File unit number for saving drawdown in a file.**
IBOUUN	**Scalar**	**File unit number for saving IBOUND in a file.**
LBHDSV	**Scalar**	**Label flag for saving head in a formatted file:** **0 –Do not write label.** **not 0 – Write label.**
LBDDSV	**Scalar**	**Label flag for saving drawdown in a formatted file:** **0 – Do not write label.** **not 0 – Write label.**
LBBOSV	**Scalar**	**Label flag for saving IBOUND in a formatted file:** **0 – Do not write label.** **not 0 – Write label.**
IBUDFL	**Scalar**	**Flag for writing overall budget in the Listing File in current time step:** **0 – Do not write budget.** **not 0 – Write budget.**
ICBCFL	**Scalar**	**Flag for writing cell-by-cell budget data in current time step:** **0 – Do not write data.** **1 – Write data using noncompact form.** **2 – Write data using compact form.**
IHDDFL	**Scalar**	**Flag for output of head and drawdown in current time step:** **0 – No output.** **1 – Write output according to IOFLG.**
IAUXSV	**Scalar**	**Auxiliary flag for saving cell-by-cell budget data:** **0 – Do not save auxiliary data in budget file.** **not 0 – Save auxiliary data in budget file.**
IBDOPT	**Scalar**	**Compact budget option:** **1 – Noncompact** **2 – Compact**
IPRTIM	**Scalar**	**A flag that is set to 1 if "PRINTTIME" is specified in the option line of the Basic Package file. This is used to determine if execution time is written to the Listing File.**
IPEROC	**Scalar**	**The stress period at which the next output control should occur when using alphabetic output control. If numeric output control is used, the value is -1.**
ITSOC	**Scalar**	**The time step at which the next output control should occur when using alphabetic output control. If numeric output control is used, the value is -1.**
ICHFLG	**Scalar**	**A flag that is set to 1 if "CHTOCH" is specified in the option line of the Basic Package file. This indicates that flow between constant-head cells should be computed when cell-by-cell budget terms are computed.**
DELT	**Scalar**	**Length of current time step. (Designated as $\Delta t = t^m - t^{m-1}$ throughout this report.)**
PERTIM	**Scalar**	**Total simulation time in current stress period.**
TOTIM	**Scalar**	**Total simulation time.**
HNOFLO	**Scalar**	**Value substituted in HNEW at no-flow cells.**
CHEDFM	**C*20**	**Format for saving head in a file (blank indicates unformatted).**
CDDNFM	**C*20**	**Format for saving drawdown in a file (blank indicates unformatted).**
CBOUFM	**C*20**	**Format for saving IBOUND in a file.**
IOFLG	**NLAY,5**	**Flags for output of data in current time step (k is a layer index):** **(k,1) Not 0 – Write head to Listing File.** **(k,2) Not 0 – Write drawdown to Listing File.** **(k,3) Not 0 – Save head to unit IHEDUN.** **(k,4) Not 0 – Save drawdown to unit IDDNUN.** **(k,5) Not 0 – Save IBOUND to unit IBOUUN.**

VBVL	4,NIUNIT	Overall budget values (n is a budget term index): (1,n) – Inflow rate for current time step. (2,n) – Outflow rate for current time step. (3,n) – Cumulative volume of inflow. (4,n) – Cumulative volume of outflow.
VBNM	C*16,NIUNIT	Names of budget terms.

GWFCHDMODULE Module

Table 9–4. Variables in Fortran module GWFCHDMODULE.

["C*n" in size column indicates a character variable of n characters]

Variable Name	Size	Description
NCHDS	Scalar	The number of constant-head cells in the current stress period.
MXCHD	Scalar	The second dimension of CHDS, which includes space for the active constant-head cells in a stress period and all constant-head cells defined by parameters.
NCHDVL	Scalar	The number of the first dimension of CHDS array, which includes five input values and the auxiliary data.
IPRCHD	Scalar	Flag for printing constant-head data – 0 indicates do not print, 1 indicates print.
NPCHD	Scalar	The number of constant-head parameters.
ICHDPB	Scalar	The value of the second index in CHDS at which parameter data begins.
NNPCHD	Scalar	The number of nonparameter constant-head cells in the current stress period.
CHDAUX	C*16,20	The name of auxiliary variables.
CHDS	NCHDVL,MXCHD	Constant-head list, which includes space for the constant-head cells active in a stress period and all constant-head cells defined by parameters. CHDS does not include constant-head cells specified through the initial value of IBOUND.

Subroutines

The Basic Package contains six primary subroutines: GWF2BAS7AR, GWF2BAS7ST, GWF2BAS7AD, GWF2BAS7FM, GWF2BAS7OC, and GWF2BAS7OT. The subroutines GWF2BAS7AR, GWF2BAS7OC, and GWF2BAS7OT each make use of secondary subroutines to reduce the size of the primary subroutines. The Time-Variant Specified-Head (CHD) Option further consists of subroutines GWF2CHD7AR, GWF2CHD7RP, and GWF2CHD7AD. These CHD subroutines are named as if they are a package because this code was originally viewed as a package (Leake and Prudic, 1991). The CHD subroutines also are called from the MAIN Program as if they are primary subroutines. Nevertheless, as mentioned in Chapter 4, CHD is viewed functionally as an option within the BAS Package because of its function to define IBOUND values.

GWF2BAS7AR

This subroutine allocates memory and reads data for the Basic (BAS) Package. The BAS Package file is read directly by this subroutine. The Name, Discretization, Output Control, Zone, and Multiplier files are read by secondary subroutines called by GWF2BAS7AR.

Arguments:

INUNIT – Input unit number for the Name File
CUNIT – Array of file types for primary options
VERSION – MODFLOW version as a text string
IUDIS – Element in IUNIT that is used for the Discretization File
IUZON – Element in IUNIT that is used for the Zone File
IUMLT – Element in IUNIT that is used for the Multiplier File
MAXUNIT – The maximum unit number used in the Name File
IGRID – The grid number, which is used for local grid refinement
IUOC – Element in IUNIT that is used for the Output Control File
HEADNG – A two-line simulation title
IUPVAL – Element in IUNIT that is used for the Parameter Value File

The AR subroutine performs its work in the following sequence:

1. Allocate the scalar variables.
2. Call secondary subroutine to open files.
3. Identify package and initialize data.
4. Initialize parameter definition variables.
5. Call a secondary subroutine to allocate memory for discretization and read the discretization file.
6. Allocate remaining global data.
7. Initialize LAYHDT and LAYHDS to -1. These are flags for each layer indicating if transmissivity and storage are head dependent. These flags should be defined by the internal flow package so they can be used by other packages. LAYHDT=0 indicates transmissivity for a layer is constant; LAYHDT=1 indicates transmissivity varies with head. LAYHDS=0 indicates storage for a layer is constant; LAYHDS=1 indicates storage varies with head.
8. Read the Basic Package file.
   - 8A. Read the comments, saving the first two in variable HEADNG.
   - 8B. Look for options in the first item after the heading.
   - 8C. Print the options.
   - 8D. Initialize TOTIM to 0.0.
   - 8E. Read boundary array. If the cross-section option is in effect, read the data as a single array. If not a cross section, read the data a layer at a time.
   - 8F. Read and print HNOFLO, which is the value to be printed at cells where IBOUND is initially 0.
   - 8G. Read initial heads into STRT as multiple layer arrays or as a single array for a cross section.
9. Copy initial heads from STRT to HNEW. If IBOUND is zero, set HNEW equal to the double precision equivalent of HNOFLO.
10. Call a secondary subroutine to allocate memory for the Output Control Option and read preliminary data.
11. Initialize volumetric budget accumulators.
12. Call a secondary subroutine to allocate memory for zone and multiplier arrays and read these arrays.
13. Call a secondary subroutine to read parameter values from the Parameter Value File.
14. Save memory pointers for the grid and return.

SGWF2BAS7OPEN

This subroutine reads the Name File and opens all of the files in the Name File. File types in the Name File are compared to values in array CUNIT, which is defined in the MAIN Program. The file unit number for a file is saved in the element of IUNIT that is the same as the element of CUNIT that matches the file type. For example if the file type of a file is “RIV” and element 4 of CUNIT contains “RIV,” then the file unit is saved in IUNIT(4). Thus, IUNIT(4) can be tested for a not 0 value to determine if the River Package is active.

Arguments:

INUNIT – Input unit number for the Name File
IOUT – File unit number for Listing File
IUNIT – Array of file unit numbers for primary options
CUNIT – Array of file types for primary options
NIUNIT – The number of elements in IUNIT
VERSION – MODFLOW version as a text string
INBAS – The input unit number for the Basic Package
MAXUNIT – The maximum unit number used in the Name File

SGWF2BAS7ARDIS

This subroutine allocates and reads discretization data, both space and time.

Arguments:

INUNIT – Input unit number for the Name File
CUNIT – Array of file types for primary options
VERSION – MODFLOW version as a text string
IUDIS – Element in IUNIT that is used for the Discretization File
IUZON – Element in IUNIT that is used for the Zone File
IUMLT – Element in IUNIT that is used for the Multiplier File
MAXUNIT – The maximum unit number used in the Name File
IGRID – The grid number, which is used for local grid refinement
IUOC – Element in IUNIT that is used for the Output Control File
HEADNG – A two-line simulation title

This subroutine performs its work in the following sequence:

1. Check that the input file for discretization has been specified in the Name File. Stop the simulation if the file is not defined.
2. Read the comments and the first line after the comments.
3. Use URWORD to get NLAY, NROW, NCOL, NPER, ITMUNI, and LENUNI from the line.
4. Print NLAY, NROW, and NCOL.
5. Print a message showing the time units.
6. Print a message showing the length units.
7. Allocate the flags that keep track of bottom elevation arrays (LBOTM) and confining bed arrays (LAYCBD).
8. Read confining bed flags into LAYCBD.
9. Loop through the number of layers counting confining beds as indicated by LAYCBD being not 0. Save a pointer to the confining bed number in LAYCBD. LAYCBD(K) will be the layer index for VKCB in the Layer-Property Flow Package. Also, set up LBOTM to contain the third index to the bottom elevation array BOTM for each model layer. Elevation arrays for model layers and confining beds are stored in BOTM in the order of depth. Compute NBOTM, which is the sum of the number of model layers and the number of confining beds.
10. Allocate space discretization arrays (DELR, DELC, and BOTM) and time discretization arrays (PERLEN, NSTP, TSMULT, and ISSFLG). BOTM requires NBOTM+1 layers of data, which are indexed from 0 to NBOTM. Layer 0 is for the top elevation for layer 1, which is the overall top of the simulated system.

11. Read DELR and DELC.

12. Read the top elevation for layer 1.

13. Loop through the model layers reading the bottom elevation for each layer as well as the confining bed bottom elevation for layers that have a confining bed.

14. Read and write data defining the NPER stress periods. For each stress period, a line from the input file contains the length of the stress period, the number of time steps, the time step multiplier, and a code indicating steady state or transient. Use URWORD to get each value. Check that the transient/steady-state code is valid and set a numeric code in ISSFLG: 0 for transient and 1 for steady state. Set ISS to 1 if at least one stress period is steady state, and set ITR to 1 if at least one stress period is transient.

15. Check for invalid stress period data. All stress periods must have at least one time step. Only steady-state stress periods can have 0 length. All time step multipliers must be positive. No stress periods can have negative length.

16. Use the ISS and ITR flags from step 14 to define the value for ITRSS, which is a flag that indicates the overall steady-state/transient status of the simulation: 0 indicates all stress periods are steady state, 1 indicates all stress periods are transient, and -1 indicates a combination of steady-state and transient stress periods.

17. Return.

SGWF2BAS7I

This subroutine sets up the Output Control Option. Arrays IOFLG, VBVL, and VBNM are allocated. If the Output Control Option is not active, default output control is setup. Values are then set in IOFLG to cause head for all layers to be written. GWF2BAS7OC further implements default output by invoking the output of head and overall budget at the end of each stress period.

Arguments:

NLAY – The number of layers in the model grid
INOC – Input unit number for the Name File
IOUT – File unit number for the Listing File
IFREFM – Free format flag:
    0 – Fixed format
    1 – Free format
NIUNIT – The number of elements in IUNIT

If the Output Control Option is active, the first record is read from the output control file. The first record is examined to determine whether alphabetic or numeric coding is being used. If alphabetic coding is used, then SGWF2BAS7J is called to set up output control. If numeric coding is used, the values from the first record are read, and IPEROC and ITSOC are set equal to -1 as an indicator of numeric output control.

SGWF2BAS7J

This subroutine sets up output control using alphabetic coding. Lines are read from the output control file until a line starting with “PERIOD” is found. The stress period and time step indicated by this line are saved in variables IPEROC and ITSOC, respectively. This subroutine primarily consists of a sequence of calls to URWORD and “IF” tests to search for the alphabetic commands.

Arguments:

INOC – Input unit number for the Name File
IOUT – File unit number for the Listing File
LINE – Line of text from the Output Control file

LLOC – Location pointer within LINE
ISTART – Location of the start of a word in LINE
ISTOP – Location of the end of a word in LINE

SGWF2BAS7ARMZ

This subroutine allocates and reads zone and multiplier arrays.

Arguments:

INZONE – Input unit number for the Zone File
INMULT – Input unit number for the Multiplier File

SGWF2BAS7ARPVAL

This subroutine reads and stores parameter values from the Parameter Value File. This subroutine creates a parameter for each of the listed parameters, but only the name and value are defined. The other values necessary to define a parameter must be defined in the internal flow or stress package that uses the parameter.

Arguments:

IUPVAL – Element in IUNIT corresponding to the Parameter Value File.

GWF2BAS7ST

This subroutine initializes a new stress period. Using the stress period data from the discretization file, the length of the first time step is computed. PERTIM is set to 0. If the first stress period, GWF2BAS7ST calls GWF2BAS7STPVAL to check that all parameters for which a parameter value was defined in the Parameter Value File have been fully defined.

Arguments:

KPER – Current stress period
IGRID – The grid number, which is used for local grid refinement. Note: Each grid has its own time data; however, this data is required to be the same for all grids.

SGWF2BAS7STPVAL

This subroutine checks that all parameters for which a parameter value was defined in the Parameter Value File have been fully defined. NPVAL contains the number of parameters specified in the Parameter Value File. A blank value for the parameter type indicates an undefined parameter.

Arguments: None

GWF2BAS7AD

This subroutine advances to a new time step of a stress period. The length of the new time step (DELT) is computed unless the new step is step 1. DELT is added to TOTIM and PERTIM. HOLD is set equal to HNEW.

Arguments:

KPER – Current stress period
KSTP – Current time step
IGRID – The grid number, which is used for local grid refinement

GWF2BAS7FM

This subroutine initializes HCOF and RHS to 0 so that the various packages can add terms to them. This is one place where an array assignment statement is used to initialize all elements of an array without an explicit Do Loop.

Arguments:

IGRID – The grid number, which is used for local grid refinement

GWF2BAS7OC

This subroutine is called every time step to set output control flags. If default output control is being used, flags are set to write head (IOFLG) and overall budget (IBUDFL) to the Listing File in the time step of each stress period.

If alphabetic coding of output control is being used, SGWF2BAS7N is called to read output control information. If numeric coding is being used, the output control records are read for the time step. If the time step is the last time step of a stress period, the overall budget flag (IBUDFL) is set even if IBUDFL was not specified in the output control data. This automatic setting of IBUDFL is done to prevent a user from running a model without having any budget output.

Arguments:

KSTP – Current time step
KPER – Current stress period
ICNVG – Convergence flag:
    0 – Not converged
    not 0 – Converged
INOC – Element in IUNIT that is used for the Output Control option
IGRID – The grid number, which is used for local grid refinement

SGWF2BAS7N

This subroutine reads alphabetic output control time step information and sets output flags. The output control information for a time step consists of a record specifying a stress period and time step followed by one or more records indicating the kinds of output. Not all time steps must be specified. When Subroutine SGWF2BAS7I detects alphabetic output control, SGWF2BAS7J is called to read preliminary output control data. Data are read until the first stress period/time step record is found. The specified stress period and time step values are stored in IPEROC and ITSOC.

SGWF2BAS7N is called every time step by GWF2BAS7OC. The output control times must be entered in order of increasing time. The current stress period and time step are compared to IPEROC and ITSOC, and three possibilities exist:

1. The current time can be earlier than IPEROC & ITSOC. If so, output does not occur this time step. The I/O flags are cleared, and no output control records are read.

2. The current time can exactly match IPEROC & ITSOC. This means output flags should be read and acted upon this time step. Lines are read until a new stress period/time step record is found (or end of file). As output records are found, the appropriate flags are set. When a new stress period/time step flag is found, they are saved in IPEROC and ITSOC so that subsequent calls to SGWF2BAS7N can compare them to the current simulation time. If the end of the output control file is found, IPEROC and ITSOC are set to 9999, which is presumably greater than the stress period and time step values for the remainder of the simulation.

3. The current time can be later than the IPEROC & ITSOC time. This can only happen if the stress period and time step values are entered out of order in the output control file or if a nonexistent time step is specified. For example, output control for stress period 1 and time step 2 might be specified after stress period 1 and time step 3. This causes IPEROC and ITSOC to be set equal to KPER and KSTP so that output occurs in the current time step.

Arguments:

KPER – The current stress period
KSTP – The current time step
INOC – Input unit number for the Output Control file
IOUT – File unit number for the Listing File

SGWF2BAS7L

When SGWF2BAS7N determines that head, drawdown, or IBOUND should be written (to the Listing File or a separate file), SGWF2BAS7L is called to determine which layers should be written. By default all layers are written, but a list of layer numbers can be specified at the end of the output control line. URWORD is repeatedly called to retrieve an Integer value. A value outside of the range of model layers terminates the check. Note that specification of no layer numbers at the end of the line results in the first call to URWORD returning 0 for the layer number, which causes all layers to be written.

Arguments:

IPOS – Value of second index of IOFLG for which layers are being specified
LINE – Line of data from the Output Control File
LLOC – Location in LINE
IOFLG – Flags for output of data in current time step
NLAY – Number of layers in the grid
IOUT – File unit number for the Listing File
LABEL – A label specifying the kind of data (head, drawdown, or IBOUND)
INOC – Input unit number for Output Control file

GWF2BAS7OT

This subroutine outputs head, drawdown, IBOUND, and overall budget for the time steps specified by GWF2BAS7OC. These are each output using a separate secondary subroutine. If head, drawdown, or the overall budget are written to the Listing File, IPFLG is set to 1, which causes SGWF2BAS7T to be called to write simulation time to the Listing File.

Arguments:

KSTP – Current time step
KPER – Current stress period
ICNVG – Convergence flag:
    0 – Not converged
    not 0 – Converged
ISA – Equation solution flag:
    0 – Flow equation was not solved this time step.
    not 0 – Flow equation was solved this time step.
IGRID – The grid number, which is used for local grid refinement

SGWF2BAS7H

This subroutine writes head to the Listing File and separate file according to flags in array IOFLG that have been previously set by output control.

Arguments:

KSTP – Current time step
KPER – Current stress period

IPFLG – Flag indicating whether output has been written to the Listing File in current time step:
0 – There has been no output to Listing File.
not 0 – There has been some form of output to the Listing File.
ISA – Equation solution flag:
0 – Flow equation was not solved this time step.
Not 0 – Flow equation was solved this time step.

SGWF2BAS7D

This subroutine writes drawdown to the Listing File and separate file according to flags in array IOFLG that have been previously set by output control. Drawdown is computed as the difference between initial head (STRT) and the current head (HNEW). Where IBOUND is 0, drawdown is specified to be the value of HNEW, which will either be HNOFLO or HDRY depending on the reason why the cell is dry. If the cell is dry because IBOUND was initially 0, HNEW will be HNOFLO. If the cell is dry because the head dropped below the bottom elevation, HNEW will be HDRY.

Arguments:

KSTP – Current time step
KPER – Current stress period
IPFLG – Flag indicating whether output has been written to the Listing File in current time step:
0 – There has been no output to Listing File.
not 0 – There has been some form of output to the Listing File.
ISA – Equation solution flag:
0 – Flow equation was not solved this time step.
Not 0 – Flow equation was solved this time step.

SGWF2BAS7IB

This subroutine writes IBOUND to a separate file according to flags in array IOFLG that have been previously set by output control.

Arguments:

KSTP – Current time step
KPER – Current stress period

SGWF2BAS7V

This subroutine writes the overall budget. All of the individual budget terms have been stored in variable VBVL by the packages that compute them, but the totals must still be computed. Budget values in the range from 0.1 to 9.99999E11 (BIGVL1) are printed using a fixed F17.4 format. This makes comparing the magnitude of values easy. Outside of this range, values are printed using an exponential format so that at least five digits are printed. The upper value of the range for using a fixed format for printing the difference between total inflow and outflow is decreased to 9.99999E10 (BIGVL2) because the difference can be negative, which requires an extra space for printing.

Arguments:

MSUM – Budget term counter
VBNM – Names of budget terms
VBVL – Overall budget values
KSTP – Current time step
KPER – Current stress period
IOUT – File unit number for Listing File

SGWF2BAS7T

This subroutine writes three values of simulation time: total time, stress period time, and length of the current time step. This subroutine is called at every time step for which head, drawdown, or overall budget are written to the Listing File. If the time unit is specified (ITMUNI>0), then the three values of time are converted to seconds. Each time is then converted to minutes, hours, days, and years; and all of the times are written in a table to the Listing File. A year is taken to be 365.25 days. If the time unit is not specified, then no conversion is made, and the time is written with the unknown units.

Arguments:

KSTP – Current time step
KPER – Current stress period
DELT – The length of the current time step
PERTIM – Total simulation time in current stress period
TOTIM – Total simulation time
ITMUNI – Time unit code
IOUT – File unit number for Listing File

Time-Variant Specified-Head Option Subroutines

Once a cell is made constant head, the cell stays constant head throughout the remainder of the simulation. The "active" constant-head cells in a stress period are those for which heads are being specified.

GWF2CHD7AR

This subroutine allocates memory for the CHD Option.

Arguments:

IN – Input unit number for the CHD Option
IGRID – The grid number, which is used for local grid refinement

GWF2CHD7RP

This subroutine reads data for the CHD Option every stress period.

Arguments:

IN – Input unit number for the CHD Option
IGRID – The grid number, which is used for local grid refinement

GWF2CHD7AD

This subroutine sets the head for the constant-head cells specified by the CHD Option every time step. Input data read by GWF2CHD7RP specifies the head at the beginning and end of each stress period. GWF2CHD7AD uses linear interpolation to define the head for each time step.

Arguments:

KPER – Current stress period
IGRID – The grid number, which is used for local grid refinement

Block-Centered Flow Package

The Block-Centered Flow (BCF) Package computes conductance terms for flow between cells and storage terms. When simulating confined flow, the computations are straightforward, but complexities such as unconfined conditions, cell drying, cell wetting, and conversion between confined and unconfined conditions cause the code to be much more complex. Shared data for the BCF Package are declared in Fortran Module GWFBCFMODULE and defined in table 9–5.

Table 9–5. Variables in Fortran module GWFBCFMODULE.

Variable Name	Size	Description
IBCFCB	**Scalar**	**Cell-by-cell budget flag and unit:** **<0 – Constant-head cell-by-cell budget data are written to the Listing File.** **0 – No cell-by-cell budget** **>0 – Unit number for saving cell-by-cell budget data**
IWDFLG	**Scalar**	**Wetting flag:** **0 – Wetting is inactive.** **not 0 – Wetting is active in layers where LAYCON is 1 or 3.**
IWETIT	**Scalar**	**The iteration interval for attempting to wet cells. Wetting is attempted every IWETIT iterations.**
IHDWET	**Scalar**	**Flag indicating which equation to use for defining the head at a cell that has just converted from dry to wet:** **0 – HNEW= BOT+WETFCT(H_n-BOT)** **not 0 – HNEW = BOT+WETFCT(THRESH)**
WETFCT	**Scalar**	**Factor included in the calculation of head at a cell that has just converted from dry to wet.**
HDRY	**Scalar**	**When a cell converts to dry, HNEW is set equal to HDRY.**
LAYCON	**NLAY**	**Layer-type code:** **0 – Confined** **1 – Unconfined** **2 – Partially convertible** **3 – Fully convertible**
LAYAVG	**NLAY**	**Interblock transmissivity flag.** **0 – Harmonic mean** **10 – Arithmetic mean** **20 – Logarithmic mean** **30 – Arithmetic-mean saturated thickness and logarithmic-mean hydraulic conductivity.**
TRPY	**NLAY**	**Ratio of transmissivity (or hydraulic conductivity) in the column direction to transmissivity (or hydraulic conductivity) in the row direction.**
HY	**NCOL,NROW,nhy**	**Hydraulic conductivity. The third dimension, nhy, is the number of layers where LAYCON is 1 or 3.**
SC1	**NCOL,NROW,NLAY**	**Primary storage capacity. Only allocated when simulation is transient.**
SC2	**NCOL,NROW,ntop**	**Secondary storage capacity. Only allocated when simulation is transient. The third dimension, ntop, is the number of layers where LAYCON is 2 or 3.**
WETDRY	**NCOL,NROW,nwet**	**Wetting threshold combined with wetting direction indicator. Absolute value is the wetting threshold. Negative indicates wetting only from below. Zero indicates no wetting. Positive indicates wetting from sides and below. The third dimension, nwet, is the number of layers where wetting can occur.**
CVWD	**NCOL,NROW,NLAY-1**	**Vertical conductance between cells. This is allocated only if wetting is active.**

BCF consists of six primary subroutines: GWF2BCF7AR, GWF2BCF7AD, GWF2BCF7FM, GWF2BCF7BDADJ, GWF2BCF7BDS, and GWF2BCF7BDCH. The budget procedure consists of three subroutines because BCF computes three budget terms: flow between adjacent cells, storage, and constant-head flow. BCF is similar to the Layer-Property Flow (LPF) Package; therefore, detailed documentation for BCF is not included. Refer to the LPF section for more details.

GWF2BCF7AR

This subroutine allocates and reads BCF data. Parameters are not supported, so GWF2BCF7AR is not as complex as the comparable LPF subroutine. However, the input data for each layer depends on the layer-type code, LAYCON. Therefore, this subroutine must test LAYCON to determine the arrays that are allocated and read for each layer. There are four main parts of this subroutine.

In the first part of the code (comments 1–5) initial records are read from the BCF input file to determine the options in effect and the resulting memory requirements. Quite a few lines of code are required to decode the layer-type and inter-block transmissivity codes and to write information to the Listing File. LAYHDT is set to 0 for when LAYCON is 0 or 2, and LAYHDT is set to 1 when LAYCON is 1 or 3. LAYHDS is set to 0 when LAYCON is 0 or 1, and LAYHDS is set to 1 when LAYCON is 2 or 3.

In the second part of the code (comment 6) the required memory is allocated. When an array is unneeded, a single element is allocated so that all pointers are associated.

In the third part of the code (comments 7–8) the arrays are read. The layer arrays are read through one do loop over the number of model layers (NLAY). All layer arrays are read using utility subroutine U2DREL.

Finally, the data are checked and preliminary computations are made by calling secondary subroutine SGWF2BCF7 (comment 9).

Arguments:

IN – Input unit number for BCF Package
IGRID – The grid number, which is used for local grid refinement

GWF2BCF7AD

This subroutine prepares BCF data for a new time step. This is done only when wetting is active. HOLD for dry cells that are eligible for conversion to wet is set equal to the bottom elevation. This is necessary to cause the change in head over the time step to be properly computed for the storage term in the flow equation. Otherwise, HOLD would have a value of HNOFLO or HDRY depending on whether the cell was initially dry or converted to dry in a prior time step.

Arguments:

KPER – Current stress period
IGRID – The grid number, which is used for local grid refinement

GWF2BCF7FM

This subroutine computes conductance and storage terms in the flow equation. The computations depend to a great extent on the layer type. For confined layers, cell drying and wetting are not allowed, and conductance has already been computed in GWF2BCF7AR. Convertible cells require computation of conductance and conversion between wet and dry, which is done by calling secondary subroutine SGWF2BCF7H.

Arguments:

KITER – Current iteration of solver
KSTP – Current time step
KPER – Current stress period
IGRID – The grid number, which is used for local grid refinement

GWF2BCF7BDS

This subroutine computes the storage budget term.

Arguments:

KSTP – Current time step
KPER – Current stress period
IGRID – The grid number, which is used for local grid refinement

GWF2BCF7BDCH

This subroutine computes the constant-head budget term.

Arguments:

KSTP – Current time step
KPER – Current stress period
IGRID – The grid number, which is used for local grid refinement

GWF2BCF7BDADJ

This subroutine computes the flow between adjacent cells, which is done only if needed. These flows are needed when saving cell-by-cell budget data to a file and when returning them in array BUFF as indicated by variable IBDRET. If values are being returned but not saved, the flows are computed only for a subgrid determined by arguments IC1, IC2, IR1, IR2, IL1, and IL2.

Arguments:

KSTP – Current time step
KPER – Current stress period
IDIR – Coordinate direction flag:
    1 – Across columns (through right face)
    2 – Across rows (through front face)
    3 – Across layers (through lower face)
IBDRET – Flag for returning budget values in BUFF:
    0 – Do not return values
    not 0 – Return values
IC1 – First column of subgrid
IC2 – Last column of subgrid
IR1 – First row of subgrid
IR2 – Last row of subgrid
IL1 – First layer of subgrid
IL2 – Last layer of subgrid
IGRID – The grid number, which is used for local grid refinement

Secondary Subroutines

SGWF2BCF7N

This secondary subroutine is called by GWF2BCF7AR to check data for consistency and to perform some initial calculations. A secondary subroutine is used because of the size of the code.

Arguments:

ISS – Steady-state flag:
0 – At least one transient stress period
not 0 – All stress periods are steady state.

SGWF2BCF7N performs its work in the following sequence:

1. Vertical conductance is computed by multiplying vertical leakance by cell area. GWF2BCF7AR has read vertical leakance into array CV.

2. Vertical conductance is saved in CVWD if wetting is active.

3. If IBOUND is 0, horizontal and vertical conductance (CC and CV) are set equal to 0. (A user might make CC and CV arrays constant for all cells as a matter of input convenience.)

4. A check is made to see if IBOUND is not 0 when transmissive properties in all directions are 0. Numeric problems occur in the solvers if a flow equation is constructed in which all conductances to adjacent cells are 0. Accordingly, when this situation is detected for a cell, IBOUND is set to 0 and a message is written to the Listing File. Also, head is set equal to 888.88 as an indicator that the cell has been converted to no flow.

5. For layers where transmissivity is constant, horizontal conductance is computed by calling one of several secondary subroutines as determined by the user-specified option for computing inter-block transmissivity, LAYAVG.

6. Confined storage coefficient and specific yield are multiplied by cell area to obtain storage capacity.

SGWF2BCF7H

This secondary subroutine computes horizontal conductance for a layer from saturated thickness and hydraulic conductivity. SGWF2BCF7H is called by GWF2BCF7FM for water table and fully convertible layers. SGWF2BCF7H converts dry cells to wet if wetting is active and computes saturated thickness for cells. SGWF2BCF7H then calls another secondary subroutine for computing the horizontal branch conductances according to the user-specified option for computing interblock transmissivity (LAYAVG).

Arguments:

K – Layer for which horizontal conductance is being calculated.
KB – Third index of arrays HY and WETDRY corresponding to layer K of the grid
KITER – Current iteration of solver
KSTP – Current time step
KPER – Current stress period

SGWF2BCF7C

This secondary subroutine computes horizontal conductance between nodes using harmonic mean transmissivity. SGWF2BCF7C can be called by SGWF2BCF7H and SGWF2BCF7N. Upon entry, CC contains the transmissivity for cells.

Arguments:

K – Layer for which conductance is being calculated.

SGWF2BCF7A

This secondary subroutine computes horizontal conductance using arithmetic-mean transmissivity. SGWF2BCF7A can be called by SGWF2BCF7H and SGWF2BCF7N. Upon entry, CC contains the transmissivity for cells.

Arguments:

K – Layer for which conductance is being calculated.

SGWF2BCF7L

This secondary subroutine computes horizontal conductance using logarithmic mean transmissivity. SGWF2BCF7L can be called by SGWF2BCF7H and SGWF2BCF7N. Upon entry, CC contains the transmissivity for cells.

Arguments:

K – Layer for which conductance is being calculated.

SGWF2BCF7U

This secondary subroutine computes horizontal conductance using arithmetic mean saturated thickness and logarithmic mean hydraulic conductivity. SGWF2BCF7U can be called by SGWF2BCF7H. Upon entry, CC contains the hydraulic conductivity and BUFF contains the saturated thickness for cells.

Arguments:

K – Layer for which conductance is being calculated.

Layer-Property Flow Package

The Layer-Property Flow (LPF) Package computes conductance terms for flow between cells and storage terms. When simulating confined flow, the computations are straightforward, but complexities such as unconfined conditions, cell drying, cell wetting, and conversion between confined and unconfined conditions cause the code to be much more complex. Support for parameters also adds complexity to the code. Shared data for the LPF Package are declared in Fortran Module GWFLPFMODULE and defined in table 9–6.

Table 9–6. Variables in Fortran module GWFLPFMODULE.

Variable Name	Size	Description
ILPFCB	Scalar	Cell-by-cell budget flag and unit: <0 – Constant-head cell-by-cell budget data are written to the Listing File. 0 – No cell-by-cell budget >0 – Unit number for saving cell-by-cell budget data
IWDFLG	Scalar	Wetting flag: 0 – Wetting is inactive. not 0 – Wetting is active in at least one layer.
IWETIT	Scalar	The iteration interval for attempting to wet cells. Wetting is attempted every IWETIT iteration.
IHDWET	Scalar	Flag indicating which equation to use for defining the head at a cell that has just converted from dry to wet: 0 – HNEW= BOT+WETFCT(Hn-BOT) not 0 – HNEW = BOT+WETFCT(THRESH)
ISFAC	Scalar	SFAC option flag: 0 – Not active not 0 – Active
ICONCV	Scalar	CONSTANTCV option flag: 0 – Not active not 0 – Active
ITHFLG	Scalar	THICKSTRT option flag: 0 – Not active not 0 – Active
NOCVCO	Scalar	NOCVCORRECTION option flag: 0 – Not active not 0 – Active
WETFCT	Scalar	Factor included in the calculation of head at a cell that has just converted from dry to wet.
HDRY	Scalar	When a cell converts to dry, HNEW is set equal to HDRY.
LAYTYP	NLAY	Layer-type code: 0 – A confined layer >0 – A convertible layer <0 – Convertible unless THICKSTRT option is active, in which case the layer is confined. After detecting that the layer should be confined, GWF2LPF7AR changes LAYTYP to 0.
LAYAVG	NLAY	Interblock transmissivity flag for layers: 0 – Harmonic mean 1 – Logarithmic mean 2 – Arithmetic-mean saturated thickness and logarithmic-mean hydraulic conductivity
CHANI	NLAY	Horizontal anisotropy flag or value for layers: ≤0 – Array HANI defines horizontal anisotropy for each cell in the layer. >0 – CHANI is the horizontal anisotropy for the entire layer.
LAYVKA	NLAY	Vertical anisotropy flag for layers: 0 – VKA contains vertical hydraulic conductivity. not 0 – VKA contains the ratio of horizontal to vertical hydraulic conductivity.
LAYWET	NLAY	Wetting flag for layers: 0 – Wetting is inactive. not 0 – Wetting is active.
LAYSTRT	NLAY	Flag indicating layers for which LAYTYP was negative when THICKSTRT was active: 0 – LAYTYP ≥0 or THICKSTRT is not active. not 0 – LAYTYP <0 and THICKSTRT is active.

LAYFLG	6,NLAY	Print codes for printing arrays when they are defined by parameters (k is the layer index): (1,k) – Print code for HK values (2,k) – Print code for VKA values (3,k) – Print code for SC1 values (4,k) – Print code for SC2 values (5,k) – Print code for VKCB values (6,k) – Print code for HANI values
HK	NCOL,NROW,NLAY	Horizontal hydraulic conductivity in the row direction.
VKA	NCOL,NROW,NLAY	Vertical hydraulic conductivity or the ratio of horizontal to vertical hydraulic conductivity depending on LAYVKA.
VKCB	NCOL,NLAY,ncb	Vertical hydraulic conductivity of Quasi–3D confining bed, where ncb is the number of Quasi–3D confining beds.
SC1	NCOL,NROW,NLAY	Confined storage capacity.
SC2	NCOL,NROW,ncvt	Unconfined storage capacity, where ncvt is the number of convertible layers.
HANI	NCOL,NROW,nhani	Horizontal anisotropy, where nhani is the number of layers in which horizontal anisotropy is not constant (see CHANI).
WETDRY	NCOL,NROW,nwet	Wetting threshold combined with wetting direction indicator. Absolute value is the wetting threshold. Negative indicates wetting only from below. 0 indicates no wetting. Positive indicates wetting from sides and below. The third dimension, nwet, is the number of layers where wetting can occur.

LPF consists of six primary subroutines: GWF2LPF7AR, GWF2LPF7AD, GWF2LPF7FM, GWF2LPF7BDADJ, GWF2LPF7BDS, and GWF2LPF7BDCH. The budget procedure consists of three subroutines because LPF computes three budget terms: flow between adjacent cells, storage, and constant-head flow.

GWF2LPF7AR

This subroutine allocates and reads LPF data. GWF2LPF7AR is fairly complex because it reads a large amount of data, there are many complex data dependencies, and much of the data can be optionally defined using parameters.

Arguments:

IN – The input unit number for the LPF Package
IGRID – The grid number, which is used for local grid refinement

The AR subroutine performs its work in the following sequence:

1. Allocate scalar data. This allows use with local grid refinement, in which this subroutine can be called multiple times to establish multiple grids. The result is that separate memory is allocated for each of the grids.
2. Identify package.
3. Read and write comments and item 1. Check for the SFAC, CONSTANTCV, THICKSTRT, and NOCVCORRECTION options.
4. Allocate and read indicator arrays for layers.
   4A. Print table of option codes for each layer. Set LAYHDT and LAYHDS to 0 for confined cells and to 1 for convertible cells. Set LAYSTRT=1 and set LAYTYP=0 if LAYTYP<0 and THICKSTRT is active.
   4B. Look through indicator arrays to find out how many layer arrays are needed. Print a second table showing the options for each layer in text form.
   4C. Print wetting information.
5. Allocate layer arrays. When an array is unneeded, a single element is allocated so that all pointers are associated.
6. Read parameter definitions. Create flags for each parameter type—the flag is 0 if no parameter of that type is specified and 1 if one or more parameters of that type are specified. If VK or VANI parameters are defined, then SGWF2LPF7CK is called to check that the layers associated with the parameter correspond to the value of LAYVKA. For example, all clusters for a VANI parameter should specify layers for which LAYVKA is not 0.
   6A. If any HANI parameters are defined, then horizontal anisotropy for all layers must be defined using parameters. Thus, CHANI cannot be a positive number for any layer, which indicates that horizontal anisotropy is a constant for a layer.

7. Define arrays for each layer. All arrays except for WETDRY can be defined either by directly reading them or by using parameters. WETDRY can be defined only by directly reading it. If an array is directly read, U2DREL is called to read the array. If an array is defined using parameters, UPARARRSUB1 is called to define the values. After an array is defined using parameters, UPARARRCK is called to check that a value has been defined for every cell. After SC1 and SC2 arrays are defined, secondary subroutine SGWF2LPF7SC is called to compute storage capacity.
8. Prepare and check data. This is performed by secondary subroutine SGWF2LPF7N.
9. Save grid pointers and return.

GWF2LPF7AD

This subroutine prepares LPF data for a new time step. This is done only when wetting is active and the stress period is transient. HOLD for dry cells that are eligible for conversion to wet is set equal to the bottom elevation. This is necessary to cause the change in head over the time step to be properly computed for the storage term in the flow equation. Otherwise, HOLD would have a value of HNOFLO or HDRY depending on whether the cell was initially dry or converted to dry in a prior time step.

Arguments:

KPER – Current stress period
IGRID – The grid number, which is used for local grid refinement

GWF2LPF7FM

This subroutine computes conductance and storage terms in the flow equation. The computations depend to a great extent on the layer type. For confined layers, cell drying and wetting are not allowed, and conductance has already been computed in GWF2LPF7AR. Convertible cells require computation of conductance and conversion between wet and dry.

Arguments:

KITER – Current iteration for the solver
KSTP – Current time step
KPER – Current stress period
IGRID – The grid number, which is used for local grid refinement

The FM subroutine performs its work in the following sequence:

1. Set pointers to the LPF data for the grid, get the steady-state flag for the current stress period, and define the constant ONE.
2. Loop through all layers and compute conductance for convertible layers. If a layer is convertible, call subroutine SGWF2LPF7HCOND to compute horizontal conductance. If a layer is convertible or the layer below is convertible, call SGWF2LPF7VCOND to compute vertical conductance between layers.
3. If stress period is transient, loop through all layers to compute storage terms—steps 4–5. Compute 1/DELT, which is needed to compute storage terms.
4. Test layer to see if convertible or confined.
5. Compute storage terms for a confined layer. SC1 contains storage capacity computed from specific storage. Subtract SC1/DELT from HCOF, and subtract SC1*HOLD/DELT from RHS.
6. Compute storage terms for a convertible layer.
7. Loop through all layers and apply the leakage correction if needed—steps 8 and 9.
8. Compute leakage correction to layer above if the current layer is convertible.
9. Compute leakage correction to layer below if the layer below is convertible.
10. Return.

GWF2LPF7BDADJ

This subroutine computes the flow between adjacent cells, which is done when saving cell-by-cell budget data to disk and when returning them in array BUFF as indicated by variable IBDRET.

Arguments:

KSTP – Current time step
KPER – Current stress period
IDIR – Coordinate direction flag:
    1 – Across columns (through right face)
    2 – Across rows (through front face)
    3 – Across layers (through lower face)
IBDRET – Flag for returning budget values in BUFF:
    0 – Do not return values.
    not 0 – Return values.
IC1 – First column of transport subgrid
IC2 – Last column of transport subgrid
IR1 – First row of transport subgrid
IR2 – Last row of transport subgrid
IL1 – First layer of transport subgrid
IL2 – Last layer of transport subgrid
IGRID – The grid number, which is used for local grid refinement

GWF2LPF7BDS

This subroutine computes the storage budget term.

Arguments:

KSTP – Current time step
KPER – Current stress period
IGRID – The grid number, which is used for local grid refinement

GWF2LPF7BDCH

This subroutine computes the constant-head budget term.

Arguments:

KSTP – Current time step
KPER – Current stress period
IGRID – The grid number, which is used for local grid refinement

Secondary Subroutines

SGWF2LPF7N

This secondary subroutine is called by GWF2LPF7AR to check data for consistency and perform some initial calculations. A secondary subroutine is used because of the size of the code.

Arguments: None

This subroutine performs its work in the following sequence:

1. Define constants ZERO and HCNV. HNEW will be set equal to HCNV at cells that are converted to no flow because all hydraulic conductivity values are 0.
2. Loop through all layers to check that the aquifer hydraulic conductivity in at least one direction is not 0 when IBOUND is not 0. Numerical problems occur in the solvers if a flow equation is constructed in which all conductances to adjacent cells are 0. Accordingly, when this situation is detected for a cell, IBOUND is set to 0 and a message is written to the Listing File. Also, head is set equal to 888.88 as an indicator that the cell has been converted to no flow. The checks depend on whether wetting is active.
3. Wetting is active, so IBOUND and WETDRY must be checked to see if a cell is wet or can become wet.
4. Wetting is inactive, so only IBOUND must be checked to see if a cell is wet.
5. For confined layers, compute horizontal conductance by calling secondary subroutine SGWF2LPF7HCOND.
6. For confined layers, compute vertical conductance by calling secondary subroutine SGWF2LPF7VCOND. Vertical conductance between two layers is constant only when both layers are confined.
7. Return.

SGWF2LPF7HCOND

This subroutine is called by GWF2LPF7FM and SGWF2LPF7N to compute horizontal branch conductance for a layer. SGWF2LPF7HCOND works for confined or convertible layers.

Arguments:

K – Layer for which conductance is being computed
KITER – Current iteration for the solver
KSTP – Current time step
KPER – Current stress period

SGWF2LPF7WET

This subroutine is called by SGWF2LPF7HCOND to scan all cells in a wettable layer and converts dry cells to wet according to the wetting criteria.

Arguments:

K – Layer for which dry cells are being converted.
KITER – Current iteration for the solver
KSTP – Current time step
KPER – Current stress period
IHDCNV – Cell conversion label print flag:
    0 – Label for cell conversion table has not been printed.
    not 0 – Label for cell conversion table has been printed.
NCNVRT – The number of cells in buffer arrays ICNVRT, JCNVRT, and ACNVRT
ICNVRT – Row indices for cells that convert
JCNVRT – Column indices for cells that convert
ACNVRT – Labels for cells that convert

This subroutine performs its work in the following sequence:

1. After defining the constant ZERO, loop through all cells of the layer. The layer is first argument, which is K.
2. Test to see if the cell is dry and that WETDRY is not 0. If so, the cell is eligible to convert to wet, and steps 3–7 are followed to find out if the cell should convert.
3. Compute the wetting elevation, TURNON, which is the absolute value of WETDRY plus the bottom elevation.
4. Check head in the cell below to see if the head exceeds the wetting elevation. If so, then GO TO statement 50, which converts the cell to wet.

5. If WETDRY is positive, then head in four surrounding cells also can cause conversion to wet. Check the four adjacent cells one at a time. In addition to the head exceeding the wetting threshold, the adjacent cell also must not have converted to wet this iteration. Otherwise, one cell going wet could cause an avalanche across the grid. An IBOUND value of 30000 indicates a cell that just went wet. If the wetting elevation is exceeded, jump to statement 50, which converts the cell to wet.
6. The wetting criteria have not been met, so go to the next cell by jumping to statement 100, which is the end of the loop.
7. Convert the cell to wet. Call SGWF2LPF7WDMSG to write a message saying the cell converted to wet. Select the equation to use for the head in the converted cell and set the head. Set IBOUND equal to 30000 as an indicator that the cell is wet but that it just converted to wet. This will be changed to 1 after returning to SGWF2LPF7HCOND.
8. This statement ends the loop for all cells in the layer.
9. Return.

SGWF2LPF7WDMSG

This subroutine is called by SGWF2LPF7HCOND or SGWF2LPF7WET to store and print wet and dry messages. To save space, five messages are printed on a line. NCNVRT counts the number of messages in the buffer. After five messages have accumulated or when the ICODE argument is 0 with any number of messages, a line is printed. Before printing the first line, a title is printed. IHDCNV is switched from 0 to 1 after the title is printed.

The format for printing is changed if the number of rows or columns is greater than 999. Five digits are used for row and column numbers rather than three. This makes the line a little longer so that the title does not align perfectly and packs the values close together, but the fields will not overflow.

Arguments:

ICODE – Operation code:
  0 – Print a partially full buffer.
  not 0 – Add a cell to the buffer.
NCNVRT – The number of cells in buffer arrays ICNVRT, JCNVRT, and ACNVRT
ICNVRT – Row indices for cells that convert
JCNVRT – Column indices for cells that convert
ACNVRT – Labels for cells that convert
IHDCNV – Cell conversion label print flag:
  0 – Label for cell conversion table has not been printed.
  not 0 – Label for cell conversion table has been printed.
IOUT – File unit number for Listing File
KITER – Current iteration for the solver
J – Column of converted cell
I – Row of converted cell
K – Layer of converted cell
KSTP – Current time step
KPER – Current stress period
NCOL – Number of columns in the grid
NROW – Number of rows in the grid

SGWF2LPF7HHARM

This secondary subroutine computes horizontal conductance between nodes using harmonic mean transmissivity. SGWF2LPF7HHARM can be called by SGWF2BCF7HCOND. Upon entry, CC contains the cell thickness.

Arguments:

K – Layer for which conductance is being computed.

SGWF2LPF7HLOG

This secondary subroutine computes horizontal conductance between nodes using logarithmic mean transmissivity. SGWF2LPF7HLOG can be called by SGWF2BCF7HCOND. Upon entry, CC contains the cell thickness.

Arguments:

K – Layer for which conductance is being computed.

SGWF2LPF7HUNCNF

This secondary subroutine computes horizontal conductance between nodes using arithmetic mean saturated thickness and logarithmic mean transmissivity. SGWF2LPF7HUNCNF can be called by SGWF2BCF7HCOND. Upon entry, CC contains the cell thickness.

Arguments:

K – Layer for which conductance is being computed.

SGWF2LPF7VCOND

This subroutine is called by GWF2LPF7FM and SGWF2LPF7N to compute vertical branch conductance between a layer and the layer below.

Arguments:

K – Upper layer for which conductance is being computed.

SGWF2LPF7SC

This subroutine is called by SGWF2LPF7AR to compute storage capacity. If the ISPST argument is not 0, argument SC is specific storage. SC is multiplied by cell area and cell thickness to get storage capacity. If ISPST is 0, SC is specific yield. SC is multiplied by cell area to get storage capacity.

Arguments:

K – Layer for which conductance is being computed.

SGWF2LPF7CK

If a VK or VANI parameter is defined, then SGWF2LPF7CK is called by GWF2LPF7AR to check that the layers associated with the parameter correspond to the value of LAYVKA. All clusters for a VANI parameter should specify layers for which LAYVKA is not 0. All clusters for a VK parameter should specify layers for which LAYVKA is 0.

Arguments:

SC – Specific storage or specific yield that will be converted to storage capacity.
K – Layer for which storage capacity is being computed.
ISPST – Flag indicating the data initially in SC:
    0 – Specific yield
    not 0 – Specific storage

Horizontal Flow Barrier Package

The Horizontal Flow Barrier (HFB) Package simulates flow barriers by reducing horizontal conductance. Shared data for the HFB Package are declared in Fortran Module GWFHFBMODULE and defined in table 9–7.

Table 9–7. Variables in Fortran module GWFHFBMODULE.

Variable Name	Size	Description
MXHFB	Scalar	The second dimension of HFB, which includes space for the active horizontal-flow barriers and the parameter definitions for horizontal-flow barriers.
NHFB	Scalar	The number of active horizontal-flow barriers.
IPRHFB	Scalar	Flag for printing HFB data – 0 indicates do not print, 1 indicates print.
NHFBNP	Scalar	The number of nonparameter horizontal-flow barriers.
NPHFB	Scalar	The number of HFB parameters.
IHFBPB	Scalar	The value of the second index in HFB at which parameter data begins.
HFB	7,MXHFB	Horizontal-flow barrier list, which includes space for the active horizontal-flow barriers and the parameter definitions for horizontal-flow barriers.

The HFB Package consists of two primary subroutines GWF2HFB7AR and GWF2WEL7FM, and four secondary subroutines SGWF2HFB7MC, SGWF2HFB7CK, SGWF2HFB7RL, and SGWF2HFB7SUB. SGWF2HFB7CK checks to insure that the two cells that define each flow barrier are adjacent. SGWF2HFB7RL is used in the AR subroutine to read lists of barriers. SGWF2HFB7SUB is used in the AR subroutine to substitute the barriers for active parameters into the list of active barriers.

Variable LAYHDT is used to determine if HFB can be used. The value for LAYHDT will be 0 for confined layers and 1 for layers for which transmissivity is head dependent. SGWF2HFB7MC is called by the Allocate and Read Procedure subroutine to compute the modified horizontal conductance caused by barriers for confined layers. When transmissivity is head dependent, the conductance of the barrier depends on the saturated thickness, which is computed in the Formulate Procedure subroutine.

GWF2HFB7AR

This subroutine allocates memory for HFB and reads all HFB input data. This subroutine must be called after the primary internal flow package (either Block-Centered Flow or Layer-Property Flow) Formulate subroutine. GWF2HFB7AR allocates memory for HFB much as is done for stress packages that use list data such as the River and Well Packages; however, the list of data is different from the data lists for the stress packages. Each line of data for the stress packages has three Integers (layer, row, and column) while HFB has five Integers: layer, row 1, column 1, row 2, and column 2. Accordingly, HFB cannot use the ULSTRD and UPARLSTSUB utility subroutines used to handle data for the stress packages. The SGWF2HFB7RL and the SGWF2HFB7SUB routines are similar to ULSTRD and UPARLSTSUB.

If variable LAYHDT is negative, then HFB will abort the simulation because an unrecognized internal flow package is in use. The value for LAYHDT will be 0 for confined layers, and SGWF2HFB7MC is called to compute the modified horizontal conductance caused by barriers for confined layers. At confined cells, the conductance does not change during the simulation. When transmissivity is head dependent, the conductance of the barrier depends on the saturated thickness, which is computed in the Formulate Procedure subroutine.

Arguments:

IN – The input unit number for the HFB Package
IGRID – The grid number, which is used for local grid refinement

GWF2HFB7FM

This subroutine modifies horizontal conductance between convertible cells for which a horizontal-flow barrier is specified. This subroutine must be called after the primary internal flow package (either Block-Centered Flow or Layer-Property Flow) Formulate subroutine. For layers for which transmissivity is head dependent (LAYHDT=1), the modified conductance caused by the barrier is computed from the saturated thickness. At confined cells (LAYHDT=1), no additional computation is made for barriers because this has already been done in GWF2HFB7AR.

Arguments:

IGRID – The grid number, which is used for local grid refinement

SGWF2HFB7MC

This subroutine modifies horizontal conductance between confined cells for which a horizontal-flow barrier is specified. This subroutine must be called after the primary internal flow package (either Block-Centered Flow or Layer-Property Flow) Allocate and Read Procedure subroutine.

Arguments: None

SGWF2HFB7CK

This subroutine checks a group of flow barriers to insure that the two cells that define each flow barrier are adjacent. This subroutine is called for each HFB parameter and for the list of barriers defined without parameters.

Arguments:

IB1 – Location in HFB of first barrier to check
IB2 – Location in HFB of last location to check

SGWF2HFB7RL

This subroutine reads a list of barriers and writes the barriers to the Listing File.

Arguments:

NLIST – Number of horizontal-flow barriers to read
HFB – List of horizontal-flow barriers
LSTBEG – Value of second index of HFB for first cell location
MXHFB – Dimensioned size of second index of HFB
INPACK – File unit number for reading barrier data
IOUT – File unit number for Listing File
LABEL – Label to be written above the list of barriers in the Listing File
NCOL – Number of columns in the grid
NROW – Number of rows in the grid
NLAY – Number of layers in the grid
IPRFLG – Print flag:
    Not 1 – Do not write barriers to Listing File.
    1 – Write barriers to Listing File.

SGWF2HFB7SUB

This subroutine reads a parameter name and substitutes values into the list of active barriers.

Arguments:

IN – File unit number for reading barrier data
PACK – Package name
IOUTU – Absolute value is the file unit number for Listing File. Negative value indicates do not print.
PTYP – Parameter type
HFB – Horizontal-flow barrier list
LSTVL – Dimensioned size of first index of HFB
MXHFB – Dimensioned size of second index of HFB
MXACTFB – Value of the second index of HFB that is the end of the active data for the current stress period
NHFB – The number of horizontal-flow barriers in the active part of HFB for the current stress period
LABEL – Label to be written above the list of barriers in the Listing File

Well Package

The Well (WEL) Package adds terms to the flow equation to represent wells. Shared data for the WEL Package are declared in Fortran Module GWFWELMODULE and defined in table 9–8.

Table 9–8. Variables in Fortran module GWFWELMODULE.

["C*n" in size column indicates a character variable of n characters]

Variable Name	Size	Description
NWELLS	Scalar	The number of wells in the current stress period.
MXWELL	Scalar	The second dimension of WELL, which includes space for the active wells in a stress period and all wells defined by parameters.
NWELVL	Scalar	The number of the first dimension of WELL array, which includes four input values, the auxiliary data, and the well budget term.
IWELCB	Scalar	Cell-by-cell budget flag and unit. Negative indicates cell-by-cell budget is written to the Listing File, 0 indicates no cell-by-cell budget, and positive is the unit number for saving cell-by-cell budget data.
IPRWEL	Scalar	Flag for printing well data – 0 indicates do not print, 1 indicates print.
NPWEL	Scalar	The number of well parameters.
IWELPB	Scalar	The value of the second index in WELL at which parameter data begins.
NNPWEL	Scalar	The number of nonparameter wells in the current stress period.
WELAUX	C*16,20	The name of auxiliary variables.
WELL	NWELVL,MXWELL	Well list, which includes space for the active wells in a stress period and all wells defined by parameters.

The Well (WEL) Package consists of four primary subroutines: GWF2WEL7AR, GWF2WEL7RP, GWF2WEL7FM, and GWF2WEL7BD. Subroutine ULSTRD is used in the AR subroutine to read lists of wells that are defined by using parameters and in the RP subroutine to read the lists of wells that are defined without using parameters. Details of the code are not contained here. The code is similar to, but simpler than, the River Package, which is fully documented later in this chapter.

Recharge Package

The Recharge (RCH) Package adds terms to the flow equation to represent areal recharge to the ground-water system. Shared data for the RCH Package are declared in Fortran Module GWFRCHMODULE and defined in table 9–9.

Table 9–9. Variables in Fortran module GWFRCHMODULE.

Variable Name	Size	Description
NRCHOP	Scalar	The recharge option: 1 – Layer 1 2 – Layer specified in IRCH 3 – Uppermost variable-head cell
IRCHCB	Scalar	The unit number for saving RCH budget data.
NPRCH	Scalar	The number of RCH parameters.
IRCHPF	Scalar	The format code for printing recharge data defined using parameters.
RECH	NCOL,NROW	Recharge rate. Initially, recharge flux is read into RECH and then multiplied by cell area.
IRCH	NCOL,NROW	Layer receiving recharge when NRCHOP is 2 or 3.

The Recharge (RCH) Package consists of four primary subroutines: GWF2RCH7AR, GWF2RCH7RP, GWF2RCH7FM, and GWF2RCH7BD. Details of the code are provided for each subroutine.

GWF2RCH7AR

Arguments:

IN – The input unit number for the RCH Package
IGRID – The grid number, which is used for local grid refinement

The AR subroutine performs its work in the following sequence:

1. Allocate scalar variables. This is necessary for the grid refinement capability. Each invocation of this AR subroutine will cause new memory to be allocated. In step 9, a pointer to this memory location is saved.
2. Identify package and initialize the parameter format print flag.
3. Read the recharge option and the cell-by-cell budget flag. The format depends on the free format flag, IFREFM, which is defined in the Basic Package.
4. Check if NRCHOP is valid. If invalid, write a message and stop the simulation.
5. Write the recharge option.
6. If cell-by-cell budget will be written (IRCHCB >0), then write IRCHCB, which is the unit number for saving cell-by-cell budget data.
7. Allocate memory for RECH and IRCH.
8. Read named parameters, if any.
9. Save recharge pointers for the current subgrid. Return.

GWF2RCH7RP

Arguments:

IN – The input unit number for the RCH Package
IGRID – The grid number, which is used for local grid refinement

The RP subroutine performs its work in the following sequence:
1. Set the recharge pointers to point to the current subgrid.
2. Read flags that indicate whether data are being reused this stress period. INIRCH is read only if NRCHOP is 2. The format depends on the free format flag, IFREFM, which is defined in the Basic Package.
3. Test INRECH to see if recharge is being reused or read.
   3A. INRECH<0, which indicates recharge is being reused from previous stress period – either parameter or nonparameter values. Write a message and skip to step 5.
   3B. INRECH is greater than or equal to 0. Read recharge data for stress period.
      3BA. If no parameters, read RECH as an array. Then skip to step 4.
      3BB. If parameters have been input, define RECH by using parameters. Start by resetting any previously active parameters (Subroutine PRESET). INRECH is the number of parameters this stress period, make sure that INRECH>0. Read the parameter names and substitute the parameter values (Subroutine UPARARRSUB2) into RECH.
4. Multiply RECH by cell area.
5. Define IRCH if NRCHOP is 2. Test INIRCH to see if IRCH is being reused or read.
   5A. INIRCH<0, which indicates that IRCH is being reused from previous stress period.
   5B. INIRCH is greater than or equal to 0. Read IRCH as an array.
6. Return.

GWF2RCH7FM

Arguments:

IGRID – The grid number, which is used for local grid refinement

The FM subroutine performs its work in the following sequence:
1. Set the recharge pointers to point to the current subgrid.
2. Determine the recharge option (NRCHOP).
3. Recharge option 1—recharge is to the top layer.
   3A. Subtract recharge from RHS at cells where IBOUND>0.
4. Recharge option 2—recharge is to the layer specified in IRCH.
   4A. Subtract recharge from RHS at cells where IBOUND>0.
5. Recharge option 3—recharge is to the highest variable-head cell, except the recharge cannot be transmitted through a constant-head cell.
   5A. If the cell is constant head, skip to the next horizontal cell location.
   5B. Subtract recharge from RHS at cells where IBOUND>0.
6. Return.

GWF2RCH7BD

Arguments:

KSTP – Current time step
KPER – Current stress period
IGRID – The grid number, which is used for local grid refinement

The BD subroutine performs its work in the following sequence:
1. Set the recharge pointers to point to the current subgrid.
2. Clear the inflow and outflow flow rate accumulators.
3. Clear the BUFF array, which will hold the flow rate at each cell. Set IBD, which is the flag that indicates if cell-by-cell budget is saved. IBD=0 for no saving, IBD=1 for save using original form, and IBD=2 for save using compact budget.

4. Determine the recharge option (NRCHOP).
5. Recharge option 1—recharge is to top layer. Loop through all cells in a layer.
   5A. If the cell is variable head, save recharge rate in Q and QQ. QQ is double precision.
   5B. Also save the recharge rate in BUFF.
   5C. Add recharge to the inflow or outflow accumulators.
6. Recharge option 2—recharge is to the layer specified in IRCH. Loop through all cells in a layer.
   6A. Get the layer index for recharge from IRCH.
   6B. If the cell is variable head, save recharge rate in Q and QQ. QQ is double precision.
   6C. Also save the recharge rate in BUFF.
   6D. Add recharge to the inflow or outflow accumulators.
7. Recharge option 3—recharge is to the highest variable-head cell, except the recharge cannot be transmitted through a constant-head cell. Loop through all cells in a layer.
   7A. Initialize the layer index (IRCH) to 1, and loop through all cells in a vertical column.
   7B. If the cell is constant head, skip to next horizontal cell location.
   7C. If the cell is variable head, save recharge rate in Q and QQ. QQ is double precision.
   7D. Also save the recharge rate in BUFF, and save the layer number in IRCH.
   7E. Add recharge to the inflow or outflow accumulators.
8. Call budget appropriate utility module if cell-by-cell budget should be saved. UBUDSV is for the original format, and UBDSV3 is for the compact format.
9. Move total inflow and outflow rates into VBVL.
10. Accumulate inflow and outflow volumes into VBVL.
11. Move budget term name into VBNM.
12. Increment budget term counter (MSUM).
13. Return.

General-Head Boundary Package

The General-Head Boundary (GHB) Package adds terms to the flow equation to represent head-dependent boundaries. Shared data for the GHB Package are declared in Fortran Module GWGHBLMODULE and defined in table 9–10.

Table 9–10. Variables in Fortran module GWFGHBMODULE.

["C*n" in size column indicates a character variable of n characters]

Variable Name	Size	Description
NBOUND	Scalar	The number of general-head boundaries in the current stress period.
MXBND	Scalar	The second dimension of BNDS, which includes space for the active general-head boundaries in a stress period and all general-head boundaries defined by parameters.
NGHBVL	Scalar	The number of the first dimension of BNDS array, which includes five input values, the auxiliary data, and the general-head boundary budget term.
IGHBCB	Scalar	The unit number for saving GHB budget data.
IPRGHB	Scalar	Flag for printing GHB data – 0 indicates do not print, 1 indicates print.
NPGHB	Scalar	The number of GHB parameters.
IGHBPB	Scalar	The value of the second index in BNDS at which parameter data begins.
NNPGHB	Scalar	The number of nonparameter general-head boundaries in the current stress period.
GHBAUX	C*16,20	The name of auxiliary variables.
BNDS	NGHBVL,MXBND	Boundary list, which includes space for the active general-head boundaries in a stress period and all general-head boundaries defined by parameters.

The General-Head Boundary (GHB) Package consists of four primary subroutines: GWF2GHB7AR, GWF2GHB7RP, GWF2GHB7FM, and GWF2GHB7BD. Subroutine ULSTRD is used in the AR subroutine to read lists of general-head boundaries that are defined using parameters and in the RP subroutine to read the lists of general-head boundaries that are defined without using parameters. Details of the code are not contained here. The code is similar to, but simpler than, the River Package, which is fully documented later in this chapter.

River Package

The River (RIV) Package adds terms to the flow equation to represent rivers. Shared data for the RIV Package are declared in Fortran Module GWFRIVMODULE and defined in table 9–11.

Table 9–11. Variables in Fortran module GWFRIVMODULE.

["C*n" in size column indicates a character variable of n characters]

Variable Name	Size	Description
NRIVER	Scalar	The number of river reaches in the current stress period.
MXRIVR	Scalar	The second dimension of RIVR, which includes space for the active river reaches in a stress period and all river reaches defined by parameters.
NRIVVL	Scalar	The number of the first dimension of RIVR array, which includes six input values, the auxiliary data, and the river budget term.
IRIVCB	Scalar	The unit number for saving RIV budget data.
IPRRIV	Scalar	Flag for printing RIV data – 0 indicates do not print, 1 indicates print.
NPRIV	Scalar	The number of RIV parameters.
IRIVPB	Scalar	The value of the second index in RIVR at which parameter data begins.
NNPRIV	Scalar	The number of nonparameter river reaches in the current stress period.
RIVAUX	C*16,20	The name of auxiliary variables.
RIVR	NRIVVL,MXRIVR	River reach list, which includes space for the active river reaches in a stress period and all river reaches defined by parameters.

The River (RIV) Package consists of four primary subroutines: GWF2RIV7AR, GWF2RIV7RP, GWF2RIV7FM, and GWF2RIV7BD. Details of the code are provided for each subroutine.

GWF2RIV7AR

Arguments:

IN – The input unit number for the RIV Package
IGRID – The grid number, which is used for local grid refinement

The AR subroutine work is performed in the following sequence:

1. Allocate scalar variables. This is necessary for the local grid refinement capability. Each invocation of this AR subroutine will cause new memory to be allocated. In step 7, the pointers to this memory are saved.
2. Identify package and initialize NRIVER and NNPRIV.
3. READ items 0, 1, and 2. Item 0 consists of optional comments. URDCOM reads lines until a non-comment line is found. After URDCOM, LINE contains the first line after item 0, which is either item 1 or item 2 because item 1 is also optional. UPARLSTAL examines LINE to see if it is item 1, which specifies that parameters are being used. If parameters are being used, UPARLSTAL decodes the parameter information and reads the next line into LINE. Thus, after URDCOM and UPARLSTAL, LINE contains item 2 in text form. Item 2 is then decoded from LINE. The use of fixed or free format is supported.
4. Look at the end of LINE to find auxiliary variables and the no print option.
5. Allocate memory for the RIVR variable.
6. Read named parameters. For each parameter, call UPARLSTRP to read the parameter header record.
   6A. If a simple parameter, use ULSTRD to read the reaches.
   6B. If parameter has instances, for each instance call UINSRP to read the instance name and ULSTRD to read reaches.
7. Save river pointers for the current subgrid. Return.

GWF2RIV7RP

Arguments:

IN – The input unit number for the RIV Package
IGRID – The grid number, which is used for local grid refinement

The RP subroutine performs its work in the following sequence:

1. Set river pointers to point to the current subgrid.
2. Read ITMP and NPRIV using free or fixed format. NPRIV is the number of parameters, which is read only if river parameters have been defined. ITMP is a flag that specifies how nonparameter reaches are defined for the stress period. A negative value indicates reaches from the previous stress period should be reused. A non-negative value indicates the number of nonparameter reaches.
3. Calculate the number of auxiliary data values and the output unit. NRIVVL is the total number of values in RIVR for each reach. Three values define the reach cell, three values define reach properties, and one value is used for saving budget data. Thus, NRIVVL–7 is the number of auxiliary data values.
4. Determine NNPRIV, the number of nonparameter reaches for the stress period. If ITMP is non-negative, NNPRIV is ITMP. IF ITMP is negative, NNPRIV is unchanged from the last stress period.
5. Read new nonparameter reaches. Before reading them, make sure that the river reach list has sufficient room. Read the reaches using subroutine ULSTRD. Set the number of rivers (NRIVER) equal to NNPRIV.
6. Call subroutine PRESET to deactivate all river parameters. For each parameter, call UPARLSTSUB to read the parameter name and substitute reaches into the active part of RIVR. UPARLSTSUB updates NRIVER and checks to make sure RIVR has sufficient room for the data.
7. Print the number of river reaches in the stress period.
8. Return.

GWF2RIV7FM

Arguments:

IGRID – The grid number, which is used for local grid refinement

The FM subroutine performs its work in the following sequence:

1. Set river pointers to point to the current subgrid.
2. Return if no active river reaches for this time step.
3. Repeat steps 4 through 9 for each reach.
4. Get cell indices for reach from RIVR.
5. If cell is constant-head or no-flow, skip this reach.
6. Get reach properties from RIVR. RRBOT is double precision equivalent of RBOT.
7. Compare current value of head (HNEW) for cell to the elevation of the bottom of the riverbed (RRBOT), which determines whether seepage is constant or head dependent.
8. Head is greater than RRBOT, so seepage is head dependent – add terms to HCOF and RHS.
9. Head is less than or equal to RRBOT, so seepage is constant – add term to RHS only.
10. Return

GWF2RIV7BD

Arguments:

KSTP – Current time step
KPER – Current stress period
IGRID – The grid number, which is used for local grid refinement

The BD subroutine performs its work in the following sequence:

1. Set river pointers to point to the current subgrid.
2. Initialize cell-by-cell flow term flag (IBD) and budget accumulators (RATIN and RATOUT). IBD is 0 for no cell-by-cell budget. IBD is -1 for writing cell-by-cell budget to Listing File. IBD is 1 for writing cell-by-cell budget to a noncompact budget file. IBD is 2 for writing cell-by-cell budget to a compact budget file.
3. If writing cell-by-cell data in compact form, call UBDSV4 to write the budget header.
4. Clear BUFF, which always stores the cell-by-cell budget for internal use even if not written to a file.
5. If no reaches for this stress period, skip to step 7. River seepage does not exist for this time step.
6. Repeat steps 6A through 6L for each reach.
   6A. Get cell indices for reach from RIVR, and initialize reach seepage to 0.
   6B. If cell is no flow or constant head, skip to 6L. Reach seepage (RATE) already has been initialized to 0.
   6C. Get reach properties from RIVR.
   6D. Compare current value of head (HNEW) for cell to the elevation of the bottom of the riverbed (RRBOT), which determines whether seepage is constant or head dependent.
   6E. Head is greater than RRBOT, so compute head-dependent seepage.
   6F. Head is less than or equal to RRBOT, so compute constant seepage.
   6G. Write seepage to Listing File if IBD is negative.
   6H. Add seepage to BUFF.
   6I. Compare seepage to 0.
   6J. Negative seepage indicates outflow from aquifer, which is subtracted from the outflow accumulator, RATOUT.
   6K. Positive (or 0) seepage indicates inflow to aquifer, which is added to the inflow accumulator, RATIN.
   6L. Save seepage for reach if writing compact form of cell-by-cell budget.
7. Accumulate rates and volumes in VBVL. Move the budget term name into VBNM.
8. Increment the budget term number (MSUM).
9. Return.

Drain Package

The Drain (DRN) Package adds terms to the flow equation to represent drains. Shared data for the DRN Package are declared in Fortran Module GWFDRNMODULE and defined in table 9–12.

Table 9–12. Variables in Fortran module GWFDRNMODULE.

["C*n" in size column indicates a character variable of n characters]

Variable Name	Size	Description
NDRAIN	Scalar	The number of drains in the current stress period.
MXDRN	Scalar	The second dimension of DRAI, which includes space for the active drains in a stress period and all drains defined by parameters.
NDRNVL	Scalar	The number of the first dimension of DRAI array, which includes five input values, the auxiliary data, and the drain budget term.
IDRNCB	Scalar	The unit number for saving DRN budget data.
IPRDRN	Scalar	Flag for printing DRN data – 0 indicates do not print, 1 indicates print.
NPDRN	Scalar	The number of DRN parameters.
IDRNPB	Scalar	The value of the second index in DRAI at which parameter data begins.
NNPDRN	Scalar	The number of nonparameter drains in the current stress period.
DRNAUX	C*16,20	The name of auxiliary variables.
DRAI	NDRNVL,MXDRN	Drain list, which includes space for the active drains in a stress period and all drains defined by parameters.

The Drain (DRN) Package consists of four primary subroutines: GWF2DRN7AR, GWF2DRN7RP, GWF2DRN7FM, and GWF2DRN7BD. Subroutine ULSTRD is used in the AR subroutine to read lists of drains that are defined by using parameters and in the RP subroutine to read the lists of drains that are defined without using parameters. Details of the code are not contained here. The code is similar to, but simpler than, the River Package, which is fully documented earlier in this chapter.

Evapotranspiration Package

The Evapotranspiration (EVT) Package adds terms to the flow equation to represent evapotranspiration from the ground-water system. Shared data for the EVT Package are declared in Fortran Module GWFEVTMODULE and defined in table 9–13.

Table 9–13. Variables in Fortran module GWFEVTMODULE.

Variable Name	Size	Description
NEVTOP	Scalar	The evapotranspiration option: 1 – Layer 1 2 – Layer specified in IEVT 3 – Uppermost variable-head cell
IEVTCB	Scalar	The unit number for saving EVT budget data.
NPEVT	Scalar	The number of EVT parameters.
IEVTPF	Scalar	The format code for printing evapotranspiration data defined using parameters.
EVTR	NCOL,NROW	Maximum evapotranspiration rate. Initially evapotranspiration flux is read into EVTR, and then multiplied by cell area.
EXDP	NCOL,NROW	Extinction depth.
SURF	NCOL,NROW	Elevation at which evapotranspiration becomes the maximum.
IEVT	NCOL,NROW	Layer receiving recharge when NEVTOP is 2 or 3.

The Evapotranspiration (EVT) Package consists of four primary subroutines: GWF2EVT7AR, GWF2EVT7RP, GWF2EVT7FM, and GWF2EVT7BD. Details of the code are not contained here because the code is similar to the Recharge Package, which is fully documented earlier in this chapter. The differences are minor. EVT adds terms to both RHS and HCOF; whereas RCH adds terms only to RHS. The EVT Package also reads more arrays than RCH reads, but the same approach for reading is used. When formulating the flow equation and computing the budget, EVT has one loop for all cells in a layer, and inside this loop a test is made to see which evapotranspiration option is being used. Conversely, RCH first tests to find which recharge option is being used, and for the selected option a loop runs for all the cells in one layer. The effect is the same for both approaches. The EVT approach results in a more compact code, but is slightly less efficient because the check for the option is done for every horizontal cell.

Strongly Implicit Procedure Package

The Strongly Implicit Procedure (SIP) Package is one of the solvers that can be used to solve the simultaneous equations resulting from the finite-difference approximation. Shared data for the SIP Package are declared in Fortran Module GWFSIPMODULE and defined in table 9–14.

Table 9–14. Variables in Fortran module GWFSIPMODULE.

Variable Name	Size	Description
NPARM	Scalar	The number of iteration parameters.
IPCALC	Scalar	Iteration parameter calculation flag: 0 – WSEED will be specified by the user. not 0 – WSEED will be calculated by the program.
IPRSIP	Scalar	Time step interval for printing table of maximum head change each iteration.
HCLOSE	Scalar	Head closure criterion for convergence.
ACCL	Scalar	Acceleration parameter.
W	NPARM	Iteration parameters.
EL	NCOL,NROW,NLAY	One of the diagonals in the upper triangular factor of [A+B].
FL	NCOL,NROW,NLAY	One of the diagonals in the upper triangular factor of [A+B].
GL	NCOL,NROW,NLAY	One of the diagonals in the upper triangular factor of [A+B].
V	NCOL,NROW,NLAY	Intermediate solution result.
HDCG	MXITER	Maximum head change for each iteration.
LRCH	3,MXITER	Layer, row, and column of the cell containing the maximum head change for each iteration.

The Strongly Implicit Procedure (SIP) Package consists of two primary subroutines, SIP7AR and SIP7AP. Except for the change to Fortran modules for memory allocation, SIP7 is identical to the SIP1 code originally documented in MODFLOW (McDonald and Harbaugh, 1988). Readers are referred to that documentation for additional details.

All data used by subroutine SIP7AP are passed as subroutine arguments. This results in faster execution on some computers as compared to passing data by using Fortran modules. Also for computational performance in SIP7AP, the arrays storing three-dimensional data for cells in the grid are accessed as one-dimensional arrays.

Preconditioned Conjugate-Gradient Package

The Preconditioned Conjugate-Gradient (PCG) Package is one of the solvers that can be used to solve the simultaneous equations resulting from the finite-difference approximation. Shared data for the PCG Package are declared in Fortran Module GWFPCGMODULE and defined in table 9–15.

Table 9–15. Variables in Fortran module GWFPCGMODULE.

Variable Name	Size	Description
ITER1	Scalar	The maximum number of inner iterations.
NPCOND	Scalar	Preconditioner flag: 1 – Incomplete Cholesky with row-sums agreement 2 – Polynomial
NBPOL	Scalar	When using polynomial preconditioning, flag for computing the upper bound of the maximum eigenvalue: 2 – A value of 2.0 is used. not 2 – The program computes the value.
IPRPCG	Scalar	Time step interval for printing table of maximum residual and head change each iteration.
MUTPCG	Scalar	Muting flag for printout: 0 – Print the convergence table. 1 – Print only the total number of iterations 2 – Do not print any convergence information. 3 – Print convergence information only if convergence fails.
NITER	Scalar	Inner iteration counter.
HCLOSEPCG	Scalar	Head closure criterion for convergence.
RCLOSE	Scalar	Residual closure criterion for convergence.
RELAX	Scalar	Relaxation parameter.
DAMP	Scalar	Damping parameter.
VPCG	NCOL,NROW,NLAY	Intermediate solution result. (Double precision)
SS	NCOL,NROW,NLAY	Matrix in PCG algorithm. (Double precision)
P	NCOL,NROW,NLAY	Matrix in PCG algorithm. (Double precision)
RES	NCOL,NROW,NLAY	The flow equation residual. (Double precision)
HPCG	NCOL,NROW,NLAY	Head at the beginning of an outer iteration. (Double precision)
CD	NCOL,NROW,NLAY	The main diagonal of [U] for incomplete Cholesky preconditioning.
HCSV	NCOL,NROW,NLAY	HCOF is saved in HCSV when using polynomial preconditioning.
LHCH	3,MXITER*ITER1	Layer, row, and column of the cell containing the maximum head change for each iteration.
HDCG	MXITER	Maximum head change for each iteration.
LRCHPCG	3,MXITER*ITER1	Layer, row, and column of the cell containing the maximum residual for each iteration.
IT1	MXITER*ITER1	Outer iteration flag: 0 – Not the first inner iteration of an outer iteration 1 – The first inner iteration of an outer iteration

The Preconditioned Conjugate-Gradient (PCG) Package consists of two primary subroutines, PCG7AR and PCG7AP. Except for the change to Fortran modules for memory allocation, PCG7 is the same as the PCG2 code in MODFLOW–2000. Readers are referred to Hill (1990) for additional details.

All data used by subroutine PCG7AP are passed as subroutine arguments. This results in faster execution on some computers as compared to passing data by using Fortran modules. Also for computational performance in PCG7AP, the arrays storing three-dimensional data for cells in the grid are accessed as one-dimensional arrays.

Direct Solver Package

The Direct Solver (DE4) Package is one of the solvers that can be used to solve the simultaneous equations resulting from the finite-difference approximation. Shared data for the DE4 Package are declared in Fortran Module GWFDE4MODULE and defined in table 9–16.

Table 9–16. Variables in Fortran module GWFDE4MODULE.

Variable Name	Size	Description
MXUP	Scalar	The maximum number of equations in the upper part of [A].
MXLOW	Scalar	The maximum number of equations in the lower part of [A].
MXEQ	Scalar	MXUP+MXLOW
MXBW	Scalar	The maximum value of the band width plus one.
ITMX	Scalar	The maximum number of iterations (internal or external) in one time step,
ID4DIR	Scalar	Flag that indicates the relative size of NCOL, NROW, and NLAY: 1 – NLAY≤NROW≤NCOL 2 – NLAY≤NCOL<NROW 3 – NROW<NLAY≤NCOL 4 – NROW≤NCOL<NLAY 5 – NCOL<NLAY≤NROW 6 – NCOL<NROW<NLAY
NITERDE4	Scalar	The maximum number of internal iterations in one time step.
IFREQ	Scalar	Flag indicating the frequency at which [A] changes: 0 – [A] has not changed since the previous solution. 1 – [A] changes only when the time step changes. 2 – [A] can change at the start of each stress period and when the time step size changes. 3 – [A] can change each time DE45AP is called (nonlinear flow equation).
IPRD4	Scalar	Time step interval for printing table of maximum head change for each iteration.
MUTD4	Scalar	Muting flag for printout: 0 – Print the convergence table 1 – Print only the total number of iterations 2 – Do not print any convergence information
ID4DIM	Scalar	First dimension of AU and IUPPNT: 5 – for a two-dimensional grid 7 – for a three-dimensional grid
ACCLDE4	Scalar	Acceleration parameter.
HCLOSEPCG	Scalar	Head closure criterion for convergence.
IUPPNT	ID4DIM,MXUP	The number of off-diagonal coefficients and the equation numbers of the off-diagonal coefficients that are stored in AU.
IEQPNT	NCOL,NROW,NLAY	D4 order number for model cells.
AU	ID4DIM,MXUP	The upper part of [A].
AL	MXBW,MXLOW	The matrix AL in the D4 algorithm.
D4B	MXEQ	The vector {b}.
HDCGDE4	MXITER	Maximum head change for each iteration.
LRCHDE4	3,ITMX	Layer, row, and column of the cell containing the maximum residual for each iteration.

The Direct Solver (DE4) Package consists of two primary subroutines, DE47AR and DE47AP. Except for the change to Fortran modules for memory allocation, DE4 is the same as the DE4 code in MODFLOW–2000. Readers are referred to Harbaugh (1995) for additional details.

All data used by subroutine DE47AP are passed as subroutine arguments. This results in faster execution on some computers as compared to passing data using Fortran modules.

Utility Subroutines

NonParameter Subroutines

The nonparameter utility subroutines use no data from Fortran modules. All data are passed as subroutine arguments, which are documented below. The code contains extensive comments, which provide the primary documentation. Some additional information is provided below for some of the subroutines.

URWORD

Arguments:

LINE – A line of text
ICOL – Element in LINE where the search for a word should start.
ISTART – Element in LINE that is the last character of the word.
ISTOP – Element in LINE that is the last character of the word.
NCODE – A code for converting the word:
    0 – No conversion
    1 – Convert to uppercase
    2 – Convert to Integer
    3 – Convert to Real number
N – Integer value to which the word is converted if indicated by NCODE.
R – Real number value to which the word is converted if indicated by NCODE.
IOUT – File unit number for the Listing File
IN – File unit from which the line was read.

URWORD returns the location of a word in a line of text. A word delimiter is one or more spaces, a comma, or a tab. If the word starts with a single quote, then the quote is removed from the word and the terminating delimiter must be a quote. After finding a word, the line index points to the remainder of the line so that URWORD can be called repeatedly to obtain successive words. The word can be optionally converted to uppercase, or it can be converted to an Integer or Real number. When converting to a number, the word must be 30 characters or less.

UPCASE

Arguments:

WORD – The text string to be converted

UPCASE converts a string to uppercase. The single argument, WORD, is a character variable that contains the string to be converted. The result overwrites WORD.

The algorithm works for a coding method for which the offset between a lowercase and uppercase character is the same for all alphabetic characters. The conversion is done by adding the offset between “a” and “A” to convert the character to uppercase. The conversion algorithm also relies on an assumption that the lowercase characters can be detected by using a relational test for greater than or equal to ‘a’ and less than or equal to‘z’.

URDCOM

Arguments:

IN – File unit number for reading data
IOUT – File unit number for the Listing File
LINE – A buffer for a line of text

URDCOM reads lines from file unit IN until a non-comment line is found. A comment line starts with "#". The comments are written to file unit IOUT if IOUT is greater than 0. The non-comment line that follows any comments is returned to the calling program in LINE.

ULSTRD

Arguments:

NLIST – The number of cells read in the list
RLIST – The array that holds the list of data (LDIM,MXLIST)
LSTBEG – The location in RLIST (second index) where the first data should be read.
LDIM – The first dimension of RLIST
MXLIST – The second dimension of RLIST
IAL – A flag indicating whether LDIM contains an extra location for saving budget data.
INPACK – The input unit number for the package that is calling ULSTRD
IOUT – The output unit number for the Listing File
LABEL – The text label that is written at the start of the list when writing the list to the Listing File.
CAUX – The names of auxiliary variables in the list
NCAUX – The dimension of CAUX
NAUX – The actual number of auxiliary variables used.
IFREFM – Free format flag defined by the Basic Package.
NCOL – The number of columns in the grid
NROW – The number of rows in the grid
NLAY – The number of layers in the grid.
ISCLOC1 – The location in RLIST (first index) that is the first value in RLIST that gets scaled if a scale factor is defined for the list.
ISCLOC2 – The location in RLIST (first index) that is the last value in RLIST that gets scaled if a scale factor is defined for the list.
IPRFLG – Printout flag indicating to print the list if 1.

ULSTRD reads and writes to the Listing File a list of cell locations with data values as used to define river reaches, drains, general-head boundaries, constant-head boundaries, and wells. A lengthy list of arguments is required because of a variety of options that are supported.

ULSTLB

Arguments:

IOUT – File unit number for the Listing File
LABEL – Text to be printed in the label
CAUX – Names of auxiliary data fields
NCAUX – The dimensioned size of CAUX
NAUX – The number of auxiliary fields being used.

This subroutine prints a label for a list of data. The label consists of the text in argument LABEL plus names of auxiliary data fields.

U1DREL

Arguments:

A – The array to read
ANAME – The name of the array
JJ – The dimensioned size of the array
IN – File unit number for reading data
IOUT – File unit number for the Listing File

U1DREL reads a one-dimensional Real array using a control record to define how the data are read. This supports the original control records using numeric codes and the newer control records using words to specify options.

U2DINT

Arguments:

IA – The array to read
ANAME – The name of the array
II – The dimension size for the second index of IA
JJ – The dimensioned size for the first index of IA
K – Layer code for the array:
    <0 – Array is a cross section
    0 – No layer
    >0 – Layer number
IN – File unit number for reading data
IOUT – File unit number for the listing File

U2DINT reads a two-dimensional Integer array using a control record to define how the data are read. This supports the original control records using numeric codes and the newer control records using words to specify options.

U2DREL

Arguments:

A – The array to read
ANAME – The name of the array
II – The dimension size for the second index of A
JJ – The dimensioned size for the first index of A
K – Layer code for the array:
    <0 – Array is a cross section.
    0 – No layer
    >0 – Layer number
IN – File unit number for reading data
IOUT – File unit number for the Listing File

U2DREL reads a two-dimensional Real array using a control record to define how the data are read. This supports the original control records using numeric codes and the newer control records using words to specify options.

UCOLNO

Arguments:

NLBL1 – The start column label (number)
NLBL2 – The stop column label (number)
NSPACE – The number of blank spaces to leave at start of line
NCPL – The number of column numbers per line
NDIG – The number of characters in each column field
IOUT – The file unit number for writing the data – usually the Listing File

UCOLNO writes a column-number header, which is used before writing a two-dimensional array.

ULAPRS

Arguments:

BUF – The array to be written
TEXT – The name of the array being written
KSTP – The current time step
KPER – The current stress period
NCOL – The number of columns in the grid
NROW – The number of rows in the grid
ILAY – The layer of the array to be written
IPRN – Format code
IOUT – The file unit number for writing the data – usually the Listing File

ULAPRS writes a two-dimensional array using strip format.

ULAPRW

Arguments:

BUF – The array to be written
TEXT – The name of the array being written
KSTP – The current time step
KPER – The current stress period
NCOL – The number of columns in the grid
NROW – The number of rows in the grid
ILAY – The layer of the array to be written
IPRN – Format code
IOUT – The file unit number for writing the data – usually the Listing File

ULAPRS writes a two-dimensional array using wrap format.

ULAPRWC

ULAPRWC writes a two-dimensional Real array, but this subroutine first checks to see if all values are the same. If all values are the same, then the single value is printed rather than writing all of the individual values. If all values are not the same, then ULAPRW is called to print the values.

ULASAV

Arguments:

BUF – The array to be written
TEXT – The name of the array being written
KSTP – The current time step
KPER – The current stress period
PERTIM – The accumulated length of the current stress period
TOTIM – The accumulated length of the simulation
NCOL – The number of columns in the grid
NROW – The number of rows in the grid
ILAY – The layer of the array to be written
ICHN – The file unit number for writing the data

ULASAV writes a header record and a layer of Real data to an unformatted file.

ULASV2

Arguments:

BUFF – The array to be written
TEXT – The name of the array being written
KSTP – The current time step
KPER – The current stress period
PERTIM – The accumulated length of the current stress period
TOTIM – The accumulated length of the simulation
NCOL – The number of columns in the grid
NROW – The number of rows in the grid
ILAY – The layer of the array to be written
ICHN – The file unit number for writing the data
FMTOUT – The format for writing the data
LBLSAV – Flag for writing a header
    0 – Do not write header.
    not 0 – Write header.
IBOUND – The boundary array

ULASV2 writes a layer of Real data to a formatted file. Optionally, a header record can be written. IBOUND is passed as an argument but is unused. IBOUND is included for convenience for someone to provide a more complex replacement for ULASV2 that makes use of IBOUND to control output.

ULASV3

Arguments:

IDATA – The array to be written
TEXT – The name of the array being written
KSTP – The current time step
KPER – The current stress period
PERTIM – The accumulated length of the current stress period
TOTIM – The accumulated length of the simulation
NCOL – The number of columns in the grid
NROW – The number of rows in the grid
ILAY – The layer of the array to be written
ICHN – The file unit number for writing the data
FMTOUT – The format for writing the data
LBLSAV – Flag for writing a header:
    0 – Do not write header.
    not 0 – Write header.

ULASV3 writes a layer of Integer data to a formatted file. A header record can be optionally written.

UBUDSV

Arguments:

KSTP – The current time step
KPER – The current stress period
TEXT – The name of the budget term
IBDCHN – The file unit number for writing budget data
BUFF – The array of data values to be written
NCOL – The number of columns in the grid
NROW – The number of rows in the grid

NLAY – The number of layers in the grid
IOUT – File unit number for the Listing File

UBUDSV writes a three-dimensional array of values to an unformatted file. This is used for writing budget data when COMPACT BUDGET is NOT specified in Output Control.

UBDSV1

Arguments:

KSTP – The current time step
KPER – The current stress period
TEXT – The name of the budget term
IBDCHN – The file unit number for writing budget data
BUFF – The array of data values to be written
NCOL – The number of columns in the grid
NROW – The number of rows in the grid
NLAY – The number of layers in the grid
IOUT – File unit number for the Listing File
DELT – The length of the current time step
PERTIM – The accumulated length of the current stress period
TOTIM – The accumulated length of the simulation
IBOUND – The boundary array

UBDSV1 writes a three-dimensional array of values to an unformatted file. This is used for writing a full 3–D array of budget data when COMPACT BUDGET is specified in Output Control. This is no more compact than using UBUDSV, but UBDSV1 adds an extra record containing a compact budget code, DELT, PERTIM, and TOTIM. The compact budget code is 1. IBOUND is passed as an argument but is unused. IBOUND is included for convenience for someone to provide a more complex replacement for UBDSV1 that makes use of IBOUND to control output.

UBDSV2

Arguments:

KSTP – The current time step
KPER – The current stress period
TEXT – The name of the budget term
IBDCHN – The file unit number for writing budget data
NCOL – The number of columns in the grid
NROW – The number of rows in the grid
NLAY – The number of layers in the grid
NLIST – The number of cells in the list
IOUT – File unit number for the Listing File
DELT – The length of the current time step
PERTIM – The accumulated length of the current stress period
TOTIM – The accumulated length of the simulation
IBOUND – The boundary array

UBDSV2 writes a header for a list of values to an unformatted file. This is used along with UBDSVA for writing constant-head budget data when COMPACT BUDGET is specified in Output Control. UBDSV2 writes a record containing a compact budget code, DELT, PERTIM, and TOTIM. The compact budget code is 2. UBDSV2 also writes a record containing the number of values to be written by UBDSVA. IBOUND is passed as an argument but is unused. IBOUND is included for convenience for someone to provide a more complex replacement for UBDSV2 that makes use of IBOUND to control output.

UBDSVA

Arguments:

IBDCHN – The file unit number for writing budget data
NCOL – The number of columns in the grid
NROW – The number of rows in the grid
J – The column of the cell whose budget value is being written
I – The row of the cell whose budget value is being written
K – The layer of the cell for which a budget value is being written
Q – The budget value to be written
IBOUND – The boundary array
NLAY – The number of layers in the grid

UBDSVA writes a list value, and is called once for each list value specified by the header written by UBDSV2. This is used for writing constant-head budget data when COMPACT BUDGET is specified in Output Control. A one-dimensional cell index (based on column, row, and layer hierarchy) is written along with the data value. IBOUND and NLAY are passed as arguments but are unused. IBOUND and NLAY are included for convenience for someone to provide a more complex replacement for UBDSVA that makes use of these variables.

UBDSV3

Arguments:

KSTP – The current time step
KPER – The current stress period
TEXT – The name of the budget term
IBDCHN – The file unit number for writing budget data
BUFF – The array of data values to be written
IBUFF – An array of layer numbers for the data values
NOPT – The writing option:
    Not 1 – Values in BUFF apply to layer 1, so IBUFF is not written.
    1 – Values in BUFF apply to layers in IBUFF, so IBUFF is written.
NCOL – The number of columns in the grid
NROW – The number of rows in the grid
NLAY – The number of layers in the grid
IOUT – File unit number for the Listing File
DELT – The length of the current time step
PERTIM – The accumulated length of the current stress period
TOTIM – The accumulated length of the simulation
IBOUND – The boundary array

UBDSV3 writes a two-dimensional array of values to an unformatted file and an optional two-dimensional array of layer numbers. This is used for writing Recharge and Evapotranspiration budget data when COMPACT BUDGET is specified in Output Control. UBDSV3 writes a record containing a compact budget code, DELT, PERTIM, and TOTIM. The compact budget code is 3 if no indicator array is written and 4 if an indicator array is written. IBOUND is passed as an argument but is unused. IBOUND is included for convenience for someone to provide a more complex replacement for UBDSV3 that makes use of IBOUND to control output.

UBDSV4

Arguments:

KSTP – The current time step
KPER – The current stress period
TEXT – The name of the budget term

NAUX – The number of auxiliary data values being used
AUXTXT – Names of auxiliary values
IBDCHN – The file unit number for writing budget data
NCOL – The number of columns in the grid
NROW – The number of rows in the grid
NLAY – The number of layers in the grid
NLIST – The number of cells in the list
IOUT – File unit number for the Listing File
DELT – The length of the current time step
PERTIM – The accumulated length of the current stress period
TOTIM – The accumulated length of the simulation
IBOUND – The boundary array

UBDSV4 writes a header for a list of values to an unformatted file. This is used along with UBDSVB for writing budget data for the River, General-head Boundary, Well, and Drain Packages when the COMPACT BUDGET is specified in Output Control. UBDSV4 writes a record specifying the number of data values to be written for each entry in the list and a record containing a compact budget code, DELT, PERTIM, and TOTIM. The compact budget code is 5. UBDSV4 also writes a record containing the number of entries to be written by UBDSVB. IBOUND is passed as an argument, but is unused. IBOUND is included for convenience for someone to provide a more complex replacement for UBDSV4 that makes use of IBOUND to control output.

UBDSVB

Arguments:

IBDCHN – The file unit number for writing budget data
NCOL – The number of columns in the grid
NROW – The number of rows in the grid
J – The column of the cell for which a budget value is being written
I – The row of the cell for which a budget value is being written
K – The layer of the cell for which a budget value is being written
Q – The budget value to be written
VAL – The list data for the cell for which a budget value is being written
NVL – The size of VAL
NAUX – The number of auxiliary data values being used
LAUX – The location in VAL of the first auxiliary data value
IBOUND – The boundary array
NLAY – The number of layers in the grid

UBDSVB writes a list entry, which can consist of multiple data values, and is called once for each list entry specified by the header written by UBDSV4. This is used for writing budget data combined with auxiliary data for the River, General-head Boundary, Well, and Drain Packages when COMPACT BUDGET is specified in Output Control. A one-dimensional cell index (based on column, row, and layer hierarchy) is written along with the data values. IBOUND and NLAY are passed as arguments but are unused. IBOUND and NLAY are included for convenience for someone to provide a more complex replacement for UBDSVB that makes use of these variables.

UMESPR

Arguments:

TEXT1 – First part of text message
TEXT2 – Second part of text message
IOUT – File unit number for the Listing File

UMESPR writes a message to the Listing File.

USTOP

Arguments:

STOPMESS – Text message to write prior to stopping.

USTOP optionally writes a message and stops execution. Throughout MODFLOW, USTOP is called to terminate the program when an error occurs rather than directly invoking STOP. This is done to facilitate the possibility of enhancing the program to capture control upon an error.

Parameter Subroutines

The parameter utility subroutines are used to read parameter definitions and apply the definitions to generate data used to construct the flow equation. Parameters are a user-selected alternative to direct input of data into program variables. The parameters are used to generate the same model variables into which data would otherwise be read directly; therefore, the Formulate Procedure subroutines receive data in the same form whether or not parameters are used.

The variables for storing parameter definitions are defined in Fortran module PARAMMODULE, which is documented in the Basic Package section of this chapter. Parameters have 10–character names stored in PARNAM. Instance names, which designate multiple versions of the same parameter that can be applied at different times, also have 10 characters and are stored in INAME. Parameter and instance names are case insensitive except that case is maintained for purposes of printing the names in the output file. For example, the parameter name “HyCond” would not be differentiated from the parameter name “HYCOND” because they are identical without consideration of case; however, the output file would contain the form of the name that was specified in the input data. This approach is implemented by storing the names exactly as read from the input file and temporarily converting the names to uppercase whenever the list of names is searched.

PARNAM contains all parameters, regardless of the parameter type. The location in the list establishes the parameter number, which is used extensively to reference the various variables that store parameter information. The parameter type is a four-character value stored in PARTYP using the same order as in PARNAM. Similarly, B contains parameter values.

The parameter index, IPLOC, stores four indices for each parameter that are used to define the model data values from parameters. The indices have different uses depending on whether the parameter is an array or list. Arrays are considered first. Each array parameter is specified by array clusters. A cluster defines a group of cells in one layer. Clusters are stored in IPCLST. A parameter can have any number of array clusters, so two values in IPLOC are used to specify the clusters for each parameter. IPLOC(1,p) specifies the first cluster for parameter p, and IPLOC(2,p) specifies the last cluster. List parameters are defined by specifying beginning and ending locations in the list array for the corresponding parameter type. For example, a river parameter is defined by a series of locations in the RIVR array. IPLOC(1,p) is the beginning location and IPLOC(2,p) is the ending location for parameter p.

Each array cluster has 14 values. The first value is the layer number, the second value is the multiplier array number, and the third is the zone array number. Multiplier array names are 10–character values stored in MLTNAM, and zone array names are 10–character values stored in ZONNAM. A multiplier array is a layer array that is multiplied by the parameter value when generating an array using parameters. A zone array is a layer array of integer zone numbers. Zone numbers restrict the cells of a layer to which a parameter applies. Each parameter cluster contains a list of zone numbers to which the parameter applies. The zone numbers are contained in elements 5–14 of the array cluster. The fourth value of the cluster specifies the last element in the cluster that is being used. Consider cluster n, for example. If IPCLST(4,n) is 7, then there are three zone numbers, which are IPCLST(5,n), IPCLST(6,n), and IPCLST(7,n).

IPLOC(3,p) and IPLOC(4,p) are used to specify instances. These can only apply to parameters that can vary with time. IPLOC(3,p) is the number of instances, which is 0 if instances are not being used for the parameter. IPLOC(4,p) is the element of INAME that contains the name of the first instance for the parameter.

IACTIVE contains a flag for each parameter. IACTIVE indicates whether a parameter is active in the current time step. This is used for parameters that can change with time, such as recharge parameters. An inactive parameter has a 0 value. Negative 1 indicates the parameter is active all the time; for example, LPF parameters will have -1 for IACTIVE. For active time varying parameters, IACTIVE is the number of the instance that is active.

The code for many of these subroutines is tedious because several levels of indexing are involved. First a parameter number must be determined from the name or type. The parameter index, IPLOC, must be dealt with to find where data are stored. For array parameters, the clusters must be evaluated to find the appropriate multiplier and zone arrays. Instances add another level of complexity. The expectation of the author is that these subroutines will not require frequent updating. If improvements are needed, new subroutines likely will be written.

The parameter utility subroutines use the data in PARAMMODULE as needed. All other data are passed as subroutine arguments.

UPARARRAL

Arguments:

IN – Package file unit from which parameter data will be read
IOUT – File unit number for the Listing File
LINE – Character variable containing a line of text already read from IN
NP – The number of parameters that the package will use

This subroutine determines if a package will use array parameters. When parameters were added as an option to existing MODFLOW packages, a new optional item was added to the input files. This subroutine looks for that item. Upon entry to this subroutine, a line from the package input file containing the next input item is contained in variable LINE. URWORD is called to look for the word "PARAMETER," and if found, URWORD is called again to get the number of parameters, which is variable NP. The next line is read from the input file so that LINE will contain the next input item upon returning. If "PARAMETER" is not found in LINE, NP is set to 0. A message is printed to IOUT giving the number of parameters.

UPARARRRP

Arguments:

IN – Package file unit from which parameter data will be read
IOUT – File unit number for the Listing File
NP – The parameter number of the new parameter
ILFLG – Layer flag:
    0 – Parameter defines a two-dimensional array.
    not 0 – Parameter defines a three-dimensional array.
PTYP – The parameter type of the new parameter
ITERP – The number of times the ground-water flow simulation has been run
ITVP – Time varying parameter flag:
    0 – Parameter cannot vary with time.
    not 0 – Parameter can vary with time using instances.
IACT – Value to save in IACTIVE array.

This subroutine reads data for defining one array parameter.

ITERP is an argument that specifies the number of times the ground-water flow simulation has been run. This was used in MODFLOW–2000 when doing parameter estimation. The ground-water flow simulation is repeatedly rerun with different parameter values. When ITERP is greater than 1, all the parameters will already be defined as a result of the first run of the simulation. In this case, the parameter data are read but the parameter information does not require saving and is not printed. Parameter estimation is not initially supported in MODFLOW–2005, so ITERP will always be 1.

Allowance is made for predefining parameter values as done in the MODFLOW–2000 Sensitivity file. In this case, a parameter is partly defined prior to being read by UPARARRRP. This situation is detected by finding the parameter name in PARNAM with an unspecified (blank) type. Parameter processing then continues to complete the definition. The parameter value in the Sensitivity file supersedes the value read by UPARARRRP.

UPARARRSUB1

Arguments:

ZZ – Two-dimensional array into which data are substituted
NCOL – Number of columns in ZZ
NROW – Number of rows in ZZ
ILAY – Layer number for ZZ—ILAY is 0 if ZZ is a two-dimensional array.
PTYP – Parameter type
IOUT – File unit number for the Listing File
ANAME – The name of ZZ
IPF – Print format code:
<0 – Do not print the array after substitution.
≤0 – Print the array after substitution using IPF for the format code.

This subroutine substitutes all array parameters of a specified type into a two-dimensional array. This is the functional equivalent for U2DREL. This is used by the Layer Property Flow Package. For three-dimensional arrays, UPARARRSUB1 must be called once for each layer.

UPARARRSUB2

Arguments:

ZZ – Two-dimensional array into which data are substituted
NCOL – Number of columns in ZZ
NROW – Number of rows in ZZ
ILAY – Layer number for ZZ—ILAY is 0 if ZZ is a 2–D array.
NP – The number of parameters to read
IN – Package input file unit
IOUT – File unit number for the Listing File
PTYP – Parameter type
ANAME – The name of ZZ
PACK – The package name, which is printed in error messages
IPF – Print format code:
<0 – Do not print the array after substitution.
≤0 – Print the array after substitution using IPF for the format code.

This subroutine reads a list of parameter names and substitutes them into a two-dimensional array. UPARARRSUB2 is similar to UPARARRSUB1. The difference is that UPARARRSUB2 applies only a limited set of parameters while UPARARRSUB1 applies all parameters of a specified type. Also, this subroutine must deal with possibility of parameter instances. The set of parameters to apply is read from an input file. This is used by the Recharge and Evapotranspiration Packages

USUB2D

Arguments:

ZZ – Two-dimensional array into which data are substituted
NCOL – Number of columns in ZZ
NROW – Number of rows in ZZ
IP – Parameter number of parameter to be substituted
ILAY – Layer number for ZZ—ILAY is 0 if ZZ is a two-dimensional array.
INIT – Initialization flag: 0 – Do not initialize.,not 0 – Initialize to zero before substituting.
NSUB – The number of values in ZZ that are substituted

This subroutine is called by UPARARRSUB1 and UPARARRSUB2 to substitute one parameter into a two-dimensional array. This adds to the values already in the array, therefore supporting the use of additive parameters. When substituting the first parameter of a type, the array must be initialized to 0, and the INIT argument specifies whether or not to initialize the array.

UPARLSTAL

Arguments:

IN – Package file unit from which parameter data will be read
IOUT – File unit number for the Listing File
LINE – Character variable containing a line of text already read from IN
NP – The number of parameters that the package will use
MXL – The number of cell locations for all parameters

This subroutine determines if a package will use list parameters. When parameters were added as an option to existing MODFLOW packages, a new optional item was added to the input files. This subroutine looks for that item. Upon entry to this subroutine, a line from the package input file containing the next input item is contained in variable LINE. URWORD is called to look for the word "PARAMETER," and if found, URWORD is called again to get the number of parameters, which is variable NP. In addition, URWORD is called to find MXL, the total number of list entries required to define these parameters. The next line is read from the input file so that LINE will contain the next input item upon returning. If "PARAMETER" is not found, NP and MXL are set to 0. A message is printed to IOUT giving the number of parameters.

UPARLSTRP

Arguments:

LSTSUM – Count of the number of cell locations in the list array
MXLST – The maximum number of cell locations in the list array
IN – Package file unit from which parameter data will be read
IOUT – File unit number for the Listing File
NP – The parameter number for the new parameter
PACK – The package name, which is used when printing an error message
PTYPX – The parameter type that the new parameter should have
ITERP – The number of times the ground-water flow simulation has been run
NUMINST – The number of instances that the new parameter has

This subroutine reads data for defining one list parameter.

ITERP is an argument that specifies the number of times the ground-water flow simulation has been run. This was used in MODFLOW–2000 when doing parameter estimation. The ground-water flow simulation is repeatedly rerun with different parameter values. When ITERP is greater than 1, all the parameters will already be defined as a result of the first run of the simulation. In this case, the parameter data are read but the parameter information does not require saving and is not printed. Parameter estimation is not initially supported in MODFLOW–2005, so ITERP will always be 1.

Allowance is made for predefining parameter values as done in the MODFLOW–2000 Sensitivity file. In this case, a parameter is partly defined prior to being read by UPARLSTRP. This situation is detected by finding the parameter name in PARNAM with an unspecified (blank) type. Parameter processing then continues to complete the definition. The parameter value in the Sensitivity file supersedes the value read by UPARLSTRP.

The list of cell data for the parameter is not read by this subroutine. The package Allocate and Read Procedure reads this data. Typically ULSTRD is called to read the list.

UINSRP

Arguments:

I – Instance number
IN – Package file unit from which parameter data will be read
IOUT – File unit number for the Listing File
IP – Parameter number
ITERP – The number of times the ground-water flow simulation has been run

This subroutine reads and stores one instance name. This subroutine is called by UPARARRRP and all packages that read list parameters. ITERP is used as described for UPARLSTRP to determine whether a ground-water flow simulation is being run multiple times. After the first run, instance names are read for all runs, but the names are printed only for the first run.

UPARLSTSUB

Arguments:

IN – Package file unit from which parameter data will be read
PACK – Package name
IOUTU – The absolute value is the file unit number for the Listing File. A negative sign indicates that the list should not be printed.
PTYP – Parameter type
RLIST – The package list array that stores the list of data that are active in the current stress period and parameter lists
LSTVL – The first dimension of RLIST
LSTDIM – The second dimension of RLIST
NREAD – The number of values in RLIST to be moved for each cell
MXLST – The number of cells in RLIST that can be active during a stress period
NTOT – The number of cells in RLIST that are currently active
IPVL1 – The value of the first index of RLIST where substitution begins
IPVL2 – The value of the first index of RLIST where substitution ends
LABEL – Label to print above the list
CAUX – Auxiliary field names
NCAUX – The dimension of CAUX
NAUX – The number of auxiliary values being used

This subroutine reads the name of a list parameter and substitutes the parameter values into the active part of the package list, RLIST.

PRESET

Arguments:

PTYP – Parameter type.

This subroutine sets IACTIVE to 0 for all parameters of the specified type, PTYP, which makes the parameters inactive. This is called at the beginning of a stress period to deactivate the time varying parameters used in the prior stress period.

UPARLSTLOC

Arguments:

IN – Package file unit from which parameter data will be read
PACK – Package name
IOUT – The file unit number for the Listing File
PTYP – Parameter type

IBEG – The location in the package data list of the first value associated with the parameter
IEND – The location in the package data list of the last value associated with the parameter
PV – The parameter value

This subroutine reads the name of a list parameter and finds the start and end of the list of values for the parameter within the package list. This is the same as the first part of UPARLSTSUB, but rather than copying the list into the active area for the current stress period, UPARLSTLOC simply returns the indices.

UPARARRCK

Arguments:

BUFF – Temporary layer array used to count how many parameters define a value for each cell
IBOUND – Boundary array
IOUT – The file unit number for the Listing File
LAY – The layer being checked
NCOL – The number of columns in the grid
NLAY – The number of layers in the grid
NROW – The number of rows in the grid
PTYP – Parameter type

This subroutine checks to see if all the array parameters of a specified type define a value for all cells in a layer. UPARARRCK also counts the number of parameters that contribute a value to each cell. This subroutine works by going through each cluster for each parameter of the specified type as done by UPARARRSUB1. However, rather than adding to a data value at each indicated cell, it counts the times each cell is indicated. The count is stored in the temporary array, BUFF. After scanning all the appropriate clusters, BUFF is examined to see if there are any variable-head or constant-head cells where BUFF is 0. If any are found, they are printed and the simulation is aborted.

UPARFIND

Arguments:

PNAME – Name of parameter to be found
PTYP – Parameter type for the parameter to be found
CPACK – Package name to whic
IFOUND – Parameter number of the found parameter
IOUT – The file unit number for the Listing File

This subroutine looks for a parameter in the list of parameters. If found, the parameter number is returned. The parameter type must match the specified type. If the parameter is not found or the parameter type is incorrect, the simulation is aborted.

REFERENCES

Ahlfeld, D.P., Barlow, P.M., and Mulligan, A.E., 2005, GWM—A ground-water management process for the U.S. Geological Survey modular ground-water model (MODFLOW–2000): U.S. Geological Survey Open-File Report 2005–1072, 124 p.

American National Standards Institute, 1992, American national standard for programming language—Fortran—Extended, X3.198-1992, 369 p.

Collins, R.E., 1961, Flow of fluids through porous materials: New York, Reinhold Publishing Corp., 270 p.

Crichlow, H.B., 1977, Modern reservoir engineering—A simulation approach: Englewood Cliffs, N.J., Prentice Hall Inc., 354 p.

Goode, D.J., and Appel, C.A., 1992, Finite-difference interblock transmissivity for unconfined aquifers and for aquifers having smoothly varying transmissivity: U.S. Geological Survey Water-Resources Investigations Report 92–4124, 79 p.

Halford, K.J., and Hanson, R.T., 2002, User guide for the drawdown-limited, Multi-Node Well (MNW) Package for the U.S. Geological Survey's modular three-dimensional finite-difference ground-water flow model, versions MODFLOW–96 and MODFLOW–2000: U.S. Geological Survey Open-File Report 02–293, 33 p.

Harbaugh, A.W., 1995, Direct solution package based on alternating diagonal ordering for the U.S. Geological Survey modular finite-difference ground-water flow model: U.S. Geological Survey Open-File Report 95–288, 46 p.

Harbaugh, A.W. and McDonald, M.G., 1996a, User's documentation for MODFLOW–96, an update to the U.S. Geological Survey modular finite-difference ground-water flow model: U.S. Geological Survey Open-File Report 96–485, 56 p.

Harbaugh, A.W. and McDonald, M.G., 1996b, Programmer's documentation for MODFLOW–96, an update to the U.S. Geological Survey modular finite-difference ground-water flow model: U.S. Geological Survey Open-File Report 96–486, 220 p.

Harbaugh, A.W., Banta, E.R., Hill, M.C., and McDonald, M.G., 2000, MODFLOW–2000, the U.S. Geological Survey Modular Ground-Water Model—User guide to modularization concepts and the Ground-Water Flow Process: U.S. Geological Survey Open-File Report 00–92, 121 p.

Hill, M.C., 1990, Preconditioned conjugate-gradient 2 (PCG2), a computer program for solving ground-water flow equations: U.S. Geological Survey Water-Resources Investigations Report 90–4048, 43 p.

Hill, M.C., Banta, E.R., Harbaugh, A.W., and Anderman, E.R., 2000, MODFLOW–2000, the U.S. Geological Survey Modular Ground-Water Model—User guide to the observation, sensitivity, and parameter-estimation processes and three post-processing programs: U.S. Geological Survey Open-File Report 00–184, 210 p.

Hsieh, P.A., and Freckleton, J.R., 1993, Documentation of a computer program to simulate horizontal-flow barriers using the U.S. Geological Survey's modular three-dimensional finite-difference ground-water flow model: U.S. Geological Survey Open-File Report 92–477, 32 p.

Konikow, L.F., Goode, D.J., and Hornberger, G.Z., 1996, A three-dimensional method-of-characteristics solute-transport model (MOC3D): U.S. Geological Survey Water-Resources Investigations Report 96–4267, 87 p.

Kuiper, L.K., 1987, Computer program for solving ground-water flow equations by the preconditioned conjugate gradient method: U.S. Geological Survey Water-Resources Investigations Report 87–4091, 34 p.

Leake, S.A., and Prudic, D.E., 1991, Documentation of a computer program to simulate aquifer-system compaction using the modular finite-difference ground-water flow model: U.S. Geological Survey Techniques of Water-Resources Investigations, book 6, chap. A2, 68 p.

McDonald, M.G. and Harbaugh, A.W., 1984, A modular three-dimensional finite-difference ground-water flow model: U.S. Geological Survey Open-File Report 83–875, 528 p.

McDonald, M.G. 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, chap. A1, 586 p.

McDonald, M.G., Harbaugh, A.W., Orr, B.R., and Ackerman, D.J., 1992, A method of converting no-flow cells to variable-head cells for the U.S. Geological Survey modular finite-difference ground-water flow model: U.S. Geological Survey Open-File Report 91–536, 99 p.

Mehl, Steffen and Hill, M.C., 2004, Three-dimensional local grid refinement for block-centered finite-difference groundwater models using iteratively coupled shared nodes: a new method of interpolation and analysis of errors: Advances in Water Resources, v. 27, no. 9, p. 899–912.

Peaceman, D.W., 1977, Fundamentals of numerical reservoir simulation: New York, Elsevier Scientific Publishing Company, 176 p.

Prudic, D.E., 1989, Documentation of a computer program to simulate stream-aquifer relations using a modular, finite-difference, ground-water flow model: U.S. Geological Survey Open-File Report 88–729, 113 p.

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

Remson, Irwin, Hornberger, G.M., and Molz, F.J., 1971, Numerical methods in subsurface hydrology: New York, Wiley-Interscience, 389 p.

Rushton, K.R., and Redshaw, S.C., 1979, Seepage and groundwater flow: Numerical analysis by analogue and digital methods: New York, John Wiley and Sons, 339 p.

Trescott, P.C., 1975, Documentation of finite-difference model for simulation of three-dimensional groundwater flow: U.S. Geological Survey Open-File Report 75–438, 32 p.

Trescott, P.C., and Larson, S.P., 1976, Supplement to Open-File Report 75–438, Documentation of finite-difference model for simulation of three-dimensional ground-water flow: U.S. Geological Survey Open-File Report 76–591, 21 p.

Trescott, P.C., Pinder, G.F., and Larson, S.P., 1976, Finite-difference model for aquifer simulation in two dimensions with results of numerical experiments: U.S. Geological Survey Techniques of Water-Resources Investigations, book 7, chap. C1, 116 p.

Weinstein, H.G., Stone, H.L., and Kwan, T.V., 1969, Iterative procedure for solution of systems of parabolic and elliptic equations in three dimensions: Industrial and Engineering Chemistry Fundamentals, v. 8, no. 2, p. 281–287.

APPENDIX
EXAMPLE SIMULATION

The example problem described in McDonald and Harbaugh (1988) and duplicated in Harbaugh and McDonald (1996a) and Harbaugh and others (2000) is used here to demonstrate the use of MODFLOW–2005. The example is implemented two ways—with and without parameters. For the implementation without parameters, the BCF Package is used as in McDonald and Harbaugh (1988). When the example is implemented using parameters, the LPF Package is used. The simulated system is shown in figure A–1.

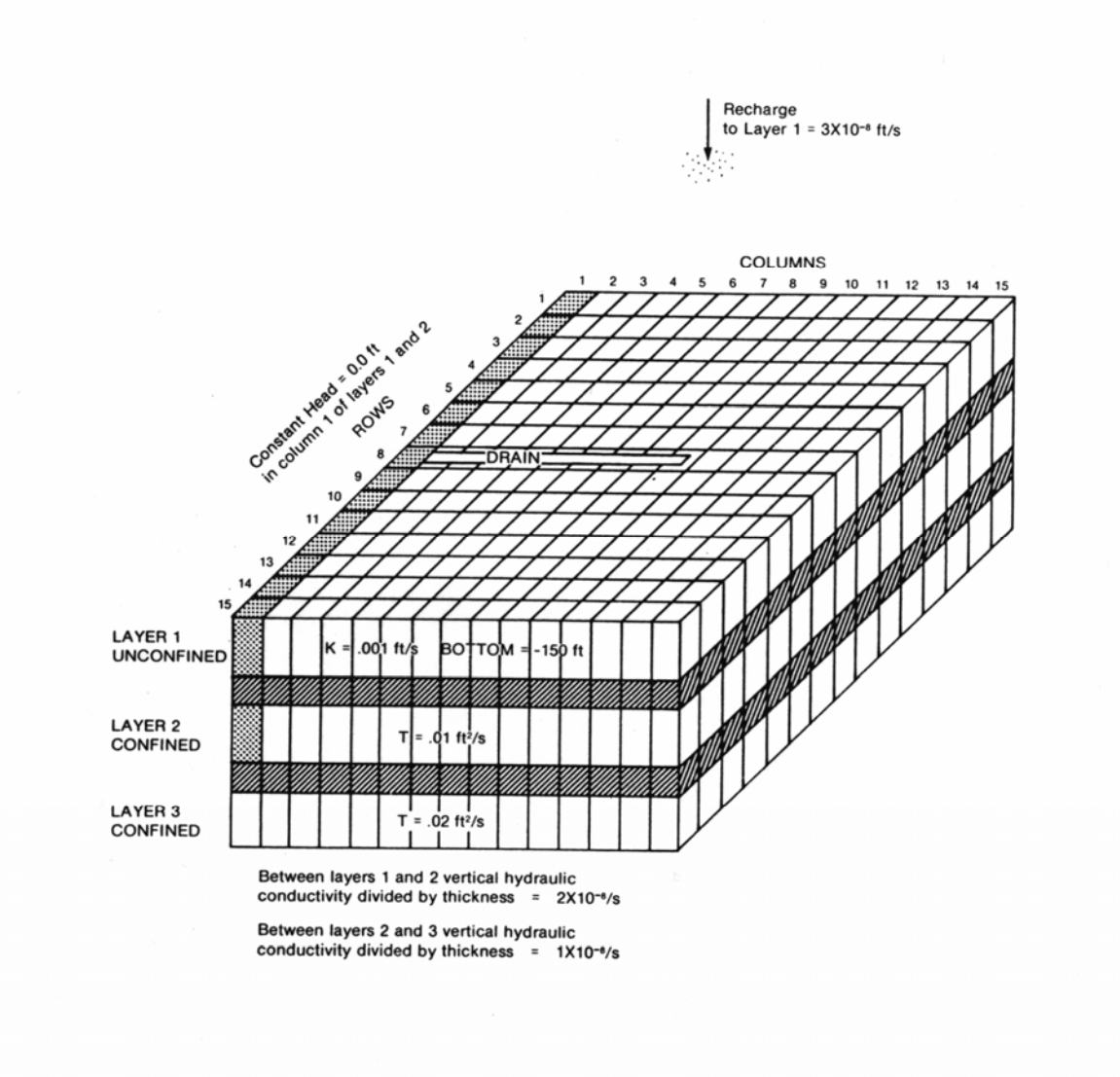


Figure A–1. Illustration of the system simulated in the example problem. (From McDonald and Harbaugh, 1988.)

Example Without Parameters

As shown in figure A–1, all of the layers are flat; the original problem definition did not specify the elevations except for the bottom of layer 1. The other elevations were not specified because they were not needed in earlier versions of MODFLOW. Layer thickness is incorporated into transmissivity and vertical hydraulic conductivity divided by thickness (VCONT) values. For MODFLOW–2005, however, all of the elevations must be specified. Any physically consistent elevations can be used if the only goal is to reproduce the heads calculated by MODFLOW–96. That is, each successive layer must have a lower elevation than the layer above, and the top elevation of layer 1 must be above the water table. Within these constraints, the following values were arbitrarily selected:

Elevation of the top of layer 1 = 200 ft,
Elevation of the bottom of the confining bed below layer 1 = -200 ft,
Elevation of the bottom of layer 2 = -300 ft,
Elevation of the bottom of the confining bed below layer 2 = -350 ft, and
Elevation of the bottom of layer 3 = -450 ft.

The following are the resulting input files:

Name File:

```
LIST  6 twri.lst
BAS6  5 twri.ba6
BCF6 11 twri.bc6
WEL  12 twri.wel
DRN  13 twri.drn
RCH  18 twri.rch
SIP  19 twri.sip
OC   22 twri.oc
DIS  10 twri.dis
```

Discretization (DIS) File

```
    3   15   15    1    1    0  NLAY,NROW,NCOL,NPER,ITMUNI,LENUNI
 1 1 0                   LAYCBD
CONSTANT   5.000000E+03  DELR
CONSTANT   5.000000E+03  DELC
CONSTANT   2.000000E+02  TOP of system
CONSTANT  -1.500000E+02  Layer BOTM layer   1
CONSTANT  -2.000000E+02  Confining bed BOTM layer   1
CONSTANT  -3.000000E+02  Layer BOTM layer   2
CONSTANT  -3.500000E+02  Confining bed BOTM layer   2
CONSTANT  -4.500000E+02  Layer BOTM layer   3
 8.640E+04         1 1.000E+00  SS  PERLEN,NSTP,TSMULT,Ss/tr
```

Basic Package (BAS6) File

```
#SAMPLE---3 LAYERS, 15 ROWS, 15 COLUMNS; STEADY STATE; CONSTANT HEADS IN
COLUMN 1
#LAYERS 1 AND 2; RECHARGE, WELLS AND DRAINS
NO OPTIONS
INTERNAL          1 (20I4)    3  IBOUND layer   1
  -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
  -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
  -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
  -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
  -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
```

```
  -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
  -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
  -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
  -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
  -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
  -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
  -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
  -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
  -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
  -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
INTERNAL          1 (20I4)    3  IBOUND layer   2
  -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
  -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
  -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
  -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
  -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
  -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
  -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
  -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
  -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
  -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
  -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
  -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
  -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
  -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
  -1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
CONSTANT          1  IBOUND layer   3
    999.99  HNOFLO
CONSTANT   0.000000E+00  Initial Head layer   1
CONSTANT   0.000000E+00  Initial Head layer   2
CONSTANT   0.000000E+00  Initial Head layer   3
```

Output Control Option (OC) File—Default output control was used when this problem was run by McDonald and Harbaugh (1988), so no file was required. The Output Control Option is used here only for the purpose of changing the format for printing head. A format has been chosen that is less than page width.

```
HEAD PRINT FORMAT 20
PERIOD 1 STEP 1
PRINT HEAD
```

Block-Centered Flow Package (BCF6) File

```
         0  1.00E+30         0  0.00E+00         0         0
 1 0 0                    Ltype
CONSTANT   1.000000E+00  TRPY
CONSTANT   1.000000E-03  HY layer   1
CONSTANT   2.000000E-08  VCONT layer   1
CONSTANT   1.000000E-02  TRAN layer   2
CONSTANT   1.000000E-08  VCONT layer   2
CONSTANT   2.000000E-02  TRAN layer   3
```

Recharge Package (RCH) File

```
      1          0    NRCHOP,IRCHBD
      1               INRECH
      0     3.E-8                                    RECH-1
```

Well Package (WEL) File

```
     15          0    MXACTW,IWELCB
     15               ITMP
      3          5        11       -5.
      2          4         6       -5.
      2          6        12       -5.
      1          9         8       -5.
      1          9        10       -5.
      1          9        12       -5.
      1          9        14       -5.
      1         11         8       -5.
      1         11        10       -5.
      1         11        12       -5.
      1         11        14       -5.
      1         13         8       -5.
      1         13        10       -5.
      1         13        12       -5.
      1         13        14       -5.
```

Drain Package (DRN) File

```
      9          0    MXACTD,IDRNCB
      9               ITMP
      1          8         2        0.     1.E00
      1          8         3        0.     1.E00
      1          8         4       10.     1.E00
      1          8         5       20.     1.E00
      1          8         6       30.     1.E00
      1          8         7       50.     1.E00
      1          8         8       70.     1.E00
      1          8         9       90.     1.E00
      1          8        10      100.     1.E00
```

Strongly Implicit Procedure Package (SIP) File

```
     50          5       MXITER,NPARM
     1.       .001          0      .001         1
```

This problem was run on a personal computer by using a runfile that was compiled with the Lahey Fortran 95 compiler. The Name File includes a file of type LIST, which is the Listing File. All of the “printed” output goes to this single file. The contents of the Listing File are shown in figure A–2.

```
                                  MODFLOW-2005
     U.S. GEOLOGICAL SURVEY MODULAR FINITE-DIFFERENCE GROUND-WATER FLOW MODEL
                             VERSION 1.00 12/15/2005

LIST FILE: twri.lst
                         UNIT    7

OPENING twri.ba6
FILE TYPE:BAS6   UNIT    8   STATUS:OLD
FORMAT:FORMATTED              ACCESS:SEQUENTIAL

OPENING twri.bc6
FILE TYPE:BCF6   UNIT   11   STATUS:OLD
FORMAT:FORMATTED              ACCESS:SEQUENTIAL

OPENING twri.wel
FILE TYPE:WEL   UNIT   12   STATUS:OLD
FORMAT:FORMATTED              ACCESS:SEQUENTIAL

OPENING twri.drn
FILE TYPE:DRN   UNIT   13   STATUS:OLD
FORMAT:FORMATTED              ACCESS:SEQUENTIAL

OPENING twri.rch
FILE TYPE:RCH   UNIT   18   STATUS:OLD
FORMAT:FORMATTED              ACCESS:SEQUENTIAL

OPENING twri.sip
FILE TYPE:SIP   UNIT   19   STATUS:OLD
FORMAT:FORMATTED              ACCESS:SEQUENTIAL

OPENING twri.oc
FILE TYPE:OC   UNIT   22   STATUS:OLD
FORMAT:FORMATTED              ACCESS:SEQUENTIAL

OPENING twri.dis
FILE TYPE:DIS   UNIT   10   STATUS:OLD
FORMAT:FORMATTED              ACCESS:SEQUENTIAL

BAS -- BASIC PACKAGE, VERSION 7, 5/2/2005 INPUT READ FROM UNIT    8

DISCRETIZATION INPUT DATA READ FROM UNIT   10
   3 LAYERS        15 ROWS        15 COLUMNS
   1 STRESS PERIOD(S) IN SIMULATION
MODEL TIME UNIT IS SECONDS
MODEL LENGTH UNIT IS UNDEFINED
 Confining bed flag for each layer:
  1   1   0

                    DELR =   5000.00

                    DELC =   5000.00
```

Figure A–2. Listing File for example problem without parameters.

```
TOP ELEVATION OF LAYER 1 =   200.000
  MODEL LAYER BOTTOM EL. =  -150.000     FOR LAYER   1

BOT. EL. OF QUASI-3D BED =  -200.000     FOR LAYER   1

  MODEL LAYER BOTTOM EL. =  -300.000     FOR LAYER   2

BOT. EL. OF QUASI-3D BED =  -350.000     FOR LAYER   2

  MODEL LAYER BOTTOM EL. =  -450.000     FOR LAYER   3


STRESS PERIOD     LENGTH       TIME STEPS     MULTIPLIER FOR DELT    SS FLAG
-----------------------------------------------------------------------------
       1         86400.00          1                    1.000         SS

STEADY-STATE SIMULATION



#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
READING ON UNIT    8 WITH FORMAT: (20I4)

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

Figure A–2. Listing File for example problem without parameters.—Continued

```
                    BOUNDARY ARRAY FOR LAYER   2
READING ON UNIT    8 WITH FORMAT: (20I4)

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

          BOUNDARY ARRAY =              1 FOR LAYER   3

AQUIFER HEAD WILL BE SET TO  999.99     AT ALL NO-FLOW NODES (IBOUND=0).

            INITIAL HEAD =   0.00000     FOR LAYER   1

            INITIAL HEAD =   0.00000     FOR LAYER   2

            INITIAL HEAD =   0.00000     FOR LAYER   3

OUTPUT CONTROL IS SPECIFIED ONLY AT TIME STEPS FOR WHICH OUTPUT IS DESIRED
HEAD PRINT FORMAT CODE IS  20    DRAWDOWN PRINT FORMAT CODE IS   0
HEADS WILL BE SAVED ON UNIT    0    DRAWDOWNS WILL BE SAVED ON UNIT    0

BCF -- BLOCK-CENTERED FLOW PACKAGE, VERSION 7, 5/2/2005
        INPUT READ FROM UNIT 11
STEADY-STATE SIMULATION
HEAD AT CELLS THAT CONVERT TO DRY=  0.10000E+31
WETTING CAPABILITY IS NOT ACTIVE
     LAYER  LAYER-TYPE CODE     INTERBLOCK T
     ------------------------------------------
        1            1          0 -- HARMONIC
        2            0          0 -- HARMONIC
        3            0          0 -- HARMONIC
```

Figure A–2. Listing File for example problem without parameters.—Continued

```
COLUMN TO ROW ANISOTROPY =  1.00000

    HYD. COND. ALONG ROWS =  1.000000E-03 FOR LAYER   1

 VERT HYD COND /THICKNESS =  2.000000E-08 FOR LAYER   1

     TRANSMIS. ALONG ROWS =  1.000000E-02 FOR LAYER   2

 VERT HYD COND /THICKNESS =  1.000000E-08 FOR LAYER   2

     TRANSMIS. ALONG ROWS =  2.000000E-02 FOR LAYER   3

 WEL -- WELL PACKAGE, VERSION 7, 5/2/2005 INPUT READ FROM UNIT   12
 No named parameters
 MAXIMUM OF     15 ACTIVE WELLS AT ONE TIME


     0 Well parameters

 DRN -- DRAIN PACKAGE, VERSION 7, 5/2/2005 INPUT READ FROM UNIT   13
 No named parameters
 MAXIMUM OF      9 ACTIVE DRAINS AT ONE TIME


     0 Drain parameters

 RCH -- RECHARGE PACKAGE, VERSION 7, 5/2/2005 INPUT READ FROM UNIT   18
 No named parameters
 OPTION 1 -- RECHARGE TO TOP LAYER


     0 Recharge parameters

 SIP -- STRONGLY-IMPLICIT PROCEDURE SOLUTION PACKAGE
                   VERSION 7, 5/2/2005 INPUT READ FROM UNIT   19
 MAXIMUM OF  50 ITERATIONS ALLOWED FOR CLOSURE
  5 ITERATION PARAMETERS

          SOLUTION BY THE STRONGLY IMPLICIT PROCEDURE
          -------------------------------------------
 MAXIMUM ITERATIONS ALLOWED FOR CLOSURE =       50
                ACCELERATION PARAMETER =     1.0000
      HEAD CHANGE CRITERION FOR CLOSURE =    0.10000E-02
      SIP HEAD CHANGE PRINTOUT INTERVAL =        1

     5 ITERATION PARAMETERS CALCULATED FROM SPECIFIED WSEED = 0.00100000 :

  0.000000E+00 0.822172E+00 0.968377E+00 0.994377E+00 0.999000E+00
1
```

Figure A–2. Listing File for example problem without parameters.—Continued

```
                          STRESS PERIOD NO.    1, LENGTH =   86400.00
                          ------------------------------------------------

                            NUMBER OF TIME STEPS =     1

                             MULTIPLIER FOR DELT =     1.000

                          INITIAL TIME STEP SIZE =   86400.00

 WELL NO.  LAYER   ROW   COL   STRESS RATE
 -------------------------------------------------
      1      3      5     11      -5.000
      2      2      4      6      -5.000
      3      2      6     12      -5.000
      4      1      9      8      -5.000
      5      1      9     10      -5.000
      6      1      9     12      -5.000
      7      1      9     14      -5.000
      8      1     11      8      -5.000
      9      1     11     10      -5.000
     10      1     11     12      -5.000
     11      1     11     14      -5.000
     12      1     13      8      -5.000
     13      1     13     10      -5.000
     14      1     13     12      -5.000
     15      1     13     14      -5.000

     15 WELLS

 DRAIN NO.  LAYER   ROW   COL     DRAIN EL.  CONDUCTANCE
 ------------------------------------------------------------
      1      1      8      2       0.000          1.000
      2      1      8      3       0.000          1.000
      3      1      8      4       10.00          1.000
      4      1      8      5       20.00          1.000
      5      1      8      6       30.00          1.000
      6      1      8      7       50.00          1.000
      7      1      8      8       70.00          1.000
      8      1      8      9       90.00          1.000
      9      1      8     10       100.0          1.000

      9 DRAINS

                 RECHARGE =  3.000000E-08

 SOLVING FOR HEAD

    31 ITERATIONS FOR TIME STEP   1 IN STRESS PERIOD    1
```

Figure A–2. Listing File for example problem without parameters.—Continued

```
 MAXIMUM HEAD CHANGE FOR EACH ITERATION:

    HEAD CHANGE   HEAD CHANGE   HEAD CHANGE   HEAD CHANGE   HEAD CHANGE
  LAYER,ROW,COL LAYER,ROW,COL LAYER,ROW,COL LAYER,ROW,COL LAYER,ROW,COL
 ---------------------------------------------------------------------
     -22.41         12.48         13.39         48.21         35.91
  (  3,  5, 11) (  1,  1, 15) (  3,  1, 14) (  1,  1, 15) (  3,  1, 13)
      2.482         1.430         6.214         7.411         13.66
  (  1,  9, 14) (  3, 10, 13) (  1, 12, 14) (  3, 11, 14) (  1, 15, 15)
     0.5503        0.4821        0.4711         2.019         2.302
  (  3,  8,  7) (  2,  6,  9) (  3,  5, 10) (  1, 11, 14) (  3,  5, 13)
     0.1108        0.7059E-01    0.2819        0.3141        0.3320
  (  1, 13, 12) (  3, 12, 11) (  1, 14, 14) (  3, 13, 14) (  1, 15, 15)
     0.7853E-02    0.1586E-01    0.1777E-01    0.7910E-01    0.8500E-01
  (  1, 13, 12) (  2, 11, 11) (  3, 11, 10) (  1, 14, 14) (  3,  7, 14)
     0.4169E-02    0.2555E-02    0.9769E-02    0.1082E-01    0.1030E-01
  (  1, 13, 14) (  3, 14, 15) (  1, 14, 14) (  3, 13, 14) (  1, 15, 15)
     0.2430E-03
  (  1, 13, 12)


 OUTPUT CONTROL FOR STRESS PERIOD    1   TIME STEP   1
    PRINT HEAD FOR ALL LAYERS
1
              HEAD IN LAYER   1 AT END OF TIME STEP   1 IN STRESS PERIOD    1
  -------------------------------------------------------------------------

            1           2           3           4           5           6
            7           8           9          10          11          12
           13          14          15
 .....................................................................
   1    0.000       24.94       44.01       59.26       71.82       82.52
        91.91       100.0       106.9       112.6       117.4       121.3
        124.3       126.4       127.4
   2    0.000       24.45       43.10       57.98       70.17       80.57
        90.12       98.40       105.3       111.0       115.7       119.6
        122.7       124.9       126.1
   3    0.000       23.45       41.30       55.43       66.78       76.21
        86.51       95.20       102.2       107.6       112.0       116.1
        119.6       122.1       123.4
   4    0.000       21.92       38.61       51.75       61.79       68.03
        81.34       90.75       97.64       102.5       106.1       110.7
        114.9       117.9       119.4
   5    0.000       19.73       34.92       47.32       57.69       66.74
        77.09       85.76       92.22       96.15       97.29       103.1
        108.8       112.5       114.3
   6    0.000       16.51       29.50       40.90       51.30       61.21
        71.19       79.85       86.47       90.82       93.03       94.23
        102.1       106.4       108.4
   7    0.000       11.55       21.10       31.21       41.40       51.84
        63.08       72.68       79.95       84.92       88.60       91.66
        96.43       99.82       101.8
```

Figure A–2. Listing File for example problem without parameters.—Continued

```
   8    0.000       3.483       6.832       16.25       26.30       36.97
        52.59       64.31       72.52       77.25       81.99       85.00
        89.27       91.72       94.33
   9    0.000       10.54       19.11       28.12       36.92       45.27
        52.95       55.38       65.15       66.07       73.93       73.79
        80.84       80.17       86.49
  10    0.000       14.62       25.86       35.38       43.49       50.11
        54.93       57.55       62.95       65.55       70.39       72.44
        76.72       78.26       81.79
  11    0.000       17.11       29.96       40.01       47.78       53.24
        55.81       53.33       60.27       59.29       66.43       65.45
        72.22       71.04       77.62
  12    0.000       18.68       32.56       43.07       50.81       55.92
        58.33       58.47       61.93       63.18       67.12       68.50
        72.29       73.46       76.85
  13    0.000       19.67       34.24       45.14       53.01       58.04
        59.91       56.75       62.59       60.91       67.22       65.75
        71.90       70.35       76.48
  14    0.000       20.27       35.27       46.48       54.61       60.08
        63.17       64.52       67.25       68.79       71.64       73.18
        75.84       77.03       79.09
  15    0.000       20.56       35.78       47.16       55.48       61.26
        65.02       67.52       69.94       72.01       74.29       76.22
        78.22       79.66       80.82
1
             HEAD IN LAYER   2 AT END OF TIME STEP   1 IN STRESS PERIOD    1
  ----------------------------------------------------------------------------

           1           2           3           4           5           6
           7           8           9          10          11          12
          13          14          15
 ...........................................................................
   1    0.000       24.66       43.73       59.02       71.61       82.32
        91.72       99.86       106.7       112.5       117.2       121.1
        124.1       126.2       127.3
   2    0.000       24.17       42.83       57.74       69.95       80.36
        89.93       98.22       105.1       110.8       115.5       119.4
        122.6       124.8       125.9
   3    0.000       23.17       41.03       55.19       66.53       75.77
        86.29       95.02       102.0       107.4       111.8       116.0
        119.5       121.9       123.2
   4    0.000       21.65       38.34       51.50       61.35       60.17
        80.90       90.55       97.45       102.3       105.4       110.4
        114.8       117.7       119.2
   5    0.000       19.48       34.65       47.07       57.44       66.30
        76.85       85.57       92.00       95.41       91.09       102.1
        108.6       112.4       114.2
   6    0.000       16.27       29.24       40.65       51.07       60.98
        70.98       79.65       86.28       90.54       92.06       86.23
        101.7       106.2       108.3
   7    0.000       11.38       20.95       31.05       41.25       51.70
        62.90       72.48       79.76       84.73       88.35       91.24
        96.22       99.65       101.6
```

Figure A–2. Listing File for example problem without parameters.—Continued

```
  8     0.000       4.209       8.330       17.58       27.58       38.25
        52.94       64.19       72.34       77.12       81.81       84.86
        89.10       91.59       94.17
   9    0.000       10.38       18.96       27.98       36.79       45.16
        52.86       56.13       65.08       66.79       73.87       74.48
        80.77       80.84       86.38
  10    0.000       14.40       25.61       35.15       43.27       49.91
        54.76       57.48       62.79       65.49       70.24       72.37
        76.57       78.20       81.64
  11    0.000       16.87       29.70       39.78       47.56       53.05
        55.68       54.09       60.20       60.04       66.37       66.18
        72.16       71.75       77.51
  12    0.000       18.43       32.31       42.85       50.60       55.73
        58.16       58.41       61.78       63.12       66.98       68.44
        72.15       73.40       76.69
  13    0.000       19.42       33.98       44.91       52.80       57.85
        59.78       57.50       62.53       61.65       67.16       66.48
        71.84       71.06       76.37
  14    0.000       20.02       35.02       46.26       54.41       59.88
        62.99       64.39       67.08       68.66       71.48       73.06
        75.68       76.91       78.93
  15    0.000       20.30       35.52       46.94       55.28       61.07
        64.84       67.34       69.76       71.84       74.11       76.04
        78.04       79.49       80.65
1
             HEAD IN LAYER   3 AT END OF TIME STEP   1 IN STRESS PERIOD    1
  ----------------------------------------------------------------------------

           1           2           3           4           5           6
           7           8           9          10          11          12
          13          14          15
 ...........................................................................
   1   1.800       24.34       43.36       58.70       71.33       82.06
       91.48       99.63       106.5       112.3       117.0       120.9
       123.9       126.0       127.1
   2   1.764       23.85       42.46       57.42       69.66       80.07
       89.68       97.99       104.9       110.6       115.3       119.2
       122.4       124.6       125.7
   3   1.691       22.86       40.67       54.87       66.20       75.28
       85.98       94.77       101.7       107.2       111.5       115.7
       119.3       121.7       123.0
   4   1.578       21.35       37.98       51.17       60.85       62.69
       80.41       90.28       97.19       101.9       104.1       110.0
       114.5       117.5       119.0
   5   1.415       19.18       34.30       46.75       57.10       65.80
       76.54       85.30       91.67       94.17       77.46       100.7
       108.2       112.1       114.0
   6   1.176       15.99       28.91       40.33       50.76       60.67
       70.70       79.38       86.01       90.12       90.60       88.55
       101.2       106.0       108.0
   7  0.8273       11.21       20.79       30.88       41.09       51.55
       62.67       72.22       79.50       84.46       87.98       90.77
       95.94       99.41       101.4
```

Figure A–2. Listing File for example problem without parameters.—Continued

```
   8   0.4331       5.131       10.19       19.27       29.19       39.84
        53.40       64.07       72.11       76.95       81.58       84.68
        88.88       91.44       93.95
   9   0.7543       10.22       18.82       27.84       36.66       45.06
        52.78       57.03       65.02       67.64       73.81       75.31
        80.72       81.64       86.24
  10    1.039       14.13       25.29       34.85       42.99       49.65
        54.54       57.44       62.61       65.44       70.05       72.33
        76.39       78.15       81.43
  11    1.224       16.59       29.37       39.47       47.28       52.79
        55.53       55.01       60.16       60.94       66.33       67.06
        72.13       72.60       77.38
  12    1.341       18.15       31.97       42.54       50.32       55.47
        57.94       58.37       61.60       63.08       66.80       68.41
        71.97       73.36       76.49
  13    1.415       19.14       33.65       44.61       52.53       57.60
        59.63       58.39       62.48       62.54       67.12       67.35
        71.80       71.90       76.24
  14    1.460       19.73       34.68       45.96       54.13       59.63
        62.76       64.24       66.87       68.52       71.27       72.91
        75.47       76.77       78.71
  15    1.481       20.01       35.18       46.63       55.00       60.81
        64.59       67.11       69.52       71.61       73.87       75.82
        77.81       79.27       80.42
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:                                IN:
          ---                                ---
            STORAGE =           0.0000            STORAGE =           0.0000
      CONSTANT HEAD =           0.0000      CONSTANT HEAD =           0.0000
              WELLS =           0.0000              WELLS =           0.0000
             DRAINS =           0.0000             DRAINS =           0.0000
           RECHARGE =    13608000.0000           RECHARGE =         157.5000

           TOTAL IN =    13608000.0000           TOTAL IN =         157.5000

         OUT:                               OUT:
         ----                               ----
            STORAGE =           0.0000            STORAGE =           0.0000
      CONSTANT HEAD =     4326522.0000      CONSTANT HEAD =          50.0755
              WELLS =     6480000.0000              WELLS =          75.0000
             DRAINS =     2801081.2500             DRAINS =          32.4199
           RECHARGE =           0.0000           RECHARGE =           0.0000

          TOTAL OUT =    13607603.0000           TOTAL OUT =         157.4954
           IN - OUT =         397.0000            IN - OUT =       4.5929E-03

 PERCENT DISCREPANCY =           0.00     PERCENT DISCREPANCY =           0.00
```

Figure A–2. Listing File for example problem without parameters.—Continued

```
          TIME SUMMARY AT END OF TIME STEP   1 IN STRESS PERIOD    1
                     SECONDS      MINUTES      HOURS       DAYS        YEARS
                     -----------------------------------------------------------
   TIME STEP LENGTH  86400.       1440.0       24.000      1.0000     2.73785E-03
 STRESS PERIOD TIME  86400.       1440.0       24.000      1.0000     2.73785E-03
         TOTAL TIME  86400.       1440.0       24.000      1.0000     2.73785E-03
1

 Run end date and time (yyyy/mm/dd hh:mm:ss): 2005/11/25  9:53:57
 Elapsed run time:  0.328 Seconds
```

Figure A–2. Listing File for example problem without parameters.

Example With Parameters

The same problem is run using the LPF Package. LPF does not have an option to enter transmissivity for confined layers; instead, horizontal hydraulic conductivity (HK) must be entered for all layers. HK values for layers 2 and 3 were not directly specified in the problem, but HK can be easily calculated from the specified values of transmissivity and the assumed layer thicknesses.

LPF also differs from BCF in the way that vertical hydraulic information is entered. LPF requires vertical hydraulic conductivity for each model layer and Quasi–3D confining bed, whereas BCF requires only the vertical leakance (VCONT) between nodes, which is the vertical hydraulic conductivity (including a Quasi–3D confining bed, if it exists) divided by the distance between nodes. In this problem, there are Quasi–3D confining layers separating the model layers, and VCONT would be calculated using equation 5–40. For example, for a cell in layer 1, VCONT would be calculated from the VCONT of half the thickness of layer 1, the full thickness of the confining bed, and half the thickness of layer 2. The problem, however, directly defines VCONT values rather than providing the component parts for computing VCONT, and the vertical hydraulic conductivity of each layer and the confining beds cannot be uniquely calculated from the BCF VCONT values alone. The approach used here is to assume that the confining beds dominate the VCONT calculation and that virtually no contribution comes from the actual model layers, which is consistent with the assumptions used in making the Quasi–3D approximation. If the model layers have no influence on VCONT, then the vertical hydraulic conductivity of the confining bed (VKCB) is VCONT times the confining-bed thickness. For layer 1, VKCB is 10^{-6} ft/s; for layer 2, $5x10^{-7}$ ft/s.

Although the assumption is that for the above calculation of VKCB there is no contribution to vertical conductance (CV) from the aquifer vertical hydraulic conductivity (VK), LPF requires VK to be specified as part of the input data. According to equation 5–26, VK should be infinite in order to have no impact on CV. There is no way to directly specify an infinite value for VK, but the effect of having an infinite value can be obtained by making the VK values large compared to VKCB. For this example problem, VK is set to 1.0 feet per second (ft/s) for all model layers, which is a million times larger than the largest VKCB. The low impact of this value on CV is demonstrated by the fact that the calculated head as shown in the Listing File when LPF is used (fig. A–3) is the same as the head in the BCF Listing File (fig. A–2).

Note that any combination of VK and CBVK that produces virtually the same CV that BCF calculates using the original VCONT values will produce the same heads in this problem. If the values of VK for layer 1 were relatively small compared to CBVK, then the VK values would have to be varied from cell to cell because the saturated thickness of layer 1 varies; that is, as indicated by equation 5–26, the conductance between cells depends on saturated thickness of the model cells. Thus, the choice to use a large VK in order to essentially eliminate the aquifer contribution to CV makes the LPF input simpler than it would be if a small VK were used.

Another difference in the input data for this problem is that parameters are used for defining some of the data in order to illustrate the use of parameters. Parameters are used for defining HK, VKCB, RECH, well recharge and drain conductance input variables. In the LPF Package, five parameters are used—three define HK, and two define VKCB. Each of these parameters is for a single layer, and each parameter has a uniform value for the entire layer. HK1, HK2, and HK3 are HK for layers 1, 2, and 3, respectively. These use “NONE” for the multiplier array, which means that a multiplier array is not used. Instead, the parameter value applies directly to all of the included cells. The

zone array is “ALL,” which means the values for all cells in the indicated layer are defined by the parameter. Parameter VKCB1 defines VKCB values for all of layer 1, and VKCB2 defines VKCB values for layer 2. Both of these parameters use the same multiplier array, which has a value of 1.0E-6 at all cells. The parameter value for VKCB1 is 1.0, and the value for VKCB2 is 0.5, which produces the required values of 1.0E-6 for layer 1 and 5.0E-7 for layer 2.

For the RCH Package, two parameters, RCH1 and RCH2, are used. RCH1 includes the left seven columns of the grid, and RCH2 includes the right 8 columns. These areas are specified using zone array RCHZONES, which contains “1” in columns 1–7, and “2” in columns 8–15. Accordingly, the zone value of 1 is specified in the definition of RCH1, and zone value 2 is specified for RCH2. The parameter values for RCH1 and RCH2 are both 3.0E-8, which results in all cells having the same recharge rate. Thus, having two parameters is not advantageous in this example; however, two zones were used to illustrate how this is done. The value of having two zones would become apparent if there were a need to change the recharge rate in the area of zone 1 without changing the zone 2 values. This could be done simply by changing the value of parameter RCH1.

The Well Package has one parameter, WELL1, and the Drain Package has one parameter, DRN1. Both of these parameters have a value of 1.0, so the multipliers in the lists for both parameters are simply the same as the data values that are used to define the input data without using parameters. Using parameters could be advantageous, however, if there were a need to change all of the input values associated with the parameters. All of the values could be changed by a constant factor simply by changing the parameter values. The parameters define only part of the wells and drains. Some of them are defined as separate nonparameter lists.

The input files that are different as a result of using LPF instead of BCF and adding parameters are shown below. The DIS, BAS, SIP, and OC files are not shown again because they are unchanged.

Name File

```
LIST   6 twri.lst
BAS6   5 twri.ba6
LPF   11 twri.lpf
WEL   12 twri.wel
DRN   13 twri.drn
RCH   18 twri.rch
SIP   19 twri.sip
OC    22 twri.oc
MULT   8 twri.mlt
ZONE   9 twri.zon
DIS   10 twri.dis
```

Multiplier File

```
1
MULT1
CONSTANT 1.0E-6
```

Figure A–2. Listing File for example problem without parameters.—Continued

Zone File

```
1
RCHZONES
INTERNAL 1 (15I2)  1
 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
```

Layer-Property Flow Package (LPF) File

```
#  LPF Package input data for example problem
0  1.00E+30    5 ILPFCB,HDRY,NPLPF
 1 0 0           LAYTYP
 0 0 0           LAYAVG
 1. 1. 1.        CHANI
 0 0 0           LAYVKA
 0 0 0           LAYWET
HK1 HK  1.0E-3 1
1 NONE ALL
HK2 HK  1.0E-4 1
2 NONE ALL
HK3 HK  2.0E-4 1
3 NONE ALL
VKCB1 VKCB 1.0 1
1 MULT1 ALL
VKCB2 VKCB 0.5 1
2 MULT1 ALL
0                        HK layer   1
CONSTANT   1.0           VK layer   1
0                        VKCB layer 1
0                        HK layer   2
CONSTANT   1.0           VK layer   2
0                        VKCB layer 2
0                        HK  ayer   3
CONSTANT   1.0           VK layer   3
```

Figure A–2. Listing File for example problem without parameters.—Continued

Recharge Package (RCH) File

```
PARAMETER  2
         1          0    NRCHOP,IRCHBD
RCH1  RCH  3.0E-8  1
NONE  RCHZONES  1
RCH2  RCH  3.0E-8  1
NONE  RCHZONES  2
         2              INRECH
RCH1
RCH2
```

Well Package (WEL) File

```
PARAMETER 1 12
        15          0    MXACTW,IWELCB
WELL1 Q  1.0 12
         1          9          8       -5.
         1          9         10       -5.
         1          9         12       -5.
         1          9         14       -5.
         1         11          8       -5.
         1         11         10       -5.
         1         11         12       -5.
         1         11         14       -5.
         1         13          8       -5.
         1         13         10       -5.
         1         13         12       -5.
         1         13         14       -5.
         3          1               ITMP,NP
         3          5         11       -5.
         2          4          6       -5.
         2          6         12       -5.
WELL1
```

Drain Package (DRN) File

```
PARAMETER  1  2
         9          0    MXACTD,IDRNCB
DRN1  DRN  1.0   2
         1          8          2        0.     1.E00
         1          8          3        0.     1.E00
         7          1     ITMP,NP
         1          8          4       10.     1.E00
         1          8          5       20.     1.E00
```

Figure A–2. Listing File for example problem without parameters.—Continued

```
          1          8          6        30.      1.E00
          1          8          7        50.      1.E00
          1          8          8        70.      1.E00
          1          8          9        90.      1.E00
          1          8         10       100.      1.E00
DRN1
```

The Listing File is shown in figure A–3.

```
                                 MODFLOW-2005
     U.S. GEOLOGICAL SURVEY MODULAR FINITE-DIFFERENCE GROUND-WATER FLOW MODEL
                           VERSION 1.00 12/15/2005

 LIST FILE: twrip.lst
                        UNIT    2

 OPENING twrip.ba6
 FILE TYPE:BAS6   UNIT    3   STATUS:OLD
 FORMAT:FORMATTED             ACCESS:SEQUENTIAL

 OPENING twrip.lpf
 FILE TYPE:LPF   UNIT   11   STATUS:OLD
 FORMAT:FORMATTED             ACCESS:SEQUENTIAL

 OPENING twrip.wel
 FILE TYPE:WEL   UNIT   12   STATUS:OLD
 FORMAT:FORMATTED             ACCESS:SEQUENTIAL

 OPENING twrip.drn
 FILE TYPE:DRN   UNIT   13   STATUS:OLD
 FORMAT:FORMATTED             ACCESS:SEQUENTIAL

 OPENING twrip.rch
 FILE TYPE:RCH   UNIT   18   STATUS:OLD
 FORMAT:FORMATTED             ACCESS:SEQUENTIAL

 OPENING twrip.sip
 FILE TYPE:SIP   UNIT   19   STATUS:OLD
 FORMAT:FORMATTED             ACCESS:SEQUENTIAL

 OPENING twrip.oc
 FILE TYPE:OC   UNIT   22   STATUS:OLD
 FORMAT:FORMATTED             ACCESS:SEQUENTIAL

 OPENING twrip.mlt
 FILE TYPE:MULT   UNIT    8   STATUS:OLD
 FORMAT:FORMATTED             ACCESS:SEQUENTIAL
```

Figure A–3. Listing File for example problem with parameters.

```
OPENING twrip.zon
 FILE TYPE:ZONE   UNIT    9   STATUS:OLD
 FORMAT:FORMATTED              ACCESS:SEQUENTIAL

 OPENING twrip.dis
 FILE TYPE:DIS   UNIT   10   STATUS:OLD
 FORMAT:FORMATTED              ACCESS:SEQUENTIAL

 BAS -- BASIC PACKAGE, VERSION 7, 5/2/2005 INPUT READ FROM UNIT    3

 DISCRETIZATION INPUT DATA READ FROM UNIT   10
    3 LAYERS        15 ROWS        15 COLUMNS
    1 STRESS PERIOD(S) IN SIMULATION
 MODEL TIME UNIT IS SECONDS
 MODEL LENGTH UNIT IS UNDEFINED
  Confining bed flag for each layer:
   1   1   0

                    DELR =   5000.00

                    DELC =   5000.00

 TOP ELEVATION OF LAYER 1 =   200.000

   MODEL LAYER BOTTOM EL. =  -150.000     FOR LAYER   1

 BOT. EL. OF QUASI-3D BED =  -200.000     FOR LAYER   1

   MODEL LAYER BOTTOM EL. =  -300.000     FOR LAYER   2

 BOT. EL. OF QUASI-3D BED =  -350.000     FOR LAYER   2

   MODEL LAYER BOTTOM EL. =  -450.000     FOR LAYER   3


 STRESS PERIOD     LENGTH       TIME STEPS     MULTIPLIER FOR DELT    SS FLAG
 ------------------------------------------------------------------------------
        1         86400.00          1                    1.000         SS

 STEADY-STATE SIMULATION



 #SAMPLE----3 LAYERS, 15 ROWS, 15 COLUMNS; STEADY STATE; CONSTANT HEADS COLUMN 1
 #LAYERS 1 AND 2; RECHARGE, WELLS AND DRAINS
```

Figure A-3. Listing File for example problem with parameters.—Continued

```
                    BOUNDARY ARRAY FOR LAYER   1
 READING ON UNIT    3 WITH FORMAT: (20I4)

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



                    BOUNDARY ARRAY FOR LAYER   2
 READING ON UNIT    3 WITH FORMAT: (20I4)

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

          BOUNDARY ARRAY =              1 FOR LAYER   3

 AQUIFER HEAD WILL BE SET TO  999.99     AT ALL NO-FLOW NODES (IBOUND=0).

             INITIAL HEAD =   0.00000     FOR LAYER   1

             INITIAL HEAD =   0.00000     FOR LAYER   2

             INITIAL HEAD =   0.00000     FOR LAYER   3
```

Figure A–3. Listing File for example problem with parameters.—Continued

```
OUTPUT CONTROL IS SPECIFIED ONLY AT TIME STEPS FOR WHICH OUTPUT IS DESIRED
 HEAD PRINT FORMAT CODE IS  20    DRAWDOWN PRINT FORMAT CODE IS   0
 HEADS WILL BE SAVED ON UNIT    0    DRAWDOWNS WILL BE SAVED ON UNIT    0

 ZONE OPTION, INPUT READ FROM UNIT    9
     1 ZONE ARRAYS

 MULTIPLIER OPTION, INPUT READ FROM UNIT    8
   1 MULTIPLIER ARRAYS

  MULT. ARRAY: MULT1      =  1.000000E-06



            ZONE ARRAY: RCHZONES
 READING ON UNIT    9 WITH FORMAT: (15I2)

      1 2 3 4 5 6 7 8 9101112131415
 .................................
   1  1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
   2  1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
   3  1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
   4  1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
   5  1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
   6  1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
   7  1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
   8  1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
   9  1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
  10  1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
  11  1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
  12  1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
  13  1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
  14  1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
  15  1 1 1 1 1 1 1 2 2 2 2 2 2 2 2

 LPF -- LAYER-PROPERTY FLOW PACKAGE, VERSION 7, 5/2/2005
         INPUT READ FROM UNIT   11
 #  LPF Package input data for example problem
 HEAD AT CELLS THAT CONVERT TO DRY=  1.00000E+30
     5 Named Parameters

   LAYER FLAGS:
 LAYER       LAYTYP        LAYAVG         CHANI        LAYVKA        LAYWET
 ------------------------------------------------------------------------------
    1             1             0     1.000E+00             0             0
    2             0             0     1.000E+00             0             0
    3             0             0     1.000E+00             0             0
```

Figure A–3. Listing File for example problem with parameters.—Continued

```
  INTERPRETATION OF LAYER FLAGS:
                        INTERBLOCK     HORIZONTAL    DATA IN
       LAYER TYPE   TRANSMISSIVITY   ANISOTROPY   ARRAY VKA   WETTABILITY
LAYER      (LAYTYP)       (LAYAVG)       (CHANI)      (LAYVKA)      (LAYWET)
--------------------------------------------------------------------------
   1   CONVERTIBLE      HARMONIC     1.000E+00    VERTICAL K  NON-WETTABLE
   2      CONFINED      HARMONIC     1.000E+00    VERTICAL K  NON-WETTABLE
   3      CONFINED      HARMONIC     1.000E+00    VERTICAL K  NON-WETTABLE

WETTING CAPABILITY IS NOT ACTIVE IN ANY LAYER

PARAMETERS DEFINED IN THE LPF PACKAGE

PARAMETER NAME:HK1          TYPE:HK     CLUSTERS:   1
Parameter value from package file is:   1.00000E-03
               LAYER:   1    MULTIPLIER ARRAY: NONE    ZONE ARRAY: ALL

PARAMETER NAME:HK2          TYPE:HK     CLUSTERS:   1
Parameter value from package file is:   1.00000E-04
               LAYER:   2    MULTIPLIER ARRAY: NONE    ZONE ARRAY: ALL

PARAMETER NAME:HK3          TYPE:HK     CLUSTERS:   1
Parameter value from package file is:   2.00000E-04
               LAYER:   3    MULTIPLIER ARRAY: NONE    ZONE ARRAY: ALL

PARAMETER NAME:VKCB1        TYPE:VKCB   CLUSTERS:   1
Parameter value from package file is:    1.0000
               LAYER:   1    MULTIPLIER ARRAY: MULT1    ZONE ARRAY: ALL

PARAMETER NAME:VKCB2        TYPE:VKCB   CLUSTERS:   1
Parameter value from package file is:   0.50000
               LAYER:   2    MULTIPLIER ARRAY: MULT1    ZONE ARRAY: ALL

   HYD. COND. ALONG ROWS FOR LAYER   1 WILL BE DEFINED BY PARAMETERS
(PRINT FLAG=   0)

   HYD. COND. ALONG ROWS is defined by the following parameters:
HK1

   HYD. COND. ALONG ROWS =  1.000000E-03 FOR LAYER   1

     VERTICAL HYD. COND. =   1.00000     FOR LAYER   1

QUASI3D VERT. HYD. COND. FOR LAYER   1 WILL BE DEFINED BY PARAMETERS
(PRINT FLAG=   0)

QUASI3D VERT. HYD. COND. is defined by the following parameters:
VKCB1

QUASI3D VERT. HYD. COND. =  1.000000E-06 FOR LAYER   1
```

Figure A–3. Listing File for example problem with parameters.—Continued

```
   HYD. COND. ALONG ROWS FOR LAYER   2 WILL BE DEFINED BY PARAMETERS
(PRINT FLAG=   0)

   HYD. COND. ALONG ROWS is defined by the following parameters:
HK2

   HYD. COND. ALONG ROWS =  1.000000E-04 FOR LAYER   2

     VERTICAL HYD. COND. =   1.00000     FOR LAYER   2

QUASI3D VERT. HYD. COND. FOR LAYER   2 WILL BE DEFINED BY PARAMETERS
(PRINT FLAG=   0)

QUASI3D VERT. HYD. COND. is defined by the following parameters:
VKCB2

QUASI3D VERT. HYD. COND. =  5.000000E-07 FOR LAYER   2

   HYD. COND. ALONG ROWS FOR LAYER   3 WILL BE DEFINED BY PARAMETERS
(PRINT FLAG=   0)

   HYD. COND. ALONG ROWS is defined by the following parameters:
HK3

   HYD. COND. ALONG ROWS =  2.000000E-04 FOR LAYER   3

     VERTICAL HYD. COND. =   1.00000     FOR LAYER   3

WEL -- WELL PACKAGE, VERSION 7, 5/2/2005 INPUT READ FROM UNIT   12
    1 Named Parameters        12 List entries
MAXIMUM OF     15 ACTIVE WELLS AT ONE TIME


    1 Well parameters

PARAMETER NAME:WELL1        TYPE:Q
Parameter value from package file is:    1.0000
  NUMBER OF ENTRIES:     12

WELL NO.  LAYER   ROW   COL   STRESS FACTOR
----------------------------------------------------
     1      1      9      8      -5.000
     2      1      9     10      -5.000
     3      1      9     12      -5.000
     4      1      9     14      -5.000
     5      1     11      8      -5.000
     6      1     11     10      -5.000
     7      1     11     12      -5.000
     8      1     11     14      -5.000
     9      1     13      8      -5.000
    10      1     13     10      -5.000
    11      1     13     12      -5.000
    12      1     13     14      -5.000
```

Figure A-3. Listing File for example problem with parameters.—Continued

```
 DRN -- DRAIN PACKAGE, VERSION 7, 5/2/2005 INPUT READ FROM UNIT   13
     1 Named Parameters         2 List entries
 MAXIMUM OF      9 ACTIVE DRAINS AT ONE TIME


     1 Drain parameters

 PARAMETER NAME:DRN1         TYPE:DRN
 Parameter value from package file is:    1.0000
   NUMBER OF ENTRIES:      2

 DRAIN NO.  LAYER   ROW   COL     DRAIN EL.  STRESS FACTOR
 ----------------------------------------------------------------
      1      1      8      2       0.000           1.000
      2      1      8      3       0.000           1.000

 RCH -- RECHARGE PACKAGE, VERSION 7, 5/2/2005 INPUT READ FROM UNIT   18
     2 Named Parameters
 OPTION 1 -- RECHARGE TO TOP LAYER


     2 Recharge parameters

 PARAMETER NAME:RCH1         TYPE:RCH    CLUSTERS:   1
 Parameter value from package file is:   3.00000E-08
                MULTIPLIER ARRAY: NONE    ZONE ARRAY: RCHZONES
                ZONE VALUES:    1

 PARAMETER NAME:RCH2         TYPE:RCH    CLUSTERS:   1
 Parameter value from package file is:   3.00000E-08
                MULTIPLIER ARRAY: NONE    ZONE ARRAY: RCHZONES
                ZONE VALUES:    2

 SIP -- STRONGLY-IMPLICIT PROCEDURE SOLUTION PACKAGE
                    VERSION 7, 5/2/2005 INPUT READ FROM UNIT   19
 MAXIMUM OF  50 ITERATIONS ALLOWED FOR CLOSURE
  5 ITERATION PARAMETERS

          SOLUTION BY THE STRONGLY IMPLICIT PROCEDURE
          -------------------------------------------
 MAXIMUM ITERATIONS ALLOWED FOR CLOSURE =       50
                ACCELERATION PARAMETER =     1.0000
      HEAD CHANGE CRITERION FOR CLOSURE =    0.10000E-02
      SIP HEAD CHANGE PRINTOUT INTERVAL =        1

     5 ITERATION PARAMETERS CALCULATED FROM SPECIFIED WSEED = 0.00100000 :

  0.000000E+00 0.822172E+00 0.968377E+00 0.994377E+00 0.999000E+00
1
```

Figure A-3. Listing File for example problem with parameters.—Continued

```
                          STRESS PERIOD NO.    1, LENGTH =   86400.00
                          -----------------------------------------------

                            NUMBER OF TIME STEPS =     1

                             MULTIPLIER FOR DELT =     1.000

                          INITIAL TIME STEP SIZE =   86400.00

WELL NO.  LAYER   ROW   COL   STRESS RATE
-------------------------------------------------
     1      3      5     11      -5.000
     2      2      4      6      -5.000
     3      2      6     12      -5.000

Parameter:  WELL1
WELL NO.  LAYER   ROW   COL   STRESS RATE
-------------------------------------------------
     4      1      9      8      -5.000
     5      1      9     10      -5.000
     6      1      9     12      -5.000
     7      1      9     14      -5.000
     8      1     11      8      -5.000
     9      1     11     10      -5.000
    10      1     11     12      -5.000
    11      1     11     14      -5.000
    12      1     13      8      -5.000
    13      1     13     10      -5.000
    14      1     13     12      -5.000
    15      1     13     14      -5.000

    15 WELLS

DRAIN NO.  LAYER   ROW   COL     DRAIN EL.  CONDUCTANCE
-------------------------------------------------------------
     1      1      8      4       10.00           1.000
     2      1      8      5       20.00           1.000
     3      1      8      6       30.00           1.000
     4      1      8      7       50.00           1.000
     5      1      8      8       70.00           1.000
     6      1      8      9       90.00           1.000
     7      1      8     10       100.0           1.000

Parameter:  DRN1
DRAIN NO.  LAYER   ROW   COL     DRAIN EL.  CONDUCTANCE
-------------------------------------------------------------
     8      1      8      2       0.000           1.000
     9      1      8      3       0.000           1.000

     9 DRAINS
```

Figure A–3. Listing File for example problem with parameters.—Continued

```
 RECH array defined by the following parameters:
 Parameter:  RCH1
 Parameter:  RCH2

                 RECHARGE =  3.000000E-08

 SOLVING FOR HEAD

    31 ITERATIONS FOR TIME STEP   1 IN STRESS PERIOD    1

 MAXIMUM HEAD CHANGE FOR EACH ITERATION:

    HEAD CHANGE   HEAD CHANGE   HEAD CHANGE   HEAD CHANGE   HEAD CHANGE
  LAYER,ROW,COL LAYER,ROW,COL LAYER,ROW,COL LAYER,ROW,COL LAYER,ROW,COL
 -----------------------------------------------------------------------
     -22.41         12.48         13.39         48.21         35.91
  (  3,  5, 11) (  1,  1, 15) (  3,  1, 14) (  1,  1, 15) (  3,  1, 13)
      2.482         1.430         6.214         7.411         13.66
  (  1,  9, 14) (  3, 10, 13) (  1, 12, 14) (  3, 11, 14) (  1, 15, 15)
     0.5503        0.4821        0.4711         2.019         2.302
  (  3,  8,  7) (  2,  6,  9) (  3,  5, 10) (  1, 11, 14) (  3,  5, 13)
     0.1108        0.7058E-01    0.2819        0.3141        0.3320
  (  1, 13, 12) (  3, 12, 11) (  1, 14, 14) (  3, 13, 14) (  1, 15, 15)
     0.7853E-02    0.1586E-01    0.1777E-01    0.7910E-01    0.8499E-01
  (  1, 13, 12) (  2, 11, 11) (  3, 11, 10) (  1, 14, 14) (  3,  7, 14)
     0.4169E-02    0.2555E-02    0.9769E-02    0.1082E-01    0.1030E-01
  (  1, 13, 14) (  3, 14, 15) (  1, 14, 14) (  3, 13, 14) (  1, 15, 15)
     0.2430E-03
  (  1, 13, 12)


 OUTPUT CONTROL FOR STRESS PERIOD    1   TIME STEP   1
    PRINT HEAD FOR ALL LAYERS
1
               HEAD IN LAYER   1 AT END OF TIME STEP   1 IN STRESS PERIOD    1
  ---------------------------------------------------------------------------

            1           2           3           4           5           6
            7           8           9          10          11          12
           13          14          15
 .....................................................................
   1    0.000       24.94       44.01       59.26       71.82       82.52
        91.91       100.0       106.9       112.6       117.4       121.3
        124.3       126.4       127.4
   2    0.000       24.45       43.10       57.98       70.17       80.57
        90.12       98.40       105.3       111.0       115.7       119.6
        122.7       124.9       126.1
   3    0.000       23.45       41.30       55.43       66.78       76.21
        86.51       95.20       102.2       107.6       112.0       116.1
        119.6       122.1       123.4
   4    0.000       21.92       38.61       51.75       61.79       68.03
        81.34       90.75       97.64       102.5       106.1       110.7
        114.9       117.9       119.4
```

Figure A–3. Listing File for example problem with parameters.—Continued

```
   5    0.000       19.73       34.92       47.32       57.69       66.74
        77.09       85.76       92.22       96.15       97.29       103.1
        108.8       112.5       114.3
   6    0.000       16.51       29.50       40.90       51.30       61.21
        71.19       79.85       86.47       90.82       93.03       94.23
        102.1       106.4       108.4
   7    0.000       11.55       21.10       31.21       41.40       51.84
        63.08       72.68       79.95       84.92       88.60       91.66
        96.43       99.82       101.8
   8    0.000       3.483       6.832       16.25       26.30       36.97
        52.59       64.31       72.52       77.25       81.99       85.00
        89.27       91.72       94.33
   9    0.000       10.54       19.11       28.12       36.92       45.27
        52.95       55.38       65.15       66.07       73.93       73.79
        80.84       80.17       86.49
  10    0.000       14.62       25.86       35.38       43.49       50.11
        54.93       57.55       62.95       65.55       70.39       72.44
        76.72       78.26       81.79
  11    0.000       17.11       29.96       40.01       47.78       53.24
        55.81       53.33       60.27       59.29       66.43       65.45
        72.22       71.04       77.62
  12    0.000       18.68       32.56       43.07       50.81       55.92
        58.33       58.47       61.93       63.18       67.12       68.50
        72.29       73.46       76.85
  13    0.000       19.67       34.24       45.14       53.01       58.04
        59.91       56.75       62.59       60.91       67.22       65.75
        71.90       70.35       76.48
  14    0.000       20.27       35.27       46.48       54.61       60.08
        63.17       64.52       67.25       68.79       71.64       73.18
        75.84       77.03       79.09
  15    0.000       20.56       35.78       47.16       55.48       61.26
        65.02       67.52       69.94       72.01       74.29       76.22
        78.22       79.66       80.82
1
             HEAD IN LAYER   2 AT END OF TIME STEP   1 IN STRESS PERIOD    1
  ---------------------------------------------------------------------------

           1           2           3           4           5           6
           7           8           9          10          11          12
          13          14          15
 ...........................................................................
   1    0.000       24.66       43.73       59.02       71.61       82.32
        91.72       99.86       106.7       112.5       117.2       121.1
        124.1       126.2       127.3
   2    0.000       24.17       42.83       57.74       69.95       80.36
        89.93       98.22       105.1       110.8       115.5       119.4
        122.6       124.8       125.9
   3    0.000       23.17       41.03       55.19       66.53       75.77
        86.29       95.02       102.0       107.4       111.8       116.0
        119.5       121.9       123.2
   4    0.000       21.65       38.34       51.50       61.35       60.17
        80.90       90.55       97.45       102.3       105.4       110.4
        114.8       117.7       119.2
```

Figure A–3. Listing File for example problem with parameters.—Continued

```
   5    0.000       19.48       34.65       47.07       57.44       66.30
        76.85       85.57       92.00       95.41       91.09       102.1
        108.6       112.4       114.2
   6    0.000       16.27       29.24       40.65       51.07       60.98
        70.98       79.65       86.28       90.54       92.06       86.23
        101.7       106.2       108.3
   7    0.000       11.38       20.95       31.05       41.25       51.70
        62.90       72.48       79.76       84.73       88.35       91.24
        96.22       99.65       101.6
   8    0.000       4.209       8.330       17.58       27.58       38.25
        52.94       64.19       72.34       77.12       81.81       84.86
        89.10       91.59       94.17
   9    0.000       10.38       18.96       27.98       36.79       45.16
        52.86       56.13       65.08       66.79       73.87       74.48
        80.77       80.84       86.38
  10    0.000       14.40       25.61       35.15       43.27       49.91
        54.76       57.48       62.79       65.49       70.24       72.37
        76.57       78.20       81.64
  11    0.000       16.87       29.70       39.78       47.56       53.05
        55.68       54.09       60.20       60.04       66.37       66.18
        72.16       71.75       77.51
  12    0.000       18.43       32.31       42.85       50.60       55.73
        58.16       58.41       61.78       63.12       66.98       68.44
        72.15       73.40       76.69
  13    0.000       19.42       33.98       44.91       52.80       57.85
        59.78       57.50       62.53       61.65       67.16       66.48
        71.84       71.06       76.37
  14    0.000       20.02       35.02       46.26       54.41       59.88
        62.99       64.39       67.08       68.66       71.48       73.06
        75.68       76.91       78.93
  15    0.000       20.30       35.52       46.94       55.28       61.07
        64.84       67.34       69.76       71.84       74.11       76.04
        78.04       79.49       80.65
1
             HEAD IN LAYER   3 AT END OF TIME STEP   1 IN STRESS PERIOD    1
  ----------------------------------------------------------------------------

           1           2           3           4           5           6
           7           8           9          10          11          12
          13          14          15
 ...........................................................................
   1    1.800       24.34       43.36       58.70       71.33       82.06
        91.48       99.63       106.5       112.3       117.0       120.9
        123.9       126.0       127.1
   2    1.764       23.85       42.46       57.42       69.66       80.07
        89.68       97.99       104.9       110.6       115.3       119.2
        122.4       124.6       125.7
   3    1.691       22.86       40.67       54.87       66.20       75.28
        85.98       94.77       101.7       107.2       111.5       115.7
        119.3       121.7       123.0
```

Figure A-3. Listing File for example problem with parameters.—Continued

```
   4    1.578       21.35       37.98       51.17       60.85       62.69
        80.41       90.28       97.19       101.9       104.1       110.0
        114.5       117.5       119.0
   5    1.415       19.18       34.30       46.75       57.10       65.80
        76.54       85.30       91.67       94.17       77.46       100.7
        108.2       112.1       114.0
   6    1.176       15.99       28.91       40.33       50.76       60.67
        70.70       79.38       86.01       90.12       90.60       88.55
        101.2       106.0       108.0
   7   0.8273       11.21       20.79       30.88       41.09       51.55
        62.67       72.22       79.50       84.46       87.98       90.77
        95.94       99.41       101.4
   8   0.4331       5.131       10.19       19.27       29.19       39.84
        53.40       64.07       72.11       76.95       81.58       84.68
        88.88       91.44       93.95
   9   0.7543       10.22       18.82       27.84       36.66       45.06
        52.78       57.03       65.02       67.64       73.81       75.31
        80.72       81.64       86.24
  10    1.039       14.13       25.29       34.85       42.99       49.65
        54.54       57.44       62.61       65.44       70.05       72.33
        76.39       78.15       81.43
  11    1.224       16.59       29.37       39.47       47.28       52.79
        55.53       55.01       60.16       60.94       66.33       67.06
        72.13       72.60       77.38
  12    1.341       18.15       31.97       42.54       50.32       55.47
        57.94       58.37       61.60       63.08       66.80       68.41
        71.97       73.36       76.49
  13    1.415       19.14       33.65       44.61       52.53       57.60
        59.63       58.39       62.48       62.54       67.12       67.35
        71.80       71.90       76.24
  14    1.460       19.73       34.68       45.96       54.13       59.63
        62.76       64.24       66.87       68.52       71.27       72.91
        75.47       76.77       78.71
  15    1.481       20.01       35.18       46.63       55.00       60.81
        64.59       67.11       69.52       71.61       73.87       75.82
        77.81       79.27       80.42
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:                                IN:
          ---                                ---
            STORAGE =           0.0000            STORAGE =           0.0000
      CONSTANT HEAD =           0.0000      CONSTANT HEAD =           0.0000
              WELLS =           0.0000              WELLS =           0.0000
             DRAINS =           0.0000             DRAINS =           0.0000
           RECHARGE =    13608000.0000           RECHARGE =         157.5000

           TOTAL IN =    13608000.0000           TOTAL IN =         157.5000
```

Figure A–3. Listing File for example problem with parameters.—Continued

```
          OUT:                                     OUT:
          ----                                     ----
             STORAGE =           0.0000               STORAGE =           0.0000
       CONSTANT HEAD =     4326522.0000         CONSTANT HEAD =          50.0755
               WELLS =     6480000.0000                 WELLS =          75.0000
              DRAINS =     2801081.2500                DRAINS =          32.4199
            RECHARGE =           0.0000              RECHARGE =           0.0000

           TOTAL OUT =    13607603.0000             TOTAL OUT =         157.4954

            IN - OUT =         397.0000              IN - OUT =       4.5929E-03

 PERCENT DISCREPANCY =           0.00     PERCENT DISCREPANCY =           0.00




          TIME SUMMARY AT END OF TIME STEP   1 IN STRESS PERIOD    1
                     SECONDS     MINUTES      HOURS       DAYS        YEARS
                     -----------------------------------------------------------
   TIME STEP LENGTH  86400.      1440.0      24.000      1.0000     2.73785E-03
 STRESS PERIOD TIME  86400.      1440.0      24.000      1.0000     2.73785E-03
         TOTAL TIME  86400.      1440.0      24.000      1.0000     2.73785E-03
1

 Run end date and time (yyyy/mm/dd hh:mm:ss): 2005/11/25  9:54:02
 Elapsed run time:  0.297 Seconds
```

Figure A–3. Listing File for example problem with parameters.—Continued