DOCUMENTATION OF FINITE-Difference Model

For

Simulation of Three-Dimensional Ground-Water Flow

By

Peter C. Trescott

U.S. Geological Survey Open File Report 75-438

September 1975

Reprint July 1976

(Supplemented by OFR 76-591)
ACKNOWLEDGMENT

The efficiency of the program has been improved significantly by Steven P. Larson who added the option of storing input arrays on disk and modified the solution routine to use only single-subscript arrays.

Requests for copies of this documentation and the card deck should be directed to:

Ralph N. Eicher, Chief
Office of Teleprocessing, M.S. 805
U.S. Geological Survey
Reston, Virginia 22092
## CONTENTS

Abstract ................................................................. 1
Introduction ......................................................... 2
Theoretical development ..............................................
  Ground-water flow equation ......................................... 3
  Finite-difference approximation ...................................... 4
  Source term .......................................................... 10
  Numerical solution by the strongly implicit procedure ........... 11
  Iteration parameters ................................................ 24
  Boundaries in the numerical scheme ................................ 26
Practical considerations for application ............................ 27
  Initial conditions ..................................................... 27
  Treatment of boundary conditions ................................... 27
  Treatment of confining layers ....................................... 28
  Designing the finite-difference grid ................................. 30
References ........................................................................ 32
APPENDIX I: NOTATION ................................................... I-1
APPENDIX II: COMPUTER PROGRAM ........................................ II-1
  Main program .......................................................... II-1
  Subroutine DATAI ...................................................... II-1
    Time parameters ..................................................... II-3
    Initialization ....................................................... II-4
  Subroutine STEP ....................................................... II-4
  Subroutine SOLVE ..................................................... II-5
  Subroutine COEF ....................................................... II-6
  Subroutine CHECK ..................................................... II-7
  Subroutine PRNTAI .................................................... II-7
  BLOCK DATA ........................................................... II-10
  Technical information ................................................ II-10
    Use of disk facilities for storage of array data and interim results ........................................ II-10
    Storage requirements ............................................... II-13
    Computation time .................................................... II-14
    FORTRAN IV ......................................................... II-14
  Program limitations .................................................. II-14
APPENDIX III: DATA DECK INSTRUCTIONS ................................ III-1
APPENDIX IV: EXAMPLE SIMULATION ...................................... IV-1
APPENDIX V: DEFINITION OF PROGRAM VARIABLES ..................... V-1
APPENDIX VI: PROGRAM LISTING ........................................... VI-1
FIGURES

1. Index scheme for finite-difference grid and the normal ordering of nodes for 3 X 3 X 3 problem. The numerical values of the indices (i,j,k) for this problem are given in the lower left-hand corner of each block .................................................. 5

2. Matrix $A$ ........................................................................................................ 13

3. Lower triangular factor $L$ of $(A+B)$ .............................................................. 14

4. Upper triangular factor $U$ of $(A+B)$ .............................................................. 14

5. Matrix $(A+B)$ .................................................................................................. 16

6. Coefficients for normal ordering scheme ......................................................... 18

7. Reverse ordering of nodes for a 3 X 3 X 3 problem. The numerical values of the indices (i,j,k) for this problem are given in the lower left-hand corner of each block .................................................. 22

8. Coefficients of reverse ordering scheme ......................................................... 23

9. Simulating two aquifers separated by a confining bed with three layers of nodes (a) or two layers of nodes (b) ................................................................. 29

10. Orientation of maps on computer page ......................................................... II-9

11. Schematic illustration of example problem ................................................. IV-2

TABLES

1. Additional arrays required for the options .................................................... II-1

2. Arrays passed to the subroutines and their relative location in the vector $Y$ ........................................................................................................ II-2

3. Core requirements for compiling and executing program ............................. II-13

4. Data input for example problem .................................................................... IV-3
INTRODUCTION

This report documents a finite-difference model for simulation of ground-water flow in three dimensions. Although the techniques for constructing numerical models of ground-water flow in three dimensions have been available in the petroleum literature for some time, the large computer-memory requirements for simulation of realistic ground-water problems have discouraged development of such models. To minimize the core requirements for these simulations, Bredehoeft and Pinder (1970) described a quasi three-dimensional model which is a sequence of two-dimensional ground-water flow models coupled by terms representing flow through intervening confining beds. In practice there has been some difficulty getting convergence with this approach. The fully three-dimensional model described in this report can be reduced to a quasi three-dimensional model in terms of the equations being solved and computer-memory requirements, yet converges to a solution much faster because all equations are solved simultaneously.

The iterative numerical technique used to solve the set of simultaneous finite-difference equations is the strongly implicit procedure (SIP) originally described by Stone (1968) for problems in two dimensions and extended to three dimensions by Weinstein, Stone and Kwan (1969). Weinstein and others (1969) claim that SIP converges faster than the iterative alternating-direction implicit procedure (ADI) (used for solution in two dimensions by Bredehoeft and Pinder, 1970) and that SIP is less subject to roundoff errors than ADI.
THEORETICAL DEVELOPMENT

Ground-water flow equation

The flow of ground water in a porous medium in three dimensions may be expressed as (Jacob, 1950, Cooper, 1966)

\[ \nabla^2 h = \frac{S_s}{K} \frac{\partial h}{\partial t} \]  

(1)

in which  
- \( h \) is hydraulic head (L);  
- \( S_s \) is specific storage (L\(^{-1}\));  
- \( K \) is hydraulic conductivity (L T\(^{-1}\)).

Permitting hydraulic conductivity to be heterogeneous and anisotropic and adding a source term, equation 1 becomes

\[ \nabla (K_{ij} \frac{\partial h}{\partial x_j}) = S_s \frac{\partial h}{\partial t} + W(x,y,z,t) \]  

(2)

in which \( W(x,y,z,t) \) is a volumetric flux per unit volume (T\(^{-1}\)).

Assuming that the coordinate axes \( x, y \) and \( z \) are aligned with the principal directions of the hydraulic conductivity tensor, the cross-product terms drop out of equation 2 and it becomes in expanded form

\[ \frac{\partial}{\partial x} (K_{xx} \frac{\partial h}{\partial x}) + \frac{\partial}{\partial y} (K_{yy} \frac{\partial h}{\partial y}) + \frac{\partial}{\partial z} (K_{zz} \frac{\partial h}{\partial z}) = S_s \frac{\partial h}{\partial t} + W(x,y,z,t) \]  

(3)

in which  
- \( K_{xx}, K_{yy}, K_{zz} \) are the principal components of the hydraulic conductivity tensor (L T\(^{-1}\)).
In the finite-difference simulator, it is often convenient to represent a hydraulic unit by one layer of nodes. For this approach, equation 3 is multiplied by \( b \), the thickness of the hydraulic unit giving

\[
\frac{\partial}{\partial x} \left( T_{xx} \frac{\partial h}{\partial x} \right) + \frac{\partial}{\partial y} \left( T_{yy} \frac{\partial h}{\partial y} \right) + \frac{\partial}{\partial z} \left( bK \frac{\partial h}{\partial z} \right) = S' \frac{\partial h}{\partial t} + bW(x,y,z,t) \quad (4)
\]

in which \( T_{xx}, T_{yy} \) are the principal components of the transmissivity tensor \( (LT^{-1}) \);

\( S' \) is the storage coefficient (dimensionless).

Although the model is designed to solve equation 4, it will solve equation 3 by substituting hydraulic conductivity, specific storage, and \( W(x,y,z,t) \) for transmissivity, storage coefficient, and \( bW(x,y,z,t) \), respectively.

**Finite-difference approximation**

In order to solve equation 3 or 4 for a heterogeneous, anisotropic porous medium with irregular boundaries, one approach is to subdivide the region into blocks in which the medium properties are assumed to be uniform. The continuous derivatives in equation 3 or 4 are replaced by finite-difference approximations for the derivatives at a point (the node at the center of the block). The result is \( N \) equations in \( N \) unknowns (head values at the nodes) where \( N \) is the number of blocks representing the porous medium.

Utilizing a block-centered, finite-difference grid in which variable grid spacing is permitted (figure 1), equation 4 may be approximated as
Figure 1.--Index scheme for finite-difference grid and the normal ordering of nodes for 3 X 3 X 3 problem. The numerical values of the indices (i,j,k) for this problem are given in the lower left-hand corner of each block.
\[
\frac{1}{\Delta x_j} \left[ (T_{xx} \frac{\partial h}{\partial x})_{i,j+\frac{1}{2},k} - (T_{xx} \frac{\partial h}{\partial x})_{i,j-\frac{1}{2},k} \right] + \\
\frac{1}{\Delta y_i} \left[ (T_{yy} \frac{\partial h}{\partial y})_{i+\frac{1}{2},j,k} - (T_{yy} \frac{\partial h}{\partial y})_{i-\frac{1}{2},j,k} \right] + \\
\frac{1}{\Delta z_k} \left[ (bK_{zz} \frac{\partial h}{\partial z})_{i,j,k+\frac{1}{2}} - (bK_{zz} \frac{\partial h}{\partial z})_{i,j,k-\frac{1}{2}} \right] = S'_{i,j,k} \\
\Delta t \left( h_{i,j,k} - \hat{h}_{i,j,k} \right) + bW_{i,j,k}
\]

in which \(\Delta x_j\) is the space increment in the x direction for column \(j\) (L);
\(\Delta y_i\) is the space increment in the y direction for row \(i\) (L);
\(\Delta z_k\) is the space increment in the z direction for layer \(k\) (L);
\(\Delta t\) is the time increment (T);
\(i\) is the index in the y dimension;
\(j\) is the index in the x dimension;
\(k\) is the index in the z dimension;
\(\hat{h}_{i,j,k}\) is the hydraulic head at the previous time step (L).
The final approximation for equation 4 is

\[
\frac{1}{\Delta x_j} \left\{ \left[ T_{xx}(i,j+\frac{1}{2},k) \frac{(h_{i+1,j,k} - h_{i,j,k})}{\Delta x_j} \right] - \left[ T_{xx}(i,j-\frac{1}{2},k) \frac{(h_{i,j,k} - h_{i-1,j,k})}{\Delta x_j} \right] \right\} +
\]

\[
\frac{1}{\Delta y_i} \left\{ \left[ T_{yy}(i+\frac{1}{2},j,k) \frac{(h_{i+1,j,k} - h_{i,j,k})}{\Delta y_i} \right] - \left[ T_{yy}(i-\frac{1}{2},j,k) \frac{(h_{i,j,k} - h_{i-1,j,k})}{\Delta y_i} \right] \right\} +
\]

\[
\frac{1}{\Delta z_k} \left\{ \left[ (b_k^{zz})_{i,j,k+\frac{1}{2}} \frac{(h_{i,j,k+1} - h_{i,j,k})}{\Delta z_k} \right] - \left[ (b_k^{zz})_{i,j,k-\frac{1}{2}} \frac{(h_{i,j,k} - h_{i,j,k-1})}{\Delta z_k} \right] \right\}
\]

\[
= \frac{S'_{i,j,k}}{\Delta t} (h_{i,j,k} - \hat{h}_{i,j,k}) + bW_{i,j,k} \quad (6)
\]

Following a convention similar to that introduced by Stone (1968), the notation in equation 6 may be simplified by writing

\[
F_{i,j,k} \frac{(h_{i+1,j,k} - h_{i,j,k})}{\Delta t} - D_{i,j,k} \frac{(h_{i,j,k} - h_{i-1,j,k})}{\Delta t} +
\]

\[
H_{i,j,k} \frac{(h_{i+1,j,k} - h_{i,j,k})}{\Delta t} - B_{i,j,k} \frac{(h_{i,j,k} - h_{i-1,j,k})}{\Delta t} +
\]

\[
S_{i,j,k} \frac{(h_{i,j,k+1} - h_{i,j,k})}{\Delta t} - Z_{i,j,k} \frac{(h_{i,j,k} - h_{i,j,k-1})}{\Delta t} +
\]

\[
= \frac{S'_{i,j,k}}{\Delta t} (h_{i,j,k} - \hat{h}_{i,j,k}) + bW_{i,j,k} \quad (7)
\]
in which

\[ B_{i,j,k} = \left( \frac{2T_{yy}[i,j,k]T_{yy}[i-1,j,k]}{T_{yy}[i,j,k] \frac{\Delta y_i}{\Delta y_{i-1}} + T_{yy}[i-1,j,k] \frac{\Delta y_i}{\Delta y_i}} \right) / \Delta y_i \]  

(8a)

and the terms in brackets is the harmonic mean of

\[ \frac{T_{yy}[i,j,k]}{\Delta y_i}, \frac{T_{yy}[i-1,j,k]}{\Delta y_{i-1}} \]

Similarly,

\[ D_{i,j,k} = \left( \frac{2T_{xx}[i,j,k]T_{xx}[i,j-1,k]}{T_{xx}[i,j,k] \frac{\Delta x_j}{\Delta x_{j-1}} + T_{xx}[i,j-1,k] \frac{\Delta x_j}{\Delta x_j}} \right) / \Delta x_j ; \]  

(8b)

\[ F_{i,j,k} = \left( \frac{2T_{xx}[i,j,k]T_{xx}[i,j+1,k]}{T_{xx}[i,j,k] \frac{\Delta x_{j+1}}{\Delta x_j} + T_{xx}[i,j+1,k] \frac{\Delta x_{j+1}}{\Delta x_{j+1}}} \right) / \Delta x_j ; \]  

(8c)

\[ H_{i,j,k} = \left( \frac{2T_{yy}[i+1,j,k]T_{yy}[i,j,k]}{T_{yy}[i,j,k] \frac{\Delta y_{i+1}}{\Delta y_i} + T_{yy}[i+1,j,k] \frac{\Delta y_{i+1}}{\Delta y_{i+1}}} \right) / \Delta y_i ; \]  

(8d)

\[ S_{i,j,k} = \left( \frac{2(bK_{zz})_{i,j,k+1}(bK_{zz})_{i,j,k}}{(bK_{zz})_{i,j,k} \Delta z_{k+1} + (bK_{zz})_{i,j,k+1} \Delta z_{k+1}} \right) / \Delta z_k ; \]  

(8e)

\[ Z_{i,j,k} = \left( \frac{2(bK_{zz})_{i,j,k-1}(bK_{zz})_{i,j,k}}{(bK_{zz})_{i,j,k} \Delta z_{k-1} + (bK_{zz})_{i,j,k-1} \Delta z_{k-1}} \right) / \Delta z_k. \]  

(8f)

Use of the harmonic mean 1) insures continuity across block boundaries at steady state for an irregular grid and 2) makes the appropriate coefficients zero at no-flow boundaries.
If the upper hydrologic unit is under water-table conditions, the specific yield is inserted for the storage coefficient and the transmissivities in equations 8 are defined as a function of the head for the previous iteration. As an example

\[ T^n_{xx}(i,j,k) = K_{xx}(i,j,k) b^{n-1}_{i,j,k} \]

in which \( b^{n-1}_{i,j,k} \) is the saturated thickness of the upper hydrologic unit at iteration \( n-1 \);

\( n \) is the iteration index.

The notation in equation 7 can be simplified by eliminating all subscripts not including \( a'+1 \) or \( '-1 \). After rearranging equation 7 by placing all of the unknowns on the left-hand side, it becomes

\[ Bh_{i-1} + Dh_{j-1} + Eh + Fh_{j+1} + Hh_{i+1} + Sh_{k+1} + Zh_{k-1} = Q \]  \( (9) \)

in which

\[ E = - (B + D + F + H + S + Z + \frac{S'}{\Delta t}) ; \]

\[ Q = - \frac{S'}{\Delta t} \hat{h} + bW. \]
Source Term

The source term $W(x,y,z,t)$ can include well discharge and recharge from precipitation. In the model the source term is computed as

$$(bw)_{i,j,k} = \frac{Q_w[i,j,k]}{\Delta x \Delta y} - q_{re[i,j,k]}$$

in which

- $Q_w[i,j,k]$ is the well discharge ($L^3 T^{-1}$);
- $q_{re[i,j,k]}$ is the volumetric flux per unit area of the uppermost hydraulic unit ($LT^{-1}$). (Adding this term to other layers will require modifications to the code.)

To solve equation 3 instead of 4, divide $Q_w[i,j,k]$ and $q_{re[i,j,k]}$ by $\Delta z_k$ for data input.
Numerical Solution by the Strongly Implicit Procedure

The following development follows that given in Weinstein, Stone and Kwan (1969). The nodes \((i,j,k)\) may be ordered as in figure 1 so that \(j\) is swept through first \((j = 1, 2, ..., J)\), \(i\) is swept through second \((i = 1, 2, ..., I)\) and \(k\) last \((k = 1, 2, ..., K')\). Ordering the set of equations 9 in the same manner, they may be represented by the matrix equation

\[
\bar{A} \bar{h} = \bar{Q}
\]  

(10)

in which

\[
\bar{h} = \begin{bmatrix}
h_{1,1,1} \\
h_{1,2,1} \\
\vdots \\
h_{1,J,1} \\
h_{2,1,1} \\
h_{2,2,1} \\
\vdots \\
h_{2,J,1} \\
\vdots \\
h_{I,J,K'}
\end{bmatrix}
\]

and \(I\) is the number of rows;

\(J\) is the number of columns;

\(K'\) is the number of layers.

11
\( \bar{Q} \) has the same form as \( \bar{h} \), and the matrix $\bar{A}$ is shown in figure 2. Equation 10 may be solved directly by factoring $\bar{A}$ into lower $\bar{L}$ and upper $\bar{U}$ triangular matrices. However, $\bar{L}$ has non-zero elements in all the diagonals from Z to E and $\bar{U}$ has non-zero elements in all of the diagonals from E to S. Consequently, solution by Gaussian elimination for problems of practical size requires excessive computation time and computer storage.

To include as many zero diagonals in $\bar{L}$ and $\bar{U}$ as possible, a modifying matrix $\bar{B}$ can be added to $\bar{A}$ such that $(\bar{A} + \bar{B})$ factors into $\bar{L}$ and $\bar{U}$ as shown in figures 3 and 4. With this modification, equation 10 becomes

$$ (\bar{A} + \bar{B}) \, \bar{h} = \bar{Q} + \bar{B} \bar{h} $$

Equation 11 can be solved readily if the right-hand side is known. This leads to an iterative scheme defined by the following equation:

$$ (\bar{A} + \bar{B}) \, \bar{h}^n = \bar{Q} + \bar{B} \bar{h}^{n-1} $$

in which $n$ is an iteration index. To reduce roundoff errors, equation 12 is put in residual form by adding and subtracting $\bar{A} \bar{h}^{n-1}$ to the right hand side:

$$ (\bar{A} + \bar{B}) \, \bar{\xi}^n = \bar{R}^{n-1} $$

in which

$$ \bar{\xi}^n = \bar{h}^n - \bar{h}^{n-1}; $$

$$ \bar{R}^{n-1} = \bar{Q} - \bar{A} \bar{h}^{n-1}. $$
Figure 2.—Matrix A.
Figure 3.—Lower triangular factor \( L \) of \((A+B)\).

Figure 4.—Upper triangular factor \( U \) of \((A+B)\).
\( R^{n-1} \) is known from the previous iteration. The \((i,j,k)\) element of \( R^{n-1} \) is given by

\[
R^{n-1} = Q - (B_h^{n-1} + D_h^{n-1} + E_h^{n-1} + F_h^{n-1} + H_h^{n-1} + S_h^{n-1} + Z_h^{n-1}).
\]

When \( \xi_{\text{max}}^{n} < c \), an arbitrary error criterion, a solution to equation 13 is obtained.

Derivation of the SIP algorithm requires 1) the relationships defining

\[
(A+B) = LU;
\]

(14)

2) an appropriate definition of \( B \); and 3) relationships among the elements of \( A \) and \( (A+B) \). The matrix \( (A+B) \) is shown in figure 5 and the elements of \( (A+B) \) or \( LU \) are defined by

\[
\begin{align*}
Z' &= a \\
A &= a e_{k-1} \\
T &= a f_{k-1} \\
B' &= b \\
C &= e_{i-1} b \\
D' &= c \\
E' &= a g_{k-1} + b f_{i-1} + e_{j-1} c + d \\
F' &= d e \\
G &= f_{j-1} c \\
H' &= f d \\
U &= b g_{i-1} \\
W &= g_{j-1} c \\
S' &= gd
\end{align*}
\]

(15)
Figure 5.—Matrix \((A+B)\).
The nodal values associated with the elements of \((A + B)\) not in the original difference equation 9 (namely, those associated with the coefficients \(C,G,A,W,T,U\)) are shown in figure 6. Stone (1968) and Weinstein, Stone and Kwan (1969) have found it effective to define \(\bar{B}\) (there are many ways of defining \(\bar{B}\) that will satisfy equation 14) so that the contribution of these additional terms is partially cancelled. Their definition of \(\bar{B}\) can be seen by writing \(\bar{B}_c\) for node \((i,j,k)\) as

\[
\begin{align*}
C [\xi_{i-1,j+1} - \omega (-\xi + \xi_{i-1} + \xi_{j+1})] + \\
G [\xi_{i+1,j-1} - \omega (-\xi + \xi_{j-1} + \xi_{i+1})] + \\
A [\xi_{j+1,k-1} - \omega (-\xi + \xi_{j+1} + \xi_{k-1})] + \\
W [\xi_{j-1,k+1} - \omega (-\xi + \xi_{j-1} + \xi_{k+1})] + \\
T [\xi_{i+1,k-1} - \omega (-\xi + \xi_{i+1} + \xi_{k-1})] + \\
U [\xi_{i-1,k+1} - \omega (-\xi + \xi_{i-1} + \xi_{k+1})]
\end{align*}
\] (16)

in which \(\omega\) is an iteration parameter. Weinstein, Stone and Kwan (1969) define different parameters for each of the three planes in space, but in practice use only one parameter each iteration. The term \((-\xi + \xi_{i-1} + \xi_{j+1})\) is the second-order correct approximation for \(\xi_{i-1,j+1}\). The remaining elements of \(\bar{B}_c\) for node \((i,j,k)\) in 16 have similar relationships. See Remson and others, 1971, p. 226 for derivation of this type of approximation.)
Figure 6.--Coefficients for normal ordering scheme.
With these definitions, equation 13 written for node \((i,j,k)\) is

\[
B\xi_{i-1}^n + DC_{j-1}^n + E\xi_j^n + F\xi_{j+1}^n + H\xi_{i+1}^n + S\xi_{k+1}^n + Z\xi_{k-1}^n + \\
C[\xi_{i-1,j+1}^n - \omega (-\xi_{i-1,j}^n + \xi_{i,j+1}^n)] + \\
G[\xi_{i+1,j-1}^n - \omega (-\xi_{i+1,j-1}^n + \xi_{i,j-1}^n)] + \\
A[\xi_{j+1,k-1}^n - \omega (-\xi_{j+1,k-1}^n + \xi_{j,k-1}^n)] + \\
W[\xi_{j-1,k+1}^n - \omega (-\xi_{j-1,k+1}^n + \xi_{j,k+1}^n)] + \\
T[\xi_{i+k-1}^n - \omega (-\xi_{i+k-1}^n + \xi_{i,k-1}^n)] + \\
U[\xi_{i-1,k+1}^n - \omega (-\xi_{i-1,k+1}^n + \xi_{i,k+1}^n)] = \mathbf{R}^{n-1}. \tag{17}
\]

By collecting coefficients associated with nodal positions in the original difference equation and using equations 15, the SIP algorithm can be derived as

\[
a = Z/(1 - \omega (e_{k-1} + f_{k-1})) \\
b = B/(1 - \omega (e_{i-1} + g_{i-1})) \\
c = D/(1 - \omega (f_{j-1} + g_{j-1})) \\
A = ae_{k-1} \\
C = be_{i-1} \\
G = cf_{j-1} \\
W = cg_{j-1} \\
T = af_{k-1} \\
U = bg_{i-1}
\]
\[ d = E + \omega (A + C + G + W + T + U) - ce_{j-1} - bf_{i-1} - ag_{k-1} \]

\[ e = (F - \omega (A + C))/d \]

\[ f = (H - \omega (G + T))/d \]

\[ g = (S - \omega (W + U))/d. \]  

Equation 14 may be combined with equation 13 to give

\[ LU\xi^n = R^{n-1}. \]  

(19)

Define the vector \( \vec{V}^n \) by

\[ \frac{\nu_n}{\xi} = \vec{V}^n. \]  

(20)

Then equation 19 becomes

\[ L\xi^n = R^{n-1}. \]  

(21)

After solving equations 18, the intermediate vector \( \vec{V}^n \) can be computed; for node \((i,j,k)\) equation 21 is (refer to figure 3 for the elements of \( L \))

\[ a\xi_{k-1}^n + b\xi_{i-1}^n + c\xi_{j-1}^n + d\xi^n = R^{n-1} \]

or

\[ \vec{V}^n = (R^{n-1} - a\xi_{k-1}^n - b\xi_{i-1}^n - c\xi_{j-1}^n)/d. \]

The vector \( \xi^n \) may then be computed by backward substitution.

Equation 20 for node \((i,j,k)\) is (refer to figure 4 for the elements of \( U \))

\[ \xi^n + e\xi_{j+1}^n + f\xi_{i+1}^n + g\xi_{k+1}^n = V^n \]

or

\[ \xi^n = V^n - (e\xi_{j+1}^n + f\xi_{i+1}^n + g\xi_{k+1}^n). \]
Stone (1968) and Weinstein and others (1969) recommend that a second ordering scheme be used every other iteration. In the second ordering scheme, j is swept through first in increasing order (j = 1, 2, ..., J), i is second in decreasing order (i = I, I-1, ..., 1), and k last in decreasing order (k = K; K'-1, ..., 1). (See figure 7.) This tends to give an overall symmetry to (A+B) for the two iterations (compare figures 6 and 8) and speeds convergence.

The algorithm for the second (or 'reverse') ordering scheme is derived in a manner analogous to that for the 'normal' ordering scheme given above. The SIP reverse algorithm is outlined below:

\[ a = S/(1+\omega(e_{k+1} + f_{k+1}) \]
\[ b = H/(1+\omega(e_{i+1} + g_{i+1}) \]
\[ c = D/(1+\omega(f_{j-1} + g_{j-1}) \]
\[ A = ae_{k+1} \]
\[ C = be_{i+1} \]
\[ G = cf_{j-1} \]
\[ W = cg_{j-1} \]
\[ T = af_{k+1} \]
\[ U = bg_{i+1} \]
\[ d = E + \omega (C+G+A+W+T+U) - ag_{k+1} - bf_{i+1} - ce_{j-1} \]
\[ e = (F-\omega(C+A))/d \]
\[ f = (B-\omega(G+T))/d \]
\[ g = (Z-\omega(W+U))/d. \] (22)
Figure 7.—Reverse ordering of nodes for a 3 X 3 X 3 problem. The numerical values of the indices \((i,j,k)\) for this problem are given in the lower left-hand corner of each block.
Figure 8.—Coefficients of reverse ordering scheme.
For node \((i,j,k)\) equation 21 is

\[
\begin{align*}
    aV^n_{k+1} + bV^n_{i+1} + cV^n_{j-1} + dV^n &= R^{n-1} \\
    \text{or} \\
    v^n = (R^{n-1} - aV^n_{k+1} - bV^n_{i+1} - cV^n_{j-1}) / d.
\end{align*}
\]

The vector \(\xi^n\) is computed with equation 20 which, for node \((i,j,k)\), is

\[
\begin{align*}
    \xi^n + e\xi^n_{j+1} + f\xi^n_{i-1} + g\xi^n_{k-1} &= v^n \\
    \text{or} \\
    \xi^n &= v^n - e\xi^n_{j+1} - f\xi^n_{i-1} - g\xi^n_{k-1}.
\end{align*}
\]

**Iteration parameters**

A sequence of iteration parameters ranging from 0 to 1 is cycled until convergence is achieved. The minimum parameter is not critical and 0 is arbitrarily chosen. The maximum parameter is given by

\[
\begin{align*}
    1 - \omega_{\max} &= \min_{\text{grid}} \left[ \frac{\pi^2}{2J^2(1+\rho_1)} , \frac{\pi^2}{21^2(1+\rho_2)} , \frac{\pi^2}{2K^2(1+\rho_3)} \right] \tag{23a}
\end{align*}
\]

in which

\[
\begin{align*}
    \rho_1 &= \frac{K_{yy[i,j,k]}(\Delta x_j)^2}{K_{xx[i,j,k]}(\Delta y_i)^2} + \frac{K_{zz[i,j,k]}(\Delta z_k)^2}{K_{xx[i,j,k]}(\Delta z_k)^2} \tag{23b} \\
    \rho_2 &= \frac{K_{xx[i,j,k]}(\Delta y_i)^2}{K_{yy[i,j,k]}(\Delta x_j)^2} + \frac{K_{zz[i,j,k]}(\Delta z_k)^2}{K_{yy[i,j,k]}(\Delta z_k)^2} \tag{23c} \\
    \rho_3 &= \frac{K_{xx[i,j,k]}(\Delta z_k)^2}{K_{zz[i,j,k]}(\Delta x_j)^2} + \frac{K_{yy[i,j,k]}(\Delta z_k)^2}{K_{zz[i,j,k]}(\Delta y_i)^2} \tag{23d}
\end{align*}
\]
Equations 23 are the same ones used to calculate the minimum parameter for the iterative alternating-direction implicit procedure (ADI) and are based on a von Neumann error analysis of the normalized ADI equations. (See Weinstein and others, 1969, for references.) Commonly 4 to 10 parameters are used in one cycle and are arranged geometrically between minimum and maximum with the equation

$$\omega_{\lambda+1} = 1 + (1 - \omega_{\text{max}})^{2/L-1}, \; \lambda = \eta, 1, \ldots, L-1$$  \hspace{1cm} (24)$$

in which $\lambda$ is the iteration parameter index;

$L$ is the number of parameters in a cycle.

In the model, equation 23b, 23c, and 23d are modified to use $B, D, F, H, S$ and $Z$ defined by equations 8. For example, equation 23b is computed as

$$\rho_1 = \frac{\text{Max} \ [B,H]}{\text{Min} \ [D,F]} + \frac{\text{Max} \ [S,Z]}{\text{Min} \ [D,F]}$$

The sequence of parameters computed with equations 23 and 24 gives rapid convergence for most anisotropic problems ($K_{xx} \approx K_{yy} >> K_{zz}$). For problems with isotropic layers, however, the parameters, computed with these equations may give slower convergence. Based on an analysis of the contribution of terms in equations 23, the second terms of equations 23 were dropped and the resulting sequence of parameters generally give good convergence rates for both isotropic and anisotropic simulations. However, if the sequence of parameters computed by the equations in the program are all (except the first parameter) close to 1.0 and if this results in slow convergence or even divergence, bypass the computations in the model and insert $W_{\text{MAX}} = 0.99863$. The resulting sequence of parameters (such as 0.0, 0.80772, 0.96303, 0.99289, 0.99863) may give a satisfactory rate of convergence.
Boundaries in the numerical scheme

Within the limits of the finite-difference grid the irregular geometry of the porous medium is simulated by assigning zero transmissivity to blocks outside the system. These blocks, however, must be included in the SIP algorithm in order to preserve the required seven-diagonal coefficient matrix $\bar{A}$. Constant-head boundaries are treated without using special conditional statements by skipping constant-head nodes in the algorithm.
Initial Conditions

In many simulations, the important results are not the computed head, but the changes in head caused by a stress, such as pumping of wells. For this objective, if the flow equations for every layer are linear (that is, all hydraulic units are confined), the computed drawdown can be superimposed on the natural flow system. Consequently, the initial head for the simulation may be horizontal and there is no need to impose a natural flow system as the initial condition. However, if the user wishes to start a simulation using the head distribution in the natural flow system as the initial surface, a steady state simulation can be made with the model to compute the initial surface. This type of simulation is also useful to check the transmissivity and leakance distribution in the model.

If initial conditions are specified so that transient flow is occurring in the system at the start of the simulation, it should be recognized that the water levels will change during the simulation, not only in response to the pumping stress, but also due to the initial conditions. Initial conditions of this type are appropriate if the simulation is started part way through the history of development of the aquifer system.

Treatment of boundary conditions

Boundaries that can be treated by the model are of two types: constant head and constant flux. Constant-head boundaries are specified by assigning a negative storage coefficient to the nodes that define the constant-head boundary. A constant flux may be zero (impermeable boundaries) or have a finite value. A zero-flux boundary is treated by assigning a zero transmissivity to nodes outside the boundary. The harmonic mean of the transmissivity at the cell boundary is zero, and consequently, the flux across
the boundary is zero. A no-flow boundary is inserted around the border of
each layer of the model as a computational expediency, and constant head or
finite-flux boundaries are placed inside this border. A finite-flux
boundary is treated by assigning recharge wells to the appropriate nodes.

Treatment of Confining Layers

A confining bed in which storage is insignificant may be simulated in
either of two ways: 1) It may be represented by one or more layers of nodes
as illustrated in figure 9a with $TK(I,J,K)$ computed in the program by
equation 26c (see Appendix II). This approach is necessary if horizontal
components of flow in the confining bed are significant 2) If horizontal
components of flow in the confining bed can be ignored, the system can be
simulated in the same manner as the quasi three-dimensional model described
by Bredehoeft and Pinder (1970) with a savings in computer time and storage.
For this approach the confining beds are not represented by layers of nodes
(figure 9b). Instead, the effects of vertical flow through the confining
bed are incorporated in the vertical components of hydraulic conductivity
of the adjacent aquifers. In practice, this is accomplished by reading the
TK values, which are normally computed by equation 26c, with the rest of the
data input. ($K'-1$ sets of TK values must be read). The TK values are equal
to the ratio $K_{zz}/b$ for each confining bed and $\Delta z = 1.0$ for all layers.
This approach is designed to be used in solving equation 4. However, for certain
simple problems in which the confining bed is horizontal, TK values can be
read and nodes representing the confining bed eliminated when the model is used
to solve equation 3 if the appropriate $\Delta z$ values are retained in the data input.

If storage in the confining bed is significant, a number of layers of
nodes will be required to give a good approximation to the gradients near the
boundaries of the confining layer. To reduce computation time and storage
Figure 9.—Simulating two aquifers separated by a confining bed with three layers of nodes (a) or two layers of nodes (b).
requirements, an analytic approximation analogous to that used in the simulator for flow in two dimensions (Bredehoeft and Pinder, 1970, P.C. Trescott, G.F. Pinder and S.P. Larson, written communication, 1975) may be more suitable.

**Designing the finite-difference grid**

In designing a finite-difference grid, the following considerations should be kept in mind:

1. Nodes representing pumping and observation wells should be close to their relative positions to facilitate calibration. If several pumping wells are close together, their discharge may be lumped, and assigned to one node since discharge is distributed over the volume of the block.

2. Boundaries within the project area should be located accurately. Distant boundaries can be located approximately and with fewer nodes by expanding the grid. In expanding a finite-difference grid in the positive J direction, experience has shown that restricting the ratio \( \Delta x_j / \Delta x_{j-1} \leq 1.5 \) will avoid large truncation errors and possible convergence problems. This rule applies in all three dimensions. If the quasi three-dimensional approach is used and TK values are used with the data input, set \( \Delta z = 1.0 \) for all layers.

3. Place nodes closer together in areas of rapidly changing transmissivity in each layer.

4. The grid should be oriented so that a minimum of nodes are outside the porous medium. Orienting the grid with respect to latitude and longitude or some other geographic grid system is a secondary consideration. However, if the aquifer is anisotropic, orient the grid with its axes parallel to the
principal directions of the transmissivity tensor. Otherwise, the flow equation would include cross-product terms which are not considered in the SIP algorithm.

5. Number the rows in the short dimension for plotting maps on the line printer or for plotting data with an X-Y drum plotter. On these plots, the X direction is vertical and, for practical purposes, this dimension is unlimited. The Y direction is across the page which limits this dimension to the maximum width of the page. (See figure 10).

6. The core requirements and computation time are proportional to the number of nodes representing the porous medium.
REFERENCES


APPENDIX I

NOTATION

a,b,c,d elements of lower triangular factor \( \mathbf{L} \);

A,B;C,D;E;F;G,
H;S;T,U,W,Z' coefficients of \( \mathbf{A} + \mathbf{B} \);

\( \mathbf{A} \) coefficient matrix;

b saturated thickness of a hydraulic unit (L);

B,D,E,F,H,S,Z coefficients in difference equation;

\( \mathbf{B} \) modifying matrix;

e,f,g elements of upper triangular factor, \( \mathbf{U} \);

h hydraulic head (L);

\( \hat{h} \) hydraulic head at the previous time step (L);

i index in the y dimension;

I number of rows;

j index in the x dimension;

J number of columns;

k index in the z dimension;

K' number of layers;

K hydraulic conductivity (LT\(^{-1}\));

\( K_{xx}, K_{yy}, K_{zz} \) principal components of the hydraulic conductivity tensor (LT\(^{-1}\));

\( \ell \) iteration parameter index;

L number of parameters in a cycle;

\( \mathbf{L} \) lower triangular factor of \( \mathbf{A} + \mathbf{B} \);

n iteration index;

N_a number of arrays required for the options;
q_{re}  \text{ volumetric flux per unit area of the uppermost hydraulic unit (LT^{-1})};

Q  \text{ known term in difference equation};

Q_w  \text{ well discharge (L}^3\text{T}^{-1});

R^{n-1}  \text{ residual for previous iteration};

S'  \text{ storage coefficient (dimensionless)};

S_s  \text{ specific storage (L}^{-1});

T_{xx}, T_{yy}  \text{ principal components of the transmissivity tensor (L}^2\text{T}^{-1});

\bar{U}  \text{ upper triangular factor of (A+B)};

\bar{V}  \text{ intermediate vector in SIP algorithm};

W  \text{ volumetric flux per unit volume (T}^{-1});

\Delta t  \text{ time increment (T)};

\Delta x  \text{ space increment in the x direction (L)};

\Delta y  \text{ space increment in the y direction (L)};

\Delta z  \text{ space increment in the z direction (L)};

\varepsilon  \text{ closure criterion};

\xi  \text{ vector of change in head for an iteration};

\rho_1, \rho_2, \rho_3  \text{ hydraulic conductivity ratios};

\omega  \text{ iteration parameter}. 

I-2
APPENDIX II
COMPUTER PROGRAM

Main Program

The main program first assigns storage space to arrays required in the simulation. Storage space is reserved in a vector \( Y \) and is allocated to the arrays based on the dimensions of the problem specified on card 3 of the data deck. (See Appendix III). The minimum dimension of \( Y \) is approximately

\[
Y_{\text{DIM}} = 15 \text{ IJK} + N_a \text{ IJ}.
\]  

(25)

in which \( N_a \) is the sum of arrays required for options (see table 1).

<table>
<thead>
<tr>
<th>Option</th>
<th>Number of arrays</th>
</tr>
</thead>
<tbody>
<tr>
<td>Water-table conditions in upper unit</td>
<td>2</td>
</tr>
<tr>
<td>Recharge from precipitation</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 1.—Additional arrays required for options

Equation 25 is approximate, but normally will give a value that is sufficient for the simulation. The exact dimension required is printed on the first page of the output as "WORDS OF VECTOR Y USED = XXXX". For new simulations, the program does not need to be recompiled as long as the dimension of \( Y \) is large enough and FORTRAN statements do not need to be modified.

After allocation of storage space to the arrays, the addresses of the first element of each array are passed to the subroutines. (See table 2 for details). In table 2 the variables giving the dimensions of the arrays are defined in Appendix V; the first array is the only double precision array.

The remainder of the main program controls the sequence of computations.

Subroutine DATAI

Instructions for the preparation of the data deck are given in Appendix III. Data may be input to the model in any consistent set of units in which second is the time unit. It is organized into four groups: Data in groups I and II
Table 2.--Arrays passed to the subroutines and their relative location in the vector $Y$.

<table>
<thead>
<tr>
<th>Array</th>
<th>Sequence number in vector $Y$</th>
<th>Subroutine</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHI</td>
<td>1</td>
<td>DATAI, STEP, SOLVEI, COEF, CHECKI, PRNTAI</td>
<td>(IO, JO, KO)*8</td>
</tr>
<tr>
<td>STRT</td>
<td>2</td>
<td>X, X, X, X, X, X</td>
<td></td>
</tr>
<tr>
<td>OLD</td>
<td>3</td>
<td>X, X, X, X, X, X</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>4</td>
<td>X, X, X, X, X, X</td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>5</td>
<td>X, X, X, X, X, X</td>
<td></td>
</tr>
<tr>
<td>TR</td>
<td>6</td>
<td>X, X, X, X, X, X</td>
<td></td>
</tr>
<tr>
<td>TC</td>
<td>7</td>
<td>X, X, X, X, X, X</td>
<td></td>
</tr>
<tr>
<td>TK</td>
<td>8</td>
<td>X, X, X, X, X, X</td>
<td></td>
</tr>
<tr>
<td>WELL</td>
<td>9</td>
<td>X, X, X, X, X, X</td>
<td></td>
</tr>
<tr>
<td>EL</td>
<td>10</td>
<td>X, X, X, X, X, X</td>
<td></td>
</tr>
<tr>
<td>FL</td>
<td>11</td>
<td>X, X, X, X, X, X</td>
<td></td>
</tr>
<tr>
<td>GL</td>
<td>12</td>
<td>X, X, X, X, X, X</td>
<td></td>
</tr>
<tr>
<td>V</td>
<td>13</td>
<td>X, X, X, X, X, X</td>
<td></td>
</tr>
<tr>
<td>XI</td>
<td>14</td>
<td>X, X, X, X, X, X</td>
<td></td>
</tr>
<tr>
<td>DELX</td>
<td>15</td>
<td>X, X, X, X, X, X</td>
<td></td>
</tr>
<tr>
<td>DELY</td>
<td>16</td>
<td>X, X, X, X, X, X</td>
<td></td>
</tr>
<tr>
<td>DELZ</td>
<td>17</td>
<td>X, X, X, X, X, X</td>
<td></td>
</tr>
<tr>
<td>DDN</td>
<td>18</td>
<td>X, X, X, X, X, X</td>
<td></td>
</tr>
<tr>
<td>FACT</td>
<td>19</td>
<td>X, X, X, X, X, X</td>
<td></td>
</tr>
<tr>
<td>TEST3</td>
<td>20</td>
<td>X, X, X, X, X, X</td>
<td></td>
</tr>
<tr>
<td>JFLO</td>
<td>21</td>
<td>X, X, X, X, X, X</td>
<td></td>
</tr>
<tr>
<td>FLOW</td>
<td>22</td>
<td>X, X, X, X, X, X</td>
<td></td>
</tr>
<tr>
<td>PERM</td>
<td>23</td>
<td>X, X, X, X, X, X</td>
<td></td>
</tr>
<tr>
<td>BOTTOM</td>
<td>24</td>
<td>X, X, X, X, X, X</td>
<td></td>
</tr>
<tr>
<td>QRE</td>
<td>25</td>
<td>X, X, X, X, X, X</td>
<td></td>
</tr>
</tbody>
</table>

II-2
are the simulation options and scalar parameters; group III cards are used to initialize the arrays. These three groups are required for each new simulation. Group IV contains data that varies with each new pumping period. The program permits changing well discharge and the time parameters each pumping period, but the program can be modified to read other data with this set of cards.

If the simulation is a continuation of a previous run, the initial head for the continuation (not the starting head for the run) is read either from cards or from disk depending on how the results of the previous run were saved. (See the section on technical information for details of the data set on disk.) A three-dimensional simulation may require a large number of data cards. To reduce the number of cards that must be read with each run, the program includes options to place the arrays on disk and, on subsequent runs, read the data from disk rather than from cards. The data requirements for this option are included in Appendix III and definition of the direct access file on disk is explained in the section on technical information.

**Time Parameters.** The time parameters include the initial time step, DELT; a multiplication factor for increasing the size of the time step, CDLT; the number of time steps; NUMT; and the simulation period, TMAX. Since the rate of water-level change decreases during a pumping period, the time step is increased by the factor CDLT each step (commonly 1.5). For any time step (k) the time increment is given by

\[ \text{DELT}_k = \text{CDLT} \times \text{DELT}_{k-1} \]

\( \text{DELT}_0 \) is the time step recorded on the data card.

The program has two options for selecting the time parameters:

1. To simulate a given period of time, select CDLT and an appropriate \( \text{DELT}_0 \), and set NUMT greater than the expected number of time steps. The program computes the required initial \( \text{DELT}_0 \) (which will not exceed the value of \( \text{DELT}_0 \) coded on card 1 of group IV) and NUMT to arrive exactly at TMAX on the final step.
2. To simulate a given number of time steps, set TMAX greater than the expected simulation period and the program will use DELT, CDLT and NUMT as specified on the time parameter card.

To minimize the error due to approximation of the time derivative, several time steps should be simulated before the first step at which results are displayed. This suggestion should be followed unless the system is nearly at steady state before the results are needed. In this case a one-step simulation may be satisfactory but this approach should be checked by making one run as a multi-step simulation so that the results can be compared.

For steady-state simulations, set the storage coefficient of each layer to zero. Compute for one time step of any length (for example, set TMAX = 1, NUMT = 1, CDLT = 1, DELT = 24) and the program should iterate to a solution. The maximum permitted number of iterations (ITMAX) should be larger for steady-state than for transient simulations.

Initialization. In addition to reading data and computing the time parameters this routine initializes other arrays and scalar parameters. In particular, note that the division of well discharge by the area of the block needs to be done only once for each pumping period.

Subroutine STEP

Subroutine STEP initializes variables for a new time step and controls the printing and punching of results for designated time steps. In this routine, the computed head values, cumulative time and cumulative values for the mass balance parameters are punched on cards if PUN2 is specified in the options in group I of data input or are written on a previously defined data set on disk if DK2 is specified in the options. (See the section on technical information for details about the data set on disk).
Subroutine SOLVE

This subroutine computes the SIP iteration parameters and has the SIP normal and reverse algorithms which are outlined in the section on numerical solution. The algorithm is shortened for computation on the lowermost and uppermost layers.

In this routine the usual (I,J,K) notation has been replaced in favor of single-subscript notation. Less time is involved in finding the value of a variable with a single subscript than in finding the value of one with three subscripts and as a consequence computational efficiency is improved. The subscripts used in this notation are defined in the code.

Computational efficiency is also improved by including the computation of the term $S'/\Delta t$ and the coefficients $B,D,F,H,S,$ and $Z$ in this routine rather than in COEF.

If the permitted number of iterations for the time step is exceeded, the message "EXCEEDED PERMITTED NUMBER OF ITERATIONS" is printed. Following the message, the mass balance, head matrix and other data specified in the options are printed. This information is useful in determining the cause of non convergence. The mass-balance parameter values and head values for the last iteration are punched if PUNC is specified in the option or written on disk if DK2 is specified. These results can be used to extend the simulation if it appears that a solution can be obtained.
Subroutine COEF

If the upper hydraulic unit is under water-table conditions, its transmissivity as a function of saturated thickness is recomputed every iteration. For those nodes where the computed head drops below the bottom elevation of the unit, a message 'NODE I, J IN LAYER K GOES DRY' is printed. A similar message is printed if a well node goes dry. The hydraulic conductivity at these nodes is set to zero and computation continues. No provision is made to permit these blocks to resaturate because the additional code for this special situation is not warranted in a general program.

The T coefficients TR, TC and TK may be computed once and saved for artesian units; for water-table units, TR and TC are recomputed every iteration and TK stays constant until block K+1 desaturates, then TK(I,J,K) = 0.0. They are defined as

\[ TR(I,J,K) = \frac{2T_{xx[i+1]} T_{xx}}{T_{xx} \Delta x_{i+1} + T_{xx[j+1]} \Delta x}; \quad (26a) \]

\[ TC(I,J,K) = \frac{2T_{yy[i+1]} T_{yy}}{T_{yy} \Delta y_{i+1} + T_{yy[j+1]} \Delta y}; \quad (26b) \]

\[ TK(I,J,K) = \frac{2(bK_{zz}^k + (bK_{zz}^k) \Delta z_{k+1}}{(bK_{zz}^k) \Delta z_{k+1} + (bK_{zz}^k) \Delta z}. \quad (26c) \]
Subroutine CHECK

A mass balance is computed in this routine. The results are expressed in two ways: 1) as a cumulative volume of water from each source and each type of discharge, and 2) as rates for the current time step. Note that 'leakage' and 'evapotranspiration' appear in the mass balance but are set to zero because these options are not included in the current version of the model.

In the cumulative mass balance, storage is treated as a source of water. Flow to and from constant-head boundaries is computed with Darcy's law using the gradients from constant-head nodes to adjacent nodes inside the porous medium. Other computations in the algorithm are self-explanatory. The difference between the sum of recharge sources and sum of discharges from the system is usually less than one percent.

To the right of the cumulative mass balance are printed the flow rates for the current time step. Below the cumulative mass balance are printed the flow rates to individual constant head nodes (if included in the simulation). This is followed by the net flow rate to the top layer from below and the net flow rate to the bottom layer from above.

The last item is the listing of the absolute value of the maximum head change for each iteration. This information is useful if convergence is slow with SIP because it may indicate that a slightly larger error criterion will give a satisfactory solution with considerably fewer iterations.

Subroutine PRNTAI

This routine prints maps of drawdown and(or) hydraulic head for selected layers. Up to three characters are plotted for each block with the right-most character as close to the location of the node as the printer will allow. The user specifies XSSCALE and YSCALE, the multiplication factors required to change from units used in the model to units used in the map (in general they
should be the same); DINCH, the number of map units per inch; FACT1 and FACT2, the multiplication factors for adjusting the values of drawdown and head to be plotted, respectively; MESUR, the name of the unit used on the map, and vectors LEVEL1 and LEVEL2 which contain the layers for which maps are to be printed. As an example, assume that the length unit used in the model is feet, the map is to be scaled at 3 miles per inch and drawdown values to the nearest foot and head values to the nearest 10 feet are to be plotted for layers 1 and 3. Then XSCALE = YSCALE = 5280, DINCH = 3, FACT1 = 1; FACT2 = 0.1; MESUR = MILES; LEVEL1 (1) = LEVEL2 (1) = 1; and LEVEL1(2) = LEVEL2(2) = 3.

To print a map of maximum possible size, number the rows in the short dimension to take advantage of the orientation of the map on the computer page where the X direction is vertical and the Y direction horizontal. (See figure 10). The origin is the upper left-hand corner of the block for row 2, column 2. Orienting the map with the origin in the upper left hand corner, the right and bottom sides of the map include the node locations for the second to last column and row, respectively. The border is located to the nearest inch outside these node locations and may or may not fall on the block boundaries depending on the scaling. The map is automatically centered on the page and is limited to a maximum of 12 inches (30 cm) in the Y direction. If the parameters for a map are specified such that the Y dimension is more than 12 inches (30 cm) adjustments are automatically made to fit the map within this limit. A common mistake is to specify a value for Y scale that is less than 1.0. This generates the message 'NOTE: GENERALLY SCALE SHOULD BE > OR = 1.0' and a suitable adjustment is made to DINCH. In the X direction the map is limited only by the dimension of the NX vector (for example, when the dimension of NX is 100, the map is limited in the X direction to 100 - 1 = 99 inches [250 cm]). Several parameters (PRNT, BLANK, N1, N2, N3, and XN1) are initialized in the BLOCK DATA routine to values that assume the line printer
Figure 10.—Orientation of maps on computer page.
prints 6 lines per inch, 10 characters per inch, and 132 characters per line. These parameter values may need to be changed for a line printer with other specifications.

The PRNTAI subroutine can be modified to cycle a set of alphanumerical symbols for drawdown. If this type of map is desired, remove the C from column 1 of statements PRN1060 and PRN1230. This will cycle the symbols 1,2,3,4,5,6,7,8,9,0 for drawdown. To plot a different set of symbols will require modification of the initialization of SYM in BLOCK DATA. To cycle more than 10 symbols will require more extensive changes to the initialization of SYM and modifications to the code in ENTRY PRNTA.

**BLOCK DATA**

The BLOCK DATA routine initializes scalar parameters and arrays used in PRNTAI and other subroutines.

**Technical Information**

*Use of disk facilities for storage of array data and interim results.*

Options are included in the program to enable storage and retrieval of array data (STRT, S, T, etc.) and the saving of interim head values without punching them on cards. Use of these options can be particularly beneficial at remote terminals with low speed data transmission or without punch output capability. Also, the type of read statements used afford more efficient data transmission from disk than from cards.

*Storage of array data* is accomplished via a direct access data set that is defined by a DEFINE FILE statement in the main program (card MAN 0310) and by a DD statement in the JCL string used to execute the program. To establish the data set, the DEFINE FILE statement and the DD statement must indicate the amount of space that is required. The DEFINE FILE statement takes the following form:
DEFINE FILE 2\{XX,???,U,FK}\{MAN0310

where ???? is the number of nodes per layer for the problem being solved (IO * JO) and XX is the number of records to be reserved for data set storage. The value of XX is 3 times KO, plus KO-1 if TK values are assigned, plus 2 for the water table option, plus 1 for the recharge option. Parameters U and KKK are indicators and do not vary.

The DD statement contains information, such as account number, that will be different for each user. Also, the first reference to the data set is somewhat different from subsequent references. To utilize one of the disk packs for semi-permanent storage of user data, the first reference to the data set will take the following general form if the FORTGCG procedure is used to compile and execute the program.

//GO.FT02FOO1 DD DSN=Data Set Name,
//UNIT=ONLINE,DISP={NEW,KEEP},
//SPACE={????,XX},DCB={RECFM=F}

where ???? is the number of bytes per record that are to be reserved and should be set equal to IO * JO [*4].

When this initial allocation is processed the system will indicate in the HASP system log, JCL string output, the volume on which the data set was established (for example, SYS011 or SYS015). Subsequent use of the data set must indicate this information by modifying the underlined parameters in the initial reference to the data set. Thus the DD statement will read:

//GO.FT02FOO1 DD DSN=Data Set Name,
//UNIT=ONLINE,DISP=SHR, VOL=SER=YYYYYY
where the DSN parameter is the same as the initial run and YYYYYY indicates the volume (for example, SYS015) on which the data set was established by the initial run. The individual data arrays that are to be stored and later retrieved from this data set are specified on the parameter card for each array. These specifications are discussed completely in Appendix III.

Interim results. The initial head, cumulative simulation time, and mass balance parameters may be read from a data set on disk if DK1 is specified in the options. These data are the results of a previous simulation run and were written on disk because DK2 was specified in the previous run. They permit intermediate results to be examined before extending the simulation period. The same procedure also applies in simulations where the permitted number of iterations is exceeded.

The unit number for the data set is 4; an example of the JCL required to generate space for the data set on a disk is given below:

```
//STEP1 EXEC PGM=IEFBR14
//SYSUT1 DD DSN=Data Name Set-, VOL=SER=Disk Name-,UNIT=3330,DISP={NEW,KEEP},
// SPACE={TRK,{5,1}},DCB={RECFM=VBS,BKSIZE=13030}
```

Note that with the data set defined this way an extra 28K bytes of storage are required for the two buffers. If the block size is reduced, the storage required for the buffers is reduced correspondingly. If a procedure such as FORTGCG is used, insert a card analogous to

```
//GO.FT04F001 DD UNIT=3330,VOL=Disk Name,DISP=SHR,DSN=Data Set Name
```

before the //GO.SYSIN DD * card to define unit 4.
JCL to execute a compiled program which includes unit 4 follows:

//Job card
//A EXEC PGM= Program Name, REGION=410K, TIME=3
//STEPLIB DD DSN='Data Set Name', VOL=SER=Disk Name, UNIT=3330, DISP=SHR
//FT04F001 DD UNIT=3330, VOL=SER=Disk Name, DISP=SHR, DSN=Data Set Name
//FT06F001 DD SYSOUT=A
//FT07F001 DD SYSOUT=B
//FT05F001 DD *

DATA
/*

Unless the program is modified, writing on unit 4 destroys data previously written on this unit.

Storage Requirements. The amount of core that is needed can be found in table 3.

<table>
<thead>
<tr>
<th>Item</th>
<th>K bytes required</th>
</tr>
</thead>
<tbody>
<tr>
<td>For compilation</td>
<td></td>
</tr>
<tr>
<td>FORTRAN G, LEVEL 21</td>
<td>120</td>
</tr>
<tr>
<td>FORTRAN H, OPT=2</td>
<td>250</td>
</tr>
<tr>
<td>For execution</td>
<td></td>
</tr>
<tr>
<td>FORTRAN G, LEVEL 21</td>
<td>88</td>
</tr>
<tr>
<td>FORTRAN H, OPT=2</td>
<td>72</td>
</tr>
<tr>
<td>BUFFERS</td>
<td></td>
</tr>
<tr>
<td>Unit 4</td>
<td>28</td>
</tr>
<tr>
<td>Unit 2</td>
<td>((IO<em>JO</em>4+Z^)/1000)*2</td>
</tr>
<tr>
<td>Vector Y</td>
<td>X^^/256</td>
</tr>
</tbody>
</table>

\^ Round IO*JO*4 up to the nearest 2000 bytes (for example, if IO*JO*4=6400, let Z=1600 and the buffer space for Unit 2 will be 16K).

\^\^ X is the dimension of Y in the main program.
Computation time. Computation time is a function of many variables but as an approximation, about 0.001 second of CPU time on the IBM 370/155 is required for each interior node each iteration using the FORTRAN H, OPT = 2 compiler. For example, a steady-state problem designed by Weeks and others (1974) with 2116 interior nodes requiring 25 iterations to satisfy the error criterion took 51 seconds of CPU time.

Using a source code generated with the FORTRAN G, LEVEL 21 compiler, the computation time is increased by about one half.

FORTRAN IV. The program includes several FORTRAN IV features that are not in ANS FORTRAN (for example, ENTRY, END parameter in read statement, mixed-mode expressions, G format code, literal enclosed in apostrophes). If the program is used at a computer center where the FORTRAN compiler does not include these extensions, programmers at the selected installations may be available to modify the computer code as necessary.

Program Limitations

The model documented in this report was motivated primarily by a need to find a substitute for Bredehoeft and Pinder's (1970) quasi three-dimensional model. Consequently, the model should be reasonably free of errors for this type of simulation. The model has not been tested on all types of simulations in which equation 3 is being solved. Consequently, undiscovered errors in the logic may appear as the model is applied to new problems. I would appreciate hearing about "bugs" in the logic so that corrections can be made and other users informed.

The finite-difference model for simulation of ground-water flow in two dimensions (P.C. Trescott, G.F. Pinder and S.P. Larson, written communication, 1975) includes evapotranspiration and an approximation to transient effects
of confining beds in the source term, and permits an aquifer to change from
artesian to water-table conditions. Some of these features (particularly
evapotranspiration) can be added to the three-dimensional model with only
moderate changes to the code.
APPENDIX III

DATA DECK INSTRUCTIONS

Group I: Title, Simulation Options and Problem Dimensions

This group of cards, which are read by the main program, contains data required to dimension the model. To specify an option on card 4 punch the characters underlined in the definition. For an option not used, that section of the card 4 can be left blank.

Note: Default typing of variables applies for all data input.

<table>
<thead>
<tr>
<th>CARD</th>
<th>COLUMNS</th>
<th>FORMAT</th>
<th>VARIABLE</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1-80</td>
<td>20A4</td>
<td>HEADING</td>
<td>Any title the user wishes to print on one line at the start of output.</td>
</tr>
<tr>
<td>2</td>
<td>1-52</td>
<td>I3A4</td>
<td></td>
<td>&quot;</td>
</tr>
<tr>
<td>3</td>
<td>1-10</td>
<td>I10</td>
<td>IO</td>
<td>Number of rows</td>
</tr>
<tr>
<td></td>
<td>11-20</td>
<td>I10</td>
<td>JO</td>
<td>Number of columns</td>
</tr>
<tr>
<td></td>
<td>21-30</td>
<td>I10</td>
<td>KO</td>
<td>Number of layers</td>
</tr>
<tr>
<td></td>
<td>31-40</td>
<td>I10</td>
<td>ITMAX</td>
<td>Maximum number of iterations per time step</td>
</tr>
<tr>
<td>4</td>
<td>1-4</td>
<td>A4</td>
<td>IDRAW</td>
<td>DRAW to print drawdown</td>
</tr>
<tr>
<td></td>
<td>6-9</td>
<td>A4</td>
<td>IHEAD</td>
<td>HEAD to print hydraulic head</td>
</tr>
<tr>
<td></td>
<td>11-14</td>
<td>A4</td>
<td>IFLOW</td>
<td>MASS to compute a mass balance</td>
</tr>
<tr>
<td></td>
<td>16-18</td>
<td>A3</td>
<td>IDK1</td>
<td>DK1 to read initial head, elapsed time, and mass balance parameters from unit 4 on disk</td>
</tr>
<tr>
<td></td>
<td>21-23</td>
<td>A3</td>
<td>IDK2</td>
<td>DK2 to write computed head, elapsed time, and mass balance parameters on unit 4 (disk)</td>
</tr>
<tr>
<td></td>
<td>26-29</td>
<td>A4</td>
<td>IWATER</td>
<td>WATE if the upper hydrologic unit is unconfined</td>
</tr>
<tr>
<td></td>
<td>31-34</td>
<td>A4</td>
<td>IQRE</td>
<td>RECH for a constant recharge that may be a function of space</td>
</tr>
</tbody>
</table>

III-1
<table>
<thead>
<tr>
<th>CARD</th>
<th>COLUMNS</th>
<th>FORMAT</th>
<th>VARIABLE</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>36-39</td>
<td>A4</td>
<td>IPU1</td>
<td>PUN1 to read initial head, elapsed time, and mass balance parameters from cards</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IPU2</td>
<td>PUN2 to punch computed head, elapsed time, and mass balance parameters on cards</td>
</tr>
<tr>
<td>46-49</td>
<td>A4</td>
<td></td>
<td>ITK</td>
<td>ITKR to read the value of TK(I,J,K) for simulations in which confining layers are not represented by layers of nodes. TK(I,J,K) = $K_{zz}/b$.</td>
</tr>
</tbody>
</table>
Group II: Scalar parameters

The parameters required in every problem are underlined. The other parameters are required as noted; when not required, their location on the card can be left blank. The G format is used to read E, F and I format data. Minimize mistakes by always right-justifying data in the field. If F format data do not contain significant figures to the right of the decimal point, the decimal point can be omitted.

<table>
<thead>
<tr>
<th>CARD</th>
<th>COLUMNS</th>
<th>FORMAT</th>
<th>VARIABLE</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1-10</td>
<td>G10.0</td>
<td>NPER</td>
<td>Number of pumping periods for the simulation</td>
</tr>
<tr>
<td></td>
<td>11-20</td>
<td>G10.0</td>
<td>KTH</td>
<td>Number of time steps between printouts</td>
</tr>
</tbody>
</table>

Note: To print only the results for the final time step in a pumping period, make KTH greater than the expected number of time steps. The program always prints the results for the final time step.

| 21-30 | G10.0  | ERR    | Error criteria for closure (L) |

Note: When the head change at all nodes on subsequent iterations is less than this value (for example, 0.01 foot), the program has converged to a solution for the time step.

| 31-40 | G10.0  | LENGTH | Number of iteration parameters |
| 1-10  | G10.0  | XSCALE | Factor to convert model length unit to unit used in X direction on maps (e.g. to convert from feet to miles, XSCALE = 5280). |

For no maps, card 2 is blank

| 11-20 | G10.0  | YSCALE | Factor to convert model length unit to unit used in Y direction on maps. |

| 21-30 | G10.0  | DINCH  | Number of map units per inch |

| 31-40 | G10.0  | FACT1  | Factor to adjust value of drawdown printed* |

| 41-49 | 9| LEVEL1(I) | Layers for which drawdown maps are to be printed. List the layers starting in column 41; the first zero entry terminates the printing of drawdown maps. |
### CARD COLUMNS

<table>
<thead>
<tr>
<th>CARD</th>
<th>COLUMNS</th>
<th>FORMAT</th>
<th>VARIABLE</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>51-60</td>
<td>G10.0</td>
<td>FACT2</td>
<td>Factor to adjust value of head printed*</td>
</tr>
<tr>
<td></td>
<td>61-69</td>
<td>9I1</td>
<td>LEVEL2(I)</td>
<td>Layers for which head maps are to be printed. List layers starting in column 61; the first zero entry terminates the printing of head maps.</td>
</tr>
<tr>
<td>71-78</td>
<td></td>
<td>A8</td>
<td>MESUR</td>
<td>Name of map length unit.</td>
</tr>
</tbody>
</table>

| 3    | 1-20    | G20.10 | SUM      | Parameters in which elapsed time and cumulative volumes for mass balance are stored. For the start of a simulation insert three blank cards. For continuation of a previous run using cards as input, replace the three blank cards with the first three cards of punched output from the previous run. Using data from disk for input, leave the three blank cards in the data deck. |
|      | 21-40   | G20.10 | SUMP     | |
|      | 41-60   | G20.10 | PUMPT    | |
|      | 61-80   | G20.10 | CFLUXT   | |

| 4    | 1-20    | G20.10 | QRET     | |
|      | 21-40   | G20.10 | CHST     | |
|      | 41-60   | G20.10 | CHDT     | |
|      | 61-80   | G20.10 | FLUXT    | |

| 5    | 1-20    | G20.10 | STORT    | |
|      | 21-40   | G20.10 | ETFLXT   | |
|      | 41-60   | G20.10 | FLXNT    | |

*Value of drawdown or head

<table>
<thead>
<tr>
<th>FACT 1 or Printed value</th>
<th>FACT 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>1</td>
</tr>
<tr>
<td>0.1</td>
<td>5</td>
</tr>
<tr>
<td>52.57</td>
<td>53</td>
</tr>
<tr>
<td>10.0</td>
<td>526</td>
</tr>
<tr>
<td>100.0</td>
<td>***</td>
</tr>
</tbody>
</table>
Each of the following data sets (except data set 1) consists of a parameter card and, if the data set contains variable data, a set of data cards. If the data set requires data for each layer, a parameter card and data cards (for layers with variable data) are required for each layer. Each parameter card contains at least five variables:

<table>
<thead>
<tr>
<th>CARD</th>
<th>COLUMNS</th>
<th>FORMAT</th>
<th>VARIABLE</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Every Parameter Card</td>
<td>1-10</td>
<td>G10.0</td>
<td>FAC</td>
<td>If IVAR = 0, FAC is the value assigned to every element of the matrix for this layer. If IVAR = 1, FAC is the multiplication factor for the following set of data cards for this layer.</td>
</tr>
<tr>
<td>11-20</td>
<td>G10.0</td>
<td>IVAR</td>
<td></td>
<td>= 0 if no data cards are to be read for this layer. = 1 if data cards for this layer follow.</td>
</tr>
<tr>
<td>21-30</td>
<td>G10.0</td>
<td>IPRN</td>
<td></td>
<td>= 0 if input data for this layer are to be printed; = 1 if input data for the layer are not to be printed.</td>
</tr>
<tr>
<td>Transmissivity Parameter Cards also have these Variables</td>
<td>31-40</td>
<td>G10.0</td>
<td>FACT(K,1)</td>
<td>multiplication factor for transmissivity in x direction</td>
</tr>
<tr>
<td></td>
<td>41-50</td>
<td>G10.0</td>
<td>FACT(K,2)</td>
<td>multiplication factor for transmissivity in the y direction</td>
</tr>
<tr>
<td></td>
<td>51-60</td>
<td>G10.0</td>
<td>FACT(K,3)</td>
<td>multiplication factor for hydraulic conductivity in the z direction. (Not used when confining bed nodes are eliminated and TK values are read)</td>
</tr>
<tr>
<td>Every Parameter Card</td>
<td>61-70</td>
<td>G10.0</td>
<td>IRECS</td>
<td>= 0 if the matrix is being read from cards or if each element is being set equal to FAC. = 1 if the matrix is to be read from disk (unit 2)</td>
</tr>
<tr>
<td></td>
<td>71-80</td>
<td>G10.0</td>
<td>IRECD</td>
<td>= 0 if the matrix is not to be stored on disk. = 1 if the matrix being read from cards or set equal to FAC is to be stored on disk (unit 2) for later retrieval.</td>
</tr>
</tbody>
</table>
When data cards are included, start each row on a new card. To prepare a set of data cards for an array that is a function of space, the general procedure is to overlay the finite-difference grid on a contoured map of the parameter and record the average value of the parameter for each finite-difference block on coding forms according to the appropriate format. In general, record only significant digits and no decimal points (except for data set 2); use the multiplication factor to convert the data to their appropriate values. For example, if DELX ranges from 1000 to 15000 feet, coded values should range from 1-15; the multiplication factor (FAC) would be 1000.

<table>
<thead>
<tr>
<th>DATA SET</th>
<th>COLUMNS</th>
<th>FORMAT</th>
<th>VARIABLE</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1-80</td>
<td>8F 10.4</td>
<td>PHI(I,J,K)</td>
<td>Head values for continuation of a previous run (L)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: For a new simulation this data set is omitted. Do not include a parameter card with this data set.</td>
</tr>
<tr>
<td>2</td>
<td>1-80</td>
<td>8F 10.4</td>
<td>STRT(I,J,K)</td>
<td>Starting head matrix (L)</td>
</tr>
<tr>
<td>3</td>
<td>1-80</td>
<td>20F 4.0</td>
<td>S(I,J,K)</td>
<td>Storage coefficient (dimensionless)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: This matrix is also used to locate constant head boundaries by coding a negative number at constant head nodes. At these nodes T must be greater than zero. If equation 3 is to be solved, read specific storage instead of storage coefficient.</td>
</tr>
<tr>
<td>4</td>
<td>1-80</td>
<td>20F 4.0</td>
<td>T(I,J,K)</td>
<td>Transmissivity (L²/t)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note 1) Zero values are required around the perimeter of the T matrix for each layer for reasons inherent in the computational scheme. This is done automatically by the program.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2) See the previous page for the additional requirements on the parameter cards for this data set.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3) If the upper active layer is unconfined and PERM and BOTTOM are to be read for this layer, insert a parameter card for this layer with only the values for FACT on it. If equation 3 is to be solved read hydraulic conductivity instead of transmissivity.</td>
</tr>
<tr>
<td>5</td>
<td>1-80</td>
<td>20F 4.0</td>
<td>TK(I,J,K)</td>
<td>Kzz/b</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: This data set is read only if specified in the options. The number of layers of TK values = K'- 1. See the discussion of the treatment of confining layers.</td>
</tr>
<tr>
<td>6</td>
<td>1-80</td>
<td>20F 4.0</td>
<td>PERM(I,J)</td>
<td>Hydraulic conductivity (L/T) (see note 1 for data set 4)</td>
</tr>
<tr>
<td>7</td>
<td>1-80</td>
<td>20F 4.0</td>
<td>BOTTOM(I,J)</td>
<td>Elevation of bottom of water-table unit (L)</td>
</tr>
<tr>
<td>DATA SET</td>
<td>COLUMNS</td>
<td>FORMAT</td>
<td>VARIABLE</td>
<td>DEFINITION</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
<td>--------</td>
<td>----------</td>
<td>------------</td>
</tr>
<tr>
<td>8</td>
<td>1-80</td>
<td>20F 4.0</td>
<td>QRE(I,J)</td>
<td>Recharge rate (L/T)</td>
</tr>
</tbody>
</table>

Note: Data sets 6 and 7 are required only for simulating unconfined conditions in the upper hydrologic unit.

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>1-80</td>
<td>8G10.0</td>
<td>DELX(J)</td>
<td>Grid spacing in x direction (L)</td>
</tr>
<tr>
<td>10</td>
<td>1-80</td>
<td>8G10.0</td>
<td>DELY(I)</td>
<td>Grid spacing in y direction (L)</td>
</tr>
<tr>
<td>11</td>
<td>1-80</td>
<td>8G10.0</td>
<td>DELZ(K)</td>
<td>Grid spacing in z direction (L)</td>
</tr>
</tbody>
</table>

Note: Omit if not used
Group IV: Parameters that change with the pumping period

The program has two options for the simulation period:

1. To simulate a given number of time steps, set TMAX to a value larger than the expected simulation period. The program will use NUMT, CDLT, and BELT as coded. If NUMT is greater than 50 change the dimension of ITT0 in subroutine STEP to the appropriate size.

2. To simulate a given pumping period, set NUMT larger than the number required for the simulation period (for example, 50). The program will compute the exact DELT (which will be < DELT coded) and NUMT to arrive exactly at TMAX on the last time step.

---

<table>
<thead>
<tr>
<th>CARD</th>
<th>COLUMNS</th>
<th>FORMAT</th>
<th>VARIABLE</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1-10</td>
<td>G10.0</td>
<td>KP</td>
<td>Number of the pumping period</td>
</tr>
<tr>
<td></td>
<td>11-20</td>
<td>G10.0</td>
<td>KPM1</td>
<td>Number of the previous pumping period</td>
</tr>
</tbody>
</table>

Note: KPM1 is currently not used

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>NWEL</th>
<th>Number of wells for this pumping period</th>
</tr>
</thead>
<tbody>
<tr>
<td>21-30</td>
<td>G10.0</td>
<td>NWEL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>TMAX</th>
<th>Number of days in this pumping period</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-40</td>
<td>G10.0</td>
<td>TMAX</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>NUMT</th>
<th>Number of time steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>41-50</td>
<td>G10.0</td>
<td>NUMT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>CDLT</th>
<th>Multiplying factor for DELT</th>
</tr>
</thead>
<tbody>
<tr>
<td>51-60</td>
<td>G10.0</td>
<td>CDLT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: 1.5 is commonly used

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>DELT</th>
<th>Initial time step in hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>61-70</td>
<td>G10.0</td>
<td>DELT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If NWEL: 0 the following set of cards is omitted

---

**DATA SET 1**

<table>
<thead>
<tr>
<th>COLUMNS</th>
<th>FORMAT</th>
<th>VARIABLE</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-10</td>
<td>G10.0</td>
<td>K</td>
<td>Layer in which well is located</td>
</tr>
<tr>
<td>11-20</td>
<td>G10.0</td>
<td>I</td>
<td>Row location of well</td>
</tr>
<tr>
<td>21-30</td>
<td>G10.0</td>
<td>J</td>
<td>Column location of well</td>
</tr>
<tr>
<td>31-40</td>
<td>G10.0</td>
<td>WELL(I,J,K)</td>
<td>Pumping rate (L^3/t), negative for a pumping well.</td>
</tr>
</tbody>
</table>

For each additional pumping period, another set of group IV cards is required (that is, NPER sets of group IV cards are required).

III-8
APPENDIX IV

EXAMPLE SIMULATION

The following pages illustrate the data input and results of the steady-state simulation of a hypothetical problem including two aquifers separated by a confining layer (figure 11). Boundaries are no-flow except for a constant-head boundary along the left side of the upper aquifer. The aquifers are identical except that the upper aquifer is unconfined. The confining bed is not represented by a layer of nodes because only vertical flow is simulated and this is incorporated in the equations for the two aquifers. The finite-difference grid, therefore, consists of two layers of nodes with uniform grid spacing. There are two discharging wells, one in each of the two aquifers.

Table 4 lists the data input required for this simulation. The format for data are given in Appendix III. The location of numbers on the data cards should not be difficult to determine because they are either in fields of 4 or 10 spaces. In general, zero values have not been coded.

The printout of this simulation follows Table 4 and is either self-explanatory or has been discussed in Appendix II.
Figure 11.—Schematic illustration of example problem.
TABLE 4.--Data Input For Example Problem.

<table>
<thead>
<tr>
<th>GROUP I</th>
<th>IFER HAS A FREE SURFACE</th>
<th>PUMPING FROM 2 AQUIFERS, UPPER AQUIFERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRAW</td>
<td>MASS</td>
<td>WATER RECH</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td>GROUP II</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ST:F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GROUP III</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.00000001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PERM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BOTTOM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0E-09</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QRE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1060</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELZ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GROUP IV</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
-------PUMPING FROM 2 AQUIFERS. UPPER AQUIFER HAS A FREE SURFACE-------

| Number of Rows | 20 |
| Number of Columns | 20 |
| Number of Layers | 2 |
| Maximum permitted number of iterations | 50 |
| Number of constant head nodes | 18 |

**Simulation Options**

<table>
<thead>
<tr>
<th>Mass</th>
<th>Water</th>
<th>Recm</th>
<th>Iter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Words of vector y used</td>
<td>12992</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of pumping periods used</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time steps between printouts</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error criteria for closure</td>
<td>$0.9999999E-03$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**On Alphabetic Map**

- Multiplication factor for x dimension = 1000.000
- Multiplication factor for y dimension = 1000.000
- Map scale in units of 1000 feet
- Number of 1000 feet per inch = 2.000000
- Multiplication factor for drawdown = 10.00000
- Multiplication factor for head = 0.0

<table>
<thead>
<tr>
<th>Starting head</th>
<th>For Layer 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Printed for layers</td>
<td>1 2 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>Storage coefficient</td>
<td>0.0</td>
</tr>
<tr>
<td>Printed for layers</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>For Layer 2</td>
<td>1 2 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>For Layer 1</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>----</td>
<td>-----</td>
</tr>
<tr>
<td>1</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>0.0</td>
</tr>
<tr>
<td>3</td>
<td>0.0</td>
</tr>
<tr>
<td>4</td>
<td>0.0</td>
</tr>
<tr>
<td>5</td>
<td>0.0</td>
</tr>
<tr>
<td>6</td>
<td>0.0</td>
</tr>
<tr>
<td>7</td>
<td>0.0</td>
</tr>
<tr>
<td>8</td>
<td>0.0</td>
</tr>
<tr>
<td>9</td>
<td>0.0</td>
</tr>
<tr>
<td>10</td>
<td>0.0</td>
</tr>
<tr>
<td>11</td>
<td>0.0</td>
</tr>
<tr>
<td>12</td>
<td>0.0</td>
</tr>
<tr>
<td>13</td>
<td>0.0</td>
</tr>
<tr>
<td>14</td>
<td>0.0</td>
</tr>
<tr>
<td>15</td>
<td>0.0</td>
</tr>
<tr>
<td>16</td>
<td>0.0</td>
</tr>
<tr>
<td>17</td>
<td>0.0</td>
</tr>
<tr>
<td>18</td>
<td>0.0</td>
</tr>
<tr>
<td>19</td>
<td>0.0</td>
</tr>
<tr>
<td>20</td>
<td>0.0</td>
</tr>
</tbody>
</table>
TRANSPISSIVITY = 0.9999996E-01 FOR LAYER 1

DIRECTIONAL TRANSMISSIVITY MULTIPLICATION FACTORS FOR LAYER 1
X = 1.000000
Y = 1.000000
Z = 0.0

TRANSPISSIVITY = 0.9999996E-01 FOR LAYER 2

DIRECTIONAL TRANSMISSIVITY MULTIPLICATION FACTORS FOR LAYER 2
X = 1.000000
Y = 1.000000
Z = 0.0

HYDRAULIC CONDUCTIVITY = 0.9999999E-03 FOR LAYER 2

BOTTOM ELEVATION = 0.0 FOR LAYER 2

RECHARGE RATE = 0.9999999E-08 FOR LAYER 1

DELZ = 1.000000

TK = 0.9999999E-08 FOR LAYER 2

DELX = 1000.000

DELY = 1000.000

DELZ = 1.000000

SOLUTION BY THE STRONGLY IMPLICIT PROCEDURE

FITERATION PARAMETERS:
0.0

0.70459100  0.91273350  0.9742207D 00  0.99236460  0.0

PUMPING PERIOD NO.:
1.00 DAYS

DELT IN HOURS =
24.000

MULTIPLIER FOR DELT =
1.000

2 WELLS

K = 1
J = 1
I = 10
L = 15
M = 2

PUMPING RATE =
1.00
-1.00
<table>
<thead>
<tr>
<th>TIME STEP NUMBER</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
</table>

**Size of Time Step in Seconds**: 86400.00

**Total Simulation Time in Seconds**: 86400.00

- **Minutes**: 1440.00
- **Hours**: 24.00
- **Days**: 1.00
- **Years**: 0.00

**Duration of Current Pumping Period**: 1.00

**Years**: 0.00

### Cumulative Mass Balances

<table>
<thead>
<tr>
<th>Sources</th>
<th>(L^3/T)</th>
<th>Rate for this Time Step</th>
<th>(L^3/T)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage</td>
<td>0.0</td>
<td>-</td>
<td>0.0</td>
</tr>
<tr>
<td>Recharge</td>
<td>26438.11</td>
<td>26438.11</td>
<td>-2.0000</td>
</tr>
<tr>
<td>Constant Flux</td>
<td>0.0</td>
<td>-</td>
<td>0.0</td>
</tr>
<tr>
<td>Constant Head</td>
<td>14596.88</td>
<td>14596.88</td>
<td>1.6894</td>
</tr>
<tr>
<td>Leakage</td>
<td>0.0</td>
<td>-</td>
<td>0.0</td>
</tr>
<tr>
<td>Total Sources</td>
<td>172399.94</td>
<td>172399.94</td>
<td>0.0</td>
</tr>
</tbody>
</table>

**Discharges**

- Evapotranspiration: 0.0
- Constant Head: 0.0
- Quantity Pumped: 172799.81
- Leakage: 0.0
- Total Discharge: 172799.81

**Discharge - Sources**: 399.88

**Percent Difference**: 0.23

### Flow Rates to Constant Head Nodes

<table>
<thead>
<tr>
<th>K</th>
<th>I</th>
<th>J</th>
<th>Rate (L^3/T)</th>
<th>K</th>
<th>I</th>
<th>J</th>
<th>Rate (L^3/T)</th>
<th>K</th>
<th>I</th>
<th>J</th>
<th>Rate (L^3/T)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0.8160883E-01</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>0.828751E-01</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>0.8543676E-01</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>2</td>
<td>0.9550179E-01</td>
<td>2</td>
<td>6</td>
<td>2</td>
<td>0.9468021E-01</td>
<td>2</td>
<td>7</td>
<td>2</td>
<td>0.1012863</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>2</td>
<td>0.4284997E-01</td>
<td>2</td>
<td>9</td>
<td>2</td>
<td>0.1193280</td>
<td>2</td>
<td>10</td>
<td>2</td>
<td>0.1194522</td>
</tr>
<tr>
<td>2</td>
<td>11</td>
<td>2</td>
<td>0.1147608</td>
<td>2</td>
<td>12</td>
<td>2</td>
<td>0.1075212</td>
<td>2</td>
<td>13</td>
<td>2</td>
<td>0.9957021E-01</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>2</td>
<td>0.9234345E-01</td>
<td>2</td>
<td>15</td>
<td>2</td>
<td>0.8635312E-01</td>
<td>2</td>
<td>16</td>
<td>2</td>
<td>0.8109365E-01</td>
</tr>
<tr>
<td>2</td>
<td>17</td>
<td>2</td>
<td>0.7939699E-01</td>
<td>2</td>
<td>18</td>
<td>2</td>
<td>0.7611233E-01</td>
<td>2</td>
<td>19</td>
<td>2</td>
<td>0.7503170E-01</td>
</tr>
</tbody>
</table>

**Flow to Top Layer**: -0.9975219

**Flow to Bottom Layer**: -0.9975219

**Positive Upward**

**Maximum Head Change for Each Iteration**
<table>
<thead>
<tr>
<th>TIME STEP</th>
<th>1.0</th>
<th>1.1</th>
<th>1.2</th>
<th>1.3</th>
<th>1.4</th>
<th>1.5</th>
<th>1.6</th>
<th>1.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterations</td>
<td>45</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
DISTANCE FROM ORIGIN IN Y DIRECTION, IN 1000 FEET

EXPLANATION

R = CONSTANT HEAD BOUNDARY
### = VALUE EXCEEDED 3 FIGURES
MULTIPLICATION FACTOR = 10,000
DISTANCE FROM ORIGIN IN Y DIRECTION, IN 1000 FEET

EXPLANATION
-------
R = CONSTANT HEAD BOUNDARY
*** = VALUE EXCEEDED 3 FIGURES
MULTIPLICATION FACTOR = 10,000
<table>
<thead>
<tr>
<th>Layer</th>
<th>Drawdown</th>
<th>Layer</th>
<th>Drawdown</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.66</td>
<td>11</td>
<td>0.71</td>
</tr>
<tr>
<td>2</td>
<td>0.72</td>
<td>12</td>
<td>0.73</td>
</tr>
<tr>
<td>3</td>
<td>0.65</td>
<td>13</td>
<td>0.69</td>
</tr>
<tr>
<td>4</td>
<td>0.62</td>
<td>14</td>
<td>0.66</td>
</tr>
<tr>
<td>5</td>
<td>0.55</td>
<td>15</td>
<td>0.65</td>
</tr>
<tr>
<td>6</td>
<td>0.49</td>
<td>16</td>
<td>0.60</td>
</tr>
<tr>
<td>7</td>
<td>0.42</td>
<td>17</td>
<td>0.54</td>
</tr>
<tr>
<td>8</td>
<td>0.35</td>
<td>18</td>
<td>0.45</td>
</tr>
<tr>
<td>9</td>
<td>0.28</td>
<td>19</td>
<td>0.35</td>
</tr>
<tr>
<td>10</td>
<td>0.21</td>
<td>20</td>
<td>0.26</td>
</tr>
</tbody>
</table>
APPENDIX V

DEFINITION OF PROGRAM VARIABLES

A
dummy array name used to input matrix data;

R
TC(I-1,J,K)/DLY(I);

RBOTTOM
elevation of the bottom of the upper unit;

CDLT
multiplying factor for the time step;

D
TR(I,J,K)/DELX(J);

ODN
vector that contains drawdown values (L);

DEL
TIME INCREMENT (T);

DELS
GRID SPACING IN THE X DIRECTION (L);

DELY
GRID SPACING IN THE Y DIRECTION (L);

DELZ
GRID SPACING IN THE Z DIRECTION (L);

DLM
dummy array used to complete the argument list for entry array when the last 3 elements are not required;

FL
element of upper triangular factor U;

ERR
closure criteria (L);

F
TR(I,J,K)/DELX(J);

FAC
see explanation in group III; array data;

FL
element of upper triangular factor U;

FLOW
flow rate to a constant-head node (L**3/T);

GL
element of upper triangular factor U;

H
TC(I,J,K)/DELY(I);

HEADING	title for simulation;

I0
number of rows;

ICHK
vector containing problem options;

ICK1
option to read head data from disk;

ICK2
option to write results on disk;

IDRAW
option to print drawdown;

IERR
= 2, program has exceeded permitted iterations;

IFINAL
= 0 all time steps except the last;

= 1 last time step in pumping period;

IFLO
option to compute a volumetric balance;

IHEAD
option to print head matrix;

IK+JK+K5
dimensions of TK array;

IP
maximum of I0+x0;

IN
dummy array used to print matrix names in data;

IPFT
an array of formats used to read matrix data;

IPG
an array of formats used to print matrix data;

IP+JP
dimensions of perm and bottom arrays;

IPRN
see explanation in group III; array data;

IPU1
option to read head and mass balance values from cards;

IPU2
option to punch results on cards;

IG+JQ
dimensions of QRE array;

IGRE
option for recharge;

ISUM
the cumulative words of storage used in the vector Y;

IRECD
see explanation in group III; array data;

IRECS
see explanation in group III; array data;

IRW
counter to indicate the current record number in the direct access data set;

IT
iteration counter;

ITMAX
maximum number of iterations per time step;

ITMX1
ITMAX+1;

ITTO
vector containing total number of iterations per time step;

V = 1
IVAR  SEE EXPLANATION IN GROUP III: ARRAY DATA!
IWATER OPTION FOR WATER-TABLE CONDITIONS IN UPPER LAYER!
JO NUMBER OF COLUMNS!
JFLC ARRAY CONTAINING LOCATION OF CONSTANT-HEAD NODES!
KO NUMBER OF LAYERS!
KP NUMBER OF THE PUMPING PERIOD!
KPM1 NOT CURRENTLY USED!
KT TIME STEP COUNTER!
KTH NUMBER OF TIME STEPS BETWEEN PRINTOUTS!
L VECTOR CONTAINING INITIAL ADDRESS OF ARRAYS!
LENGTH NUMBER OF ITERATION PARAMETERS!
LEVEL1 VECTOR CONTAINING LAYERS FOR WHICH DRAWDOWN MAPS ARE TO BE PRINTED!
LEVEL2 VECTOR CONTAINING LAYERS FOR WHICH HEAD MAPS ARE TO BE PRINTED!
NAME AN ARRAY OF MATRIX NAMES!
NCH NUMBER OF CONSTANT-HEAD NODES!
NPFR NUMBER OF PUMPING PERIODS!
NLMT NUMBER OF TIME STEPS!
NWEL NUMBER OF WELLS FOR A PUMPING PERIOD!
OLD HEAD AT THE END OF THE PREVIOUS TIME STEP!
PERM HYDRAULIC CONDUCTIVITY OF THE UPPER UNIT!
PHI HYDRAULIC HEAD (L)!
QR RECHARGE RATE (L/T)!
GRE RECHARGE RATE (L/T)!
RD0 S/DLT (1/T)!
RDP VECTOR CONTAINING ITERATION PARAMETERS!
S STORAGE COEFFICIENT!
START HYDRAULIC HEAD AT THE START OF THE SIMULATION!
SU TK(I+J+K)/DELT(K)!
SUM TOTAL ELAPSED TIME IN THE SIMULATION (T)!
SUMP TOTAL ELAPSED TIME IN THE PUMPING PERIOD (T)!
T TRANSMISSIVITY (L**2/T)!
TC HARMONIC AVERAGE OF T/DELY # I+1/2+J+K (L/T)!
TEST = 0 CLOSURE CRITERIA SATISFIED!
TEST = 1 CLOSURE CRITERIA NOT SATISFIED!
TEST3 MAXIMUM CHANGE IN HEAD FOR THE TIME STEP!
TF AN ARRAY USED TO READ AND TRANSFER DIRECTIONAL TRANSMISSIVITY FACTORS!
TK HARMONIC AVERAGE OF BK/DELY # I+J+K*1/2 (L/T)!
TMAX NUMBER OF DAYS IN THE PUMPING PERIOD (T)!
TR HARMONIC AVERAGE OF T/DELY # I+J+1/2+K (L/T)!
V INTERMEDIATE VECTOR!
WELL TOTAL ELAPSED TIME IN THE SIMULATION (L*3/T)!
XI ARRAY CONTAINING INCREMENTAL HEAD VALUES IN SIP SOLUTION (L)!
Y VECTOR CONTAINING ARRAY STORAGE!
Z TK(I+J+K-1)/DELT(K)!

DEFINITION OF VARIABLES IN CHECKI SUBROUTINE
-----------------------------------------------------
CFLUX INFLOW FROM RECHARGE WELLS (L**3/T)!
CFLUXT CUMULATIVE VOLUME OF WATER FROM RECHARGE WELLS (L**3)!
CHD1 RATE OF INFLOW TO CONSTANT HEAD BOUNDARY (L**3/T)!
CHD2 RATE OF INFLOW FROM CONSTANT HEAD BOUNDARY (L**3/T)!
CHDT CUMULATIVE DISCHARGE TO CONSTANT HEAD BOUNDARY (L**3)!

V = 2
CUMULATIVE VOLUME OF WATER INFLOW FROM CONSTANT HEAD BOUNDARY (L**3)
ERROR IN MASS BALANCE (L**3)
EVAPOTRANSPIRATION RATE (L**3/T)
CUMULATIVE DISCHARGE BY ET (L**3)
RATE OF LEAKAGE DUE TO GRADIENTS AT THE START OF THE PUMPING PERIOD (L**3/T)
NET LEAKAGE RATE (L**3/T)
RATE OF DISCHARGE BY LEAKAGE (L**3/T)
CUMULATIVE VOLUME OF WATER DISCHARGED BY LEAKAGE (L**3)
CUMULATIVE VOLUME OF WATER INFLOW FROM LEAKAGE (L**3)
PERCENT ERROR IN CUMULATIVE MASS BALANCE
DISCHARGE FROM WELLS (L**3/T)
CUMULATIVE VOLUME OF WATER DISCHARGED BY PUMPING WELLS (L**3)
RECHARGE RATE (L**3/T)
CUMULATIVE VOLUME OF WATER DERIVED FROM RECHARGE (L**3)
RATE OF CHANGE IN STORAGE FOR THE TIME STEP (L**3/T)
CUMULATIVE VOLUME OF WATER DERIVED FROM STORAGE (L**3)
SUM OF RECHARGE AND DISCHARGE RATES FOR THE TIME STEP (L**3/T)
CUMULATIVE VOLUME OF WATER FROM ALL SOURCES (L**3)
NET FLOW TO BOTTOM LAYER (L**3/T)
NET FLOW TO TOP LAYER (L**3/T)

DEFINITION OF VARIABLES IN THE PRINTAI SUBROUTINE

BLANK CONTAINS BLANK SYMBOLS
DINCH NUMBER OF MAP UNITS PER INCH
DIST LOCATION OF NEXT COLUMN OF NODAL VALUES TO BE PRINTED
FACT1 FACTOR FOR ADJUSTING VALUE OF DRAWDOWN PRINTED
FACT2 FACTOR FOR ADJUSTING VALUE OF HEAD PRINTED
K ADJUSTED VALUE OF DRAWDOWN OR HEAD
LA LAYER FOR WHICH A MAP IS BEING PRINTED
MESUR NAME OF MAP LENGTH UNIT
N INDEX FOR SYMBOLS
NA INDICES FOR LOCATING X LABEL
NC NUMBER OF BLANKS BEFORE GRAPH
NG = 1, FOR DRAWDOWN MAP
 = 2, FOR HEAD MAP
N1 NUMBER OF LINES PER INCH
N2 NUMBER OF CHARACTERS PER INCH
N3 NUMBER OF CHARACTERS PER LINE
N4 NUMBER OF LINES IN THE PLOT
N8 MAXIMUM NUMBER OF CHARACTERS IN Y DIRECTION
NXO NUMBER OF INCHES IN THE X DIMENSION OF PLOT
NYO NUMBER OF INCHES IN THE Y DIMENSION OF PLOT
PRNT CONTAINS THE ARRANGEMENT OF SYMBOLS FOR EACH LINE
SPACNG CONTOUR INTERVAL (L)
SYM VECTOR CONTAINING SYMBOLS USED IN THE PLOT
TITLE TITLE FOR PLOT
VF1, VF2, VF3 VARIABLE FORMATS FOR CENTERING PLOT
WIDTH WIDTH OF MODEL (L)
XLABEL LABEL FOR X AXIS
XN NUMBERS FOR X AXIS
XM 1 INCH/(N1#2)
XSFM X SCALE FACTOR
YDIM LENGTH OF AQUIFER IN Y DIRECTION (L).
YLABEL LABEL FOR Y AXIS
YLEN LOCATION OF NEXT VALUE IN THE COLUMN TO BE PRINTED
YN NUMBERS FOR Y AXIS
YSFM Y SCALE FACTOR
Z LOCATION OF NEXT LINE TO BE PRINTED.
APPENDIX VI
Program Listing

--- Program Listing ---

FINITE-DIFFERENCE MODEL FOR SIMULATION OF GROUND-WATER FLOW IN
THREE DIMENSIONS, SEPTEMBER, 1975 BY P.C. TRECOTT, U.S. G.S.
WITH CONTRIBUTIONS TO MAIN, DATAI AND SOLVE BY S.P. LARSON

SPECIFICATIONS:

REAL *8YSTR
DIMENSION Y(102000), L(25), HEADNG(33), NAME(42), INFT(2,2), IOFT
19,4), DUM(3)

EQUIVALENCE (YSTR,Y(1))
COMMON /INTEGR/ IO,J0,K0,I1,J1,K1,I,J,K,NP,E,KTH,ITMAX,LENGTH,KP,N
1WEL,NUMT,FINAL,IT,K,HEAD,IDRAW,IFLO,IER,12,J2,K2,IMAX,ITMX,NC,
2H, IDK1, IDK2, IWATER, ICRE, IP, J,P, 10, JO, JK, K5, IPU1, IPU2, ITK
COMMON /SPARAM/ TMAX, COLT, DELT, ERR, TEST, SUM, SUMP, QR
COMMON /SARRAY/ ICHIP, LEVEL(9), LEVEL(2)

DATA NAME/2*4H t4H START r4HMEAD,4HRA6/!
2*4H TK,4H,FFIC,4HIENT,2*4H, TR,4HANSM,4HISI,4HVITY,2*4H/!
3T,4HLEV1/4HELEVA,4HTION,2*4H,4H R,4HECHA,4HRGE,4HRATE/!

DATA INFT/4H(20F,4H4.0),4H(8F1,4H0.4)/!
DATA IOFT/4H(lHO»4H12,4H2X,2,4HOF6,t4Hl/(5t4HX,20»4HF6,1,4H))/!
14H /4H(1H0,4H15,4H14F9,4H.5/(4H1H +4H5X,1,4HF9,4H5))/4H/!
2 4H(1H0,4H15,4H10EU,4H25,4H(1H +4H5X,4H10E1,4H2.5,4H)/!
3 4H(1H0,4H15,4H10E1,4H1.3,4H1H +4H5X,4H10E1,4H1.3,4H)/

DEFINE FILE 2(8,1520,U»KKK)

--- READ TITLE, PROGRAM SIZE AND OPTIONS ---
READ (5,200) HEADNG
WRITE (6,190) HEADNG
READ (5,160) IO,J0,K0,ITMAX,NCH
WRITE (6,180) IO, J0, K0, ITMAX, NCH
READ (5,210) IDRAW, HEAD, IFLO, IDK1, IDK2, IWATER, ICRE, IPU1, IPU2, ITK
WRITE (6,220) IDRAW, HEAD, IFLO, IDK1, IDK2, IWATER, ICRE, IPU1, IPU2, ITK
IER=0

--- COMPUTE DIMENSIONS FOR ARRAYS ---
J1=JO-1
I1=10-1
K1=K0-1
I2=JO-2
J2=JO-2
K2=K0-2
IMAX=MAX0(IO+J0)
NCH=MAX0(1,NCH)
ITMX=ITMAX+1
ISIZ=10+K0
IM=10+J0
IK=MAX0(IK1*K1,1)
ISUM=2*ISIZ+1
L(1) = 1
DC 30 I=2,14
IF (I.NE.8) GO TO 20
L(8) = ISUM
ISUM = ISUM + IK2
IF (IK2.EQ.1) GO TO 10
IK = 10
JK = J0
K5 = K1
GO TO 30
10 IK = 1
JK = 1
K5 = 1
GO TO 30
20 L(I) = ISUM
ISUM = ISUM + ISIZ
30 CONTINUE
L(15) = ISUM
ISUM = ISUM + J0
L(16) = ISUM
ISUM = ISUM + IO
L(17) = ISUM
ISUM = ISUM + K0
L(18) = ISUM
ISUM = ISUM + IK1
L(19) = ISUM
ISUM = ISUM + K0 = 3
L(20) = ISUM
ISUM = ISUM + ITMX1
L(21) = ISUM
ISUM = ISUM + 3*NCD
L(22) = ISUM
ISUM = ISUM + NCD
L(23) = ISUM
IF (IWATER.NE.ICHK(6)) GO TO 40
ISUM = ISUM + IK1
L(24) = ISUM
ISUM = ISUM + IK1
IP = 10
JP = J0
GO TO 50
40 ISUM = ISUM + 1
L(24) = ISUM
ISUM = ISUM + 1
IP = 1
JP = 1
50 L(25) = ISUM
IF (IQRE.NE.ICHK(7)) GO TO 60
ISUM = ISUM + IK1
IQ = 10
JQ = J0
GO TO 70
60 ISUM = ISUM + 1
IQ = 1
JQ = 1
70 WRITE (6,170) ISUM

VI-2
---PASS INITIAL ADDRESSES OF ARRAYS TO SUBROUTINES---
CALL DATA1(Y(L(1)),Y(L(2)),Y(L(3)),Y(L(4)),Y(L(5)),Y(L(6)),Y(L(7)))
1,Y(L(8)),Y(L(9)),Y(L(10)),Y(L(11)),Y(L(12)),Y(L(13)),Y(L(14)),Y(L(15))
224),Y(L(25)))
CALL STEPl(Y(L(1)),Y(L(2)),Y(L(3)),Y(L(4)),Y(L(5)),Y(L(6)),Y(L(7)))
1,Y(L(8)),Y(L(9)),Y(L(10)),Y(L(11)),Y(L(12)),Y(L(13)),Y(L(14)),Y(L(15))
20))
CALL SOLL(Y(L(1)),Y(L(2)),Y(L(3)),Y(L(4)),Y(L(5)),Y(L(6)),Y(L(7)))
1,Y(L(8)),Y(L(9)),Y(L(10)),Y(L(11)),Y(L(12)),Y(L(13)),Y(L(14)),Y(L(15))
211,Y(L(16)),Y(L(17)),Y(L(18)),Y(L(19)),Y(L(20)),Y(L(21)),Y(L(22)),Y(L(23))
24),Y(L(25)))
CALL CCF(Y(L(1)),Y(L(2)),Y(L(3)),Y(L(4)),Y(L(5)),Y(L(6)),Y(L(7)))
1,Y(L(8)),Y(L(9)),Y(L(10)),Y(L(11)),Y(L(12)),Y(L(13)),Y(L(14)),Y(L(15))
224),Y(L(25)))
CALL CHK(Y(L(1)),Y(L(2)),Y(L(3)),Y(L(4)),Y(L(5)),Y(L(6)),Y(L(7)))
1,Y(L(8)),Y(L(9)),Y(L(10)),Y(L(11)),Y(L(12)),Y(L(13)),Y(L(14)),Y(L(15))
222),Y(L(25)))
CALL PRNTA1(Y(L(1)),Y(L(2)),Y(L(3)),Y(L(4)),Y(L(5)),Y(L(6)),Y(L(7)))
1,Y(L(8)),Y(L(9)),Y(L(10)),Y(L(11)),Y(L(12)),Y(L(13)),Y(L(14)),Y(L(15))
222),Y(L(25)))
---START COMPUTATIONS---
---READ AND WRITE DATA FOR GROUPS II AND III---
CALL DATA1
INN=1
N1J=10*J0
DO 80 K=1,K0
LOC=L(2),(K-1)*N1J
80 CALL ARRAY(Y(LOC),INFT(1,2),IOFT(1,1),NAME(1),IRN,DUM)
DO 90 K=1,K0
LOC=L(5),(K-1)*N1J
90 CALL ARRAY(Y(LOC),INFT(1,1),IOFT(1,2),NAME(7),IRN,DUM)
DO 100 K=1,K0
LOC=L(4),(K-1)*N1J
L1=L(19)+K-1
L2=L(19)+K0*K-1
L3=L(19)+2*K0*K-1
CALL ARRAY(Y(LOC),INFT(1,1),IOFT(1,2),NAME(13),IRN,DUM)
Y(L1)=DUM(1)
Y(L2)=DUM(2)
Y(L3)=DUM(3)
100 WRITE (6,230) K,Y(L1),Y(L2),Y(L3)
IF (ITK.NE.ICHK(10)) GO TO 120
DO 110 K=1,K1
LOC=L(8),(K-1)*N1J
110 CALL ARRAY(Y(LOC),INFT(1,1),IOFT(1,3),NAME(19),IRN,DUM)
120 IF (IHWATER.NE.ICHK(6)) GO TO 130
K = K0
CALL ARRAY(Y(L(23)),INFT(1,1),IOFT(1,4),NAME(25),IRN,DUM)
CALL ARRAY(Y(L(24)),INFT(1,1),IOFT(1,1),NAME(31),IRN,DUM)
130 IF (IQR.EQ.ICHK(7)) CALL ARRAY(Y(L(25)),INFT(1,1),IOFT(1,4),NAME(25),IRN,DUM)
CALL MDAT
---COMPUTE TRANSMISSIVITY FOR UNCONFINED LAYER---
IF (IHWATER.EQ.ICHK(6)) CALL TRANS(1)

VI-3
C ---COMPUTE T COEFFICIENTS---
CALL TCOF
C ---COMPUTE ITERATION PARAMETERS---
CALL ITER
C ---READ TIME PARAMETERS AND PUMPING DATA FOR A NEW PUMPING PERIOD---
140 CALL NEWPER
C
KT=0
IFINAL=0
C ---START NEW TIME STEP COMPUTATIONS---
150 CALL NEWSTP
C ---START NEW ITERATION IF MAXIMUM NO. ITERATIONS NOT EXCEEDED---
CALL NEWITA
C ---PRINT OUTPUT AT DESIGNATED TIME STEPS---
C ---LAST TIME STEP IN PUMPING PERIOD?---
IF (IFINAL.NE.1) GO TO 150
C ---CHECK FOR NEW PUMPING PERIOD---
IF (KP.LT.NPER) GO TO 140
C
STOP
C ---FORMATS---

160 FORMAT (8I10)
170 FORMAT (A54X,WORDS OF VECTOR Y USED = I7)
180 FORMAT (A62X,NUMBER OF ROWS = I5/60X,NUMBER OF COLUMNS = I5/39X,
MAXIMUM PERMITTED NUMBER OF ITEMS = I5/28X,
NUMBER OF CONSTANT HEAD NODES = I5)
190 FORMAT (A33A4)
200 FORMAT (20A4)
210 FORMAT (16(A4,X))
220 FORMAT (21(A4,X))
230 FORMAT (1H0/44X,DIRECTIONAL TRANSMISSIVITY MULTIPLICATION FACTORS
1 FOR LAYER = 015.7/76X,Y = 015.7/76X,Z = 015.7)
END
SUBROUTINE DATA (PHI, STRT, OLD, T, S, STR, TC, TK, WELL, DELX, DELY, DELZ, FACD)

C ---READ AND WRITE DATA---
C
C SPECIFICATIONS:
REAL *8 PHI
REAL *8 XLABEL, YLABEL, TITLE

DIMENSION PHI(I, J, K), STRT(I, J, K), OLD(I, J, K), T(I, J, K), WELL(I, J, K), DELX(J), DELY(J), DELZ(K), FACT(K), PERM(IP, JP), QRE(IC, JQ), TF(3), A(IO, JO), IK(IP, JP), ITP(JP), QRET, CHST, CHDT, FLUXT, STORT, ETF(3, I), PLXNT

COMMON /INTEGR/ IO, JO, K, I1, J1, K1, IK, IP, ITMAX, LENGTH, KP, NST
COMMON /NUMT/ INITIAL, IT, IHEAD, IREAD, IFLX, IERR, I2, J2, K2, IMAX, ITMAX, N,
COMMON /SPARAM/ TMAX, COLT, DELT, ERR, TEST, SUM, SUMP, QR
COMMON /SARRAY/ ICHK(13), LEVEL1(9), LEVEL2(9)
COMMON /PRINC/ XLABEL(6), YLABEL(6), TITLE(6), XN, M1, M2, M3, SYM(17), XN(100), YN(13)
COMMON /FACT1/ FACT1, FACT2

RETURN
C
C ENTRY DATAI
C
C ---READ SCALAR PARAMETERS---
READ (5, 330) NPER, KTH, ERR, LENGTH
WRITE (6, 340) NPER, KTH, ERR
READ (5, 460) XSCALE, YSCALE, XN, YN, M1, M2, M3, SYM, XN, YN
IF (XSCALE.EQ.0.) WRITE (6, 470) XSCALE, YSCALE, XN, YN, M1, M2, M3, SYM

COMMON /ICMK/ ICMK(13), LEVEL1(9), LEVEL2(9)

C ---READ CUMULATIVE MASS BALANCE PARAMETERS---
READ (5, 450) SUM, SUMP, PUMPT, CFLUXT, CHST, CHDT, FLUXT, STORT, ETF
WRITE (6, 430) SUM, SUMP, PUMPT, CFLUXT, CHST, CHDT, FLUXT, STORT, ETF
REWIND 4
DO 40 K = 1, 100
40 WRITE (6, 430) SUM
C
C ---READ INITIAL HEAD VALUES FROM CARDS---
DO 10 K = 1, K
DO 10 I = 1, 10
10 READ (5, 360) PHI(I, J, K)
GO TO 30
C
C ---READ INITIAL HEAD AND MASS BALANCE PARAMETERS FROM DISK---
REWIND 4
30 WRITE (6, 430) SUM
DO 40 K = 1, K
C

VI-5
WRITE (6,440) K
DO 40 I=1,10
40 WRITE (6,350) I*(PHI(I,J,K)+J=1,J0)
C
DO 60 K=1,K0
DO 60 I=1,10
DO 60 J=1,J0
WELL(I,J,K)=0.
TR(I,J,K)=0.
TC(I,J,K)=0.
IF (K.NE.K0) TK(I,J,K)=0.
60 CONTINUE
RETURN
C
ENTRY ARRAY(A,INFT,IOFT,IN,IRN,TF)
C
ENTRY MODAT
C
DO 150 K=1,K0
DO 150 I=1,10
DO 150 J=1,J0
IF (I.EQ.1.OR.I.EQ.IO.OR.J.EQ.1.OR.J.EQ.J0) T(I,J,K)=0.
IF (IDK4.NE.ICMK(4).AND.IP11.NE.ICMK(8)) PHI(I,J,K)=STRT(I,J,K)
IF (K.NE.K0.OR.IWATER.NE.ICMK(6)) GO TO 150
IF (I.EQ.1.OR.I.EQ.IO.OR.J.EQ.1.OR.J.EQ.J0) PERM(I,J)=0.
150 CONTINUE
C
ENTRY DELX
C
READ (5,330) FAC,IVAR,IPRN,TF
READ (5,330) FAC,IVAR,IPRN,TF,IRECS,IRECD
IC=4*IRECS+2*IVAR*IPRN+1
GO TO (70,70,90,90,120,120), IC
70 DO 80 I=1,10
DO 80 J=1,J0
A(I,J)=FAC
WRITE (6,280) I*(FAC+K
GO TO 140
80 IF (IC.EQ.3) WRITE (6,290) I*(K
DO 110 I=1,10
READ (5,INFT) (A(I,J),J=1,J0)
DO 100 J=1,J0
A(1,J)=A(I,J)*FAC
IF (IC.EQ.3) WRITE (6*290) I*(A(I,J),J=1,J0)
60 TO 140
READ (2'IRN) A
IF (IC.EQ.6) GO TO 140
WRITE (6,290) I*(K
IF (IRECD.EQ.1) WRITE (2*IRN) A
140 IRN=IRN+1
RETURN
C
ENTRY DELX
C
READ (5,330) FAC,IVAR,IPRN
IF (IVAR.EQ.1) READ (5,330) (DELX(J),J=1,J0)
IF (IVAR.NE.1) GO TO 160
DELX(J)=FAC
DAT0570
DAT0580
DAT0590
DAT0600
DAT0610
DAT0620
DAT0630
DAT0640
DAT0650
DAT0660
DAT0670
DAT0680
DAT0690
DAT0700
DAT0710
DAT0720
DAT0730
DAT0740
DAT0750
DAT0760
DAT0770
DAT0780
DAT0790
DAT0800
DAT0810
DAT0820
DAT0830
DAT0840
DAT0850
DAT0860
DAT0870
DAT0880
DAT0890
DAT0900
DAT0910
DAT0920
DAT0930
DAT0940
DAT0950
DAT0960
DAT0970
DAT0980
DAT0990
DAT1000
DAT1010
DAT1020
DAT1030
DAT1040
DAT1050
DAT1060
DAT1070
DAT1080
DAT1090
DAT1100
DAT1110
DAT1120
VI-6
GO TO 170
160 DELX(J)=FAC
170 CONTINUE
IF (IVAR.EQ.1.AND.IPRN.NE.1) WRITE (6,370) (DELX(J),J=1,JO)
IF (IVAR.EQ.0) WRITE (6,300) FAC
C
*************** DELY ***************
READ (5,330) FAC,IVAR,IPRN
IF (IVAR.EQ.1) READ (5,330) (DELY(I),I=1,IO)
DO 190 I=1,IO
IF (IVAR.NE.1) GO TO 180
DELY(I)=DELY(I)*FAC
GO TO 190
180 DELY(I)=FAC
190 CONTINUE
IF (IVAR.EQ.1.AND.IPRN.NE.1) WRITE (6,380) (DELY(I),I=1,IO)
IF (IVAR.EQ.0) WRITE (6,310) FAC
READ (5,330) FAC,IVAR,IPRN
IF (IVAR.EQ.1) READ (5,330) (DELY(I),I=1,IO)
C
*************** DELZ ***************
READ (5,330) FAC,IVAR,IPRN
IF (IVAR.EQ.1) READ (5,330) (DELZ(K),K=1,K0)
DO 210 K=1,K0
IF (IVAR.NE.1) GO TO 200
DELZ(K)=DELZ(K)*FAC
GO TO 210
200 DELZ(K)=FAC
210 CONTINUE
IF (IVAR.EQ.1.AND.IPRN.NE.1) WRITE (6,390) (DELZ(K),K=1,K0)
IF (IVAR.EQ.0) WRITE (6,320) FAC
C
---INITIALIZE VARIABLES---
B=0.
D=0.
F=0.
M=0.
SU=0.
Z=0.
IF (XSCALE.NE.0.) CALL MAP
RETURN
C
---READ TIME PARAMETERS AND PUMPING DATA FOR A NEW PUMPING PERIOD---
ENTRY NEWPER
C
***************
READ (5,330) KP,KPM1,NWEL,TMAX,NUMT,CDLT,DELT
C
---COMPUTE ACTUAL DELT AND NUMT---
DT=DELT/24.
TM=0.
DO 220 I=1,NUMT
DT=CDLT*DT
TM=TM+DT
IF (TM.GE.TMAX) GO TO 230
220 CONTINUE
GO TO 240
230 DELT=TMAX/TM*DELT
NUMT=I
WRITE (6,400) KPT,TMAX,NUMT,DELT,CDLT
DELT=DELT*3600.
TMAX=TMAX*86400.
SLMP=0.0

---READ AND WRITE WELL PUMPING RATES---
WRITE (6,410) NWEL
IF (NWEL.EQ.0) GO TO 260
DC 245 K = 1*K0
DC 245 I = 1+I0
DC 245 J = 1+J0
P45 WELL(I+J+K) = 0.0
DO 250 I=WEL,1,NWEL
READ (5,330) K*IK,WELL(I,J,K)
WRITE (6,420) K,I,J,WELL(I,J,K)
WELL(I+J+K) = WELL(I,J,K)/(DELX(J)*DELY(I))
250 CONTINUE
RETURN

---FORMATS---
280 FORMAT (1H0,52X,6A4,1*0,15.7,1*FOR LAYE1+,3)
290 FORMAT (1H1,46X,6A4,1*MATRIX+ LAYER+,3/46X,4('I'))
300 FORMAT (10+72X,DELY,1*0,15.7)
310 FORMAT (10+72X,DELT,1*0,15.7)
320 FORMAT (10+72X,DELZ,1*0,15.7)
330 FORMAT (8G10.0)
340 FORMAT (10+51X,NUMBER OF PUMPING PERIODS =*,15/49X,TIME STEPS BDAT1930
1ETWEEN PRINTOUTS =*,15/51X,ERROR CRITERIA FOR CLOSURE =*,15,7//DAT1940
350 FORMAT (10+12,2X,12F6.1/(5X,12F6.1))
360 FORMAT (8F10.4)
370 FORMAT (1H1,46X,40MGRID SPACING IN PROTOTYPE IN X DIRECTION/47X,40DAT1970
1(*=+)//(10+12F10.0))
380 FORMAT (1H1,46X,40MGRID SPACING IN PROTOTYPE IN Y DIRECTION/47X,40DAT1990
1(*=+)//(10+12F10.0))
390 FORMAT (1H1,46X,40MGRID SPACING IN PROTOTYPE IN Z DIRECTION/47X,40DAT2010
1(*=+)//(10+12F10.0))
400 FORMAT (=1+50X,PUMPING PERIOD NO.,14,4,1,F10.2,1*TIME STEPS =*,16/59X,DELH IN HOURS =*,11,3//DAT2030
253X,MULTIPLIER FOR DELT =*,10.3)
410 FORMAT (=1+63X,WELL,(9F10.0)//50X,K,9X,I,9X,J,9PUDAT2060
1MPING RATE//)
420 FORMAT (41X,3110,2F13.2)
430 FORMAT (=1+40X,CONTINUATION = HEAD AFTER ,*G20.7,1*SEC PUMPING DAT2090
1*/42X,58(*=+))
440 FORMAT (1+55X,INITIAL HEAD MATRIX, LAYER+,3/56X,30(*=+))
450 FORMAT (4G20.10)
460 FORMAT (3G10.0,2G10.0,9I1,1X,A8)
470 FORMAT (10+,30X,ON ALPHAMERIC MAP!,/40X,MULTIPLICATION FACTOR FODAT2140
1R X DIMENSION =*,15,7/40X,MULTIPLICATION FACTOR FOR Y DIMENSION DAT2150
2=*,15,7/55X,MAP SCALE IN UNITS OF ,1*11/50X,NUMBER OF ,8A,1MULTIPLICATION FACTOR FOR DRAWDOWN =*,15,7,1DAT2170
4* PRINTED FOR LAYERS,1912/47X,MULTIPLICATION FACTOR FOR HEAD =*,20DAT2180
515,7,* PRINTED FOR LAYERS,1912)
END
SUBROUTINE STEP(PHI, STRT, OLD, T, TC, TK, WELL, DELX, DELY, DELZ, FACT, STP)
1) DDT (= TEST3)

INITIALIZE DATA FOR A NEW TIME STEP AND PRINT RESULTS

SPECIFICATIONS:
REAL *8 PHI
REAL *8 XLABEL, YLABEL, TITLE, XN1, MESUR

DIMENSION PHI(IO, JO, KO), STRT(IO, JO, KO), OLD(IO, JO, KO), T(IO, JO, KO)
1, S(IO, JO, KO), TR(IO, JO, KO), TC(IO, JO, KO), TK(IX, JK, KS), WELL(IO, STP)
2, JO, KO, DELX(JO), DELY(IO), DELZ(KO), FACT(KO, 3), DDN(IMAX), TEST3
3, ITMAX, ITTO(50)

COMMON /INTEGR/ IO, JO, KO, I1, J1, K1, I, J, K, NPER, KTH, ITMAX, LENGTH, KP, NSTP
1, WELL, NUNT, ITFINAL, IT, KT, IMAX, IDRAW, IFLO, IERR, IZ, J2, K2, IMAX, ITMX1, NCSTP
2, H1, OK1, OK2, I, WATER, IQRE, IP, JP, IQ, JO, IK, JS, KS, IP1, IPU2, ITK

COMMON /SPARAM/ TMAX, CQLT, OELT, ERR, TEST, SUMP, SUMP, PR
COMMON /SARRAY/ ICHR(13), LEVEL1(9), LEVEL2(9)

COMMON /PR/ XLABEL(3), YLABEL(6), TITLE(6), XN1, MESUR, PRINT(12), BLANKSTP
1, (60), DIGIT(122), VF1(6), VF2(6), VF3(7), XSCALE, DINCH, SYM(17), XN(100)
2, YTH(13), NA(4), N1, N2, N3, YSCALE, FACT1, FACT2

RETURN

ENTRY NEWSTP

KT = KT + 1
IT = 0
DO 10 K = 1, KO
DO 10 I = 1, IO
DO 10 J = 1, JO
10 OLD(I + J, K) = PHI(I + J, K)
DELX = DLX / DELT
SUM = SUM / DELT
SUMP = SUMP / DELT
YRSP = DAYS / 365.

RETURN

---PRINT OUTPUT AT DESIGNATED TIME STEPS---

ENTRY OUTPUT

IF (KT EQ NUNT) IFINAL = 1
ITTO(KT) = IT
IF (IT LE ITMAX) GO TO 20
IT = IT - 1
ITTO(KT) = IT
IERR = 2

RETURN
---IF MAXIMUM ITERATIONS EXCEEDED WRITE RESULTS ON DISK OR CARDS---
IF (ID2.EQ.ICHM5) WRITE (4) PHI, SUM, PUMP, CFLUX, QRET, CHST
1, CHDT, FLUTX, STOR, ETFLX, FLXNT
IF (IPU2.EQ.ICHM9) WRITE (7, 230) SUM, PUMP, CFLUX, QRET, CHST
1, CHDT, FLUTX, STOR, ETFLX, FLXNT

20 IF (IFLO.EQ.ICHM3) CALL CHECK
IF (IERR.EQ.2) 60 TO 30
IF (MOD(KT, KTH) .NE. 0. AND. IFINAL .NE. 1) RETURN
30 WRITE (6, 210) KT, DELT, SUM, SMIN, HRS, DAYS, YRS, DAYS, YRSP
IF (IFLO.EQ.ICHM3) CALL CWRITE
IT = IT + 1
WRITE (6, 180) (TEST3(J), J = 1, IT)
I3 = 1
I5 = 0
352 IS = IS + 40
I4 = MIN0(KT, IS)
WRITE (6, 240) (I, I = I3, I4)
WRITE (6, 260)
WRITE (6, 250) (ITTO(I), I = I3, I4)
WRITE (6, 260)
IF (KT .LE. IS) GO TO 353
I3 = I3 + 40
GO TO 352

C ---PRINT MAPS---
353 IF (XSCALE .EQ. 0.) 60 TO 70
IF (FACT1 .EQ. 0.) 60 TO 50
DO 40 IA = 1, 9
II = LEVEL1(IA)
IF (II .EQ. 0) GO TO 50
40 CALL PRNATA(1, II)
50 IF (FACT2 .EQ. 0.) GO TO 70
DO 60 IA = 1, 9
II = LEVEL2(IA)
IF (II .EQ. 0) GO TO 70
60 CALL PRNATA(2, II)
70 IF (IDRAW .NE. ICHK(1)) GO TO 100

C ---PRINT DRAWDOWN---
DO 90 K = 1, KO
WRITE (6, 200) K
DO 90 I = 1, IO
DO 80 J = 1, JO
80 DON(J) = STRT(I, J, K) - PHI(I, J, K)
90 WRITE (6, 170) I, (DON(J), J = 1, JO)
100 IF (IHEAD .NE. ICHK(2)) GO TO 120

C ---PRINT HEAD MATRIX---
DO 110 K = 1, KO
WRITE (6, 190) K
DO 110 I = 1, IO
110 WRITE (6, 170) I, (PHI(I, J, K), J = 1, JO)

C ---WRITE ON DISK---
120 IF (IERR.EQ.2) GO TO 130
   IF (KP.LT.NPER.OR.IFINAL.NE.1) RETURN
   IF (IDK2.EQ.ICHK(5)) WRITE (*), PHI, SUM, SUMP, PUMPT, CFLUXT, QRET, CHST
   IF (CHDT.EQ.1) WRITE (*), CFLUXT, SORT, EFLXNT

C        ---PUNCHED OUTPUT---
130 IF (IPU2.NE.ICHK(9)) GO TO 160
   IF (IERR.EQ.2) GO TO 140
   WRITE (7,230) SUM, SUMP, PUMPT, CFLUXT, QRET, CHST, CHDT, FLXNT, SORT, EFLXNT

C        ---FORMATS---
C        C        C
170 FORMAT (+0*14*18F7.2/(5*18F7.2))
180 FORMAT (+0MAXIMUM HEAD CHANGE FOR EACH ITERATION:/+39('>')+/)
210 FORMAT (+1*55X, HEAD MATRIX, LAYER+/3/56X+1*(='))
200 FORMAT (+1*55X, DRAWDOWN, LAYER+/3/59X+18(*='))
210 FORMAT (+1H144X,57(=')/45X,14X, 'TIME STEP NUMBER =', 9, '14X, ')
210 FORMAT (+1H144X,57(=')/50X, 29HSIZE OF TIME STEP IN SECONDS=/55X, 'TO'
210 FORMAT (+1H144X,57(=')/60X, 8HMINUTES=/62X, 6HSTEPS=/70X, 'TIME STEP')
210 FORMAT (+1H144X,57(=')/75X, 5EEEE=14X, 'TIME STEP')
210 FORMAT (+1H144X,57(=')/80X, 8HMINUTES=/82X, 6HSTEPS=/90X, 'TIME STEP')
210 FORMAT (+1H144X,57(=')/95X, 5EEEE=14X, 'TIME STEP')
210 FORMAT (+1H144X,57(=')/100X, 8HMINUTES=/102X, 6HSTEPS=/110X, 'TIME STEP')
210 FORMAT (+1H144X,57(=')/115X, 5EEEE=14X, 'TIME STEP')
210 FORMAT (+1H144X,57(=')/120X, 8HMINUTES=/122X, 6HSTEPS=/130X, 'TIME STEP')
210 FORMAT (+1H144X,57(=')/135X, 5EEEE=14X, 'TIME STEP')
210 FORMAT (+1H144X,57(=')/140X, 8HMINUTES=/142X, 6HSTEPS=/150X, 'TIME STEP')
210 FORMAT (+1H144X,57(=')/155X, 5EEEE=14X, 'TIME STEP')
210 FORMAT (+1H144X,57(=')/160X, 8HMINUTES=/162X, 6HSTEPS=/170X, 'TIME STEP')
210 FORMAT (+1H144X,57(=')/175X, 5EEEE=14X, 'TIME STEP')
210 FORMAT (+1H144X,57(=')/180X, 8HMINUTES=/182X, 6HSTEPS=/190X, 'TIME STEP')
210 FORMAT (+1H144X,57(=')/195X, 5EEEE=14X, 'TIME STEP')
210 FORMAT (+1H144X,57(=')/200X, 8HMINUTES=/202X, 6HSTEPS=/210X, 'TIME STEP')
210 FORMAT (+1H144X,57(=')/215X, 5EEEE=14X, 'TIME STEP')
210 FORMAT (+1H144X,57(=')/220X, 8HMINUTES=/222X, 6HSTEPS=/230X, 'TIME STEP')
210 FORMAT (+1H144X,57(=')/235X, 5EEEE=14X, 'TIME STEP')
210 FORMAT (+1H144X,57(=')/240X, 8HMINUTES=/242X, 6HSTEPS=/250X, 'TIME STEP')
210 FORMAT (+1H144X,57(=')/255X, 5EEEE=14X, 'TIME STEP')
210 FORMAT (+1H144X,57(=')/260X, 8HMINUTES=/262X, 6HSTEPS=/270X, 'TIME STEP')
210 FORMAT (+1H144X,57(=')/275X, 5EEEE=14X, 'TIME STEP')
210 FORMAT (+1H144X,57(=')/280X, 8HMINUTES=/282X, 6HSTEPS=/290X, 'TIME STEP')
210 FORMAT (+1H144X,57(=')/295X, 5EEEE=14X, 'TIME STEP')
210 FORMAT (+1H144X,57(=')/300X, 8HMINUTES=/302X, 6HSTEPS=/310X, 'TIME STEP')
210 FORMAT (+1H144X,57(=')/315X, 5EEEE=14X, 'TIME STEP')
210 FORMAT (+1H144X,57(=')/320X, 8HMINUTES=/322X, 6HSTEPS=/330X, 'TIME STEP')
210 FORMAT (+1H144X,57(=')/335X, 5EEEE=14X, 'TIME STEP')
210 FORMAT (+1H144X,57(=')/340X, 8HMINUTES=/342X, 6HSTEPS=/350X, 'TIME STEP')
210 FORMAT (+1H144X,57(=')/355X, 5EEEE=14X, 'TIME STEP')
210 FORMAT (+1H144X,57(=')/360X, 8HMINUTES=/362X, 6HSTEPS=/370X, 'TIME STEP')
210 FORMAT (+1H144X,57(=')/375X, 5EEEE=14X, 'TIME STEP')
210 FORMAT (+1H144X,57(=')/380X, 8HMINUTES=/382X, 6HSTEPS=/390X, 'TIME STEP')
210 FORMAT (+1H144X,57(=')/395X, 5EEEE=14X, 'TIME STEP')
210 FORMAT (+1H144X,57(=')/400X, 8HMINUTES=/402X, 6HSTEPS=/410X, 'TIME STEP')
SUBROUTINE SOLVE(Phi,Start,Old,T,S,Tc,Tk,Well,Delx,Delay,Delz,Fac3)
IT=EL+FL+V+x1+Test3+Qre)
SOLUTION BY THE STRONGLY IMPLICIT PROCEDURE
---------------------------------------------------------
SPECIFICATIONS:
REAL *8Phi,Rho,B,F,H,Z,SRMop,W,WMn,RHO1,RHO2,RHO3,XPART,YPART
1,ZPART,DMIN1,WHAX,XT,YT,ZT,OABS,DMAX1,OLE,TXM,TYM,TZM
REAL *8E,AL,CL,A,CG,WTU,DL,RES,SUPH,GLX1,ZPHI
DIMENSION PHI(l),Start(l),Old(l),T(l),S(l),Tc(l),Tk(l)
WELL(l),Delx(l),Delay(l),Delz(l),Fact(l),Rmop(l),Test3(l)
El(l),Fl(l),Gl(l),V(l),XI(l),Qre(l)
COMMON /XINT/JO,JO,KO,K1,J1,K,K0,LEN,KM1,LEN,KM2,LEN,KM3
WEL,NUM,Ifinal,IT,KIMAX,Draw,IGF1,II,IER,IT,J2,K2,IMAX,ITM3
ITM2,ITM1,NSPM
COMMON /SPARAM/ TMAX,CDLT,CEL,T,ERR,Test,T,Sum,Sump,QR
COMMON /SARRAY/ ZCK(13),LEVEL1(9),LEVEL2(9)
RETURN
ENTRY ZTER
*****COMPUTE AND PRINT ITERATION PARAMETERS*****
WRITE (6,240)
WIN=1.0
DEL=1.0
P2=LENGTH+1
N=I0*JO*K0
NIJ=I0*JO
XT=3.141593*2/(2.*J2*J2)
YT=3.141593*2/(2.*Z2*Z2)
ZT=3.141593*2/(2.*K0*K0)
RHO1=0.0
RHO2=0.0
RHO3=0.0
DO 40 K=1,K0
DO 40 I=2,l
DO 40 J=2,J1
N=Z*(J-1)*Z0*(K-1)*NIJ
IF (T(N).EQ.O.) GO TO 40
D=TR(N)/DELX(J)
F=TR(N)/DELX(J)
B=TC(N-1)/DELY(I)
M=TC(N)/DELY(I)
SU=0.0
Z=0.0
IF (K.NE.1) Z=Tc(K0-NIJ)/DELZ(K)
IF (K.NE.K0) SU=Tc(K)/DELZ(K)
RHO1=S(N)/DELT
QR=0.0
IF (K.NE.K0) GO TO 10
IF (IDR.EQ.ICHK(7)) QR=QRE(I*(J=1)*10)
10 CONTINUE
TXM=DMAX1(D+F)
TYM=DMAX1(B+H)
TZM=DMAX1(S+Z)
DEN=DMIN1(D+F)
IF (DEN.EQ.0.0) DEN=TXM
IF (DEN.EQ.0.0) GO TO 20
RHO1=DMAX1(RHO1, TYM/DEN)
20 DEN=DMIN1(B+H)
IF (DEN.EQ.0.0) DEN=TYM
IF (DEN.EQ.0.0) GO TO 30
RHO2=DMAX1(RHO2, TXM/DEN)
30 DEN=DMIN1(S+Z)
IF (DEN.EQ.0.0) DEN=TZM
IF (DEN.EQ.0.0) GO TO 40
RHO3=DMAX1(RHO3, TXM/DEN)
40 CONTINUE
XPART=XT/(1.0+RHO1)
YPART=YT/(1.0+RHO2)
ZPART=ZT/(1.0+RHO3)
WMIN=DMIN1(WMIN, XPART, YPART, ZPART)
WMAX=1.0-DMIN1
PJ=1.0
DO 50 I=1, LENGTH
50 RHOP(I)=1.0-(1.0-WMAX)*PJ/P2
WRITE (6, 230) LENGTH, (RHOP(I), I=1, LENGTH)
RETURN
C
C *** INITIALIZE DATA FOR A NEW ITERATION ***
60 IT=IT+1
IF (IT.LE. ITMAX) GO TO 70
WRITE (6, 220)
CALL OUTPUT
70 IF (MOD(DT, LENGTH)) 80, 80, 90
ENTRY NEWITA
80 NTH=0
90 NTH*NTH+1
W=RHOP(NTH)
TEST3(IT+1)=0.
TEST=0.0
BIG=0.
DO 100 I=1, NT
EL(I)=0.
FL(I)=0.
GL(I)=0.
V(I)=0.
100 XI(I)=0.
C
C *** COMPUTE TRANSMISSIVITY AND T COEFFICIENTS FOR UPPER
C HYDROLOGIC UNIT WHEN IT IS UNCONFINED ***
C IF (IWATER.NE.ICHK(6)) GO TO 110
C CALL TRANS(0)
C
--- CHOOSE SIP NORMAL OR REVERSE ALGORITHM ---

110 IF (MOD(IT*2) 120) 120*120+170

120 DC 150 K=1,K0
  DC 150 I=2,11
  DC 150 J=2,11
  N=I*(J-1)*I0*(K-1)*NIJ
  NIA=N+1
  NIB=N-1
  NJ8=N+10
  NK8=N+NIJ
  NKB=N-NIJ

--- SKIP COMPUTATIONS IF NODE OUTSIDE MODEL ---

IF (T(N).EQ.O..OR.S(N).LT.O.) GO TO 150

--- COMPUTE COEFFICIENTS ---

D=TR(NJ8)/DELX(J)
F=TR(N)/DELX(I)
B=TC(NIB)/DELY(I)
H=TC(N)/DELY(I)
SU=0.D0
Z=0.D0

IF (K.NE.1) Z=TK(NKB)/DELZ(K)
IF (K.NE.K0) SU=TK(N)/DELZ(K)
RHO=S(N)/DEL
QR=0.

IF (K.NE.K0) GO TO 130
IF (IQRE.EQ.1.CMK(7)) QR=QRE(I*(J-1)*I0)

--- SIP NORMAL ALGORITHM ---

--- FORWARD SUBSTITUTE, COMPUTING INTERMEDIATE VECTOR V ---

130 E=-B-D-F=H-SU-Z=RHO
BL=B/[(1.0*WL(EL(NIB)+GL(NIB))]
CL=D/[1.0*WL(FL(NJ8)+GL(NJB))]
C=BL*EL(NIB)
G=CL*FL(NJB)
WU=CL*GL(NJB)
U=BL*GL(NIB)

IF (K.NE.1) GO TO 140
AL=Z/[1.0*WL(EL(NKB)+FL(NKB))]
A=AL*EL(NKB)
TU=AL*FL(NKB)
DL=E*WL(A+C*GL+WU+TU+U)=CL*EL(NJB)-BL*FL(NIB)-AL*GL(NKB)
EL(N)=(F-W*(A+C))/DL
FL(N)=(H-W*(G+TU))/DL
GL(N)=(SU-W*(WU+U))/DL
SUPH=0.D0
SUPH=SU+PHI(NKA)
RES=B*PHI(NIB)-D*PHI(NJB)-E*PHI(N)+F*PHI(NJA)+H*PHI(NIB)+SUPH-Z*PS
1HI(NKB)-WELL(N)=RHO*OLD(N)=QR
V(N)=(RES=AL*V(NKB)-BL*V(NIB)-CL*V(NJB))/DL
GO TO 150

140 DL=E*WL(C*GL+WU+U)=CL*EL(NJB)-BL*FL(NIB)
EL(N)=(F-W*G)/DL
FL(N)=(H-W*G)/DL
GL(N) = (SU*W*(WU+U))/DL
SUP=0.0
IF (K.NE.KO) SUP=SU*PHI(NKA)
RES=B*PHI(NIB)-D*PHI(NJB)-E*PHI(N)-F*PHI(NJA)-H*PHI(NIA)-SUP=WEL
V(N)=(RES-BL*V(NIB)-CL*V(NJB))/DL
CONTINUE

--- BACK SUBSTITUTE FOR VECTOR XI ---

DO 160 K=1,KO
K3=K0-K+1
DO 160 I=1,I2
I3=I0-I
DO 160 J=1,J2
J3=J0-J
N=I3*(J3-1)*I0+(K3-1)*NIJ+I
IF (T(N).EQ.0.0.,OR.S(N).LT.0.) GO TO 160
GLXI=0.0
IF (K3.NE.KO) GLXI=GL(N)*XI(N+NIJ)
XI(N)=V(N)-EL(N)*XI(N+I0)-FL(N)*XI(N+1)-GLXI
CONTINUE

--- COMPARE MAGNITUDE OF CHANGE WITH CLOSURE CRITERIA ---

TCHK=ABS(XI(N))
IF (TCHK.GT.8I6) BIG=TOHK
PHI(N)=PHI(N)+XI(N)
CONTINUE
IF (BIO.GT.ERR) TEST=1.
TEST3(IT+1)=BIG
IF (TEST.EQ.0.) RETURN
GO TO 60

--- SKIP COMPUTATIONS IF NODE OUTSIDE AQUIFER ---

IF (T(N).EQ.0.0.,OR.S(N).LT.0.) GO TO 200

--- COMPUTE COEFFICIENTS ---

D=TR(NJB)/DELX(J)
E=TR(N)/DELX(J)
F=TR(N)/DELY(I)
G=TC(NJB)/DELY(I)
H=TC(N)/DELY(I)
SU=0.0
2=0.0
IF (K.NE.K0) Z=TK(NKB)/DELZ(K)
IF (K.NE.K0) SU=TK(N)/DELZ(K)

VI-15
RHO=S(N)/DELT
QR=0.
IF (K.NE.KO) GO TO 180
IF (IGRE.EQ.ICHK(7)) QR=QRE(I*(J-1)+10)

---SIP REVERSE ALGORITHM---
---FORWARD SUBSTITUTE, COMPUTING INTERMEDIATE VECTOR V---
180 E=-B-D-F-H-SU-Z-RHO
BL=N/(1.*W*(EL(NIA)*GL(NIA)))
CL=D/(1.*W*(FL(NJB)*6L<NJB)))
C=BL*EL(NIA)
G=CL*FL(NJB)
W=CL*GL(NJB)
U=BL*GL(NIA)
IF (K.EQ.KO) GO TO 190
AL=SU/(L.*W*(EL(NKA)*FL(NKA)))
A=AL*EL(NKA)
TU=AL*FL(NKA)
DL=E*(C*(C*A+WU+TU)+AL*GL(NKA)+BL*FL(NIA)+CL*EL(NJB)
EL(N)=(F-W*(C*A))/DL
FL(N)=(B-W*(G+TU))/DL
GL(N)=(Z-W*(WU+U))/DL
ZPHI=0.00
IF (K.NE.1) ZPHI=ZPHI(NKB)
RES=0*PHI(NIB)-D*PHI(NJB)-E*PHI(NJA)-H*PHI(NIA)-SU*PHI(NSP)
1L(N)=ZPHI=WELL(N)-RHO*OLD(N)-QR
V(N)=(RES-AL*V(NKA)-BL*V(NIA)-CL*V(NJB))/DL
GO TO 200
190 DL=E*(C*(C*A+WU+U)+BL*FL(NIA)+CL*EL(NJB)
EL(N)=(F-W*C)/DL
FL(N)=(B-W*G)/DL
GL(N)=(Z-W*(WU+U))/DL
ZPHI=0.00
IF (K.NE.1) ZPHI=ZPHI(NKB)
RES=0*PHI(NIB)-D*PHI(NJB)-E*PHI(NJA)-H*PHI(NIA)-ZPHI=WEL
1L(N)=RHO*OLD(N)-QR
V(N)=(RES-AL*V(NKA)-BL*V(NIA)-CL*V(NJB))/DL
GO TO 200
200 CONTINUE

---BACK SUBSTITUTE FOR VECTOR XI---
DO 210 K=1,KO
DO 210 I=2,J1
DO 210 J=J1,J2
J3=J0-J
N=I*(J3-1)+10*(K-1)+NIJ
IF (T(N).EQ.0..OR.S(N).LT.0.) GO TO 210
GLXI=0.00
IF (K.NE.1) GLXI=GL(N)*XI(N-1)-XI(N-1)*XI(N)+GLXI
XI(N-1)+XI(N)+FL(N)*XI(N-1))=GLXI
210 CONTINUE

---COMPARE MAGNITUDE OF CHANGE WITH CLOSURE CRITERIA---
TCHK=ABS(XI(N))
IF (TCHK.GT.BIG) BIG=TCHK
PHI(N)=PHI(N)+XI(N)
210 CONTINUE

IF (BIG.GT.ERR) TEST=1.

VI-16
TEST3(IT+1)=BIG
IF (TEST.EQ.0.) RETURN
GO TO 60

***FORMATS***

220 FORMAT ('0EXCEEDED PERMITTED NUMBER OF ITERATIONS/* ','39('**'))
230 FORMAT ('/1H0,15.22H ITERATION PARAMETERS!6E15.7/6E15.7'/)
240 FORMAT ('**44X,'SOLUTION BY THE STRONGLY IMPLICIT PROCEDURE*/45X')
 143('S')
END
SUBROUTINE COEF( PHI, STRT, OLD, TS, TR, TC, TK, WELL, DELX, DELY, DELZ, FACT)

1. PERM, BOTTOM, GRE)

COF 10

C --------------- COF 20

C COMPUTE COEFFICIENTS COF 30

C --------------- COF 40

C SPECIFICATIONS: COF 50

C REAL *8PHI COF 60

DIMENSION PHI(IO,JO,KO), STRT(IO,JO,KO), OLD(IO,JO,KO), T(I0,JO,KO) COF 90

C SPECIFICATIONS: COF 100

C COMMON /IMEGRX/ IO,JO,KO,II,J1,K1,II,J1,K,NPER,KTM,ITMAX,LENGTH,KP,NCDF 110

C COMMON /SPARAM/ TMAX,CDLT,DEL T,ERR,TST,TSUM,SP,T Q,PER, K,NUM, QRE(IQ,JQ) COF 120

COMMON /SARRAY/ ICHK(13),LEVEL1(9),LEVEL2(9) COF 130

RETURN COF 140

C --------------- COF 150

C ENTRY TCOF COF 160

C --------------- COF 170

C ENTRY TCOF COF 180

C --------------- COF 190

C ENTRY TCOF COF 200

C --------------- COF 210

C ENTRY TCOF COF 220

C ENTRY TCOF COF 230

C ENTRY TCOF COF 240

C ENTRY TCOF COF 250

C ENTRY TCOF COF 260

C ENTRY TCOF COF 270

C ENTRY TCOF COF 280

C ENTRY TCOF COF 290

C ENTRY TCOF COF 300

C ENTRY TCOF COF 310

C ENTRY TCOF COF 320

C ENTRY TCOF COF 330

C ENTRY TCOF COF 340

C ENTRY TCOF COF 350

C ENTRY TCOF COF 360

C ENTRY TCOF COF 370

C ENTRY TCOF COF 380

C ENTRY TCOF COF 390

C ENTRY TCOF COF 400

C ENTRY TCOF COF 410

C ENTRY TCOF COF 420

C ENTRY TCOF COF 430

C ENTRY TCOF COF 440

C ENTRY TCOF COF 450

C ENTRY TCOF COF 460

C ENTRY TCOF COF 470

C ENTRY TCOF COF 480

C ENTRY TCOF COF 490

C ENTRY TCOF COF 500

C ENTRY TCOF COF 510

C ENTRY TCOF COF 520

C ENTRY TCOF COF 530

C ENTRY TCOF COF 540

C ENTRY TCOF COF 550

C ENTRY TCOF COF 560

VI-18
DO 40 J=1,J1
  IF (T(I,J,K) .EQ. 0.) GO TO 40
  IF (T(I,J+1,K) .EQ. 0.) GO TO 30
  TR(I,J,K) = (2.*T(I,J+1,K) + T(I,J,K)) / (T(I,J+1,K) + T(I,J,K))
  IDELX(J) = FACT(K+1)
  30 IF (T(I+1,J,K) .EQ. 0.) GO TO 40
  TC(I,J,K) = (2.*T(I+1,J,K) + T(I,J,K)) / (T(I+1,J,K) + T(I,J,K))
  IDELY(I) = FACT(K+2)
  40 CONTINUE
  IF (K .EQ. 1. OR. ITK .EQ. ICHK(10) .OR. N3 .EQ. 0) RETURN
  DC 50 K=N4+K1
  DO 50 I=2*I1
    DO 50 J=2*J1
      IF (T(I,J,K+1) .EQ. 0.) GO TO 50
      T1 = T(I,J,K) * FACT(K+3)
      T2 = T(I,J,K+1) * FACT(K+4)
      TK(I,J,K) = (2.*T2 + T1) / (T1 + DELZ(K+1) + T2 + DELZ(K))
  50 CONTINUE
  RETURN
C
60 FORMAT ('='*120(***), 'WELL', '2I3', ' IN LAYER', 'I3', ' GOES DRY', '20(***))
70 FORMAT ('='*120(***), 'NODE', '2I3', ' IN LAYER', 'I3', ' GOES DRY', '20(***))
END
SUBROUTINE CHECKI(PHI,STRT,QLD,T,S,TR,TC,TK,WELL,DELX,DELY,OELZ,FAC)
C
C COMPUTE A VOLUMETRIC BALANCE
C
C SPECIFICATIONS:
C
REAL *8PHI
C
DIMENSION PHI(IOrjOtKO), STRT(IO,JOrKO), QLD(IO,JOrKO), T(IO,JOrKO)
1), S(IO,JOrKO), TR(IO,JOrKO), TC(IO,JOrKO), TK(IKtJKtK5), WELL(IO,CHK)
10, DELX(J0), DELY(IO), DELZ(IO), FACT(KO+3), JFLO(NCH,3), FLO
3W(NCH), QRE(IQtJQ), IQQ(40,38)
C
COMMON /INTEGR/ IO,J0,KO,IKtJKtKtI,IPtJK,ITMAX,LNGTH,KP,NCHK
1WEL*NUMT,FINAL,ITtK,HEAD,PIX,IFLO,ITRM,IT2,J2,K2,ITMAX,ITMX1,NCCK
2H, IIK1, IIK2, IHEAD, IPMC, IPF, IPtI, IPJ, IQ, IQQ, IQ3, IQ4, IQ5
3, IQ6, IQ7, IQ8, IQ9, IQ10, IQ11, IQ12, IQ13, IQ14, IQ15, IQ16
C
COMMON /SPARAH/ TNAX,CDLT,OELT,ERR,TEST,SUM,TUMP,FLUXNT
COMMON /SARRAY/ ICHK(13),LEVEL1(9),LEVEL2(9)
COMMON /CK/ ETFLUX,STORT,ORET,CHST,CHOT,FLUXT,PUMPT,CFLUX,FLXNT
C
RETURN
C
ENTRY CHECK
C
C - INITIALIZE VARIABLES -
PUMP=0.
STOR=0.
FLUX=0.0
CH0=0.0
OD2=0.0
QREFLX=0.
CFLUX=0.
FLUX=0.
ETFLUX=0.
FLXN=0.0
II=0
C
C ---COMPUTE RATES, STORAGE AND PUMPA6E FOR THIS STEP---
DO 220 K=1,KO
DO 220 I=2,II
DO 220 J=2,J1
IF (T(I,JtK).EQ.0.) GO TO 220
AREA=DELY(I)*DELY(J)
IF (S(I,JtK).GE.0.) GO TO 220
FLOW(II)=FLOW(II)+X
C
C ---COMPUTE FLOW RATES TO AND FROM CONSTANT HEAD BOUNDARIES---
II=II+1
FLOW(II)=0.
JFLO(II+1)=K
JFLO(II+2)=I
JFLO(II+3)=J
IF (S(I,J-1tK).LT.0. OR T(I,J-1tK).EQ.0.) GO TO 30
H=(PHI(I,J+1tK)-PHI(I,J-1tK))*TR(I,J-1tK)*DELY(I)
FLOW(II)=FLOW(II)+X

VI-20
IF (X) 10*30*20
10 CND1=CDN1*X
GO TO 30
20 CD2=CD2*X
30 IF (S(I*J*K)+LT_.0._OR_T(I*J*K)+EQ._0.) GO TO 60
X=(PHI(I*J*K)-PHI(I*J*K))*DELY(I)*TR(I*J*K)
FLOW(I)=FLOW(I)*X
IF (X) 40*40*50
40 CND1=CND1*X
GO TO 60
50 CD2=CD2*X
60 IF (K=EQ._1.) GO TO 90
IF (S(I*J*K)+LT_.0._OR_T(I*J*K)+EQ._0.) GO TO 90
X=(PHI(I*J*K)-PHI(I*J*K))*TR(I*J*K)*AREA*2/(DELZ(K)+DELZ(K+1))
FLOW(I)=FLOW(I)*X
IF (X) 70*70*60
70 CND1=CND1*X
GO TO 90
80 CD2=CD2*X
90 IF (K=EQ._K0.) GO TO 120
IF (S(I*J*K)+LT_.0._OR_T(I*J*K)+EQ._0.) GO TO 120
X=(PHI(I*J*K)-PHI(I*J*K))*TR(I*J*K)*AREA*2/(DELZ(K)+DELZ(K+1))
FLOW(I)=FLOW(I)*X
IF (X) 100*100*110
100 CND1=CND1*X
GO TO 120
110 CD2=CD2*X
120 IF (S(I*J*K)+LT_.0._OR_T(I*J*K)+EQ._0.) GO TO 150
X=(PHI(I*J*K)-PHI(I*J*K))*DELX(J)
FLOW(I)=FLOW(I)*X
IF (X) 130*130*140
130 CND1=CND1*X
GO TO 150
140 CD2=CD2*X
150 IF (S(I*J*K)+LT_.0._OR_T(I*J*K)+EQ._0.) GO TO 220
X=(PHI(I*J*K)-PHI(I*J*K))*DELX(J)
FLOW(I)=FLOW(I)*X
IF (X) 160*160*170
160 CND1=CND1*X
GO TO 220
170 CD2=CD2*X
GO TO 220

--- RECHARGE AND WELLS ---
180 IF (K=EQ._K0.AND._IGRE.EQ._ICMK(7)) QREFLX=QREFLX+QRE(I*J)*AREA
180 IF (WELL(I*J*K)) QREFLX=QREFLX+QRE(I*J)*AREA
190 PUMP=PUMP+WELL(I*J*K)*AREA
GO TO 210
200 CFLUX=CFLUX+WELL(I*J*K)*AREA

--- COMPUTE VOLUME FROM STORAGE ---
210 STOR=STOR+(I*J*K)-(OLD(I*J,K)+PHI(I*J,K))*AREA
220 CONTINUE

VI-21
---COMPUTE CUMULATIVE VOLUMES, TOTALS, AND DIFFERENCES---

CFLXPT = 0.0
STOR = STOR + STOR
STOR = STOR / DELT
QRET = QRET + QREFLX * DELT
CHDT = CHDT + CHD1 * DELT
CHST = CHST + CHD2 * DELT
PUMPT = PUMPT - PUMP * DELT
QRET = QRET * CFLUX * OELT
CHOT = CHDT - CHD1 - OELT
CHST = CHST * CHD2 * OELT
PUMPT = PUMPT - PUMP * DELT
CFLUXT = CFLUXT * CFLUX * OELT
TCTL1 = TCTL1 + ORET * CFLUX * STOR * CFLUX
TCTL2 = TCTL2 + PUMP * ETFLUX * FLUX
SLMR = QREFLX * CFLUX * CHD2 * CHD1 * PUMP * ETFLUX * FLUX * STOR
DIFF = TOTL2 - TOTL1
PERCNT = DIFF / TOTL2 * 100.

230 RETURN

---PRINT RESULTS---

ENTRY CWRITE

WRITE (6, 260) STOR, QREFLX, STOR, CFLUX, QRET, PUMP, CFLUX, ETFLUX, CHST
1, CFLXPT, CHD2, TOTL1, CHD1, CFLUX, ETFLUX, CHDT, CHOT, PUMPT, ETFLUX, TOTL1
2L2, DIFF, PERCNT
IF (NCH.EQ.0) GO TO 240
WRITE (6, 270)
WRITE (6, 280) ((JFL0(I + J1), J = 1, 3), FLOW(I), I = 1, NCH)

---COMPUTE VERTICAL FLOW---

240 X = 0
Y = 0.
IF (K0.EQ.1) RETURN
DO 250 I = 2, 11
DO 250 J = 2, J1
X = X + PHI(I, J1) - PHI(I, J2) * TK(I, J1) * DELX(J) * DELY(I) * 2. / (DELZ(1) * DELZ(K1))
1, DELZ(2)
Y = Y + PHI(I, J1) - PHI(I, J2) * TK(I, J1) * DELX(J) * DELY(I) * 2. / (DELZ(K1) * DELZ(0))
1, DELZ(0)
WRITE (6, 290) Y, X
RETURN

---FORMATS---

260 FORMAT (10, 10X, 'CUMULATIVE VOLUMES: TOTL1', 16X, 'L = 3', 23X, 'RATES', 'CHK1', '160')
2RATES = 16X, 'L = 3', 23X, 'RATES', 'CHK1', '160')
2RCES: 1, 16X, 'STORAGE = ', F20.4 / 20X, 'SOU', 'CHK1', '160')
3, 'STORAGE = ', F20.4 / 20X, 'SOU', 'CHK1', '160')
3, 'STORAGE = ', F20.4 / 20X, 'SOU', 'CHK1', '160')
40, 2, 41X, 'PUMPING = ', F20.4 / 20X, 'SOU', 'CHK1', '160')
40, 2, 41X, 'PUMPING = ', F20.4 / 20X, 'SOU', 'CHK1', '160')
5ANSPIRATION = ', F20.4 / 20X, 'SOU', 'CHK1', '160')
5ANSPIRATION = ', F20.4 / 20X, 'SOU', 'CHK1', '160')
VI-23
SUBROUTINE PRNTAI(PHI, STRT, T, S, WELL, DELX, DELY)

--- PRINT MAPS OF DRAWDOWN AND HYDRAULIC HEAD ---

SPECIFICATIONS:
REAL PHIT, Z, XLABEL, YLABEL, TITLE, XN1, MESUR
REAL K

DIMENSION PHI(10*J0*K0), STRT(I0,J0,K0), S(I0,J0,K0), WELL(I0,J0,K0)

COMMON /INTEGR/I0,J0,K0,I1,J1,K1,I,J,K,NPER,KTH,ITMAX,LENGTH,KP,NPR
1WEL,NUMT,IFINAL,IT, IHEAD, IDRAW, IFLO, IERR, I2,J2,K2, IMAX, ITMX, NCPR
2H, ICK1, IDK2, INAT, IQ, IQJ, IQK, IK5, IP1, IP1U, ITK

COMMON /PR/ XLABEL(3), YLABEL(6), TITLE(6), XN1, MESUR, PRNT(122), BLANK
1(60), DIGIT(122), VF1(6), VF2(6), VF3(7), XSCALE, DINCH, SYM(17), XN(100), PRN
2H, LIM, NA(4), XN1, N2, N3, YSCALE, FACT1, FACT2

--- INITIALIZE VARIABLES FOR PLOT ---

ENTRY MAP

DO 10 J=2+J1
10 WIDTH=WIDTH+DELX(J)
DO 20 I=2+I1
20 YDIM=YDIM+DELY(I)
30 XSF=DINCH*XSCALE
YSF=DINCH*YSCALE
NYD=YDIM/YSF
IF (NYD<YSF LE.YDIM-DELY(I1)/2.) NYD=NYD+1
IF (NYD>LE.12) GO TO 40
DINCH=YDIM/(12.*YSCALE)
WRITE (6,330) DINCH
390 IF (YSCALE.LT.1.0) WRITE (6,340)
GO TO 30
40 NXD=WIDTH/XSF
IF (NXD<XSFSF.LE.WIDTH-DELX(J1)/2.) NXD=NXD+1
N4=NXD-N1+1
N5=NXD+1
N6=NYD+1
N8=N2*NYD+1
NA(1)=N4/2+1
NA(2)=N4/2
NA(3)=N4/3
NC=(N3=N8-10)/2
ND=NC+N8
NE=MAXO(N5,N6)
VF1(3)=DIGIT(ND)
VF2(3)=DIGIT(ND)
VF3(3)=DIGIT(NC)
XLABEL(3)=MESUR

RETURN
YLABEL(6)=MESUR
DO 60 I=1,NE
NX=N5-I
NY=I-1
IF (NY.GE.N6) GO TO 50
YH(I)=YSF*NY/YSCLAE
50 IF (NX.LT.O) GO TO 60
XH(I)=XSF*NX/YSCLAE
60 CONTINUE
RETURN
C
******************************************************************************
ENTRY PRNTA(NG,LA)
******************************************************************************
C ---VARIABLES INITIALIZED EACH TIME A PLOT IS REQUESTED---
DIST=WIDTH=DELX(J1)/2,
JJ=1
LL=1
Z=NXD*XSF
IF (NG.EQ.l) WRITE (6,300) (TITLE(I)*I=1,3),LA
IF (NG.EQ.2) WRITE (6,300) (TITLE(I)*I=4,6),LA
DO 290 I=1,N4
LOCATE X AXES
IF (I.EQ.1.LO.1.EQ.N4) GO TO 70
PRNT(1)=SYM(12)
PRNT(N8)=SYM(12)
IF ((I-1)/N1.EQ.I-1) GO TO 90
PRNT(1)=SYM(14)
PRNT(N8)=SYM(14)
GO TO 90
LOCATE Y AXES
70 DC 80 J=1,N8
IF ((J-1)/N2.EQ.J-1) PRNT(J)=SYM(14)
80 IF ((J-1)/N2.NE.J-1) PRNT(J)=SYM(13)
C ---LOCATE Y AXES---
C ---LOCATE Y AXES---
90 IF (DIST.LT.O.OR.DIST.LT.Z-XN1*XSF) GO TO 240
YLEN=DELAY(2)/2,
DO 220 L=2,I
J=YLEN*N2/YSF+I.5
IF (T(LJJLA)*EQ.0.) GO TO 160
IF (S(LJJLA)*LT.0.) GO TO 210
INDX3=0
GO TO (100,110),NG
100 K=(STRT(LJJLA)-PHI(LJJLA))*FACT1
C =TO CYCLE SYMBOLS FOR DRAWDOWN, REMOVE C FROM COL. 1 OF NEXT CARD=PRN1050
C K=AMOD(K,10.)
GO TO 120
110 K=PHI(LJJLA)*FACT2
120 IF (K) 130+160+140
130 IF (J-2.GT.O) PRNT(J-2)=SYM(13)
N=K*5
IF (N.LT.100) GO TO 150

VI-25
GO TO 190

140 N=N+.5
  IF (N.LT.100) GO TO 150
  IF (N.GT.999) GO TO 190
  INDX3=N/100
  IF (J-2.GT.0) PRNT(J-2)=SYM(INDX3)
  N=N-INDX3*100
150 INDX1=MOD(N+10)
  IF (INDX1.EQ.0) INDX1=10
  GOTO 190
C
  TO CYCLE SYMBOLS FOR DRAWDOWN, REMOVE C FROM COL. 1 OF NEXT CARD
C
  IF (NG.EQ.1) GO TO 170
  INDX2=N/10
  IF (INDX2.EQ.0) GO TO 160
  INDX2=10
  IF (INDX3.EQ.0) INDX3=15
  GO TO 180
160 INDX1=15
170 INDX2=15
180 IF (J-1.GT.0) PRNT(J-1)=SYM(INDX2)
  PRNT(J)=SYM(INDX1)
  GO TO 220
190 DO 200 JJ=1+3
  JJ=J-3+II
200 IF (J.I.EQ.0) PRNT(JJ)=SYM(11)
210 IF (J.I.EQ.0) (LL,JJ)=(IT,JJ,0.) PRNT(JJ)=SYM(16)
220 YLEN=YLEN+(DELY(L)+DELY(JJ-1))/2.
230 DIST=DIST+(DELX(JJ-1)*DELY(JJ-1))/2.
240 CONTINUE

C
---PRINT AXES LABELS, AND SYMBOLS---
  IF (I-NA(LL).EQ.0) GO TO 260
  IF ((I-1)/N1=N1-(I-1)) 270,250,270
250 WRITE (6,VF1) (BLANK(J)+J=1,NC),(PRNT(J)+J=1,N8)+(XN(1*(I-1)/6))
  GO TO 280
260 WRITE (6,VF2) (BLANK(J)+J=1,NC),(PRNT(J)+J=1,N8)+XLABEL(LL)
  LL=LL+1
  GO TO 280
270 WRITE (6,VF2) (BLANK(J)+J=1,NC),(PRNT(J)+J=1,N8)
C
---COMPUTE NEW VALUE FOR Z AND INITIALIZE PRNT---
280 Z=Z+2.*XN1*XSF
  DO 290 J=1,N8
290 PRNT(J)=SYM(15)
C
---NUMBER AND LABEL Y AXIS AND PRINT LEGEND---
  WRITE (6,VF3) (BLANK(J)+J=1,NC),(YN(I),I=1,N6)
  WRITE (6,VF3) (YLABEL(I),I=1,N6)
  IF (NG.EQ.1) WRITE (6,310) FACT1
  IF (NG.EQ.2) WRITE (6,310) FACT2
  RETURN
C
---FORMATS---
C
---
C PRN1690
C PRN1700
C PRN1710
C PRN1720
C 300 FORMAT ('1*49X*3A8*"LAYER",I4//)
C 310 FORMAT ('*EXPLANATION*/"+11(=)"/" R = CONSTANT HEAD BOUNDARY"/PRN1730
C 320 FORMAT ('0*39X*6A8) PRN1740
C 330 FORMAT ('0*25X*10('')) TO FIT MAP WITHIN 12 INCHES, D INCH REVIS
C 340 FORMAT ('0*45X*"NOTE: GENERALLY SCALE SHOULD BE > OR = 1.0") PRN1760
C END PRN1770
C
VI-27