COMPUTER PROGRAM FOR SOLVING GROUND-WATER FLOW EQUATIONS BY THE PRECONDITIONED CONJUGATE GRADIENT METHOD

By Logan K. Kuiper

U.S. GEOLOGICAL SURVEY

Water-Resources Investigations Report 87-4091

Austin, Texas
1987

DEPARTMENT OF THE INTERIOR

DONALD PAUL HODEL, Secretary

U.S. GEOLOGICAL SURVEY

Dallas L. Peck, Director

For more information
write to:

Project Chief
U.S. Geological Suvey
Gulf Coast RASA
N. Shore Plaza Bldg., Rm. 104
55 North Interregional Hwy.
Austin, Texas 78702

Copies of this report can
be purchased from:

U.S. Geologicl Survey
Books and Open-File Reports
Federal Center, Bldg., 41
Box 25425
Denver, Colorado 80225
Telephone (303) 236-7476

CONTENTS

ILLUSTRATIONS

TABLES

COMPUTER PROGRAM FOR SOLVING GROUND-WATER FLOW EQUATIONS BY THE PRECONDITIONED CONJUGATE GRADIENT METHOD

By Logan K. Kuiper

ABSTRACT

This report documents a numerical code for use with the U.S. Geological Survey modular three-dimensional finite-difference ground-water flow model. The code uses the preconditioned conjugate gradient method for the solution of the finite difference approximating equations generated by the modular flow model. These equations are a system of simultaneous linear equations except when the river, drain, or evapotranspiration packages of the modular model are being used, in which case they are a system of simultaneous nonlinear equations. When these equations are linear, they are solved by the basic preconditioned conjugate gradient method as available in the literature. Five preconditioning types may be chosen: three different types of incomplete Choleski, point Jacobi, or block Jacobi. When the approximating equations are nonlinear, the solution method is that of Picard preconditioned conjugate gradient with the same preconditioning choices. Either a head change or residual error criteria may be used as an indicator of solution accuracy and iteration termination.

The use of the computer program that performs the calculations in the numerical code is emphasized. Detailed instructions are given for using the computer program, including data entry formats and the method of linking the program into the modular model. A sample data listing and listing of the Fortran program are included.

INTRODUCTION

This report documents a numerical code for use with the U.S. Geological Survey modular three-dimensional finite-difference ground-water flow model (McDonald and Harbaugh, 1984). The code uses the preconditioned conjugate gradient method for the solution of the finite difference approximating equations generated by the modular flow model. These equations are a system of simultaneous linear equations except when the river, drain, or evaporation packages of the modular model are being used, in which case they are a system of simultaneous nonlinear equations. When these equations are linear, they are solved by the basic preconditioned conjugate gradient method as available in the literature. Five preconditioning types may be chosen: three different types of incomplete Choleski, point Jacobi, or block Jacobi. When the approximating equations are nonlinear the solution method is that of Picard preconditioned conjugate gradient with the same preconditioning choices. Either a head change or residual error criteria may be used as an indicator of solution accuracy and iteration termination.

The preconditioned conjugate gradient (PCG) method as presented is sometimes faster than the strongly implicit procedure (SIP) and slice-succesive overrelaxation (SOR) available in the modular model (Kuiper, 1981, 1987). It is frequently faster on problems having a wide variation in the conductances between model nodes, or on problems having a complex geometry such as pinched out layers. The PCG method presented has the advantage of not requiring any convergence parameters. The user has the option of using residual error as a criteria for iteration termination. This option assures that the flow rate into each cell is equal to the flow rate out of the same cell, to within a small amount selected by the user.

PRECONDITIONED CONJUGATE GRADIENT PACKAGE

Description and Use

The preconditioned conjugate gradient (PCG) method to be presented here is an iterative method for solving a system of simultaneous linear or non-linear equations.

Finite difference discretization of the ground-water flow equation gives a set of finite difference approximating equations (McDonald and Harbaugh, 1984), the solution of which gives an approximate solution to the ground-water flow equation. For a cell location i,j,k the finite-difference equation (McDonald and Harbaugh, 1984, p. 30, equation 27) is:

$$CV_{i,j,k-\frac{1}{2}}h_{i,j,k-1} + CC_{i-\frac{1}{2},j,k}h_{i-1,j,k} + CR_{i,j-\frac{1}{2},k}h_{i,j-1,k}$$

$$+ (-CV_{i,j,k-\frac{1}{2}} - CC_{i-\frac{1}{2},j,k} - CR_{i,j-\frac{1}{2},k}$$

$$-CR_{i,j+\frac{1}{2},k} - CC_{i+\frac{1}{2},j,k} - CV_{i,j,k+\frac{1}{2}} + HCOF_{i,j,k})h_{i,j,k}$$

$$+CR_{i,j+\frac{1}{2},k}h_{i,j+1,k} + CC_{i+\frac{1}{2},j,k}h_{i+1,j,k}$$

$$+ CV_{i,j,k+\frac{1}{2}}h_{i,j,k+1} = RHS_{i,j,k} \qquad (1)$$

where $CV_{i,j,k-\frac{1}{2}}$ is the conductance (McDonald and Harbaugh, 1984, p. 16) between nodes i,j,k and i,j,k-1 , $CV_{i,j,k+\frac{1}{2}}$ is the conductance between nodes i,j,k and i,j,k+1, and corresponding definitions apply to the CC and CR terms. The hydraulic head at node i,j,k is $h_{i,j,k}$, the hydraulic head at node i,j,k-1 is denoted by $h_{i,j,k-1}$, and so on. Conductance is defined (McDonald and Harbaugh, 1984, p. 16) as that quantity associated with a particular node face which, when multiplied by the difference between the heads of those two nodes lying on either side, gives the flow across the node face. An equation like (1) is written for each cell in the finite-difference grid. This grid fills the volume within which the solution to the ground-water flow equation is to be approximated. Equation (1) expresses the relation among the heads h at node

i,j,k and at each of the six adjacent nodes at the end of a time step. Note that head at any node appears in the equation for that node and also in the equation for adjoining nodes. Thus, the equations are coupled and must be solved simultaneously. It is convenient to write equation (1) as

$$Z_{i,j,k}h_{i,j,k-1} + B_{i,j,k}h_{i-1,j,k} + D_{i,j,k}h_{i,j-1,k} + E_{i,j,k}h_{i,j,k}$$

$$+ F_{i,j,k}h_{i,j+1,k} + H_{i,j,k}h_{i+1,j,k} + S_{i,j,k}h_{i,j,k+1} = Q_{i,j,k} \qquad (2)$$

(McDonald and Harbaugh, 1984, p. 370, equation 80), which in matrix form becomes

$$Ah = q \; . \qquad (3)$$

A is a square matrix and h and q are vectors. The components of the vector h are the hydraulic heads $h_{i,j,k}$. The components of the vector q are the "source" terms $Q_{i,j,k}$ of equation (2). Figure 1 shows the elements of the matrix A and the vectors h and q. Notice that nonzero elements in A appear only on seven diagonals (fig. 2). Because the number of nonzero elements in A is small compared to the total number of elements, the matrix is said to be sparse.

The coefficients in equation (2) all have the index i,j,k to show that they belong to the equation for node i,j,k. Furthermore, the Z coefficient for the equation at node i,j,k ($Z_{i,j,k}$), is equal to $CV_{i,j,k-\frac{1}{2}}$, which is the same as the S coefficient for the equation at node $i,j,k-1$ ($S_{i,j,k-1}$), so that (Mcdonald and Harbaugh, 1984, p. 371)

$$Z_{i,j,k} = S_{i,j,k-1} \; . \qquad (4)$$

Similiarly,

$$B_{i,j,k} = H_{i-1,j,k} \qquad (5)$$

and

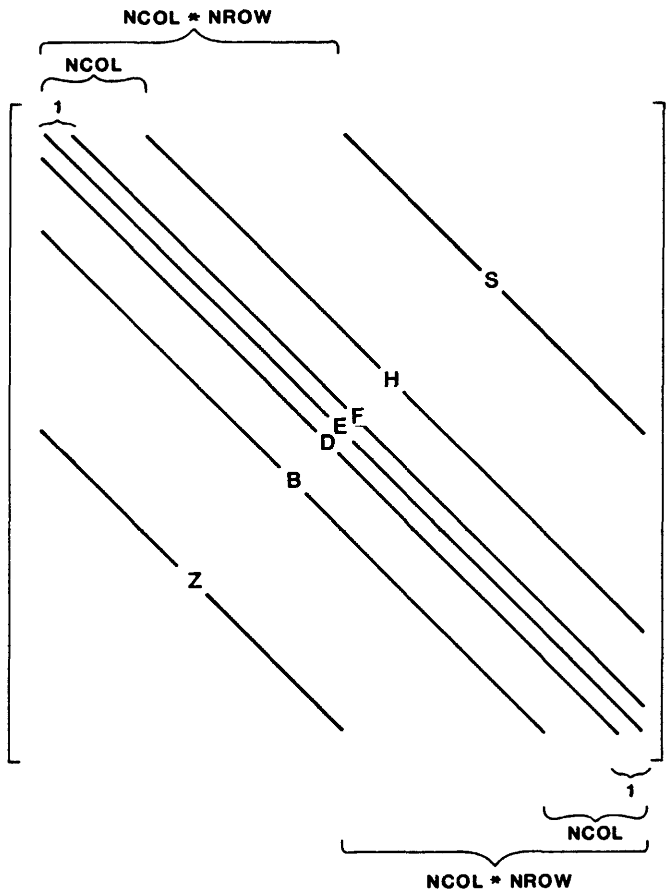
$$D_{i,j,k} = F_{i,j-1,k} \; . \qquad (6)$$

Thus, the matrix A is symmetric. Because $E_{i,j,k}$ is equal to sum of $-Z_{i,j,k}$, $- B_{i,j,k}$, $-D_{i,j,k}$, $-F_{i,j,k}$, $-H_{i,j,k}$, $-S_{i,j,k}$, and $HCOF_{i,j,k}$, the negative of the matrix A is positive definite.

$$
\begin{array}{ccc}
A & h & q \\
\begin{bmatrix}
E_1 & F_1 & H_1 & O & O & O & S_1 & O & O & O & O & O \\
D_2 & E_2 & O & H_2 & O & O & O & S_2 & O & O & O & O \\
B_3 & O & E_3 & F_3 & H_3 & O & O & O & S_3 & O & O & O \\
O & B_4 & D_4 & E_4 & O & H_4 & O & O & O & S_4 & O & O \\
O & O & B_5 & O & E_5 & F_5 & O & O & O & O & S_5 & O \\
O & O & O & B_6 & D_6 & E_6 & O & O & O & O & O & S_6 \\
Z_7 & O & O & O & O & O & E_7 & F_7 & H_7 & O & O & O \\
O & Z_8 & O & O & O & O & D_8 & E_8 & O & H_8 & O & O \\
O & O & Z_9 & O & O & O & B_9 & O & E_9 & F_9 & H_9 & O \\
O & O & O & Z_{10} & O & O & O & B_{10} & D_{10} & E_{10} & O & H_{10} \\
O & O & O & O & Z_{11} & O & O & O & B_{11} & O & E_{11} & F_{11} \\
O & O & O & O & O & Z_{12} & O & O & O & B_{12} & D_{12} & E_{12}
\end{bmatrix}
\times
&
\begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \\ h_{10} \\ h_{11} \\ h_{12} \end{bmatrix}
=
&
\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \\ q_7 \\ q_8 \\ q_9 \\ q_{10} \\ q_{11} \\ q_{12} \end{bmatrix}
\end{array}
$$

Modified from MacDonald and Harbaugh, 1984

Figure 1.--Correspondence between the finite-difference equations and the matrix equation for a grid of three rows, two columns, and two layers.



From MacDonald and Harbaugh, 1984

Figure 2.--Structure of coefficient matrix showing nonzero diagonals.

Equation (3) could be written as

$$A(h^m,m)h^m = q(h^m,m) \ , \qquad (7)$$

where vector h^m is vector h at time t_m. The parenthesis indicate that the elements of the matrix A and vector q may depend on the vector h^m. An example of when the elements of matrix A depend on head is the case of a water-table aquifer. In this case, the conductance between two adjacent nodes in an aquifer depends on the saturated thickness of the aquifer in the vicinity of the nodes and, thus, on the head in the vicinity of the nodes. Therefore, the conductances CR, CC, and CV, which appear in the off-center diagonals of the matrix A are head dependent. When matrix A and vector q are h^m dependent, equation (7) is said to be nonlinear and is more difficult to solve for h^m than the linear case for which the elements of matrix A and vector q are constants.

An alternative to solving equation (7), which is done by SIP and SOR in the modular model, is to solve

$$A(h^{m-1},m-1)h^m = q(h^{m-1},m-1) \ . \qquad (8)$$

In this case, the system is linear and easily solved, but the solutions h^m, $m=0,1,2,\ldots$ may tend to be unstable. Use of equation (8) corresponds, for the water-table aquifer situation, to using the conductance between two adjacent nodes corresponding to the head h^{m-1} at time t_{m-1}, when calculating the change in head between times t_m and t_{m-1}, or in other words when calculating h^m. The PCG package allows the use of equation (7) or (8) but because of the instability mentioned, the use of equation (8) usually is not recommended except, perhaps, when equation (7) is too difficult to solve. When the problem being solved is linear, matrix A and vector q are constant, and equations (7) and (8) are identical.

The PCG method presented here is ideal for solving a sparse symmetric positive definite system of simultaneous linear equations. It also can be used for solving a sparse symmetric positive definite system of simultaneous nonlinear equations, such as (7), but with perhaps somewhat decreased efficiency.

An important part of the PCG method presented here is the basic PCG method for sparse symmetric positive definite linear systems, as taken from the literature (Van Der Vorst, 1982):

$$a_v = \frac{(r^v, K^{-1} r^v)}{(p_v, Ap_v)} , \tag{9}$$

$$x_{v+1} = x_v + a_v p_v , \tag{10}$$

$$r_{v+1} = r_v - a_v A p_v , \tag{11}$$

$$B_v = \frac{(r^{v+1}, K^{-1} r^{v+1})}{(r_v, K^{-1} r_v)} , \text{ and} \tag{12}$$

$$p_{v+1} = K^{-1} r_{v+1} + B_v p_v . \tag{13}$$

where

$$(x,y) = \sum_{i=1}^{N} x_i y_i$$

is the inner product of the vectors x and y. Iteration of equations (9) through (13) using v=1,2,... , and using $r_1=b-Ax_1$, $p_1=K^{-1}r_1$, and some initial choice x_1 for the solution vector gives an approximate solution to the matrix equation Ax=b, where A is a N by N symmetric positive definite matrix. The residual error vector is r=b-Ax. Matrix K is called a preconditioning or splitting matrix. It is chosen to be as nearly equal to A as possible but readily invertible. The PCG package allows for five choices of the preconditioning matrix. The first three choices for matrix K (corresponding to NPCOND=1,2,and 3) are three different types of incomplete Choleski factorization (Kershaw, 1978), where the first two differ only in the manner of treating inactive nodes. The fourth choice (NPCOND=4) is point Jacobi (Hageman and Young, 1981) for which matrix K is simply the diagonal of the matrix A. The fifth choice (NPCOND=5) is block Jacobi for which matrix K is the diagonal and the two off-center diagonals adjacent to the center diagonal of A.

The basic PCG method in equations (9) through (13) is part of the PCG method presented here. The solution of equation (7), h^m, is the head, h, at time t_m. The PCG method presented here finds an approximation to h^m iteratively. Let these sucessive approximations to h^m be denoted by $h_s{}^m$, s=1,2,...,sm. Let $h_{sm}{}^m$ denote that iteration taken to be a satisfactory approximate solution to h^m. The first estimate for h^{m+1}, $h_1{}^{m+1}$, is taken to be $h_{sm}{}^m$. To explain the way succesive iterations $h_s{}^m$ are chosen, it is necessary to break the index s into two indices, u and v, where index v changes fastest. The indices u and v go from 1 to um, and from 1 to vm(u) respectively. The procedure for finding the approximate solution $h_{sm}{}^m$, to h^m is:

Approximately solve

$$A(h_{u,1}{}^m,m)h_u{}^m = q(h_{u,1}{}^m,m) \ , \quad \text{for } u=1,2,\ldots,um \ , \quad (14)$$

where the succesive approximations to $h_u{}^m$ are denoted $h_{u,v}{}^m$, v=1,2,...,vm(u). Figue 3 ia a flowchart showing how the main program of the modular model and module PCG1AP interact to do the u and v iterations of equation (14). The flowchart shows only the major elements of module PCG1AP and only those elements of the main program relating to its connection with module PCG1AP. The iterations in v occur each time the module PCG1AP is called. Values of u are a counter of the number of times the module PCG1AP is called by the main program. The value for $h_{u,1}{}^m$ is $h_{u-1,vm(u-1)}{}^m$. Note that equation (14) is linear with respect to solving for $h_u{}^m$. The iterations in v are those of the basic PCG method as given by equations (9) through (13). The PCG method as given by (14) for the solution of (7) would be called Picard-PCG using the naming procedure of the mathematical literature on the solution of nonlinear systems (Kuiper, 1987).

In the main program in figure 3, MXITER is the chosen maximum allowable value for u, and also for MCNT, the total number of iterations used in the search for the approximate solution to h^m. The maximum allowable number for v is v_{max}. ICNVG is the variable indicating whether a suitably accurate solution to h^m has been obtained. IFLAG is the variable that indicates whether an exit from the u iteration loop in the main program is desired upon return to the main program. IFLAG causes an exit at the first return when the linear case is being solved or equation (8) is being used, or in the nonlinear case when MCNT$\geq$MXITER.

In module PCG1AP in figure 3, note that when a sufficiently small value is chosen for v_{max}, that the v loop may not be exited but v instead reaches its maximum value v_{max}, corresponding to a situation in which for a given u<um, equation (14) is not solved accurately enough to cause a v-loop exit. This situation may be understood by considering the way that the use of equation (14) implements the solution of a problem with a declining head in a water-table aquifer for some given time step m. Values of u correspond to evaluations of the conductances between nodes as determined by using heads $h_{u,1}{}^m$. For these evaluations at u of the conductances, the head decrease is determined using iterations in v. Having obtained $h_{u,vm(u)}{}^m$ where vm(u) may be equal to v_{max}, new values of the decreased conductance are determined using the just acquired decreased heads $h_{u,vm(u)}{}^m$. Then the head decline is

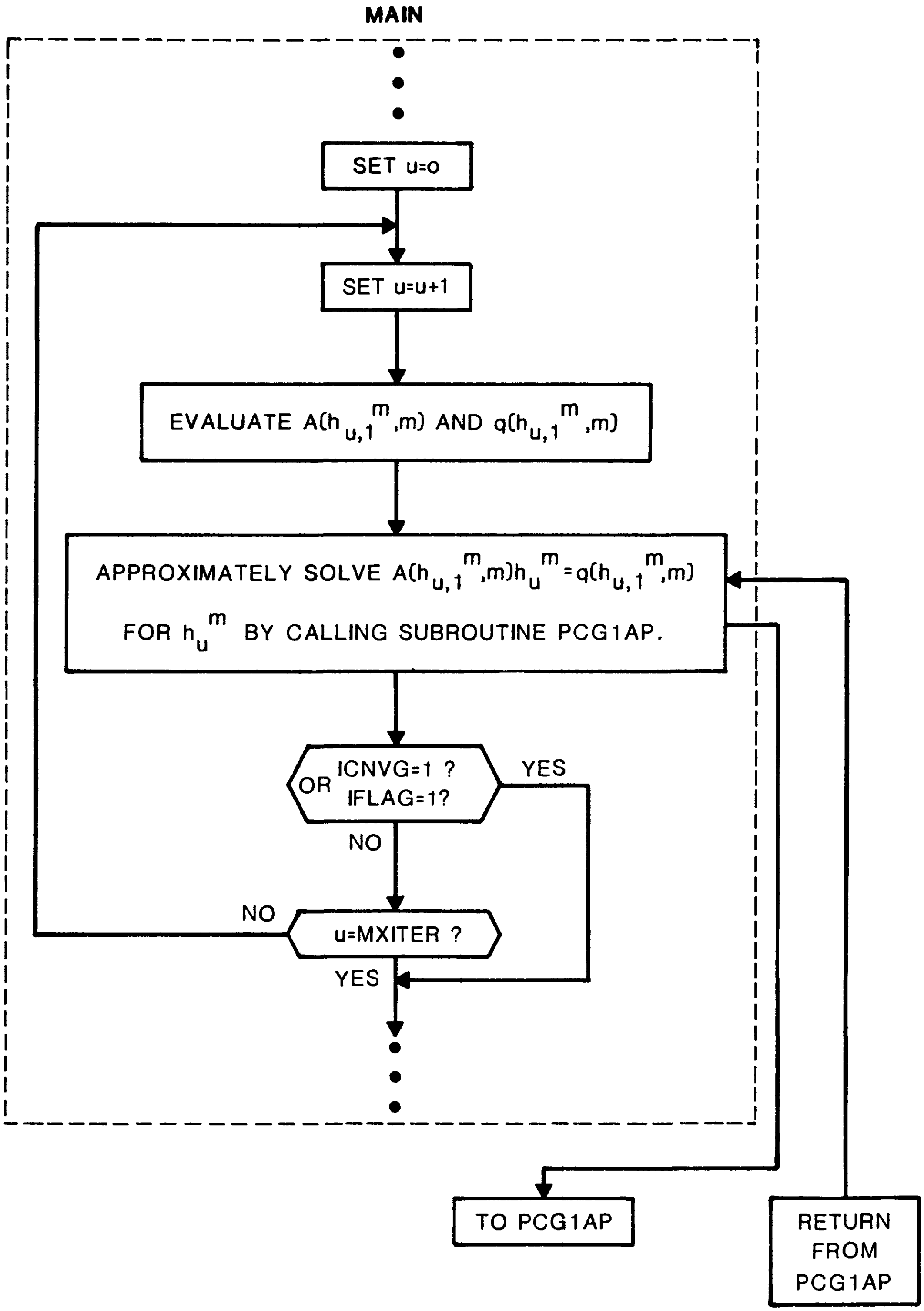


Figure 3.--Flowchart showing major elements of module PCG1AP and related parts of the main program of the modular model.

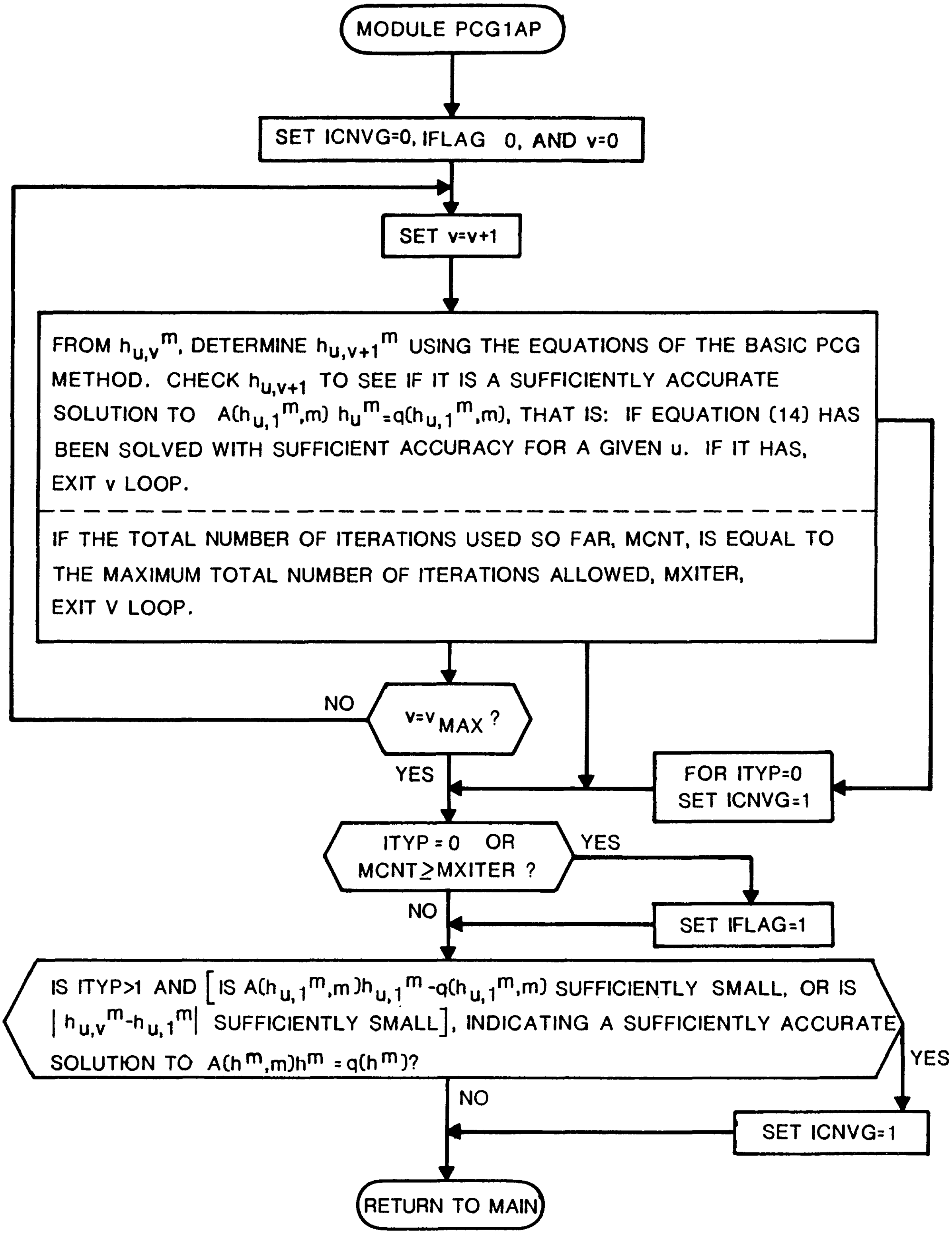


Figure 3.--(Concluded)

redetermined corresponding to these new decreased values for the conductances. The process continues in this manner. Therefore, one need not necessarily solve for head declines accurately for a given u<um because the conductances corresponding to this value for u are too large anyway. Two approaches are thus allowed: for a given evaluation of the conductances corresponding to $h_{u,1}{}^{m}$, either solve for the head accurately enough to meet some accuracy criteria, or just stop the iteration in v at some v_{max}. In many cases, problems which are nearly linear are solved with a smaller total number of iterations by choosing a large value for v_{max} resulting in the program exiting the v loop. On the other hand, extremely nonlinear problems are solved more readily by choosing v_{max} to be small.

In module PCG1AP, the choice of v_{max} is controlled by the users choice of the variable ITYP. For a choice of ITYP=0, equation (8), or in the linear case the equivalent equation (7), is solved. For choices of ITYP≥1, equation (7) is solved by means of equation (14), corresponding to the nonlinear case, using v_{max}=MXITER when ITYP=1, and v_{max}=ITYP-1 when ITYP≥2.

Input Instructions

The Preconditioned Conjugate Gradient (PCG) Package reads values from the unit specified in IUNIT(13) in the Basic (BAS) Package of the modular model.

For each simulation:

PCG1AL

1.	Data:	MXITER	NPCOND	ITYP
	Format:	I10	I10	I10

PCG1AL

2.	Data:	HCLOSE	RESERR	IWRT
	Format:	F10.0	F10.0	I10

Read only if IWRT=2:

3.	Data:	NU1(I),I=1,9
	Format:	(9I4)

Explanation of Fields Used in Input Instructions

MXITER--

is the maximum total number iterations allowed in an attempt to solve the system of finite difference equations. One hundred iterations should be sufficient for most (ITYP=0) problems.

NPCOND--

has the values 1 to 5 corresponding to the five preconditioning types which may be chosen. The first three are incomplete Choleski, the fourth is point Jacobi, and the fifth is block Jacobi. NPCOND equal 1 or 3 are common choices. On rare occasions NPCOND equal 4 or 5 could be faster than 1 or 3. NPCOND equal 2 is at present identical to NPCOND equal 1, but may be changed at a later time. The advised procedure is to use either NPCOND equal 1 or 3, or use both and compare computation times if the model is going to be run many times and computation time is important. NPCOND equal 1 is usually a bit slower than NPCOND equal 3, but it is also more stable. For those who do not wish to experiment with NPCOND, the best choice is 1.

ITYP--

is a flag indicating the type of problem solved:

0 - linear problems: LAYCON=0; river, drain, or evaporation packages are not being used. Also for nonlinear problems to be solved with equation (8). Such equation (8) solutions for nonlinear problems may be inaccurate and are therefore not recommended unless a solution cannot be obtained using ITYP$\geq$1 or SIP. For equation (8) solutions, the usual budget and flowchart calculations printed by the modular model may be innacurate, and cannot be used as a measure of solution accuracy. See page 20 for more detail.

1 - nonlinear problems with weakly nonlinear conditions.

2 - nonlinear problems with strongly nonlinear conditions. If you do not know how nonlinear the problem is, use ITYP equal 1 and 2 and compare results. See table 1 on page 21 for more detail.

HCLOSE--

is the head change criteria for convergence. When the maximum absolute head change for all nodes from the last iteration(s) is less than HCLOSE, iteration is terminated.

RESERR--

is the residual error criteria for convergence. Residual error is the flow rate into a variable head cell minus the flow rate out of the cell. When the maximum, over all the variable head cells in the modeled region, of the absolute values of the residual errors for each of the variable head cells is less than RESERR, iteration is terminated.

Both HCLOSE and RESERR may be used concurrently in which case both have nonzero values. Set HCLOSE=0 if you want to use only RESERR, and vise-versa.

IWRT--

is a flag indicating the amount of output produced regarding the numerical solution of the finite difference equations.

0 - no output is produced other than that normally provided by the modular model program.

1 - the maximum absolute head change (ER5) from the last iteration, the maximum absolute residual error (SRZ or SRZ1) for variable head cells, and the total residual error for the entire model obtained by summing the residual errors for all of the variable head cells, are produced for each time step.

2 - includes the output produced in option 1 plus an output for watching convergence which shows the numerical solution process at each time step, including the head at 3 locations specified by NU1.

NU1--

specifies the 3 locations:

1 -- J=NU1(1), I=NU1(2), K=NU1(3)

2 -- J=NU1(4), I=NU1(5), K=NU1(6)

3 -- J=NU1(7), I=NU1(8), K=NU1(9)

at which head is printed for each time step when IWRT=2. NU1 is not read when IWRT equal to 0 or 1.

Sample Input to PCG Package

Data item	Explanation	Input Records								
1	MXITER,NPCOND,ITYP	50			1			1		
2	HCLOSE,RESERR,IWRT	.1			10			2		
3	NU1	3	3	4	5	7	4	5	7	5

Module Documentation for the Preconditioned Conjugate Gradient Package

The Preconditioned Conjugate Gradient Package (PCG1) consists of three primary modules. They are:

Primary Modules

PCG1AL	Allocates space for the PCG Package work arrays.
PCG1RP	Reads control information needed by the PCG Package.
PCG1AP	Performs one or more iterations of the preconditioned conjugate gradient method.

PCG1AL

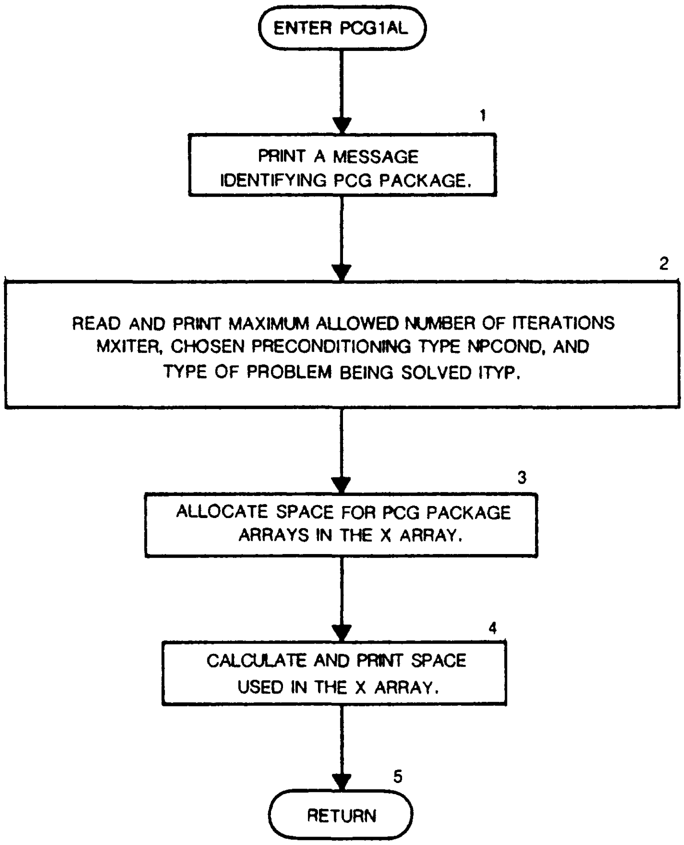
Narrative for Module PCG1AL

Module PCG1AL allocates space in the X array for the PCG Package arrays. The five arrays DT, E2, F2, G2, and VV hold intermediate results during the solution process. Each of these arrays contains one element for each model cell.

Module PCG1AL performs its functions in the following order:

1. Print a message identifying the PCG Package.
2. Read and print MXITER, NPCOND, and ITYP.
3. Allocate the required space in the X array. The X-array location pointer (ISUM) is saved in variable ISOLD prior to allocation so that the space required for the PCG Package can be calculated in step 4.
4. Calculate and print the space used in the X array. The space used by the PCG Package is ISUM-ISOLD. The total allocated by all packages so far is ISUM-1.
5. Return.

Flow Chart for Module PCG1AL



Program Listing for Module PCG1AL

```
      SUBROUTINE PCG1AL(ISUM,LENX,LCXXV,LCXXS,LCDT,LCE2,LCF2,LCG2,
     1LCVV,LCE22,LCD2S,LCNU1,MXITER,NPCOND,ITYP,NCOL,NROW,NLAY,IN,IOUT,
     2NOD)
C
C-----VERSION 1002 19JAN1987 PCG1AL
C
C     ******************************************************************
C     ALLOCATE STORAGE IN THE X ARRAY FOR PCG ARRAYS
C     ******************************************************************
C
C        SPECIFICATIONS:
C     ------------------------------------------------------------------
C     ------------------------------------------------------------------
C
C1------PRINT A MESSAGE IDENTIFYING PCG PACKAGE
      WRITE(IOUT,1)IN
    1 FORMAT(1H0,'PCG1 -- PRECONDITIONED CONJUGATE GRADIENT SOLUTION PAC
     1KAGE',', VERSION 1, 06/25/85',' INPUT READ FROM UNIT',I3)
```

```
C
C2------READ AND PRINT MXITER, NPCOND, AND ITYP
      READ(IN,2) MXITER,NPCOND,ITYP
    2 FORMAT(3I10)
      WRITE(IOUT,3) MXITER,NPCOND,ITYP
    3 FORMAT(1X,'MAXIMUM OF',I4,' ITERATIONS ALLOWED FOR CLOSURE'/
     1  1X,'PRECONDITIONING TYPE ',I2,'   PROBLEM TYPE ',I2)
C
C3------ALLOCATE SPACE FOR THE PCG ARRAYS
      ISOLD=ISUM
      NRC=NROW*NCOL
      NOD=NROW+NCOL
      ISIZ=NRC*NLAY
      LCXXV=ISUM
      ISUM=ISUM+ISIZ*2
      LCXXS=ISUM
      ISUM=ISUM+ISIZ
      LCDT=ISUM
      ISUM=ISUM+ISIZ
      LCE2=ISUM
      ISUM=ISUM+ISIZ
      LCF2=ISUM
      ISUM=ISUM+ISIZ
      LCG2=ISUM
      ISUM=ISUM+ISIZ
      LCVV=ISUM
      ISUM=ISUM+ISIZ
      LCE22=ISUM
      ISUM=ISUM+ISIZ
      LCD2S=ISUM
      ISUM=ISUM+NOD
      LCNU1=ISUM
      ISUM=ISUM+9
C
C4------CALCULATE AND PRINT THE SPACE USED IN THE X ARRAY
      ISP=ISUM-ISOLD
      WRITE(IOUT,4) ISP
    4 FORMAT(1X,I6,' ELEMENTS IN X ARRAY ARE USED BY PCG')
      ISUM1=ISUM-1
      WRITE(IOUT,5) ISUM1,LENX
    5 FORMAT(1X,I6,' ELEMENTS OF X ARRAY USED OUT OF',I7)
      IF(ISUM1.GT.LENX) WRITE(IOUT,6)
    6 FORMAT(1X,'   ***X ARRAY MUST BE DIMENSIONED LARGER***')
C
C5------RETURN
      RETURN
      END
```

List of Variables for Module PCG1AL

Variable	Range	Definition
IN	Package	Primary unit number from which input for this package will be read.
IOUT	Global	Primary unit number for all printed output. IOUT=6.
ISIZ	Module	Number of cells in the grid.
ISOLD	Package	Before this module allocates space, ISOLD is set equal to ISUM. After allocation, ISOLD is subtracted from ISUM to get ISP, the amount of space in the X array allocated by this module.
ISP	Module	Number of words in the X array allocated by this module.
ISUM	Global	Index number of the lowest element in the X array which has not yet been allocated. When space is allocated for an array, the size of the array is added to ISUM.
ISUM1	Module	ISUM-1.
ITYP	Package	Flag indicating the type of problem being solved.
LCXXV	Package	Location in the X array of the first element of array XXV.
LCXXS	Package	Location in the X array of the first element of array XXS.
LCDT	Package	Location in the X array of the first element of array DT.
LCE2	Package	Location in the X array of the first element of array E2.
LCF2	Package	Location in the X array of the first element of array F2.
LCG2	Package	Location in the X array of the first element of array G2.
LCVV	Package	Location in the X array of the first element of array VV.
LCNU1	Package	Location in the X array of the first element of array NU1.
LENX	Global	Length of the X array in words. This should always be equal to the dimension of array X specified in the main program.
MXITER	Package	Maximum total number of iterations allowed.

NCOL	Global	Number of columns in the grid.
NLAY	Global	Number of layers in the grid.
NPCOND	Module	Has the values 1 to 5 correspondingto the 5 preconditioning types.
NRC	Module	Number of cells in a layer.
NROW	Global	Number of rows in the grid.

PCG1RP

Narrative for Module PCG1RP

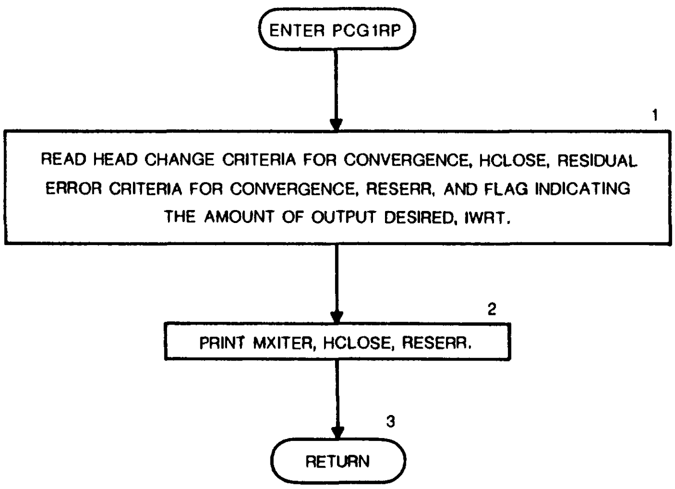
Module PCG1RP reads data for the PCG Package: the head change criteria for convergence, HCLOSE, the residual error criteria for convergence, RESERR, and a flag IWRT indicating the amount of output desired regarding the numerical solution. If IWRT is chosen to be 2, then module PCG1RP also reads NU1(1),NU1(2),...,NU1(9). These quantities specify 3 locations at which the head is printed by module PCG1AP for each iteration. Module PCG1RP performs its functions in the following order:

1. Read the data.

2. Print the data read in step 1, except array NU1.

3. Return.

Flow Chart for Module PCG1RP



Program Listing for Module PCG1RP

```
      SUBROUTINE PCG1RP(MXITER,HCLOSE,RESERR,NU1,IN,IOUT,IWRT)
C
C-----VERSION 1001 25JUN1985 PCG1RP
C
C     ******************************************************************
C     READ DATA FOR PCG
C     ******************************************************************
C
C        SPECIFICATIONS:
C     ------------------------------------------------------------------
      DIMENSION NU1(9)
C     ------------------------------------------------------------------
C
C1------READ HCLOSE,RESERR,IWRT
      READ(IN,1) HCLOSE,RESERR,IWRT
    1 FORMAT(2F10.0,I10)
      DO 2 I=1,9
    2 NU1(I)=0
      IF(IWRT.GE.2) READ(IN,3) (NU1(I),I=1,9)
    3 FORMAT(9I4)
C
C2------PRINT DATA VALUES JUST READ
      WRITE(IOUT,100)
  100 FORMAT(1H0,///55X,'SOLUTION BY PRECONDITIONED CONGUGATE GRADIENT '
     1/55X,45('-'))
      WRITE(IOUT,115) MXITER
  115 FORMAT(1H0,47X,'MAXIMUM ITERATIONS ALLOWED FOR CLOSURE =',I9)
      WRITE(IOUT,125) HCLOSE
  125 FORMAT(1H ,52X,'HEAD CHANGE CRITERION FOR CLOSURE =',E15.5)
      WRITE(IOUT,120) RESERR
  120 FORMAT(1H ,41X,'MAXIMUM ALLOWABLE RESIDUAL ERROR FOR CLOSURE =',
     1E15.5)
 1000 RETURN
      END
```

List of Variables for Module PCG1RP

Variable	Range	Definition
HCLOSE	Module	Head change criteria for convergence.
IN	Package	Primary unit number from which input for this package will be read.
IOUT	Global	Primary unit number for all printed output. IOUT=6.
IWRT	Package	Flag indicating the amount of output desired.

MXITER	Package	Maximum total number of iterations allowed.
NU1	Package	An array holding the location of three nodes for which head values are printed at each iteration, if IWRT=2.
RESERR	Module	Residual error criteria for convergence.

PCG1AP

Narrative for Module PCG1AP

Module PCG1AP performs one or more iterations of the preconditioned conjugate gradient (PCG) method. To save computation time, all arrays are declared one dimensional. The one-dimensional indexes are calculated from the layer, row, and column indexes normally used to access the arrays in three dimensions. Computation time is saved because calculations are not repeated for identical indexes as would be done by internal FORTRAN addressing routines if three-dimensional subscripts were used.

Module PCG1AP has several important steps. First there are several initialization steps. These are followed by several repeatable steps which are passed through once for each value of the iteration index v in equations (9) through (13).

The first initialization step is setting XX, MHD, DD, BB, ZZ, XXS, and YQ equal to HNEW, 1-IBOUND, -CR, -CC, -CV, -HCOF, and -RHS from the main program. Another initialization step is the calculation of arrays F2 and G2. These arrays store useful values used when calculating the vector $K^{-1}r_v$ in equations (9) and (12). These values are stored so that they do not have to be recalculated for each evaluation of $K^{-1}r_v$, v=1,2,... . F2 and G2 are calculated in the FORTRAN DO loops, "do through 51" for NUM4=NPCOND=1,2,3 and "do through 54" for NPCOND=4,5.

The first repeatable step, in the "do through 100" DO loop, is the calculation of Ap_v, which is put into array DT, from p_v, which is in array E2. The quantity (p_v,Ap_v) is also calculated and placed into variable SPP. These calculations occur in the "do through 3" DO loop.

Next, a_v, called variable A1 in the module, is calculated using the value just calculated for SPP=(p_v,Ap_v) and a previously calculated value for SRP=$(r_v,K^{-1}r_v)$. At this point in the program, equation (9) has been completed and equations (10)-(13) still remain to be evaluated.

Now equations (10) and (11) are evaluated in the "do through 4" DO loop. New values for x and r, x_{v+1} and r_{v+1}, replace old values in the arrays XX and VV respectively. Ap_v from a previous step located in array DT is used, along with p_v from array E2, and a_v in variable A1. At this point, based on the latest results for x_{v+1} and residual error vector r_{v+1}, the program may exit the v iteration loop as shown in figure 3, and go to statement 201.

If no exit has occured, the program proceeds and calculates $K^{-1}r_{v+1}$ and places it into array DT, which for the time being is no longer needed to hold Ap_v. At the same time $(r_{v+1},K^{-1}r_{v+1})$ is calculated and placed into the variable SPR after putting the old value $(r_v,K^{-1}r_v)$ into variable SPRS. These calculations are done in the "do through 10" and "do through 11" DO loops for NPCOND=1,2,3 , in the "do through 63" DO loop for NPCOND=4, and in the "do through 651" and "do through 652" DO loops for NPCOND=5.

In the next step, B_v in equation (12) is calculated as SPR/SPRS and placed in the variable B6.

In the final repeatable step, p_{v+1} of equation (13) is evaluated in the "do through 5" DO loop. Array DT containing $K^{-1}r_{v+1}$ from a previous step is used as is variable B6 containing B_v, and array E2 containing p_v. At this point the end of the "do through 100" DO loop occurs, so the first repeatable step is again processed and the v iteration of equations (9) through (13) continues.

The ITER=1 iteration of the "do through 100" DO loop, with index ITER=1,I300, is essentially a null iteration in the program and only sets up initial values for x, r, and p: x_1, r_1, and p_1. The ITER=2 iteration corresponds to v=1 in equations (9) through (13), ITER=3 corresponds to v=2, and so on, so that ITER=v+1. Since v=ITER-1, the upper limit, v_{max}, for the index v in equations (9) through (13) and (14) is I300-1.

For ITYP=0, ITER=v+1 has an upper limit I300 of MXITER+1 (table 1). The index KITER in the main program, denoted by u in equation (14), does not iterate and has the value 1. Thus the finite difference equations are formulated only once per time step, and A and q in equation (7) are constant. This situation is appropriate for the solution of linear problems (LAYCON=0, river, drain, or evapotranspiration packages are not being used). It also gives the solution of equation (8) for non-linear problems as discussed previously. This latter situation will give a poor solution to the overall volumetric budget as calculated by the module BAS1OT because this module (and others) assumes equation (7), not equation (8) is being solved. This poor budget result and other faulty flow rate values do not indicate that the solution obtained is in error. They arise because budget and flow rate calculations appropriate to equation (7) are being applied to equation (8).

For ITYP=1, I300 is set to MXITER+1. For ITYP≥2, I300 is set to ITYP. Thus for ITYP=2, only one iteration of equations (9) through (13) occurs and index v has the single value 1 in equations (9) through (13) and (14). For ITYP≥1, the index u in equation (14), also denoted by KITER in the main program and in the module PCG1AP, has values 1,2,...,um. The upper limit for u=KITER, um, as set in the main program, is MXITER.

ITYP≥1 solutions are for non-linear problems and solve equation (7), so that budget and flow quantities are calculated correctly. ITYP=1 is for non-linear problems with weakly non-linear conditions. ITYP=2 is for non-linear problems with strongly non-linear conditions.

For each value of the time index m, iteration is terminated when equation (7), or (8) when used, is approximately satisfied according to some indication

Table 1.--ITYP control of v and u iterations.

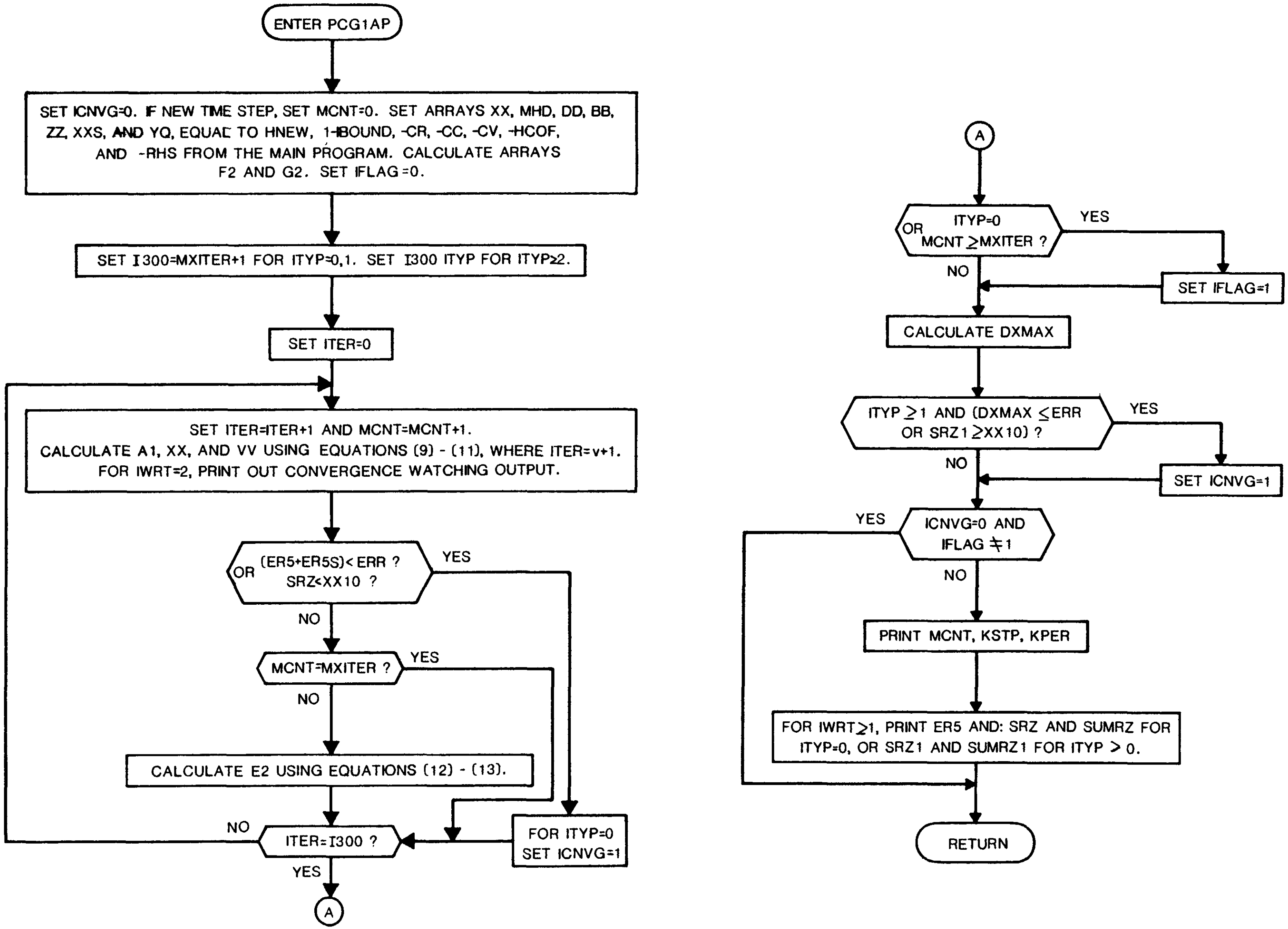
ITYP	use	u values	$I300=v_{max}+1$	v_{max}
0	linear problems or equation (8).	1	MXITER+1	MXITER
1	weakly nonlinear	1,...,um	MXITER+1	MXITER
2	strongly nonlinear	1,...,um	ITYP=2	ITYP-1=1
3	strongly nonlinear	1,...,um	ITYP=3	ITYP-1=2
4	strongly nonlinear	1,...,um	ITYP=4	ITYP-1=3
.	....	....	......	....
.	....	....	......	....
	decreasing nonlinearity			

of solution accuracy. Two basic criteria for iteration termination are used. Criteria one causes iteration to terminate when the change in head from one iteration to the next is small by some measure. Criteria two causes iteration to terminate when the residual error vector $r=q-Ah$ becomes small. Note that neither criteria actually uses a measure of the true error of the approximate solution $h_{sm}{}^{m}$ to equation (7), since the true solution h^{m} cannot be found. The second criteria for iteration termination, however, actually does consider the true value of the residual error vector $r=q-Ah$ which is available.

For criteria 1, iteration termination with ITYP≥1, termination occurs when the maximum absolute component of the vector $dh_1=h_{u+1,1}{}^{m}-h_{u,1}{}^{m}$ is less than ERR=HCLOSE. When ITYP=0, for which u=1, termination occurs when the sum of the maximium absolute components of the vectors $dh_1=h_{1,v+1}{}^{m}-h_{1,v}{}^{m}$ and $dh_2=h_{1,v}{}^{m} - h_{1,v-1}{}^{m}$ is less than ERR=HCLOSE.

For criteria 2, iteration termination with ITYP≥1, termination occurs when the maximum absolute component of the residual error vector $r=q(h_{u,1}{}^{m})-A(h_{u,1}{}^{m})h_{u,1}{}^{m}$ is less than XX10=RESERR. For ITYP=0, $r=q-Ah_{1,v}{}^{m}$ is used.

Flow Chart for Module PCG1AP



Program Listing for Module PCG1AP

```
      SUBROUTINE PCG1AP(XX,MHD,DD,BB,ZZ,XXSP,YQ,XXV,XXS,DT,E2,F2,G2,VV,
     1E22,D2S,NU1,NUM4,ITYP,KITER,ERR,XX10,ICNVG,MXITER,NCOL,NROW,NLAY,
     2IOUT,IWRT,NODES,NOD,KSTP,KPER,MCNT,KSTPS,KPERS,IFLAG)
C      ********************************************
C      ******** VERSION 1008 19 FEB 1987 PCG1AP **
C      ********   L. K. KUIPER   ****************
C      ********************************************
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*4 DD(NODES),BB(NODES),ZZ(NODES),XXS(NODES),YQ(NODES),
     1ERR,XX10,XXSP(NODES)
     2,DT(NODES),VV(NODES),E2(NODES),F2(NODES),G2(NODES)
     3,E22(NODES),D2S(NOD)
      DIMENSION XX(NODES),MHD(NODES),NU1(9),XXV(NODES)
      IFLAG=0
      ICNVG=0
      IF((KPER.NE.KPERS).OR.(KSTP.NE.KSTPS)) MCNT=0
      KPERS=KPER
      KSTPS=KSTP
      NI10=NCOL
      NJ10=NROW
      NK10=NLAY
      NI11=NI10+1
      NJ11=NJ10+1
      NK11=NK10+1
      NIJ10=NI10*NJ10
      N320=NIJ10*NK10
      NW1=NU1(1)+NI10*(NU1(2)-1)+NIJ10*(NU1(3)-1)
      NW2=NU1(4)+NI10*(NU1(5)-1)+NIJ10*(NU1(6)-1)
      NW3=NU1(7)+NI10*(NU1(8)-1)+NIJ10*(NU1(9)-1)
      IWR1+IWRT
      IF((IWR1.LT.2).OR.(KSTP*KITER.GT.1)) GO TO 901
      WRITE(IOUT,5007)
      WRITE(IOUT,5072) NU1(1),NU1(4),NU1(7)
      WRITE(IOUT,5073) NU1(2),NU1(5),NU1(8)
      WRITE(IOUT,5074) NU1(3),NU1(6),NU1(9)
901   CONTINUE
      DO 909 IJ=1,N320
      XXV(IJ)=XX(IJ)
      XXS(IJ)=-XXSP(IJ)
      MHD(IJ)=1-MHD(IJ)
      DD(IJ)=-DD(IJ)
      BB(IJ)=-BB(IJ)
      ZZ(IJ)=-ZZ(IJ)
909   YQ(IJ)=-YQ(IJ)
      DO 91 IJB=1,N320
      IJ=N320+1-IJB
      IM1JK=IJ-1
      IJM1K=IJ-NI10
      IJKM1=IJ-NIJ10
      DD(IJ)=DD(IM1JK)
```

```
      BB(IJ)=BB(IJM1K)
      ZZ(IJ)=ZZ(IJKM1)
91    CONTINUE
      DO 92 K=1,NK10
      DO 92 J=1,NJ10
      DO 92 I=1,NI10
      IJ=I+NI10*(J-1)+NIJ10*(K-1)
      IF(I.EQ.1) DD(IJ)=0
      IF(J.EQ.1) BB(IJ)=0
      IF(K.EQ.1) ZZ(IJ)=0
92    CONTINUE
      DO 1 IJ=1,N320
      DT(IJ)=0
      E2(IJ)=0
      F2(IJ)=0
      G2(IJ)=0
      E22(IJ)=0
1     VV(IJ)=0
      NN=NROW+NCOL
      DO 2 IJ=1,NN
2     D2S(IJ)=0
      DO 16 IJ=1,N320
      E2(IJ)=XX(IJ)
      IF(MHD(IJ).GE.1) GO TO 16
      VV(IJ)=YQ(IJ)
16    CONTINUE
      IF(NUM4.GE.4) GO TO 74
71    CONTINUE
      Y1=0
      Z1=0
      DO 51 IJ=1,N320
      IF(.NOT.(MHD(IJ).GE.1)) GO TO 777
      Y1=0
      Z1=0
      D2S(NI10+1)=0
      DO 94 I=1,NI10
94    D2S(I)=D2S(I+1)
      GO TO 51
777   CONTINUE
      IM1JK=IJ-1
      IJM1K=IJ-NI10
      IJKM1=IJ-NIJ10
      IDM=1
      IBM=1
      IZM=1
      IF(IM1JK.LT.1) IDM=0
      IF(IJM1K.LT.1) IBM=0
      IF(IJKM1.LT.1) IZM=0
      IP1JK=IJ+1
      IJP1K=IJ+NI10
      IJKP1=IJ+NIJ10
      IDP=1
      IBP=1
```

```
      IZP=1
      IF(IP1JK.GT.N320) IDP=0
      IF(IJP1K.GT.N320) IBP=0
      IF(IJKP1.GT.N320) IZP=0
      X=DD(IJ)
      Y=BB(IJ)
      Z=ZZ(IJ)
      EEIJ=-(X+Y+Z+IDP*DD(IP1JK)+IBP*BB(IJP1K)+IZP*ZZ(IJKP1))+XXS(IJ)
      IF(MHD(IM1JK)*IDM.GE.1) X=0
      IF(MHD(IJM1K)*IBM.GE 1) Y=0
      IF(MHD(IJKM1)*IZM.GE.1) Z=0
      Y1Z1=0
      XP=X
      YP=Y
      F2X=F2(IM1JK)*IDM
      F2Y=F2(IJM1K)*IBM
      F2Z=F2(IJKM1)*IZM
      IF(NUM4.EQ.1) GO TO 20
      JP1=IJ-NI10+1
      KP1=IJ-NIJ10+1
      IJMMN=IJ-(NIJ10-NI10)
      IJP1=1
      IKP1=1
      IMMN=1
      IF(JP1.LT.1) IJP1=0
      IF(KP1.LT.1) IKP1=0
      IF(IJMMN.LT.1) IMMN=0
      WY1=0
      WZ1=0
      IF(F2Y.NE.0.0) WY1=Y/F2Y
      IF(F2Z.NE.0.0) WZ1=Z/F2Z
      XP=X-(WY1*Y1+WZ1*Z1)
      G2(IJ)=XP
      YP=Y-WZ1*D2S(1)
      E22(IJ)=YP
      Y1=-WY1*G2(JP1)*IJP1
      Z1=-WZ1*G2(KP1)*IKP1
      ZB=-WZ1*E22(IJMMN)*IMMN
      D2S(NI10+1)=ZB
      DO 93 I=1,NI10
93    D2S(I)=D2S(I+1)
      F21=F2(JP1)*IJP1
      F22=F2(KP1)*IKP1
      F23=F2(IJMMN)*IMMN
      P1=0
      P2=0
      P3=0
      IF(F21.NE.0.0) P1=Y1*Y1/F21
      IF(F22.NE.0.0) P2=Z1*Z1/F22
      IF(F23.NE.0.0) P3=ZB*ZB/F23
      Y1Z1=P1+P2+P3
20    CONTINUE
      XF=0
```

```
      YF=0
      ZF=0
      IF(F2X.NE.0.0) XF=XP*XP/F2X
      IF(F2Y.NE.0.0) YF=YP*YP/F2Y
      IF(F2Z.NE.0.0) ZF=Z*Z/F2Z
      F2(IJ)=EEIJ-(XF+YF+ZF+Y1Z1)
51    CONTINUE
      GO TO 80
74    CONTINUE
      DO 54 IJ=1,N320
      IF(MHD(IJ).GE.1) GO TO 54
      IP1JK=IJ+1
      IJP1K=IJ+NI10
      IJKP1=IJ+NIJ10
      IDP=1
      IBP=1
      IZP=1
      IF(IP1JK.GT.N320) IDP=0
      IF(IJP1K.GT.N320) IBP=0
      IF(IJKP1.GT.N320) IZP=0
      EEIJ= (-(DD(IJ)+BB(IJ)+ZZ(IJ)+IDP*DD(IP1JK)
     1+IBP*BB(IJP1K)+IZP*ZZ(IJKP1))+XXS(IJ))
      IF(NUM4.EQ.4) GO TO 539
      X=DD(IJ)
      F2X=F2(IJ-1)
      XF=0
      IF(F2X.NE.0.0) XF=X*X/F2X
      F2(IJ)=EEIJ-XF
      GO TO 54
539   F2(IJ)=EEIJ
54    CONTINUE
80    CONTINUE
      SPR=1.D-50
      ICNT=-1
      ER5S=1.D40
      I300=MXITER+1
      IF(ITYP.GE.2) I300=ITYP
      DO 100 ITER=1,I300
      ICNT=ICNT+1
      IF(ITER*KITER*KSTP*KPER.EQ.1) MCNT=0
      IF(ITER.NE.1) MCNT=MCNT+1
      SPP=0
      DO 3 IJ=1,N320
      IF(MHD(IJ).GE.1) GO TO 3
      IP1JK=IJ+1
      IJP1K=IJ+NI10
      IJKP1=IJ+NIJ10
      IDP=1
      IBP=1
      IZP=1
      IF(IP1JK.GT.N320) IDP=0
      IF(IJP1K.GT.N320) IBP=0
      IF(IJKP1.GT.N320) IZP=0
```

```
      IJM1K=IJ-NI10
      IJKM1=IJ-NIJ10
      DDIJ=DD(IJ)
      DDIP1=DD(IP1JK)*IDP
      BBIJ=BB(IJ)
      BBIJP=BB(IJP1K)*IBP
      ZZIJ=ZZ(IJ)
      ZZIJK=ZZ(IJKP1)*IZP
      E2IJ=E2(IJ)
      DTIJ=(-(DDIJ+DDIP1+BBIJ+BBIJP+ZZIJ+ZZIJK)+XXS(IJ))*E2IJ
     1+DDIJ*E2( IJ-1)+DDIP1*E2(IP1JK)
     2+BBIJ*E2(IJM1K)+BBIJP*E2(IJP1K)
     3+ZZIJ*E2(IJKM1)+ZZIJK*E2(IJKP1)
      DT(IJ)=DTIJ
      SPP=SPP+E2IJ*DTIJ
3     CONTINUE
      A1=SPR/(SPP+1.D-50)
      A2=A1
      IF(ITER.GT.1) GO TO 35
      A1=0
      A2=1.
35    SRZ=0
      SUMRZ=0
      ER5=0
      DO 4 K=1,NK10
      DO 4 J=1,NJ10
      DO 4 I=1,NI10
      IJ=I+NI10*(J-1)+NIJ10*(K-1)
      IF(MHD(IJ).GE.1) GO TO 4
      DX=A1*E2(IJ)
      IF(MHD(IJ).EQ.2) DX=0
      X=XX(IJ)+DX
      XX(IJ)=X
      ADX=DABS(DX)
      IF(ADX.LT.ER5) GO TO 111
      IMX=I
      JMX=J
      KMX=K
      XXPMX=X
      ER5=ADX
111   CONTINUE
      X=VV(IJ)-A2*DT(IJ)
      SUMRZ=SUMRZ+X
      DSR=DABS(X)
      IF(DSR.GT.SRZ) SRZ=DSR
      VV(IJ)=X
4     CONTINUE
      IF(ITER.EQ.1) SRZ1=SRZ
      IF(ITER.EQ.1) SUMRZ1=SUMRZ
      IF((IWR1.GE.2).AND.(ITER.NE.1)) WRITE(IOUT,1000) MCNT,ICNT,IMX,
     1JMX,KMX,XXPMX,ER5,SRZ,XX(NW1),XX(NW2),XX(NW3)
      X5=.5
      IF(ITER.LE.2) X5=1.0
```

```
      IF(((ER5+ER5S)*X5).LT.ERR) GO TO 201
      IF(SRZ.LT.XX10) GO TO 201
      IF(MCNT.EQ.MXITER) GO TO 202
      ER5S=ER5
      SPRS=SPR
      SPR=0
      GO TO (81,81,81,83,85),NUM4
81    CONTINUE
      DO 10 IJ=1,N320
      IF(MHD(IJ).GE.1) GO TO 10
      IJM1K=IJ-NI10
      IJKM1=IJ-NIJ10
      B6=0
      Z6=0
      DDIJ=DD(IJ)
      BBIJ=BB(IJ)
      IF(NUM4.EQ.1) GO TO 21
      DDIJ=G2(IJ)
      BBIJ=E22(IJ)
      JP1=IJ-NI10+1
      KP1=IJ-NIJ10+1
      IJMMN=IJ-(NIJ10-NI10)
      B6=0
      Z6=0
      F2J=F2(IJM1K)
      F2K=F2(IJKM1)
      IF(F2J.NE.0.D0) B6=DT(JP1)*G2(JP1)/F2J
      IF(F2K.NE.0.D0) Z6=(DT(KP1)*G2(KP1)+DT(IJMMN)*E22(IJMMN))/F2K
21    CONTINUE
      DT(IJ)=(VV(IJ)-DDIJ*DT(IJ-1)-BBIJ*(DT(IJM1K)-B6)
     1-ZZ(IJ)*(DT(IJKM1)-Z6))/F2(IJ)
10    CONTINUE
      DO 11 IJB=1,N320
      IJ=N320+1-IJB
      IF(MHD(IJ).GE.1) GO TO 11
      IP1JK=IJ+1
      IJP1K=IJ+NI10
      IJKP1=IJ+NIJ10
      IDP=1
      IBP=1
      IZP=1
      IF(IP1JK.GT.N320) IDP=0
      IF(IJP1K.GT.N320) IBP=0
      IF(IJKP1.GT.N320) IZP=0
      XAD=0
      DDD=DD(IP1JK)
      BBB=BB(IJP1K)
      IF(NUM4.EQ.1) GO TO 22
      JM1=IJ+NI10-1
      KM1=IJ+NIJ10-1
      IJPMN=IJ+(NIJ10-NI10)
      IJM1=1
      IKM1=1
```

```
      IPMN=1
      IF(JM1.GT.N320) IJM1=0
      IF(KM1.GT.N320) IKM1=0
      IF(IJPMN.GT.N320) IPMN=0
      IM1JK=IJ-1
      IJM1K=IJ-NI10
      IDM=1
      IBM=1
      IF(IM1JK.LT.1) IDM=0
      IF(IJM1K.LT.1) IBM=0
      DDD=G2(IP1JK)
      BBB=E22(IJP1K)
      XAD1=0
      XAD2=0
      F2I=F2(IM1JK)*IDM
      F2J=F2(IJM1K)*IBM
      IF(F2I.NE.0.D0) XAD1=-(E22(JM1)*IJM1*DT(JM1)+ZZ(KM1)*IKM1*
     1DT(KM1))*G2(IJ)/F2I
      IF(F2J.NE.0.D0) XAD2=-ZZ(IJPMN)*IPMN*E22(IJ)*DT(IJPMN)/F2J
      XAD=XAD1+XAD2
22    CONTINUE
      DTIJ=DT(IJ)-(DDD*IDP*DT(IP1JK)+BBB*IBP*DT(IJP1K)+ZZ(IJKP1)
     1*IZP*DT(IJKP1)+XAD)/F2(IJ)
      DT(IJ)=DTIJ
      SPR=SPR+DTIJ*VV(IJ)
11    CONTINUE
      GO TO 90
83    CONTINUE
      DO 63 IJ=1,N320
      IF(MHD(IJ).GE.1) GO TO 63
      F2IJ=F2(IJ)
      VVIJ=VV(IJ)
      DTIJ=VVIJ/F2IJ
      DT(IJ)=DTIJ
      SPR=SPR+DTIJ*VVIJ
63    CONTINUE
      GO TO 90
85    CONTINUE
      DO 651 IJ=1,N320
      IF(MHD(IJ).GE.1) GO TO 651
      DT(IJ)=(VV(IJ)-DD(IJ)*DT(IJ-1))/F2(IJ)
651   CONTINUE
      DO 652 IJB=1,N320
      IJ=N320+1-IJB
      IF(MHD(IJ).GE.1) GO TO 652
      IP1JK=IJ+1
      DTIJ=DT(IJ)-DD(IP1JK)*DT(IP1JK)/F2(IJ)
      DT(IJ)=DTIJ
      SPR=SPR+DTIJ*VV(IJ)
652   CONTINUE
90    CONTINUE
      B6=SPR/SPRS
      IF(ITER.EQ.1) B6=0
```

```
      DO 5 IJ=1,N320
      E2IJ=DT(IJ)+B6*E2(IJ)
      IF(MHD(IJ).GE.1) E2IJ=0
      E2(IJ)=E2IJ
5     CONTINUE
100   CONTINUE
      GO TO 202
201   IF(ITYP.EQ.0) ICNVG=1
202   CONTINUE
      IF((MCNT.GE.MXITER).OR.(ITYP.EQ.0)) IFLAG=1
      DXMAX=0
      DO 19 IJ=1,N320
      XXVIJ=XXV(IJ)
      DX=DABS(XX(IJ)-XXVIJ)
      IF(DX.GT.DXMAX) DXMAX=DX
      IP1JK=IJ+1
      IJP1K=IJ+NI10
      IJKP1=IJ+NIJ10
      DD(IJ)=DD(IP1JK)
      BB(IJ)=BB(IJP1K)
      ZZ(IJ)=ZZ(IJKP1)
19    CONTINUE
      IF((ITYP.GE.1).AND.((DXMAX.LE.ERR).OR.(SRZ1.LT.XX10))) ICNVG=1
      DO 919 IJ=1,N320
      MHD(IJ)=1-MHD(IJ)
      DD(IJ)=-DD(IJ)
      BB(IJ)=-BB(IJ)
      ZZ(IJ)=-ZZ(IJ)
919   YQ(IJ)=-YQ(IJ)
      DO 991 IJ=1,N320
991   XXSP(IJ)=-XXS(IJ)
      IF((ICNVG.EQ.0).AND.(IFLAG.NE.1)) GO TO 600
      IF(KSTP.EQ.1) WRITE(IOUT,500)
      S1=SRZ
      S2=SUMRZ
      IF(ITYP.EQ.0) GO TO 518
      S1=SRZ1
      S2=SUMRZ1
518   CONTINUE
      WRITE(IOUT,501) MCNT,KSTP,KPER
      IF(IWRT.GE.1) WRITE(IOUT,5075) ER5,S1,S2
  500 FORMAT(1H0)
  501 FORMAT(1X,I5,' ITERATIONS FOR TIME STEP',I4,' IN STRESS PERIOD',
     1I3)
1000  FORMAT(' ',2I3,3I4,6D18.7)
5007  FORMAT('0',56X,'WATCHING CONVERGENCE'//25X,'THE MAXIMUM CHANGE IN
     1HEAD OCCURED AT COLUMN J, ROW I, LAYER K.'
     2                                              /25X,'MAXIMUM RESIDU
     3AL ERROR  = THE MAXIMUM OVER ALL THE GRID ELEMENTS OF THE'/25X,'DI
     4FFERENCE BETWEEN THE WATER FLOW RATE INTO AND OUT OF EACH GRID ELE
     5MENT.')
5074  FORMAT(7X,'    J    I    K',5X,'HEAD AT J,I,K',5X,'HEAD AT J,I,K',13X
     1,'ERROR',3(12X,'K=',I4))
```

```
5072  FORMAT('0',72X,3(4X,'HEAD AT J=',I4))
5073  FORMAT(46X,'CHANGE IN',6X,'MAX RESIDUAL',3(12X,'I=',I4))
5075  FORMAT(
     1'0','MAXIMUM CHANGE IN HEAD BETWEEN LAST 2 ITERATIONS =',D11.3/
     2' MAXIMUM RESIDUAL ERROR (L**3/T) FOR GRID ELEMENTS NOT HAVING FIX
     3ED HYDRAULIC HEAD =',D11.3,'      TOTAL =',D11.3)
600   RETURN
      END
```

List of Variables for Module PCG1AP

Variable	Range	Definition
A1	Module	a_v in equation (9).
B6	Module	B_v in equation (12).
BB	Module	-B in equation (2).
DD	Module	-D in equation (2).
DT	Module	Work space array used when calculating equations (9) through (13).
DXMAX	Module	dh_1 for criteria 1 ITYP$\geq$1 iteration termination.
E2	Module	Vector p_v in equations (9) through (13).
ER5	Module	dh_1 for criteria 1 ITYP=0 iteration termination. Called maximum absolute head change when printed out.
ER5S	Module	dh_2 for criteria 1 ITYP=0 iteration termination.
ERR	Module	HCLOSE in module PCG1RP. The head change criteria for convergence.
F2	Module	Storage array used when calculating $K^{-1}r_v$ in equations (9) and (12).
G2	Module	Same as above.
I300	Module	Maximum value for iteration counter ITER. Equal to v_{max}+1 of figure 3.
ICNVG	Global	This flag is set to one when iteration has met the conditions for iteration termination.
IFLAG	Global	This flag is set to one when exit is desired from the u iteration loop in the main program, due either to MCNT$\geq$MXITER or ITYP=0.

IOUT	Global	Primary unit number for all printed output. IOUT=6.
ITER	Module	Iteration counter for v in equations (9) through (13), ITER=v+1.
ITYP	Package	Flag indicating the type of problem being solved.
IWRT	Package	Flag indicating the amount of output desired.
KITER	Global	Iteration counter u in equation (14). Reset at the start of each time step.
KPER	Global	Stress period counter.
KPERS	Module	Stored value of KPER.
KSTP	Global	Time step counter. Reset at the start of each stress period.
KSTPS	Module	Stored value of KSTP.
MCNT	Module	Iteration number counter that counts the total number of iterations that are used as indices u and v increase in equations (9) through (13) and (14).
MHD	Module	An array to indicate when a node is active and if it has a fixed head. MHD is equal to 1-IBOUND from the main program.
MXITER	Package	Maximum total number of iterations allowed.
NCOL	Global	Number of columns in the grid.
NLAY	Global	Number of layers in the grid.
NODES	Global	Number of cells (nodes) in the finite difference grid.
NROW	Global	Number of rows in the grid.
NU1	Package	An array holding the location of three nodes for which head values are printed at each iteration, if IWRT=2.
NUM4	Module	Called NPCOND in module PCG1AL. Has values of 1 to 5 for the 5 preconditioning types.
SRZ	Module	Maximum absolute component of the residual error vector r for ITYP=0 criteria 2 iteration termination. Called maximum absolute residual error when printed out.
SRZ1	Module	Maximum absolute component of the residual error vector r for ITYP$\geq$1 criteria 2 iteration termination. Called maximum absolute residual error when printed out.

SUMRZ	Module	Total residual error, ITYP=0.
SUMRZ1	Module	Total residual error, ITYP$\geq$1.
VV	Module	Error vector r_v in equations (9) through (13).
XX10	Module	Called RESERR in module PCG1RP. The residual error criteria for convergence.
XX	Module	Head vector h. Vector x_v in equations (9) through (13). HNEW in the main program.
XXS	Module	-HCOF in equation (1).
XXV	Module	Stored XX array. Used to get DXMAX.
YQ	Module	-RHS in equation (1).
ZZ	Module	-Z in equation (2).

REFERENCES

Hageman, L. A., and Young, D. M., 1981, Applied Iterative Methods, New York, Academic Press, 386 p.

Kershaw, D. S., 1978, The incomplete Choleski-conjugate gradient method for the iterative solution of linear equations: Journal of Computational Physics, v. 26, p. 43-65.

Kuiper, L. K., 1981, A comparision of the incomplete Choleski-conjugate gradient method with the strongly implicit method as applied to the solution of two-dimensional groundwater flow equations: Water Resources Research, v. 17, no. 4, p. 1082.

Kuiper, L. K., 1987, A comparison of iterative methods as applied to the solution of the nonlinear three-dimensional groundwater flow equation: Society for Industrial and Applied Mathematics Journal on Scientific and Statistical Computing, v. 8, no. 4, p. 521-528.

McDonald, M. G., and Harbaugh, A. W., 1984, A modular three-dimensional finite difference ground-water flow model: U.S. Geological Survey Open-File Report 83-875, 528 p.

Van Der Vorst, H. A., 1982, A vectorizable variant of some ICCG methods: Society for Industrial and Applied Mathematics Journal on Scientific and Statistical Computing, v. 3, no. 3, p. 350-356.