USGS
science for a changing world

PRECONDITIONED CONJUGATE-GRADIENT 2 (PCG2), A COMPUTER PROGRAM FOR SOLVING GROUND-WATER FLOW EQUATIONS

U.S. GEOLOGICAL SURVEY

Water-Resources Investigations Report 90-4048

PRECONDITIONED CONJUGATE-GRADIENT 2 (PCG2), A COMPUTER PROGRAM FOR SOLVING GROUND-WATER FLOW EQUATIONS

By Mary C. Hill

U.S. GEOLOGICAL SURVEY

Water-Resources Investigations Report 90-4048

Denver, Colorado
1990

DEPARTMENT OF THE INTERIOR

MANUEL LUJAN, JR., Secretary

U.S. GEOLOGICAL SURVEY

Dallas L. Peck, Director

For additional information
write to:

Chief, Branch of Regional Research
U.S. Geological Survey
Box 25046, Mail Stop 418
Federal Center
Denver, CO 80225-0046

Copies of this report can
be purchased from:

U.S. Geological Survey
Books and Open-File Reports Section
Box 25425
Federal Center
Denver, CO 80225-0425

CONTENTS

FIGURES

PRECONDITIONED CONJUGATE-GRADIENT 2 (PCG2), A COMPUTER PROGRAM FOR SOLVING GROUND-WATER FLOW EQUATIONS

By Mary C. Hill

ABSTRACT

This report documents PCG2: a numerical code to be used with the U.S. Geological Survey modular three-dimensional, finite-difference, ground-water flow model. PCG2 uses the preconditioned conjugate-gradient method to solve the equations produced by the model for hydraulic head. Linear or nonlinear flow conditions may be simulated.

PCG2 includes two preconditioning options: modified incomplete Cholesky preconditioning, which is efficient on scalar computers; and polynomial preconditioning, which requires less computer storage and, with modifications that depend on the computer used, is most efficient on vector computers. Convergence of the solver is determined using both head-change and residual criteria. Nonlinear problems are solved using Picard iterations.

This documentation provides a description of the preconditioned conjugate-gradient method and the two preconditioners, detailed instructions for linking PCG2 to the modular model, sample data inputs, a brief description of PCG2, and a FORTRAN listing.

INTRODUCTION

Purpose and Scope

Finite-difference numerical models commonly are used to investigate ground-water flow systems. Effective use of these models requires that the matrix equations they produce be solved efficiently, that is, that a correct solution is produced using as little computer processing time as possible. Effective use of the models also requires that the amount of computer storage be minimized to allow for solution on small computers and to avoid overburdening large computers.

The purpose of this work is to document PCG2, a numerical code which uses the preconditioned conjugate-gradient method to solve the matrix equations produced by the U.S. Geological Survey modular three-dimensional, finite-difference ground-water flow model. Two preconditioning options are included which have not previously been available for use with this model, and which perform better than available solvers for many problems.

Previous Investigations

Matrix equations have been solved using direct or iterative methods. In most direct methods the matrix is factored exactly and the true solution is obtained by executing one backward and one forward substitution. In most iterative methods an initial estimate of the solution is refined iteratively using an approximately factored matrix, and successive solutions should approach the true solution. Solution convergence is assumed to have been reached when some measure of the residual and(or) the difference in results between successive iterations is less than some user-specified convergence criteria. Direct solution is straightforward, but iterative methods are less susceptible to round-off error, are more efficient for large problems, and require less computer storage (Remson, Hornberger and Molz, 1971, p. 177; Aziz and Settari, 1979, p. 261).

The preconditioned conjugate-gradient method (Concus, Golub and O'Leary, 1976) is an iterative method which can be used to solve matrix equations if the matrix is symmetric (matrix element $a_{ij} = a_{ji}$, where the first subscript is the matrix row number, and the second is the matrix column number) and positive-definite (all eigen values are positive) (see Hildebrand, 1965, p. 30 and 48 for further discussion of these terms). The matrix produced in ground-water-flow models always is symmetric and positive-definite. The preconditioned conjugate-gradient method has been the subject of considerable interest in recent years because of its efficiency and ability to solve difficult problems (Meijerink and van der Vorst, 1977). It works well, in part, because the required iteration parameters are calculated internally and need not be estimated.

Various preconditioners may be used in the preconditioned conjugate-gradient method. Among the different preconditioners there often is a direct relationship between increased efficiency and increased computer storage (Meijerink and van der Vorst, 1977). To avoid this tradeoff, the only preconditioners considered are those that produce a solver that has computer storage requirements less than or equal to the strongly implicit procedure (SIP) as programmed for the ground-water flow problem. SIP requires additional computer storage equal to four arrays with dimensions equal to the number of grid nodes (McDonald and Harbaugh, 1988, chap. 12).

The incomplete Cholesky preconditioner (ICCG) has been very popular (Meijerink and van der Vorst, 1977; Kuiper, 1981, 1987). However, alternative methods of matrix preconditioning have been developed to achieve more efficient conjugate-gradient solvers. Axelsson and Lindskog (1986) presented a preconditioner that commonly is called the modified incomplete Cholesky preconditioner (MICCG). It is similar to preconditioners presented by Dupont and others (1968), Gustafsson (1978), Wong (1979), and Ashcraft and Grimes (1988). In this paper, MICCG refers to Axelsson and Lindskog's (1986) method. Hill (in press) showed that MICCG was more efficient than ICCG in solving eight ground-water flow test cases on a scalar computer. Saad (1985) presented a least-squares polynomial preconditioner (POLCG), in which the matrix inverse is approximated by a truncated Neuman polynomial series. It is similar to preconditioners presented by Dubois and others (1979) and Johnson and others (1983). In this paper POLCG refers to Saad's (1985) method.

POLCG is not as efficient as SIP or MICCG on scalar computers (Scandrett, 1989; Hill, in press), but is more efficient on vector computers (Scandrett, 1989; Meyer and others, 1989, p. 1445). Storage requirements are less than for SIP: POLCG requires additional storage equal to three arrays with dimensions equal to the number of grid nodes.

Many preconditioners were excluded from PCG2. Modifications of ICCG presented by Meijerink and van der Vorst (1977; 1981), Gustafsson (1978; 1979), Wong (1979), and Ashcraft and Grimes (1988) apparently converge in fewer iterations than ICCG or MICCG. These were not included in PCG2 because they require that several additional arrays with dimensions equal to the number of grid nodes be added to computer storage. Additional polynomial preconditioners have been presented in several papers (Saad, 1985; Ashby, 1987; Meyer and others, 1989). Based on Saad's results, the POLCG was chosen because it appears to be at least as efficient as the other polynomial methods and it is easier to use. However, for very large grids (greater than 100,000 cells) the optimal Chebyshev polynomial preconditioner (Meyer and others, 1989) may be more efficient, and users with large grids may wish to consider this alternative polynomial preconditioner.

Watts (1981) and Hill (in press) indicate that the greatest differences in solver efficiency on scalar computers occur for three-dimensional, nonlinear problems. Thus, for these types of problems, it may be well worth the time and effort to try more than one solver. SIP generally is a good alternative to consider.

Notation

The following notation is used in this work:

Underlined capital letters indicate matrices: $\underline{A}$

Underlined lower-case letters indicate column vectors: $\underline{r}$

The element located in matrix row i and column j is designated as follows: matrix $\underline{A}$, element a_{ij}.

Exception: a single index is used to simplify notation in some cases. These are described in the text.

THREE-DIMENSIONAL GROUND-WATER FLOW MODEL

In this work, selected numerical methods are presented for solving the matrix equations that arise when the finite-difference method is used to discretize the ground-water flow equation as applied to a two-dimensional aquifer or a three-dimensional layered aquifer system. The finite-difference model is described in detail by McDonald and Harbaugh (1988), and only aspects relevant to this report are discussed here.

The finite-difference model produces a set of linear equations which can be expressed in matrix notation as:

$$\underline{A}\,\underline{x} = \underline{b} \quad , \tag{1}$$

where $\underline{A}$ is a coefficient matrix which is discussed below, $\underline{x}$ is a vector of hydraulic heads at each grid cell, and $\underline{b}$ is a vector of defined flows, terms associated with head-dependent boundary conditions, and storage terms (for

transient problems) at each grid cell. Because of the rigid structure of the finite-difference grid, and because there are six neighboring cells to each internal cell of a three-dimensional grid, $\underline{A}$ is symmetric and there may be as many as six off-diagonals in $\underline{A}$--three above the main diagonal and three below. The elements on the off-diagonals equal the negatives of the horizontal or vertical conductances between the centers of the cells which make up the finite-difference grid (fig. 1). The horizontal conductances along columns (c_n of fig. 1) equal $T_i T_{i+1} \Delta w / (T_i \Delta L_{i+1}/2 + T_{i+1} \Delta L_i/2)$, where T_i and T_{i+1} are transmissivities between two adjoining cells, Δw is the width of the two cells, and ΔL_i and ΔL_{i+1} are lengths of the two cells. Horizontal conductances along rows (r_n of fig. 1) are defined analogously. The vertical conductances (v_n of fig. 1) equal $K_z A/\Delta z$, where K_z is the vertical hydraulic conductivity, A is the area of the cell, and Δz is the vertical distance between the centers of the two cells. Each component on the main diagonal of $\underline{A}$, a_{ii}, equals:

$$a_{ii} = \sum_{\substack{j=1 \\ i \neq j}}^{N} (-a_{ij}) + w_i \ , \qquad (2)$$

where N is the total number of nodes in the grid; a_{ij} are the off-diagonal elements of row i, which are negative numbers; and w_i are the sum of the conductances associated with head-dependent boundaries and storage terms (for transient problems), which are positive numbers. Besides being symmetric, $\underline{A}$ is also positive-definite (its eigenvalues are always positive) (Varga, 1962, p. 23, 181-188; Hildebrand, 1965, p. 48), and these properties allow equation 1 to be solved using the methods presented in this work.

Nonlinearities occur if any aquifer is unconfined or if a head-dependent boundary condition is nonlinear. If any aquifer is unconfined, the horizontal conductances are a function of hydraulic head, and the main-diagonal and four of the six off-diagonals of matrix $\underline{A}$ must be updated during the solution process. If a head-dependent boundary condition is nonlinear, the boundary condition may change from being head-dependent to defined flux depending on the hydraulic head in the aquifer adjacent to the boundary, and the main diagonal of $\underline{A}$ and vector $\underline{b}$ (eq. 1) must be updated.

SOLUTION BY THE PRECONDITIONED CONJUGATE-GRADIENT METHOD

The preconditioned conjugate-gradient method for solving a set of linear equations is iterative. In iterative methods, it is assumed that the matrix $\underline{A}$ can be split into the sum of two matrices; that is $\underline{A} = \underline{M} + \underline{N}$ (Varga, 1962, p. 87-93; Remson and others, 1971, p. 177). $\underline{M}$ is called the preconditioned form of $\underline{A}$, and the goal is to define $\underline{M}$ so that it is easy to invert and resembles $\underline{A}$ as much as possible. These two criteria generally are impossible to satisfy simultaneously, and the optimal definition of $\underline{M}$ has been the focus of much research. In the preconditioned conjugate-gradient method, $\underline{M}$ must always be symmetric and positive definite. (This brief description of matrix splitting does not include important requirements that $\underline{M}$ and $\underline{N}$ must satisfy to achieve a convergent solver. Please refer to Varga (1962) for additional information).

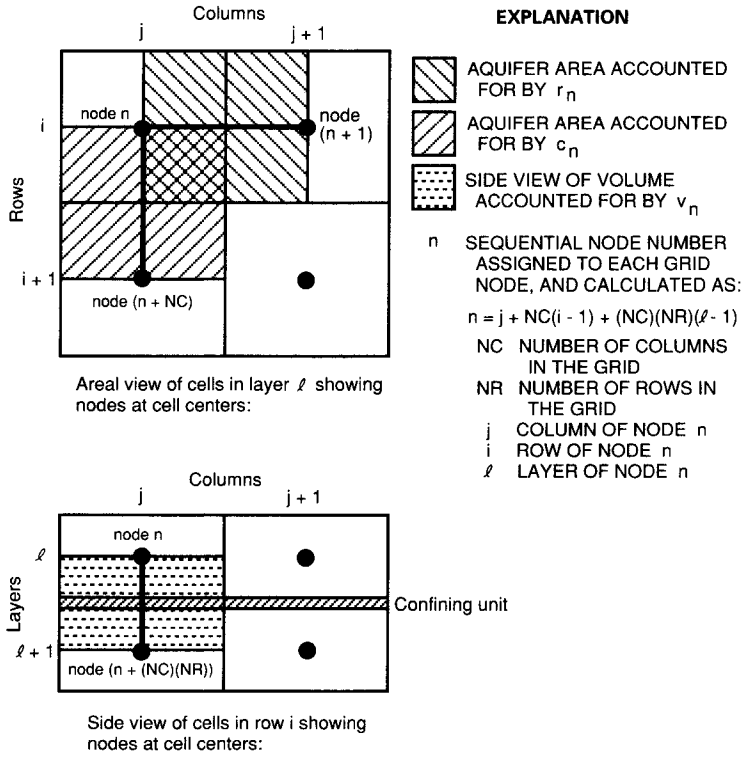


Figure 1.--Aquifer system volumes accounted for by conductances r_n, c_n, and v_n in the finite-difference method.

Once $\underline{M}$ has been defined, the basic iterative equation is developed from equation 1 and the splitting of $\underline{A}$, and can be written as:

$$\underline{M}\ \underline{x}_{k+1} = \underline{M}\ \underline{x}_k + \underline{b} - \underline{A}\ \underline{x}_k \qquad (3)$$

where k is the iteration index. Noting that $\underline{b} - \underline{A}\ \underline{x}_k$ is the residual ($\underline{r}_k$) of the original set of equations at the k^{th} iteration, and setting $\underline{s}_k = \underline{x}_{k+1} - \underline{x}_k$ gives:

$$\underline{M}\ \underline{s}_k = \underline{r}_k \qquad (4)$$

or

$$\underline{s}_k = \underline{M}^{-1} \underline{r}_k \quad . \tag{5}$$

The new heads may then be calculated as $\underline{x}_{k+1} = \underline{x}_k + \underline{s}_k$. More generally, some function of $\underline{s}_k$ may be used to calculate $\underline{x}_{k+1}$.

Conjugate-gradient methods are second-order iterative techniques because at each iteration the new change in $\underline{x}$, which is called $\underline{p}_k$, is calculated using the change from the prior iteration, $\underline{p}_{k-1}$, in addition to the vector $\underline{s}_k$ of equation 5. Conjugate-gradient methods begin by calculating $\underline{r}_o = \underline{b} - \underline{A}\,\underline{x}_o$. The following steps are executed for each iteration, starting with k = 0:

$$\underline{s}_k = \underline{M}^{-1} \underline{r}_k \tag{6a}$$

$$\text{for } k = 0 \qquad \underline{p}_k = \underline{s}_k \tag{6b}$$

$$\text{for } k > 0 \begin{cases} \beta_k = \dfrac{\underline{s}_k^T \underline{r}_k}{\underline{s}_{k-1}^T \underline{r}_{k-1}} & (6c) \\ \underline{p}_k = \underline{s}_k + \beta_k \underline{p}_{k-1} & (6d) \end{cases}$$

$$\alpha_k = \frac{\underline{s}_k^T \underline{r}_k}{\underline{p}_k^T \underline{A}\, \underline{p}_k} \tag{6e}$$

$$\underline{x}_{k+1} = \underline{x}_k + \alpha_k \underline{p}_k \tag{6f}$$

$$\underline{r}_{k+1} = \underline{r}_k - \alpha_k \underline{A}\, \underline{p}_k \tag{6g}$$

where the superscripted T indicates the transpose of the vector. Because $\underline{r}_{k+1}$ can be calculated using the last statement, $\underline{b}$ need not be saved within the solver. Iteration parameters β_k and α_k are calculated internally such that successive updating vectors, $\underline{p}_k$, are $\underline{A}$-orthogonal to previous $\underline{p}_k$ vectors -- that is, $\underline{p}_k^T \underline{A}\, \underline{p}_\ell = 0$, $k \neq \ell$ (Hestenes and Stiefel, 1952).

Whether or not an iterative method will converge and, if so, how fast depends on the preconditioner and how $\underline{x}_k$ is updated. A discussion of convergence is beyond the scope of this report, but references are cited for the convergence properties of each solver. See Varga (1962) for the general theory of convergence of iterative methods.

Scaling of the matrix $\underline{A}$ simplifies POLCG (Dubois and others, 1979), and is accomplished as:

$$\underline{B} = \underline{S}^T \underline{A}\, \underline{S} \ , \tag{7}$$

where $\underline{B}$ is the scaled matrix, all off-diagonal entries of $\underline{S}$ are zero, and

$$s_{ii} = \frac{1}{\sqrt{a_{ii}}} \quad . \tag{8}$$

This type of scaling is called diagonal scaling, and it preserves the symmetry of the original matrix. Diagonal scaling may improve the matrix characteristics that are most important to convergence because the scaled matrix is still symmetric and positive definite, and the condition number of $\underline{A}$ (the largest eigenvalue divided by the smallest eigenvalue) is minimized (Forsythe and Strauss, 1955). However, although $\underline{A}$ is diagonally dominant because:

$$a_{ii} \geq \sum_{\substack{j=1 \\ i \neq j}}^{N} |a_{ij}| \quad , \tag{9}$$

$\underline{B}$ may not be diagonally dominant if some of the values along the diagonal of $\underline{A}$ are much smaller than others. For example, consider the following original and scaled matrices:

$$\underline{A} = \begin{bmatrix} 0.9 & -0.1 & -0.75 \\ -0.1 & 0.2 & 0.0 \\ -0.75 & 0.0 & 0.8 \end{bmatrix} \qquad \underline{B} = \begin{bmatrix} 1.0 & -0.23 & -0.88 \\ -0.23 & 1.0 & 0.0 \\ -0.88 & 0.0 & 1.0 \end{bmatrix}$$

The first row of $\underline{B}$ is no longer diagonally dominant. The lack of diagonal dominance can cause problems when using MICCG. Scaling also may cause more rounding and truncation errors because all components of the diagonal of $\underline{A}$ must be summed, as in equation 2. Without scaling, the conductance terms can be manipulated individually to reduce rounding and truncation errors (Dorn and McCracken, 1972, p. 94). Scaling was only used for POLCG.

The precision with which numbers are stored in the computer can significantly affect solver performance. For example, making the four arrays required by SIP double precision (14 to 15 significant digits on the computer used) instead of single precision (6 to 7 significant digits) can make the difference between convergence and nonconvergence for some problems (McDonald and Harbaugh, 1988, p. A-2; A.W. Harbaugh, U.S. Geological Survey, written commun., 1989). However, increasing the precision of arrays doubles the required computer storage space. All the arrays required by the solvers presented in this work were declared as single precision. Double-precision scalar variables were used to improve the accuracy of calculations, where possible. The only double-precision array in the model is, then, the array used to store calculated heads (McDonald and Harbaugh, 1988, p. A-2). In PCG2, problems caused by the limited precision of the solver arrays are most prevalent for POLCG (Hill, in press). If the limited precision of the solver arrays is suspected as the cause of convergence problems, arrays V, SS, and P may be converted to double precision by doubling their allocated storage in PCG2AL, and declaring them as double precision in PCG2AP and SPCG2E.

Nonlinear problems are solved by Picard iterations, in which the matrix $\underline{A}$ and vector $\underline{b}$ are periodically updated between iterations using the newly calculated heads. For nonlinear problems, convergence using the conjugate-gradient solvers was found to be most efficient if several iterations (called inner iterations in this report) were accomplished between Picard iteration updates (Kuiper, 1981 and 1987). This allows the solver to use equations 6c and 6d to calculate several orthogonal $\underline{p}_k$ vectors before updating $\underline{A}$ and $\underline{b}$, and thus take advantage of the orthogonality of the conjugate-gradient method.

The total number of iterations equals the sum of the inner iterations for all updates of $\underline{A}$ and $\underline{b}$. For any one $\underline{A}$ and $\underline{b}$, the inner iterations continue until one of the following occurs: (1) the user-defined maximum number of inner iterations (ITER1 of the input file) are executed; or (2) the final convergence criteria are met. Outer iterations continue until the final convergence criteria are met on the first inner iteration after an update. The total number of iterations required is minimized by adjusting ITER1 and re-executing the problem. For most problems, the optimal value of ITER1 ranges from 3 to 10.

In the absence of round-off error, only inner iterations would be required for linear problems. However, in practice, round-off error may adversely affect the residual calculated by the conjugate-gradient method (eq. 6g) when more than 50 iterations are required. Recalculating the residual as $\underline{r} = \underline{b} - \underline{A}\,\underline{x}$ occasionally by limiting the number of inner iterations to less than 50 in linear problems alleviates the error. This can be accomplished using ITER1 of the input file.

A flowchart which displays the steps discussed above is presented in the section "Documentation of PCG2" of this report.

MICCG

In modified incomplete Cholesky preconditioning, $\underline{M} = \underline{U}^T\,\underline{D}\,\underline{U}$, where $\underline{U}$ is an upper triangular matrix with nonzero values along the main diagonal and at off-diagonal locations where $\underline{A}$ has nonzero values. $\underline{D}$ is a positive diagonal matrix with $d_{ii} = 1/u_{ii}$. When $\underline{A}$ is structured as in the finite-difference model with natural ordering of the nodes and there are more than two columns in the grid, the off-diagonal components of $\underline{U}$ equal the off-diagonal components of $\underline{A}$. That is, $u_{ij} = a_{ij}$, for $j > i$. As an example, figure 2 shows $\underline{U}$ for a problem with 2 rows, 3 columns and 2 layers. To more clearly indicate the physical quantities involved, the variables r_n, c_n, and v_n, which are depicted in figure 1 and were described earlier in this paper, are used. To be consistent, the same subscript, n, is used for the diagonal of matrix $\underline{U}$, so that u_{ii} now becomes u_n, where $n = i$.

Calculation of the u_{ii} is explained by executing the matrix multiplication for the simple problem shown in figure 2. In matrix $\underline{U}^T\,\underline{D}\,\underline{U}$ (fig. 3), $-r_n$, $-c_n$, and $-v_n$ appear in the same places they occupied in the $\underline{A}$ matrix. The additional off-diagonal terms occur because $\underline{U}^T\,\underline{D}\,\underline{U}$ is an incomplete factorization of $\underline{A}$. The u_n are defined such that the sum of the elements along a row of $\underline{U}^T\,\underline{D}\,\underline{U}$ equals the sum of the elements along the same row of $\underline{A}$. To accomplish this for the matrix shown in figure 3:

$$
\begin{bmatrix}
u_1 & -r_1 & 0 & -c_1 & 0 & 0 & -v_1 & 0 & 0 & 0 & 0 & 0 \\
 & u_2 & -r_2 & 0 & -c_2 & 0 & 0 & -v_2 & 0 & 0 & 0 & 0 \\
 & & u_3 & 0 & 0 & -c_3 & 0 & 0 & -v_3 & 0 & 0 & 0 \\
 & & & u_4 & -r_4 & 0 & 0 & 0 & 0 & -v_4 & 0 & 0 \\
 & & & & u_5 & -r_5 & 0 & 0 & 0 & 0 & -v_5 & 0 \\
 & & & & & u_6 & 0 & 0 & 0 & 0 & 0 & -v_6 \\
 & & & & & & u_7 & -r_7 & 0 & -c_7 & 0 & 0 \\
 & & & & & & & u_8 & -r_8 & 0 & -c_8 & 0 \\
 & & & & & & & & u_9 & 0 & 0 & -c_9 \\
 & & & & & & & & & u_{10} & -r_{10} & 0 \\
 & & & & & & & & & & u_{11} & -r_{11} \\
 & & & & & & & & & & & u_{12}
\end{bmatrix}
$$

Figure 2.--Matrix $\underline{U}$ for MICCG. The r_n, c_n, and v_n are the conductances along rows and columns and between layers, respectively.

$$
\begin{aligned}
u_1 &= a_{11} \quad , \\
u_2 &= a_{22} - \frac{r_1^2}{u_1} - \frac{r_1 c_1}{u_1} - \frac{r_1 v_1}{u_1} \quad , \\
u_3 &= a_{33} - \frac{r_2^2}{u_2} - \frac{r_2 c_2}{u_2} - \frac{r_2 v_2}{u_2} \quad , \\
&\text{etc.}
\end{aligned}
\tag{10}
$$

The general algorithm can be expressed as (R.L. Cooley, written commun., 1988):

$$
u_{ii} = a_{ii} - \sum_{k=1}^{i-1} \frac{u_{ki}^2}{u_{kk}} - \alpha \left(\sum_{j=1}^{i-1} f_{ji} + \sum_{j=i+1}^{N} f_{ij} \right) \tag{11a}
$$

$$
\begin{bmatrix}
u_1 & -r_1 & 0 & -c_1 & 0 & 0 & -v_1 & 0 & 0 & 0 & 0 & 0 \\
 & u_2+\xi_1 & -r_2 & \varphi_1 & -c_2 & 0 & \theta_1 & -v_2 & 0 & 0 & 0 & 0 \\
 & & u_3+\xi_2 & 0 & \varphi_2 & -c_3 & 0 & \theta_2 & -v_3 & 0 & 0 & 0 \\
 & & & u_4+\tau_1 & -r_4 & 0 & \phi_1 & 0 & 0 & -v_4 & 0 & 0 \\
 & & & & u_5+\xi_4+\tau_2 & -r_5 & 0 & \phi_2 & 0 & \theta_4 & -v_5 & 0 \\
 & & & & & u_6+\xi_5+\tau_3 & 0 & 0 & \phi_3 & 0 & \theta_5 & -v_6 \\
 & & & & & & u_7+\gamma_1 & -r_7 & 0 & -c_7 & 0 & 0 \\
 & & & & & & & u_8+\xi_7+\gamma_2 & -r_8 & \varphi_7 & -c_8 & 0 \\
 & & & & & & & & u_9+\xi_8+\gamma_3 & 0 & \varphi_8 & -c_9 \\
 & & & & & & & & & u_{10}+\tau_7+\gamma_4 & -r_{10} & 0 \\
 & & & & & & & & & & u_{11}+\xi_{10}+\tau_8+\gamma_5 & -r_{11} \\
 & & & & & & & & & & & u_{12}+\xi_{11}+\tau_9+\gamma_6
\end{bmatrix}
$$

symmetric

$$\xi_n = \frac{r_n^2}{u_n} \qquad \tau_n = \frac{c_n^2}{u_n} \qquad \gamma_n = \frac{v_n^2}{u_n} \qquad \varphi_n = \frac{r_n c_n}{u_n} \qquad \theta_n = \frac{r_n v_n}{u_n} \qquad \phi_n = \frac{c_n v_n}{u_n}$$

Figure 3.--Matrix $\underline{M} = \underline{U}^T \underline{D} \underline{U}$ for MICCG. The r_n, c_n, and v_n are the conductances along rows and columns and between layers, respectively.

where,

$$f_{ij} = \begin{cases} \sum_{k=1}^{i-1} u_{ki}\, u_{kj}/u_{kk} & a_{ij} = 0 \\ = 0 & a_{ij} \neq 0 \end{cases} , \qquad (11b)$$

and

$$u_{ij} = 0 \text{ for } j < i \ .$$

Again, the more general notation for components of $\underline{U}$ is used. Note that the f_{ji} and f_{ij} are equivalent to the ψ_n, θ_n, and ϕ_n of figure 3, and that they occur off the main diagonal of $\underline{U}^T \underline{D} \underline{U}$. The variable α is a user-defined relaxation parameter and is used to diminish the value of the f_{ij} of equation 11a. Ashcraft and Grimes (1988) found that using a relaxation parameter value of 0.97, 0.98, or 0.99 instead of 1.00 sometimes improved convergence by as much as 50 percent. However, consistently using a value of 1.00 for the relaxation parameter generally produces solutions which are at least as efficient as those attained with other commonly used solvers (Hill, in press).

The equation $\underline{U}^T \underline{D} \underline{U} \underline{s}_k = \underline{r}_k$ is solved by a two-step process. First, $\underline{U}^T \underline{v}_k = \underline{r}_k$ is solved for $\underline{v}_k$ by forward substitution, then $\underline{D} \underline{U} \underline{s}_k = \underline{v}_k$ is solved for $\underline{s}_k$ by backward substitution.

The MICCG preconditioned matrix presented in this paper is from Axelsson and Lindskog (1986), and is nearly identical to that presented by Dupont and others (1968), Gustafsson (1978, eq. 3.1, and 1979, eq. 6), Wong (1979), and Ashcraft and Grimes (1988). In the method used in this report, the overcompensation parameter used by Gustafsson (1978) to augment a_{ii} of equation 11a equals zero. Ashcraft and Grimes (1988) found that the number of iterations required to achieve convergence was insensitive to values of the overcompensation factor, so using a value of zero should not affect convergence.

Convergence properties of incomplete Cholesky with row-sums agreement are discussed by Gustafsson (1978).

POLCG

In Neuman series polynomial preconditioning, $\underline{M}^{-1}$ equals the sum of several terms of a power-series expansion for the inverse of matrix $\underline{A}$ (Dubois and others, 1979; Johnson and others, 1983; Saad, 1985), so that

$$\underline{M}^{-1} = \underline{I} + \underline{A} + \underline{A}^2 + \cdots + \underline{A}^{\ell} \ . \qquad (12)$$

Then, by weighting the terms as suggested by Johnson and others (1983) and Saad (1985), an approximate solution can be written as

$$\underline{s}_k = \underline{M}^{-1} \underline{r}_k = c_0 \underline{r}_k + c_1 \underline{A} \underline{r}_k + c_2 \underline{A}^2 \underline{r}_k + \underline{A}^3 \underline{r}_k \ , \qquad (13)$$

when $\ell = 3$, and c_0, c_1, and c_2 are coefficients chosen to optimize convergence. For computational efficiency, equation 13 is calculated by the following series of steps:

$$\begin{aligned} \underline{z}_1 &= c_2 \underline{r}_k + \underline{A} \underline{r}_k \\ \underline{z}_2 &= c_1 \underline{r}_k + \underline{A} \underline{z}_1 \\ \underline{s}_k &= c_0 \underline{r}_k + \underline{A} \underline{z}_2 \end{aligned} \qquad (14)$$

so that the powers of $\underline{A}$ are never formed explicity (Dubois and others, 1979, p. 259). One of the advantages of POLCG is that the steps of equation 14 are efficient on vector and parallel computers.

The coefficients can be calculated using the method described by Saad (1985, p. 869-871; 880) as: $c_0 = \frac{-15}{32} g^3$, $c_1 = \frac{27}{16} g^2$, $c_2 = \frac{-9}{4} g$, where g is the upper bound on the maximum eigenvalue of A, estimated as the largest sum of the absolute values of the components in any row of $\underline{A}$ (Varga, 1962, p. 17; Gerschgorin, 1931). For a scaled matrix, g is generally close to 2. Scandrett (1989) and Hill (in press) used g=2, and the number of iterations required to achieve solutions in their test cases were generally insensitive to changes in g. Using NBPOL of the input file, the user can specify that g=2 or that g is to be estimated as described above. Estimation of g uses slightly more execution time per iteration.

Convergence properties of polynomial preconditioners are discussed by Saad (1985).

Convergence Criteria

An iterative matrix solver is assumed to have converged when some measure of the residual and(or) the difference in results between successive iterations is less than user-specified convergence criteria. In PCG2, the difference between results of successive iterations is measured using both the maximum absolute value of the change in hydraulic head and the maximum absolute value of the residual for that iteration. Typical values for these error criteria are 0.01 ft and 0.01 ft^3/s, respectively.

The defined convergence criteria are too large if the global ground-water flow budget errors calculated by the modular model (McDonald and Harbaugh, 1988, p. 3-16 to 3-22) are unacceptably large. What is unacceptable depends on the problem being considered and must be determined by the user. For most ground-water flow problems, global budget errors greater than one percent are unacceptable. If unacceptably large global budget errors occur, the error criteria should be reduced.

The defined convergence criteria are too small if the accuracy achieved by the solver exceeds the accuracy required by the user. For example, Hill (in press) found that reducing both error criteria from 10^{-3} to 10^{-6} increased the execution time by as much as 55 percent, and, especially for POLCG, resulted in lack of convergence in some test cases. If the solver is taking more iterations than expected to achieve convergence and the calculated global budget error is smaller than required by the user, or if the solver is not converging and the convergence criteria are very small, the user can increase the convergence criteria.

INPUT INSTRUCTIONS

Input for PCG2 is read from a unit specified in the IUNIT array of the Basic Package input file. In the example provided in this report, IUNIT(13) is used, but this can easily be changed, as noted below in the section "Linking this program to the modular model". The input for PCG2 is as follows.

FOR EACH SIMULATION

PCG2AL

1.	Data:	MXITER	ITER1	NPCOND
	Format:	I10	I10	I10

PCG2RP

2.	Data:	HCLOSE	RCLOSE	RELAX	NBPOL	IPRPCG	MUTPCG	IPCGCD
		F10.0	F10.0	F10.0	I10	I10	I10	I10

Explanation of Fields Used in Input Instructions

MXITER--is the maximum number of outer iterations -- that is, calls to the solution routine. For a linear problem MXITER should be 1, unless more than 50 inner iterations are required, when MXITER could be as large as 10. A larger number (generally less than 100) is required for a nonlinear problem.

ITER1---is the maximum number of inner iterations. For nonlinear problems, ITER1 usually ranges from 3 to 10; a value of 30 will be sufficient for most linear problems.

NPCOND--is the flag used to select the matrix preconditioning method. The following options are available.

NPCOND	PRECONDITIONING METHOD
1	Modified Incomplete Cholesky (for use on scalar computers)
2	Polynomial (for use on vector computers or to conserve computer storage)

HCLOSE--is the head change criterion for convergence, in units of length. When the maximum absolute value of the head change at all nodes during an iteration is less than or equal to HCLOSE, and the criterion for RCLOSE is satisfied (see below), iteration stops. Commonly, HCLOSE equals 0.01.

RCLOSE--is the residual criterion for convergence, in units of cubic length per time. When the maximum absolute value of the residual at all nodes during an iteration is less than or equal to RCLOSE, and the criterion for HCLOSE is satisfied (see above), iteration stops. Commonly, RCLOSE equals HCLOSE.

For nonlinear problems, convergence is achieved when the convergence criteria are satisfied on the first inner iteration.

RELAX---is the relaxation parameter used with NPCOND=1 (MICCG). Usually, RELAX=1.0, but for some problems a value of 0.99, 0.98, or 0.97 will reduce the number of iterations required for convergence. RELAX is not used if NPCOND≠1.

NBPOL---is used when NPCOND=2 to indicate whether the estimate of the upper bound on the maximum eigenvalue is 2.0, or whether the estimate will be calculated. NBPOL=2 is used to specify the value as 2.0; for any other value of NBPOL, the estimate is calculated. Convergence is generally insensitive to this parameter. NBPOL is not used if NPCOND does not equal 2.

IPRPCG--is the printout interval for PCG. If IPRPCG is equal to zero, it is changed to 999. The extreme head change and residual (positive or negative) are printed for each iteration of a time step whenever the time step is an even multiple of IPRPCG. The printout also occurs at the end of each stress period regardless of the value of IPRPCG.

MUTPCG--is a flag which controls printing from the solver. If MUTPCG≠0, printing from the solver is suppressed. If MUTPCG=1, the number of iterations is printed, but the lists of extreme head changes and residuals is suppressed. If MUTPCG=2, all printing is suppressed.

IPCGCD--is a flag which is used when NPCOND=1 to control whether the same Cholesky decomposition may be used for multiple calls to PCG2AP. IPCGCD should be zero for most applications. However, future packages might benefit from nonzero values of IPCGCD.

SAMPLE DATA INPUTS

Example data set for a linear problem:

```
        10        20        30        40        50
123456789 123456789 123456789 123456789 123456789
         1        99         2
      .001      .001        1.         2         1
```

Example data set for a nonlinear problem:

```
        10        20        30        40        50
123456789 123456789 123456789 123456789 123456789
        10         5         1
       .01       .01        1.         2         1
```

LINKING PCG2 TO THE MODULAR MODEL

The following statements must be included in the main program of the modular model. Note that IUNIT(13) can be changed to allow the PCG2 input unit number to be read from a different position of the IUNIT array. IUNIT(15) is the input number for another package which may benefit from IPCGCD≠0. The "15" can be changed if this is not applicable for the package represented by IUNIT(15).

```
C     ADD BETWEEN COMMENT STATEMENTS 4 AND 5
       IF(IUNIT(13).GT.0) CALL PCG2AL(ISUM,LENX,LCV,LCSS,LCP,LCCD,
      1     LCHCHG,LCLHCH,LCRCHG,LCLRCH,MXITER,ITER1,NCOL,NROW,NLAY,
      2     IUNIT(13),IOUT,NPCOND)

C     ADD BETWEEN COMMENT STATEMENTS 6 AND 7
       IF(IUNIT(13).GT.0) CALL PCG2RP(MXITER,ITER1,HCLOSE,RCLOSE,
      1     NPCOND,NBPOL,RELAX,IPRPCG,IUNIT(13),IOUT,MUTPCG,IPCGCD)

C     ADD BETWEEN COMMENT STATEMENTS 7C2B AND 7C2C
       ICD=0
       IF(IUNIT(13).GT.0) CALL PCG2AP(X(LCHNEW),XICD=0(LCIBOU),X(LCCR),
      1     X(LCCC),X(LCCV),X(LCHCOF),X(LCRHS),X(LCV),X(LCSS),X(LCP),
      2     X(LCCD),X(LCHCHG),X(LCLHCH),X(LCRCHG),X(LCLRCH),KITER,
      3     NITER,HCLOSE,RCLOSE,ICNVG,KSTP,KPER,IPRPCG,MXITER,ITER1,
      4     NPCOND,NBPOL,NSTP,NCOL,NROW,NLAY,NODES,RELAX,IOUT,MUTPCG,
      5     IPCGCD,STEPL,DELT,IUNIT(15),IP)
```

LAYCON=3 layers may produce a matrix which is not diagonally dominant when using the BCF package as presented in McDonald and Harbaugh (1988, p. 5-22 and 5-59). MICCG (NPCOND=1) of the PCG2 package will not function if the matrix is not diagonally dominant, and POLCG may not converge. To correct this, make the following change in module BCF1FM (M.G. McDonald, U.S. Geological Survey, written commun., 1989):

McDonald and Harbaugh (1988) version:

```
C7D-----WITH HEAD BELOW TOP ADD CORRECTION TERMS TO RHS AND HCOF.
      RHS(J,I,K)=RHS(J,I,K) + CV(J,I,K-1)*TOP(J,I,KT)
      HCOF(J,I,K)=HCOF(J,I,K) + CV(J,I,K-1)
  220 CONTINUE
```

Modified version:

```
C7D-----WITH HEAD BELOW TOP ADD CORRECTION TERMS TO RHS AND HCOF.
C7D-----MODIFIED TO PUT CORRECTION COMPLETELY ONTO RIGHT HAND SIDE
      RHS(J,I,K)=RHS(J,I,K) + CV(J,I,K-1)*(TOP(J,I,KT)-HTMP)
  220 CONTINUE
```

After making the required changes and including the FORTRAN listed in this document, compile and load the modular model as usual.

DOCUMENTATION OF PCG2

Brief Description of Modules

Three primary modules and two submodules were created for the preconditioned conjugate-gradient method. The following is a brief description of the purpose of each of these modules:

Primary modules:

PCG2AL Allocates space for the conjugate-gradient calculations.

PCG2RP Reads, stores and prints the input data.

PCG2AP Performs multiple iterations of the conjugate-gradient method and checks the convergence criteria.

Submodules:

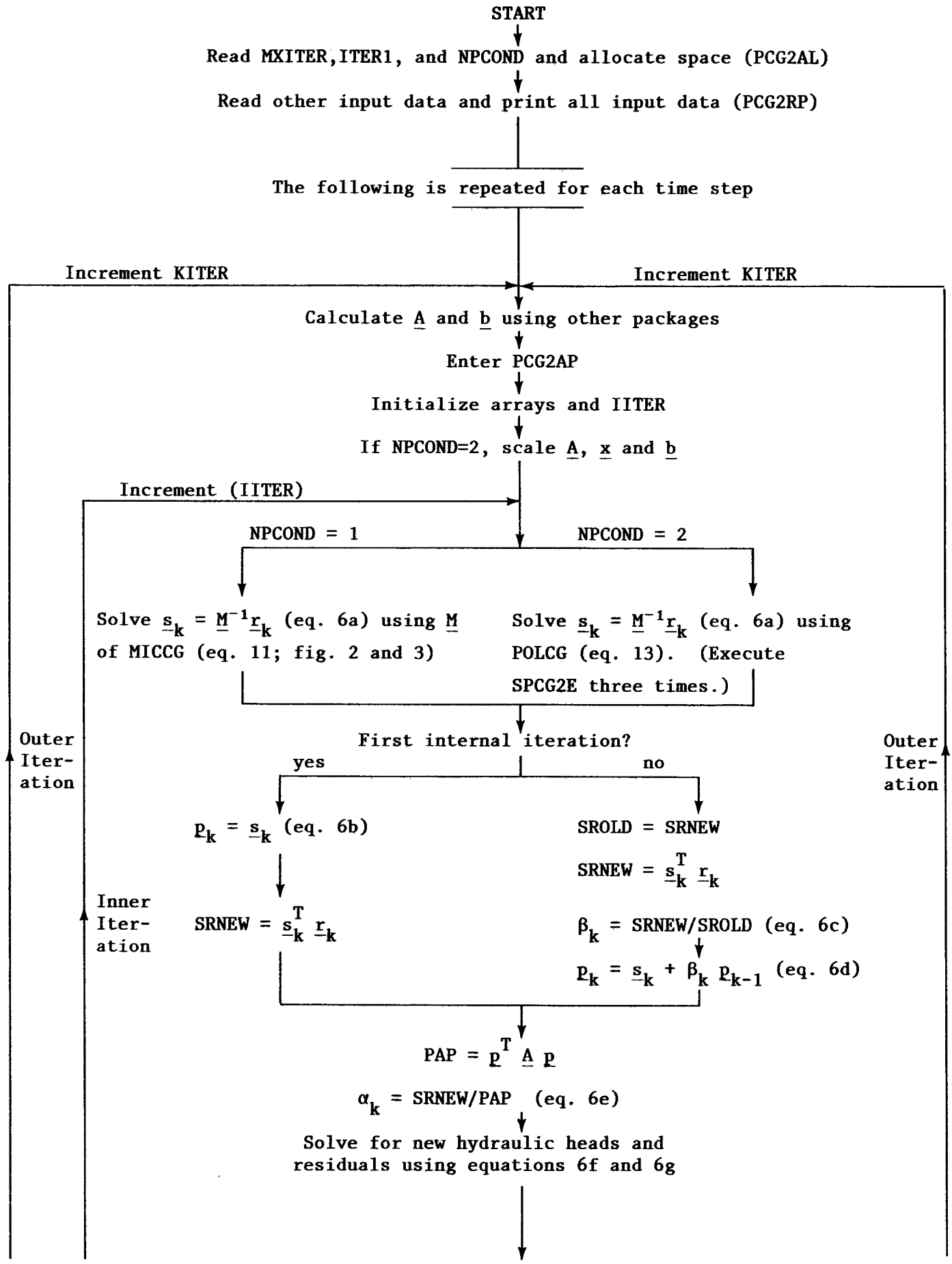
SPCG2P Called by PCG2AP to print the extreme head changes and residuals that occurred at each iteration.

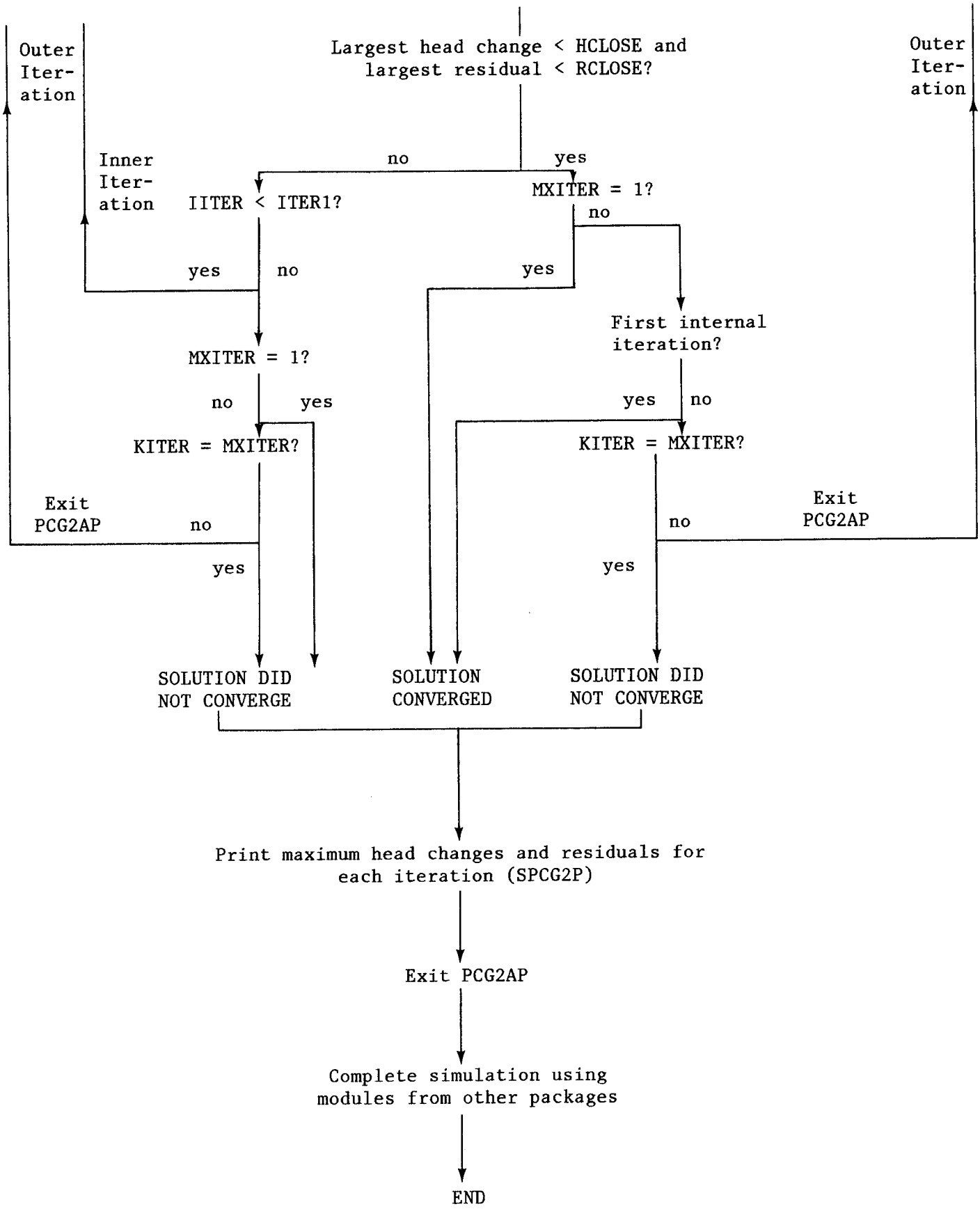
SPCG2E Called by PCG2AP to perform one matrix multiplication required by the polynomial preconditioner. This submodule is executed three times for each iteration of POLCG.

Flowchart

The flowchart for the package is shown below, and includes the functions performed by the three primary modules and the two submodules. The following variable names used in the flowchart are taken from the FORTRAN code. Some were defined in the input instructions; all are defined later in the following list of variables.

HCLOSE, IITER, ITER1, MXITER,
NPCOND, PAP, RCLOSE, SRNEW, SROLD


START
Read MXITER,ITER1, and NPCOND and allocate space (PCG2AL)
Read other input data and print all input data (PCG2RP)
The following is repeated for each time step
Increment KITER
Increment KITER
Calculate A and b using other packages
Enter PCG2AP
Initialize arrays and IITER
If NPCOND=2, scale A, x and b
Increment (IITER)
NPCOND = 1
NPCOND = 2
Solve s_k = M^-1 r_k (eq. 6a) using M of MICCG (eq. 11; fig. 2 and 3)
Solve s_k = M^-1 r_k (eq. 6a) using POLCG (eq. 13). (Execute SPCG2E three times.)
Outer Iteration
First internal iteration?
yes
no
Outer Iteration
p_k = s_k (eq. 6b)
SROLD = SRNEW
SRNEW = s_k^T r_k
Inner Iteration
SRNEW = s_k^T r_k
β_k = SRNEW/SROLD (eq. 6c)
p_k = s_k + β_k p_k-1 (eq. 6d)
PAP = p^T A p
α_k = SRNEW/PAP (eq. 6e)
Solve for new hydraulic heads and residuals using equations 6f and 6g


Outer Iter-ation
Largest head change < HCLOSE and largest residual < RCLOSE?
Outer Iter-ation
Inner Iter-ation
no
yes
IITER < ITER1?
MXITER = 1?
no
yes
no
yes
First internal iteration?
MXITER = 1?
yes
no
no
yes
KITER = MXITER?
KITER = MXITER?
Exit PCG2AP
no
Exit PCG2AP
no
yes
yes
SOLUTION DID NOT CONVERGE
SOLUTION CONVERGED
SOLUTION DID NOT CONVERGE
Print maximum head changes and residuals for each iteration (SPCG2P)
Exit PCG2AP
Complete simulation using modules from other packages
END

Narrative for Modules

PCG2AL

Module PCG2AL reads MXITER, ITER1, and NPCOND, and allocates space in the X array for the arrays required for the solver. Arrays SS, P and V are required for both values of NPCOND; array CD is required only for NPCOND=1. Each of these four arrays are vectors dimensioned equal to the number of grid nodes. Additionally, four smaller arrays are required to store the maximum head change and residual at each iteration (HCHG and RCHG) and the cell locations where these occurred (LHCH and LRCH).

PCG2RP

Module PCG2RP reads HCLOSE, RCLOSE, RELAX, NBPOL, IPRPCG, MUTPCG, and IPCGCD, and prints all variables read for this package.

PCG2AP

Module PCG2AP performs up to ITER1 iterations of the preconditioned conjugate-gradient algorithm for solving the flow equation. To save computational time, all arrays are declared one dimensional. They are accessed by a single index which is calculated from the layer, row, and column indexes normally used to access the arrays in three dimensions.

Double precision scalar variables are used for most calculations in this module to improve the accuracy of the results. Modification of the present use of double precision may affect simulated results.

For the polynomial preconditioner, c_0 and c_1 of equation 14 are negative in the text, but are calculated as positive numbers in the FORTRAN code. The FORTRAN code is correct because, as programmed in the modular model, matrix $\underline{A}$ of equation 1 is negative definite instead of positive definite. While this poses no mathematical difficulty, it does require that the odd-numbered coefficients of equation 14 be positive instead of negative.

The steps executed by PCG2AP were outlined in the flowchart previously presented in this section.

SPCG2P

Submodule SPCG2P prints the extreme values of the head change (HCHG) and residual (RCHG) out of all cells for each iteration of a time step. The cell location (LHCH and LRCH) where the values occur also is printed.

SPCG2E

Submodule SPCG2E calculates the matrix-vector multiplication and the vector addition required by each of the three parts of equation 14. It is called three times for each polynomial iteration.

Adapting SPCG2E for Computers with Vector and Parallel Architecture

Use of the polynomial preconditioner on computers with vector and(or) parallel architecture will be most efficient if SPCG2E is modified to take advantage of the computer used. The following points should be considered when modifying SPCG2E.

The diagonal entries of $\underline{A}$ all equal -1.0 because $\underline{A}$ is scaled and is negative definite (see narrative for modeule PCG2AP). Vectors CR, CC, and CV contain the off-diagonals of $\underline{A}$ along which nonzero entries occur. Note that although array CV is dimensioned using NODES in SPCG2E and elsewhere in the model, it is originally only given storage space in X for cells in NLAY-1 layers (McDonald and Harbaugh, 1988, p. 4-25). In making changes, care must be taken to avoid overwriting elements of array HCOF, which is stored in X after CV.

Using $\underline{w}$ to represent $\underline{r}_k$, $\underline{z}_1$, or $\underline{z}_2$, and n to represent the cell number, the nth element of each vector produced by the matrix-vector multiplications of equation 14 is calculated by summing -w(n) with:

CR(n) * w(n+1)		(15a)
CR(n) * w(n)		(15b)
CC(n) * w(n+NCOL)		(15c)
CC(n) * w(n)		(15d)
CV(n) * w(n+NRC)	n≤NODES-NRC	(15e)
CV(n) * w(n)		(15f)

Equations 15a through 15f are vector-vector multiplications which can be calculated quickly on computers with vector architecture, but one problem exists. Along with rows and columns for active cells, $\underline{A}$ includes rows and columns for inactive and constant-head cells. Inactive cells are accounted for by setting appropriate entries of CR, CC, and CV to zero in the beginning of the "DO 115" loop in PCG2AP and, therefore, cause no problem. However, constant-head cells are accounted for using the IF statements in SPCG2E, and these IF statements must be eliminated to vectorize the multiplications.

One way to eliminate the IF statements in SPCG2E is to add a work vector of length NODES (allocate space in PCG2AL) and use this vector to store CR, CC, or CV temporarily while performing the multiplications of equation 15 with the work vector. In the work vector, the entries along rows associated with constant-head cells can be set to zero prior to performing the multiplications. Thus, in equation 15a, CR(n)=0 if cell n+1 is constant head; in equation 15c, CC(n)=0 if cell n+NCOL is constant head; in equation 15e, CV(n)=0 if cell n+NRC is constant head. In equations 15b, 15d, and 15f, CR(n)=0, CC(n)=0, and CV(n)=0 if cell n is constant head.

Alternatively, vectors CR, CC, and CV may be used directly in equation 15, and the values which are set to zero may be stored in a separate array and replaced once the multiplication has been completed. This would require a work vector with length equal to the number of constant-head cells in the grid. At present, the number of constant-head cells is not available when space is allocated by calling PCG2AL, and the user would have to modify the data input and the module to provide this information.

List of Variables

Variables for the entire package are listed below. Variables not listed below are used only briefly in a few calculations and their meaning can be identified from nearby lines of the code. The name of the module is listed under 'Range' for variables used by only one module.

Variable	Range	Definition
ALPHA	PCG2AP	Double-precision field containing α_k of equation 6e.
BIGH	PCG2AP	Value of head change with the largest absolute value for one iteration.
BIGR	PCG2AP	Value of residual with the largest absolute value for one iteration.
BPOLY	PCG2AP	Estimated upper bound of the maximum eigen value of $\underline{A}$; used in polynomial preconditioning.
C0, C1, C2	PCG2AP	Scalar coefficients of equations 13 and 14.
CC	Global	DIMENSION (NCOL,NROW,NLAY), conductance along columns (see fig. 1).
CD	PCG2AP	DIMENSION (NCOL,NROW,NLAY), u_{ii} of equation 11a.
CD1	PCG2AP	The first nonzero component of CD. Used to ensure that all values in CD are all ≥ 0 or ≤ 0.
CDCC, CDCR, CDCV	PCG2AP	Double-precision fields containing u_{ki}^2/u_{kk} terms of equation 11a.
CR	Global	DIMENSION (NCOL,NROW,NLAY), conductance along rows (see fig. 1).
CV	Global	DIMENSION (NCOL,NROW,NLAY), conductance between layers (see fig. 1).
DONE	Package	Double-precision field containing a one.
DZERO	PCG2AP	Double-precision field containing a zero.
FCC, FCR, FCV	PCG2AP	Double-precision fields containing f_{ij} terms of equation 11.
HCHG	Package	DIMENSION (MXITER*ITER1), extreme head change (BIGH) for each iteration.
HCHGN	PCG2AP	Double-precision field containing the head change at one cell at one iteration.
HCLOSE	Package	Closure criteria for the head change for the iterative procedure.
HCOF	Global	DIMENSION (NCOL,NROW,NLAY), coefficient of head at cell (J,I,K) in the finite-difference equation.
HNEW	Global	DIMENSION (NCOL,NROW,NLAY), most recent estimate of head in each cell. HNEW changes at each iteration.
IBOUND	Global	DIMENSION (NCOL,NROW,NLAY), status of each cell <0, constant-head cell =0, inactive cell >0, variable-head cell
ICNVG	Global	Flag set equal to zero until the iteration procedure has converged, when it is set to one.
IICNVG	PCG2AP	Inner iteration convergence flag.
IITER	PCG2AP	Inner iteration counter. Reset each time PCG2AP is called.
IN	Package	Primary unit number from which input for this package will be read.

Variable	Range	Definition
IOUT	Global	Primary unit number for all printed output. IOUT=6.
IPRPCG	Package	Frequency (in time steps) with which the extreme head changes and residuals for each iteration will be printed.
ISIZ	PCG2AL	Number of cells (nodes) in the finite-difference 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	PCG2AL	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.
ITER1	Package	Maximum number of inner iterations.
KITER	Global	Counts the number of times PCG2AP is called.
KPER	Global	Stress period counter.
KSTP	Global	Time step counter. Reset at the start of each stress period.
LCname	Package	Location in the X array of the first element of array 'name'.
LENX	Global	Length of the X array in words. This should always be equal to the dimension of X specified in the MAIN program.
LHCH	Package	DIMENSION (3,MXITER*ITER1), Layer, row, and column of the cell containing the extreme head change (BIGH) for each iteration.
LRCH	Package	DIMENSION (3,MXITER*ITER1), Layer, row, and column of the cell containing the extreme residual (BIGR) for each iteration.
MXITER	Package	Maximum number of calls of PCG2AP.
MUTPCG	Package	Flag to control printing from the solver (see input instructions).
N	Package	Cell index.
NBPOL	Package	Used when NPCOND=2 to indicate how the value of the upper bound of the maximum eigenvalue is calculated (see input instructions).
NCD	Package	One-dimensional subscript of conductance to the adjacent cell, which is in the last column.
NCF	Package	One-dimensional subscript of conductance to the adjacent cell, which is in the next column.
NCL	Package	One-dimensional subscript of the cell index of the adjacent cell which is in the last column.
NCN	Package	One-dimensional subscript of the cell index of the adjacent cell which is in the next column.
NCOL	Global	Number of columns in the grid.
NITER	PCG2AP	Counts the total number of inner iterations that are executed.
NLAY	Global	Number of layers in the grid.
NLL	Package	One-dimensional subscript of the cell index of the adjacent cell which is in the last layer.
NLN	Package	One-dimensional subscript of the cell index of the adjacent cell which is in the next layer.

Variable	Range	Definition
NLS	Package	One-dimensional subscript of conductance to the adjacent cell which is in the next layer.
NLZ	Package	One-dimensional subscript of conductance to the adjacent cell which is in the last layer.
NODES	Global	Number of cells (nodes) in the finite-difference grid.
NORM	PCG2AP	Flag for scaling the matrix set of equations. NORM=1 and scaling is executed only when NPCOND=2.
NPCOND	Package	Preconditioner used: 1 Incomplete Cholesky with row-sums agreement 2 Polynomial
NRB	Module	One-dimensional subscript of conductance to the adjacent cell which is in the last row.
NRC	Package	Number of cells in a model layer.
NRH	Package	One-dimensional subscript of conductance to the adjacent cell which is in the next row.
NRL	Package	One-dimensional subscript of the cell index of the adjacent cell which is in the last row.
NRN	Package	One-dimensional subscript of the cell index of the adjacent cell which is in the next row.
NROW	Global	Number of rows in the grid.
NSTP	Global	Number of time steps in the current stress period.
P	PCG2AP	DIMENSION (NCOL,NROW,NLAY), $\underline{p}_k$ and $\underline{p}_{k-1}$ of equations 6d-6g.
PAP	PCG2AP	Double-precision field containing the denominator of equation 6e.
RCHG	PCG2AP	DIMENSION (MXITER*ITER1), Extreme residual (BIGR) for each iteration.
RCHGN	PCG2AP	Double-precision field containing the residual change at one cell at one iteration.
RCLOSE	Package	Closure criteria for the residual for the iterative procedure.
RELAX	Package	Relaxation parameter of equation 11a.
RHS	Global	DIMENSION (NCOL,NROW,NLAY), right-hand side of the finite-difference equation. RHS is an accumulation of terms from several different packages.
SROLD	PCG2AP	Double-precision field containing the denominator of equation 6c.
SRNEW	PCG2AP	Double-precision field containing the numnerator of equations 6c and 6e.
SS	PCG2AP	DIMENSION (NCOL,NROW,NLAY), $\underline{s}_k$ of equation 6a-6e.
V	PCG2AP	DIMENSION (NCOL,NROW,NLAY), intermediate solution when solving equation 6a, and when calculating PAP.

REFERENCES

Ashby, S.F., 1987, Polynomial preconditioning for conjugate gradient methods: Department of Computer Science, Report UIUCDCS-R-87-1355, University of Illinois, Urbana-Champaign, IL., 131 p.

Ashcraft, C.C., and Grimes, R.G., 1988, On vectorizing incomplete factorization and SSOR preconditioners: SIAM Journal of Scientific and Statistical Computing, v. 9, no. 1, p. 122-151.

Axelsson, O., and Lindskog, G., 1986, On the eigenvalue distribution of a class of preconditioning methods: Numerical Mathematics, v. 48, p. 479-498.

Aziz, Khalid and Settari, Antonin, 1979, Petroleum reservoir simulation, Elsevier, 476 p.

Concus, Paul, Golub, G.H., and O'Leary, D.P., 1976, A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations, in Bunch, J.P., and Rose, D.J., eds., Sparse Matrix computations, Academic Press, p. 303-332.

Dorn, W.S. and McCracken, D.D., 1972, Numerical methods with FORTRAN IV case studies: John Wiley, 447 p.

Dubois, P.F., Greenbaum, A., Rodrique, G.H., 1979, Approximating the inverse of a matrix for use in iterative algorithms on vector processors: Computing, vol. 22, p. 257-268.

Dupont, Todd, Kendall, R.P., and Rachford, H.H., Jr.,1968, An approximate factorization procedure for solving self-adjoint elliptic difference equations: SIAM Journal on Numerical Analysis, v. 5, no. 3, p. 559-573.

Forsythe, G.E. and Strauss, E.G., 1955, On best conditioned matrices: American Mathematical Society Proceedings, vol. 6, no. 3, p. 340-345.

Gerschgorin, S., 1931, Über die abrenzung der eigenwerte einer matrix: Isv. Akad. Nauk SSSR Ser. Mat., vol. 7, p. 749-754.

Gustafsson, Ivar, 1978, A class of first order factorization methods: BIT v. 18, p. 142-156.

_____1979, On modified incomplete Cholesky factorization methods for the solution of problems with mixed boundary conditions and problems with discontinuous material coefficients: International Journal for Numerical Methods in Engineering, v. 14, p. 1127-1140.

Hestenes, M.R. and Stiefel, Eduard, 1952, Methods of conjugate gradients for solving linear systems: Journal of Research for the National Bureau of Standards, vol. 49, no. 6, p. 409-436.

Hildebrand, F.B., 1965, Methods of applied mathematics: Prentice-Hall, Inc., 362 p.

Hill, M.C., in press, Solving ground-water flow problems by conjugate-gradient methods and the strongly implicit procedure: Water Resources Research.

Johnson, O.G., Micchelli, C.A., and Paul, George, 1983, Polynomial preconditioners for conjugate gradient calculations: SIAM Journal of Numerical Analysis, v. 20, no. 2, p. 362-376.

Kuiper, L.K., 1981, A comparison of the incomplete Cholesky-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. 1082-1086.

_____1987, Computer program for solving ground-water flow equations by the preconditioned conjugate gradient method: U.S. Geological Survey Water-Resources Investigations Report 87-4091, 34 p.

McDonald, M.G., and Harbaugh, A.W., 1988, A modular three-dimensional groundwater flow model: U.S. Geological Survey Techniques of Water-Resources Investigations, bk. 6, ch. A1, 548 p.

Meijerink, J.A., and van der Vorst, H.A., 1977, An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-Matrix: Mathematics of Computation, v. 31, no. 137, p. 148-162.

_____1981, Guidelines for the usage of incomplete decompositions in solving sets of linear equations as they occur in practical problems, Journal of Computational Physics, v. 44, p. 134-155.

Meyer, P.D., Valocchi, A.J., Ashby, S.F., and Saylor, P.E., 1989, A numerical investigation of the conjugate gradient method as applied to three-dimensional groundwater flow problems in randomly heterogeneous porous media: Water Resources Research, vol., 25, no. 6, p. 1440-1446.

Remson, Irwin, Hornberger, G.M. and Molz, F.J., 1971, Numerical methods in subsurface hydrology: John Wiley, 389 p.

Saad, Y., 1985, Practical use of polynomial preconditionings for the conjugate gradient method: SIAM Journal of Scientific and Statistical Computing, v. 6, no. 4, p. 865-881.

Varga, R.S., 1962, Matrix iterative analysis: Prentice-Hall, Inc., 322 p.

Watts, J.W., 1981, A conjugate gradient-truncated direct method for the iterative solution of the reservoir simulation pressure equation: Society of Petroleum Engineers, v. 21, no. 3, p. 345-353.

Wong, Y.S., 1979, Pre-conditioned conjugate gradient methods for large sparse matrix problems, *in* Proceedings of the First International Conference on Numerical Methods in Thermal Problems, Swansea, U.K., July 2-6, 1979, eds. R.W. Lewis and K. Morgan, p. 967-979.

FORTRAN LISTING

```
      SUBROUTINE PCG2AL(ISUM,LENX,LCV,LCSS,LCP,LCCD,LCHCHG,LCLHCH,
     1       LCRCHG,LCLRCH,MXITER,ITER1,NCOL,NROW,NLAY,IN,IOUT,NPCOND)
C
C-----VERSION 0002 01MAY1989 PCG2AL
C
C     ******************************************************************
C     ALLOCATE STORAGE IN THE X ARRAY FOR PCG ARRAYS
C     ******************************************************************
C
C        SPECIFICATIONS:
C     ------------------------------------------------------------------
C     ------------------------------------------------------------------
C
C-------PRINT A MESSAGE IDENTIFYING PCG PACKAGE
      WRITE(IOUT,1)
    1 FORMAT(1H0,'PCG2 -- CONJUGATE GRADIENT SOLUTION PACKAGE'
     1,', VERSION 2, 5/1/88')
C
C-------READ AND PRINT MXITER,ITER1 AND NPCOND
      READ(IN,2) MXITER,ITER1,NPCOND
    2 FORMAT(3I10)
      WRITE(IOUT,3) MXITER,ITER1,NPCOND
    3 FORMAT(' MAXIMUM OF',I4,' CALLS OF SOLUTION ROUTINE'/
     1       ,' MAXIMUM OF',I4,' INTERNAL ITERATIONS PER '
     2       ,'CALL TO SOLUTION ROUTINE'/
     3       ,' MATRIX PRECONDITIONING TYPE :',I5)
C
C-------ALLOCATE SPACE FOR THE PCG ARRAYS
      ISOLD=ISUM
      NRC=NROW*NCOL
      ISIZ=NRC*NLAY
      LCV=ISUM
      ISUM=ISUM+ISIZ
      LCSS=ISUM
      ISUM=ISUM+ISIZ
      LCP=ISUM
      ISUM=ISUM+ISIZ
      LCCD=ISUM
      IF(NPCOND.NE.2) ISUM=ISUM+ISIZ
      LCHCHG=ISUM
      ISUM=ISUM+MXITER*ITER1
      LCLHCH=ISUM
      ISUM=ISUM+3*MXITER*ITER1
      LCRCHG=ISUM
      ISUM=ISUM+MXITER*ITER1
      LCLRCH=ISUM
      ISUM=ISUM+3*MXITER*ITER1
C
C-------CALCULATE AND PRINT THE SPACE USED IN THE X ARRAY
      ICG=ISUM-ISOLD
      WRITE(IOUT,4) ICG
    4 FORMAT(1X,I7,' ELEMENTS IN X ARRAY ARE USED BY PCG')
      ISUM1=ISUM-1
      WRITE(IOUT,5) ISUM1,LENX
    5 FORMAT(1X,I7,' 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
C-------RETURN
      RETURN
      END
```

```
      SUBROUTINE PCG2RP(MXITER,ITER1,HCLOSE,RCLOSE,NPCOND,NBPOL,
     1                  RELAX,IPRPCG,IN,IOUT,MUTPCG,IPCGCD)
C
C-----VERSION 0002 01MAY1989 PCG2RP
C
C     ******************************************************************
C     READ DATA FOR PCG
C     ******************************************************************
C
C        SPECIFICATIONS:
C     ------------------------------------------------------------------
C     ------------------------------------------------------------------
C
C-------READ HCLOSE,RCLOSE,RELAX,NBPOL,IPRPCG,MUTPCG
      READ(IN,1) HCLOSE,RCLOSE,RELAX,NBPOL,IPRPCG,MUTPCG,IPCGCD
    1 FORMAT(3F10.0,4I10)
C
C-------PRINT MXITER,ITER1,NPCOND,HCLOSE,RCLOSE,RELAX,NBPOL,IPRPCG,
C-------MUTPCG, IPCGCD
      WRITE(IOUT,100)
  100 FORMAT(1H0,///57X,'SOLUTION BY THE CONJUGATE-GRADIENT METHOD'
     1/57X,43('-'))
      WRITE(IOUT,115) MXITER
  115 FORMAT(1H0,38X,'MAXIMUM NUMBER OF CALLS TO PCG ROUTINE =',I9)
      WRITE(IOUT,120) ITER1
  120 FORMAT(1H ,42X,'MAXIMUM ITERATIONS PER CALL TO PCG =',I9)
      WRITE(IOUT,122) NPCOND
  122 FORMAT(1H ,49X,'MATRIX PRECONDITIONING TYPE =',I9)
      IF(NPCOND.EQ.2) WRITE(IOUT,123)
  123 FORMAT(1H ,58X,'THE MATRIX WILL BE SCALED')
      WRITE(IOUT,124) RELAX,NBPOL
  124 FORMAT(1H ,26X,'RELAXATION FACTOR (ONLY USED WITH',
     1' PRECOND. TYPE 1) =',E15.5,/,
     2 1H ,19X,'PARAMETER OF POLYMOMIAL PRECOND.'
     3 ,' = 2 (2) OR IS CALCULATED :',I9)
      WRITE(IOUT,125) HCLOSE
  125 FORMAT(1H ,43X,'HEAD CHANGE CRITERION FOR CLOSURE =',E15.5)
      WRITE(IOUT,127) RCLOSE
  127 FORMAT(1H ,39X,'RESIDUAL CHANGE CRITERION FOR CLOSURE =',E15.5)
      IF(IPRPCG.LE.0)IPRPCG=999
      WRITE(IOUT,130) IPRPCG,MUTPCG
  130 FORMAT(1H ,30X,'PCG HEAD AND RESIDUAL CHANGE PRINTOUT INTERVAL ='
     1,I9,/,1H ,30X,'ALL PRINTING FROM THE SOLVER IS SUPPRESSED (1) ='
     2,I9)
      WRITE(IOUT,135) IPCGCD
  135 FORMAT(1H ,5X,'FOR NPCOND=1, DO (0) OR DO NOT (1) RECALC.',
     1  ' CHOL. DIAG. EACH OUTER ITER. =',I9,/)
C
      RETURN
      END
```

```
      SUBROUTINE PCG2AP(HNEW,IBOUND,CR,CC,CV,HCOF,RHS,V,SS,P,CD,
     1      HCHG,LHCH,RCHG,LRCH,KITER,NITER,HCLOSE,RCLOSE,ICNVG,
     2      KSTP,KPER,IPRPCG,MXITER,ITER1,NPCOND,NBPOL,NSTP,NCOL,NROW,
     3      NLAY,NODES,RELAX,IOUT,MUTPCG,IPCGCD,STEPL,DELT,IU,IP)
C-----VERSION 0002 01MAY1989 PCG2AP
C
C     ******************************************************************
C     SOLUTION BY THE CONJUGATE GRADIENT METHOD -
C                                          UP TO ITER1 ITERATIONS
C     ******************************************************************
C
C        SPECIFICATIONS:
C     ------------------------------------------------------------------
      PARAMETER (DZERO=0.D0,DONE=1.D0)
      DOUBLE PRECISION HNEW,HHCOF,RRHS,RES
      DOUBLE PRECISION Z,B,D,E,F,H,S,ALPHA
      DOUBLE PRECISION ZHNEW,BHNEW,DHNEW,FHNEW,HHNEW,SHNEW,HCHNEW
      DOUBLE PRECISION SRNEW,SROLD,SSCR,SSCC,SSCV,VCC,VCR,VCV
      DOUBLE PRECISION CDCC,CDCR,CDCV,CDN
      DOUBLE PRECISION PN,VN,SSN,HCHGN,RCHGN,PAP
      DOUBLE PRECISION FCC,FCR,FCV,FV
C
      DIMENSION HNEW(NODES), IBOUND(NODES), CR(NODES), CC(NODES),
     1  CV(NODES), HCOF(NODES), RHS(NODES),
     2  V(NODES), SS(NODES), P(NODES), CD(NODES), HCHG(MXITER*ITER1),
     3  LHCH(3,MXITER*ITER1), RCHG(MXITER*ITER1), LRCH(3,MXITER*ITER1)
C     ------------------------------------------------------------------
C
C-------ASSIGN VALUES TO FIELDS THAT ARE CONSTANT DURING AN ITERATION
      NRC=NROW*NCOL
C-------INITIALIZE VARIABLES USED TO CALCULATE ITERATION PARAMETERS
      SRNEW=DZERO
      BPOLY=0.
      IF(NPCOND.NE.1) RELAX=1.
      NORM=0
      IF(NPCOND.EQ.2) NORM=1
C-------INITIALIZE VARIABLE USED TO TEST FOR NEGATIVE CHOLESKY DIAGONAL
      CD1=0.
C------CLEAR PCG WORK ARRAYS.
      DO 100 N=1,NODES
      SS(N)=0.
      P(N)=0.
  100 V(N)=0.
      IF(NPCOND.EQ.1) THEN
        ITYPE=0
        IF(IPCGCD.EQ.1.AND.(IU.EQ.0.OR.IP.GT.0.OR.KPER.GT.1)) THEN
          IF(STEPL.EQ.DELT) ITYPE=1
          STEPL=DELT
        ENDIF
        IF(ITYPE.EQ.0) THEN
          DO 105 N=1,NODES
  105     CD(N)=0.
        ENDIF
      ENDIF
C
```

```
C------CALCULATE THE RESIDUAL. IF NORM=1, CALCULATE THE DIAGONALS OF
C------THE A MATRIX,AND STORE THEM IN HCOF.
      DO 115 K=1,NLAY
      DO 115 I=1,NROW
      DO 115 J=1,NCOL
C
C-------CALCULATE 1 DIMENSIONAL SUBSCRIPT OF CURRENT CELL AND
C-------SKIP CALCULATIONS IF CELL IS INACTIVE
      N=J+(I-1)*NCOL+(K-1)*NRC
      IF(IBOUND(N).EQ.0) THEN
        CC(N)=0.
        CR(N)=0.
        IF(N.LE.(NODES-NRC)) CV(N)=0.
        IF(N.GE.2) CR(N-1)=0.
        IF(N.GE.NCOL+1) CC(N-NCOL)=0.
        IF(N.LE.(NODES-NRC).AND.N.GE.NRC+1) CV(N-NRC)=0.
        HCOF(N)=0.
        RHS(N)=0.
        GO TO 115
      ENDIF
C
C-------CALCULATE 1 DIMENSIONAL SUBSCRIPTS FOR LOCATING THE 6
C-------SURROUNDING CELLS
      NRN=N+NCOL
      NRL=N-NCOL
      NCN=N+1
      NCL=N-1
      NLN=N+NRC
      NLL=N-NRC
C
C-------CALCULATE 1 DIMENSIONAL SUBSCRIPTS FOR CONDUCTANCE TO THE 6
C-------SURROUNDING CELLS.
      NCF=N
      NCD=N-1
      NRB=N-NCOL
      NRH=N
      NLS=N
      NLZ=N-NRC
C
C-----GET CONDUCTANCES TO NEIGHBORING CELLS
C-------NEIGHBOR IS 1 ROW BACK
      B=DZERO
      BHNEW=DZERO
      IF(I.NE.1) THEN
        B=CC(NRB)
        BHNEW=B*(HNEW(NRL)-HNEW(N))
      ENDIF
C
C-------NEIGHBOR IS 1 ROW AHEAD
      H=DZERO
      HHNEW=DZERO
      IF(I.NE.NROW) THEN
        H=CC(NRH)
        HHNEW=H*(HNEW(NRN)-HNEW(N))
      ENDIF
```

```
C
C-------NEIGHBOR IS 1 COLUMN BACK
      D=DZERO
      DHNEW=DZERO
      IF(J.NE.1) THEN
        D=CR(NCD)
        DHNEW=D*(HNEW(NCL)-HNEW(N))
      ENDIF
C
C-------NEIGHBOR IS 1 COLUMN AHEAD
      F=DZERO
      FHNEW=DZERO
      IF(J.NE.NCOL) THEN
        F=CR(NCF)
        FHNEW=F*(HNEW(NCN)-HNEW(N))
      ENDIF
C
C-------NEIGHBOR IS 1 LAYER BEHIND
      Z=DZERO
      ZHNEW=DZERO
      IF(K.NE.1) THEN
        Z=CV(NLZ)
        ZHNEW=Z*(HNEW(NLL)-HNEW(N))
      ENDIF
C
C-------NEIGHBOR IS 1 LAYER AHEAD
      S=DZERO
      SHNEW=DZERO
      IF(K.NE.NLAY) THEN
        S=CV(NLS)
        SHNEW=S*(HNEW(NLN)-HNEW(N))
      ENDIF
C
      IF(I.EQ.NROW) CC(N)=0.
      IF(J.EQ.NCOL) CR(N)=0.
C
C-------CALCULATE THE RESIDUAL AND STORE IT IN RHS.  TO SCALE A,
C-------CALCULATE THE DIAGONAL OF THE A MATRIX, AND STORE IT IN HCOF.
      E=-Z-B-D-F-H-S
      RRHS=RHS(N)
      HHCOF=HNEW(N)*HCOF(N)
      RHS(N)=RRHS-ZHNEW-BHNEW-DHNEW-HHCOF-FHNEW-HHNEW-SHNEW
      IF(NORM.EQ.1) HCOF(N)=HCOF(N)+E
      IF(IBOUND(N).LT.0.) RHS(N)=0.
  115 CONTINUE
C
C-------SCALE CC,CR,CV,RHS AND HNEW IF NORM=1.
      IF(NORM.EQ.1) THEN
        DO 120 K=1,NLAY
        DO 120 I=1,NROW
        DO 120 J=1,NCOL
          N=J+(I-1)*NCOL+(K-1)*NRC
          IF(IBOUND(N).EQ.0) GO TO 120
          HHCOF=SQRT(-HCOF(N))
          IF(N.LE.(NODES-NCOL).AND.CC(N).GT.0.)
```

```
     #       CC(N)=CC(N)/(HHCOF*(SQRT(-HCOF(N+NCOL))))
             IF(CR(N).GT.0.) CR(N)=CR(N)/(HHCOF*(SQRT(-HCOF(N+1))))
             IF(N.LE.(NODES-NRC).AND.CV(N).GT.0.)
     #       CV(N)=CV(N)/(HHCOF*(SQRT(-HCOF(N+NRC))))
             HNEW(N)=HNEW(N)*HHCOF
             RHS(N)=RHS(N)/HHCOF
  120    CONTINUE
         ENDIF
C
C-------CALCULATE PARAMETER B OF THE POLYNOMIAL PRECONDITIONING METHOD
         IF(NPCOND.NE.2) GO TO 152
         IF(NBPOL.EQ.2) THEN
           BPOLY=2
           GO TO 151
         ENDIF
         DO 150 K=1,NLAY
         DO 150 I=1,NROW
         DO 150 J=1,NCOL
C
         N=J+(I-1)*NCOL+(K-1)*NRC
         IF(IBOUND(N).LE.0)GO TO 150
C
         NCF=N
         NCD=N-1
         NRB=N-NCOL
         NRH=N
         NLS=N
         NLZ=N-NRC
C
         B=DZERO
         IF(I.NE.1) B=CC(NRB)
         H=DZERO
         IF(I.NE.NROW) H=CC(NRH)
         D=DZERO
         IF(J.NE.1) D=CR(NCD)
         F=DZERO
         IF(J.NE.NCOL) F=CR(NCF)
         Z=DZERO
         IF(K.NE.1) Z=CV(NLZ)
         S=DZERO
         IF(K.NE.NLAY) S=CV(NLS)
C
C-------NOTE : ABS. VAL. OF THE DIAG. OF THE SCALED A MATRIX IS 1.
         HHCOF=HCOF(N)
         IF(NORM.EQ.1) HHCOF=DONE
         T=DABS(Z)+DABS(B)+DABS(D)+ABS(HHCOF)+DABS(F)+DABS(H)+DABS(S)
         IF(T.GT.BPOLY) BPOLY=T
  150 CONTINUE
  151 CONTINUE
C
C-------CALCULATE ITERATION PARAMETERS FOR POLYNOMIAL PRECONDITIONING
C-------METHOD FOR A NEGATIVE DEFINITE MATRIX.
           C0=(15./32.)*(BPOLY**3)
           C1=(27./16.)*(BPOLY**2)
           C2=(9./4.)*BPOLY
```

```
  152 CONTINUE
C
C-------START INTERNAL ITERATIONS
      IITER=0
      IF(KITER.EQ.1) NITER=0
      ICNVG=0
      IICNVG=0
153   CONTINUE
      IITER=IITER+1
      NITER=NITER+1
C
C-------INITIALIZE VARIABLES THAT TRACK MAXIMUM HEAD CHANGE AND RESIDUAL
C-------VALUE DURING EACH ITERATIONS
      BIGH=0.
      BIGR=0.
C
C
C-------CHECK NPCOND FOR PRECONDITIONING TYPE AND EXECUTE PROPER CODE
      IF(NPCOND.EQ.2) GO TO 165
C
C-------CHOLESKY PRECONDITIONING
C
C-------STEP THROUGH CELLS TO CALCULATE THE DIAGONAL OF THE CHOLESKY
C-------MATRIX (FIRST INTERNAL ITERATION ONLY) AND THE INTERMEDIATE
C-------SOLUTION.  STORE THEM IN CD AND V, RESPECTIVELY.
      DO 155 K=1,NLAY
      DO 155 I=1,NROW
      DO 155 J=1,NCOL
C
      N=J+(I-1)*NCOL+(K-1)*NRC
      IF(IBOUND(N).LE.0)GO TO 155
C
C-------CALCULATE V
      H=DZERO
      VCC=DZERO
      IC=N-NCOL
      IF(I.NE.1) THEN
        H=CC(IC)
        IF(CD(IC).NE.0.) VCC=H*V(IC)/CD(IC)
      ENDIF
C
      F=DZERO
      VCR=DZERO
      IR=N-1
      IF(J.NE.1) THEN
        F=CR(IR)
        IF(CD(IR).NE.0.) VCR=F*V(IR)/CD(IR)
      ENDIF
C
      S=DZERO
      VCV=DZERO
      IL=N-NRC
      IF(K.NE.1) THEN
        S=CV(IL)
        IF(CD(IL).NE.0.) VCV=S*V(IL)/CD(IL)
```

```
      ENDIF
      V(N)=RHS(N)-VCR-VCC-VCV
C
C-------CALCULATE CD - FIRST INTERNAL ITERATION ONLY
      IF(IITER.EQ.1.AND.ITYPE.EQ.0) THEN
          CDCR=DZERO
          CDCC=DZERO
          CDCV=DZERO
          FCC=DZERO
          FCR=DZERO
          FCV=DZERO
          IF(IR.GT.0.AND.CD(IR).NE.0.) CDCR=(F**2)/CD(IR)
          IF(IC.GT.0.AND.CD(IC).NE.0.) CDCC=(H**2)/CD(IC)
          IF(IL.GT.0.AND.CD(IL).NE.0.) CDCV=(S**2)/CD(IL)
          IF(NPCOND.EQ.1) THEN
              IF(IR.GT.0) THEN
                  FV=CV(IR)
                  IF(K.EQ.NLAY.AND.((J+I).GT.1)) FV=DZERO
                  IF(CD(IR).NE.0.) FCR=(F/CD(IR))*(CC(IR)+FV)
              ENDIF
              IF(IC.GT.0) THEN
                  FV=CV(IC)
                  IF(K.EQ.NLAY.AND.(I.GT.1)) FV=DZERO
                  IF(CD(IC).NE.0.) FCC=(H/CD(IC))*(CR(IC)+FV)
              ENDIF
              IF(IL.GT.0) THEN
                  IF(CD(IL).NE.0.) FCV=(S/CD(IL))*(CR(IL)+CC(IL))
              ENDIF
          ENDIF
          IF(NORM.EQ.0) THEN
             B=DZERO
             H=DZERO
             D=DZERO
             F=DZERO
             Z=DZERO
             S=DZERO
             IF(I.NE.1) B=CC(IC)
             IF(I.NE.NROW) H=CC(N)
             IF(J.NE.1) D=CR(IR)
             IF(J.NE.NCOL) F=CR(N)
             IF(K.NE.1) Z=CV(IL)
             IF(K.NE.NLAY) S=CV(N)
             HHCOF=HCOF(N)-Z-B-D-F-H-S
          ENDIF
          IF(NORM.EQ.1) HHCOF=-DONE
          CD(N)=HHCOF-CDCR-CDCC-CDCV-RELAX*(FCR+FCC+FCV)
         IF(CD1.EQ.0..AND.CD(N).NE.0.) CD1=CD(N)
         IF(CD(N)*CD1.LT.0.) THEN
           WRITE(IOUT,510)
  510      FORMAT(//,'  CHOLESKY DIAGONAL LESS THAN ZERO -- '
     1        'EXECUTION TERMINATED (MATRIX NOT DIAGONALLY DOMINANT)')
           STOP
         ENDIF
      ENDIF
C
```

```
  155 CONTINUE
C
C-------STEP THROUGH EACH CELL AND SOLVE FOR S OF THE CONJUGATE
C-------GRADIENT ALGORITHM BY BACK SUBSTITUTION. STORE RESULT IN SS.
      DO 160 KK=NLAY,1,-1
      DO 160 II=NROW,1,-1
      DO 160 JJ=NCOL,1,-1
C
      N=JJ+(II-1)*NCOL+(KK-1)*NRC
      IF(IBOUND(N).LE.0)GO TO 160
C
      NC=N+1
      NR=N+NCOL
      NL=N+NRC
C
C-------BACK SUBSTITUTE, STORING RESULT IN ARRAY SS
      SSCR=DZERO
      SSCC=DZERO
      SSCV=DZERO
      IF(JJ.NE.NCOL) SSCR=CR(N)*SS(NC)/CD(N)
      IF(II.NE.NROW) SSCC=CC(N)*SS(NR)/CD(N)
      IF(KK.NE.NLAY) SSCV=CV(N)*SS(NL)/CD(N)
      VN=V(N)/CD(N)
      SS(N)=VN-SSCR-SSCC-SSCV
  160 CONTINUE
C-------SKIP OVER OTHER PRECONDITIONING TYPES
      GO TO 199
  165 CONTINUE
C
C-------POLYNOMIAL PRECONDITIONING
      DO 170 N=1,NODES
      V(N)=RHS(N)
  170 CONTINUE
      CALL SPCG2E(IBOUND,RHS,HCOF,CR,CC,CV,V,SS,C2,NORM,NCOL,NROW,
     1            NLAY,NODES)
      CALL SPCG2E(IBOUND,RHS,HCOF,CR,CC,CV,SS,V,C1,NORM,NCOL,NROW,
     1            NLAY,NODES)
      CALL SPCG2E(IBOUND,RHS,HCOF,CR,CC,CV,V,SS,C0,NORM,NCOL,NROW,
     1            NLAY,NODES)
  199 CONTINUE
C
C-------CALCULATE P OF THE CONJUGATE GRADIENT ALGORITHM
      SROLD=SRNEW
      SRNEW=DZERO
      DO 200 N=1,NODES
        IF(IBOUND(N).LE.0)GO TO 200
        SRNEW=SRNEW+SS(N)*RHS(N)
  200 CONTINUE
C
      IF(IITER.EQ.1) THEN
         DO 205 N=1,NODES
  205    P(N)=SS(N)
      ELSE
         DO 210 N=1,NODES
  210    P(N)=SS(N)+(SRNEW/SROLD)*P(N)
```

```
      ENDIF
C
C-------CALCULATE ALPHA OF THE CONJUGATE GRADIENT ROUTINE.
C-------FOR THE DENOMINATOR OF ALPHA, MULTIPLY THE MATRIX A BY THE
C-------VECTOR P, AND STORE IN V; THEN MULTIPLY P BY V.  STORE IN PAP.
      PAP=DZERO
      DO 290 K=1,NLAY
      DO 290 I=1,NROW
      DO 290 J=1,NCOL
C
      N=J+(I-1)*NCOL+(K-1)*NRC
      V(N)=0.
      IF(IBOUND(N).LE.0)GO TO 290
C
      NRN=N+NCOL
      NRL=N-NCOL
      NCN=N+1
      NCL=N-1
      NLN=N+NRC
      NLL=N-NRC
C
      NCF=N
      NCD=NCL
      NRB=NRL
      NRH=N
      NLS=N
      NLZ=NLL
C
      B=DZERO
      IF(I.NE.1) B=CC(NRB)
      H=DZERO
      IF(I.NE.NROW)H=CC(NRH)
      D=DZERO
      IF(J.NE.1) D=CR(NCD)
      F=DZERO
      IF(J.NE.NCOL) F=CR(NCF)
      Z=DZERO
      IF(K.NE.1) Z=CV(NLZ)
      S=DZERO
      IF(K.NE.NLAY) S=CV(NLS)
C
      IF(NORM.EQ.0) PN=P(N)
      IF(NORM.EQ.1) PN=DZERO
      BHNEW=DZERO
      HHNEW=DZERO
      DHNEW=DZERO
      FHNEW=DZERO
      ZHNEW=DZERO
      SHNEW=DZERO
      IF(NRL.GT.0) BHNEW=B*(P(NRL)-PN)
      IF(NRN.LE.NODES) HHNEW=H*(P(NRN)-PN)
      IF(NCL.GT.0) DHNEW=D*(P(NCL)-PN)
      IF(NCN.LE.NODES) FHNEW=F*(P(NCN)-PN)
      IF(NLL.GT.0) ZHNEW=Z*(P(NLL)-PN)
      IF(NLN.LE.NODES) SHNEW=S*(P(NLN)-PN)
```

```
C
C-------CALCULATE THE PRODUCT OF MATRIX A AND VECTOR P AND STORE
C-------RESULT IN V.
      PN=HCOF(N)*P(N)
      IF(NORM.EQ.1) PN=-P(N)
      VN=ZHNEW+BHNEW+DHNEW+PN+FHNEW+HHNEW+SHNEW
      V(N)=VN
      PAP=PAP+P(N)*VN
  290 CONTINUE
C
C-------CALCULATE ALPHA
      ALPHA=SRNEW/PAP
C
C-------CALCULATE NEW HEADS AND RESIDUALS, AND SAVE THE LARGEST
C-------CHANGE IN HEAD AND THE LARGEST VALUE OF THE RESIDUAL.
      DO 300 K=1,NLAY
      DO 300 I=1,NROW
      DO 300 J=1,NCOL
C
      N=J+(I-1)*NCOL+(K-1)*NRC
      IF(IBOUND(N).LE.0) GO TO 300
C
C-------HEAD
      HCHGN=ALPHA*P(N)
      IF(DABS(HCHGN).GT.ABS(BIGH)) THEN
          BIGH=HCHGN
          IH=I
          JH=J
          KH=K
          NH=N
      ENDIF
      HNEW(N)=HNEW(N)+HCHGN
C
C--------RESIDUAL (V IS THE PRODUCT OF MATRIX A AND VECTOR P)
      RCHGN=-ALPHA*V(N)
      RHS(N)=RHS(N)+RCHGN
      IF(ABS(RHS(N)).GT.ABS(BIGR)) THEN
          BIGR=RHS(N)
          IR=I
          JR=J
          KR=K
          NR=N
      ENDIF
  300 CONTINUE
C
C-------UNSCALE LARGEST CHANGE IN HEAD AND LARGEST RESIDUAL, AND
C-------CHECK THE CONVERGENCE CRITERION
       IF(NORM.EQ.1) THEN
          BIGH=BIGH/SQRT(-HCOF(NH))
          BIGR=BIGR*SQRT(-HCOF(NR))
       ENDIF
       IF(MXITER.EQ.1) THEN
          IF(ABS(BIGH).LE.HCLOSE.AND.ABS(BIGR).LE.RCLOSE) ICNVG=1
       ELSE
          IF(IITER.EQ.1.AND.
```

```
     1       ABS(BIGH).LE.HCLOSE.AND.ABS(BIGR).LE.RCLOSE) ICNVG=1
        ENDIF
        IF(ABS(BIGH).LE.HCLOSE.AND.ABS(BIGR).LE.RCLOSE) IICNVG=1
C
C-------STORE THE LARGEST UNSCALED HEAD CHANGE AND RESIDUAL VALUE
C-------(THIS ITERATION) AND THEIR LOCATIONS.
       II=NITER
       HCHG(II)=BIGH
       LHCH(1,II)=KH
       LHCH(2,II)=IH
       LHCH(3,II)=JH
C
       RCHG(II)=BIGR
       LRCH(1,II)=KR
       LRCH(2,II)=IR
       LRCH(3,II)=JR
C
C-------GO TO NEXT INTERNAL ITERATION IF CONVERGENCE HAS NOT BEEN
C-------REACHED AND IITER IS LESS THAN ITER1
       IF(MXITER.EQ.1) THEN
         IF(ICNVG.EQ.0.AND.IITER.LT.ITER1) GO TO 153
       ELSE
         IF(IICNVG.EQ.0.AND.IITER.LT.ITER1) GO TO 153
       ENDIF
C
C-------UNSCALE CR,CC,CV AND HNEW
       IF(NORM.EQ.1) THEN
       DO 310 N=1,NODES
       IF(IBOUND(N).EQ.0) GO TO 310
       HHCOF=SQRT(-HCOF(N))
       IF(N.LE.(NODES-NCOL).AND.CC(N).GT.0.)
      #   CC(N)=CC(N)*(HHCOF*(SQRT(-HCOF(N+NCOL))))
       IF(N.LE.(NODES-1).AND.CR(N).GT.0.)
      #   CR(N)=CR(N)*(HHCOF*(SQRT(-HCOF(N+1))))
       IF(N.LE.(NODES-NRC).AND.CV(N).GT.0.)
      #   CV(N)=CV(N)*(HHCOF*(SQRT(-HCOF(N+NRC))))
       HNEW(N)=HNEW(N)/HHCOF
  310 CONTINUE
       ENDIF
C
C-------IF END OF TIME STEP, PRINT # OF ITERATIONS THIS STEP
       IF(ICNVG.EQ.0 .AND. KITER.NE.MXITER) GO TO 600
       IF(MUTPCG.GT.1) GO TO 600
       IF(KSTP.EQ.1) WRITE(IOUT,500)
  500 FORMAT(1H0)
       WRITE(IOUT,501) KITER,KSTP,KPER,NITER
  501 FORMAT(1X,I5,' CALLS TO PCG ROUTINE FOR TIME STEP',I4,
     1' IN STRESS PERIOD',I3,/1X,I5,' TOTAL ITERATIONS')
       IF(MUTPCG.EQ.1) GO TO 600
C
C-------PRINT HEAD CHANGE EACH ITERATION IF PRINTOUT INTERVAL IS REACHED
       IF(ICNVG.EQ.0 .OR. KSTP.EQ.NSTP .OR. MOD(KSTP,IPRPCG).EQ.0)
     1      CALL SPCG2P(HCHG,LHCH,RCHG,LRCH,IITER,KITER,KSTP,KPER,
     2                  ITER1,NITER,MXITER,IOUT,NPCOND,BPOLY)
C
```

```
C--------RETURN
600    RETURN
C
       END
```

```
      SUBROUTINE SPCG2P(HCHG,LHCH,RCHG,LRCH,IITER,KITER,KSTP,KPER,ITER1,
     1                  NITER,MXITER,IOUT,NPCOND,BPOLY)
C
C
C-----VERSION 0001 01MAY1988 SPCG2P
C     ******************************************************************
C     PRINT MAXIMUM HEAD CHANGE AND RESIDUAL VALUE FOR EACH ITERATION
C                           DURING A TIME STEP
C     ******************************************************************
C
C        SPECIFICATIONS:
C     ------------------------------------------------------------------
C
      DIMENSION HCHG(MXITER*ITER1), LHCH(3,MXITER*ITER1)
      DIMENSION RCHG(MXITER*ITER1), LRCH(3,MXITER*ITER1)
C     ------------------------------------------------------------------
C
      IF(NPCOND.EQ.2) WRITE(IOUT,2) BPOLY
    2 FORMAT(1H0,'B OF THE POLYNOMIAL PRECONDITIONING METHOD: ',E12.4)
      WRITE(IOUT,5)
    5 FORMAT(1H0,'MAXIMUM HEAD CHANGE FOR EACH ITERATION:'/
     1    1H0,4('    HEAD CHANGE  LAYER,ROW,COL')/1X,120('-'))
      WRITE (IOUT,10) (HCHG(J),(LHCH(I,J),I=1,3),J=1,NITER)
      WRITE(IOUT,11)
      WRITE(IOUT,15)
   15 FORMAT(1H0,'MAXIMUM RESIDUAL FOR EACH ITERATION:'/
     1    1H0,4('     RESIDUAL    LAYER,ROW,COL')/1X,120('-'))
      WRITE (IOUT,10) (RCHG(J),(LRCH(I,J),I=1,3),J=1,NITER)
      WRITE(IOUT,11)
C
      RETURN
C
   10 FORMAT((1X,4(4X,G12.4,' (',I3,',',I3,',',I3,')')))
   11 FORMAT(1H0)
C
      END
```

```
      SUBROUTINE SPCG2E(IBOUND,RHS,HCOF,CR,CC,CV,VIN,VOUT,C,NORM,NCOL,
     1                  NROW,NLAY,NODES)
C
C
C-----VERSION 0001 01MAY1989 SPCG2E
C     ******************************************************************
C           MATRIX MULTIPLICATIONS FOR POLYNOMIAL PRECONDITIONING
C     ******************************************************************
C
C        SPECIFICATIONS:
C     ------------------------------------------------------------------
C
      DOUBLE PRECISION VN,CRHS,Z,B,D,F,H,S,ZV,BV,DV,FV,HV,SV,DZERO
      DIMENSION IBOUND(NODES),CR(NODES),CC(NODES),CV(NODES),
     1  RHS(NODES),VIN(NODES),VOUT(NODES),HCOF(NODES)
C     ------------------------------------------------------------------
C
      DZERO=0.
      NRC=NROW*NCOL
      DO 290 K=1,NLAY
      DO 290 I=1,NROW
      DO 290 J=1,NCOL
C
      N=J+(I-1)*NCOL+(K-1)*NRC
      VOUT(N)=0.
      IF(IBOUND(N).LE.0)GO TO 290
C
      NRN=N+NCOL
      NRL=N-NCOL
      NCN=N+1
      NCL=N-1
      NLN=N+NRC
      NLL=N-NRC
C
      NCF=N
      NCD=NCL
      NRB=NRL
      NRH=N
      NLS=N
      NLZ=NLL
C
      B=DZERO
      BV=DZERO
      IF(I.NE.1.AND.IBOUND(NRL).GE.0) THEN
        B=CC(NRB)
        BV=B*VIN(NRL)
      ENDIF
      H=DZERO
      HV=DZERO
      IF(I.NE.NROW.AND.IBOUND(NRN).GE.0) THEN
        H=CC(NRH)
        HV=H*VIN(NRN)
      ENDIF
      D=DZERO
      DV=DZERO
```

```
      IF(J.NE.1.AND.IBOUND(NCL).GE.0) THEN
        D=CR(NCD)
        DV=D*VIN(NCL)
      ENDIF
      F=DZERO
      FV=DZERO
      IF(J.NE.NCOL.AND.IBOUND(NCN).GE.0) THEN
        F=CR(NCF)
        FV=F*VIN(NCN)
      ENDIF
      Z=DZERO
      ZV=DZERO
      IF(K.NE.1.AND.IBOUND(NLL).GE.0) THEN
        Z=CV(NLZ)
        ZV=Z*VIN(NLL)
      ENDIF
      S=DZERO
      SV=DZERO
      IF(K.NE.NLAY.AND.IBOUND(NLN).GE.0) THEN
        S=CV(NLS)
        SV=S*VIN(NLN)
      ENDIF
C
C-------CALCULATE THE PRODUCT OF MATRIX A AND VECTOR VIN AND STORE
C------ RESULT IN VOUT
      VN=HCOF(N)*VIN(N)
      IF(NORM.EQ.1) VN=-VIN(N)
      CRHS=C*RHS(N)
      VOUT(N)=CRHS+ZV+BV+DV+VN+FV+HV+SV
  290 CONTINUE
      RETURN
      END
```


*U.S. GOVERNMENT PRINTING OFFICE: 1990-0-773-204/20022