

PRECONDITIONED CONJUGATE-GRADIENT 2 (PCG2),

A COMPUTER PROGRAM FOR SOLVING

GROUND-WATER FLOW EQUATIONS

By Mary C. Hill

U. S. GEOLOGICAL SURVEY

Denver, Colorado
1990

DISCLAIMER

This is a scanned reproduction of the original document for distribution purposes via electronic format. Effort has been made to provide an accurate and correct document. The document is supplied "as-is" without guarantee or warranty, expressed or implied.

To reduce the file size of this document, listings of source code, which are available on some original open-file reports, are not included.

Wen-Hsing Chiang
Hamburg, April 1998.

CONTENTS

	Page
Abstract-----	1
Introduction-----	1
Purpose and scope-----	1
Previous investigations-----	2
Three-dimensional ground-water flow model-----	3
Solution by the preconditioned conjugate-gradient method-----	4
MICCG-----	8
POLCG-----	11
Convergence criteria-----	12
Input instructions-----	13
Sample data inputs-----	14
Linking PCG2 to the modular model-----	15
Documentation of PCG2-----	16
Brief description of modules-----	16
Flowchart-----	16
Narratives for modules-----	19
PCG2AL-----	19
PCG2RP-----	19
PCG2AP-----	19
SPCG2P-----	19
SPCG2E-----	19
Adapting SPCG2E for computers with vector and parallel architecture-----	20
List of variables-----	21
References-----	24
Fortran listing-----	26
PCG2AL-----	27
PCG2RP-----	29
PCG2AP-----	30
SPCG2P-----	41
SPCG2E-----	42

FIGURES

	Page
Figure 1. Diagram showing aquifer system volumes accounted for by conductances r_n , c_n , and v_n in the finite-difference method-----	5
2. Matrix \underline{U} for MICCG-----	9
3. Matrix $\underline{M} = \underline{U}^T \underline{D} \underline{U}$ for MICCG-----	10

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 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, non-linear 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: A

Underlined lower-case letters indicate column vectors: r

The element located in matrix row i and column j is designated as follows: matrix 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:

$$\underline{A} \underline{x} = \underline{b} \quad (1)$$

where A is a coefficient matrix which is discussed below, x is a vector of hydraulic heads at each grid cell, and 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:

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

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).

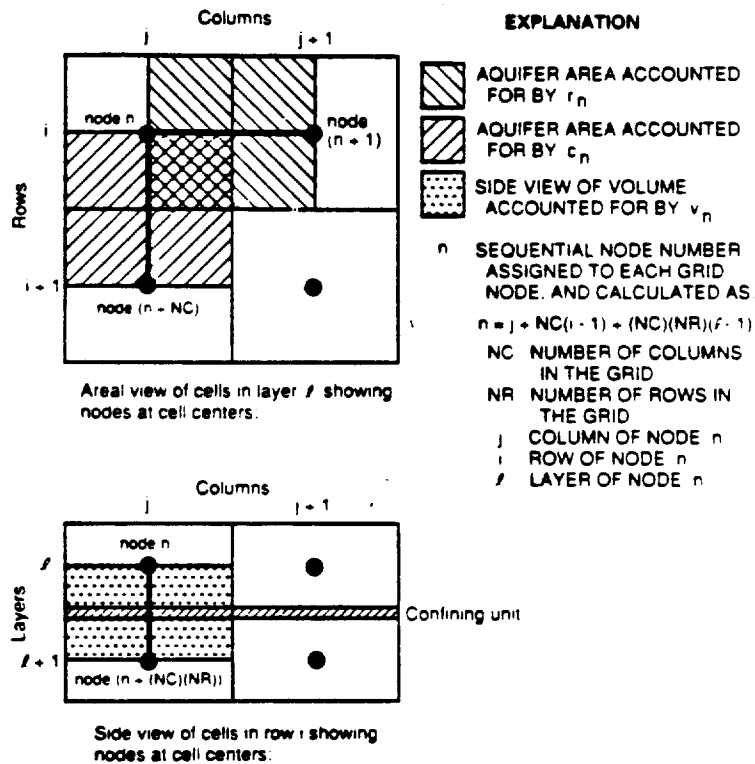


Figure 1.--Aquifer system volumes accounted for by conductances r_n , c_n , and v_n in the finite-difference method.

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:

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

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:

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

or

$$\underline{s}_k = \underline{M}^{-1} \underline{r}_k \quad (5)$$

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 $\underline{r}_0 = \underline{b} - \underline{A} \underline{x}_0$. The following steps are executed for each iteration, starting with $k = 0$:

$$\underline{s}_k = \underline{M}^{-1} \underline{r}_k \quad (6a)$$

$$\text{for } k = 0 \quad \underline{p}_k = \underline{s}_k \quad (6b)$$

$$\text{for } k > 0 \quad \left\{ \begin{array}{l} \beta_k = \frac{\underline{s}_k^T \underline{r}_k}{\underline{s}_{k-1}^T \underline{r}_{k-1}} \end{array} \right. \quad (6c)$$

$$\underline{p}_k = \underline{s}_k + \beta_k \underline{p}_{k-1} \quad (6d)$$

$$\alpha_k = \frac{\underline{s}_k^T \underline{r}_k}{\underline{p}_k^T \underline{A} \underline{p}_k} \quad (6e)$$

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

$$\underline{r}_{k+1} = \underline{r}_k - \alpha_k \underline{A} \underline{p}_k \quad (6g)$$

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:

$$\underline{B} = \underline{S}^T \underline{A} \underline{S} , \quad (7)$$

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

$$s_{ii} = \frac{1}{\sqrt{a_{ii}}} \quad (8)$$

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:

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

\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:

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

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_{ij} 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:

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

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.

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

etc.

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

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

where,

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

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

$$\underline{M}^{-1} = \underline{I} + \underline{A} + \underline{A}^2 + \dots + \underline{A}^l . \quad (12)$$

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

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

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:

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

so that the powers of \underline{A} are never formed explicitly (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 \underline{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 ft³/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 I1C

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	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	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,IPRPGC,IUNIT(13),IOUT,MUTPCG,IPCGCD)

C   ADD BETWEEN COMMENT STATEMENTS 7C2B AND 7C2C
      ICD=0
      IF(IUNIT(13).GT.0) CALL PCG2AP(X(LCHNEW),X(ICD=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,IPRPGC,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

