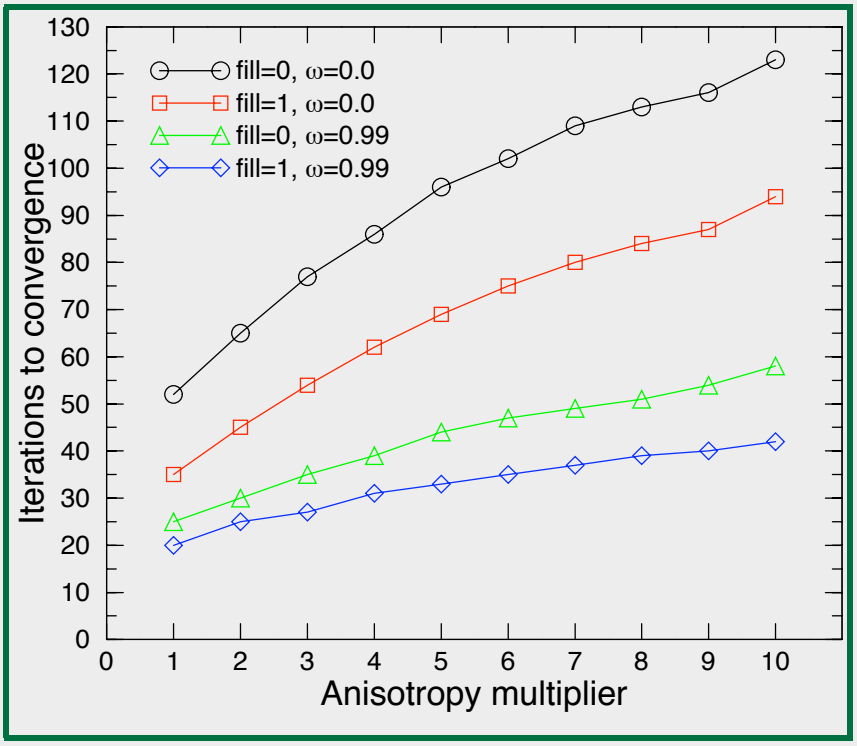
USGS
science for a changing world

The U.S. Geological Survey Modular Ground-Water Model—PCGN: A Preconditioned Conjugate Gradient Solver with Improved Nonlinear Control



Open-File Report 2008–1331

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

Cover. Convergence properties of incomplete Cholesky preconditioners.

The U.S. Geological Survey Modular Ground-Water Model—PCGN: A Preconditioned Conjugate Gradient Solver with Improved Nonlinear Control

By Richard L. Naff and Edward R. Banta

Open-File Report 2008–1331

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

U.S. Department of the Interior
DIRK KEMPTHORNE, Secretary

U.S. Geological Survey
Mark D. Myers, Director

U.S. Geological Survey, Reston, Virginia: 2008

For more information about the USGS and its products:
Telephone: 1-888-ASK-USGS
World Wide Web: http://www.usgs.gov/

Suggested citation:
Naff, R.L., and Banta, E.R., 2008, The U.S. Geological Survey modular ground-water model—PCGN: A preconditioned conjugate gradient solver with improved nonlinear control: U.S. Geological Survey Open-File Report 2008–1331, 35 p.

Typeset on November 12, 2008

Preface

This report describes the preconditioned conjugate gradient with improved nonlinear control (PCGN) package to solve the linear and nonlinear equations associated with the U.S. Geological Survey (USGS) MODFLOW-2000 (MF2K) computer program. The linear solver in the PCGN package is based on a method known in the literature as Preconditioned Conjugate Gradient; preconditioning is based in the modified incomplete Cholesky scheme. The implementation herein is coupled with Picard iteration so as to efficiently solve the nonlinear equations associated with many ground-water flow problems. The nonlinear solution is controlled principally by two parameters: the convergence parameter for the linear solver and the damping factor which limits the updating of the nonlinear head solution in the Picard iteration. The usual procedure with Picard iteration to solve a nonlinear problem is to hold these parameters constant. The PCGN package provides the user with several mechanisms by which these parameters are allowed to vary through the course of a simulation. The mechanism by which these changes are instituted is generally performance based; if the Picard iteration is not progressing satisfactorily toward a solution for the nonlinear equations, then the parameters are modified. Many of these procedures are *ad hoc* in nature and may not withstand the test of usage, eventually requiring revision in some future version of this package. The modified incomplete Cholesky preconditioning, as realized herein, allows for two levels of fill and uses a relaxation parameter to institute the modification to the pivots of the incomplete Cholesky decomposition. Because algorithms for a level 1 fill are infrequently encountered in the literature, a derivation of the modified incomplete Cholesky preconditioning, as instituted in this package, is detailed in this document.

Contents

Figures

Tables

Algorithms

Conversion Factors

Multiply	By	To obtain
foot (ft)	0.3048	meter (m)

Blank page

The U.S. Geological Survey Modular Ground-Water Model—PCGN: A Preconditioned Conjugate Gradient Solver with Improved Nonlinear Control

By Richard L. Naff and Edward R. Banta

Abstract

The preconditioned conjugate gradient with improved nonlinear control (PCGN) package provides additional means by which the solution of nonlinear ground-water flow problems can be controlled as compared to existing solver packages for MODFLOW. Picard iteration is used to solve nonlinear ground-water flow equations by iteratively solving a linear approximation of the nonlinear equations. The linear solution is provided by means of the preconditioned conjugate gradient algorithm where preconditioning is provided by the modified incomplete Cholesky algorithm. The incomplete Cholesky scheme incorporates two levels of fill, 0 and 1, in which the pivots can be modified so that the row sums of the preconditioning matrix and the original matrix are approximately equal. A relaxation factor is used to implement the modified pivots, which determines the degree of modification allowed. The effects of fill level and degree of pivot modification are briefly explored by means of a synthetic, heterogeneous finite-difference matrix; results are reported in the final section of this report. The preconditioned conjugate gradient method is coupled with Picard iteration so as to efficiently solve the nonlinear equations associated with many ground-water flow problems. The description of this coupling of the linear solver with Picard iteration is a primary concern of this document.

Introduction

The preconditioned conjugate gradient solver with improved nonlinear control (PCGN) is a new solver package for the MODFLOW ground-water flow model (Harbaugh and others, 2000). The principal objective of the PCGN package is to provide the modeler with more options when faced with a poorly converging nonlinear problem. In MODFLOW, nonlinear problems are solved by iteratively solving a linearized approximation of the problem. Because MODFLOW uses a cell-centered finite-difference (CCFD) approximation of the ground-water flow equations, the linear approximation consists of a system of equations represented by a sparse, regular matrix. The linear equation solver in the PCGN package is based in the preconditioned conjugate gradient (PCG) algorithm; preconditioning is provided by means of the the incomplete Cholesky algorithm with two fill-level options: 0 and 1. A complete Cholesky decomposition would produce factors that would almost completely fill the upper or lower triangle of the CCFD matrix; as a consequence, complete decompositions are seldom used for large problems. For a fill level of 0, the incomplete Cholesky algorithm decomposition does not allow for factors outside the existing nonzero entries in the CCFD matrix; this fill level

is generally accepted as the traditional incomplete Cholesky preconditioning (van der Vorst, 2003). The fill level 1 is associated with the next higher-order factorization; additional Cholesky factors are formed and stored for this fill level. An incomplete Cholesky algorithm incorporating this fill level should provide the user with a smoother PCG solution for a given level of convergence. This additional capacity comes at the expense of approximately doubling the computer memory requirements of the PCG solver.

The iterative procedure used in MODFLOW for solving nonlinear problems is commonly referred to as Picard iteration; a review of this technique, as it applies to ground-water flow modeling, is to be found in Mehl (2006). Within the PCGN package, the principal controls on the Picard iteration are the convergence parameter for obtaining a head-change solution from the PCG solver and the damping factor for updating the nonlinear solution. The accuracy of the updated head change is determined by the convergence parameter, while the damping factor dictates the proportion of the updated head change to be added to the nonlinear solution. The PCGN package gives the modeler some additional tools to manipulate these parameters within the context of the Picard iteration. In particular, both the damping factor and convergence parameter can be made adaptive in that their values are made to depend on the progress of the nonlinear iteration. In addition, when adaptive damping is elected, it is also feasible to limit the maximum head change applied in any given Picard iteration. Limiting the maximum head is useful when modeling dewatering scenarios as the linear approximation can produce very large and abrupt head changes. Finally, options also exist to institute smaller values for the damping factor and convergence parameter initially in each new stress period, but relax those values with favorable progress in the nonlinear iteration.

There are two other PCG solver packages currently (2008) available to MODFLOW users: PCG2 (Hill, 1990) and GMG (Wilson and Naff, 2004). The principal difference between the solvers in these packages is in the use of preconditioners: PCG2 offers the user a choice of an incomplete Cholesky preconditioning with 0 fill or polynomial preconditioning, and GMG offers a choice of incomplete Cholesky preconditioning with 0 fill or geometric multigrid preconditioning. As noted previously, the PCGN package offers a choice of an incomplete Cholesky preconditioning with either 0 or 1 fill. All three solver packages use Picard iteration to solve nonlinear problems; in most cases, all three function reasonably well in this task. However, for those cases where convergence is not easily obtained, PCGN offers an additional suite of tools that may be useful in obtaining convergence for a recalcitrant nonlinear problem. In particular, PCGN has been found to be useful in solving nonlinear problems simulating aquifer dewatering; with this type of nonlinearity, achieving convergence often can be problematic. The linear solver in the PCGN package does tend to be faster than PCG2, as the incomplete Cholesky preconditioner is built around one dimensional loops that can be unrolled easily, thus promoting better use of a computer processor's cache. In this version of PCGN, certain oft-used loops have been manually unrolled in an effort to diminish the impact of compiler differences on code execution time. The GMG package, with geometric multigrid preconditioning, is likely to be faster than PCGN in solving a linear problem. By providing a smooth approximation to the solution, geometric multigrid preconditioning is simply superior to incomplete Cholesky. However, for solving nonlinear problems, the superiority of GMG over PCGN is less evident, in part because of the nature of the Picard iteration instituted in PCGN. The approximate solves of the linear approximation for the nonlinear problem can be relatively swift in PCGN, thus giving it an advantage over the GMG package.

In the process of forming the incomplete Cholesky factorization, it is sometimes possible to produce zero pivots; zero pivots are indicative of a singular matrix. Should this happen in PCGN, the solver is immediately halted and a warning message is issued. In a nonlinear simulation, the production of a zero pivot is usually associated with a domain that has separated into two or more partitions. This separation is usually precipitated by a dewatering simulation in which active cells are transformed into inactive cells. Because self-partitioning of the domain is fairly common in dewatering simulations, a means of testing the integrity of the domain

has been devised. Should a zero pivot be encountered, the PCGN package analyzes the domain for multiple partitions; should they exist, the approximate grid locations of the smaller partition or partitions are reported back to the MODFLOW listing file (either the List or Global files) before execution is terminated. This feature is automatically enabled by the solver and will not be discussed further; the remainder of this document is devoted to describing features of the Picard iteration and aspects of the PCG solver.

A description of the Picard iteration contained in the PCGN package is detailed in the next section of this report. The subsequent section, *Input Instructions for the PCGN Solver*, contains a descriptive list of the input parameters that control the PCGN package, while the following section, *Application of PCGN to a Nonlinear Problem*, is an example of an application of PCGN to a nonlinear problem where simulation of dewatering is a major concern. This example application is followed by the *Output Diagnostics for the Picard Iteration* section, containing a descriptive list of useful diagnostics that are, upon request, output by the PCGN package. The subsequent sections constitute a lengthy description of the PCG method and incomplete Cholesky preconditioning as applied in the PCGN package; these sections are not particularly useful to the end user but are included as documentation of the code.

Description of Picard Iteration Scheme

Whenever a parameter within the flow model is dependent on the hydraulic head, then the model and the resulting simulation is considered to be nonlinear. For example, if the transmission of water is dependent on the saturated thickness and the saturated thickness is dependent on the hydraulic head, then the resulting simulation is nonlinear. The PCGN package effects the solution of nonlinear ground-water flow problems by Picard iteration (Wilson and Naff, 2004; Mehl, 2006). Picard iteration solves a nonlinear flow problem very simply by alternately solving the linear matrix equations and then updating the head-dependent model parameters with the new approximation of the hydraulic heads. The linear matrix equations can be represented as $Ax = b$, where A is a coefficient matrix, x is the vector of unknowns, and b is the right-hand-side (RHS) vector. The unknown vector x can represent either the hydraulic heads or the changes in hydraulic head. This linear matrix equation results from the CCFD formulation, which is the basis of the numerical approximation in MODFLOW. Nonlinear parameters can be found in either the CCFD coefficient matrix A or the RHS vector b or both. Iteration between solving the linear equations and updating head-dependent model parameters continues until the head solution produces a set of model parameters that require no further updating, at which point the head solution stabilizes. In the PCGN package, the linear solve of the matrix equations is carried out by means of the preconditioned conjugate gradient (PCG) method using modified incomplete Cholesky (MIC) preconditioning; these concepts are explained in the *Preconditioned Conjugate Gradient Method* and *Incomplete Cholesky Preconditioning* sections. As the PCG method is an iterative solution technique, it is frequently considered that iteration associated with the PCG method is the "inner" iteration, while the Picard iteration is the "outer" iteration. An objective of the PCGN package is, for a nonlinear problem, to solve the linear problem only to the accuracy necessary to advance the Picard iteration. That is, as any given exact solution of the linear equations is only an approximation of the solution of the nonlinear problem, then updating the Picard iteration with a very accurate solution to the linear equations is not particularly beneficial. Rather, if the PCG solver can be made to solve the linear equations to a sufficient accuracy so as to advance the Picard iteration, then it is expected that efficiencies will be gained without sacrificing an accurate and stable nonlinear solution. In the PCGN package, the Picard iteration is tied to the linear PCG solver by two variables: the convergence parameter for the PCG solver, which dictates the accuracy of the solution for the linear equations, and the damping factor, which determines how much of the latest hydraulic head update should be applied to the new solution.

In addition to efficiency of the Picard iteration, one must also be concerned with the accuracy of the non-

linear solution. This is particularly true when attempting to solve the steady-state flow equations under dewatering conditions. Under these conditions, solutions to the nonlinear problem may exhibit variable dewatering solely as a function of the parameter selection for the Picard iteration. When this occurs, the modeler frequently attempts to minimize the degree of dewatering, as measured by the number of dry cells, with the assumption that this Picard solution is the most accurate representation of the steady-state water-table configuration. The degree of dewatering has been observed to be dependent on the accuracy of the solution to the linear equations and the amount of damping applied. In a given Picard iteration, a more accurate solution to the linear equations assures that the head changes are accurately approximated, while the damping factor assures that large head changes, even accurately calculated, are modulated when used to update the hydraulic heads. A desirable solution is frequently a compromise between efficiency of the Picard iteration and minimizing the total number of dry cells.

Variables and parameters described in the following two subsection are more fully described in the *Input Instructions for the PCGN Solver* section of this report; they are introduced here in the context of a general overview of Picard iteration as employed by the PCGN package.

Limiting the Inner Iteration

A "standard" convergence scheme is built into the PCGN package; the standard scheme is accessed by setting the PCGN variable `ACNVG` $= 0$. Under this scheme, the PCG linear equation solver is required to iterate until an order of magnitude improvement is obtained. In general, a weighted residual norm is used as a measure of the error for the linear solution. Allow $r = b - A\tilde{x}_i$ to be a vector of residuals, where $\tilde{x}$ is an approximation of the true solution x; the weighted residual norm is defined as $\nu = r^T M^{-1} r$, where M represents the preconditioning matrix. (Here and elsewhere the symbol T used as a superscript represents the matrix transpose operator.) On entry, this norm is denoted as ν_0, and at some later PCG iteration i, it is denoted as ν_i. With the exception of the initial Picard iteration for the first time step of the initial stress period, the PCGN package allows the PCG solver to iterate until $\nu_i/\nu_0 < \varepsilon$, at which point program execution leaves the linear solver. The parameter ε is referred to as the relative convergence criterion; except for the initial Picard iteration for the first time step of a new stress period, the relative convergence parameter ε is set to ε_s, the standard relative convergence criterion, the value for which is set internally: $\varepsilon_s = 0.1$. Certain instances are taken as exceptions to this convergence rule in the standard scheme. Firstly, the initial PCG iteration of the first Picard iteration in the first stress period generally provides a very rough estimate of the residuals r; the initial guess for the hydraulic head is commonly taken to be a constant over the domain. Thus, for the case where MODFLOW parameters `KSTP` $= 1$, `KITER` $= 1$, and `KPER` $= 1$, the absolute convergence criterion $\nu_i < 10$ is employed as a stopping criterion for the inner iteration rather than the relative convergence criterion; the choice of this value is somewhat arbitrary, other than it appears to have performed successfully in a large number of test cases. Secondly, in the event of a stress-period change (MODFLOW parameters `KSTP` $= 1$, `KITER` $= 1$ and `KPER` > 1), then the relative convergence parameter is reset for the first Picard iteration: $\varepsilon = \varepsilon_s^2$. Because stress-period changes are points where new stresses can be introduced, large head changes are to be expected. Thus, the initial PCG solution for a new stress period should reflect a possible new direction in the nonlinear solution. Increasing the accuracy of the linear solution should ensure that any new direction in the Picard iteration is captured.

The standard convergence scheme can be modified in two ways; these modifications are available when the PCGN variable `ACNVG` $= 2$. Firstly, the relative convergence can be modified to require a tighter relative convergence in all cases; setting the PCGN variable `MCNVG` > 1 causes the relative convergence ε to be modified as $\varepsilon = \varepsilon_s^p$, where $p =$ `MCNVG`. Provided PCGN variable `RATE_C` ≤ 0, the relative convergence parameter ε is maintained constant throughout the Picard iteration; otherwise it is variable as described later in this section. This modification generally causes the total number of PCG iterations to increase. However, by other mea-

sures, it may produce a superior Picard iteration solution to the nonlinear problem. The second modification is available by setting the PCGN variable RATE_C > 0. The value for the relative convergence parameter obtained when MCNVG > 1 is now treated as an initial value: $\varepsilon_0 = \varepsilon_s^p$, $p =$ MCNVG. This initial value is depreciated away with each Picard iteration j such that $\varepsilon_j = \varepsilon_{j-1} +$ RATE_C ε_{j-1}, where $0 <$ RATE_C < 1. This reverse depreciation of the relative convergence parameter continues so long as the Picard solution improves; when ε_j becomes large such that $\varepsilon_j \geq \varepsilon_s$, then the relative convergence parameter is set equal to ε_s. The improvement in the Picard iteration is measured by the norm $n_j = r_j^T r_j$, where r_j is a vector of residuals obtained using updated hydraulic heads from the previous Picard iteration. If $n_j < n_{j-1}$, then the reverse depreciation of ε is allowed. For stress periods subsequent to the initial stress period, the starting relative convergence parameter ε_0 is increased slightly by allowing $\varepsilon_0 = \varepsilon_s^{(p+1)/2}$ ($p =$ MCNVG). The logic behind this scheme is that it allows the modeler to place greater emphasis on obtaining a relatively good solution to the linear equations early in the stress period when changes in the linear solution from iteration to iteration are likely to be the most extreme. When solving the steady-state flow equations under dewatering conditions, using this procedure can produce solutions comparable to those obtained with a constant $\varepsilon = \varepsilon_s^p$ at less computation expense.

In addition to the standard convergence scheme and its modifications, PCGN supports an adaptive convergence scheme for adjusting the relative convergence parameter ε; this scheme is based on a measure of the nonlinearity of a problem and is described in detail in the *PCG with Adaptive Convergence* subsection. When the problem departs significantly from linear behavior, the relative convergence ε is decreased, thus producing more accurate solutions to the linear matrix equations by which the hydraulic heads are updated. The assumption here is that a more accurate solve to the linear equations, when the problem becomes more nonlinear, is necessary to keep the Picard iteration on track. This option is available when PCGN variable ACNVG $= 1$. When this option is activated, the PCGN variable CNVG_LB is used to place a lower bound on the adaptive convergence parameter so as to prevent excessive number of inner iterations; typical values are $0 <$ CNVG_LB $< \varepsilon_s$. Also, when through adaptive convergence $\varepsilon > \varepsilon_s$, then ε is set to ε_s. For the initial stress period, the adaptive convergence scheme also requires that the first solution of the linear equations be subject to the standard absolute convergence criterion: $v_i < 10$.

Adjusting the Damping Parameter

In the PCGN package, Picard iteration takes the form $h_j = h_{j-1} + \theta\Delta_j$, where h_j is a vector of hydraulic heads for Picard iteration j, Δ_j is the head-change vector and θ is the damping parameter. The damping parameter θ is usually taken constant, but the PCGN package incorporates two variants where θ can vary with Picard iteration. These variants are made available when PCGN variable ADAMP > 0; otherwise θ is constant. When ADAMP $= 1$, a somewhat complex algorithm is used to adjust θ as a function of the previous convergence history of the Picard iteration; this scheme is presented in algorithm 1. The output from this algorithm is θ_j, the damping parameter to be used with the jth Picard update of the hydraulic heads. Similar to the adaptive damping scheme suggested by Mehl and Hill (2001), this algorithm uses the previous convergence history to adjust the current damping parameter, but the mechanism by which the adjustment is effected is different. In adddition, the value given θ_j is the geometric mean of ϕ, the surrogate for θ_j that receives the adjustment, and θ_{j-1}, the previous damping factor: $\theta_j = \sqrt{\phi\theta_{j-1}}$. Using this geometric mean for value of θ_j inhibits abrupt changes in the damping factor from one Picard iteration to the next. With regard to algorithm 1, the norm n_j is defined as $n_j = \sqrt{r_j^T r_j \Delta_j^T \Delta_j}$, where r_j is the vector of residuals on entry to the Picard iteration and Δ_j is the vector of head changes returned from the PCG solver. The norm H_j is a maximum norm of the hydraulic head changes Δ_j. Variables θ_u, θ_l, H_{lim}, and ψ are supplied by the user: θ_u represents the maximum allowable damping parameter, θ_l is the minimum allowable damping parameter, and H_{lim} is the maximum allowable

head change (discussed in the next paragraph). The parameter ψ is the minimum value for which the ratio $\rho_n = n_j/n_{j-1}$ can dictate an increase in ϕ (the surrogate for θ_j) under favorable Picard progress.

given: θ_u, θ_l, ψ, n_j, H_j, H_{lim} **then**
$cnt = 0$
for every Picard iteration j **do**
  if not reentrant **then**
    return
  end if
  $\rho_n = n_j/n_{j-1}$; $\rho_h = H_j/H_{j-1}$
  if $\rho_n < 1$ and $\rho_h < 1$ **then**
    $\lambda = \log_{10}\rho_n / \log_{10}\psi$
    if $\lambda < 1$ **then**
      $\phi = \theta_{j-1} + \lambda(\theta_u - \theta_{j-1})$
    else
      $\phi = \theta_u$
    end if
    $cnt = 0$
  end if
  if $\rho_n > 1$ **then**
    $\phi = \theta_{j-1}/\rho_n$
  end if
  if $\rho_h > 1$ **then**
    $\phi = \theta_{j-1}/\rho_h$
  end if
  $\theta_j = \sqrt{\phi\,\theta_{j-1}}$
  if $|H_j| > H_{lim}$ and $\theta_j > H_{lim}/|H_j|$ **then**
    $\theta_j = H_{lim}/|H_j|$
  end if
  if $\theta_j < \theta_l$ **then**
    $\theta_j := \theta_l$
    $cnt = cnt + 1$
    if $cnt > 10$ **then**
      $\theta_j := \sqrt[3]{\theta_l^2\,\theta_u}$
    end if
  end if
end for

Algorithm 1. Adaptive damping.

Other measures for the norm n_j are possible, but limited experimentation has shown that this particular choice functions reasonably well. As the algorithm depends upon information from the previous Picard iteration (n_{j-1}, H_{j-1}, and θ_{j-1}), a starting value, θ_0, for the damping parameter is needed. For the initial stress period, the starting value for the damping parameter is taken to be $\theta_0 = \sqrt{\theta_u\,\theta_l}$. For subsequent stress periods, the starting value is weighted somewhat toward θ_u: $\theta_0 = \sqrt[3]{\theta_u^2\,\theta_l}$. If, through the progress of the Picard iteration, $\psi \geq \rho_n$, then ϕ is simply set to θ_u. The equivalent PCGN variables are `DAMP` $= \theta_u$, `DAMP_LB` $= \theta_l$, and `RATE_D` $= \psi$. Smaller values of `RATE_D` provide for a slower recovery rate, thus preventing θ from increasing

too rapidly. Values of RATE_D in the range 0.01 to 0.1 have provided an adequate recovery rate in test problems, but other values are possible. It should be noted that the purpose of this scheme is to obtain convergence of a nonlinear problem that might not otherwise converge; the scheme is heavily weighted to backing down the damping parameter when adversity in the Picard iteration is encountered. As a result, this scheme usually requires more total iterations to achieve convergence than the other damping modes available with the PCGN variable ADAMP. For any given problem, adaptive damping can improve the accuracy of the Picard solution, but perversely can also have the reverse effect.

Also included in the adaptive damping algorithm is the capability to directly control the effect of the maximum head change, H_j, on the updated head vector, h_j; the usage here is essentially identical to that of Banta (2006). By specifing a positive, non-zero value for H_{lim}, the head-change limit variable, then H_j is constrained such that its contribution to h_j will not exceed H_{lim}. This action is accomplished by altering the damping parameter θ_j; if θ_j is greater than the ratio H_{lim}/H_j, then θ_j is reset to this ratio. Overriding the damping parameter causes all head changes Δ_j to be so damped, including the element of Δ_j that corresponds to H_j; thus, the effective maximum allowable head change becomes H_{lim}. So long as the H_j is less than H_{lim}, the damping parameter produced by the adaptive damping algorithm is unaffected by the head-change limit. It should be noted that a damping factor produced by limiting the maximum head change supersedes any damping factor produced by the adaptive damping algorithm. Activating the H_{lim} parameter can be beneficial to the convergence of the Picard iteration, particularly in simulations where large, isolated changes in head occur. However, selecting an appropriate value for H_{lim} is largely a trial and error process; initially, the user may need to run the simulation with this option turned off ($H_{lim} = 0$) and examine the output diagnostics for the Picard iteration. As noted in the *Output Diagnostics for the Picard Iteration* section of this report, these diagnostics contain a listing of the maximum head change for every Picard iteration. The equivalent PCGN variable for H_{lim} is CHGLIMIT.

The other damping variant available is obtained when PCGN variable ADAMP $= 2$. In this mode, the damping variable θ is increased from some minimum value θ_0 to some maximum θ_u, so long as the Picard iteration is progressing satisfactorily. The model for the increase is $\theta_j = \theta_{j-1} +$ RATE_D θ_{j-1}, where RATE_D is a PCGN variable that determines the rate of increase and j is the Picard iteration number. Should $\theta_j > \theta_u$, then θ_j is set equal to θ_u. The ratios $\rho_n = n_j/n_{j-1}$ and $\rho_h = H_j/H_{j-1}$ are used to measure the progress of the Picard iteration; if $\rho_n < 1$ and $\rho_h < 1$, then progress is considered to be satisfactory. The upper limit θ_u for the damping parameter is set with the PCGN variable DAMP. For the first stress period, the starting damping factor θ_0 is set to the PCGN variable DAMP_LB; for subsequent stress periods, an intermediate value is used where θ_0 is set to the geometric mean of DAMP and DAMP_LB. The assumption here is that more damping is required early in a stress period and that, as the simulation progresses, damping can be relaxed to some upper limit DAMP $= \theta_u$. Values of RATE_D in the range 0.1 to 0.01 provided an adequate increase in θ in test problems, but other values are possible.

From the previous discussion, it is apparent an implicit assumption has been made that the first stress period will likely be the most difficult as far as convergence of a nonlinear problem is concerned. This assumption is frequently correct, particularly when the first stress period consists of a steady-state approximation of predevelopment stresses in the aquifer and dewatering is involved in the simulation. If a static pressure head is assumed as the starting head for the nonlinear iteration, convergence of the predevelopment simulation under dewatering conditions often proves difficult. On the other hand, a large increase in applied stresses in some later stress period also may result in convergence difficulties; again these difficulties usually arise when simulating dewatering conditions. Thus, it is not inconceivable that, in the selection of Picard iteration parameter, the modeler will be more concerned with resolving a convergence problem caused by stresses introduced in a later simulation period.

Input Instructions for the PCGN Solver

The PCGN solver package is invoked by inserting the file type "PCGN" in the MODFLOW Name file (Harbaugh and others, 2000). This entry in the Name file also is used to associate the file type "PCGN" with the name of a file from which the input values for the PCGN solver package are read; these input values are described in this section. Either fixed or free format is available for reading all values on the input list. The PCGN data file should contain the following data items in the order given:

```
1. ITER_MO, ITER_MI, CLOSE_R, CLOSE_H
2. RELAX, IFILL, UNIT_PC, UNIT_TS
3. ADAMP, DAMP, DAMP_LB, RATE_D, CHGLIMIT
4. ACNVG, CNVG_LB, MCNVG, RATE_C, IPUNIT
```

Optional comments may be added to the input data file by preceding the comment with the symbol # in the first column; these comments may appear anywhere in the data file. If fixed format is selected, then the corresponding format types, for the preceding variables, are as follows:

```
1. 2I10,2F10.0
2. F10.0,3I10
3. I10,4F10.0
4. I10,F10.0,I10,F10.0,I10
```

The variables on the line 1 above generally manage the inner and outer iterations and overall convergence of the problem. The line 2 variables generally pertain to the PCG solver. The line 3 variables generally manage the damping applied when updating a nonlinear problem, and the line 4 variables manage the convergence of the PCG solver.

General Solver Parameters: Line 1

A description of the line 1 variables, which manage the inner and outer iterations and overall convergence of the problem, follows:

`ITER_MO`, integer variable: `ITER_MO` is the maximum number of Picard (outer) iterations allowed. For nonlinear problems, this variable must be set to some number greater than one, depending on the problem size and degree of nonlinearity. If `ITER_MO` is set to 1, then the PCGN solver assumes that the problem is linear and the input requirements are greatly truncated.

`ITER_MI`, integer variable: `ITER_MI` is the maximum number of PCG (inner) iterations allowed. Generally, this variable is set to some number greater than one, depending on the matrix size, degree of convergence called for, and the nature of the problem. For a nonlinear problem, `ITER_MI` should be set large enough that the PCG iteration converges freely with the relative convergence parameter ε described in the *Parameters Related to Convergence of Inner Iteration: Line 4* subsection.

`CLOSE_R`, real variable: `CLOSE_R` is the residual-based stopping criterion for iteration. This parameter is used differently, depending on whether it is applied to a linear or nonlinear problem:

`ITER_MO` $= 1$: For a linear problem, the variant of the conjugate gradient method outlined in algorithm 2 is employed, but uses the absolute convergence criterion in place of the relative convergence criterion. `CLOSE_R` is used as the value in the absolute convergence criterion for quitting the PCG

iterative solver. CLOSE_R is compared to the square root of the weighted residual norm v. This norm is defined as $v = r^T M^{-1} r$, where M represents the preconditioning matrix with the PCG algorithm and r is a vector of residuals. In particular, if $\sqrt{v}$ is less than CLOSE_R, then the linear PCG iterative solve is said to have converged, causing the PCG iteration to cease and control of the program to pass out of the PCG solver.

ITER_MO > 1: For a nonlinear problem, CLOSE_R is used as a criterion for quitting the Picard (outer) iteration. CLOSE_R is compared to the square root of the inner product of the residuals (the residual norm), $[r^T r]^{\frac{1}{2}}$, as calculated on entry to the PCG solver at the beginning of every Picard iteration. If this norm is less than CLOSE_R, then the Picard iteration is considered to have converged.

CLOSE_H, real variable: CLOSE_H is used as an alternate stopping criterion for the Picard iteration needed to solve a nonlinear problem. The maximum value of the head change is obtained for each Picard iteration, after completion of the inner, PCG iteration. If this maximum head change is less than CLOSE_H, then the Picard iteration is considered tentatively to have converged. However, as nonlinear problems can demonstrate oscillation in the head solution, the Picard iteration is not declared to have converged unless the maximum head change is less than CLOSE_H for three Picard iterations. If these Picard iterations are sequential, then a good solution is assumed to have been obtained. If the Picard iterations are not sequential, then a warning is issued advising that the convergence is conditional and the user is urged to examine the mass balance of the solution.

As convergence of the Picard iteration may be acheived by meeting either the CLOSE_R or the CLOSE_H criterion, care must be taken in their selection. One should, in any case, check the mass balance of the solution for the problem in question to verify that a reasonable result has been obtained. Maximum head-change values at convergence are generally two to four order of magnitude smaller than the residual norm $[r^T r]^{\frac{1}{2}}$.

Parameters Related to PCG Solver: Line 2

A description of the line 2 variables, pertaining to the PCG solver, follows:

RELAX, real variable: RELAX is the so-called relaxation parameter for the modified incomplete Cholesky (MIC) preconditioner (see algorithms 7 and 11); under MIC preconditioning, row sum agreement between the original matrix and the preconditioning matrix is created by pivot modification. When RELAX $= 0$, then the MIC corresponds to the ordinary incomplete Cholesky preconditioner, the effect of the modifications to the incomplete Cholesky having been nullified. When RELAX $= 1$, then these modifications are in full force. Generally speaking, it is of advantage to use the modifications to the incomplete Cholesky algorithm; a value of RELAX such that $0.9 <$ RELAX < 1 is generally advised for most problems. Values RELAX $= 1$ are not advised, particularly when IFILL $= 0$, as poor performance of the PCG solver may result (van der Vorst, 2003). However, experience has shown that a value close to 1, such as RELAX $= 0.99$, usually provides good performance.

IFILL, integer variable: IFILL is the fill level of the MIC preconditioner. Preconditioners with fill levels of 0 and 1 are available (IFILL $= 0$ and IFILL $= 1$, respectively). Generally, the higher the fill level, the more preconditioning imparted by a MIC preconditioner. However, the actual preconditioning provided is also influenced by the modification to the incomplete Cholesky algorithm (see RELAX above). For most well-behaved CCFD matrices, a MIC preconditioner with fill level 0 will slightly outperform a MIC preconditioner with fill level 1, provided RELAX ≈ 1. For problems where the matrix equation is not well behaved or for a nonlinear problem where convergence is not easily achieved, a fill level of 1

may provide the additional preconditioning necessary to obtain convergence. One should be aware that the PCGN solver computer memory requirements of the level 1 MIC preconditioner are about double those of the level 0 preconditioner.

UNIT_PC, integer variable: UNIT_PC is the unit number of an optional output file where progress for the inner PCG iteration can be written. Progress diagnostics consist of the weighted residual norm v_i for every iteration i of the PCG solver; this information is ouput for every time step and every Picard iteration in the simulation. If this option is used (UNIT_PC > 0), the integer value of the unit, along with the file name and type "DATA," should be given in the MODFLOW Name file (Harbaugh and others, 2000). In many instances, asking for this information will cause very large data files to be produced; it is not expected that this option will be used by most modelers.

UNIT_TS, integer variable: UNIT_TS is the unit number of an optional output file where the actual time in the PCG solver is accumulated. The object here is to capture actual PCG solver time rather than total run time. If this option is used (UNIT_TS > 0), the integer value of the unit, along with the file name and type "DATA," should be given in the MODFLOW Name file. It is not expected that this option will be used by most modelers.

If ITER_MO $= 1$, then no further data are read (or needed) by the PCGN package to solve a linear problem. If the problem is nonlinear (ITER_MO > 1), then the remaining two lines (3 and 4) of data are read and processed.

Parameters Related to Damping: Line 3

A description of the line 3 variables, controlling the damping of the nonlinear problem, follows:

ADAMP, integer variable: ADAMP defines the mode of damping applied to the linear solution. In general, damping determines how much of the head changes vector Δ_j shall be applied to the hydraulic head vector h_j in Picard iteration j: $h_j = h_{j-1} + \theta\Delta_j$, where θ is the damping parameter. The available damping modes are:

ADAMP $= 0$: Ordinary damping is employed and a constant value of damping parameter $\theta =$ DAMP will be used throughout the Picard iteration. This option requires a valid value for DAMP.

ADAMP $= 1$: Adaptive damping is employed; see algorithm 1. Adaptive damping changes the damping parameter θ in response to the difficulty the nonlinear solver encounters in solving a given problem. Essentially, the nonlinear solver looks to increase θ should the convergence of the Picard iteration proceed satisfactorily, but otherwise causes θ to decrease. Adaptive damping can be useful for problems that do not converge readily, but otherwise should be avoided as it generally requires more total iterations. This option requires valid values for variables DAMP, DAMP_LB, RATE_D, and CHGLIMIT. Adaptive damping also admits the possibility of directly limiting the the maximum head change applicable to update the hydraulic heads; see CHGLIMIT below. If this option is not desired, then CHGLIMIT should be set to zero.

ADAMP $= 2$: Enhanced damping algorithm in which the value of θ is increased (but never decreased) provided the Picard iteration is proceeding satisfactorily. This enhanced damping allows θ to increase from a minimum value to a maximum value DAMP by a rate equal to RATE_D. The minimum value in the first stress period is DAMP_LB; for subsequent stress periods it is the geometric mean of DAMP and DAMP_LB. This option requires valid values for DAMP, DAMP_LB, and RATE_D.

DAMP, real variable: The varible DAMP restricts the damping parameter θ; generally, $0 <$ DAMP < 1. Its function for the various modes of ADAMP are:

ADAMP $= 0$: The damping parameter θ takes on the value DAMP and is maintained constant throughout the simulation.

ADAMP > 0: The value of DAMP will be treated as the upper limit for θ in the enhanced damping or adaptive damping algorithms.

DAMP_LB, real variable: DAMP_LB represents a bound placed on θ; generally, $0 <$ DAMP_LB $<$ DAMP. For the various modes of ADAMP > 0, DAMP_LB serves the following purposes:

ADAMP $= 1$: In the adaptive damping algorithm, DAMP_LB represents the lower limit to which θ, under adverse adaptive damping conditions, will be allowed to fall.

ADAMP $= 2$: In the enhanced damping algorithm, DAMP_LB is the starting value (or a component of the starting value) for the damping parameter θ used in the initial Picard iteration of every stress period.

RATE_D, real variable: This variable is a rate parameter; generally, $0 <$ RATE_D < 1. For the various modes of ADAMP > 0, RATE_D serves the following purposes:

ADAMP $= 1$: RATE_D sets the recovery rate for the damping factor θ in response to the progress in the Picard iteration; it also forms a limit on the response function to progress in the Picard iteration. See algorithm 1 for usage when ADAMP $= 1$; in this algorithm, RATE_D $= \psi$. Typical values for RATE_D, under this scenario, are $0.01 <$ RATE_D < 0.1. Under adaptive damping, if the user finds that the damping factor θ increases too rapidly, then reducing RATE_D will slow the rate of increase.

ADAMP $= 2$: Provided the Picard iteration is progressing satisfactorily, RATE_D adjusts the damping factor θ upward such that $\theta_j = \theta_{j-1} +$ RATE_D θ_{j-1}, where j is the Picard iteration number. Typical values for RATE_D, under this scenario, are $0.01 <$ RATE_D < 0.1, although larger or smaller values may be used.

CHGLIMIT, real variable: This variable limits the maximum head change applicable to the updated hydraulic heads in a Picard iteration. Provided that the current damping factor is greater than the ratio of CHGLIMIT to the maximum head change and that this ratio is less than one, then the damping factor is reset to the value of the ratio. This option is available only in association with adaptive damping: ACNVG $= 1$. If CHGLIMIT $= 0.0$, then adaptive damping proceeds without this feature.

Parameters Related to Convergence of Inner Iteration: Line 4

The PCGN variables that control the convergence of the inner iteration are read in the order given on line 4; a description of these variables follows:

ACNVG, integer variable: ACNVG defines the mode of convergence applied to the PCG solver. In general, the relative stopping criterion for PCG iteration is $v_i < \varepsilon v_0$, where v_0 is the weighted residual norm on entry to the PCG solver, ε is the relative convergence parameter, and v_i is the same norm at PCG iteration i. The available convergence modes are:

ACNVG $= 0$: The standard convergence scheme is employed using the variant of the conjugate gradient method outlined in algorithm 3. The standard relative convergence is denoted by ε_s and usually

takes the value 0.1; this value is assigned to the relative convergence ε. No additional variables are used.

ACNVG $= 1$: Adaptive convergence is employed using the variant of the conjugate gradient method outlined in algorithm 2. The adaptive convergence scheme adjusts the relative convergence ε of the PCG iteration based on a measure of the nonlinearity of the problem. Under this scheme, ε is allowed to vary such that CNVG_LB $< \varepsilon < \varepsilon_s$, where the exact value of ε is dependent on the measure of nonlinearity. This option requires a valid value for variable CNVG_LB.

ACNVG $= 2$: Enhanced convergence is employed using variant of the conjugate gradient method outlined in algorithm 3. If the variable enhancement option is employed (RATE_C > 0), then ε_s is taken as the upper limit for ε; see *Limiting the Inner Iteration* subsection for details. This option requires valid values for variables MCNVG and RATE_C.

CNVG_LB, real variable: Variable CNVG_LB is used only in convergence mode ACNVG $= 1$. CNVG_LB is the minimum value that the relative convergence ε is allowed to take under the self-adjusting convergence option. The objective here is to prevent ε from becoming so small that the PCG solver takes an excessive number of iterations. Valid range for variable: $0 <$ CNVG_LB $< \varepsilon_s$; a value of CNVG_LB $= 0.001$ usually produces reasonable results.

MCNVG, integer variable: Variable MCNVG is used only in convergence mode ACNVG $= 2$. MCNVG increases the relative PCG convergence criteria by a power equal to MCNVG; that is, letting $p =$ MCNVG, then the relative convergence criterion ε is enhanced such that $\varepsilon = \varepsilon_s^p$, where $0 < p \leq 6$. If MCNVG is set to a value greater than 6, then PCGN resets MCNVG $= 6$ and issues a warning message. If RATE_C $= 0$, then this enhanced relative convergence criterion is applied uniformly throughout the simulation; the relative convergence, in this case, is not adjusted for stress changes in the simulation. MCNVG must be set to a value greater than zero when ACNVG $= 2$; otherwise a data error is declared and ACNVG is reset to zero.

RATE_C, real variable: Variable RATE_C is used only in convergence mode ACNVG $= 2$; this option results in variable enhancement of ε. If $0 <$ RATE_C < 1, then enhanced relative convergence is allowed to decrease by increasing ε as follows: $\varepsilon_j = \varepsilon_{j-1} +$ RATE_C ε_{j-1}, where j is the Picard iteration number; this change in ε occurs so long as the Picard iteration is progressing satisfactorily. If RATE_C ≤ 0, then the value of ε set by MCNVG remains unchanged through the Picard iteration. It should be emphasized that RATE_C must have a value greater than 0 for the variable enhancement to be effected; otherwise ε remains constant. The assumption here is that a better solution of the linear equations is needed initially to improve the nonlinear Picard solution, but that this need abates with additional Picard iterations. Typical values for RATE_C are $0.01 <$ RATE_C < 0.1, although larger or smaller values may be used.

IPUNIT, integer variable: Variable IPUNIT enables progress reporting for the Picard iteration. If IPUNIT ≥ 0, then a record of progress made by the Picard iteration for each time step is printed in the MODFLOW Listing file (Harbaugh and others, 2000). This record consists of the total number of dry cells at the end of each time step as well as the total number of PCG iterations necessary to obtain convergence. In addition, if IPUNIT > 0, then extensive diagnostics for each Picard iteration is also written in comma-separated format to a file whose unit number corresponds to IPUNIT; the name for this file, along with its unit number and type "DATA," should be entered in the MODFLOW Name file. Diagnostics output by this last option are given in the *Output Diagnostics for the Picard Iteration* section of this report. If IPUNIT < 0 then printing of all progress concerning the Picard iteration is suppressed, as well as information on the nature of the convergence of the Picard iteration.

Generally speaking, if little is known of the characteristics of the nonlinear problem, one is advised to start the modeling process with ACNVG = 0, ADAMP = 0 and DAMP = 0.5; if damping at this level does not produce convergence of the nonlinear problem, then damping should be reduced, in the extreme to the DAMP = 0.01 level. If convergence is still not achieved, and the Picard iteration progress report (IPUNIT > 0) indicates that residuals are not being reduced, then the modeler should use enhanced convergence (ACNVG = 2, MCNVG > 1) with no variation allowed (RATE_C ≤ 0). If difficulties persist, then the modeler may wish to attempt adaptive damping (ADAMP = 1) with small starting and limiting parameters: DAMP_LB = 0.001 and DAMP = 0.1. Memory usage permitting, convergence of the Picard iteration might also be aided if fill level IFILL = 1 can be used. If any of these modes produce a Picard solution with a reasonable mass balance, then the modeler will likely wish to refine these modes so as to increase the efficiency and accuracy of the Picard iteration.

Results thus far of testing the PCGN package on various nonlinear problems have not indicated that any particular set of Picard parameters have preference. Some nonlinear problems simply will not converge without use of adaptive damping, although simultaneous use of the standard convergent scheme is frequently adequate. Convergence of dewatering problems is frequently enhanced if a bound for the maximum head change, CHGLIMIT, can also be instituted. On the other hand, several nonlinear problems showed a strong preference to increasing the accuracy of the linear solution by allowing ACNVG > 0. Here, the ACNVG = 1 option was generally found to be less useful than the ACNVG = 2 option when used in conjunction with RATE_C set to a small positive value. That is, using an option that initially tightens the relative convergence criterion ε, but slowly relaxes it so that standard convergence is gradually reestablished, is more likely to enhance performance of the solver. In dewatering simulations, the ACNVG = 2 option frequently diminishes the number of cells that go dry. That is, having a slightly more rigorous solution to the linear approximation early in the Picard iteration may decrease the number of cells that go dry.

With regard to the Picard parameters (ITER_MO > 1), the PCGN package does limited checking to ascertain that the input data are consistent and correct. If an inconsistent or out of range entry is encountered, the package usually resets ADAMP and (or) ACNVG to 0, issues a warning, and allows the computation to continue. The user must examine the MODFLOW Listing file to detect such warnings; if such warnings are found, the PCGN data input file should be examined for errors.

Output Diagnostics for the Picard Iteration

The IPUNIT option in the PCGN input file allows selection of output diagnostics showing the progress (or lack thereof) of the Picard iteration. These diagnostics can be helpful to the modeler in a number of ways and thus the option is highly advisable. The diagnostics are written in a comma separated value (CSV) format and thus can be displayed with any standard spreadsheet program. The column values for these diagnostics are as follows:

Iteration: Picard iteration number.

ib0_count: The number of cells that have become dewatered since the beginning of the time step. This diagnostic reflects changes in the MODFLOW integer array IBOUND, which tracks active and inactive cells.

ratio_l: Measure of nonlinearity of problem. If the problem is linear or nearly so, then ratio_l ≈ 1; otherwise ratio_l < 1. This measure only appears in the output if ACNVG = 1; see algorithm 2 for further explanation.

Damp: Value of damping parameter applied in Picard iteration Iteration.

L2hr: A measure of the current error in the solution. L2hr is defined as $\sqrt{r^T r \Delta^T \Delta}$, where r is the vector of residuals on entry to the Picard iteration and Δ is the vector of raw head changes returned from the PCG solver.

Hprev: Hydraulic head from the previous Picard iteration at the location of the current maximum head change.

Hcurr: Hydraulic head from the current Picard iteration at the location of the current maximum head change.

Max_chg: Maximum head change from the current Picard iteration.

Layer: Layer location of maximum head change.

Row: Row location of maximum head change.

Column: Column location of maximum head change.

Maximum head changes for Stress Period 1, Time Step 1

Iteration	ib0_count	ratio_l	Damp	L2hr	Hprev	Hcurr	Max_chg	Layer	Row	Column
1	0	0	0.22361	1.64E+15	66	8300.73	36827	3	57	111
2	58	1.86E−11	0.33437	1.30E+13	272.4462	789.9128	1547.6	3	17	57
3	75	0.002721	0.3466	8.32E+12	2526.714	2800.7	790.49	3	59	71
4	217	0.002827	0.35786	5.32E+12	2800.7	3006.825	576	3	59	71
5	353	0.003710	0.36938	3.25E+12	3275.678	3444.911	458.15	3	59	72
6	465	0.008221	0.38111	1.88E+12	3444.911	3590.265	381.4	3	59	72
7	540	0.018372	0.39161	1.10E+12	3343.895	3473.099	329.93	4	60	142
8	597	0.04388	0.4013	6.39E+11	3701.577	3820.887	297.31	4	61	73
9	654	0.11078	0.1	6.42E+12	5506.659	5926.742	4200.8	4	68	154
10	704	0.075091	0.11368	5.16E+12	5910.416	6336.156	3744.9	8	68	154
11	790	0.077735	0.12675	4.16E+12	7173.872	7595.293	3324.9	13	66	159
12	892	0.040519	0.14215	3.20E+12	7903.551	8317.932	2915	4	65	91
13	1058	0.039211	0.15892	2.38E+12	9725.18	10128.13	2535.5	4	52	182
14	1189	0.039928	0.17645	1.72E+12	10124.3	10523.43	2262	11	54	181
15	1291	0.013858	0.196	1.17E+12	10523.43	10890.78	1874.2	11	54	181
16	1375	0.017671	0.21573	7.80E+11	10890.34	11217.19	1515.1	12	54	181
17	1459	0.024648	0.23839	4.72E+11	11217.19	11501.72	1193.6	12	54	181
18	1516	0.033835	0.26089	2.75E+11	11499.33	11737.99	914.77	12	54	42
19	1556	0.039872	0.28385	1.51E+11	11942.89	12136.33	681.51	3	54	179
20	1583	0.044593	0.30689	7.76E+10	12136.33	12288.17	494.78	3	54	179
21	1597	0.03964	0.3297	3.73E+10	12299.39	12413.66	346.59	3	54	41
22	1602	0.047049	0.35301	1.59E+10	10998.61	11075.29	217.22	12	50	28
23	1606	0.05339	0.37355	6.73E+09	12176.16	12231.69	148.64	13	60	177
24	1608	0.048071	0.39276	2.65E+09	12312.31	12352.28	101.77	13	58	180
25	1608	0.034048	0.40973	1.00E+09	12352.28	12379.11	65.499	13	58	180

Figure 1. Sample spreadsheet display of output diagnositics in comma separate value format; see text for explanation.

An example of a spreadsheet presentation of these output diagnostics is presented in figure 1. Diagnostics presented in these figures represent the first 25 Picard iterations for the first stress period in which a predevelopment steady state is being simulated. Picard parameters selected for this problem include both adaptive damping (ADAMP $= 1$) and adaptive convergence (ACNVG $= 1$). Generally, both L2hr and Max_chg become smaller as the Picard iteration tends toward convergence. However, at iteration 9 a sudden jump in the head change, Max_chg, and concurrent increase in error norm, L2hr, causes the damping factor to be reset to DAMP_LB $= 0.1$, the lower limit to which damping can fall under adverse conditions. This jump occurred in layer 4, row 68, column 154 of the computational mesh. At this point, a total of 654 cells have gone dry. Note that Hcurr $=$ damp $\times$ Max_chg $+$ Hprev for each picard iteration.

The total number of dry cells at the end of each time step is reported in the listing file for every simulation. This count is equivalent to appropriately summing of the maximum value of ib0_count reported for every time step in the CSV file. The maximum number of dry cells for a simulation is a good indicator of the uniqueness of the solution for the nonlinear problem. If the maximum number of dry cell changes substantially with changes in Picard parameters (increased or decreased damping and PCG convergence parameters), then the modeler is faced with the result that, although the nonlinear iteration may have converged in every case, the nonlinear solution is not unique. Usually, the modeler will select Picard parameters that tend to minimize the maximum number of dry cells that occur in any given nonlinear simulation.

Application of PCGN to a Nonlinear Problem

Results from ongoing work on flow modeling in the Denver Basin are presented as an example of an application of the PCGN package (S.S. Paschke and E.R. Banta, written commun., 2008); because the study is ongoing, discussion of the model itself is necessarily limited. A discussion of the hydrogeologic units found in the Denver Basin can be found in Robson (1987). The model comprises 12 layers, 5 of which are confining layers. Each layer is discretized with 124 rows and 84 columns, for a total of 124,992 cells. The simulation includes approximations of hydraulic connections with rivers and approximations of evapotranspiration. The model is run over 16 stress periods, with the first being a steady-state approximation of the early predevelopment stresses. The starting hydraulic head field for the predevelopment steady-state simulation was set equal to the land-surface elevation. A fair number of cells become dry in the course of this modeling; the majority of the dewatering occurs in the first stress period as the heads adjust to the assumed predevelopment stresses. The relation of convergence of the Picard iteration, total dry cells, and execution time to Picard parameter values, for this nonlinear simulation, is the main objective of this discussion.

Table 1. Sample results from Denver Basin simulation.

[DC: domain integrity compromised; NC: simulation did not converge; NA: not applicable; Trial 9: run with CNVG_LB = 0.01; Trial 10: run with MCNVG = 2 and RATE_C = 0.01]

	Picard parameters						IFILL = 0		IFILL = 1	
Trial no.	ADAMP	DAMP	DAMP_LB	RATE_D	CHGLIMIT, ft	ACNVG	Total no. dry cells	Run time, min	Total no. dry cells	Run time, min
1	0	0.1				0	DC	NA	DC	NA
2	0	0.05				0	1103	21.86	NC	NA
3	0	0.01				0	774	104.2	821	99.1
4	2	0.1	0.001	0.01		0	NC	NA	NC	NA
5	1	0.5	0.001	0.01	0.0	0	1145	9.41	1145	8.41
6	1	0.5	0.001	0.01	100.0	0	1064	10.72	1147	10.17
7	1	0.5	0.001	0.01	10.0	0	931	13.55	946	12.97
8	1	0.5	0.001	0.01	1.0	0	730	35.81	775	32.87
9	1	0.5	0.001	0.01	1.0	1	783	47.89	812	32.17
10	1	0.5	0.001	0.01	1.0	2	737	36.34	773	32.76

A series of runs were made by varying Picard parameters while holding most PCG solver parameters and iteration parameters constant. Parameters held constant had the following values: ITER_MO = 20000,

ITER_MI = 80, CLOSE_R = 100.0, CLOSE_H = 0.0001 and RELAX = 0.99. The solver parameter IFILL can affect the nonlinear solution; both fill levels (0 and 1) were tested. The very small value used for CLOSE_H, relative to CLOSE_R, was found to be necessary to ensure a small mass-balance error; the relatively large value of CLOSE_R did not contribute significantly to the mass-balance error. The Picard parameters used during each run, along with the number of cells that went dry and the run time of completed simulations, are given in table 1; the standard convergence scheme (ACNVG = 0) was used in all trials except trials 9 and 10. The three initial trials in this table show efforts to force convergence with small, constant values for the damping parameter. In the first trial (DAMP = 0.1), the domain integrity was compromised when a small section became separated from the main body of the domain. This self-partitioning affected both IFILL = 0 and IFILL = 1 runs; both were terminated by the presence of a zero pivot. In trial 2 (DAMP = 0.05), the IFILL = 0 run did converge while the IFILL = 1 run did not. For IFILL = 0, the run time was rather attractive (22 min), but a large number of cells went dry. In trial 3 (DAMP = 0.01), both IFILL runs converged; IFILL = 0 produced fewer dry cells. However, the run time, at approximately 100 minutes, was excessive in both cases. Trial 4 was an attempt to remedy this excessive run time by using enhanced damping (ADAMP = 2) such that, for the early part of each stress period, a small damping parameter would be used but the parameter would increase until 10 percent of each head change would be applied to the solution. This effort failed to converge for either IFILL = 0 or IFILL = 1. Apparently, simply allowing the damping parameter to increase with Picard iterations still allowed excessive head change to be applied to the updated hydraulic head. In trial 5, adaptive damping (ADAMP = 1) was used with a relatively conservative lower bound (DAMP_LB = 0.001) but without a head-change limit (CHGLIMIT = 0.0). This trial was successful in that the Picard iteration did converge for both IFILL values; indeed, the run time was reduced to approximately 9 minutes. However, now the number of cells going dry became excessively large for both IFILL cases. To remedy this situation, a series of maximum head-change limits (CHGLIMIT) were used, starting with a value of 100.0 ft but eventually reducing to 1.0 ft (trials 6, 7, and 8). Trial 6 used CHGLIMIT = 100.0 ft; little change from trial 5 was found, although 81 fewer cells went dry for IFILL = 0. Change limits of 10.0 ft and 1.0 ft were more successful, causing substantially fewer cells to go dry but at the expense of a longer run times. From the perspective of fewest cells going dry, trial 8 with IFILL = 0 was the most successful run overall with 730 dry cells. The run time, at approximately 36 minutes, was also the longest of the CHGLIMIT trials. At this point, the modeler could investigate a compromise between the total number of dry cells and the run time performance of the simulation by varying the maximum head-change values between 1.0 ft and 10.0 ft. Here, it is assumed that the modeler was satisfied with the result when CHGLIMIT = 1.0 ft. In trials 9 and 10, the effect of changing the convergence criterion for the linear solver was explored. Adaptive convergence (ACNVG = 1) was applied in trial 9; adaptive convergence caused a small but substantial increase in the number of dry cells and, in the case of IFILL = 0, a substantial increase in the run time. The use of enhanced convergence, trial 10, did not offer a significant advantage over the results of trial 8 and, in the case of IFILL = 0, resulted in a small increase in the number of dry cells. It should be noted that, in all trials, IFILL = 1 produced shorter run times relative to IFILL = 0; however, in most cases the difference was on the order of 10 percent or less.

These results, representing the solution of a highly nonlinear problem, are likely to be typical for this class of problem. It should be pointed out that not all nonlinear problems are this sensitive to the choice of damping parameters; in other cases, forcing a better solution to the linear equations by selecting ACNVG > 0 can be the critical parameter.

Description of Program Modules

The PCGN package is written in standard Fortran 90. MODFLOW, with the PCGN source code included, has been compiled and run successfully with the Lahey/Fujitsu compiler, the Intel Fortran compiler, and the gfortran compiler. There are five main modules in the PCGN solver; these are:

`PCGN`: This module is the interface between the PCG solver and MODFLOW. The module contains subroutines to take the raw coefficients from MODFLOW and form the linear matrix equation and also to read in solver-control parameters as given in the *Input Instructions for the PCGN Solver* section. Most of the subroutines necessary to the nonlinear Picard iteration are contained in this module. Subroutines here are called from MAIN in MODFLOW and call subroutines in module `PCG_MAIN`

`PCG_MAIN`: This module acts as the front end to the PCG solver. Here, requests from subroutines in `PCGN` are interpreted and turned into requests for variations in the form of the linear solver. Subroutines here are called from `PCGN` and call subroutines in modules `PCG_SOLVE` and `MiUDU`.

`PCG_SOLVE`: This module contains the conjugate gradient loop necessary to the PCG algorithm (see the *Preconditioned Conjugate Gradient Method* section). This module is called from subroutines in `PCG_MAIN` and calls subroutines in modules `MAT_VEC_MULT` and `MiUDU`.

`MAT_VEC_MULT`: This module contains the vector matrix multiply algorithm necessary to the PCG algorithm. This module is called exclusively by subroutines in module `PCG_SOLVE`.

`MiUDU`: This module contains subroutines for the MIC preconditoners. These include the MIC fill level 0 and MIC fill level 1 factorizations, as well as subroutines for the forward elimination and back substitutions for these factorizations. Algorithms for these incomplete factorizations and inversions are given in the *Incomplete Cholesky Preconditioner* section of this report. The factorization subroutines in this module are called from subroutines in module `PCG_MAIN`, and the approximate inversions are called from `PCG_SOLVE`.

Preconditioned Conjugate Gradient Method

The PCG method is well illuminated in Golub and Van Loan (1983) and Axelsson (1996) and will not be described in detail in this document. The principal objective of this section is to describe two variants of the PCG method for use with Picard iteration, as implemented in PCGN, for solving nonlinear problems: adaptive convergence and standard inner convergence. In all cases, M represents the preconditioning matrix; two preconditioning matrices are used in the PCGN solver and can be applied to the two variants of the PCG method: a modified incomplete Cholesky with no fill, algorithms 8 and 9, and a modified incomplete Cholesky with one fill level, algorithms 12 and 13. The details of these preconditioners are presented in the *Incomplete Cholesky Preconditioner* section. Preconditioning in these algorithms is represented as the act of solving the system of equations $My_i = r_i$. The quantity r_0, on entry, is equivalent to the residual vector $r_0 = b - Ax_0$, where x_0 is the initial guess for a solution vector of the matrix equation $Ax = b$. With regard to Picard iteration to solve a nonlinear problem, x_0 is the solution to the linear equations obtained in the previous Picard iteration. In both variants of the algorithm, two stopping procedures for the PCG iteration are available. In the first stopping procedure, the PCG iteration is stopped when the value of a weighted norm of the updated residuals r_i falls below the value of ε_a, the absolute convergence criterion. The weighted norm of residuals ν_i, where $\nu_i = r_i^T M^{-1} r_i$, is used as it occurs naturally within the PCG algorithm; the absolute stopping criterion requires

that $\sqrt{\nu_i} < \varepsilon_a$. In the second stopping procedure, a relative criterion is used. The value of the weighted norm on entry to the PCG solver, ν_0, is calculated and compared with later values of ν_i; if $\nu_i < \varepsilon\nu_0$, where $0 < \varepsilon < 1$, then the PCG iteration is stopped. This relative procedure is used primarily in conjunction with the Picard iteration scheme for solving nonlinear problems, as it is desirable to produce a reasonably good solution for each Picard iteration but not necessarily a perfect solution. Thus the second stopping criterion only demands that the solution be improved until the ratio ν_i/ν_0 becomes less than ε. A discussion of the Picard iteration scheme, in conjunction with relative convergence, is presented in the *Limiting the Inner Iteration* subsection of this report. As we are mostly concerned with the nonlinear problem, it will be assumed that this latter stopping criterion is in force in the following discussion of PCG algorithms.

$\varepsilon = \varepsilon_u$
$y_0 = M^{-1} r_0$
$\nu_0 = r_0 \cdot y_0$
if reentrant **and** $\varepsilon_u < \nu_f/\nu_0$ **then**
  if $\nu_f/\nu_0 > \varepsilon_l$ **then**
    $\varepsilon = \nu_f/\nu_0$
  else
    $\varepsilon = \varepsilon_l$
  end if
end if
$\xi_0 = \rho_0$
$\rho_0 = A\,\xi_0$
$\mu_0 = \nu_0/(\xi_0^T \rho_0)$
for $i = 1, \cdots, m+1$ **do**
  $x_i = x_{i-1} + \mu_{i-1}\xi_{i-1}$
  $r_i = r_{i-1} - \mu_{i-1}\rho_{i-1}$
  $y_i = M^{-1} r_i$
  $\nu_i = r_i \cdot y_i$
  if $\nu_i < \varepsilon\nu_0$ **or** $i = m$ **then**
    $\nu_f = \nu_i$
    exit loop
  end if
  $\xi_i = \rho_i + \nu_i\,\xi_{i-1}/\nu_{i-1}$
  $\rho_i = A\,\xi_i$
  $\mu_i = \nu_i/(\xi_i^T \rho_i)$
end for

Algorithm 2. Conjugate gradient method, version 1.

PCG with Adaptive Convergence

Algorithm 2 contains pseudocode for the first PCG variant; in this variant, the relative convergence criterion ε is adjusted internally according to the nonlinearity of the problem. Nonlinearity is determined by comparing the entry value ν_0 of the weighted norm with the final value, ν_f, obtained in the previous Picard iteration. This ratio, ν_f/ν_0, has a value of 1 if the problem is linear; for extremely nonlinear problems, the ratio can become much less than 1. Algorithm 2 allows ε to vary between two bounds, ε_l, a lower bound, and ε_u, an upper bound, depending on the value of the ratio ν_f/ν_0. On entry to algorithm 2, the relative convergence

criterion ε is initially set to ε_u and then, if warranted, adjusted. If the problem is sufficiently nonlinear in that this ratio is less than ε_u, then the ratio ν_f/ν_0 is used for the value of the relative convergence criterion ε. If, through the Picard iteration, $\nu_f/\nu_0 < \varepsilon_l$, then ε is set to value of ε_l. The purpose of the lower bound, ε_l, is to prevent the PCG solver from taking an excessive number of iterations should ν_f/ν_0 become extremely small. PCGN uses ε_s, the standard relative convergence criterion, for the value of ε_u. The lower bound ε_l is set with the input variable `CNVG_LB`; a commonly used value for ε_l is 0.001. For details on the `CNVG_LB` parameter see the *Input Instructions for the PCGN Solver* section of this report. This procedure has the effect of improving the quality of the PCG solution whenever this indicator demonstrates nonlinear behavior. The assumption here is that as this indicator deviates farther from 1, a more accurate solution of the linear matrix equations will enhance the Picard iteration. Algorithm 2 also is used to solve linear problems, in which case the relative convergence criterion is replaced with an absolute convergence criterion.

$y_0 = M^{-1} r_0$
$\nu_0 = r_0 \cdot y_0$
$\xi_0 = \rho_0$
$\rho_0 = A\,\xi_0$
$\mu_0 = \nu_0/(\xi_0^T \rho_0)$
for $i = 1, \cdots, m$ **do**
  $x_i = x_{i-1} + \mu_{i-1}\xi_{i-1}$
  if $i > 1$ **and** $\nu_{i-1} < \varepsilon\nu_0$ **then**
    exit loop
  end if
  $r_i = r_{i-1} - \mu_{i-1}\,\rho_{i-1}$
  $y_i = M^{-1} r_i$
  $\nu_i = r_i \cdot y_i$
  $\xi_i = \rho_i + \nu_i\,\xi_{i-1}/\nu_{i-1}$
  $\rho_i = A\,\xi_i$
  $\mu_i = \nu_i/(\xi_i^T \rho_i)$
end for

Algorithm 3. Conjugate gradient method, version 2.

PCG with Standard Inner Convergence

The second variant of the PCG method assumes that a value for the relative convergence ε is determined prior to entering the conjugate gradient loop; pseudocode for this variant is given in algorithm 3. As there is no attempt to determine a value for ε internally, the exit point for the method is moved to a point just after the latest update of the solution, x_i. By placing the stopping criterion at this point, the weighted norm of residuals ν associated with the updated x_i is not calculated, but the expense of an additional approximate inversion, $y_i = M^{-1} r_i$, is avoided. Thus, this second variant always produces the most recent update of x_i, but the value of ν_i at convergence reflects the weighted norm of the residuals from the previous PCG iteration. With the design of the Picard iteration scheme herein, the exact stopping point of PCG solution is not particularly critical to the Picard iteration; see the *Description of Picard Iteration Scheme* section of this report for details of the Picard method. This procedure produces a slightly better PCG solution than indicated by the relative convergence at a small additional computational expense.

Incomplete Cholesky Preconditioning

A derivation of incomplete Cholesky preconditioning for the cell-centered finite-difference (CCFD) coefficient matrix is presented in this section; algorithms presented are based in preconditioners used in the PCGN solver. Concepts introduced include the basics of the pointwise Cholesky decomposition and incomplete Cholesky decomposition with 0 and 1 fill levels. A relaxation factor also is discussed whereby the pivots of the incomplete Cholesky decomposition are modified to improve the performance of the incomplete Cholesky preconditioners.

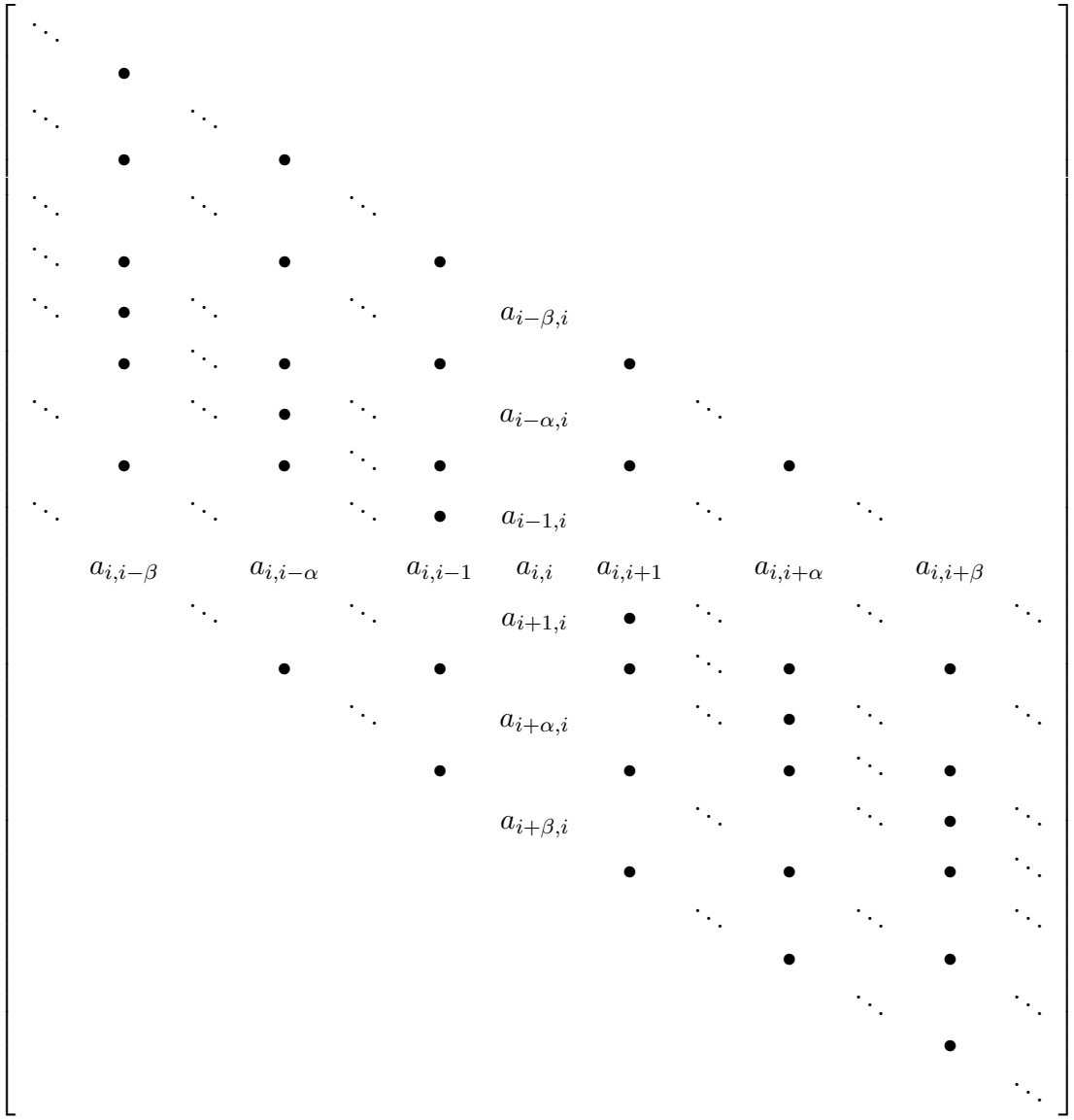


Figure 2. Generalized seven-point stencil cell-centered finite-difference matrix showing banding; large black dots denote locations of other potential non-zero entries.

The development for the incomplete Cholesky preconditioning presented herein makes extensive use of the sparsity of the banded matrix common to CCFD methods. Banding results when a regular sequential numbering scheme is used to determine the ordering of the CCFD coefficients. The seven-point stencil (SPS)

associated with a three-dimensional application of the CCFD method produces a symmetric matrix with seven bands: a diagonal band and three off-diagonal bands above and below the diagonal band. Matrices resulting from CCFD methods applied in lesser dimensions contain fewer bands; the five-point stencil (FPS) associated with two dimensions produces five bands. The banding arrangement of the FPS matrix is equivalent to the SPS matrix in which the outer bounds have been eliminated. The fill within bands, however, is different; with the exception of trivial equations, the outer bands, whether originating in FPS or SPS matrices, will be completely filled. This discussion will center on the SPS paradigm, with the assumption that lesser dimensional results will follow by logical reduction of these algorithms. For a cell near the center of a three-dimensional mesh, the matrix arrangement of the CCFD coefficients can be represented as shown in figure 2. In reference to this figure, if a three-dimensional grid is laid out such that its cell dimensions are n_x, n_y, n_z with numbering proceeding along the n_x dimension first and the n_z dimension last, then allowing $\alpha = n_x$ and $\beta = n_x n_y$, the bands can be denoted as $(i, i-\beta)$, $(i, i-\alpha)$, $(i, i-1)$, (i, i), $(i, i+1)$, $(i, i+\alpha)$, and $(i, i+\beta)$. As the CCFD matrix is symmetric, only the main diagonal and upper three bands, $(i, i+1)$, $(i, i+\alpha)$ and $(i, i+\beta)$, must be stored. The total number of unknowns n associated with this matrix is $n = n_x n_y n_z$. In the next section, the complete Cholesky decomposition is presented as a prelude to the incomplete Cholesky development.

Pointwise Cholesky Decomposition

The incomplete Cholesky preconditioner is based in the pointwise Cholesky decomposition procedure for a full, symmetric matrix A. In general, the Cholesky decomposition results from the knowledge that any symmetric, positive-definite $n \times n$ matrix $A = \{a_{ij}\}$ can be represented as a product of triangular matrices. In the following derivation, a $U^T DU$ variant of the Cholesky decomposition is used, avoiding the need for square roots. In this presentation, the symbol U represents an $n \times n$ upper triangular matrix such that

$$\begin{aligned} u_{ij} &\geq 0, \qquad j > i \\ u_{ij} &= 1, \qquad i = j \\ u_{ij} &= 0, \qquad j < i \end{aligned} \tag{1}$$

and $D = \text{diag}\,[d_1, \cdots, d_n]$, where $d_i > 0$; then (for example, Golub and Van Loan, 1983, p. 85)

$$U^T DU = A. \tag{2}$$

As there exists a one-to-one correspondence between the entries of $U^T DU$ and A, the following pointwise factorization can easily be developed:

$$d_i = a_{ii} - \sum_{\ell=1}^{i-1} d_\ell u_{\ell i}^2 \tag{3}$$

and

$$u_{ij} = \left[a_{ij} - \sum_{\ell=1}^{i-1} d_\ell u_{\ell i} u_{\ell j} \right] \Big/ d_i, \quad i < j, \tag{4}$$

where $D = \{d_i\}$ and $U = \{u_{ij}\}$. Clearly, the sequence of work must be $i = 1, \cdots, n$ so that all factors u_{rs}, where $r < i$ and $s > r$, have already been determined prior to the formation of the factors in row i.

Given a symmetric matrix equation $Ax = b$, where b is the known vector and x is to be determined, then x can be identified by using this factorization with the application of forward elimination and back substitution

(Golub and Van Loan, 1983; Saad, 2003). First allowing $L = U^T D$, then the forward step becomes the solution of $Ly = b$; as L is lower triangular, this is relatively easy. The backward step consists of the solution of $Ux = y$; this also is relatively easy because U is upper triangular. While the above procedure provides for good visualization, a more practical procedure provides for the vector b to be continuously replaced by the solution. Forward elimination and back substitution pseudocodes, using continuous replacement, are presented in algorithms 4 and 5.

```
b_1 := b_1/d_1
for i = 2,···,n do
   for j = 1,···,i−1 do
      b_i := [b_i − d_j u_ji b_j]/d_i
   end for
end for
```

Algorithm 4. Forward elimination, full matrix.

```
b_n := b_n
for i = n−1,···,1 do
   for j = i+1,···,n do
      b_i := b_i − u_ij b_j
   end for
end for
```

Algorithm 5. Back substitution, full matrix.

Unfortunately, the amount of work involved is proportional to n^3 (Golub and Van Loan, 1983), which makes the method computationally expensive; in addition this direct factorization method is subject to round-off error. Incomplete versions of these factorizations, however, lead to efficient preconditioners for the conjugate gradient method.

Incomplete Cholesky Decomposition with 0 Fill

The presentation here for the incomplete Cholesky preconditioning follows that given in Saad (2003) for the *LDU* decomposition of asymmetric matrices. Incomplete Cholesky preconditioning with 0 fill (IC(0)) is fairly standard for discretizations based on a regular grid; this fill level requires that only factors corresponding to non-zero entries a_{ij} in the matrix A be preserved. Thus, entries in the triangular matrix of the IC(0) incomplete decomposition will correspond to the entries in the upper triangle of A itself. The existence of a pattern S_0 that records these locations may be imagined; this pattern can be denoted as:

$$S_0 = \left\{(i,j) \mid 1 \le i < j \le n \text{ and } a_{ij} \neq 0\right\}. \tag{5}$$

The resulting IC(0) triangular matrix will be denoted as $\tilde{U}$ and the associated diagonal matrix as $\tilde{D}$. Elements of $\tilde{U}$ and $\tilde{D}$ are defined as follows:

$$\tilde{d}_i = a_{ii} - \sum_{\ell=1}^{i-1} \tilde{d}_\ell \tilde{u}_{\ell i}^2 \tag{6}$$

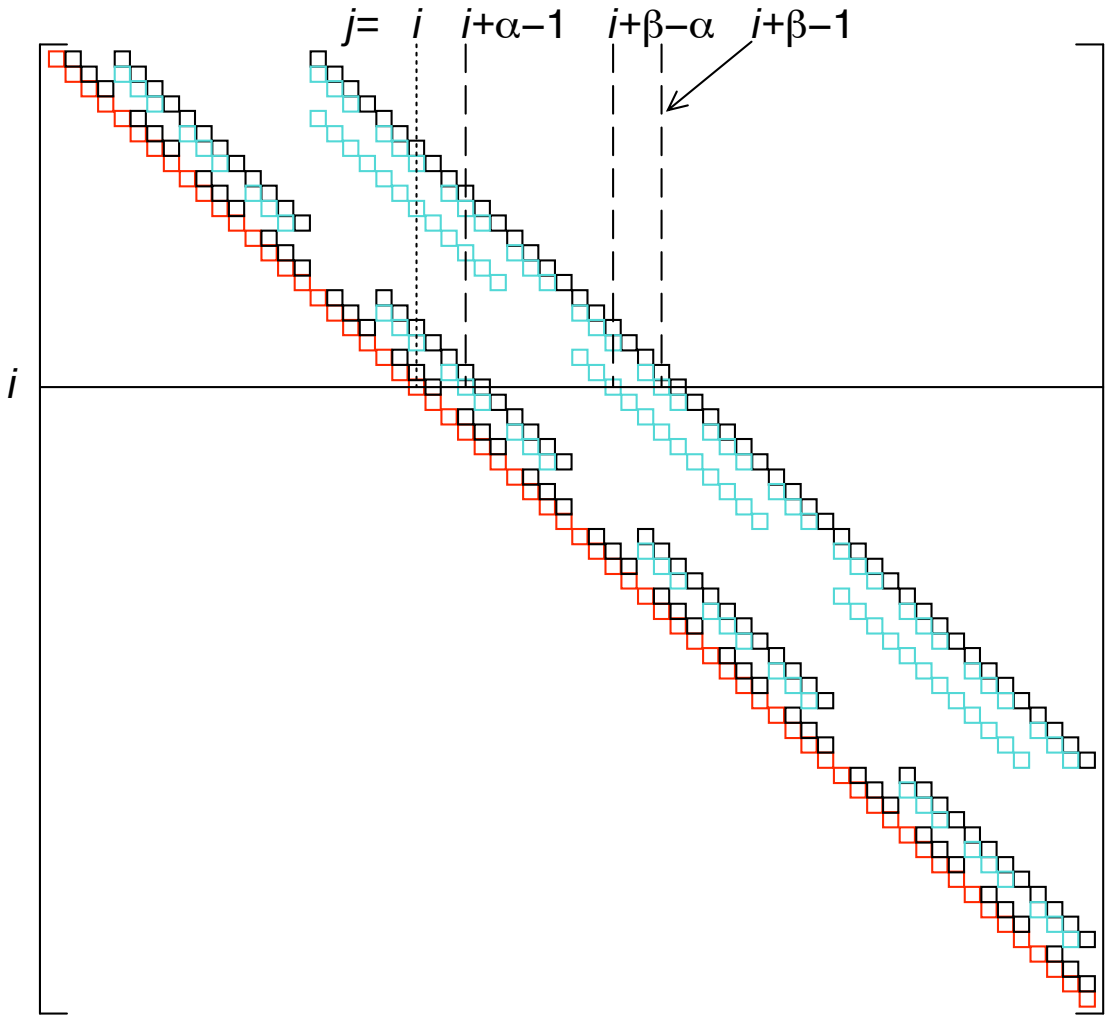


Figure 3. Pattern matrices for a seven-point stencil cell-centered finite-difference matrix derived from a $4 \times 4 \times 4$ domain. Black squares: pattern matrix for $\tilde{U}$ for fill level 0. Turquoise squares: augmentation to $\tilde{U}$ pattern matrix to form $\hat{U}$ pattern matrix for fill level 1. Red squares: $\mathrm{diag}[\tilde{u}_{11}, \cdots, \tilde{u}_{nn}] = I$ or $\mathrm{diag}[\hat{u}_{11}, \cdots, \hat{u}_{nn}] = I$.

and

$$\tilde{u}_{ij} = \begin{cases} \left[a_{ij} - \sum_{\ell=1}^{i-1} \tilde{d}_\ell \, \tilde{u}_{\ell i} \, \tilde{u}_{\ell j}\right] \Big/ \tilde{d}_i & (i,j) \in S_0 \\ 0, & (i,j) \notin S_0. \end{cases} \tag{7}$$

The elements of the diagonal matrix $\tilde{D}$ are affected insofar as the incomplete factors $\tilde{u}_{ij}$ are present or not present. In the event that A is the result of a regularly numbered SPS CCFD discretization, the banded structure (fig. 2) will be reflected in S_0. This banding allows the above equations to be written more simply as follows:

$$\tilde{d}_i = a_{ii} - \left[\tilde{d}_{i-1} \, \tilde{u}^2_{i-1,i} + \tilde{d}_{i-\alpha} \, \tilde{u}^2_{i-\alpha,i} + \tilde{d}_{i-\beta} \, \tilde{u}^2_{i-\beta,i}\right] \tag{8}$$

and

$$\tilde{u}_{ij} = a_{ij}/\tilde{d}_i, \quad j = i+1,\, i+\alpha,\, i+\beta. \tag{9}$$

In the event that the indices $i-1$, $i-\alpha$, or $i-\beta$ in equation 8 are non-positive, the corresponding factors and pivots do not contribute to $\tilde{d}_i$. Similarly, when the indices $i+1$, $i+\alpha$, or $i+\beta$ in equation 9 are greater than n, the corresponding factors $\tilde{u}_{ij}$ are not formed. In this manner, an approximate decomposition $\tilde{U}^T\tilde{D}\tilde{U}$ is formed, where $\tilde{U} = \{\tilde{u}_{ij}\}$, and $\tilde{D} = \{\tilde{d}_i\}$ result from the IC(0) procedure. A sample pattern matrix for $\tilde{U}$, corresponding to pattern S_0, is given in figure 3; this pattern matrix represents a SPS CCFD matrix derived from a $4\times4\times4$ domain.

The IC(0) factorization is frequently modified to include in the pivot d_i the next higher-order terms dropped from the incomplete factorization (Axelsson, 1996; van der Vorst, 2003). This modification has the effect of decreasing the condition number of the preconditioned system of equations (Barrett and others, 1994), thus decreasing the iterations required for convergence of the PCG algorithm. Following Hill (1990), the next higher-order contributions $\bar{u}_{ik}$ are denoted as follows:

$$\bar{u}_{ik} = \begin{cases} \sum_{\ell=1}^{i-1} \tilde{d}_\ell \tilde{u}_{\ell i}\tilde{u}_{\ell k}, & (i,k) \notin S_0 \\ 0, & (i,k) \in S_0. \end{cases} \tag{10}$$

When applied to SPS banded matrices, the product pairs forming $\bar{u}_{ik}$ are $\tilde{u}_{i-1,i}\,\tilde{u}_{i-1,i+\alpha-1}$, $\tilde{u}_{i-1,i}\,\tilde{u}_{i-1,i+\beta-1}$, and $\tilde{u}_{i-\alpha,i}\,\tilde{u}_{i-\alpha,i+\beta-\alpha}$. Contributions from $\tilde{U}^T$ also must be included in this modification; for column i these contributions are:

$$\bar{u}_{ki} = \begin{cases} \sum_{\ell=1}^{i-1} \tilde{d}_\ell \tilde{u}_{\ell k}\tilde{u}_{\ell i}, & (k,i) \notin S_0 \\ 0, & (k,i) \in S_0. \end{cases} \tag{11}$$

When applied to SPS banded matrices, the product pairs forming $\bar{u}_{ki}$ are $\tilde{u}_{i-\alpha,i-\alpha+1}\,\tilde{u}_{i-\alpha,i}$, $\tilde{u}_{i-\beta,i-\beta+1}\,\tilde{u}_{i-\beta,i}$, and $\tilde{u}_{i-\beta,i-\beta+\alpha}\,\tilde{u}_{i-\beta,i}$. Replacing $\tilde{d}_i$ of equation 8 with its modified form, then the pivots for the IC(0) decomposition can be written as follows:

$$\tilde{d}_i := \tilde{d}_i - \omega\left[\sum_{k=1}^{i-1}\bar{u}_{ki} + \sum_{k=i+1}^{n}\bar{u}_{ik}\right], \tag{12}$$

where ω is termed a relaxation factor; this parameter allows the user to decide on the degree of permitted modification ($0 \le \omega \le 1$). Taking note that S_0 reflects a banded matrix, the new pivot produced by incorporating these terms has the following form:

$$\begin{aligned} \tilde{d}_i &= a_{ii} - \left\{\tilde{d}_{i-1}\,\tilde{u}_{i-1,i}^2 + \omega\left[\tilde{d}_{i-1}\,\tilde{u}_{i-1,i}\,\tilde{u}_{i-1,i+\alpha-1} + \tilde{d}_{i-1}\,\tilde{u}_{i-1,i}\,\tilde{u}_{i-1,i+\beta-1}\right]\right. \\ &\quad + \tilde{d}_{i-\alpha}\,\tilde{u}_{i-\alpha,i}^2 + \omega\left[\tilde{d}_{i-\alpha}\,\tilde{u}_{i-\alpha,i}\,\tilde{u}_{i-\alpha,i-\alpha+1} + \tilde{d}_{i-\alpha}\,\tilde{u}_{i-\alpha,i}\,\tilde{u}_{i-\alpha,i+\beta-\alpha}\right] \\ &\quad \left. + \tilde{d}_{i-\beta}\,\tilde{u}_{i-\beta,i}^2 + \omega\left[\tilde{d}_{i-\beta}\,\tilde{u}_{i-\beta,i}\,\tilde{u}_{i-\beta,i-\beta+1} + \tilde{d}_{i-\beta}\,\tilde{u}_{i-\beta,i}\,\tilde{u}_{i-\beta,i-\beta+\alpha}\right]\right\} \\ &= a_{ii} - \tilde{d}_{i-1}\,\tilde{u}_{i-1,i}\left\{\tilde{u}_{i-1,i} + \omega\left[\tilde{u}_{i-1,i+\alpha-1} + \tilde{u}_{i-1,i+\beta-1}\right]\right\} \\ &\quad - \tilde{d}_{i-\alpha}\,\tilde{u}_{i-\alpha,i}\left\{\tilde{u}_{i-\alpha,i} + \omega\left[\tilde{u}_{i-\alpha,i-\alpha+1} + \tilde{u}_{i-\alpha,i+\beta-\alpha}\right]\right\} \\ &\quad - \tilde{d}_{i-\beta}\,\tilde{u}_{i-\beta,i}\left\{\tilde{u}_{i-\beta,i} + \omega\left[\tilde{u}_{i-\beta,i-\beta+1} + \tilde{u}_{i-\beta,i-\beta+\alpha}\right]\right\}. \end{aligned} \tag{13}$$

As in equation 8, terms containing factors with non-positive indices do not contribute to the pivot $\tilde{d}_i$. Also,

when the indices $i+1$, $i+\alpha$, or $i+\beta$ in equation 13 are greater than n, then terms containing factors with these indices do not contribute. When this modification is incorporated into the incomplete Cholesky decomposition with zero fill, the factorization is termed the modified incomplete Cholesky with 0 fill and denoted MIC$(0,\omega)$.

One of the great advantages of the MIC$(0,\omega)$ preconditioner for an SPS banded matrix is that it requires very little additional storage beyond that required to hold the original matrix. Because of the simplicity of the factors $\tilde{u}_{ij}$ (see eq. 9), it is apparent that they can be characterized immediately within the elimination and substitution steps, eliminating the need to define actual factors. The pivots $\tilde{d}_i$, also depending on the factors $\tilde{u}_{ij}$, also are modified to depend on entries a_{ij} of matrix A. Pseudocode for evaluating the pivots is given in two parts: a leading algorithm 6 which must precede the trailing algorithm 7. In general, both algorithms are applied sequentially when evaluating pivots $\tilde{d}_i$. The trailing algorithm 7 generally ensures that the algorithm does not access indirect factors a_{ij} outside of the $n \times n$ profile of the matrix.

$\tilde{d}_1 = a_{11}$
for $i = 2, \cdots, \alpha$ **do**
  $\tilde{d}_i = a_{ii} - a_{i-1,i}\left\{a_{i-1,i} + \omega\left[a_{i-1,i+\alpha-1} + a_{i-1,i+\beta-1}\right]\right\}/\tilde{d}_{i-1}$
end for
for $i = \alpha+1, \cdots, \beta$ **do**
  $\tilde{d}_i = a_{ii} - a_{i-1,i}\left\{a_{i-1,i} + \omega\left[a_{i-1,i+\alpha-1} + a_{i-1,i+\beta-1}\right]\right\}/\tilde{d}_{i-1}$
    $- a_{i-\alpha,i}\left\{a_{i-\alpha,i} + \omega\left[a_{i-\alpha,i-\alpha+1} + a_{i-\alpha,i+\beta-\alpha}\right]\right\}/\tilde{d}_{i-\alpha}$
end for
for $i = \beta+1, \cdots, n-\beta$ **do**
  $\tilde{d}_i = a_{ii} - a_{i-1,i}\left\{a_{i-1,i} + \omega\left[a_{i-1,i+\alpha-1} + a_{i-1,i+\beta-1}\right]\right\}/\tilde{d}_{i-1}$
    $- a_{i-\alpha,i}\left\{a_{i-\alpha,i} + \omega\left[a_{i-\alpha,i-\alpha+1} + a_{i-\alpha,i+\beta-\alpha}\right]\right\}/\tilde{d}_{i-\alpha}$
    $- a_{i-\beta,i}\left\{a_{i-\beta,i} + \omega\left[a_{i-\beta,i-\beta+1} + a_{i-\beta,i-\beta+\alpha}\right]\right\}/\tilde{d}_{i-\beta}$
end for

Algorithm 6. Pivots for MIC$(0,\omega)$: leading.

for $i = n-\beta+1, \cdots, n-\alpha$ **do**
  $\tilde{d}_i = a_{ii} - a_{i-1,i}\{a_{i-1,i} + \omega a_{i-1,i+\alpha-1}\}/\tilde{d}_{i-1}$
    $- a_{i-\alpha,i}\{a_{i-\alpha,i} + \omega a_{i-\alpha,i-\alpha+1}\}/\tilde{d}_{i-\alpha}$
    $- a_{i-\beta,i}\left\{a_{i-\beta,i} + \omega\left[a_{i-\beta,i-\beta+1} + a_{i-\beta,i-\beta+\alpha}\right]\right\}/\tilde{d}_{i-\beta}$
end for
for $i = n-\alpha+1, \cdots, n$ **do**
  $\tilde{d}_i = a_{ii} - a_{i-1,i}^2/\tilde{d}_{i-1}$
    $- a_{i-\alpha,i}\{a_{i-\alpha,i} + \omega a_{i-\alpha,i-\alpha+1}\}/\tilde{d}_{i-\alpha}$
    $- a_{i-\beta,i}\left\{a_{i-\beta,i} + \omega\left[a_{i-\beta,i-\beta+1} + a_{i-\beta,i-\beta+\alpha}\right]\right\}/\tilde{d}_{i-\beta}$
end for

Algorithm 7. Pivots for MIC$(0,\omega)$: trailing.

MIC$(0,\omega)$ preconditioning is effected by solving the equation $\tilde{U}^T\tilde{D}\tilde{U}x = b$; pseudocode for the forward elimination and back substitution, to solve this system using indirect factors, is presented in algorithms 8 and 9.

$b_1 := b_1/\tilde{d}_1$
for $i = 2, \cdots, \alpha$ **do**
  $b_i := \{b_i - a_{i-1,i} b_{i-1}\}/\tilde{d}_i$
end for
for $i = \alpha+1, \cdots, \beta$ **do**
  $b_i := \{b_i - [a_{i-1,i} b_{i-1} + a_{i-\alpha,i} b_{i-\alpha}]\}/\tilde{d}_i$
end for
for $i = \beta+1, \cdots, n$ **do**
  $b_i := \{b_i - [a_{i-1,i} b_{i-1} + a_{i-\alpha,i} b_{i-\alpha} + a_{i-\beta,i} b_{i-\beta}]\}/\tilde{d}_i$
end for

Algorithm 8. Forward elimination, MIC(0, ω).

$b_n := b_n$
for $i = n-1, \cdots, n-\alpha+1$ **do**
  $b_i := b_i - [a_{i,i+1} b_{i+1} + a_{i,i+\alpha} b_{i+\alpha}]/\tilde{d}_i$
end for
for $i = n-\alpha, \cdots, n-\beta+1$ **do**
  $b_i := b_i - [a_{i,i+1} b_{i+1} + a_{i,i+\alpha} b_{i+\alpha}]/\tilde{d}_i$
end for
for $i = n-\beta, \cdots, 1$ **do**
  $b_i := b_i - [a_{i,i+1} b_{i+1} + a_{i,i+\alpha} b_{i+\alpha} + a_{i,i+\beta} b_{i+\beta}]/\tilde{d}_i$
end for

Algorithm 9. Back substitution, MIC(0, ω).

Banded matrices are frequently stored in banded storage schemes; the following is a brief description of the scheme used in the PCGN solver. Allowing vectors D, X, Y, and Z to represent the diagonal, the first off diagonal, the second off diagonal, and the third off diagonal, respectively, of an SPS banded matrix, then band (i,i) corresponds to vector D, $(i,i+1)$ to X, $(i,i+\alpha)$ to Y, and $(i,i+\beta)$ to Z. Typically, these vectors would all have the same dimension, n, and the initial entry in each vector would correspond to the entry in row $i = 1$ of matrix A. Note that vector D cannot be replaced by the diagonal matrix $\tilde{D}$ holding the MIC(0, ω) pivots $\tilde{d}_i$, as D is needed elsewhere in the PCG algorithm. Thus, pivots $\tilde{D}$ must be stored in a separate vector; however, this is practically the only additional storage required in the MIC(0, ω) scheme. Use of a banded storage scheme to store the non-zero elements of A also simplifies the pseudocode for the factorization of the pivots in that the trailing algorithm 7 can be dispensed with. This follows because the entries in vectors X, Y, and Z corresponding to locations where index j of a_{ij} would exceed the $n \times n$ profile of A are simply set to zero. Thus, by replacing the end index, $n-\beta$, of the last **for** with n, in algorithm 6, the leading algorithm can be used to perform the complete factorization as the function of the trailing algorithm 7 is then performed implicitly.

Incomplete Cholesky Decomposition with 1 Fill

The incomplete Cholesky algorithm with 1 fill level (IC(1)) requires a criterion for picking the next level of infill. Here, it is helpful to consider the IC(0) factors given in equation 7. If the CCFD problem were homogeneous and isotropic, then all these factors would be more or less uniform and less than one. Under these conditions, the next largest contributions will be proportional to two-term products of $\tilde{u}_{ij}$; this is apparent from

the form of equation 4. Given this definition of the next-larger contributors, the IC(1) pattern S_1 is defined as follows:

$$S_1 = \left\{(i,j) \mid 1 \le i < j \le n \text{ and } \left[a_{ij} \neq 0 \text{ or } a_{\ell i} a_{\ell j} \neq 0 \text{ for arbitrary } \ell,\ 1 \le \ell < i\right]\right\}. \tag{14}$$

Given S_1, then identical to equations 6 and 7 the entries in the IC(1) decomposition, $\hat{U}^T \hat{D} \hat{U}$, can be defined as follows:

$$\hat{d}_i = a_{ii} - \sum_{\ell=1}^{i-1} \hat{d}_\ell \, \hat{u}_{\ell i}^2 \tag{15}$$

$$\hat{u}_{ij} = \begin{cases} \left[a_{ij} - \sum_{\ell=1}^{i-1} \hat{d}_\ell \, \hat{u}_{\ell i} \, \hat{u}_{\ell j}\right] \Big/ \hat{d}_i, & (i,j) \in S_1 \\ 0, & (i,j) \notin S_1. \end{cases} \tag{16}$$

For an SPS banded matrix, the products forming the additional entries have been identified previously in the discussion of the MIC$(0,\omega)$ pivots: $(\tilde{u}_{i-1,i}\tilde{u}_{i-1,i+\alpha-1})$, $(\tilde{u}_{i-1,i}\tilde{u}_{i-1,i+\beta-1})$, and $(\tilde{u}_{i-\alpha,i}\tilde{u}_{i-\alpha,i+\beta-\alpha})$. These results indicate that the additional IC(1) factors will occupy three diagonal bands: $(i, i+\alpha-1)$, $(i, i+\beta-1)$, and $(i, i+\beta-\alpha)$. The banded augmentation of IC(1) factors to the upper triangular matrix $\tilde{U}$ for the $4\times4\times4$ CCFD matrix is shown in figure 3. With reference to equation 16, it is seen that these factors $\hat{u}_{ij}$ are derived from the weighted inner product of two partial column vectors of $\hat{U}$; for columns i and j, where $j > i$ and $(i,j) \in S_1$, these vectors consist of the column entries $\hat{u}_{pi},\ p > i$ and $\hat{u}_{qj},\ q > i$. In figure 3, the left-most column vector is indicated with a dotted line, while column vectors corresponding to argumentation bands contained in $\hat{U}$ are depicted with dashed lines. For example, the argumentation factor $\hat{u}_{i,i+\alpha-1}$ consists of the weighted inner product of column vectors corresponding to the dotted and left-most dashed lines; this factor can be written as follows:

$$\hat{u}_{i,i+\alpha-1} = -\left[\hat{d}_{i-1}\,\hat{u}_{i-1,i}\,\hat{u}_{i-1,i+\alpha-1} + \hat{d}_{i-\beta+\alpha}\,\hat{u}_{i-\beta+\alpha,i}\,\hat{u}_{i-\beta+\alpha,i+\alpha-1}\right] \Big/ \hat{d}_i. \tag{17}$$

Similarly, the other argumentation factors may be written

$$\hat{u}_{i,i+\beta-\alpha} = -\left[\hat{d}_{i-\alpha+1}\,\hat{u}_{i-\alpha+1,i}\,\hat{u}_{i-\alpha+1,i+\beta-\alpha} + \hat{d}_{i-\alpha}\,\hat{u}_{i-\alpha,i}\,\hat{u}_{i-\alpha,i+\beta-\alpha}\right] \Big/ \hat{d}_i \tag{18}$$

and

$$\hat{u}_{i,i+\beta-1} = -\hat{d}_{i-1}\,\hat{u}_{i-1,i}\,\hat{u}_{i-1,i+\beta-1} \Big/ \hat{d}_i. \tag{19}$$

Note that, unlike the IC(0) factors, the derivation of these factors requires the inclusion of higher-order terms: $\hat{u}_{i-\beta+\alpha,i}\,\hat{u}_{i-\beta+\alpha,i+\alpha-1}$ and $\hat{u}_{i-\alpha+1,i}\,\hat{u}_{i-\alpha+1,i+\beta-\alpha}$. The factors $\hat{u}_{i,i+1}$ and $\hat{u}_{i,i+\alpha}$, corresponding to IC(0) factors $\tilde{u}_{i,i+1}$ and $\tilde{u}_{i,i+\alpha}$, are similarly augmented:

$$\hat{u}_{i,i+1} = \left\{a_{i,i+1} - \left[\hat{d}_{i-\alpha+1}\,\hat{u}_{i-\alpha+1,i}\,\hat{u}_{i-\alpha+1,i+1} + d_{i-\beta+1}\,\hat{u}_{i-\beta+1,i}\,\hat{u}_{i-\beta+1,i+1}\right]\right\} \Big/ \hat{d}_i. \tag{20}$$

$$\hat{u}_{i,i+\alpha} = \left[a_{i,i+\alpha} - \hat{d}_{i-\beta+\alpha}\,\hat{u}_{i-\beta+\alpha,i}\,\hat{u}_{i-\beta+\alpha,i+\alpha}\right] \Big/ \hat{d}_i. \tag{21}$$

The IC(1) factor $\hat{u}_{i,i+\beta}$ remains identical in form to $\tilde{u}_{i,i+\beta}$:

$$\hat{u}_{i,i+\beta} = a_{i,i+\beta}/\hat{d}_i. \tag{22}$$

Note that factors $\hat{u}_{i,i+1}$ and $\hat{u}_{i,i+\alpha}$, because of their augmented nature of equations 20 and 21 compared to equation 9, suggest that fill may occur in cases where the CCFD matrix entries $a_{i,i+1}$ and $a_{i,i+\alpha}$ are zero. However, the structure of the CCFD matrices, originating from the ordered numbering of a regular grid, does not allow for this fill to occur; thus, the structure of bands of the pattern matrix for $\hat{U}$ in figure 3 remain identical. The IC(1) pivots become

$$\begin{aligned}\hat{d}_i &= a_{ii} - \big[\hat{d}_{i-1}\,\hat{u}^2_{i-1,i} + \hat{d}_{i-\alpha}\,\hat{u}^2_{i-\alpha,i} + \hat{d}_{i-\beta}\,\hat{u}^2_{i-\beta,i} \\ &\quad + \hat{d}_{i-\alpha-1}\,\hat{u}^2_{i-\alpha-1,i} + \hat{d}_{i-\beta+\alpha}\,\hat{u}^2_{i-\beta+\alpha,i} + \hat{d}_{i-\beta-1}\,\hat{u}^2_{i-\beta-1,i}\big].\end{aligned} \tag{23}$$

Again, terms containing factors with non-positive indices or indices with values greater than n do not contribute.

With regard to an SPS banded matrix and the IC(1) factorization, the order of factors and terms contained therein can be related to the bands themselves. Bands $(i,i+1)$, $(i,i+\alpha)$, and $(i,i+\beta)$ contain one first-order term, equivalent to $\tilde{u}_{ij}$; factors $\hat{u}_{ij}$ contained in these bands are designated as $\mathscr{O}_1$. Bands $(i,i+\alpha-1)$, $(i,i+\beta-\alpha)$, and $(i,i+\beta-1)$ contain at least second-order terms, equivalent to two-term products of $\tilde{u}_{ij}$; factors $\hat{u}_{ij}$ contained in these bands are designated as $\mathscr{O}_2$. In addition, there exist terms in $\hat{U}$ that are products of $\mathscr{O}_1$ and $\mathscr{O}_2$ factors; these factors are designated as $\mathscr{O}_3$. Products such as $\hat{u}_{i-\alpha+1,i}\,\hat{u}_{i-\alpha+1,i+1}$, $\hat{u}_{i-\beta+1,i}\,\hat{u}_{i-\beta+1,i+1}$, and $\hat{u}_{i-\beta+\alpha,i}\,\hat{u}_{i-\beta+\alpha,i+\alpha}$ are examples of such terms. Finally, terms also exist that are products of $\mathscr{O}_2$ factors; these factors are designated as $\mathscr{O}_4$. Products such as $\hat{u}_{i-\beta+\alpha,i}\,\hat{u}_{i-\beta+\alpha,i+\alpha-1}$ and $\hat{u}_{i-\alpha+1,i}\,\hat{u}_{i-\alpha+1,i+\beta-\alpha}$, mentioned previously in association with equations 17 and 18, are examples of $\mathscr{O}_4$ terms.

Per the development in equation 12, the modified incomplete Cholesky with fill 1 (MIC(1, ω)) can be written as follows:

$$\hat{d}_i := \hat{d}_i - \omega\left[\sum_{k=1}^{i-1}\bar{u}_{ki} + \sum_{k=i+1}^{n}\bar{u}_{ik}\right], \tag{24}$$

where now

$$\bar{u}_{ik} = \begin{cases}\sum_{\ell=1}^{i-1}\hat{d}_\ell\hat{u}_{\ell i}\hat{u}_{\ell k}, & (i,k)\notin S_1 \\ 0, & (i,k)\in S_1\end{cases} \tag{25}$$

and

$$\bar{u}_{ki} = \begin{cases}\sum_{\ell=1}^{i-1}\hat{d}_\ell\hat{u}_{\ell k}\hat{u}_{\ell i}, & (k,i)\notin S_1 \\ 0, & (k,i)\in S_1.\end{cases} \tag{26}$$

When applied to SPS banded matrices, the product pairs forming $\bar{u}_{ik}$ are

$\hat{u}_{i-1,i}\,\hat{u}_{i-1,i+\alpha-2}$,
$\hat{u}_{i-1,i}\,\hat{u}_{i-1,i+\beta-\alpha-1}$,
$\hat{u}_{i-1,i}\,\hat{u}_{i-1,i+\beta-2}$,
$\hat{u}_{i-\alpha+1,i}\,\hat{u}_{i-\alpha+1,i+\beta-2\alpha+1}$,
$\hat{u}_{i-\alpha+1,i}\,\hat{u}_{i-\alpha+1,i+\beta-\alpha+1}$,
$\hat{u}_{i-\alpha,i}\,\hat{u}_{i-\alpha,i+\beta-2\alpha}$, and
$\hat{u}_{i-\alpha,i}\,\hat{u}_{i-\alpha,i+\beta-\alpha-1}$.

Similarly, the product pairs forming $\bar{u}_{ki}$ are

$\hat{u}_{i-\alpha+1,i-\alpha+2}\,\hat{u}_{i-\alpha+1,i}$,
$\hat{u}_{i-\beta+\alpha,i-\beta+\alpha+1}\,\hat{u}_{i-\beta+\alpha,i}$,
$\hat{u}_{i-\beta+\alpha,i-\beta+2\alpha}\,\hat{u}_{i-\beta+\alpha,i}$,
$\hat{u}_{i-\beta+\alpha,i-\beta+2\alpha-1}\,\hat{u}_{i-\beta+\alpha,i}$,
$\hat{u}_{i-\beta+1,i-\beta+2}\,\hat{u}_{i-\beta+1,i}$,
$\hat{u}_{i-\beta+1,i-\beta+\alpha+1}\,\hat{u}_{i-\beta+1,i}$, and
$\hat{u}_{i-\beta,i-\beta+\alpha-1}\,\hat{u}_{i-\beta,i}$.

However, while most of these products would introduce terms of order $\mathscr{O}_3$, products $\hat{u}_{i-\alpha+1,i}\,\hat{u}_{i-\alpha+1,i+\beta-2\alpha+1}$ and $\hat{u}_{i-\beta+\alpha,i-\beta+2\alpha-1}\,\hat{u}_{i-\beta+\alpha,i}$ are of order $\mathscr{O}_4$. Experimentation shows that including these $\mathscr{O}_4$ terms in the pivots $\hat{d}_i$ of MIC$(1,\omega)$ causes a small degradation in the performance of the PCG solver; thus, these terms were not included as part of the modification used in PCGN. In its final form, modified pivots take the form:

$$\begin{aligned}
\hat{d}_i \;&= a_{ii} - \big\{\hat{d}_{i-1}\,\hat{u}_{i-1,i}^2 + \omega\big[\hat{d}_{i-1}\,\hat{u}_{i-1,i}\,\hat{u}_{i-1,i+\alpha-2} \\
&\quad + \hat{d}_{i-1}\,\hat{u}_{i-1,i}\,\hat{u}_{i-1,i+\beta-\alpha-1} + \hat{d}_{i-1}\,\hat{u}_{i-1,i}\,\hat{u}_{i-1,i+\beta-2}\big] \\
&\quad + \hat{d}_{i-\alpha+1}\,\hat{u}_{i-\alpha+1,i}^2 + \omega\big[\hat{d}_{i-\alpha+1}\,\hat{u}_{i-\alpha+1,i}\,\hat{u}_{i-\alpha+1,i+\beta-\alpha+1} \\
&\quad + \hat{d}_{i-\alpha+1}\,\hat{u}_{i-\alpha+1,i-\alpha+2}\,\hat{u}_{i-\alpha+1,i}\big] \\
&\quad + \hat{d}_{i-\alpha}\,\hat{u}_{i-\alpha,i}^2 + \omega\big[\hat{d}_{i-\alpha}\,\hat{u}_{i-\alpha,i}\,\hat{u}_{i-\alpha,i+\beta-2\alpha} + \hat{d}_{i-\alpha}\,\hat{u}_{i-\alpha,i}\,\hat{u}_{i-\alpha,i+\beta-\alpha-1}\big] \\
&\quad + \hat{d}_{i-\beta+\alpha}\,\hat{u}_{i-\beta+\alpha,i}^2 + \omega\big[\hat{d}_{i-\beta+\alpha}\,\hat{u}_{i-\beta+\alpha,i-\beta+\alpha+1}\,\hat{u}_{i-\beta+\alpha,i} \\
&\quad + \hat{d}_{i-\beta+\alpha}\,\hat{u}_{i-\beta+\alpha,i-\beta+\alpha+2\alpha}\,\hat{u}_{i-\beta+\alpha,i}\big] \\
&\quad + \hat{d}_{i-\beta+1}\,\hat{u}_{i-\beta+1,i}^2 + \omega\big[\hat{d}_{i-\beta+1}\,\hat{u}_{i-\beta+1,i-\beta+2}\,\hat{u}_{i-\beta+1,i} \\
&\quad + \hat{d}_{i-\beta+1}\,\hat{u}_{i-\beta+1,i-\beta+\alpha+1}\,\hat{u}_{i-\beta+1,i}\big] \\
&\quad + \hat{d}_{i-\beta}\,\hat{u}_{i-\beta,i}^2 + \omega\hat{d}_{i-\beta}\,\hat{u}_{i-\beta,i-\beta+\alpha-1}\,\hat{u}_{i-\beta,i}\big\}. \qquad (27)
\end{aligned}$$

As in the case of equation 13, this equation can be rationalized as follows:

$$\begin{aligned}
\hat{d}_i \;&= a_{ii} - \hat{d}_{i-1}\,\hat{u}_{i-1,i}\left\{\hat{u}_{i-1,i} + \omega\left[\hat{u}_{i-1,i+\alpha-2} + \hat{u}_{i-1,i+\beta-\alpha-1} + \hat{u}_{i-1,i+\beta-2}\right]\right\} \\
&\quad - \hat{d}_{i-\alpha+1}\,\hat{u}_{i-\alpha+1,i}\left\{\hat{u}_{i-\alpha+1,i} + \omega\left[\hat{u}_{i-\alpha+1,i+\beta-\alpha+1} + \hat{u}_{i-\alpha+1,i-\alpha+2}\right]\right\} \\
&\quad - \hat{d}_{i-\alpha}\,\hat{u}_{i-\alpha,i}\left\{\hat{u}_{i-\alpha,i} + \omega\left[\hat{u}_{i-\alpha,i+\beta-2\alpha} + \hat{u}_{i-\alpha,i+\beta-\alpha-1}\right]\right\} \\
&\quad - \hat{d}_{i-\beta+\alpha}\,\hat{u}_{i-\beta+\alpha,i}\left\{\hat{u}_{i-\beta+\alpha,i} + \omega\left[\hat{u}_{i-\beta+\alpha,i-\beta+\alpha+1} + \hat{u}_{i-\beta+\alpha,i-\beta+2\alpha}\right]\right\} \\
&\quad - \hat{d}_{i-\beta+1}\,\hat{u}_{i-\beta+1,i}\left\{\hat{u}_{i-\beta+1,i} + \omega\left[\hat{u}_{i-\beta+1,i-\beta+2} + \hat{u}_{i-\beta+1,i-\beta+\alpha+1}\right]\right\} \\
&\quad - \hat{d}_{i-\beta}\,\hat{u}_{i-\beta,i}\left\{\hat{u}_{i-\beta,i} + \omega\hat{u}_{i-\beta,i-\beta+\alpha-1}\right\}. \qquad (28)
\end{aligned}$$

As noted previously, terms containing factors with non-positive indices or indices with values greater than n do not contribute. It should be noted that this form of the MIC$(1,\omega)$ pivots likely does not comply with the row-sum requirement suggested in van der Vorst (2003) for a modified incomplete Cholesky decomposition. While pivots (eq. 28) appear to function effectively, more research could possibly produce a form compliant with the row-sum requirement and possibly produce a more effective modification.

$s_{1,2} = a_{1,2}$
$s_{1,1+\alpha} = a_{1,1+\alpha}$
$s_{1,1+\beta} \equiv a_{1,1+\beta}$
$\hat{d}_1 = a_{11}$
for $i = 2, \cdots, \alpha$ **do**
$s_{i,i+1} = a_{i,i+1}$
$s_{i,i+\alpha} = a_{i,i+\alpha}$
$s_{i,i+\beta} \equiv a_{i,i+\beta}$
$s_{i,i+\alpha-1} = -s_{i-1,i}\, s_{i-1,i+\alpha-1} / \hat{d}_{i-1}$
$s_{i,i+\beta-1} = -s_{i-1,i}\, s_{i-1,i+\beta-1} / \hat{d}_{i-1}$
$\hat{d}_i = a_{ii} - s_{i-1,i} \left\{ s_{i-1,i} + \omega \left[s_{i-1,i+\alpha-2} + s_{i-1,i+\beta-2} \right] \right\} / \hat{d}_{i-1}$
end for
for $i = \alpha+1, \cdots, \beta$ **do**
$s_{i,i+1} = a_{i,i+1} - s_{i-\alpha+1,i}\, s_{i-\alpha+1,i+1} / \hat{d}_{i-\alpha+1}$
$s_{i,i+\alpha} = a_{i,i+\alpha}$
$s_{i,i+\beta} \equiv a_{i,i+\beta}$
$s_{i,i+\alpha-1} = -s_{i-1,i}\, s_{i-1,i+\alpha-1} / \hat{d}_{i-1}$
$s_{i,i+\beta-\alpha} = -s_{i-\alpha+1,i}\, s_{i-\alpha+1,i+\beta-\alpha} / \hat{d}_{i-\alpha+1} - s_{i-\alpha,i}\, s_{i-\alpha,i+\beta-\alpha} / \hat{d}_{i-\alpha}$
$s_{i,i+\beta-1} = -s_{i-1,i}\, s_{i-1,i+\beta-1} / \hat{d}_{i-1}$
$\hat{d}_i = a_{ii} - s_{i-1,i} \left\{ s_{i-1,i} + \omega \left[s_{i-1,i+\alpha-2} + s_{i-1,i+\beta-\alpha-1} + s_{i-1,i+\beta-2} \right] \right\} / \hat{d}_{i-1}$
$- s_{i-\alpha+1,i} \left\{ s_{i-\alpha+1,i} + \omega \left[s_{i-\alpha+1,i+\beta-\alpha+1} + s_{i-\alpha+1,i-\alpha+2} \right] \right\} / \hat{d}_{i-\alpha+1}$
$- s_{i-\alpha,i} \left\{ s_{i-\alpha,i} + \omega \left[s_{i-\alpha,i+\beta-2\alpha} + s_{i-\alpha,i+\beta-\alpha-1} \right] \right\} / \hat{d}_{i-\alpha}$
end for
for $i = \beta+1, \cdots, n-\beta$ **do**
$s_{i,i+1} = a_{i,i+1} - s_{i-\alpha+1,i}\, s_{i-\alpha+1,i+1} / \hat{d}_{i-\alpha+1} - s_{i-\beta+1,i}\, s_{i-\beta+1,i+1} / \hat{d}_{i-\beta+1}$
$s_{i,i+\alpha} = a_{i,i+\alpha} - s_{i-\beta+\alpha,i}\, \hat{s}_{i-\beta+\alpha,i+\alpha} / \hat{d}_{i-\beta+\alpha}$
$s_{i,i+\beta} \equiv a_{i,i+\beta}$
$s_{i,i+\alpha-1} = -s_{i-1,i}\, s_{i-1,i+\alpha-1} / \hat{d}_{i-1} - s_{i-\beta+\alpha,i}\, s_{i-\beta+\alpha,i+\alpha-1} / \hat{d}_{i-\beta+\alpha}$
$s_{i,i+\beta-\alpha} = -s_{i-\alpha+1,i}\, s_{i-\alpha+1,i+\beta-\alpha} / \hat{d}_{i-\alpha+1} - s_{i-\alpha,i}\, s_{i-\alpha,i+\beta-\alpha} / \hat{d}_{i-\alpha}$
$s_{i,i+\beta-1} = -s_{i-1,i}\, s_{i-1,i+\beta-1} / \hat{d}_{i-1}$
$\hat{d}_i = a_{ii} - s_{i-1,i} \left\{ s_{i-1,i} + \omega \left[s_{i-1,i+\alpha-2} + s_{i-1,i+\beta-\alpha-1} + s_{i-1,i+\beta-2} \right] \right\} / \hat{d}_{i-1}$
$- s_{i-\alpha+1,i} \left\{ s_{i-\alpha+1,i} + \omega \left[s_{i-\alpha+1,i+\beta-\alpha+1} + s_{i-\alpha+1,i-\alpha+2} \right] \right\} / \hat{d}_{i-\alpha+1}$
$- s_{i-\alpha,i} \left\{ s_{i-\alpha,i} + \omega \left[s_{i-\alpha,i+\beta-2\alpha} + s_{i-\alpha,i+\beta-\alpha-1} \right] \right\} / \hat{d}_{i-\alpha}$
$- s_{i-\beta+\alpha,i} \left\{ s_{i-\beta+\alpha,i} + \omega \left[s_{i-\beta+\alpha,i-\beta+\alpha+1} + s_{i-\beta+\alpha,i-\beta+2\alpha} \right] \right\} \hat{d}_{i-\beta+\alpha}$
$- s_{i-\beta+1,i} \left\{ s_{i-\beta+1,i} + \omega \left[s_{i-\beta+1,i-\beta+2} + s_{i-\beta+1,i-\beta+\alpha+1} \right] \right\} / \hat{d}_{i-\beta+1}$
$- s_{i-\beta,i} \left\{ s_{i-\beta,i} + \omega\, s_{i-\beta,i-\beta+\alpha-1} \right\} / \hat{d}_{i-\beta}$
end for

Algorithm 10. Pivots and indirect factors for MIC(1, ω): leading.

The storage requirements for the MIC(1, ω) preconditioner are substantially greater than those for the MIC(0, ω) preconditioner, as not all the MIC(1, ω) factors can be subsumed into the storage scheme for the CCFD matrix. However, factor $\hat{u}_{i,i+\beta}$ (eq. 22) has the same property as the MIC(0, ω) factors: $\hat{u}_{i,i+\beta} = a_{i,i+\beta} / \hat{d}_i$. In order to garner some storage savings from this one off-diagonal band, equations 16 – 22 are expressed as "indirect" factors $s_{ij} = \hat{d}_i \hat{u}_{ij}$. Pseudocode for formation of indirect factors s_{ij} and pivots $\hat{d}_{ij}$ is presented in algorithms 10 and 11. Again, the major development is contained in the leading algorithm, while the trailing algorithm ensures the absence of indirect factors $s_{ij},\ i > n-\beta,\ j > n$ outside of the $n \times n$ profile of the matrix.

for $i = n-\beta+1, \cdots, n-\alpha$ **do**
$s_{i,i+1} = a_{i,i+1} - s_{i-\alpha+1,i}\, s_{i-\alpha+1,i+1}/\hat{d}_{i-\alpha+1} - s_{i-\beta+1,i}\, s_{i-\beta+1,i+1}/\hat{d}_{i-\beta+1}$
$s_{i,i+\alpha} = a_{i,i+\alpha} - s_{i-\beta+\alpha,i}\, \hat{s}_{i-\beta+\alpha,i+\alpha}/\hat{d}_{i-\beta+\alpha}$
$s_{i,i+\alpha-1} = -s_{i-1,i}\, s_{i-1,i+\alpha-1}/\hat{d}_{i-1} - s_{i-\beta+\alpha,i}\, s_{i-\beta+\alpha,i+\alpha-1}/\hat{d}_{i-\beta+\alpha}$
$\hat{d}_i = a_{ii} - s_{i-1,i}\{s_{i-1,i} + \omega\, s_{i-1,i+\alpha-2}\}/\hat{d}_{i-1}$
$\quad - s_{i-\alpha+1,i}\{s_{i-\alpha+1,i} + \omega\, s_{i-\alpha+1,i-\alpha+2}\}/\hat{d}_{i-\alpha+1} - s^2_{i-\alpha,i}/\hat{d}_{i-\alpha}$
$\quad - s_{i-\beta+\alpha,i}\left\{s_{i-\beta+\alpha,i} + \omega\left[s_{i-\beta+\alpha,i-\beta+\alpha+1} + s_{i-\beta+\alpha,i-\beta+2\alpha}\right]\right\}\hat{d}_{i-\beta+\alpha}$
$\quad - s_{i-\beta+1,i}\left\{s_{i-\beta+1,i} + \omega\left[s_{i-\beta+1,i-\beta+2} + s_{i-\beta+1,i-\beta+\alpha+1}\right]\right\}/\hat{d}_{i-\beta+1}$
$\quad - s_{i-\beta,i}\left\{s_{i-\beta,i} + \omega\, s_{i-\beta,i-\beta+\alpha-1}\right\}/\hat{d}_{i-\beta}$
end for
for $i = n-\alpha+1, \cdots, n$ **do**
$s_{i,i+1} = a_{i,i+1} - s_{i-\alpha+1,i}\, s_{i-\alpha+1,i+1}/\hat{d}_{i-\alpha+1} - s_{i-\beta+1,i}\, s_{i-\beta+1,i+1}/\hat{d}_{i-\beta+1}$
$\hat{d}_i = a_{ii} - s^2_{i-1,i}/\hat{d}_{i-1} - s^2_{i-\alpha,i}/\hat{d}_{i-\alpha}$
$\quad - s_{i-\alpha+1,i}\{s_{i-\alpha+1,i} + \omega\, s_{i-\alpha+1,i-\alpha+2}\}/\hat{d}_{i-\alpha+1}$
$\quad - s_{i-\beta+\alpha,i}\left\{s_{i-\beta+\alpha,i} + \omega\left[s_{i-\beta+\alpha,i-\beta+\alpha+1} + s_{i-\beta+\alpha,i-\beta+2\alpha}\right]\right\}\hat{d}_{i-\beta+\alpha}$
$\quad - s_{i-\beta+1,i}\left\{s_{i-\beta+1,i} + \omega\left[s_{i-\beta+1,i-\beta+2} + s_{i-\beta+1,i-\beta+\alpha+1}\right]\right\}/\hat{d}_{i-\beta+1}$
$\quad - s_{i-\beta,i}\left\{s_{i-\beta,i} + \omega\, s_{i-\beta,i-\beta+\alpha-1}\right\}/\hat{d}_{i-\beta}$
end for

Algorithm 11. Pivots and indirect factors for MIC(1, ω): trailing.

MIC(1, ω) preconditioning is effected by solving the equation $\hat{U}^T\hat{D}\hat{U}x = b$; pseudocode for the forward elimination and back substitution, to solve this system using indirect factors, is presented in algorithms 12 and 13.

$b_1 := b_1/\hat{d}_1$
for $i = 2, \cdots, \alpha$ **do**
$b_i := \{b_i - s_{i-1,i}b_{i-1}\}/\hat{d}_i$
end for
for $i = \alpha+1, \cdots, \beta$ **do**
$b_i := \{b_i - [s_{i-1,i}b_{i-1} + s_{i-\alpha+1,i}b_{i-\alpha+1} + s_{i-\alpha,i}b_{i-\alpha}]\}/\hat{d}_i$
end for
for $i = \beta+1, \cdots, n$ **do**
$b_i := \{b_i - [s_{i-1,i}b_{i-1} + s_{i-\alpha+1,i}b_{i-\alpha+1} + s_{i-\alpha,i}b_{i-\alpha}$
$\quad + s_{i-\beta+\alpha,i}b_{i-\beta+\alpha} + s_{i-\beta+1,i}b_{i-\beta+1} + s_{i-\beta,i}b_{i-\beta}]\}/\hat{d}_i$
end for

Algorithm 12. Forward elimination, MIC(1, ω).

$b_n := b_n$
for $i = n-1, \cdots, n-\alpha+1$ **do**
  $b_i := b_i - s_{i,i+1} b_{i+1} / \hat{d}_i$
end for
for $i = n-\alpha, \cdots, n-\beta+1$ **do**
  $b_i := b_i - [s_{i,i+1} b_{i+1} + s_{i,i+\alpha-1} b_{i+\alpha-1} + b_{i,i+\alpha} b_{i+\alpha}] / \hat{d}_i$
end for
for $i = n-\beta, \cdots, 1$ **do**
  $b_i := b_i - [s_{i,i+1} b_{i+1} + s_{i,i+\alpha-1} b_{i+\alpha-1} + s_{i,i+\alpha} b_{i+\alpha}$
    $+ s_{i,i+\beta-\alpha} b_{i+\beta-\alpha} + s_{i,i+\beta-1} b_{i+\beta-1} + s_{i,i+\beta} b_{i+\beta}] / \hat{d}_i$
end for

Algorithm 13. Back substitution, MIC(1, ω).

If the banded matrix is stored in the banded matrix storage scheme detailed previously, then vectors D, X, Y, and Z, each of length n, would contain the matrix. These vectors are necessary to the PCG algorithm and cannot be overwritten. New vectors are needed to hold the indirect factors s_{ij} created in the MIC(1, ω) factorization; herein these are referred to as X_0, Y_0, Z_0, X_1, Y_1, and Z_1. The vectors X_0, Y_0, and Z_0 will contain the $\mathscr{O}_1$ factors, while X_1, Y_1, and Z_1 will contain the $\mathscr{O}_2$ factors. In terms of matrix $\hat{U}$, diagonal $(i, i+1)$ corresponds to X_0, $(i, i+\alpha)$ to Y_0, $(i, i+\beta)$ to Z_0, $(i, i+\alpha-1)$ to X_1, $(i, i+\beta-\alpha)$ to Y_1, and $(i, i+\beta-1)$ to Z_1. Of course, Z_0 is actually set equivalent to Z as it is unchanged by the factorization. Moreover, the diagonal matrix $\hat{D}$ of pivots is stored separately from D and thus requires another vector of length n. As these new vectors are all of length n, it is seen that the memory storage requirements of the PCG solver with the MIC(1, ω) factorization is essentially twice that of the PCG solver with the MIC(0, ω) factorization. Again, with this banded storage scheme, the pseudocode for the factorization is somewhat simplified in that only the leading algorithm 10, with the end index of the last **for**, $n-\beta$, replaced with n need be used for the complete factorization. Those entries in vectors X_0, Y_0, X_1, Y_1, and Z_1 that correspond to locations where j exceeds the $n \times n$ profile of A are simply set to zero.

Test Results for Modified Incomplete Cholesky Preconditioning

To give the user an appreciation of the effect of the fill number and relaxation parameter ω on a linear problem, a pseudo-CCFD coefficient matrix was generated that could be easily altered to produce varying degrees of anisotropy. The solution of the resulting matrix equations occurred entirely outside MODFLOW; the study was conceived as a way to directly investigate the PCG solver that resides within the PCGN package. Using hydraulic conductivities obtained from a [0,1] uniform random variable generator, CCFD-like coefficients were constructed in a manner analogous to the algorithm used in MODFLOW. For the x direction, these coefficients were multiplied by a^2, where a is the anisotropy multiplier; for the y direction, they were multiplied by a^1; and for the z direction, they were multiplied by a^0. A right hand side for the matrix equations was generated by multiplying these matrices with a vector of random numbers; the vector of random numbers constituted an exact solution for comparison purposes. The anisotropy multiplier was allowed to vary from 1 (isotropic) to 10 (very anisotropic); the resulting matrices were solved with PCG using a MIC fill level of 0 or 1 and $\omega = 0.0$ or $\omega = 0.99$. The generated matrices had size n =200,000; the number of iterations to solve each case to a specified absolute convergence (0.01) and the time spent by the solver in iteration mode was recorded. The iteration results are presented in figure 4. The number of iterations required for a specified convergence can be looked upon as the degree of conditioning that the combination of fill level and ω imparts

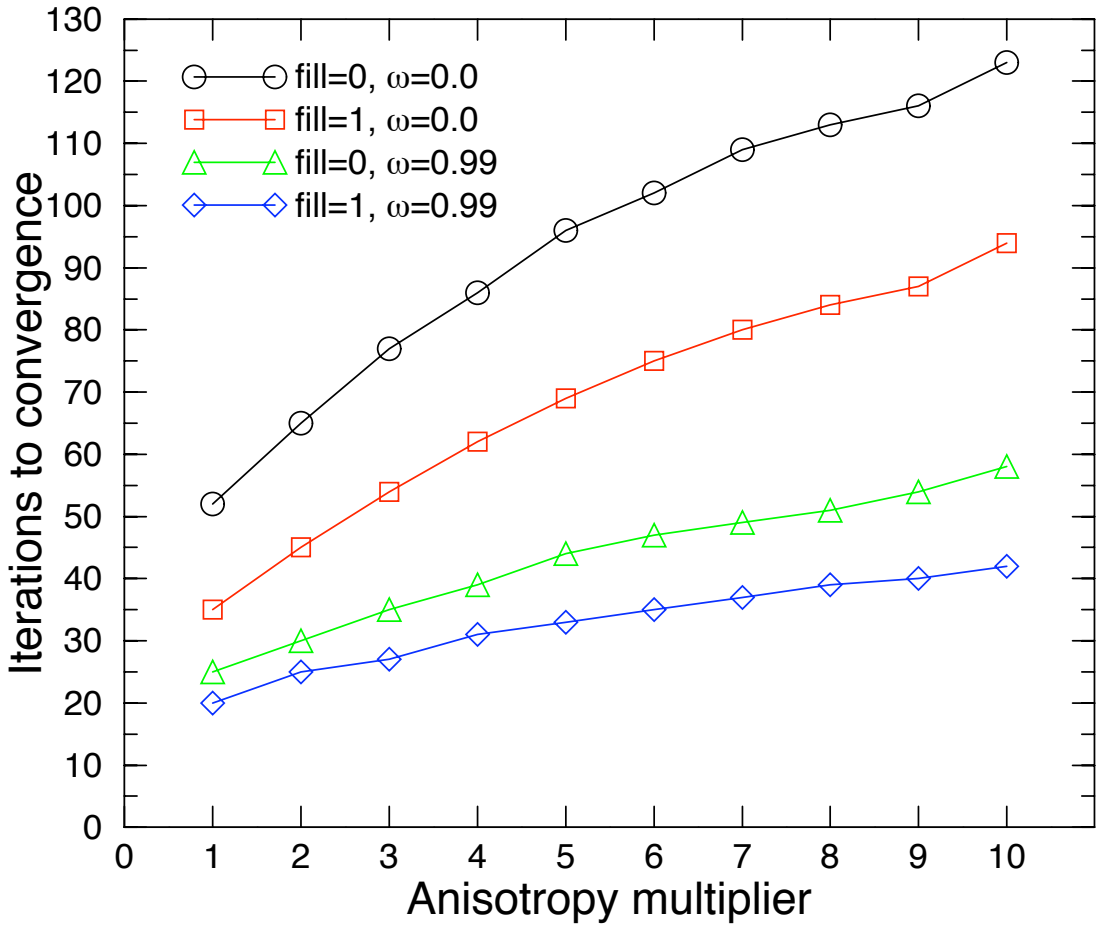


Figure 4. Iterations required to obtain a specified convergence as a function of anistropy; see text for explanation of anisotropy multiplier.

to each level of anisotropy. Clearly, a relaxation factor such that $\omega = 0.99$, imparting almost full modification to the incomplete Cholesky algorithm, is superior to $\omega = 0.0$, where no modification is allowed. When the fill level is 0, selecting $\omega = 1$ results in a serious degradation in the performance of the PCG solver for these problems, particularly at higher values of the anisotropy multiplier; this value of ω should in general be avoided. For $\omega = 0.99$, the ratio of iterations for fill level 0 to fill level 1 varies from 1.2 ($a = 2$) to 1.38 ($a = 10$). Clearly, a fill level 1 exhibits superior conditioning of the test matrix. However, for this particular problem, processor and compiler combination, the PCG solver for a fill level 0 solved at a rate of 37 iterations/sec while the fill level 1 solver rate was 29 iterations/sec; the ratio for the two rates (0 to 1) is 1.28. Thus, for an isotropic problem, a fill level 0 should produce a superior run time, while a fill level 1 run time should be superior for highly anisotropic problems. Anisotropic problems are generally more poorly conditioned than isotropic problems; it is not surprising that these problems should respond better to the additional conditioning provided by the fill level 1 preconditioner.

References Cited

Axelsson, O., 1996, Iterative solution methods: Cambridge, U.K., Cambridge University Press, 654 p.

Banta, E., 2006, Modifications to MODFLOW boundary conditions and an adaptive-damping scheme for Picard iterations for a highly nonlinear regional model, *in* Managing ground water systems—MODFLOW and More 2006, Golden, Colo., May 21–24, 2006, Proceedings: Golden, Colo., Colorado School of Mines, International Ground Water Modeling Center, p. 596–600.

Barrett, R., Berry M., Chan, T., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., and van der Vorst, H., 1994, Templates for the solution of linear systems—Building blocks for iterative methods 2d ed.: Philadelphia, Society for Industrial and Applied Mathematics, 112 p.

Golub, G., and Van Loan, C., 1983, Matrix computations: Baltimore, John Hopkins University Press, 476 p.

Harbaugh, A., Banta, E., Hill, M., and McDonald, M., 2000, Modflow-2000, the U.S. Geological Survey modular ground-water model—User guide to modularization concepts and the ground-water flow process: U.S. Geological Survey Open-File Report 00–92, 121 p.

Hill, M., 1990, Preconditioned conjugate-gradient 2 (PCG2)—A computer program for solving ground-water flow equations: U.S. Geological Survey Water-Resources Investigations Report 90–4048, 25 p.

Mehl, S., 2006, Use of Picard and Newton iteration in solving nonlinear ground water flow equations: Ground Water, v. 44, no. 4, p. 583–594.

Mehl, S., and Hill, M., 2001, Modflow-2000, the U.S. Geological Survey modular ground-water model—User guide to the Link-AMG (LMG) package for solving matrix equations using an algebraic multigrid solver: U.S. Geological Survey Open-File Report 01–177, 33 p.

Robson, S., 1987, Bedrock aquifers in the Denver Basin, Colorado—A quantitative water-resources appraisal: U.S. Geological Survey Professional Paper 1257, 73 p., 5 plates.

Saad, Y., 2003, Iterative methods for sparse linear systems: Philadelphia, Society for Industrial and Applied Mathematics, 528 p.

van der Vorst, H., 2003, Iterative Krylov methods for large linear systems: Cambridge, U.K., Cambridge University Press, 221 p.

Wilson, J. and Naff, R., 2004, Modflow-2000, the U.S. Geological Survey modular ground-water model—GMG linear equation solver package documentation: U.S. Geological Survey Open-File Report 2004–1261, 47 p.

Publishing support provided by:
Denver Publishing Service Center

For more information concerning this publication, contact:
Chief, Branch of Regional Research, Central Region
Box 25046, Mail Stop 418
Denver, CO 80225
(303)236-5021

Or visit the USGS National Research Program Web site at:
http:/water.usgs.gov/nrp