2

NPS-53-89-006

NAVAL POSTGRADUATE SCHOOL

Monterey, California

COMPARISON OF SEVERAL ITERATIVE TECHNIQUES IN THE SOLUTION OF SYMMETRIC BANDED EQUATIONS ON A TWO-PIPE CYBER 205

Clyde Scandrett

November 1988

Prepared for: Naval Postgraduate School Research Council,
U.S. Department of Energy, Washington DC

NAVAL POSTGRADUATE SCHOOL
Department of Mathematics

Rear Admiral R. C. Austin Superintendent	Harrison Shull Provost

This report was prepared in conjunction with research conducted for the U.S. Department of Energy and for the Naval Postgraduate School Research Council and funded by the Naval Postgraduate School Research Council. Reproduction of all or part of this report is authorized.

Prepared by:

[illegible]
CLYDE SCANDRETT
Assistant Professor of
Mathematics

Reviewed by:

[illegible]
HAROLD M. FREDRICKSEN
Chairman
Department of Mathematics

Released by:

[illegible]
KNEALE T. MARSHALL
Dean of Information and
Policy Sciences

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) NPS-53-89-006	5 MONITORING ORGANIZATION REPORT NUMBER(S) NPS-53-89-006
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b OFFICE SYMBOL (If applicable) 53	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School and the U.S. Department of Energy
6c ADDRESS (City, State, and ZIP Code) Monterey, CA 93943	7b ADDRESS (City, State, and ZIP Code) Monterey, CA 9394 and Washington, D.C.
8a NAME OF FUNDING/SPONSORING ORGANIZATION Naval Postgraduate School 8b OFFICE SYMBOL (If applicable) 53	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER 02MN, Direct Funds
8c ADDRESS (City, State, and ZIP Code) Monterey, CA 93943	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO. / WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)

Comparison of Several Iterative Techniques in the Solution of Symmetric Banded Equations on A Two-Pipe Cyber 205

12 PERSONAL AUTHOR(S)

Clyde Scandrett → Gradient Method)

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Technical Report	FROM 10/1/87 TO 9/30/88	November 1988	

16 SUPPLEMENTARY NOTATION

Modified Incomplete Cholesky Conjugate (Polynomial Preconditioned Conjugate Gradient Method)

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	symmetric banded matrices, iterative matrix solvers, conjugate gradient methods, Cyber 205 vector processor

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

The effectiveness of several iterative techniques for solving matrix equations resulting from finite difference approximations to self-adjoint parabolic and elliptic partial differential equations is reviewed. The techniques include Stone's Strongly Implicit Procedure (SIP) and several conjugate gradient algorithms with varying preconditioners. The comparison is made on a vector machine (two-pipe Cyber 205) where vectorization of the code is done primarily by the vector compiler available. It is found that of the methods studied, POLCG and MICCG appear to require the least amount of ~~CPU~~ time. An advantage of MICCG and POLCG is that it is less sensitive to increasing matrix size. Its disadvantages are that it requires an iteration parameter, has a greater set-up time, and needs more storage than POLCG. (KR)

Vector processing

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL Clyde Scandrett	22b TELEPHONE (Include Area Code) (408) 646-2027 22c. OFFICE SYMBOL 53Sd

ABSTRACT

The effectiveness of several iterative techniques for solving matrix equations resulting from finite difference approximations to self-adjoint parabolic and elliptic partial differential equations is reviewed. The techniques are:

1. SIP- Strongly Implicit Procedure
2. ICCG- Incomplete Cholesky Conjugate Gradient Method
3. VICCG- Vectorized ICCG
4. MICCG- Modified ICCG
5. DSCG- Diagonally Scaled Conjugate Gradient Method
6. POLCG- Polynomial Preconditioned Conjugate Gradient Method.
7. PICCG- Polynomial form of ICCG

The comparison is made on a vector machine (two-pipe Cyber 205) where vectorization of the code is done primarily by the vector compiler available. It is found that of the methods studied, POLCG and MICCG appear to require the least amount of CPU time. An advantage of MICCG over POLCG is that it is less sensitive to increasing matrix size. Its disadvantages are that it requires an iteration parameter, has a greater set-up time, and needs more storage than POLCG.

1. Introduction

In the numerical solution of two and three dimensional partial differential equations one is often led to a system of linear equations (or a matrix equation) which must be solved. Efficient performance of this task by the researcher has become complicated with the advent of vector and parallel processing machines. Depending upon the algorithm, machine, and problem to be solved, an "optimum" strategy taken by the researcher should take into consideration properties of various algorithms such as their vectorizability and parallelizability.

The present work is an effort to study the efficiency of several iterative methods used in the solution of symmetric banded systems of linear equations. They are: Stone's[1] Strongly Implicit Procedure(SIP); the Incomplete Cholesky Conjugate Gradient Method (ICCG) of Meijerink and van der Vorst[2], including its vectorized variant (VICCG(0))[3], and Gustafsson's[4] modified form (MICCG); a diagonally scaled conjugate gradient method (DSCG); Saad's[5] polynomial preconditioned conjugate gradient method (POLCG); and finally a polynomial variation of ICCG (PICCG). The machine on which comparisons are made is a Cyber 205 which has a peak operating performance of 100 megaflops for vector addition and/or multiplication (200 megaflops for linked triadic vector operations).

New algorithms are not introduced (although some modifications to existing methods are tried). The purpose of this effort is to take a series of known methods, apply them to problems with similar characteristics to what might be found in real physical settings (including three dimensional problems of groundwater modelling), and determine which method(s) reviewed have superior performance on a Cyber 205.

Little has been written in the water resource journals on many of the now available vectorizable iterative techniques for solving large sparse matrices. Trescott and Larson[10] studied the convergence properties of SIP in comparison to ADI(Alternating Direction Implicit) and LSOR(Line Successive Overrelaxation). In 1981 Kuiper[11] compared SIP to ICCG concluding that for confined aquifer problems ICCG appeared to be superior while for water table conditions(nonlinear) the methods performed comparably well. More recently Kuiper[12] compared a number of iterative techniques on a set of linear and nonlinear groundwater problems. The conclusion of his most recent study was that the Picard-preconditioned conjugate gradient methods performed best for both the linear and nonlinear problems. Each of these papers are concerned with performance on scalar machines.

In the open literature there are several comparisons of iterative methods which take into account vectorizability of particular algorithms, and their efficiency on given "supercomputers". A few of the more pertinent references are given below.

Meurant[6] and Jordan[7] have compared VICCG, POLCG, and block preconditioned conjugate gradient methods for two dimensional problems. Meurant's study concluded that for "small" problems (grids of 60x60) POLCG seemed optimal on the 2-pipe Cyber

205 while for larger problems the block preconditioning methods appeared to do better. Jordan preferred polynomial preconditioners from the standpoint of parallel processing.

Hayami and Harada[8] have recently estimated that on the NEC SX-2 supercomputer (assuming an acceleration due to vector processing of 40 times the scalar rate) DSCG will run ten times faster than ICCG for most problems. As the Cyber 205 is also a highly vectorizable machine DSCG has been included as one of the algorithms to be tested.

Gustafsson[4] considered a class of first order methods (SSOR and MICCG(n)) which were shown to have markedly better convergence properties than ICCG. Extending Gustafsson's MICCG, Ashcroft and Grimes[9] applied the algorithm to three dimensional problems and in particular programmed it in such a way that (at least on a Cray machine) much of the code can be made to run concurrently. Their analysis compares MICCG, SSOR, DSCG and Nofill Red/black Incomplete factorization preconditioners concluding that the "concurrent" MICCG method, while not as vectorizable as some of the other methods, is superior due to its relatively low iteration count.

The present work differs from that previously reported in the following respects. The machine used is exclusively a two-pipe Cyber 205, both two and three dimensional problems are run, two of the five test problems have anisotropic conditions, and finally, the set of iterative techniques tested is larger than previously reported.

The matrix equations used in the present comparison arise from finite difference approximations to the 2D and 3D partial differential equations governing the flow of groundwater in confined aquifers. These equations involve self adjoint parabolic (or elliptic for the case of steady state flow) differential operators for which all of the iterative schemes tested are applicable. Equations of a similar type result in various transport phenomenon such as heat conduction and laser fusion.

The paper is outlined as follows. In section 2, the five model problems are described. This is followed in section 3 by a short explanation of the algorithms. Section 4 concludes with comparitive results of each on the set of test problems.

2. Formulation of the Problems

The derivation of the three dimensional partial differential equation governing the distribution of hydraulic head in a confined aquifer can be found in Reddell and Sunada[13]. Assuming components of the transmissivity tensor lie along coordinate axes the equation can be concisely written as:

$$Sh_t = (K^x h_x)_x + (K^y h_y)_y + (K^z h_z)_z + R \tag{1}$$

where h is the pressure head to be determined; S is the storage coefficient of the porous media; K^x, K^y, and K^z are transmissivities of the aquifer in the x,y, and z directions; and R is the volumetric flux of recharge or withdrawal per unit volume of water from the aquifer. Sources and sinks are approximated by delta functions with strengths equal their volumetric flux.

The primary boundary conditions are Dirichlet and homogeneous Neumann, which correspond physically to constant head and no-flux boundaries respectively.

In each of the test problems, the aquifer is contained in a rectangular solid whose faces are assigned appropriate boundary conditions. For aquifers of a more general shape, points exterior to the aquifer boundaries are assigned zero transmissivities for the purpose of eliminating their role in the determination of head values interior to the aquifer (see for example [14]). This is done in problem five where an "L" shaped region is considered.

In the first four problems, a steady state solution is sought. Of these the fourth has a nonunique solution since all of its boundary conditions are homogeneous Neumann. The initial guess for h in each of the first four problems is zero except at points where sources/sinks are present or non-homogeneous Dirichlet boundary conditions apply. The fifth test problem considers a fully time dependent problem, but solves for only one time step, given values of the head everywhere for the immediately preceding time level:

$$h(x,y,z)=10^{2(x-y+1)}. \tag{2}$$

problem 1:

The first problem models an isotropic homogeneous aquifer with constant head boundaries. No sources or sinks are present. The transmissivities and boundary conditions are:

$$\begin{gathered} K^x=K^y=K^z=1.0 \quad \text{and} \\ h=1 \text{ on all boundaries.} \end{gathered} \tag{3}$$

problem 2:

The second test problem examines the effect of using transmissivities with linear gradients along each of the coordinate axes. As before, no sources or sinks are present. Specification of case two is as follows:

$$K^x=1-16y/17$$

$$K^y=(16x+1)/17$$

$$K^z=(96z+1)/10$$

(4)

$$h_n=0 \text{ at } x=0 \text{ and } z=1$$
$$h_n=1 \text{ at } x=1$$
$$h=10 \text{ at } y=0, \text{ and}$$
$$h=1 \text{ at } y=1, z=0.$$

problem 3:

Refer to figure 1 in the description of the third test problem. There are four subsections of the aquifer which are isotropic and homogeneous, but whose transmissivities lie in the range zero to forty. A feature of this problem which makes it difficult is that one of the subsections is an impermeable layer with a relatively small transmitting aperture. A source of strength 1.0 is positioned at (.875,.9375,.1875), and a sink of strength .25 is located at (.4375,.625,.5). Specification of case three is:

(5)

$$K^x=K^y=K^z=1.0 \text{ in region A}$$
$$K^x=K^y=K^z=0.0 \text{ in B}$$
$$K^x=K^y=K^z=20.0 \text{ in C}$$
$$K^x=K^y=K^z=40.0 \text{ in D}$$
$$h_n=0.0 \text{ at } y=0, z=1$$
$$h_n=-1.0 \text{ at } x=1$$
$$h=1 \text{ at } y=1,\ z=0, \text{ and}$$
$$h=5 \text{ at } x=0.$$

The center point of the impermeable shell (region B) is at (.5,.4375,.5) with side length .75 and width .125. The center-point for region C is the same as for B but has a side length of .375. The center for region D is (.5,.9,.5) with side length .125, and finally the aperture is centered at (.75,.5,.5) with side length in the y and z directions of .25, and width .125.

problem 4:

Test case four is a two dimensional example taken from Stones's paper[1]. Two dimensional problems are solved by the three dimensional code by keeping boundary conditions and transmissivities independent of the z coordinate. See figure 2 for an illustration of the domain modelled. Specifications are:

(6)
$$\begin{gathered} K^z=1 \text{ everywhere} \\ K^x=K^y=1 \text{ in region A} \\ K^x=1,\ K^y=100 \text{ in B} \\ K^x=100,\ K^y=1 \text{ in C} \\ K^x=K^y=K^z=0 \text{ in D, and} \\ h_n=0 \text{ on all boundaries.} \end{gathered}$$

Line sources for problem four were located at (x,y) equal to (.1,.1),(.1,.9), (.767,.133) with strengths 1.0,.5 and .6 respectively; while sinks were located at (.467,.5) and (.9,.9) with strengths 1.83 and .27. They are represented in figure 2 as (+)'s and (-)'s.

problem 5:

Problem five can be found in Kershaw's paper[15]. It is two dimensional and is without sources or sinks. In figure 3, D or N at a boundary signifies homogeneous Dirichlet or homogeneous Neumann conditions respectively. Equation (2) gives the initial condition for this problem, and the transmissivities are:

$$\begin{gathered} K^x=K^y=K^z=10^{-4} \text{ in region A} \\ K^x=K^y=K^z=10^{-2} \text{ in B} \end{gathered}$$

(7)
$$K^x=K^y=K^z=10 \text{ in C, and}$$
$$K^x=K^y=K^z=10^6 \text{ in D.}$$

3. Summary of the Numerical Methods

Equation (1) is approximated by a node-centered finite difference formulation. The truncation error is second order in the spatial increments if a uniform mesh is used. The accuracy drops to first order if the mesh is nonuniform. The boundary conditions are handled in such a way that the second order truncation error is maintained regardless of the spacing.

By employing the "natural" ordering of the grid points a seven banded symmetric matrix equation results,

(8) $$A\underset{\sim}{h}=\underset{\sim}{b}$$

where A is positive semi-definite, $\underset{\sim}{b}$ represents known boundary conditions and possible sources or sinks, and $\underset{\sim}{h}$ is the vector of unknown pressure head values to be solved for.

Solving for $\underset{\sim}{h}$ involves iterating on an initial guess to a final solution which in some sense must be close to the exact answer. Defining the mth iterate as $\underset{\sim}{h}^{(m)}$, an error vector associated with the mth iterate is introduced:

(9) $$\underset{\sim}{e}^{m}=\underset{\sim}{b}-A\underset{\sim}{h}^{(m)}$$

The inf-norm of $\underset{\sim}{e}^{(m)}$ scaled to $\underset{\sim}{e}^{(0)}$ defines the mth residual used in the results section of this paper. The iterative process is halted when the absolute magnitude of the scaled residual is less than a tolerance level of 10^{-8}.

The iterative algorithms are outlined below. For a complete description of any particular method the reader is referred to the original works from which the methods have been extracted.

<u>SIP</u>:

The original version of Stone's[1] SIP was extended to three dimensional problems by Weinstein et al[16]. The method splits up the matrix A(eq. 8) into an approximate LU factorization, where L and U are upper and lower triangular matrices. The product of these two matrices, however, yields thirteen nonzero diagonals rather than seven which were originally in A. Two term Taylor series corrections (weighted by an iteration parameter) are "lumped" into the original LU factors producing a revised factorization which minimizes somewhat the effects of the concommitant diagonals.

Values for the iteration parameters range between zero and one, with values close to one being near optimal. Careful selection of this parameter is critical. Too large a value could cause divergence while too small a value may lead to very slow convergence. A cycle of 3 parameters was selected (see eq 10) where α_{max} is found by employing the formula given in Weinstein et al[16] (using an average value, rather than a maximum value over the entire domain).

$$(10) \qquad 1-\alpha_t=(1-\alpha_{max})^{t/3}, \text{ for } t=1,2,3.$$

A refinement of SIP involves renumbering the nodes in producing the matrix equation (8). Weinstein et al use two out of a a possible four distinct numberings that can be made by reversing the order of the nodes along one or two of the coordinate axes. Including a second renumbering of the A matrix and performing two approximate inversions of A for each SIP iteration was found to increase the total computation time required to converge to a solution. In addition, if (as is done here) the L and U matrices are stored for each renumbering of the A matrix (as well as for each α_t) rather than recalculated at each SIP iteration, the amount of storage required can become quite large. For these reasons only the natural ordering of the nodes is used.

Conjugate Gradient Methods:

With the recent introduction of incomplete Cholesky preconditioners[2] to the basic conjugate gradient algorithm[17, 18,19], there has been a renewal of interest in this algorithm and an ongoing search for "optimal" preconditioners[6,7,8,9].

Conjugate gradient methods differ from SIP in that they are based upon the assumption that A is a symmetric matrix. SIP has no such restriction. Variations of preconditioned conjugate gradient methods do exist for nonsymmetric matrices [20,21] but the theory and application is significantly different than for the symmetric case.

The incomplete Cholesky conjugate gradient method of Meijerink and van der Vorst[2], involves a similar factorization of the A matrix as found in SIP. It is given by:

$$A \approx LDL^T, \tag{11}$$

where D is a diagonal matrix, and L is a lower triangular matrix whose nonzero diagonals match positions of those in the A matrix.

The product of LU from SIP and LDL^T from ICCG are identical provided the iteration parameter used in SIP is set to zero. The ICCG factorization neglects the six additional diagonals which result in the product of LDL^T. Alternative factorizations include extra diagonals in the L matrix in an effort to give a more accurate Cholesky decomposition, but these are not considered here. Such variants tend to increase the storage requirements of the method and the amount of recursive work in each iteration. This was observed by Kershaw[15].

Gustafsson[4] improved on ICCG by incorporating Stone's idea of minimizing the effect of concommitant diagonals produced in the LDL^T factorization. Unlike Stone, who uses a two term Taylor expansion of the unwanted terms of the factorization, Gustafsson employs a one term Taylor expansion. This guarantees that a symmetric factorization can be found and that the conjugate gradient algorithm remains applicable. Gustafsson's method is referred to as MICCG.

Just as in Stone's method an iteration parameter is used which ranges in value between zero and one. MICCG was found to be less sensitive to the actual value of the iteration parameter than is SIP. For each of the five problems tested, a trial and error system was used to find optimal values (the magnitudes ranged between .95 and 1.0).

The advent of vector processing machines has resulted in efforts to modify the inherent recursion involved in inverting the LDL^T approximation to the A matrix. This was partially accomplished by van der Vorst[3] who split the lower triangular matrix L into diagonal blocks which could then be individually inverted via truncated Neumann series.

Van der Vorst's method of vectorizing the recursive part of the iteration loop has been employed in "vectorizing" MICCG. The resultant algorithm is referred to in this paper as VICCG and should not be confused with van der Vorst's vectorized ICCG. As with MICCG an iteration parameter must be provided and in general is found to have a magnitude less than the optimal value found for MICCG. It is also found by trial and error.

One final variant of the MICCG algorithm involves using a three term truncated Neumann series approximation for inverting the lower triangular matrix L in the LDL^T factorization. While this variation eliminates recursion altogether in the iteration loop, it was found that for the fastest convergence the "optimal" iteration parameter was usually zero. The resulting algorithm therefore is actually a polynomial form of the unmodified original ICCG method. It is here referred to as PICCG.

An alternative to factoring the A matrix to obtain a preconditioner, involves approximating A's inverse by a matrix polynomial[22]. Recently, Dubois et al[23] have reported success in using a truncated Neumann series expansion in A for an approximation to A's inverse. Johnson et al[24] have proposed a polynomial preconditioner whose coefficients are determined in an effort to reduce the condition number of the product of A and its preconditioner. Minimizing this number is important since theoretically the rate of convergence of the conjugate gradient algorithm increases as the magnitude of the condition number is reduced.

Saad's[5] method is a variant of the method of Johnson et al. It attempts to minimize

(12) $$\int_0^{\mu_{max}} (1-s(\mu))^2 w(\mu) d\mu$$

over all polynomials $s(\mu)$ less than or equal to a specified degree, and where μ_{max} is the largest eigenvalue of the matrix A. In the present work a polynomial of third degree is sought, with weight factor:

(13) $$w(\mu)=(\mu_{max}-\mu)^{-\frac{1}{2}}\mu^{-\frac{1}{2}}.$$

Saad originally proposed finding μ_{max} by employing Gershgorin circles[25]. In the present work μ_{max} is fixed at 2 which is near the value found by employing Gershgorin's theorem to matrices (of the type considered here) which have been diagonally scaled. In almost every instance the choice of 2 as an upper bound over that found via Gershgorin's theorem produced faster convergence. This algorithm is referred to as POLCG.

If instead of a polynomial of order three one of zeroth order is selected, and the A matrix is diagonally scaled, the preconditioner for the conjugate gradient algorithm becomes the identity matrix. This simple preconditioner is referred to here as DSCG.

4. Results

Results of the tests are presented in tables I and II of this section. Included are: timings, iteration parameters, iteration counts, and megaflop rate for the iteration loop of each algorithm. In table II, results from a subset of the problems listed in table I which have been modified to increase the number of nodes and thereby the size of matrix equation to be solved are given.

The categories in the tables are relatively self explanatory. By set-up time is meant the time necessary to set up the matrix equation and to initiate the iteration loop. The megaflop rate was found by dividing the total number of operations performed by the average CPU time needed to complete one iteration loop. Parenthetical values alongside those of VICCG are results of setting its iteration parameter to zero. This produces a code which mimics van der Vorst's[3] vectorized version of ICCG.

The burden of vectorizing each of the algorithms was carried primarily by the Cyber 205 vector compiler with scalar optimizer. Few special Q8 calls are used. In particular they are Q8MAX and Q8SDOT. Q8MAX is used by each of the schemes for computing the inf-norm of the residual vector for convergence checking. Q8SDOT gives the scalar dot product of two vectors and is used solely by the conjugate gradient codes. Special "chaining" of do loops for the purpose of increasing the number of linked triadic operations was not done.

Diagonal scaling of the A matrix was performed only on

POLCG and DSCG. To find the scaled residual as defined by (9) the computed residuals are first multiplied by the diagonal scaling factors to find "true" values of the residuals. Scaling the remaining conjugate gradient algorithms as outlined by Eisenstat[26] would not reduce the number of calculations in the present instance because the same number of operations saved in one portion of the algorithm would need to be performed in finding the "true" residual as given in (9).

Two of the codes seem to perform better than all others in table I. They are MICCG and POLCG, which have the fastest iteration times on 2 and 3 of the test problems respectively. SIP performed relatively poorly on all but the first two test cases. (If the tolerance level for convergence changes from 10^{-8} to 10^{-3} SIP does dramatically better-particularly on problem 2-but for such a stopping criterion SIP gives incorrect results for problem 3. Given this larger tolerance level, POLCG and SIP performed the best on two problems each.) DSCG and PICCG did better than both ICCG and VICCG for the problems in table I leading one to believe that vectorizing the ICCG code in the manner of van der Vorst is not particularly efficient for small three dimensional problems because the vector lengths are not long enough. The situation for VICCG is worse if an iteration parameter is not used(as it is for MICCG) to improve on the iteration count. The phenomenal timings of SIP and MICCG for test problem 1 indicate that the preconditioner(for MICCG) and the LU factorization(for SIP) are nearly exact inverses of the original matrix. (The exact answer for test problem one was a constant head value throughout the domain.)

Table I highlights some striking differences between POLCG and MICCG. POLCG has nearly three times the megaflop rate of MICCG while MICCG requires on average only half as many iterations. On a scalar machine POLCG could not compete with MICCG because it requires more iterations and actually performs more operations in its inner loop than MICCG. The vectorizability of POLCG allows it to be competitive with MICCG on a Cyber 205.

Continuing the comparison between POLCG and MICCG it is noted that for "optimal" convergence in MICCG a suitable iteration parameter must be found while for POLCG no such parameter is needed. Ashcroft and Grimes[9] recommend an optimal iteration parameter of just less than unity for MICCG based upon empirical evidence. While these observations appear to be generally true, there are exceptions. In the first test problem if a value of .999 is used rather than 1.0 the iteration count increases from 2 to 16. While problem 1 is a special case, instances occur where the value of an "optimal" iteration parameter is rather sharply defined (in particular when modelling anisotropic media). For such cases several runs of MICCG may be required to ensure that a given choice of iteration parameter is not far from optimal. For the "larger" problems reported in table II, it was found that the iteration count varied from 61 to 42 to 51 as the iteration parameter for MICCG changed from .95 to .994 to .9996 on problem 2. Similarly on problem 4 of table II, altering the iteration parameter of MICCG from .95 to .999 to .9995 produced iteration counts of 102, 63, and 66 respectively.

The neccesity of an iteration parameter for MICCG may become a problem if the matrix equation is periodically updated (as is done when modelling groundwater problems for which water table conditions apply). If revision of the matrix equation is frequently done during the solution of a time dependent problem one would expect some variation in the value of the "optimal" iteration parameter. To find its value at each update would be impractical, while using the same value throughout the iterations may or may not be optimal for the whole problem. The only way to be assured of its "optimality" is to run a series of tests on the complete time dependent problem for a variety of iteration parameters. Analysis of a single time step is not sufficient.

A further disadvantage in using MICCG rather than POLCG on problems requiring frequent updates is that the set-up time for MICCG is roughly three times that of POLCG.

An observable advantage in using MICCG over all of the other methods has been previously reported[9] and is confirmed in the tests reported here. Comparing iteration counts in problems 2 and 4 of table I with those in table II, it can be seen that the increase for MICCG compared to that for all of the other methods is much less. The rate at which the number of iterations increases as a function of N (number of nodes along one dimension of the numerical grid) is $O(N^{\frac{1}{2}})$ for MICCG while for POLCG and ICCG the increase appears to be $O(N)$. The rate for VICCG falls somewhere in between depending upon the test problem.

VICCG improves substantially as the size of the problem increases as can be seen by comparing table I results to those of table II. However in cases of strong anisotropy, as is present in test problem 4, VICCG has some definite difficulty. In fact the presence of an iteration parameter in the algorithm is hardly warranted.

In conclusion, it would appear that for relatively small domains (on the order of a few thousand unknowns) POLCG is a very good algorithm as applied to the solution of two and three dimensional groundwater flow equations. For larger problems MICCG is perhaps a better choice when circumstances are such that frequent updates of the matrix equation are unnecessary. As the number of unknowns continues to increase, VICCG is an attractive alternative to MICCG because of its vectorizability. However, strongly anisotropic conditions seem to severely increase VICCG's iteration count and for such problems MICCG may give superior performance.

Table 1

Problem 1 (4913 nodes)

Algorithm	Set up time	Iteration parameter	Number of iterations	Iteration time	*Megaflop* rate
SIP	.157	1.0	2	.007	20
ICCG	.038	n/a	19	.107	28
POLCG	.029	n/a	13	.039	92
VICCG	.099(.099)	.9(0.0)	19(21)	.223(.252)	21
MICCG	.091	1.0	2	.001	28
DSCG	.026	n/a	39	.045	94
PICCG	.039	n/a	21	.050	89

Problem 2 (4913 nodes)

Algorithm	Set up time	Iteration parameter	Number of iterations	Iteration time	Megaflop rate
SIP	.155	.9994	31	.206	20
ICCG	.038	n/a	44	.278	28
POLCG	.028	n/a	55	.183	92
VICCG	.097(.097)	.98(0.0)	30(44)	.367(.556)	21
MICCG	.089	.975	29	.184	28
DSCG	.026	n/a	178	.211	94
PICCG	.036	n/a	88	.224	89

Problem 3 (4913 nodes)

Algorithm	Set up time	Iteration parameter	Number of iterations	Iteration time	Megaflop rate
SIP	.147	.9974	524	3.585	20
ICCG	.038	n/a	45	.278	28
POLCG	.029	n/a	39	.128	92
VICCG	.090(.090)	.92(0.0)	40(48)	.500(.609)	21
MICCG	.082	.95	31	.182	28
DSCG	.026	n/a	124	.146	94
PICCG	.037	n/a	52	.131	89

Problem 4 (1922 nodes)

Algorithm	Set up time	Iteration parameter	Number of iterations	Iteration time	Megaflop rate
SIP	.042	.9994	127	.315	18
ICCG	.017	n/a	72	.166	26
POLCG	.014	n/a	103	.128	85
VICCG	.043(.043)	.4(0.0)	114(116)	.271(.275)	26
MICCG	.043	.9953	43	.101	26
DSCG	.013	n/a	336	.150	90
PICCG	.017	n/a	177	.167	83

Table 1(continued)

Problem 5 (5202 nodes)

Algorithm	Set up time	Iteration parameter	Number of iterations	Iteration time	Megaflop rate
SIP	.166	.9998	189	1.241	19
ICCG	.046	n/a	58	.373	25
POLCG	.038	n/a	52	.160	93
VICCG	.102(.102)	.9(0.0)	47(59)	.224(.282)	47
MICCG	.105	.95	35	.212	27
DSCG	.035	n/a	162	.186	94
PICCG	.046	n/a	67	.162	87

Table 2

Problem 2 (35937 nodes)

Algorithm	Set up time	Iteration parameter	Number of iterations	Iteration time	Megaflop rate
SIP	1.034	.9994	169	8.562	19
ICCG	.207	n/a	84	3.995	27
POLCG	.150	n/a	139	3.768	84
VICCG	.570(.570)	.996(0.0)	47(84)	2.792(5.082)	33
MICCG	.566	.994	42	1.967	27
DSCG	.136	n/a	457	4.306	87
PICCG	.207	n/a	290	6.254	78

Problem 4 (7442 nodes)

Algorithm	Set up time	Iteration parameter	Number of iterations	Iteration time	Megaflop rate
SIP	.251	.994	223	2.120	19
ICCG	.745	n/a	145	1.298	26
POLCG	.065	n/a	193	.855	93
VICCG	.154(.154)	.4(0.0)	221(232)	1.444(1.510)	50
MICCG	.174	.9988	63	.589	26
DSCG	.058	n/a	640	1.056	94
PICCG	.073	n/a	328	1.132	88

References

1. Stone, H.L., Iterative solution of implicit approximations of multidimensional partial differential equations, SIAM J. Numer. Anal., 5, 530-558, 1968.

2. Meijerink, J.A. & H.A. van der Vorst, An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix, Math. of Comput., 31, 148-162, 1977.

3. van der Vorst, H.A., A vectorized variant of some ICCG Methods, SIAM J. Sci. & Stat. Comput., 3, 350-356, 1982.

4. Gustafsson, I., A class of first order factorization methods, BIT, 18, 142-156, 1978.

5. Saad, Y., Practical use of polynomial preconditionings for the conjugate gradient method, SIAM J. Sci. & Stat. Comput., 6, 865-881, 1985.

6. Meurant, G. Vector preconditioning for the conjugate gradient on Cray-1 and CDC Cyber 205, in Computing Methods in Applied Sciences and Engineering, VI, ed. by R. Glowinski and J.L.Lions, North-Holland, 1984.

7. Jordan, T.L., Conjugate gradient preconditioners for vector and parallel processors, in Elliptic Problem Solvers II, Proceedings of the Elliptic Problem Solvers Conference, Monterey, CA, 1983, Academic Press, New York, pp 127-139, 1983.

8. Hayami, K. & N. Harada, The scaled conjugate gradient method on vector processors, in Supercomputing Systems, Proc. of the first International Conference, St Petersberg, FL, Dec. 16-20, 1985, eds. Kartashev & Karteshev.

9. Ashcroft, C.C. and R.G. Grimes, On vectorizing incomplete factorization and SSOR preconditioners, SIAM J. Sci. Stat. Comput., 9, 122-151, 1988.

10. Trescott, P.C. & S.P. Larson, Comparison of iterative methods of solving two-dimensional groundwater flow equations, Water Resources Research, 13, 125-136, 1977.

11. Kuiper, L.K., A comparison of the incomplete Cholesky conjugate gradient method with the strongly implicit method as applied to the solution of two-dimensional groundwater flow equations, Water Resources Research, 17, 1082-1086, 1981.

12. Kuiper, L.K., A comparison of iterative methods as applied to the solution of the nonlinear three-dimensional groundwater flow equation, SIAM J. Sci. Stat. Comput., 8, 521-528, 1987.

13. Reddell, D.L. & D.K. Sunada, Numerical simulation of dispersion in groundwater aquifers, Colorado State Univ. Hydrology Paper 41, Fort Collins, CO, 1970.

14. Mercer, J.W. & C.R. Faust, Groundwater modelling: Aplications, Ground Water, 18, 486-497, 1980.

15. Kershaw, D.S., The incomplete Cholesky conjugate gradient method for the iterative solution of systems of linear equations, J. Comp. Physics, 26, 43-65, 1978.

16. Weinstein, H.G., H.L. Stone, & T.V. Kwan, Iterative procedure for solution of systems of parabolic and elliptic equations in three dimensions, Ind. Eng. Chem. Fundam., 8, 281-287, 1969.

17. Hestenes, M.R. & E. Steifel, Methods of conjugate gradients for solving linear systems, Nat. Bur. Standards J. Res., 49, 409-436, 1952.

18. Golub, G.H. & C.F. Van Loan, Matrix Computations, Johns Hopkins University Press, Baltimore, MD, 1985.

19. Noble, B., Applied Linear Algebra, Prentice Hall, Englewood Cliffs, NJ, 1969.

20. Eisenstat, S.C., H.C. Elman, & M.H. Schultz, Variational iterative methods for nonsymmetric systems of linear equations, SIAM J. Numer. Anal., 20, 345-357, 1983.

21. Elman, H.C., Y. Saad, & P.E. Saylor, A hybrid Chebyshev Krylov subspace algorithm for solving nonsymmetric systems of linear equations, SIAM J. Sci. Stat. Comput., 7, 840-896, 1986.

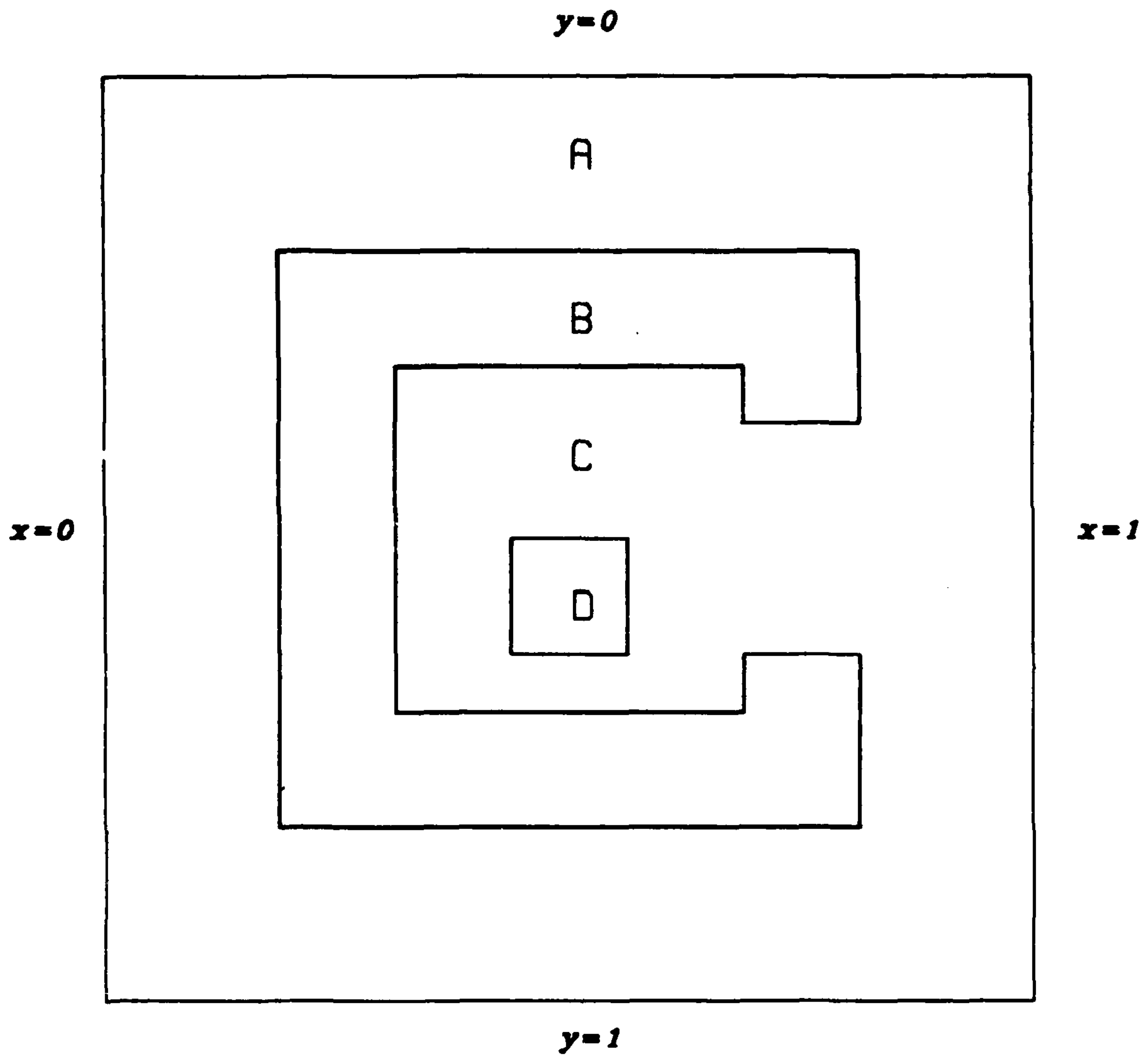
22. Rutishauser, H., Theory of gradient methods, Refined iterative methods for computation of the solution and the eigenvalues of self-adjoint boundary value problems, Inst. of Applied Mathematics, Zurich, 24-49, 1959.

23. Dubo's, P.F., A. Greenbaum, & G.H. Rodrigue, Approximating the inverse of a matrix for use in iterative algorithms on vector processors, Computing, 22, 257-268, 1979.

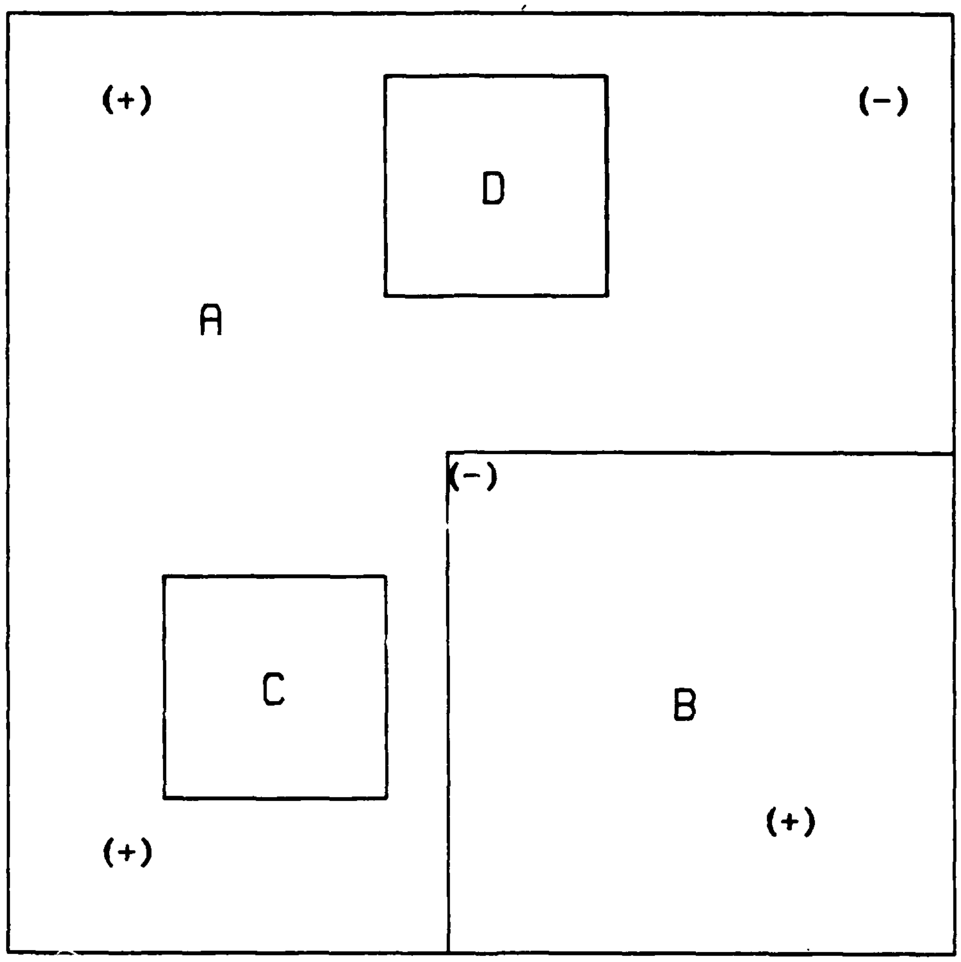
24. Johnson, O.G., C.A. Micchelli, & G. Paul, Polynomial preconditioners for conjugate gradient calculations, SIAM J. Numer. Anal., 20, 362-376, 1983.

25. Isaacson, E. & H.B. Keller, Analysis of Numerical Methods, Wiley and Sons Inc., New York, 1966.

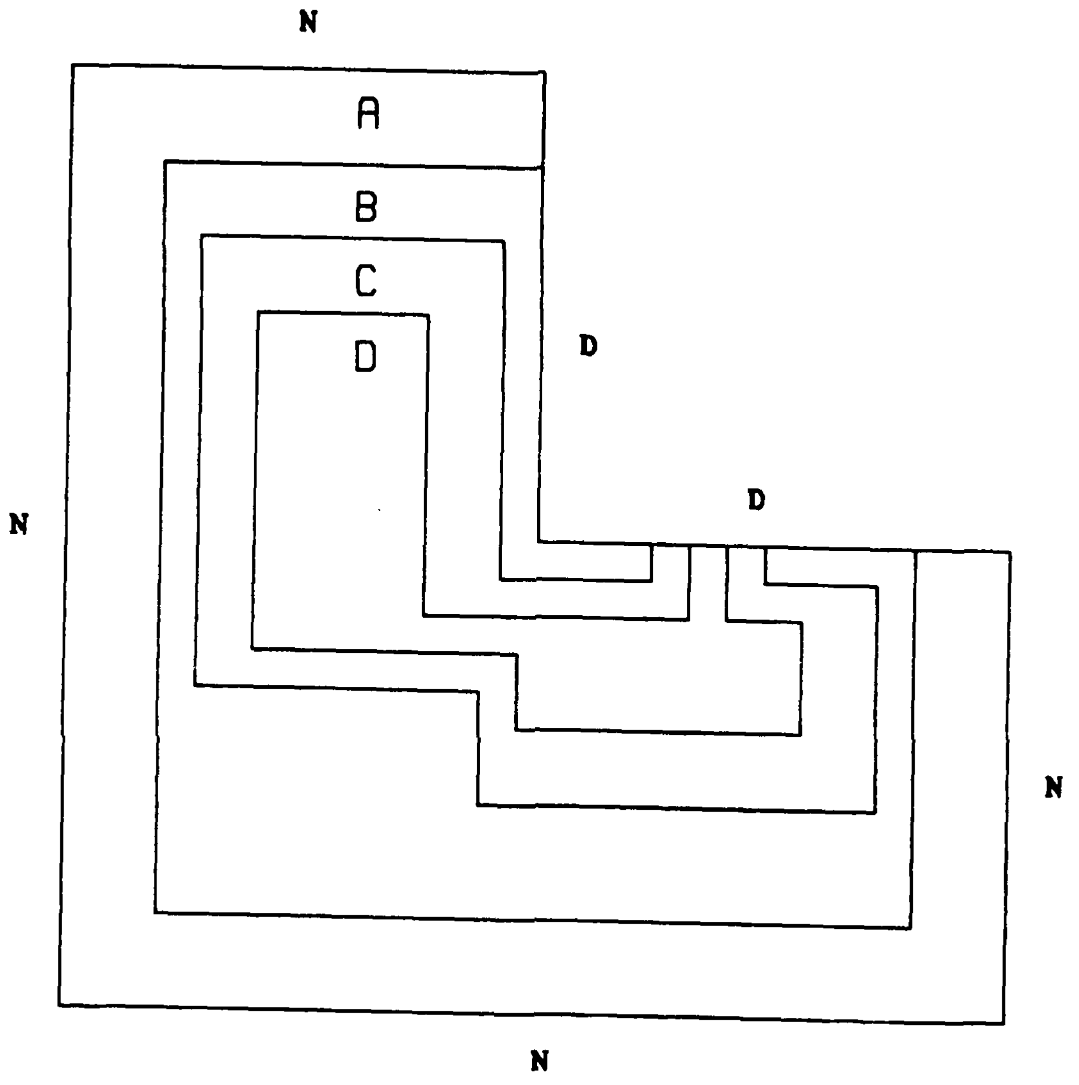
26. Eisenstat, S.C., Efficient implementation of a class of preconditioned conjugate gradient methods, _SIAM J. Sci. Stat. Comput._, 2, 1-4, 1981.



LATERAL SLICE OF PROBLEM # 3 (Z=0.5)



PROBLEM # 4



PROBLEM # 5

INITIAL DISTRIBUTION LIST

DIRECTOR (2)
DEFENSE TECH. INFORMATION
CENTER, CAMERSON STATION
ALEXANDRIA, VA 22314

CENTER FOR NAVAL ANALYSES
4401 FORD AVENUE
ALEXANDRIA, VA 22302-0268

LIBRARY (2)
CODE 0142
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943

DIRECTOR OF RESEARCH ADMINISTRATION
CODE 012
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943

DEPARTMENT OF MATHEMATICS
CODE 53
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943

ASST. PROFESSOR CLYDE SCANDRETT (15)
CODE 53Sd
DEPARTMENT OF MATHEMATICS
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943

U. S. DEPARTMENT OF ENERGY
OFFICE OF ENERGY RESEARCH
WASHINGTON, DC 20545