AD-A250 961

DTIC
S ELECTE JUN 2 1992 D
C

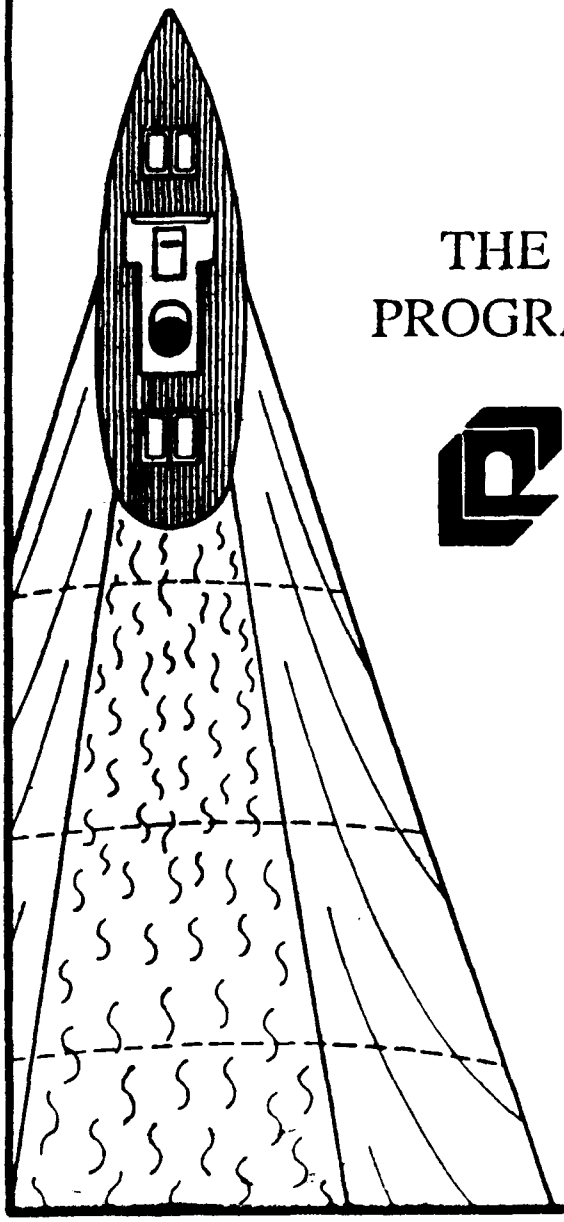
GMRES Iterative Solution of Matrix Systems Derived from Boundary Element Techniques

Lorraine G. Olson
Assistant Professor
Department of Mechanical Engineering and Applied Mechanics

Contract Number N00014-86-K-0684
Technical Report Number 89-04

June, 1989

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited



THE UNIVERSITY OF MICHIGAN
PROGRAM IN SHIP HYDRODYNAMICS

COLLEGE OF ENGINEERING

NAVAL ARCHITECTURE &
MARINE ENGINEERING

AEROSPACE ENGINEERING

MECHANICAL ENGINEERING &
APPLIED MECHANICS

SHIP HYDRODYNAMIC
LABORATORY

SPACE PHYSICS RESEARCH
LABORATORY

ERIM

Summary

We apply the Generalized Minimal Residual (GMRES) iterative equation solution technique to a set of full, unsymmetric matrix systems generated by a standard boundary element method. The test problems chosen produced well conditioned matrices. The GMRES technique, when used without preconditioning and with a sufficient number of trial vectors, solved the matrix system using as few as 23% of the operations required by a direct Gauss reduction. The class of partial LU decomposition preconditioners tested degraded the condition number of the matrices, and consequently did not reduce the GMRES solution time. In general, the GMRES technique does not appear to be of practical interest compared to the direct reduction unless other factors (availability of a good approximation to the final solution, etc.) intervene.

Accession For
NTIS GRA&I
DTIC TAB
Unannounced
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

Statement A per telecon Dr. Edwin Rood
ONR/Code 1132
Arlington, VA 22217-5000

NWW 6/1/92

92-13784

92 5 26 010

1 Introduction

Iterative equation solvers are becoming increasingly popular for certain special computational applications. One such application is in solving large sparse matrix systems, such as those generated in 3-D finite element or finite difference analysis. In these problems, the storage required for the nonzero matrix entries may be only .02 N^2 [1], and similarly the number of operations required for a matrix-vector multiply is significantly less than N^2 (so that we can afford many iterations). Another widely-investigated application for iterative methods is in cases where the solution is already approximately known, such as in time-dependent problems or nonlinear problems. Finally, iterative solvers are generally more amenable to vectorization/parallelization than direct solvers.

In this study, we have examined whether a particular iterative solver (GMRES [2]) is competitive with direct Gauss solvers for *full, unsymmetric* matrices derived from standard boundary integral techniques. Here, we do not have the storage or matrix-vector multiply advantages that occur in solving sparse matrices.

The Generalized Minimal Residual algorithm developed by Saad and Schultz [2], and presented in preconditioned form by Shakib, Hughes, and Johan [3], is very similar to conjugate gradient algorithms. However, GMRES is intended specifically for unsymmetric matrices, it requires only a single matrix-vector multiply in each iteration, and its rate of convergence depends only on the condition number of the matrix. (Some conjugate gradient algorithms' convergence rates depend on the square of the condition number.) Brussino and Sonnad [4] present an extensive survey of the various conjugate-gradient and GMRES algorithms. Their test cases [4,5] indicate that GMRES is comparable or superior to preconditioned conjugate gradient algorithms[1] for many problems.

Section 2 introduces the test problems we investigated, and discusses the matrix condition numbers. In addition, we discuss the residual error from a direct solution and some results for a least squares conjugate gradient solver. In section 3 we discuss the GMRES algorithm and proposed preconditioner.

[1]It should be noted that the least-squares conjugate gradient algorithm discussed in section 2.3 is apparently one of the least efficient conjugate gradient algorithms for unsymmetric matrices.

Section 4 discusses our numerical results, and section 5 details our conclusions. Appendix A includes a listing of the GMRES program.

2 Test Problem

To generate the systems of equations used for testing, we employed Brebbia's boundary integral program PROGRAM1 [6]. This program solves Laplace's equation

$$\nabla^2 u = 0 \tag{1}$$

in two dimensions, subject to the boundary conditions $u = \overline{u}$ on boundary Γ_1 and $q = \overline{q}$ on boundary Γ_2. (The entire boundary $\Gamma = \Gamma_1 + \Gamma_2$.) The boundary integral equation is derived by the method of weighted residuals, and becomes

$$\frac{1}{2}u^i + \int u\frac{\partial u^*}{\partial n}d\Gamma_2 + \int \overline{u}\frac{\partial u^*}{\partial n}d\Gamma_1 = \int \overline{q}u^* d\Gamma_2 + \int q u^* d\Gamma_1 \tag{2}$$

where u^i is the solution for u at point i, u^* is the Green's function for u, and $q* = \frac{\partial u^*}{\partial n}$. Constant elements are used to generate a matrix system of the form

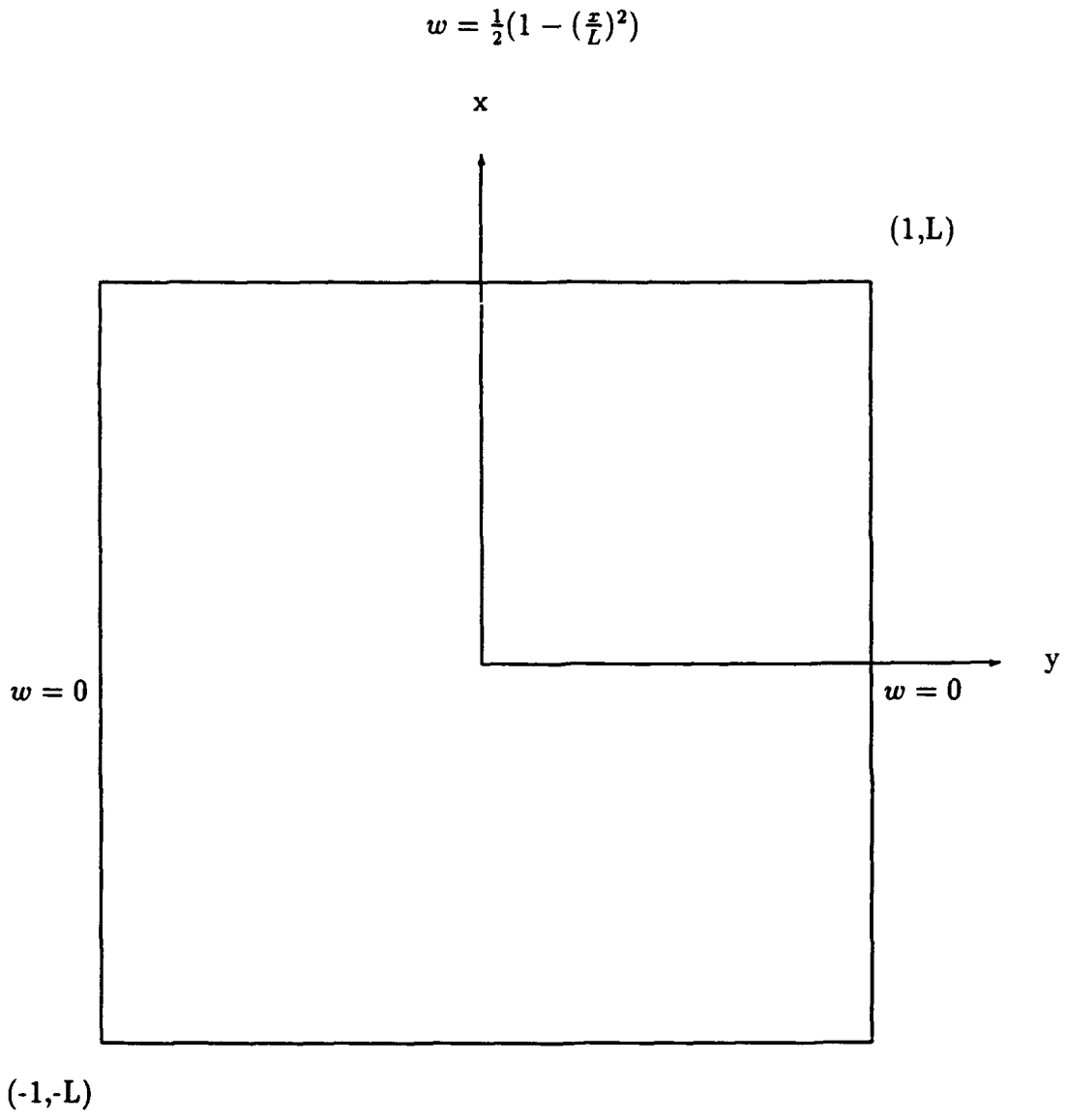
$$\mathbf{Ax} = \mathbf{b} \tag{3}$$

We chose to examine one physical problem with different aspect ratios, and produce successively finer boundary element discretizations to generate our test cases. Figure 1 shows the physical case considered, and its exact solution. We examined three cases: L=1, L=10, and L=100. Figure 2 shows a typical mesh with 12 unknowns for the square (L=1) example.

2.1 Condition Number Results

It is well known that the rate of convergence of iterative methods, and therefore the solution time, is closely tied to the condition number of the matrix **A**. The accuracy of direct solution methods may be impacted by the condition number of **A**, but of course the solution time is unaffected. Table 1 shows the condition numbers[2] for our test problems with various N. Notice that the meshes are very well conditioned.

[2]Condition numbers estimated with LINPAK subroutine DGBCO.



$$w = -2 \sum_{n=0}^{n=\infty} \frac{(-1)^n}{(\lambda_n L)^3} \frac{cosh(\lambda_n y)}{cosh(\lambda_n)} cosh(\lambda_n x)$$

$$\lambda_n = \frac{2n+1}{2}\pi$$

Figure 1: Physical Test Case and Analytical Solution

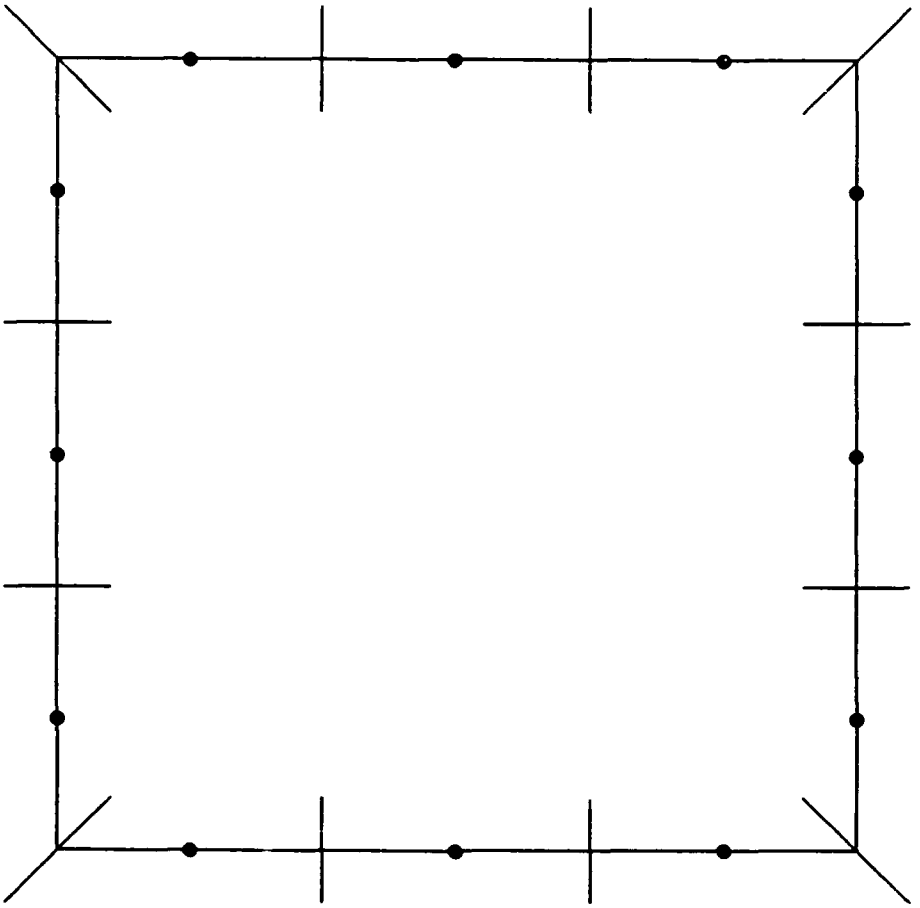


Figure 2: Typical Square (L=1) Mesh with 12 Elements

Number of Elements N	Inverse of Condition Number $1/\kappa$		
	L=1	L=10	L=100
40	4.8×10^{-2}	2.0×10^{-3}	9.1×10^{-5}
80	2.4×10^{-2}	9.5×10^{-4}	4.7×10^{-5}
160	1.1×10^{-2}	4.5×10^{-4}	2.3×10^{-5}
320	5.4×10^{-3}	2.1×10^{-4}	1.1×10^{-5}

Table 1: Condition Number for Test Cases

2.2 Direct Solver Results

We used the direct solver SLNPD from Brebbia [6]. This routine uses direct Gauss elimination with row interchanges to solve unsymmetric, non-positive definite matrices. (Row interchanges are only employed when the magnitude of the diagonal entry is less than 10^{-7}.) As is well known [8], it requires $N^3/3$ floating point operations (flops)[3] to solve an $N \times N$ matrix system. Aside from the storage required for the original (full) matrix and the forcing vector, the routine requires no additional memory.

In choosing a convergence tolerance for iterative methods, we obviously should not request more accuracy than we can obtain from a direct solution. Table 2 shows the 2-norm ($\sqrt{\mathbf{r}^T\mathbf{r}}$) of the residual $\mathbf{r}$

$$\mathbf{b} - \mathbf{A}\mathbf{x} = \mathbf{r} \tag{4}$$

computed with double precision from the direct solution results. Notice that the residual norm is extremely small for the direct solution.

2.3 Least-Squares Conjugate Gradient Results

As a basis for comparison, table 3 shows the ratio of the flops[4] for the least-squares conjugate gradient method to the flops for a direct solution. (The least squares conjugate gradient algorithm has been successfully used in numerous other applications at University of Michigan [7].) Convergence is assumed to occur when the norm of the residual is reduced by a factor (ϵ_{tol}) of 10^{-6}. Notice that the unpreconditioned conjugate gradient method is never competitive with direct solutions for the test problem considered.

[3] "...a flop roughly constitutes the effort of doing a floating point add, a floating point multiply, and a little subscripting." [8], p. 32

[4] Each iteration requires $2N^2$ operations for large N.

Number of Elements N	Norm of Residual $\|\|\mathbf{r}\|\|$ L=1	L=10	L=100
40	2.1×10^{-13}	1.1×10^{-14}	3.0×10^{-15}
80	4.9×10^{-14}	4.4×10^{-15}	3.4×10^{-15}
160	8.0×10^{-13}	5.5×10^{-14}	2.1×10^{-14}
320	2.0×10^{-12}	2.0×10^{-13}	6.6×10^{-14}

Table 2: Residual Error for Test Cases

Number of Elements N	Iterative/Direct flops L=1	L=10	L=100
40	1.95	3.00	5.55
80	2.03	4.65	—
160	2.06	—	—
320	1.61	—	—

Table 3: Conjugate Gradient Results for Test Cases

3 GMRES Solver

We implement the preconditioned GMRES algorithm with restarts as described in Shakib, Hughes, and Johan [3] (see Figure 3, which is reproduced directly from their paper). Appendix A contains the GMRES subroutine listing. The preconditioned system is assumed to have the form

$$(\mathbf{L}^{-1}\mathbf{A}\mathbf{U}^{-1})(\mathbf{U}\mathbf{x}) = (\mathbf{L}^{-1}\mathbf{b}) \tag{5}$$

In Figure 3, k is the number of vectors we use before restarting the algorithm (initiating a new GMRES cycle). If $k = N$, then the algorithm will converge in one cycle. We also must supply a convergence tolerance (ϵ_{tol}) for the norm of the preconditioned residual, and a maximum number of cycles (l_{max}).

We examined the performance of the GMRES algorithm with no preconditioner (**L**=**U**=**I**) and with a partial **L U** factorization. Because of the nature of the boundary integral technique, it seems reasonable to expect that the points which are physically closest to point i on the mesh will have the dominant effect on the solution at point i. We therefore select a number of neighboring points m to include in a banded approximation to **A** (which has the same structure as a finite element mesh). This approximate **A** matrix $\tilde{\mathbf{A}}$ is then decomposed to form **L U**. A diagonal preconditioner corresponds to m=0. Figure 4 shows a schematic of the banded factorization scheme for m=1 and $m = 2$. Note that both **L** and **U** are now tightly banded matrices, and it is possible to take advantage of this fact in storing and inverting them.

The number of floating point operations required to solve the system of equations obviously depends on the number of cycles l required, the number of vectors k chosen, and the size of the band m chosen for the preconditioner. If the system has no preconditioner, it requires roughly $(k + 1)N^2$ flops for each of the l cycles. (This figure neglects the costs associated with all dot products, the Q-R algorithm, solving for y, and updating the solution.) If the procedure terminates within a cycle (with less than k vectors used), the number of flops will be less. If the system is preconditioned, then we need $(k + 1)(N^2 + 2mN)$ flops which is of the same order *so long as m does not need to grow with N.*

The storage required for the GMRES algorithm is modest so long as k remains small. We must store k vectors of length N for the iterations, plus approximately k^2 entries in the h matrix. In addition, the two preconditioner matrices each require mN storage spaces.

Box 1 - Preconditioned GMRES Algorithm.

Given $\boldsymbol{A}$, $\boldsymbol{b}$, $\boldsymbol{L}$, $\boldsymbol{U}$, k, ε_{tol} and $l_{\max}$, proceed as follows:

(Initialization)

$\boldsymbol{b} \leftarrow \boldsymbol{L}^{-1}\boldsymbol{b}$

$\varepsilon = \varepsilon_{\text{tol}} \|\boldsymbol{b}\|$

$\boldsymbol{x} = \boldsymbol{0}$

(GMRES cycles)

For $l = 1, \ldots, l_{\max}$

$\boldsymbol{u}_1 = \boldsymbol{b} - \boldsymbol{L}^{-1}\boldsymbol{A}\boldsymbol{U}^{-1}\boldsymbol{x}$

$\bar{e}_1 = \|\boldsymbol{u}_1\|$

$\boldsymbol{u}_1 \leftarrow \dfrac{\boldsymbol{u}_1}{\|\boldsymbol{u}_1\|}$

(GMRES iteration)

For $i = 1, \ldots, k$

$\boldsymbol{u}_{i+1} = \boldsymbol{L}^{-1}\boldsymbol{A}\boldsymbol{U}^{-1}\boldsymbol{u}_i$

(Modified Gram-Schmidt orthogonalization)

For $j = 1, \ldots, i$

$\beta_{i+1,j} = (\boldsymbol{u}_{i+1}, \boldsymbol{u}_j)$

$\boldsymbol{u}_{i+1} \leftarrow \boldsymbol{u}_{i+1} - \beta_{i+1,j}\boldsymbol{u}_j$

(End modified Gram-Schmidt orthogonalization)

$\bar{\boldsymbol{h}}^{(i)} = \{\beta_{i+1,1}, \ldots, \beta_{i+1,i}, \|\boldsymbol{u}_{i+1}\|\}^T$

$\boldsymbol{u}_{i+1} \leftarrow \dfrac{\boldsymbol{u}_{i+1}}{\|\boldsymbol{u}_{i+1}\|}$

(Q-R algorithm)

For $j = 1, \ldots, i-1$

$$\begin{Bmatrix} \bar{h}_j^{(i)} \\ \bar{h}_{j+1}^{(i)} \end{Bmatrix} \leftarrow \begin{bmatrix} c_j & s_j \\ -s_j & c_j \end{bmatrix} \begin{Bmatrix} \bar{h}_j^{(i)} \\ \bar{h}_{j+1}^{(i)} \end{Bmatrix}$$

Figure 3: Preconditioned GMRES Algorithm (continued on next page)

(End j loop)

$r = \sqrt{\left(\bar{h}_i^{(i)}\right)^2 + \left(\bar{h}_{i+1}^{(i)}\right)^2}$

$c_i = \dfrac{\bar{h}_i^{(i)}}{r}$

$s_i = \dfrac{\bar{h}_{i+1}^{(i)}}{r}$

$\bar{h}_i^{(i)} \leftarrow r$

$\bar{h}_{i+1}^{(i)} \leftarrow 0$

$\bar{e}_{i+1} = -s_i \bar{e}_i$

$\bar{e}_i \leftarrow c_i \bar{e}_i$

(End Q-R algorithm)

Convergence check: **If** $|\bar{e}_{i+1}| \leq \varepsilon$, **Exit** i loop

(End GMRES iteration)

Solve for $\boldsymbol{y}$:

$$\begin{bmatrix} \bar{h}_1^{(1)} & \cdots & \bar{h}_1^{(i-1)} & \bar{h}_1^{(i)} \\ 0 & \ddots & \vdots & \vdots \\ \vdots & \ddots & \bar{h}_{i-1}^{(i-1)} & \bar{h}_{i-1}^{(i)} \\ 0 & \cdots & 0 & \bar{h}_i^{(i)} \end{bmatrix} \begin{Bmatrix} y_1 \\ \vdots \\ y_{i-1} \\ y_i \end{Bmatrix} = \begin{Bmatrix} \bar{e}_1 \\ \vdots \\ \bar{e}_{i-1} \\ \bar{e}_i \end{Bmatrix}$$

Update solution:

$$\boldsymbol{x} \leftarrow \boldsymbol{x} + \sum_{j=1}^{i} y_j \boldsymbol{u}_j$$

Convergence check: **If** $|\bar{e}_{i+1}| \leq \varepsilon$, **Exit** l loop

(End GMRES cycles)

$\boldsymbol{x} \leftarrow \boldsymbol{U}^{-1}\boldsymbol{x}$

Return

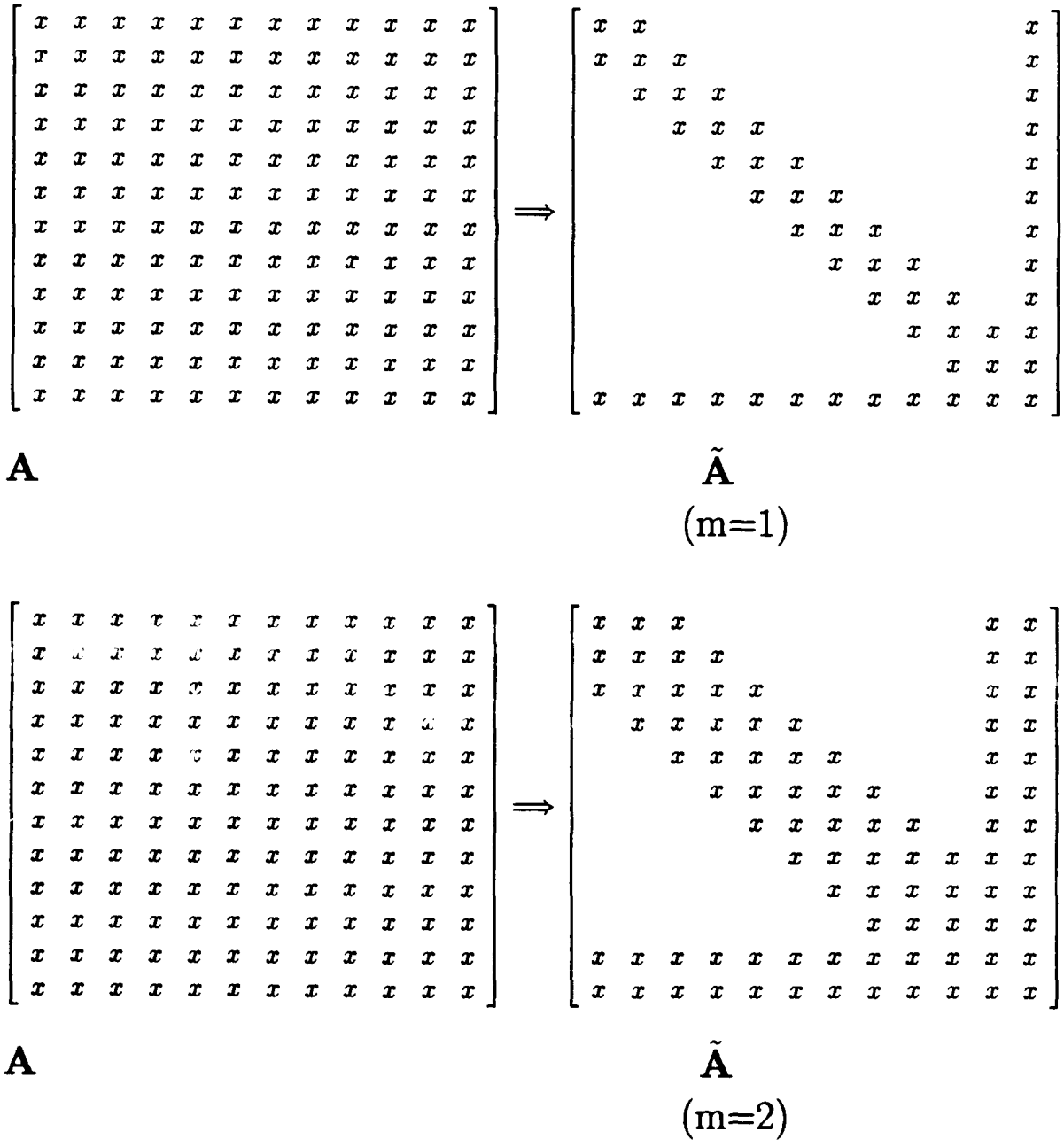


Figure 4: Banded Factorization Scheme

4 GMRES Results

4.1 No Preconditioner

Tables 4 to 7 show the results of applying the GMRES algorithm without preconditioning to the test problem considered. The convergence tolerance ϵ_{tol} was set at 10^{-6}. Note that if the maximum number of cycles l_{max} is greater than $N/(3(k+1))$, then the GMRES algorithm with one cycle requires *more* flops than the direct solver. Consequently, we set that as an upper limit for our tests for l. The notation "—" in the table indicates that the solution did not converge within the allowed cycle limits. In addition, the table shows the ratio of the number of iterative flops[5] to the direct flops. Notice that the GMRES method becomes more competitive with the direct solution as the matrices become larger. Also it is clear that the minimum in flops occurs when k is large enough that the routine does not restart, i.e. when only one cycle is used. Unfortunately, this corresponds to the most storage-intensive use of the algorithm. Finally, we observe that the unpreconditioned algorithm is not significantly faster than the direct solve.

[5] Assumed to be $(l-1)(k+1)N^2+(i+1)N^2$ where i is the number of iterations in the last cycle.

# of vectors	L=1			L=10			L=100		
k	l	i	I/D	l	i	I/D	l	i	I/D
1 - 7	—	—	>1.0	—	—	>1.0	—	—	>1.0
8	2	1	0.83	—	—	>1.0	—	—	>1.0
9	1	9	0.75	2	8	1.42	—	—	>1.0
10	1	9	0.75	2	3	1.13	2	10	1.65
11	1	9	0.75	1	11	0.90	2	2	1.13
12 - N	1	9	0.75	1	11	0.90	1	12	0.98

Table 4: GMRES with No Preconditioner, N=40

# of vectors	L=1			L=10			L=100		
k	l	i	I/D	l	i	I/D	l	i	I/D
1 - 5	—	—	>1.0	—	—	>1.0	—	—	>1.0
6	4	6	1.05	—	—	>1.0	—	—	>1.0
7	4	1	0.98	—	—	>1.0	—	—	>1.0
8	3	6	0.94	—	—	>1.0	—	—	>1.0
9	2	9	0.75	—	—	>1.0	—	—	>1.0
10	2	6	0.68	—	—	>1.0	—	—	>1.0
11	2	3	0.60	—	—	>1.0	—	—	>1.0
12	2	1	0.64	—	—	>1.0	—	—	>1.0
13	1	14	0.60	—	—	>1.0	—	—	>1.0
14	1	14	0.56	—	—	>1.0	—	—	>1.0
15	1	14	0.56	—	—	>1.0	—	—	>1.0
16	1	14	0.56	2	10	1.05	—	—	>1.0
17	1	14	0.56	2	10	1.09	—	—	>1.0
18	1	14	0.56	2	11	1.16	—	—	>1.0
19	1	14	0.56	2	3	0.90	—	—	>1.0
20	1	14	0.56	1	20	0.79	2	20	1.58
21	1	14	0.56	1	20	0.79	2	15	1.42
22	1	14	0.56	1	20	0.79	2	15	1.46
23	1	14	0.56	1	20	0.79	2	11	1.35
24	1	14	0.56	1	20	0.79	2	11	1.39
25	1	14	0.56	1	20	0.79	2	6	1.24
26	1	14	0.56	1	20	0.79	—	—	>1.00
27 - N	1	14	0.56	1	20	0.79	1	27	1.05

Table 5: GMRES with No Preconditioner, N=80

# of vectors	L=1			L=10			L=100		
k	l	i	I/D	l	i	I/D	l	i	I/D
1 - 6	—	—	>1.0	—	—	>1.0	—	—	>1.0
7	6	3	0.83	—	—	>1.0	—	—	>1.0
8	5	1	0.71	—	—	>1.0	—	—	>1.0
9	4	5	0.68	—	—	>1.0	—	—	>1.0
10	3	10	0.62	—	—	>1.0	—	—	>1.0
11	3	5	0.56	—	—	>1.0	—	—	>1.0
12	3	1	0.53	—	—	>1.0	—	—	>1.0
13	2	13	0.53	—	—	>1.0	—	—	>1.0
14	2	12	0.53	4	11	1.07	—	—	>1.0
15	2	10	0.51	4	7	1.05	—	—	>1.0
16	2	7	0.47	3	15	0.94	—	—	>1.0
17	2	3	0.41	3	12	0.92	—	—	>1.0
18	2	3	0.43	3	11	0.94	—	—	>1.0
19	2	1	0.41	3	7	0.90	—	—	>1.0
20	1	20	0.39	3	3	0.86	—	—	>1.0
21	1	20	0.39	2	19	0.79	—	—	>1.0
22	1	20	0.39	2	18	0.79	—	—	>1.0
23	1	20	0.39	2	18	0.81	—	—	>1.0
24	1	20	0.39	2	16	0.79	—	—	>1.0
25	1	20	0.39	2	14	0.77	—	—	>1.0
26	1	20	0.39	2	13	0.77	—	—	>1.0
27	1	20	0.39	2	13	0.79	—	—	>1.0
28	1	20	0.39	2	10	0.75	—	—	>1.0
29	1	20	0.39	2	8	0.73	—	—	>1.0
30	1	20	0.39	2	6	0.71	—	—	>1.0

Table 6: GMRES with No Preconditioner, N=160 (continued on next page)

# of vectors	L=1			L=10			L=100		
k	l	i	I/D	l	i	I/D	l	i	I/D
31	1	20	0.39	2	4	0.69	—	—	>1.0
32	1	20	0.39	1	32	0.62	—	—	>1.0
33	1	20	0.39	1	32	0.62	—	—	>1.0
34	1	20	0.39	1	32	0.62	2	30	1.24
35	1	20	0.39	1	32	0.62	2	30	1.26
36	1	20	0.39	1	32	0.62	2	22	1.13
37	1	20	0.39	1	32	0.62	2	22	1.14
38	1	20	0.39	1	32	0.62	2	19	1.11
39	1	20	0.39	1	32	0.62	2	18	1.11
40	1	20	0.39	1	32	0.62	2	17	1.11
41	1	20	0.39	1	32	0.62	2	18	1.14
42	1	20	0.39	1	32	0.62	2	5	0.92
43 - N	1	20	0.39	1	32	0.62	1	43	0.82

4.2 Banded Preconditioner

We initially tested the preconditioner on the L=1 cases, which are the most well-conditioned systems, with disappointing results. The diagonal preconditioner (m=0) had essentially no effect, since the diagonals of the original matrix are almost all equal. A full **LU** decomposition (m=$N-1$) caused the iteration to converge in exactly one step, as expected. However, other values of m actually degraded the performance of the algorithm. For instance, m=1 (a psuedo-tridiagonal preconditioner) caused slower convergence for N = 40,80, and 160. Larger values of m also caused slower convergence than no preconditioner. Checking the condition number of the preconditioned matrices showed that it was, in fact, worse than for the original matrices. One possible explanation for this is that the condition number of the original matrices is already so good that little heuristic improvement is possible.

As a second check, we examine the effect of the preconditioner on the L=100, N=80 matrix, which converges poorly and has a condition number of 4.7×10^{-5} in its original form. Using a preconditioner with m=1 to 10, we find no convergence at all in the allowed number of iterations.

5 Conclusions

The GMRES technique requires somewhat fewer flops than a direct solve as the full, unsymmetric boundary integral matrices we tested become large. However, the ratio was only .23 at best. The heuristic preconditioner we tested did not improve the convergence. In general, unless a good preconditioner or a good approximation to the solution **x** is available, the additional storage and uncertainty associated with the iterative solver make it impractical for full matrices.

# of vectors	L=1			L=10			L=100		
k	l	i	I/D	l	i	I/D	l	i	I/D
1 - 5	—	—	>1.0	—	—	>1.0	—	—	>1.0
6	15	3	0.96	—	—	>1.0	—	—	>1.0
7	10	4	0.72	—	—	>1.0	—	—	>1.0
8	7	6	0.57	—	—	>1.0	—	—	>1.0
9	5	8	0.46	—	—	>1.0	—	—	>1.0
10	4	17	0.38	—	—	>1.0	—	—	>1.0
11	4	3	0.38	—	—	>1.0	—	—	>1.0
12	3	8	0.33	8	10	0.96	—	—	>1.0
13	3	5	0.32	7	9	0.88	—	—	>1.0
14	3	1	0.30	6	10	0.81	—	—	>1.0
15	2	13	0.28	5	14	0.74	—	—	>1.0
16	2	13	0.29	5	6	0.70	—	—	>1.0
17	2	12	0.29	4	14	0.65	—	—	>1.0
18	2	8	0.26	4	11	0.65	—	—	>1.0
19	2	5	0.24	4	8	0.65	—	—	>1.0
20	2	5	0.25	3	19	0.58	—	—	>1.0
21	2	3	0.24	3	17	0.58	—	—	>1.0
22	2	1	0.23	3	15	0.58	—	—	>1.0
23	1	23	0.23	3	12	0.57	—	—	>1.0
24	1	23	0.23	3	7	0.54	—	—	>1.0
25	1	23	0.23	3	3	0.52	—	—	>1.0
26	1	23	0.23	2	25	0.50	—	—	>1.0
27	1	23	0.23	2	24	0.50	—	—	>1.0
28	1	23	0.23	2	23	0.50	—	—	>1.0
29	1	23	0.23	2	23	0.51	—	—	>1.0
30	1	23	0.23	2	22	0.51	—	—	>1.0
31	1	23	0.23	2	21	0.51	—	—	>1.0
32	1	23	0.23	2	19	0.50	—	—	>1.0
33	1	23	0.23	2	17	0.49	—	—	>1.0
34	1	23	0.23	2	16	0.49	—	—	>1.0
35	1	23	0.23	2	14	0.48	—	—	>1.0

Table 7: GMRES with No Preconditioner, N=320, (continued next page)

# of vectors	L=1			L=10			L=100		
k	l	i	I/D	l	i	I/D	l	i	I/D
36	1	23	0.23	2	13	0.48	—	—	>1.0
37	1	23	0.23	2	11	0.47	—	—	>1.0
38	1	23	0.23	2	9	0.46	—	—	>1.0
39	1	23	0.23	2	5	0.43	—	—	>1.0
40	1	23	0.23	2	5	0.44	—	—	>1.0
41	1	23	0.23	1	41	0.39	—	—	>1.0
42	1	23	0.23	1	41	0.39	—	—	>1.0
43	1	23	0.23	1	41	0.39	3	42	1.23
44	1	23	0.23	1	41	0.39	3	42	1.25
45	1	23	0.23	1	41	0.39	3	33	1.18
46	1	23	0.23	1	41	0.39	3	34	1.21
47	1	23	0.23	1	41	0.39	3	30	1.19
48	1	23	0.23	1	41	0.39	3	27	1.18
49	1	23	0.23	1	41	0.39	2	49	0.94
50	1	23	0.23	1	41	0.39	3	10	1.06
51	1	23	0.23	1	41	0.39	2	50	0.97
52	1	23	0.23	1	41	0.39	2	50	0.98
53	1	23	0.23	1	41	0.39	2	48	0.97
54	1	23	0.23	1	41	0.39	2	41	0.91
55	1	23	0.23	1	41	0.39	2	42	0.93
56	1	23	0.23	1	41	0.39	2	42	0.94
57	1	23	0.23	1	41	0.39	2	34	0.87
58	1	23	0.23	1	41	0.39	2	34	0.88
59	1	23	0.23	1	41	0.39	2	33	0.88
60	1	23	0.23	1	41	0.39	2	33	0.89
61	1	23	0.23	1	41	0.39	2	24	0.82
62	1	23	0.23	1	41	0.39	2	23	0.82
63	1	23	0.23	1	41	0.39	2	21	0.81
64	1	23	0.23	1	41	0.39	2	22	0.82
65	1	23	0.23	1	41	0.39	2	13	0.75
66	1	23	0.23	1	41	0.39	2	11	0.74
67	1	23	0.23	1	41	0.39	2	13	0.77
68 - N	1	23	0.23	1	41	0.39	1	68	0.65

References

[1] F. Angeleri, V. Sonnad, and K. J. Bathe. Studies of finite element procedures - An evaluation of preconditioned iterative solvers, *IBM Report KGN-189*, IBM Corporation, Kingston, NY 12401 (1988).

[2] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, *SIAM J. Sci. Stat. Comp.* **7** (1986) 856-869.

[3] F. Shakib, T. J. R. Hughes, and Z. Johan. A multi-element group preconditioned GMRES algorithm for nonsymmetric systems arising in finite element analysis, to appear in *Computer Methods in Applied Mechanics and Engineering.*

[4] G. Brussino and V. Sonnad. A comparison of preconditioned iterative techniques for sparse, indefinite, unsymmetric systems of linear equations, *IBM Report KGN-83*, IBM Corporation, Kingston, NY 12401 (1986).

[5] G. Brussino and V. Sonnad. A comparison of preconditioned iterative techniques for sparse, indefinite, unsymmetric systems of linear equations: Part II, *IBM Report KGN-189*, IBM Corporation, Kingston, NY 12401 (1987).

[6] C. A. Brebbia. *The Boundary Element Method for Engineers*, Pentech Press, London, 1978.

[7] W. W. Schultz, *personal communication.*

[8] G. H. Golub and C. F. Van Loan. *Matrix Computations*, The Johns Hopkins University Press, Baltimore, MD, 1983.

A GMRES Program Listings

The routines listed below are available on the Apollo network. The main routine is in //kilroy/users/olson/gmres/gmres.ftn and the utility routines are in //kilroy/users/olson/gmres/util.ftn.

```
      SUBROUTINE GMRES (AK,X,B,AU,AL,Z,U,EBAR,BETA,HBAR,C,S,Y,
     1                  N,NX,K,KP1,TOL,MAX,NITER,NK)
C
C     USE THE GENERALIZED MINIMAL RESIDUAL METHOD TO SOLVE
C     AK*X=R, WHERE AK IS A FULL UNSYMMETRIC N X N
C     MATRIX  (SEE SHAKIB, HUGHES, JOHAN, P17)
C
C     INPUTS:
C     AK (N,N)     = STIFFNESS MATRIX
C     X(N)         = SOLUTION VECTOR
C     B(N)         = LOAD VECTOR
C     AU(N,N)      = UPPER TRIANGULAR
C                    PRECONDITIONING MATRIX
C                    (STORED AS A FULL MATRIX FOR TESTING PURPOSES)
C     AL(N,N)      = LOWER TRIANGULAR
C                    PRECONDITIONING MATRIX
C                    (STORED AS A FULL MATRIX FOR TESTING PURPOSES)
C     Z(N)         = WORKING VECTOR
C     U(N,K+1)     = WORKING VECTOR
C     EBAR(K+1)    = WORKING VECTOR
C     BETA(K+1,K)  = WORKING VECTOR
C     HBAR(K+1,K)  = WORKING VECTOR
C     C(K)         = WORKING VECTOR (COSINES IN Q-R)
C     S(K)         = WORKING VECTOR (SINES IN Q-R)
C     Y(K)         = WORKING VECTOR (WEIGHTS FOR U)
C     N            = ACTUAL DIMENSION OF STIFFNESS MATRIX
C     NX           = STATED DIMENSION OF STIFFNESS MATRIX
C     K            = NUMBER OF VECTORS IN GMRES ITERATION
C     KP1          = K+1
C     TOL          = TOLERANCE FOR CONVERGENCE CHECK
C     MAX          = MAXIMUM NUMBER OF CYCLES ALLOWED
C
C     OUTPUTS:
```

```
C     X(N)         = SOLUTION VECTOR
C     NITER        = ACTUAL NUMBER OF CYCLES USED
C     NK           = NUMBER OF VECTORS REQUIRED IN LAST ITERATION
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION AK(NX,*),X(*),B(*),AU(NX,*),AL(NX,*),Z(*),
     1          U(NX,*),EBAR(*),BETA(KP1,*),HBAR(KP1,*),C(*),S(*),Y(*)
C
C     INITIALIZE VARIABLES
C
      RZERO=1.0E-20
      DO 10 I=1,N
          DO 5 J=1,I-1
               B(I)=B(I) - AL(I,J)*B(J)
    5     CONTINUE
        B(I)=B(I)/AL(I,I)
   10 CONTINUE
        CALL DOT (B,B,N,EPS)
        EPS=TOL*SQRT(EPS)
        CALL ZERO (X,N)
        CALL ZERO (Z,N)
        NMATEL=N*KP1
        CALL ZERO (U,NMATEL)
        CALL ZERO (EBAR,KP1)
        NMATEL=K*KP1
        CALL ZERO (BETA,NMATEL)
        CALL ZERO (HBAR,NMATEL)
        CALL ZERO (C,K)
        CALL ZERO (S,K)
        CALL ZERO (Y,K)
C
C     ITERATE AT MOST MAX TIMES (GMRES CYCLES)
C
      DO 800 L=1,MAX
           NITER=L
           CALL ZERO (Z,N)
           CALL PREMUL (AL,AK,AU,X,U(1,1),N,NX,Z)
           DO 310 INDEX=1,N
                U(INDEX,1)=B(INDEX)-Z(INDEX)
```

```
  310       CONTINUE
            CALL DOT (U(1,1),U(1,1),N,EBAR(1))
            EBAR(1)=SQRT(EBAR(1))
            DO 320 INDEX=1,N
                 U(INDEX,1)=U(INDEX,1)/EBAR(1)
  320       CONTINUE
C
C           GMRES ITERATION
C
            DO 400 I=1,K
                 NK=I
                 CALL ZERO (U(1,I+1),N)
                 CALL PREMUL (AL,AK,AU,U(1,I),Z,N,NX,U(1,I+1))
C
C                MODIFIED GRAM-SCHMIDT ORTHOGONALIZATION
C
                 DO 330 J=1,I
                      CALL DOT (U(1,I+1),U(1,J),N,BETA(I+1,J))
                      DO 325 INDEX=1,N
                           U(INDEX,I+1)=U(INDEX,I+1) -
     1                                  BETA(I+1,J)*U(INDEX,J)
  325                 CONTINUE
  330            CONTINUE
C
C                END MODIFIED GRAM-SCHMIDT ORTHOGONALIZATION
C
                 CALL DOT (U(1,I+1),U(1,I+1),N,TEMP)
                 TEMP=SQRT(TEMP)
                 DO 340 INDEX=1,I
                      HBAR(INDEX,I)=BETA(I+1,INDEX)
  340            CONTINUE
                 HBAR(I+1,I)=TEMP
                 IF (ABS(TEMP).GT.1.0E-20) TEMP=1.0/TEMP
                 DO 350 INDEX=1,N
                      U(INDEX,I+1)=U(INDEX,I+1)*TEMP
  350            CONTINUE
C
C                Q-R ALGORITHM
C
```

```
                DO 360 J=1,I-1
                     TEMP1=C(J)*HBAR(J,I) + S(J)*HBAR(J+1,I)
                     TEMP2=-S(J)*HBAR(J,I) + C(J)*HBAR(J+1,I)
                     HBAR(J,I)=TEMP1
                     HBAR(J+1,I)=TEMP2
  360           CONTINUE
                R=SQRT(HBAR(I,I)*HBAR(I,I) + HBAR(I+1,I)*HBAR(I+1,I))
                C(I)=HBAR(I,I)/R
                S(I)=HBAR(I+1,I)/R
                HBAR(I,I)=R
                HBAR(I+1,I)=0.0
                EBAR(I+1)=-S(I)*EBAR(I)
                EBAR(I)=C(I)*EBAR(I)
C
C               END Q-R ALGORITHM
C               CHECK FOR CONVERGENCE
C
                IF (ABS(EBAR(I+1)).LT.EPS) GO TO 500
C
C               END GMRES ITERATION?
C
  400       CONTINUE
  500       CONTINUE
C
C           SOLVE FOR \
C
            CALL ZERO (Y,K)
            DO 510 INDEX=NK,1,-1
                  Y(INDEX)=EBAR(INDEX)
                  DO 505 IND=INDEX+1,K
                       Y(INDEX)=Y(INDEX) - Y(IND)*HBAR(INDEX,IND)
  505             CONTINUE
                  IF (ABS(HBAR(INDEX,INDEX)).LT.RZERO) THEN
                        Y(INDEX)=0.0
                  ELSE
                       Y(INDEX)=Y(INDEX)/HBAR(INDEX,INDEX)
                  ENDIF
  510       CONTINUE
C
```

```
C          UPDATE SOLUTION
C
           DJ 650 J=1,K
                DO 600 INDEX=1,N
                     X(INDEX)=X(INDEX)+ Y(J)*U(INDEX,J)
  600           CONTINUE
  650      CONTINUE
C
C          CHECK FOR CONVERGENCE
C
           IF (ABS(EBAR(NK+1)).LT.EPS) GO TO 900
C
C          END GMRES CYCLES, FAILURE TO CONVERGE
C
  800 CONTINUE
      WRITE (*,*) 'SOLUTION DID NOT CONVERGE IN',MAX,'CYCLES'
      WRITE (*,*) 'X=',(X(I),I=1,N)
      STOP
C
C     SOLUTION CONVERGED
C
  900 CONTINUE
      DO 920 I=N,1,-1
           DO 910 J=I+1,N
                X(I)=X(I) - AU(I,J)*X(J)
  910       CONTINUE
           X(I)=X(I)/AU(I,I)
  920 CONTINUE
      RETURN
      END



      SUBROUTINE DOT (W1,W2,N,TEMP)
C
C     FIND THE DOT PRODUCT OF TWO VECTORS
C
C     INPUTS:
```

```
C     W1(N) = FIRST VECTOR
C     W2(N) = SECOND VECTOR
C     N     = LENGTH OF VECTORS
C
C     OUTPUTS:
C     TEMP  = W*W
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION W1(N),W2(N)
C
      TEMP=0.0
      DO 100 I=1,N
           TEMP=TEMP + W1(I)*W2(I)
  100 CONTINUE
      RETURN
      END
C
      SUBROUTINE ZERO (X,N)
C
C     SET X TO ZERO
C
C     INPUTS:
C     X(N)  =  VECTOR TO BE ZEROED
C     N     =  SIZE OF VECTOR
C
C     OUTPUTS:
C     X     = 0
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(N)
C
      DO 100 I=1,N
           X(I)=0.0
  100 CONTINUE
C
      RETURN
      END
C
      SUBROUTINE PREMUL (AL,AK,AU,X,W,N,NX,Z)
```

```
C
C     FORM Z=(AL)^-1 * AK * (AU)^-1 * X
C
C     INPUTS:
C     AL(N,N) = LOWER DIAGONAL PRECONDITIONING MATRIX
C     AK(N,N) = STIFFNESS MATRIX
C     AU(N,N) = UPPER DIAGONAL PRECONDITIONING MATRIX
C     X(N)    = SOLUTION VECTOR
C     W(N)    = WORKING VECTOR
C     N       = SIZE OF ARRAYS
C     NX      = DECLARED SIZE OF ARRAYS
C
C     OUTPUTS:
C     Z(N)    = AL^-1 * AK * AU^-1 * X
C
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION AL(NX,*),AK(NX,*),AU(NX,*),X(*),W(*),Z(*)
C
C     AU^-1 * X
C
      DO 20 I=N,1,-1
            W(I)=X(I)
            DO 10 J=I+1,N
                  W(I)=W(I) - AU(I,J)*W(J)
   10       CONTINUE
            W(I)=W(I)/AU(I,I)
   20 CONTINUE
C
C     AK* AU^-1 *X
C
      DO 100 I=1,N
            Z(I)=0.0
            DO 50 J=1,N
                 Z(I)=Z(I) + AK(I,J)*W(J)
   50       CONTINUE
  100 CONTINUE
C
C     AL^-1 * AK * AU^-1 *X
```

```
C
      DO 200 I=1,N
          DO 150 J=1,I-1
               Z(I)=Z(I) - AL(I,J)*Z(J)
  150     CONTINUE
      Z(I)=Z(I)/AL(I,I)
  200 CONTINUE
C
      RETURN
      END
```