

**Domain Decomposition for Elliptic  
Partial Differential Equations with  
Neumann Boundary Conditions**

by

Ruth Gonzalez

and

M.F. Wheeler

Technical Report 87-10, May 1987



Domain Decomposition for Elliptic Partial Differential Equations  
with Neumann Boundary Conditions

Ruth Gonzalez\* and Mary Fanett Wheeler\*\*

1. *Introduction.* Discretization of a self-adjoint elliptic partial differential equation by finite differences or finite elements yields a large, sparse, symmetric system of equations,  $Ax = b$ . We use the preconditioned conjugate gradient method with domain decomposition to develop an effective, vectorizable preconditioner which is suitable for solving large two-dimensional problems on vector and parallel machines.

The convergence of the preconditioned conjugate gradient method is determined by the condition number of the matrix  $M^{-1}A$ , where  $A$  and  $M$  correspond to the matrix for the discretized differential equation and to the preconditioning matrix, respectively. By appropriately preconditioning the system  $Ax = b$  we can significantly reduce the computational effort that is required in solving for  $x$ .

The basic approach in domain decomposition techniques is to break up the domain of integration into many pieces, solve the appropriate equation on each piece, then somehow construct the global solution from these local solutions. Bramble, Pasciak, and Schatz [1, 2] have defined preconditioners for discretized, self-adjoint elliptic partial differential equations with Dirichlet boundary conditions using a domain decomposition technique. In their algorithm every conjugate gradient iteration involves solving exactly on each subdomain two linear systems by fast Fourier transforms. The use of FFT's requires that the coefficients of the elliptic operator be nearly constant on each subdomain.

In this paper, we modify the Bramble-Pasciak-Schatz preconditioner by removing the restriction that the subdomain problems be solved exactly. Instead, we apply one iteration of the block preconditioning technique, MINV, developed by Concus, Golub, and Meurant [4]. The

---

\*Exxon Production Research Co., Houston, Texas.

\*\*Mathematical Sciences Department, Rice University, Houston, Texas. Partial support of research under NSF grant 2320.

robustness of the MINV preconditioner allows for highly varying coefficients within each sub-domain. Moreover, the coupling of the inherently recursive MINV algorithm with domain decomposition allows vectorization and parallel implementation of the global preconditioner across sub-domains.

For simplicity, we consider a five-point cell-centered finite difference discretization of the boundary value problem

$$\begin{aligned} -\nabla \cdot a(x, y) \nabla u &= f(x, y), & (x, y) \in \Omega, \\ a \nabla u \cdot n &= 0, & (x, y) \in \partial \Omega, \end{aligned} \quad (1)$$

where  $0 < a_0 \leq a(x, y) \leq a_1$ ,  $f$  satisfies a compatibility assumption,  $\Omega$  is a rectangle in  $\mathbb{R}^2$ , and  $n$  is the outward normal on the boundary of  $\Omega$ . The preconditioner developed here can be generalized to a wider class of pde's and finite element schemes. See Gonzalez [5].

In Section 2 we formulate the procedure. The efficiency of this algorithm in its vectorized form is demonstrated by numerical results in Section 3.

## 2. Notation and Formulation of the Preconditioner

We partition  $\Omega = [0, x_L] \times [0, y_L]$  by  $\delta_x: 0 = x_{\frac{1}{2}} < x_{\frac{3}{2}} < \dots < x_{N_x + \frac{1}{2}} = x_L$  and  $\delta_y: 0 = y_{\frac{1}{2}} < y_{\frac{3}{2}} < \dots < y_{N_y + \frac{1}{2}} = y_L$ . Set  $x_i = (x_{i-\frac{1}{2}} + x_{i+\frac{1}{2}})/2$  and  $y_j = (y_{j-\frac{1}{2}} + y_{j+\frac{1}{2}})/2$ . Denote by  $\hat{\delta}_x$  and  $\hat{\delta}_y$  the partitions defined by the  $x_i$ 's and  $y_j$ 's respectively (see Figure 1). We define two additional partitions  $\Delta x$  and  $\Delta y$  such that  $\Delta x$  and  $\Delta y$  are subsets of  $\hat{\delta}_x$  and  $\hat{\delta}_y$ ; i.e.

$$\begin{aligned} \Delta x: x_1 &= \bar{x}_0 < \bar{x}_1 < \dots < \bar{x}_{N_x} = x_{N_x}, \\ \Delta y: y_1 &= \bar{y}_0 < \bar{y}_1 < \dots < \bar{y}_{N_y} = y_{N_y}. \end{aligned}$$

Define the set of  $\Omega_k$ ,  $k = 1, 2, \dots, N_x N_y$ , to be the subregions which are determined by the tensor product of  $\Delta x$  and  $\Delta y$ . Let the set of  $V_i, i = 1, 2, \dots, N_x N_y$ , be the vertices of the  $\Omega_k$ 's and  $\Gamma_{ij}$  be the segments connecting vertices  $V_i$  and  $V_j$  (see Figure 2). In the above all orderings are assumed to be natural, the  $x$ -component increasing most rapidly.

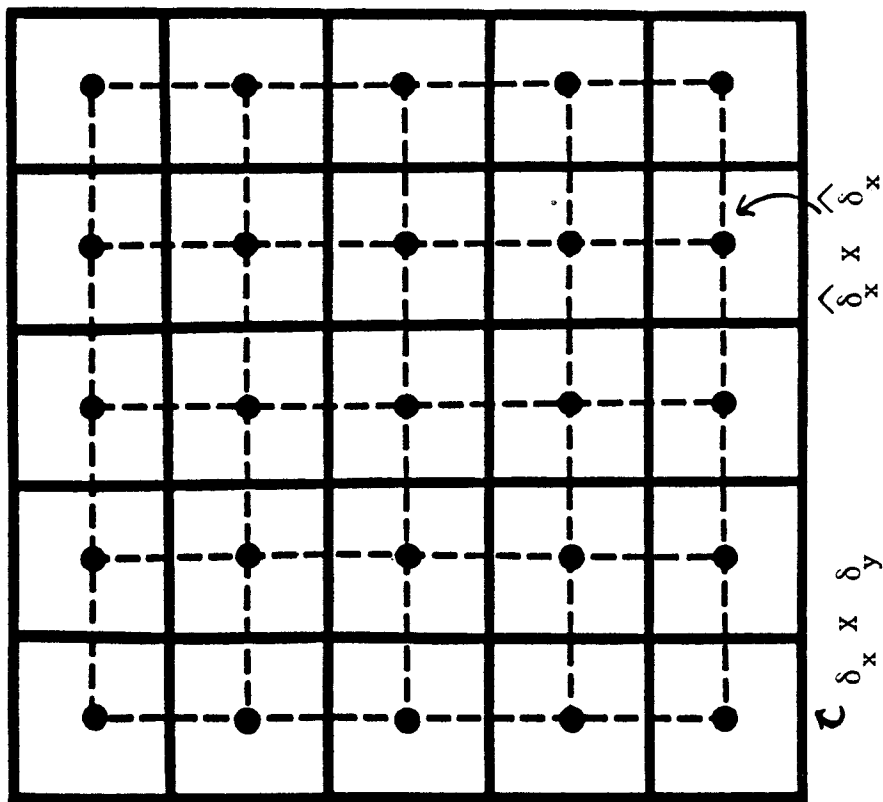


Figure 1 Grids for the Neumann Problem

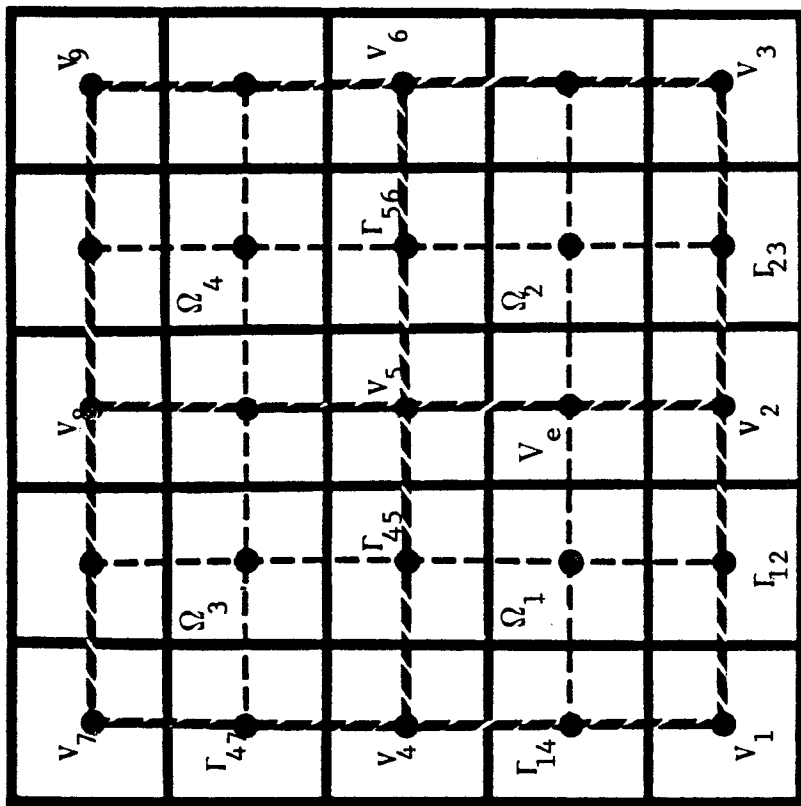


Figure 2 Subdomain Grids for the Neumann Problem

The cell-centered five-point finite difference approximation to (1) is defined by

$$\begin{aligned}
& - \frac{1}{x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}} \left( a_{i+\frac{1}{2},j} \frac{u_{i+1,j} - u_{ij}}{x_{i+1} - x_i} - a_{i-\frac{1}{2},j} \frac{u_{ij} - u_{i-1,j}}{x_i - x_{i-1}} \right) \\
& - \frac{1}{y_{j+\frac{1}{2}} - y_{j-\frac{1}{2}}} \left( a_{i,j+\frac{1}{2}} \frac{u_{i,j+1} - u_{ij}}{y_{j+1} - y_j} - a_{i,j-\frac{1}{2}} \frac{u_{ij} - u_{i,j-1}}{y_j - y_{j-1}} \right) \\
& = f_{ij} ,
\end{aligned} \tag{2}$$

where  $a_{i+\frac{1}{2}} = a(x_{i+\frac{1}{2}}, y_j)$  and similar definitions for  $f$  and  $u$ .

In each iteration of the preconditioned conjugate gradient method [3], we need to solve  $Mz^l = r^l$ , where  $r^l$  is the  $l^{\text{th}}$  residual and  $M$  is the preconditioner. To solve this linear system we compute as follows:

*Step 1:* On each  $\Omega_k$  perform one iteration of MINV to obtain  $U_p$ , where  $U_p$  is the approximate solution to a homogeneous Dirichlet problem. More precisely, apply one iteration of MINV to the system of equations defined by (2) restricted to  $\Omega_k$  with homogeneous boundary conditions on  $\partial\Omega_k$ .

*Step 2:* Compute the residual  $g^l = r^l - A U_p$ .

*Step 3:* Use fast Fourier transforms to compute  $U_e \in H_0^1(\Gamma_k)$  by  $\|U_e\|_{\frac{1}{2}, \Gamma_k}^2 = \langle l_0^{\frac{1}{2}} U_e, U_e \rangle = \langle q^l |_{\Gamma_k}, w \rangle_{\Gamma_k} = \langle w', v' \rangle_{\Gamma_k}$ ,  $w \in H_0^1(\Gamma_k)$ . Here  $\langle \cdot, \cdot \rangle$  denotes  $L^2$  inner product on  $\Gamma_k$ .

*Step 4:* Find  $U_v$  for all  $V_i$  using (2) with right hand side  $g_{\text{average}}^l = \text{average of the residual of the appropriate edges}$ .

*Step 5:* Compute the residual  $\bar{g}^l = r^l - A(U_p + U_v + U_e)$ .

*Step 6:* On each  $\Omega_k$  perform one iteration of MINV to obtain  $U_H$ , where  $U_H$  is the approximate harmonic solution with boundary conditions  $\bar{g}^l$ .

Step 7: Form  $U = U_p + U_H$ .

3. *Numerical Results.* Let  $\Omega = (0, 1) \times (0, 1)$  and  $f = \delta(0, 0) - \delta(1, 1)$ , where  $\delta(\cdot, \cdot)$  denotes the Dirac measure. We consider three test problems with different coefficients  $a(x, y) = a_i(x, y)$  where  $a_1(x, y) \equiv 1$ ,

$$a_2(x, y) = \begin{cases} 1 & \text{if } 0 < x < \frac{1}{2} \\ 10^{-6} & \text{if } \frac{1}{2} < x < 1, \end{cases}$$

and

$$a_3(x, y) = \frac{1}{1 + 100(x^2 + y^2)}.$$

For each of the test problems we use the stopping criteria in the preconditioned conjugate gradient algorithm that the relative residual be less than  $10^{-6}$ . In problem 1 choose  $N_x = N_y$  to be 65 or 129, and  $\bar{N}_x = \bar{N}_y$  to be 2, 4, 8, 16, or 32, with the number of unknowns in each subregion being  $(N_x - 1)/\bar{N}_x$  for the  $x$ -direction and  $(N_y - 1)/\bar{N}_y$  for the  $y$  direction. Table 1 summarizes these numerical results.

Table 1

Iteration and CPU Statistics for  $a(x, y) = a_1(x, y)$

$N_x \times N_y$	$\bar{N}_x \times \bar{N}_y$	Number of Iterations	CPU (seconds)
65 × 65	2 × 2	39	2.25
	4 × 4	33	0.9
	8 × 8	31	0.75
	16 × 16	27	1.1
129 × 129	4 × 4	46	4.0
	8 × 8	39	2.5
	16 × 16	37	2.9
	32 × 32	34	5.3

It is evident from studying Table 1 that for a given  $N_x$  and  $N_y$ , the number of iterations decreases as the number of subregions  $\bar{N}_x$  and  $\bar{N}_y$  increases. It is also apparent that the cpu time initially decreases as the number of subregions increases, but then it increases again. This is

because the vectorized portions of the algorithm compete with one another. The algorithm vectorizes across the total number of subregions when the individual subproblems are solved, but the data movement part of the process vectorizes over the number of  $x$ -variables in a given subdomain (e.g.  $(N_x - 1)/\bar{N}_x$ ). Therefore when the number of total subregions increases, the cpu time needed for the solution of the individual problems decreases; but the cpu time that is needed in moving the data in order to solve these subproblems is increased. It appears that the best compromise is to select  $\bar{N}_x \simeq \sqrt{N_x}$  and  $\bar{N}_y \simeq \sqrt{N_y}$ , so that one phase never dominates the other phase. For  $N_x = N_y = 65$  we would choose  $\bar{N}_x = \bar{N}_y = 8$  and for  $N_x = N_y = 129$  we would choose  $\bar{N}_x = \bar{N}_y = 8$ , even though fewer iterations are needed for other choices of  $N_x$  and  $N_y$ .

Tom Hewitt of CRAY Research optimized some of the code for the CRAY-XMP machine. Table 2 indicates what kind of performance is possible on the XMP with only one processor. The average MFLOP rate for the  $N_x = 129$  by  $N_y = 129$  problem was 90.4 MFLOPS, while the average for an  $N_x = 257$  by  $N_y = 257$  problem was 94.4 MFLOPS. Since about twenty-five percent of the cpu time was devoted to the data movement, use of the hardware GATHER/SCATTER equipment would essentially eliminate all of the data movement time. Modifying the code to support all four processors would also decrease the cpu time by a factor of about four, since no extra data communication is needed to utilize all the processors.

Table 2

Iteration, CPU, and MFLOPS Statistics for

$a(x, y) = a_1(x, y)$  on the CRAY-XMP

$N_x \times N_y$	$\bar{N}_x \times \bar{N}_y$	Number of Iterations	CPU (seconds)	MFLOPS
$129 \times 129$	$8 \times 8$	39	0.70	90.4
$257 \times 257$	$16 \times 16$	52	3.48	94.4

Table 3 summarizes convergence results  $a(x, y) = a_2(x, y)$  and for  $N_x = N_y = 33, 65,$  and  $129$ . Again we see the same behavior in the iteration count versus the cpu time as in the previous problems. Based on this information, the optimal selection is  $\bar{N}_x = \bar{N}_y = 8$  for  $N_x = N_y = 65$

and  $\bar{N}_x = \bar{N}_y = 8$  for  $N_x = N_y = 129$ . Even though the coefficient in the differential equation  $a_2(x, y)$  varies drastically from one side of the domain to the other side, the number of iterations needed for convergence is only about thirty percent higher than needed for  $a_1(x, y)$ .

Table 3

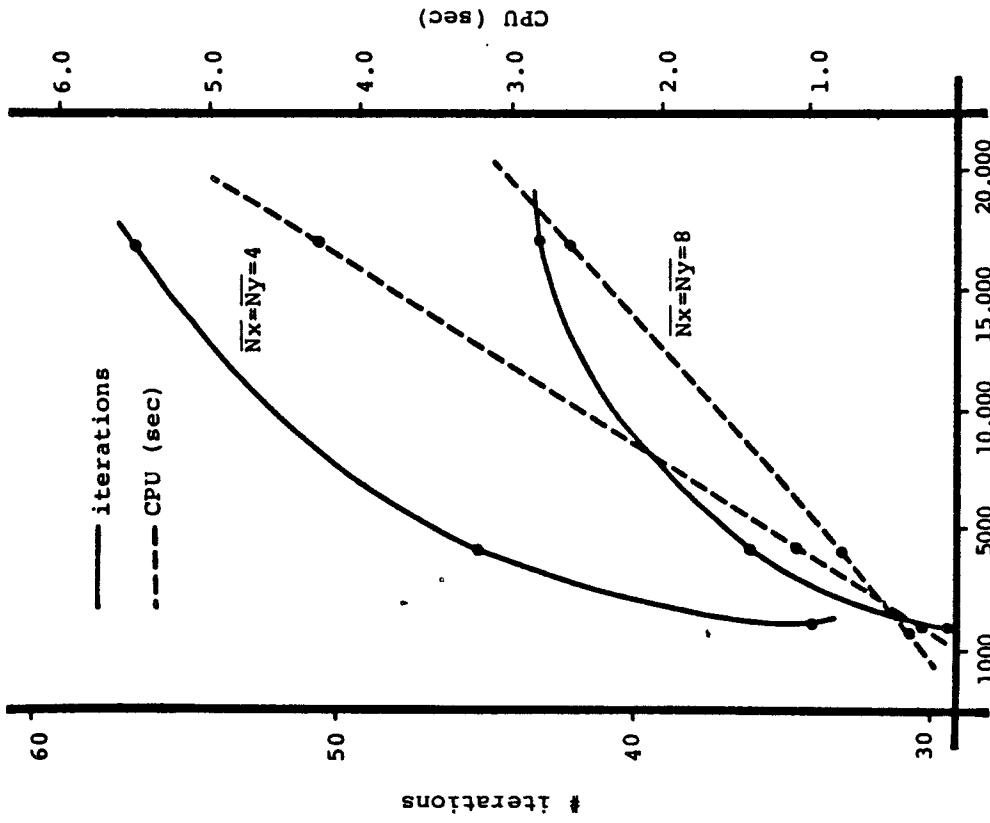
Iteration and CPU Statistics for  $a(x, y) = a_2(x, y)$

$N_x \times N_y$	$\bar{N}_x \times \bar{N}_y$	Number of Iterations	CPU (seconds)
33 × 33	2 × 2	32	0.48
	4 × 4	38	.34
	8 × 8	37	0.44
65 × 65	2 × 2	44	2.6
	4 × 4	41	1.1
	8 × 8	44	1.0
	16 × 16	46	1.9
129 × 129	4 × 4	53	4.5
	8 × 8	53	3.3
	16 × 16	49	3.8

Table 4 illustrates the convergence properties of test problem 3 for  $N_x = N_y = 33, 65,$  and  $129$ . The information for  $N_x = N_y = 33$  is plotted in Figure 3. From this plot it is clear that the number of iterations increases as the total number of subregions ( $\bar{N}_x \times \bar{N}_y$ ) goes to unity, but that the number of iterations decreases rapidly for more than twelve subdomains

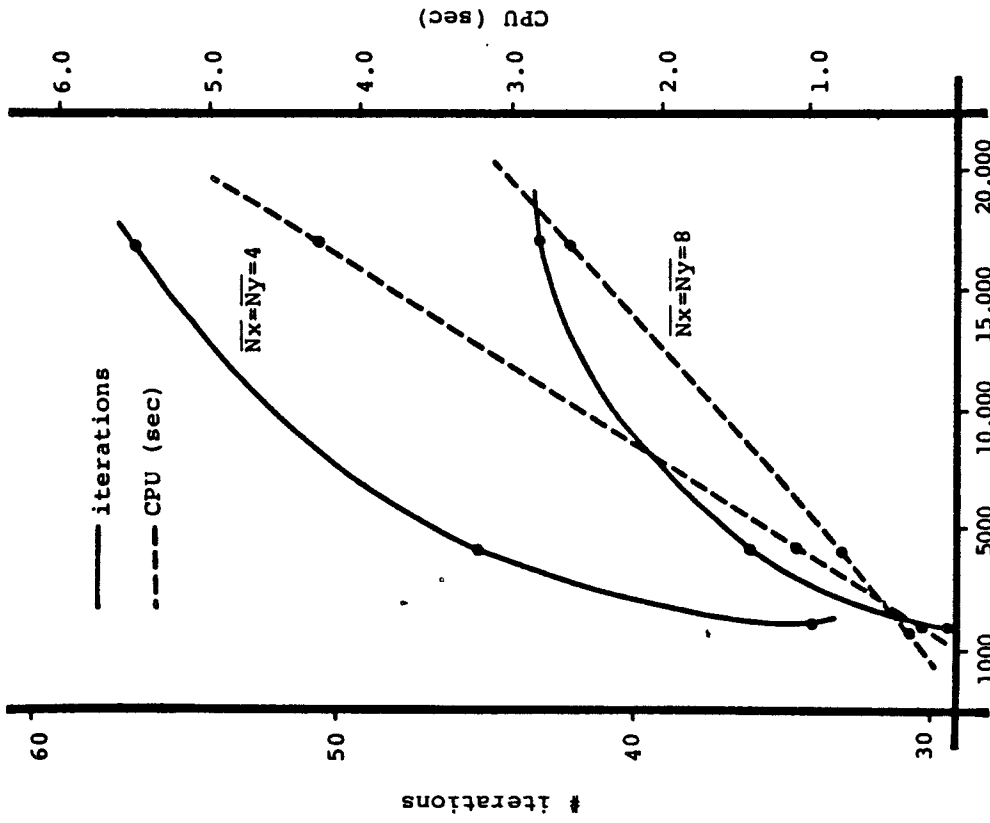
In Figure 4 the plots illustrate the convergence results for  $\bar{N}_x = \bar{N}_y = 4$  and  $\bar{N}_x = \bar{N}_y = 8$ . The curve for cpu time increases almost linearly with respect to the increase in mesh, and the curve for number of iterations appears to increase logarithmically with respect to the increase in mesh. In fact, this is the same logarithmic behavior that Bramble, et al. derived, even though we only use the first step of the MINV block preconditioner for each subdomain for the computation of  $U_p$  and  $U_H$ .

From Table 4 we determine the "best cases" for each of the meshes (here the best case means choosing  $\bar{N}_x \simeq \sqrt{N_x}$  and  $\bar{N}_y \simeq \sqrt{N_y}$ ). In Figure 5 we see this same information as in Table 4. It is interesting to observe that both the number of iterations and the cpu time increase



$$\bar{N}_x \times \bar{N}_y$$

Figure 3 Iterations and CPU versus Number of Subregions for  $a_3(x,y)$  with  $N_x = N_y = 33$



$$N = \bar{N}_x \times \bar{N}_y$$

Figure 4 Iterations and CPU versus Number of Subregions for  $a_3(x,y)$  with  $\bar{N}_x = \bar{N}_y = 4$  and  $\bar{N}_x = \bar{N}_y = 8$

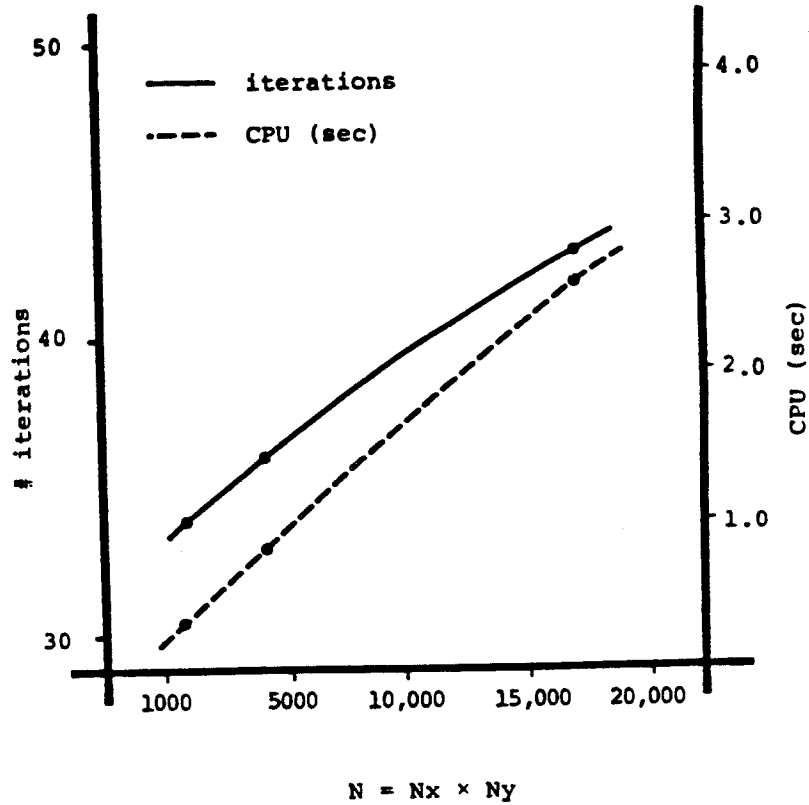


Figure 5 Iterations and CPU versus Number of Unknowns for the Best Cases for  $a_3(x,y)$

almost linearly with respect to the total number of unknowns and not quadratically. From these calculations we conclude, especially for large problems, that this method is a practical method for solving the two-dimensional elliptic equation.

Table 4

Iteration and CPU Statistics for  $a(x, y) = a_3(x, y)$ .

$N_x \times N_y$	$\bar{N}_x \times \bar{N}_y$	Number of Iterations	CPU (seconds)
33 × 33	1 × 1	71	3.70
	2 × 2	44	0.72
	4 × 4	34	0.27
	8 × 8	29	0.35
65 × 65	2 × 2	62	3.5
	4 × 4	45	1.1
	8 × 8	36	0.8
	16 × 16	35	1.5
129 × 129	4 × 4	55	4.7
	8 × 8	43	2.6
	16 × 16	46	3.6
	32 × 32	44	6.8

#### REFERENCES

1. J.H. Bramble, J.E. Pasciak, and A.H. Schatz, "The Construction of Preconditioners for Elliptic Problems by Substructuring, I," to appear in *Math. Comp.*
2. \_\_\_\_\_, "An Iterative Method for Elliptic Problems on Regions Partitioned into Substructures," to appear in *Math. Comp.*
3. R. Chandra, "Conjugate Gradient Methods for Partial Differential Equations," Technical Report YALEU/DCS/RR-129, Computer Science Dept., Yale University, January 1978.
4. P. Concus, G. Golub, and G. Meurant, "Block Preconditioning for the Conjugate Gradient Method," *SIAM Journal on Scientific and Statistical Computing*, 6, 1985, pp. 220-252.
5. R. Gonzalez, "Domain Decomposition for Two-Dimensional Elliptic Operators on Vector and Parallel Machines," Ph.D. thesis, Rice University, 1986.