CHAPTER 27

The Parallel Block Preconditioned Conjugate Gradient Algorithm†

Wang Jin-xian*

Abstract In this paper we discuss the parallel algorithm of block preconditioned conjugate gradient for obtaining a numerical solution of elliptic partial differential equation boundary value problems on a MIMD computer. In section 2 we give two algorithms: BLICG(1) and its modification BLICG(n). In section 3 we discuss their convergence and point out that algorithm 2 has almost the same convergence rate on any number of processors as on one processor even for an anisotropic problem. In section 4 we give a numerical example

1. Introduction

Preconditioning by incomplete decomposition is one of the standard techniques in iterative methods for the solution of large sparse linear system. In combination with conjugate gradient, the so-called ICCG algorithm is one of the efficient methods for solving symmetric positive definite linear systems arising in the numerical solution of elliptic partial differential equation. Recent years ICCG for multiprocessor computers with local memory has been of great interest. We partition the domain into a number of strip subdomains and call the interface between two subdomains an inner boundary. Each processor is related to one subdomain and inner boundary. However, the main problem is that the convergence rate will decrease when the number of processors increases for the same number of mesh points, especially for anisotropic problems. In this paper we are concerned with finding a block preconditioner for a parallel CG algorithm. In section 2 we give an algorithm BLICG(1) and its modification whose basic idea is to increase the completeness of the decomposition on the inner boundary. In section 3 we discuss some problems about the convergence and convergence rate of the algorithm and come to the conclusion that algorithm BLICG(n) is of the same convergence rate on $N(N > 1)$ processors and on one processor. In section 4 we give a numerical example.

2. The parallel block ICCG algorithm.

We consider elliptic partial differential equations with Dirichlet or Neumann boundary con-

* Computing center, Academia Sinica, Beijing 100080, P.R.China.
† This work supported by the National Natural Science Foundation of China.

dition.

$$\begin{cases} -\frac{\partial}{\partial x}a(x,y)\frac{\partial u}{\partial x} - \frac{\partial}{\partial y}b(x,y)\frac{\partial u}{\partial y} + c(x,y)u = f(x,y) & (x,y) \in \Omega = (0,1)\times(0,1) \\ \alpha(x,y)\frac{\partial u}{\partial n} + \beta(x,y) = g(x,y) & (x,y) \in \partial\Omega \end{cases} \tag{1}$$

$$C(x,y) \geq 0, a(x,y) > 0, b(x,y) > 0$$

The equation is discretized in the usual way with a five-point difference scheme. We obtain the following system of linear equations

$$Au = f \tag{2}$$

Here A is a positive definite M-matrix, u is the discrete solution vector and f the discrete right hand side. We consider 'splitting' $A = K - R$ with a symmetric positive definite preconditioner K and a remainder matrix R. A preconditioned Conjugate Gradient(PCG) algorithm with preconditioner K is as follows:

Let $u^{(0)}$ be an arbitrary initial approximation to u and $r^{(0)} = f - Au^{(0)}$, the initial residual, and for $k = 0, 1, \cdots$ do

$$\begin{cases} Kz^{(k)} = r^k \\ \beta_k = (z^{(k)}, Kz^{(k)})/(z^{(k-1)}, Kz^{(k-1)}), \ \beta_0 = 0 \\ p^{(k)} = z^{(k)} + \beta_k p^{(k-1)}, \qquad p^{(-1)} = 0 \\ \alpha_k = (z^{(k)}, Kz^{(k)})/(p^{(k)}, Ap^{(k)}) \\ u^{(k+1)} = u^{(k)} + \alpha_k p^{(k)} \\ r^{(k+1)} = r^{(k)} - \alpha_k Ap^{(k)} \end{cases} \tag{3}$$

until convergence. In the algorithm (3) the crucial point is the solution of the equation

$$Kz^{(k)} = r^{(k)} \tag{4}$$

There are many kinds of preconditioners[1-3,5-8] but the incomplete block decomposition of A is one of the most effective. We partition the domain Ω into N strips $\Omega_1, \Omega_2, ..., \Omega_N$ and N-1 interior interface $\partial\Omega_{1,2}, \partial\Omega_{2,3}, ..., \partial\Omega_{N-1,N}$; each Ω_i has n lines. Every subdomain Ω_i and $\partial\Omega_{i-1,i}$ are treated by one processor and some exchange of data among processors is needed. After renumbering of the points, the coefficient matrix A has following form:

$$A = \begin{bmatrix} A_1 & & & & & C_{1,0}^T & & & & \\ & A_2 & & & & C_{1,1}^T & C_{2,0}^T & & & \\ & & \ddots & & & & C_{2,1}^T & \ddots & & \\ & & & \ddots & & & & \ddots & & C_{N-1,0}^T \\ & & & & A_N & & & & & C_{N-1,1}^T \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ C_{1,0} & C_{1,1} & & & & B_1 & & & & \\ & C_{2,0} & C_{2,1} & & & & B_2 & & & \\ & & \ddots & \vdots & & & & \ddots & & \\ & & \ddots & \ddots & & & & & \ddots & \\ & & & C_{N-1,0} & C_{N-1,1} & & & & & B_{N-1} \end{bmatrix}$$

Where $A_i (i = 1, ..., N)$ are block tridiagonal matrices.

$$A_i = \begin{bmatrix} D_{i,1} & B_{i,1}^T & & \\ B_{i,1} & D_{i,2} & B_{i,2}^T & \\ & \ddots & \ddots & \ddots \\ & & B_{i,n-1} & D_{i,n} \end{bmatrix}$$

$D_{i,j}(j = 1, ..., n)$ tridiagonal matrices, $C_{i,0} = (0, 0, ..., c_{i,0})$, $C_{i,1} = (c_{i,1}, 0, ..., 0)$, $B_{i,j}$ $(j = 1, ..., n-1)$, $c_{i,0}, c_{i,1}$ are diagonal matrices. For direct solution algorithms based on the same decomposition see [2a].

Algorithm 1 - BLICG(1)

We take a preconditioner K in the form

$$K = L \wedge^{-1} L^T \tag{5}$$

Where

$$L = \begin{bmatrix} L_1 & & & & & \vdots & \\ & L_2 & & & & \vdots & \\ & & \ddots & & & \vdots & \\ & & & \ddots & & \vdots & \\ & & & & L_N & \vdots & \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots\cdots\cdots\cdots\cdots & \cdots \\ C_{1,0}\ C_{1,1} & & & & & \vdots\ \Delta_1 & \\ & C_{2,0}\ C_{2,1} & & & & \vdots\quad \Delta_2 & \\ & & \ddots & \ddots & & \vdots\quad\quad \ddots & \\ & & & \ddots & \ddots & \vdots\quad\quad\quad \ddots & \\ & & & C_{N-1,0}\ C_{N-1,1} & & \vdots\quad\quad\quad\quad \Delta_{N-1} & \end{bmatrix} \quad L_i = \begin{bmatrix} \wedge_{i,1} & & & \\ B_{i,1} & \wedge_{i,2} & & \\ & \ddots & \ddots & \\ & & B_{i,n-1} & \wedge_{i,n} \end{bmatrix}$$

$$\wedge = diag\{\wedge_1, \wedge_2, \cdots, \wedge_N, \Delta_1, \cdots, \Delta_{N-1}\}$$
$$\wedge_i = diag\{\wedge_{i,1}, \wedge_{i,2}, \cdots, \wedge_{i,n}\}$$

$$\begin{cases} \wedge_{i,1} = D_{i,1} \\ \wedge_{i,2} = D_{i,2} - B_{i,1} \wedge_{i,1}^{-1} B_{i,1}^T \\ \cdots \\ \wedge_{i,n} = D_{i,n} - B_{i,n-1} \wedge_{i,n-1}^{-1} B_{i,n-1}^T \end{cases} \tag{6}$$

$$\Delta_i = B_i - C_{i,0} \wedge_{i,n}^{-1} C_{i,0}^T - C_{i,1} \wedge_{i+1,1}^{-1} C_{i,1}^T$$

In (6) $\wedge_{i,j}^{-1}$ $(j = 1, \cdots, n-1)$ are usually full matrices. In order to maintain the original sparsity, we use the banded approximate inverse $\Pi_{i,j} = [\wedge_{i,j}^{-1}]^{(p)}$ instead of $\wedge_{i,j}^{-1}$. Here $[C]^{(p)}$ denotes the matrix with entries

$$[C]_{i,j}^{(p)} = \begin{cases} c_{i,j} & | i - j | \leq p \\ 0 & otherwise \end{cases} \tag{7}$$

$p \geq 0$ is an integer. We obtain

$$\begin{cases} \overline{\wedge}_{i,1} = D_{i,1} \\ \overline{\wedge}_{i,2} = D_{i,2} - B_{i,1} \Pi_{i,1} B_{i,1}^T \\ \cdots \\ \overline{\wedge}_{i,n} = D_{i,n} - B_{i,n-1} \Pi_{i,n-1} B_{i,n-1}^T \\ \overline{\Delta}_i = B_i - C_{i,0} \Pi_{i,n} C_{i,0}^T - C_{i,1} \Pi_{i+1,1} C_{i,1}^T \end{cases} \tag{8}$$

Where

$$\Pi_{i,j} = [\overline{\wedge}_{i,j}^{-1}]^{(p)} \quad (j = 1, \cdots, n-1)$$

We denote the matrix corresponding to K by $\overline{K}$:

$$\overline{K} = \overline{L} \overline{\wedge}^{-1} \overline{L}^T \tag{9}$$

The remainder matrix $\overline{R} = \overline{K} - A$ is

$$\overline{R} = \begin{bmatrix} \overline{R}_1 & & & & \vdots & 0 & & \\ & \overline{R}_2 & & & \vdots & \overline{R}_{1,1}^T & \ddots & \\ & & \ddots & & \vdots & & \ddots & \ddots \\ & & & \overline{R}_N & \vdots & & & \overline{R}_{N-1,1}^T \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \overline{R}_{1,1} & & & \vdots & \overline{R}_{1,B} & & \\ & \ddots & \ddots & & \vdots & & \ddots & \\ & & 0 & \overline{R}_{N-1,1} & \vdots & & & \overline{R}_{N-1,B} \end{bmatrix} \tag{10}$$

$$\begin{aligned} \overline{R}_i &= diag(0, B_{i,1}(\overline{\wedge}_{i,1}^{-1} - \Pi_{i,1})B_{i,1}^T, \cdots, B_{i,n-1}(\overline{\wedge}_{i,n-1}^{-1} - \Pi_{i,n-1})B_{i,n-1}^T) \\ \overline{R}_{i,1} &= (0, c_{i,1}\overline{\wedge}_{i+1,1}^{-1}B_{i+1,1}^T, 0, \cdots, 0) \\ \overline{R}_{i,B} &= c_{i,0}(\overline{\wedge}_{i,n}^{-1} - \Pi_{i,n})c_{i,0}^T + c_{i,1}(\overline{\wedge}_{i+1,1}^{-1} - \Pi_{i+1,1})c_{i,1}^T \end{aligned}$$

The main problem with the algorithm 1 is the convergence rate will decrease rapidly when we increase the number of processors for the same mesh, especially for anisotropic problems of the type discussed in next section. This is because of the incompleteness of the decomposition of K in two respects: 1) The exact inverse of the tridiagonal matrix has been approximated by a banded matrix. 2) There exists remainder matrix $R = A - K$ corresponding to the interior interface. When the number of subdomains increases, the fraction of boundary points also increases. In order to improve the convergence rate, we take K such that it is a complete or almost complete decomposition of A on the interior interface. Thus, we consider the following algorithm.

Algorithm 2 - BLICG(n)

We consider the following preconditioner $\widetilde{K}$

$$\widetilde{K} = \widetilde{L}\widetilde{\wedge}^{-1}\widetilde{L}^T \tag{11}$$

where

$$\widetilde{L} = \begin{bmatrix} \widetilde{L}_1 & & & & \vdots & & & & & \\ & \widetilde{L}_2 & & & \vdots & & & & & \\ & & \ddots & & \vdots & & & & & \\ & & & \ddots & \vdots & & & & & \\ & & & & \widetilde{L}_N & \vdots & & & & \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ c_{1,0} & \widetilde{c}_{1,1} & & & & \vdots & \widetilde{\Delta}_{1,1} & & & \\ & c_{2,0} & \widetilde{c}_{2,1} & & & \vdots & \widetilde{\Delta}_{2,1} & \widetilde{\Delta}_{2,2} & & \\ & & \ddots & \ddots & & \vdots & & \ddots & \ddots & \\ & & & c_{N-1,0} & \widetilde{c}_{N-1,1} & \vdots & & & \widetilde{\Delta}_{N-1,N-2} & \widetilde{\Delta}_{N-1,N-1} \end{bmatrix}, \widetilde{L}_i = \begin{bmatrix} \overline{\wedge}_{i,1} & & & \\ B_{i,1} & \overline{\wedge}_{i,2} & & \\ & \ddots & \ddots & \\ & & B_{i,n-1} & \overline{\wedge}_{i,n} \end{bmatrix}$$

$$\widetilde{\wedge} = diag\left\{\widetilde{\wedge}_1, \widetilde{\wedge}_2, \cdots, \widetilde{\wedge}_N, \widetilde{\Delta}_{1,1}, \cdots, \widetilde{\Delta}_{N-1,N-1}\right\}$$

$$\widetilde{\wedge}_i = diag\left\{\overline{\wedge}_{i,1}, \overline{\wedge}_{i,2}, \cdots, \overline{\wedge}_{i,n}\right\}$$

$c_{i,0}, \overline{\wedge}_{i,j}$ are the same as in algorithm 1.

$$\widetilde{c}_{i,1} = (\widetilde{c}_{i,1}, \widetilde{c}_{i,2}, \cdots, \widetilde{c}_{i,n})^T \quad (i = 1, \cdots, N-1)$$

$$\begin{cases} \widetilde{c}_{i,1} = c_{i,1} \\ \widetilde{c}_{i,j} = -\widetilde{c}_{i,j-1}\Pi_{i+1,j-1}B_{i+1,j-1}^T \quad (j = 2, \cdots, n) \\ \widetilde{\Delta}_{i,i} = B_i - C_{i,0}\Pi_{i,n}C_{i,0}^T - \sum_{j=1}^{n} \widetilde{C}_{i,j}\Pi_{i+1,j}\widetilde{C}_{i,j}^T - \widetilde{\Delta}_{i,i-1}[\widetilde{\Delta}_{i-1,i-1}]^{(p)}\widetilde{\Delta}_{i,i-1}^T, \quad \widetilde{\Delta}_{1,0} = 0 \\ \widetilde{\Delta}_{i,i-1} = -C_{i,0}\Pi_{i,n}\widetilde{C}_{i-1,n}^T \end{cases} \tag{12}$$

The remainder matrix, $\widetilde{R} = \widetilde{K} - A$, is

$$\widetilde{R} = \begin{bmatrix} \widetilde{R}_1 & & & & & \vdots & 0 & & & & \\ & \widetilde{R}_2 & & & & \vdots & \widetilde{R}_{1,1}^T & 0 & & & \\ & & \ddots & & & \vdots & & \ddots & \ddots & & \\ & & & \ddots & & \vdots & & & \ddots & 0 & \\ & & & & \widetilde{R}_N & \vdots & & & & \widetilde{R}_{N-1,1} & \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \\ 0 & \widetilde{R}_{1,1} & & & & \vdots & \widetilde{R}_{1,1} & \widetilde{r}_2^T & & & \\ & 0 & \widetilde{R}_{2,1} & & & \vdots & \widetilde{r}_2 & \widetilde{R}_{2,2} & \widetilde{r}_3^T & & \\ & & \ddots & \ddots & & \vdots & & \ddots & \ddots & \ddots & \\ & & & 0 & \widetilde{R}_{N-1,1} & \vdots & & & \widetilde{r}_{N-1} & \widetilde{R}_{N-1,N-1} & \end{bmatrix}$$

where

$$\begin{cases} \widetilde{R}_i = \overline{R}_i \\ \widetilde{R}_{i,1} = (0, \widetilde{C}_{i,1}(\overline{\wedge}_{i+1,1}^{-1} - \Pi_{i+1,1})B_{i+1,1}^T, \cdots, \widetilde{C}_{i,n-1}(\overline{\wedge}_{i+1,n-1}^{-1} - \Pi_{i+1,n-1})B_{i+1,n-1}^T) \\ \widetilde{R}_{i,i} = C_{i,0}(\overline{\wedge}_{i,n}^{-1} - \Pi_{i,n})C_{i,0}^T + \sum \widetilde{C}_{i,j}(\overline{\wedge}_{i+1,j}^{-1} - \Pi_{i+1,j})\widetilde{C}_{i,j}^T + \widetilde{\Delta}_{i,i-1}(\widetilde{\Delta}_{i,i}^{-1} - [\widetilde{\Delta}_{i,i}^{-1}]^{(p)})\widetilde{\Delta}_{i,i-1}^T, \\ \qquad \widetilde{\Delta}_{1,0} = 0 \\ \widetilde{r}_i = C_{i,0}(\overline{\wedge}_{i,n}^{-1} - \Pi_{i,n})\widetilde{C}_{i-1,n}^T, \quad (i = 2, \cdots, N-1) \end{cases} \tag{13}$$

When $j = 1$, algorithm 2 coincides with algorithm 1.

3. Some problems on convergence and convergence rate for algorithm BLICG(n)

In this section we discuss some problems about the convergence and convergence rate of the algorithm mentioned in the previous section. Let $S_N = \{(i,j) \mid_{1 \le i \le j \le N}\}$

Theorem 1. Let $A = (A_{i,j})$ be a block $N \times N$ positive definite M-matrix, $A_{i,i}$ is diagonally dominant, $A_{i,j} \le 0 (i \ne j)$, then there exists for every $S \subset S_N$ a lower block triangular matrix $L = (L_{i,j})(L_{i,i} = I)$ and block diagonal matrix $D = diag\{D_1, \cdots, D_N\}$ and $R = (R_{i,j})$ with

$$\begin{aligned} L_{i,j} &= 0 \ (i,j) \in S \\ R_{i,j} &= 0 \ (i,j) \in S \end{aligned}$$

and D_i are diagonally dominant M-matrixes, such that $A = (D+L)D^{-1}(D+L)^T - R$ is a regular splitting.

Proof. We can take

$$\begin{aligned} D_i &= A_{i,i} - \sum_{\substack{j=1 \\ (i,j) \in S}}^{i-1} L_{i,j} D_j^{-1} L_{i,j}^T \quad (i = 1, \cdots, N) \\ L_{i,j} &= A_{i,j} - \sum_{\substack{k=1 \\ (i,k) \in S, \\ (j,k) \in S}}^{j-1} L_{i,k} D_k^{-1} L_{j,k}^T \ (i,j) \in S, \ \ i > j \\ R_{i,j} &= \sum_{\substack{k=1 \\ (i,k) \in S, \\ (j,k) \in S}}^{j-1} L_{i,k} D_k^{-1} L_{j,k}^T \qquad (i,j) \in S \end{aligned} \tag{14}$$

In (14) the D_i are diagonally dominant M-matrices[3], so the D_i are positive definite M-matrices, hence $D_i^{-1} \ge 0, L_{i,j} \le 0, R_{i,j} \ge 0$, and satisfy the relation

$$A = (D+L)D^{-1}(D+L)^T - R \equiv K - R$$

Since $D^{-1} \geq 0, L \leq 0, R \geq 0$, it can be shown easily that $(D+L)^{-1}$ exists and $(D+L)^{-1} \geq 0, D(D+L)^{-1} \geq 0$, hence $K^{-1} = (D+L)^{-T} D (D+L)^{-1} \geq 0$, and $A = K - R$ is a regular splitting.

Corollary. A natural selection of the index set S is $\overline{S} = \{(i,j) \mid A_{i,j} \neq 0\}$. In (5),(6) the incomplete decomposition K is undertaken with $\overline{S}$, hence (5), (6), is a regular splitting. As for the decomposition (11), (12) it would be complete if all inverse matrices were exact i.e, if $\Lambda_{i,j}^{-1}$ were used instead of $\pi_{i,j}$ in(12). Because all $\wedge_{i,j}$ of (6) are diagonally dominant M-matrices, $\overline{\wedge}_{i,j}$ of (8) are diagonally dominant also, so $\overline{K}$ of (9) is positive definite and algorithm 1 converges. Similarly $\widetilde{K}$ in algorithm 2 is positive definite also and algorithm 2 converges.

Application to anisotropic problems. Consider the following anisotropic problem

$$\begin{cases} -(\varepsilon u_{xx} + u_{yy}) = f, \ \varepsilon \ll 1 & (x,y) \in \Omega = (0,1)^2 \\ u = g & (x,y) \in \partial\Omega \end{cases} \tag{15}$$

with the five-point difference scheme, the main submatrix of A is

$$d_i = \begin{bmatrix} 2(1+\varepsilon) & -\varepsilon & & & \\ -\varepsilon & 2(1+\varepsilon) & -\varepsilon & & \\ & \ddots & \ddots & \ddots & \\ & & -\varepsilon & 2(1+\varepsilon) & \end{bmatrix}_{n \times n} \tag{16}$$

The entries of $D_i^{-1} = (\alpha_{i,j})$ decay along each row or column[4], i.e. there exists constant C_1 such that $\alpha_{i,j} \leq C_1 r^{|i-j|}$, where

$$\begin{aligned} & r = (\sqrt{k}-1)/(\sqrt{k}+1) \ \ and \ \ k = \lambda_{max}(D_i)/\lambda_{min}(D_i) \\ & \lambda_{max}(D_i) \simeq 2 + 4\varepsilon, \ \ \lambda_{min}(D_i) \simeq 2 + \varepsilon(\pi h)^2, \ \ h = \tfrac{1}{n+1} \\ & k = 1 + c_1\varepsilon, \ \ r = c_2\varepsilon \\ & \alpha_{i,j} \leq C_1 (C_2\varepsilon)^{|i-j|} = C\varepsilon^{|i-j|}, \ \ C_1, C_2, C \ are \ constants \\ & D_i^{-1} - [D_i^{-1}]^{(p)} = \begin{cases} 0 & |\, i-j \,| \leq p \\ \alpha_{i,j} = 0(\varepsilon^{p+1}) & otherwise \end{cases} \end{aligned} \tag{17}$$

$$i.e. \quad In(10), \ \ \| \overline{\wedge}_{i,1}^{-1} - \Pi_{i,1} \| = \max_{1 \leq i \leq n} \sum_{j=1}^{n} | \alpha_{i,j} | = 0(\varepsilon^{p+1})$$

By induction from (8) we can prove that $\| \overline{\wedge}_{i,j}^{-1} - \Pi_{i,j} \| = 0(\varepsilon^{p+1}), j = 2, \cdots, n$. So in (13), all norms of submatrices $\widetilde{R}_i, \widetilde{R}_{i,1}, \widetilde{R}_{i,i}, \widetilde{r}_i$ are of $0(\varepsilon^{p+1})$

$$\| \widetilde{R} \| = \| K - A \| = 0(\varepsilon^{p+1})$$

We have [9,theorem 3.13]

$$\begin{aligned} & \rho(\widetilde{K}^{-1}\widetilde{R}) \leq \tfrac{\|\widetilde{R}\| \cdot \|A^{-1}\|}{1 + \|\widetilde{R}\| \cdot \|A^{-1}\|} = 0(\varepsilon^{p+1}) \\ & \lambda(\widetilde{K}^{-1}A) = 1 - \lambda(\widetilde{K}^{-1}\widetilde{R}) = 1 + 0(\varepsilon^{p+1}) \end{aligned} \tag{18}$$

Theorem 2. In algorithm 2 for preconditioner $\widetilde{K}$, the eigenvalues of $\widetilde{K}^{-1}A$ are of form $1 + 0(\varepsilon^{p+1})$.

For algorithm 1, from (10) the norms of $\overline{R}_{i,1}$ only are of $0(1)$, hence $\lambda(\overline{K}^{-1}A) = 1 + 0(1)$. Therefore for anisotropic problem(16) when $\varepsilon \ll 1$ algorithm 2 is much more efficient than algorithm 1.

When $N =$ there is no interior interface, from (10) the remaining matrix $R = diag\{0, B_{i,1}(\overline{\wedge}_{i,1}^{-1} - \Pi_{i,1})B_{i,1}^T, 0, \cdots, 0\}$. Hence $\| \overline{R} \| = 0(\varepsilon^{p+1}), \lambda(K^{-1}A) = 1 + 0(\varepsilon^{p+1})$. So algorithm 2 using N processors has the same convergence rate as on one processor.

4. Numerical results.

We use algorithm 1 and 2 described in section 2 to solve problem(15).

We take the 5 point difference scheme with $h = 1/64, r = 10^{-10}$. In table 1, N is the number of processors, m the number of iterations needed for $\mid r_m \mid \leq r, r_m = \max_i (f - AU^{(m)})_i$ and in (12) we take $\Pi_{i,j}$ diagonal matrices, i.e. $p = 0$. From the table we can find that the smaller the ε and the bigger the N are, the more efficient the algorithm 2 is. In that case algorithm 2 has almost the same number of iterations as the serial algorithm on one processor.

Algorithm	N	$\varepsilon = 10^{-4}$ m	10^{-3} m	10^{-2} m	10^{-1} m	10^{0} m
1	1	3	6	13	25	34
	2	15	28	47	46	45
	4	30	46	60	57	50
	8	53	77	84	75	55
	16	85	120	133	104	69
2	1	—	—	—	—	—
	2	5	9	18	28	39
	4	5	9	20	32	39
	8	5	8	19	37	45
	16	4	6	15	33	51

TABLE 1

REFERENCES

[1] O. Axelsson, Incomplete block matrix factorization preconditioning methods. The ultimate answer? J.Comput. and Appl. Math., 12,13(1985),3-18.

[2] J.H. Bramble, J.E. Pasciak, A.H. Schatz, The construction of preconditioners for elliptic problems by substructuring I, Math. Comput., 47(1986), 103-134.

[2a] T.F.Chan and D.Resasco, A domain-decomposed fast Poisson solver on a rectangle, SIAM J.Sci. Stat.Comp.8(1987),s14-s26.

[3] P.Concus, G.H. Golub, G.Meurant, Block preconditioning for the conjugate gradient method, SIAM J. Sci. Stat. Comput., 6(1985), 220-252.

[4] S. Demko, W.F. Moss, P.W. Smith, Decay rate for inverses of band matrices, Math. Comput., 43 (1984), 491-499.

[5] G.H. Golub, D. Mayers, The use of preconditioning over irregular regions, Proc. Sixth Internat. Conf. on Sci. and Engineering, 1984, 3-14.

[6] J.A. Meijerink, H. A. Van der Vorst, An iterative solution method for linear system of which the coefficient matrix is a symmetric M-matrix, Math. Comput., 31(1977), 148-162

[7] G. Meurant, Multitasking the conjugate gradient method on the CRAY X-MP/48, Parallel Computing, 5(1987), 267-280.

[8] G. Meurant, Domain decomposition versus block preconditioning, Proc. of the internat. Symp. on Domain Decomposition Method for PDE (1988).

[9] R.S. Varga, Matrix Iterative Analysis, Prentic Hall, Englewood Cliffs, NJ, 1962.

[10] Wang Jin-xian, The parallel ICCG algorithm for elliptic PDE with boundary value problems on MIMD computer (in Chinese), Proc. of National Parallel Comput. Conf., Hebei, China, 1989,1-6.