IMPACT OF COMPUTING IN SCIENCE AND ENGINEERING 3, 235–243 (1991)

A Parallel Version of Some Block Preconditionings*

P. AMODIO AND L. BRUGNANO

Dipartimento di Matematica, Università di Bari, Campus, 70125 Bari, Italy

Received January 4, 1990

P. Amodio and L. Brugnano, A. Parallel Version of Some Block Preconditionings, *IMPACT of Computing in Science and Engineering* 3, 235–243 (1991).

The block preconditioned conjugate gradient (BPCG) methods, even if very effective for solving the linear systems deriving from the discretization of partial differential equations, are not efficient for parallel computers. The bottleneck for their parallel implementation is represented by the solution of the linear system to obtain the preconditioned residual. Here, we present a parallel version of some BPCG algorithms. They are based on an approximate solver for tridiagonal systems, which utilizes an incomplete 2×2 block odd–even reduction. The stability properties of the approximate solver are investigated. Some numerical tests, on a net of transputers, are also included.

1. INTRODUCTION

Block incomplete Cholesky factorizations are used to obtain very effective preconditionings for the conjugate gradient method [4]. In particular, we consider the INV(1) preconditioning previously published [4, 5], even if the same approach can be extended to other preconditionings in this class.

It is well known that the obstacle to a significant parallel speed-up for a BPCG algorithm using block preconditioning like INV is the solution of the linear system to calculate the preconditioned residual. The BPCG algorithm is intended to be used for the solution of the linear system

$$A\mathbf{x} = \mathbf{b}, \tag{1.1}$$

* Work supported by the C.N.R. (Contract 89.1791.01) and by the European Community (ESPRIT project, P.C.A. Contract 4040).

0899-8248/91 $3.00

where A is a symmetric M-matrix with the structure

$$A = \begin{bmatrix} B_1 & F_2^T & & \\ F_2 & \ddots & \ddots & \\ & \ddots & \ddots & F_k^T \\ & & F_k & B_k \end{bmatrix}, \tag{1.2}$$

and the blocks B_i are tridiagonal $m \times m$.

If $(D_i \cdots D_k)$ are the $m \times m$ tridiagonal blocks of the approximate block factorization, then the solution of the linear system to calculate the preconditioned residual follows from the solution of $2k - 1$ linear sub-systems, which have the blocks D_i as coefficient matrices. For the INV preconditioning, these blocks [4] are tridiagonal, symmetric, strictly diagonally dominant M-matrices [8].

As the preconditioner itself is an approximation of the inverse of the matrix (1.2), instead of exactly solving such tridiagonal subsystems [7], one can try to make use of an "approximate" solution reached with a more parallel method (this is equivalent to using a slightly different preconditioner). The chosen approximation is obtained by means of an incomplete block cyclic reduction.

2. Approximating the Solution of the Tridiagonal Subsystems

The basic idea is to use an incomplete 2×2 block odd–even reduction to approximately solve the tridiagonal linear subsystems needed to compute the preconditioned residual. If P_0 is the block permutation matrix $m \times m$ which first takes the odd couples of lines, and then the even ones, we have (let T_0 be the original tridiagonal coefficient matrix)

$$P_0 T_0 P_0^T = \begin{bmatrix} B_0 & S_0^T \\ S_0 & C_0 \end{bmatrix}, \tag{2.1}$$

where B_0 and C_0 are block diagonal, with 2×2 diagonal blocks, while S_0 is upper diagonal. The LU block factorization of (2.1) is given by

$$\begin{bmatrix} B_0 & 0 \\ S_0 & T_1 \end{bmatrix} \cdot \begin{bmatrix} I & B_0^{-1} S_0^T \\ 0 & I \end{bmatrix},$$

where

$$T_1 = C_0 - S_0 B_0^{-1} S_0^T. \tag{2.2}$$

It is evident that B_0^{-1} is a block diagonal matrix, and T_1 is again tridiagonal of dimension $\lfloor m/2 \rfloor \times \lfloor m/2 \rfloor$. Therefore we can repeat iteratively the above operations. Moreover, at every step of the reduction, the elements that make T_i different from a 2×2 block diagonal matrix tend to 0 exponentially as the process of reduction continues. It follows that after a few steps of reduction one can approximate the tridiagonal block (the Schur complement (2.2) of the last reduced matrix) with a 2×2 block diagonal matrix. Then the *solution* of the linear system to calculate the preconditioned residual, can be split among processors working in parallel. In particular, if $m = 2^r$, and we have made s steps of the cyclic reduction before making the approximation, we can use 2^{r-s-1} processors to solve the linear system (each processor will deal with 2×2 blocks) with a constant degree of parallelism. Otherwise, we can use 2^{r-2} processors, with a degree of parallelism increasing from 2^{r-s-1} to 2^{r-2}.

Remark. We have chosen an incomplete 2×2 block cyclic reduction, instead of an incomplete pointwise cyclic reduction, for the following reasons:

(1) the computational cost of the resulting algorithms is the same;
(2) the stability properties are similar (see Section 3);
(3) the error made in the approximation is much smaller with the 2×2 block cyclic reduction than with the pointwise cyclic reduction, when the matrix is at least weakly diagonally dominant.

3. Analysis of Stability

We now investigate the properties of the reduced matrices T_i considering the simpler case of T_0 symmetric Toeplitz matrix. Let

$$T_0 = \begin{bmatrix} a & b & & \\ b & \ddots & \ddots & b \\ & \ddots & b & a \end{bmatrix}_{m \times m}, \qquad m = 2^r.$$

It follows that the generic reduced matrix T_i, at the ith step, has the structure

$$T_i = \begin{bmatrix} a_i & b & & & & & \\ b & a_i & s_i & & & & \\ & s_i & a_i & b & & & \\ & & b & \ddots & \ddots & & \\ & & & \ddots & \ddots & & \\ & & & & a_i & b & \\ & & & & b & a & \end{bmatrix}_{m/2^i \times m/2^i}, \tag{3.1}$$

where the a_i and s_i satisfy the following first order recurrences:

$$\begin{cases} a_i = a_{i-1} - \dfrac{s_{i-1}^2 a_{i-1}}{a_{i-1}^2 - b^2}, & a_0 = a, \\ s_i = \dfrac{s_{i-1}^2 b}{a_{i-1}^2 - b^2}, & s_0 = b. \end{cases} \tag{3.2}$$

To simplify the notations, we consider the quantities $z_i = a_i/b$, $u_i = s_i/b$. It follows that the matrix (3.1) becomes

$$T_i = b \cdot \begin{bmatrix} z_i & 1 & & & & & \\ 1 & z_i & u_i & & & & \\ & u_i & z_i & 1 & & & \\ & & 1 & \cdot & \cdot & & \\ & & & \cdot & \cdot & \cdot & \\ & & & & \cdot & z_i & 1 \\ & & & & & 1 & z_0 \end{bmatrix}_{m/2^i \times m/2^i}, \tag{3.3}$$

where

$$\begin{cases} z_i = z_{i-1} - \dfrac{u_{i-1}^2 z_{i-1}}{z_{i-1}^2 - 1}, & z_0 = a/b, \\ u_i = \dfrac{u_{i-1}^2}{z_{i-1}^2 - 1}, & u_0 = 1. \end{cases} \tag{3.4}$$

We can summarize our results as follows:

THEOREM. *If T_0 is weakly (strictly) diagonally dominant, then T_i is weakly (strictly) diagonally dominant.*

Proof. The proof is obtained by induction. The thesis holds for $i = 0$. If $|z_i| = 1 + |u_i|$, then $0 < u_{i+1} = u_i^2/(z_i^2 - 1) < 1$. It follows that

$$\begin{aligned} |z_{i+1}| - 1 &= \left| z_i - \frac{u_i^2 z_i}{z_i^2 - 1} \right| - 1 = |z_i| - \frac{u_i^2}{z_i^2 - 1} \cdot |z_i| - 1 \\ &= \frac{(|z_i| - 1)^2(|z_i| + 1) - u_i^2 |z_i|}{z_i^2 - 1} = \frac{u_i^2}{z_i^2 - 1} = |u_{i+1}|. \end{aligned}$$

In a similar manner $|z_{i+1}| > 1 + |u_{i+1}|$ if $|z_i| > 1 + |u_i|$. ∎

COROLLARY. *If T_0 is weakly diagonally dominant, u_i tends to 0 exponentially.*

Proof. At each step of the reduction we have $|z_i| = 1 + u_i$. Let us consider, for the sake of simplicity, the case $z_0 > 0$. It follows that $z_0 > z_i > 0$, $i \geqslant 1$ (if $z_0 < 0$, then $z_0 < z_i < 0$ holds). From the previous theorem we have $z_i = 1 + u_i$, and (3.4) can be split into two distinct equations:

$$z_i = \frac{2 \cdot z_{i-1}}{z_{i-1} + 1}, \qquad z_0 = 2,$$

$$u_i = \frac{u_{i-1}}{u_{i-1} + 2}, \qquad u_0 = 1.$$

The latter is a Riccati equation [6], whose solution is given by

$$u_i = (2^{i+1} - 1)^{-1}. \qquad \blacksquare$$

Remark. We observe that z_i tends to 1, as u_i approaches 0. Moreover, it is clear that if T_0 is strictly diagonally dominant, then u_i tends to 0 at least exponentially (by the comparison theorem [6]), while z_i tends to some $z^* > 1$.

4. NUMERICAL TESTS

The numerical tests here reported are obtained from the discretization of three test problems. The acronym INV_s^b is used for the modified INV algorithm with s-steps of an incomplete 2×2 cyclic reduction. All the tests were carried out on a net of transputers T800-20. The programming language used was Fortran, with the Express communication library [9].

The first test problem derives from the discretization of

$$-\Delta u = f, \qquad \text{in} \qquad \Omega = (0, 1) \times (0, 1)$$
$$u|_{\partial\Omega} = 0.$$

By using the step of discretization $h = (1 + 2^r)^{-1}$, one obtains $m = k = 2^r$. The dimension of the resulting linear system (1.1) is $m \cdot k = 2^{2r}$. In Table 1, the number of iterates to obtain convergence is reported for INV, INV_1, INV_2, INV_3.

The stop criterion used is, for all the problems, $\|\mathbf{r}_i\|_2 / \|\mathbf{r}_0\|_2 < 10^{-6}$ (where $\mathbf{r}_i$ is the current true residual, while $\mathbf{r}_0$ is the initial one), and the chosen starting point is $\mathbf{x}_0 = \mathbf{0}$.

TABLE I

r	INV	INV_3	INV_2	INV_1
4	7	7	7	9
5	12	12	12	14
6	20	20	20	23
7	36	36	36	42
8	69	69	69	81

From the above Table 1, it results that INV_2 is the best compromise between efficiency and degree of parallelism. By recalling that INV_2 can be efficiently implemented on a parallel machine with at most $p = 2^{r-3}$ processors, in Fig. 1 the speed-ups with respect to INV are outlined for $p = 2^{r-3b}(+)$ and $p = 2^{r-4b}(\mathrm{o})$.

The second test problem derives from the discretization of

$$-\lambda \cdot \Delta u = f, \qquad \text{in} \qquad \Omega = (0, 2) \times (0, 1)$$
$$u|_{\partial\Omega} = 0,$$

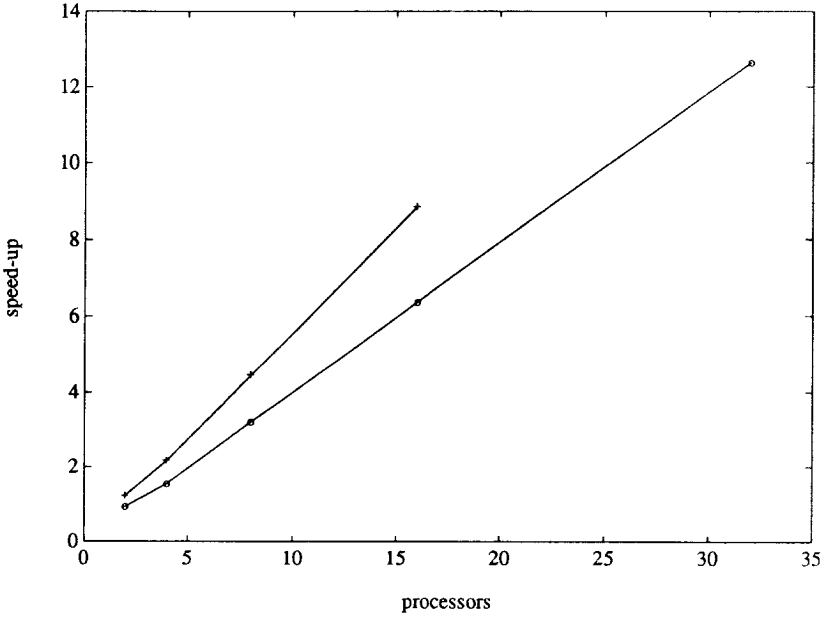

FIG. 1. Speed-up of problem 1 for INV_2.

TABLE II

r	INV	INV_3	INV_2	INV_1
4	6	6	6	7
5	10	10	10	11
6	18	18	18	21
7	32	32	32	39
8	62	62	62	75

$$\lambda = \begin{cases} 1, & \text{in} \quad (0, 1) \times (0, 1), \\ 1000, & \text{in} \quad (1, 2) \times (0, 1). \end{cases}$$

By using the steps of discretization $h = 2 \cdot (1 + 2^r)^{-1}$, one obtains $m = 2k = 2^r$. The dimension of the resulting linear system (1.1) is $m \cdot k = 2^{2r-1}$. In Table 2, the number of iterates to obtained convergence is reported for INV, INV_1, INV_2, INV_3.

In Fig. 2, the obtained speed-ups for INV_2 with respect to INV are outlined for $p = 2^{r-3b}(+)$ and $p = 2^{r-4b}(o)$.

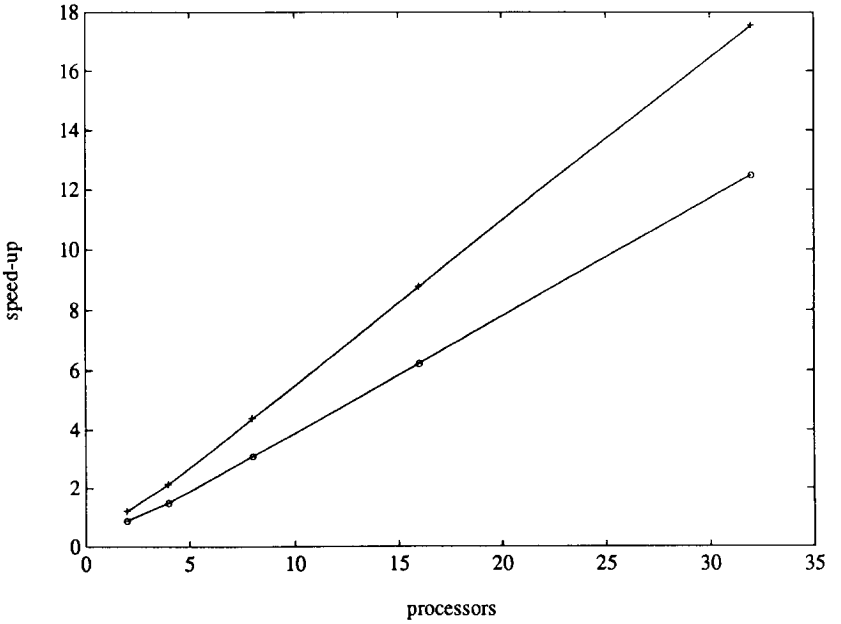

FIG. 2. Speed-up of problem 2 for INV_2.

TABLE III

r	INV	INV_3	INV_2	INV_1
4	11	11	11	13
5	18	18	19	23
6	34	34	34	43
7	65	65	65	81
8	127	127	127	160

The third test problem derives from the discretization of:

$$-\lambda \cdot \Delta u + \sigma \cdot u = \sigma, \qquad \text{in} \quad \Omega = (0, 2) \times (0, 1)$$

$$\left.\frac{\partial u}{\partial \mathbf{n}}\right|_{\partial\Omega} = 0,$$

$$(\lambda, \sigma) = \begin{cases} (1, .01), & \text{in} \quad (0, .25] \times (0, 1), \\ (2, .03), & \text{in} \quad (.25, 1] \times (0, 1), \\ (3, .05), & \text{in} \quad (1, 2) \times (0, 1), \end{cases}$$

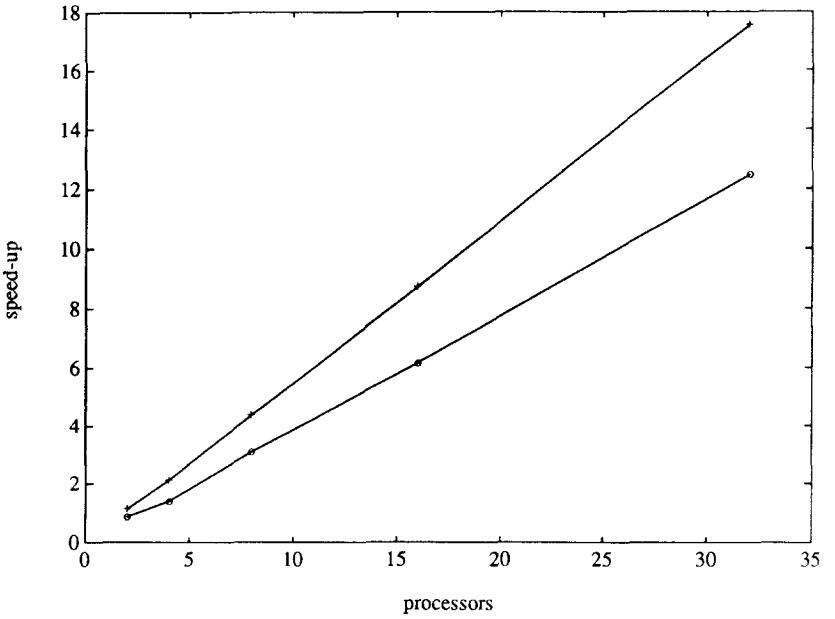

FIG. 3. Speed-up of problem 3 for INV_2.

where n is the unit vector normal to $\partial\Omega$. By using the step of discretization $h = 2\cdot(1 + 2^r)^{-1}$, one obtains $m = 2k = 2^r$. The dimension of the resulting linear system (1.1) is $m\cdot k = 2^{2r-1}$. In Table 3, the number of iterates to obtain convergence is reported for INV, INV_1, INV_2, INV_3.

In Fig. 3, the obtained speed-ups for INV_2 with respect to INV are outlined for $p = 2^{r-3b}(+)$ and $p = 2^{r-4b}(o)$.

5. ACKNOWLEDGMENTS

We are very indebted to Prof. Donato Trigiante for his suggestions and helpful comments and to Mrs. Paulene Butts for carefully reading the manuscript.

REFERENCES

1. P. Amodio, Optimized cyclic reduction for the solution of tridiagonal linear systems on parallel computers. Submitted for publication.
2. P. Amodio and F. Mazzia, Stability of cyclic reduction for the solution of linear tridiagonal systems. Submitted for publication.
3. L. Brugnano and M. Marrone, Vectorization of some block preconditioned conjugate gradient methods. *Parallel Comput.* **14,** 191–198 (1990).
4. P. Concus, G. H. Golub, and G. Meurant, Block preconditioning for the conjugate gradient method. *SIAM J. Sci. Statist. Comput.* **6,** 220–252 (1985).
5. P. Concus and G. Meurant, On computing the INV block preconditionings for the conjugate gradient method. *BIT* **26,** 493–504 (1986).
6. V. Lakshmikantham and D. Trigiante, *Theory of Difference Equations: Numerical Methods and Applications,* Ser. Mathematics in Science and Engineering, Vol. 181. Academic Press, New York/London (1988).
7. H. A. van der Vorst, Large tridiagonal and block tridiagonal linear systems on vector and parallel computers. *Parallel Comput.* **5,** 45–54 (1987).
8. R. S. Varga, *Matrix Iterative Analysis.* Ser. Automatic Computation, Prentice-Hall, Englewood Cliffs, NJ (1962).
9. Express: 3L Fortran Reference. Parasoft Corporation.