BIT 24 (1984), 623–633

THE BLOCK PRECONDITIONED CONJUGATE GRADIENT METHOD ON VECTOR COMPUTERS

GÉRARD MEURANT

CEA Centre d'Etudes de Limeil-Valenton, BP 27, 94190 Villeneuve St Georges, France

Dedicated to Germund DALHQUIST on his sixtieth birthday

Abstract.

We present vectorizable versions of the block preconditioners introduced by Concus, Golub & Meurant [2] for use with the conjugate gradient method. In [2] it was shown that the block preconditioners give less computational work than the classical point ones on conventional serial computers.

Here we give numerical results for various vector computers, CDC Cyber 205, CRAY 1-S, CRAY X-MP and for several problems, which show that for most cases the block method with slight modifications, gives better results also on vector computers.

1. Introduction.

In this paper we are concerned with the use of the preconditioned conjugate gradient method on vector computers.

We consider linear systems arising from the standard five point finite difference discretization of second order elliptic equations $Ax = b$, where A is a block tridiagonal symmetric matrix of order $N = n^2$:

$$A = \begin{pmatrix} D_1 & A_2^T & & & \\ A_2 & D_2 & A_3^T & & \\ & \ddots & \ddots & \ddots & \\ & & A_{n-1} & D_{n-1} & A_n^T \\ & & & A_n & D_n \end{pmatrix}$$

The matrices D_i of order n are tridiagonal and the matrices A_i of order n are diagonal.

For the sake of simplicity we make the following hypothesis. Hypothesis (H): Let the elements a_{ij} of A be such that $a_{ii} > 0$, $\forall i$, $a_{ij} \leqslant 0$, $\forall i$, $\forall j \neq i$

$$a_{ii} \geqslant \sum_{j \neq i} |a_{ij}|, \qquad \forall i.$$

Received January 1984. Revised May 1984.

The matrices D_i are strictly diagonally dominant i.e.

$$a_{ii} > |a_{i,i+1}| + |a_{i,i-1}|, \quad \forall i.$$

The elements whose indices have no meaning are supposed to be zero.

Hypothesis (H) implies that A is a symmetric positive definite M-matrix.

Let M be a symmetric positive definite matrix, the preconditioned conjugate gradient algorithm is as follows (see [3]).

Let x^0 be given, $r^0 = b - Ax^0$ and define p^{-1} arbitrarily; for $k = 0, 1, \ldots$ perform the following steps until convergence

$$Mz^k = r^k$$

$$\beta_k = (Mz^k, z^k)/(Mz^{k-1}, z^{k-1}) \qquad k \geqslant 1, \qquad \beta_0 = 0$$

$$p^k = z^k + \beta_k p^{k-1}$$

$$\alpha_k = (Mz^k, z^k)/(Ap^k, p^k)$$

$$x^{k+1} = x^k + \alpha_k p^k$$

$$r^{k+1} = r^k - \alpha_k Ap^k.$$

For our problem most parts of the algorithm are trivially vectorizable. Each iteration of the conjugate gradient requires two innerproducts and three linked triads. On the CRAYs we use Fortran except for SDOT (innerproduct) and SAXPY (linked triad) which are assembly language (CAL) routines that run asymptotically at 74 and 45 Mflops (Million of floating point operations per second). On the 2 pipes Cyber 205 we use Q8SDOT and the CDC Fortran dialect which give respectively 90 and 165 Mflops for innerproducts and linked triads.

It has been shown in [6] how to compute efficiently the product Ap^k by diagonals on the STAR 100, and we will use the same method on the CRAY and the Cyber 205.

The main difficulties arise in solving the system $Mz^k = r^k$ at each iteration. Several preconditioners suited for vector computers were presented recently. Dubois, Greenbaum and Rodrigue [4] used a truncated Neumann series, H. A. Van der Vorst [14, 15] vectorized the classical IC(1, 1), Lichnewsky [11] suggested to use nested dissection techniques, Johnson, Michelli and Paul [8] show how to use efficient polynomial preconditioners, Jordan [9, 10] coded the conjugate gradient algorithm in CAL and compared several of the previous approaches with Chebyshev polynomials.

Here we introduce new solutions to this problem for block tridiagonal matrices. We use the block preconditioner developed by Concus, Golub and Meurant [2] and show how to make it efficient for vector computers.

In section 2 we recall the block method. Section 3 shows how to modify this method for use on a vector computer. In section 4 we give numerical results and show comparisons with the Van der Vorst method for CDC Cyber 205, CRAY 1-S, CRAY X-MP with one processor.

It appears that, for most problems, the modified block methods give the best results on vector computers.

For simplicity the technique of Eisenstat [5] for reducing the work requirements of the conjugate gradient is not incorporated here since we wish only to compare the relative merits of the different preconditionings.

2. Block preconditioning.

The preconditioner INV(1) given in Concus, Golub and Meurant [2] is as follows.

Let L be the matrix

$$L = \begin{pmatrix} 0 & & & & \\ A_2 & 0 & & & \\ & A_3 & 0 & & \\ & & \ddots & \ddots & \\ & & & A_n & 0 \end{pmatrix}$$

The preconditioning matrix M is given by

$$M = (\Delta + L)\Delta^{-1}(\Delta + L^T),$$

where Δ is a block diagonal matrix

$$\Delta = \begin{pmatrix} \Delta_1 & & & \\ & \Delta_2 & & \\ & & \ddots & \\ & & & \Delta_n \end{pmatrix}$$

whose matrices Δ_i are computed by the following formulas

$$\Delta_1 = D_1$$
$$\Delta_i = D_i - A_i \Omega_{i-1}(1) A_i^T, \; 2 \leqslant i \leqslant n;$$

$\Omega_i(1)$ is a tridiagonal matrix and

$$(\Omega_i(1))_{lk} = (\Delta_i^{-1})_{lk}, \; k = l-1, l, l+1.$$

In [2] the following theorem is proved.

THEOREM 1. *Under hypothesis* (H) *each* Δ_i *computed by* $INV(1)$ *is strictly diagonally dominant with positive diagonal elements and non-positive off diagonal elements.*

This proves that we can perform the incomplete block factorization and that M is positive definite. Actually a more general result can be proved, namely that if A is a general M-matrix then each Δ_i is an M-matrix but hypothesis (H) will be convenient for our purposes.

To solve the system $Mz = r$ we successively solve

$$(\Delta + L)y = r$$

$$(I + \Delta^{-1}L^T)z = y.$$

Partitioning r, z and y in an obvious way, we get

$$\begin{cases} \Delta_1 y_1 = r_1 \\ \Delta_i y_i = r_i - A_i y_{i-1}, & 2 \leqslant i \leqslant n \end{cases}$$

$$\begin{cases} z_n = y_n \\ \Delta_i w_i = A_{i+1}^T z_{i+1}, & i = n-1, \dots, 1. \\ z_i = y_i - w_i \end{cases}$$

We have to solve $2n$ dependent tridiagonal systems which is not a vector operation. So if we want to use INV(1) on a vector computer we have to use methods to vectorize the solution of tridiagonal systems (see, for instance, Hockney, Jesshope [7]) or we have to modify the preconditioner.

3. Vector preconditioner.

The idea to vectorize INV(1) is to use an approximation to Δ_i^{-1} to solve the tridiagonal systems.

We give two methods to approximate the inverses of the Δ_i.

3.1. *Banded approximation of the inverse.*

We replace the exact solution z_i of $\Delta_i z_i = c_i$ by y_i

$$(*) \qquad y_i = \Omega_i(j)c_i,$$

where $\Omega_i(j)$ is a symmetric banded matrix with $2j+1$ diagonals whose elements within the band are the same as those of Δ_i^{-1}.

This can be computed efficiently using the Choleski decomposition of Δ_i.

Let

$$\Delta_i = \begin{pmatrix} a_1 & -b_1 & & & \\ -b_1 & a_2 & -b_2 & & \\ & \ddots & \ddots & \ddots & \\ & & -b_{n-2} & a_{n-1} & -b_{n-1} \\ & & & -b_{n-1} & a_n \end{pmatrix},$$

where $a_i > 0$, $1 \leqslant i \leqslant n$, $b_i > 0$, $1 \leqslant i \leqslant n-1$, $a_i > b_i + b_{i-1}$.

In [2] we use the known fact that the inverse Δ_i^{-1} can be written as

$$\Delta^{-1} = \begin{pmatrix} u_1v_1 & u_1v_2 & \dots & u_1v_n \\ u_1v_2 & u_2v_2 & \dots & u_2v_n \\ \vdots & \vdots & & \vdots \\ u_1v_n & u_2v_n & \dots & u_nv_n \end{pmatrix},$$

$u_i > 0, v_i > 0,$ $i = 1, \dots, n$.

The products u_iv_j can be computed by the following algorithm (P. Concus [1]).

Let

$$\Delta_i = \tilde{L}\tilde{D}\tilde{L}^T$$

be the Choleski decomposition of Δ_i.

$$\tilde{L} = \begin{pmatrix} 1 & & & & \\ -\gamma_1 & 1 & & & \\ & -\gamma_2 & 1 & & \\ & & \ddots & \ddots & \\ & & & -\gamma_{n-1} & 1 \end{pmatrix} \qquad \tilde{D} = \begin{pmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_n \end{pmatrix},$$

$d_1 = a_1$, $d_i = a_i - b_{i-1}^2/d_{i-1}$, $2 \leqslant i \leqslant n$, $\gamma_i = b_i/d_i$, $1 \leqslant i \leqslant n$.

Define $\gamma_0 = 1$, $q_n = \gamma_{n-1}/d_n$ and $q_i = \gamma_{i-1}\gamma_i q_{i+1} + \gamma_{i-1}/d_i$, $i = n-1, \dots, 1$. Then

$$u_iv_i = q_i/\gamma_{i-1}$$

$$u_iv_{i+1} = q_{i+1}$$

$$u_iv_j = \gamma_i \dots \gamma_{j-2}q_j, \qquad j > i+1.$$

This is not a vector operation (although first order recurrences can be vectorized, see [7]) but we need only to compute the d_i, γ_i and q_i once at the beginning of the algorithm and then we store the $j+1$ diagonals we need to compute: $y_i = \Omega_i(j)c_i$ which is now a vector operation. The conjugate gradient iteration is then completely vectorized. We dénote this method by INVVj(1).

But one can ask if the preconditioning matrix M corresponding to this method is positive definite. The answer is yes if the matrices Δ_i are enough diagonally dominant.

In [2] it is proved that the sequence $\{u_i\}_{i=1}^n$ is strictly increasing and the sequence $\{v_i\}_{i=1}^n$ strictly decreasing.

Let α_i and β_i defined by

$$u_{i+1} = \alpha_i u_i, \quad v_{i+1} = \beta_i^{-1} v_i, \quad i = 1,\dots,n-1,$$

then

$$\alpha_i > \frac{a_i - b_{i-1}}{b_i} > 1, \qquad \beta_i > \frac{a_{i+1} - b_{i+1}}{b_i} > 1$$
$$\alpha_1 = \frac{a_1}{b_1} \qquad \beta_{n-1} = \frac{a_n}{b_{n-1}}$$

Let

$$\varrho = \left\{ \min_{2 \leqq i \leqq n-1} (a_1/b_1, (a_i - b_{i-1})/b_i) \right\}^{-1} < 1;$$
$$\delta = \left\{ \min_{2 \leqq i \leqq n-1} (a_n/b_{n-1}, (a_i - b_i)/b_{i-1}) \right\}^{-1} < 1.$$

We have the following sufficient condition.

PROPOSITION 2. *If*

$$(\varrho - \varrho^{j+1})/(1-\varrho) + (\delta - \delta^{j+1})/(1-\delta) < 1$$

then $\Omega_i(j)$ is positive definite.

The proof is straightforward. Consider the kth line of $\Omega_i(j)$ and the sum of the off-diagonal elements

$$S_k = v_k \sum_{l=k-j}^{k-1} u_l + u_k \sum_{l=k+1}^{k+j} v_l$$

where the elements with negative indices are taken to be zero.

$$S_k = u_k v_k \left(\frac{1}{\alpha_{k-1}} + \frac{1}{\alpha_{k-1}\alpha_{k-2}} + \dots + \frac{1}{\alpha_{k-1} \cdots \alpha_{k-j}} \right)$$
$$+ u_k v_k \left(\frac{1}{\beta_k} + \frac{1}{\beta_k \beta_{k+1}} + \dots + \frac{1}{\beta_k \cdots \beta_{k+j-1}} \right)$$

$$S_k \leqslant u_k v_k (\varrho + \varrho^2 + \dots + \varrho^j + \delta + \delta^2 + \dots + \delta^j) < u_k v_k.$$

This shows that $\Omega_i(j)$ is strictly diagonally dominant and hence positive definite.

Note that, for instance, the matrix with $a_i = 4$, $b_i = 1$, $\forall i$ verify the criteria of proposition 2 and therefore $\Omega_i(j)$ is positive definite for any value of j, but this is not the case for the matrix with $a_i = 2.2$, $b_i = 1$, $\forall i$, $j = 1$.

The condition of proposition 2 is verified for most problems but, of course, one can construct examples where the matrix M given by INVVj(1) is not positive definite. A way to obtain a positive definite preconditioner is the following method.

3.2. *Approximation from the Choleski factors.*

For the approximate solution of $\Delta_i z_i = c_i$ we use a similar method as the one in [14] and take as a solution $y_i = \sum_i(j)c_i$, where

$$\textstyle\sum_i(j) = (I+\bar{L}^T+(\bar{L}^T)^2+\cdots+(\bar{L}^T)^j)\tilde{D}^{-1}(I+\bar{L}+\bar{L}^2+\cdots+\bar{L}^j)$$

and

$$\bar{L} = \begin{pmatrix} 0 & & & & \\ \gamma_1 & 0 & & & \\ & \gamma_2 & 0 & & \\ & & \ddots & \ddots & \\ & & & \gamma_{n-1} & 0 \end{pmatrix}.$$

Then it is easy to show that

$$\bar{L}^2 = \begin{pmatrix} 0 & & & & & \\ 0 & 0 & & & & \\ \gamma_1\gamma_2 & 0 & 0 & & & \\ & \gamma_2\gamma_3 & 0 & 0 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \gamma_{n-1}\gamma_{n-2} & 0 & 0 \end{pmatrix}, \quad \bar{L}^3 = \begin{pmatrix} 0 & & & & & & \\ 0 & 0 & & & & & \\ 0 & 0 & 0 & & & & \\ \gamma_1\gamma_2\gamma_3 & 0 & \ddots & \ddots & & & \\ & \gamma_2\gamma_3\gamma_4 & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \ddots & \\ & & & \gamma_{n-3}\gamma_{n-2}\gamma_{n-1} & 0 & 0 & 0 \end{pmatrix}$$

and so on.

Once again everything is completely vectorizable. We denote this preconditioner by INVCj(1); obviously the matrix M is positive definite.

4. Numerical results.

Here we compare the preconditioners introduced in the last section on several problems and different vector computers CDC Cyber 205 2 pipes, CRAY 1-S, CRAY X-MP (one processor). As a reference we give results for the standard IC(1, 1) see [12] and the vectorized version (hereafter denoted by ICVDV) given by H. A. Van der Vorst [14] using a three term truncation.

For all the methods on the CRAYs everything is coded in Fortran except for the two innerproducts and the three linked triads using SDOT and SAXPY, while on the Cyber 205 we use Q8SDOT and CDC vector Fortran.

4.1. *Model problem.*

We use the standard five point discretization for

$$-\Delta u = f, \qquad u|_{\partial\Omega} = 0,\ \Omega =]0,1[\times]0,1[.$$

The mesh size is $h = 1/(n+1)$ and hence $N = n^2$.

The stopping criterion is $\|r^k\|_\infty/\|r^0\|_\infty \leqq 10^{-6}$.

In this example the matrix M for INVVj(1) is positive definite.

First of all, we choose j for INVVj(1). The results are the following for $n = 50$

	it.	cp time (s)	storage/N
INV(1)	15	0.135	2
INVV1(1)	31	0.064	2
INVV2(1)	20	0.048	3
INVV3(1)	16	0.046	4

In the sequel we use INVV3(1).

In tables 1 and 2 we give the number of iterations, the computing time (not including the time to compute the preconditionings) and the Megaflops rates for different computers.

Table 1. *Results for n = 50.*

	ICVDV	INVV3(1)
CDC Cyber 205	34 it 0.069 sec 45 Mflops	16 it 0.046 sec 42 Mflops
CRAY 1-S	35 it 0.098 sec 33 Mflops	16 it 0.046 sec 42 Mflops
CRAY X-MP	35 it 0.043 sec 75.4 Mflops	16 it 0.020 sec 96 Mflops

Table 2. *Results for* $n = 150$.

	ICVDV	INVV3(1)
CDC Cyber 205	84 it 0.993 sec 70.4 Mflops	39 it 0.634 sec 67.8 Mflops
CRAY 1-S	79 it 1.741 sec 37.7 Mflops	36 it 0.831 sec 47.7 Mflops
CRAY X-MP	79 it 0.671 sec 98 Mflops	36 it 0.336 sec 118 Mflops

One can remark that ICVDV gives a better Mflops rate than INVV3(1) on the Cyber 205 but the oppositie is true on the CRAY. This can be explained because the relative part of the solution of $Mz = r$ is smaller in ICVDV and consequently the gain given by the very fast linked triads on the Cyber 205 becomes more important.

Table 3 compares vectorized and non vectorized methods on the CRAY-1 for $n = 50$.

Table 3

	IC(1, 1)	ICVDV	INV(1)	INVV3(1)	INVC3(1)
# it.	35	35	15	16	20
comp. t.	0.172 s	0.098 s.	0.135 s	0.046 s	0.065 s
Mflops	14.7	32.9	9.4	42	37.7

Even if solving $Mz = r$ is a scalar operation in both cases, IC(1, 1) has a better Mflops rate than INV(1) because of differences in the coding. IC(1, 1) uses scalar loops of length N when INV(1) uses nested loops of length n.

For this problem the benefits of vectorization are clear and INVV3(1) is the best preconditioner.

4.2. *Second test problem.*

We solve the five point discretization of

$$-\frac{\partial}{\partial x}\left(\lambda_1 \frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(\lambda_2 \frac{\partial u}{\partial y}\right) + \sigma u = f \quad \text{in} \quad \Omega =]0, 1[\times]0, 1[, \quad \left.\frac{\partial u}{\partial n}\right|_{\partial\Omega} = 0,$$

where $\sigma = 0.01$, $\lambda_1 = 100$, $\lambda_2 = 1$, $h = 1/49$, i.e. $n = 50$ and with the same stopping criteria as for the model problem.

This problem is designed to be difficult for the vectorized methods because the block matrices involved are very weakly diagonally dominant and it is not easy to approximate their inverses. In fact, the matrix M given by INVV3(1) is not positive definite. The results are given in Table 4 for the CRAY-1.

Table 4

	IC(1, 1)	ICVDV	INV(1)	INVV3(1)	INVC3(1)
# it.	30	148	12	> 200	181
comp. t.	0.143 s	0.416 s	0.112 s	–	0.591 s
Mflops	14.7	32.9	9.1	–	37.5

For this problem the non vectorized versions are faster than the vectorized ones because Neumann series with a few terms are too poor approximations for the inverses. It would have been better to use INV(1) with a standard method to vectorize the solution of tridiagonal systems.

Note also that H. A. Van der Vorst proposed in [15] to vectorize IC(1, 1) changing the order of computation (diagonalwise). This could work quite well for this example.

Of course one can argue that we can interchange x and y and get good results for all the methods, but this example was intended to model problems with varying coefficients and one cannot know a priori what is a good numbering of the unknowns.

4.3. *Third test problem.*

This is an example described in Varga [16] and used in [2] and [14].

$$-\operatorname{div}(\lambda \nabla u)+\sigma u = 0 \quad \text{in} \quad \Omega =]0, 2.1[\times]0, 2.1[\quad , \left.\frac{\partial u}{\partial n}\right|_{\partial\Omega} = 0.$$

Let the areas A, B and C be defined by the points:
A: $(0,0)$, $(2.1,0)$, $(2.1,1)$, $(1,1)$, $(1,2.1)$, $(0,2.1)$, $(0,0)$;
B: $(1,1)$, $(2,1)$, $(2,2)$, $(1,2)$, $(1,1)$ and
C: $(2,1)$, $(2.1,1)$, $(2.1,2.1)$, $(1,2.1)$, $(1,2)$, $(2,2)$, $(2,1)$

and further

	λ	σ
A	1	0.02
B	2	0.03
C	3	0.05

We take $h = 1/42$, i.e. $N = 1849$.

Results are given in Table 5 for the CRAY-1.

Table 5

	IC(1, 1)	ICVDV	INV(1)	INVV3(1)	INVC3(1)
# it.	39	40	18	20	20
comp. t.	0.141 s	0.086 s	0.121 s	0.045 s	0.0499 s
Mflops	14.8	31.7	6.6	40.7	36.3

5. Conclusions.

We have presented several possibilities to vectorize the block methods given in Concus, Golub and Meurant [2]. When it works INVV3(1) is the best preconditioner, but from the experiments it seems that there is no all-round best method.

However, for a large class of problems the block methods are to be preferred over the point ones both on scalar and vector computers.

Acknowledgement.

The author thanks the referee for helpful comments on a first version of this paper.

REFERENCES

1. P. Concus, Private communication (1983).
2. P. Concus, G. H. Golub and G. Meurant, *Block preconditioning for the conjugate gradient method*, Lawrence Berkeley Laboratory LBL-14856 Berkeley, CA (1982), to appear in Siam J. on Stat. and Sci. Comp.
3. P. Concus, G. H. Golub and D. P. O'Leary, *A Generalized Conjugate Gradient Method for the Numerical Solution of Elliptic Partial Differential Equations* in *Sparse Matrix Computations*, J. R. Bunch & D. J. Rose eds. Academic Press, New York (1976).
4. P. F. Dubois, A. Greenbaum and G. Rodrigue, *Approximating the inverse of a matrix for use in iterative algorithms on vector computers*, Computing v. 22, pp. 257–268 (1979).
5. S. Eisenstat, *Efficient implementation of a class of preconditoned conjugate gradient methods*, Siam J. on Stat. and Sci. Comp. v. 2, pp. 1–4 (1981).
6. A. Greenbaum and G. Rodrigue, *The incomplete Choleski conjugate gradient method for the STAR*, Res. Rep. UCID–17574 Lawrence Livermore Laboratory Livermore, CA (1977).
7. R. W. Hockney and C. R. Jesshope, *Parallel Computers*, Adam Hilger, Bristol (1981).
8. O. G. Johnson, C. A. Michelli and G. Paul, *Polynomial preconditioning for conjugatě gradient calculations*, Siam J. on Numer. Anal. v. 20, pp. 363–376 (1983).
9. T. Jordan, *A Guide to Parallel Computations and some CRAY-1 Experiences* in *Parallel Computations*, G. Rodrigue ed., Academic Press, New York (1982).
10. T. Jordan, *Conjugate gradient preconditioners for vector and parallel processors*, in *Elliptic Problem Solvers*, G. Birkhoff ed. Academic Press, New York (1983).
11. A. Lichnewsky, *Solving some linear systems arising in finite elements methods on parallel processors*, Siam national meeting, Stanford CA (1981).
12. J. A. Meijerink and H. A. Van Der Vorst, *Guidelines for the usage of incomplete decompositions in solving sets of linear equations as they occur in practical problems*, J. of Comp. Physics v. 44, pp. 134–155 (1981).
13. G. Rodrigue and D. Wolitzer, *Preconditioning by incomplete block cyclic reduction*, Lawrence Livermore Laboratory UCID-19502, Livermore CA (1982).
14. H. A. Van Der Vorst, *A vectorizable version of some ICCG methods*, Siam J. on Stat. and Sci. Comp. v. 3, pp. 350–356 (1982).
15. H. A. Van Der Vorst, *On the vectorization of some simple ICCG methods*, Talk at 1er Colloque Calcul vectoriel et parallèle AFCET, GAMNI, ISINA 17–18 Mars 1983, Paris.
16. R. S. Varga, *Matrix Iterative Analysis*, Prentice Hall, Englewood Cliffs (1962).