Computing 22, 257—268 (1979)

Computing
© by Springer-Verlag 1979

Approximating the Inverse of a Matrix for Use in Iterative Algorithms on Vector Processors*

P. F. Dubois, A. Greenbaum, and G. H. Rodrigue, Livermore, California

Received November 7, 1977

Abstract — Zusammenfassung

Approximating the Inverse of a Matrix for Use in Iterative Algorithms on Vector Processors. Most iterative techniques for solving the symmetric positive-definite system $Ax=b$ involve approximating the matrix A by another symmetric positive-definite matrix M and then solving a system of the form $Mz=d$ at each iteration. On a vector machine such as the CDC-STAR-100, the solution of this new system can be very time consuming. If, however, an approximation M^{-1} can be given to A^{-1}, the solution $z=M^{-1}d$ can be computed rapidly by matrix multiplication, a fast operation on the STAR. Approximations using the Neumann expansion of the inverse of A give reasonable forms for M^{-1} and are presented. Computational results using the conjugate gradient method for the "5-point" matrix A are given.

Näherungsweise Berechnung einer in iterativen Algorithmen auf Vektor-Prozessoren verwendbaren Matrixinversen. Die meisten iterativen Methoden zur Lösung des symmetrischen positiv-definitiven Systems $Ax=b$ enthalten die Näherung der Matrix A durch eine andere symmetrische positiv-definitive Matrix M und anschließend daran die Lösung eines Systems der Art $Mz=d$ bei jeder Wiederholung. Auf einer Vektor-Maschine wie der CDC-STAR-100 kann die Lösung dieses neuen Systems sehr zeitraubend sein. Wenn jedoch eine Näherung M^{-1} zu A^{-1} gegeben werden kann, so kann die Lösung $z=M^{-1}d$ sehr schnell durch Matrixmultiplikation errechnet werden. Diese Kalkulation kann auf dem STAR schnell ausgeführt werden. Näherungen, bei denen die Neumann-Entwicklung der Inversen von A verwendet wird, ergeben angemessene Ausdrücke für M^{-1}. Diese Ausdrücke sind angeführt. Die mit Hilfe der Konjugierten-Gradienten-Methode errechneten Resultate für die „5-Punk"-Matrix A sind angegeben.

1. Introduction

Most iterative techniques for solving a linear system $A\underset{\sim}{x}=\underset{\sim}{b}$, involve approximating the matrix A by a matrix M and, at each iteration, solving a system of the form $M\underset{\sim}{z}=\underset{\sim}{d}$. Often M is lower triangular, as in SOR, or a product of a known lower triangular matrix and upper triangular matrix, as in incomplete Cholesky decomposition [5]. In either case, backsolving on the triangular matrix is required to calculate the vector $\underset{\sim}{z}$, and the process is inherently scalar, since previous components of $\underset{\sim}{z}$ must be known before the next component can be calculated.

* Work performed under the auspices of the U.S. Energy Research & Development Administration under contract No. W-7405-Eng-48.

To avoid the high cost of backsolving on a vector machine, one might consider taking an easily calculated and/or easily stored matrix M^{-1} as an approximation to A^{-1}, and, at each iteration, performing matrix multiplication to obtain $\underline{z}=M^{-1}\underline{d}$. On a scalar machine, the cost of backsolving on a triangular matrix and that of multiplying by a matrix with the same number of nonzero elements, is approximately the same. On the CDC-7600, however, matrix multiplication can be speeded up by the use of STACKL 1 B [4], whereas a comparable gain for the backsolving does not appear possible unless one goes to assembly language coding (in which case one may be able to put several instructions in the instruction stack). On the CDC-STAR-100, the difference becomes even more dramatic, since the vector operations involved in matrix multiplication are faster, the scalar operations involved in backsolving are slower, and stacking of instructions is not possible [6]. For these reasons, then, we are led to consideration of the problem of calculating an approximation M^{-1} to the matrix A^{-1}.

A very simple such approximation would be to take $M^{-1}=(\text{diagonal}(A))^{-1}$. In this paper we discuss a method for improving on this, or any other initial approximation M^{-1}, using a truncated Neumann series approximation to A^{-1}. We will show that for first order iterative methods, one iteration with a p-term Neumann series is equivalent to p iterations with the usual 1-term approximation, and involves exactly the same amount of work. Hence there is no gain to be made here. When used with a second order method such as the Chebyshev method or the conjugate gradient method, however, the situation changes. We discuss the effect on the eigenvalues of the iteration matrix and the corresponding bounds on the number of iterations required for the Chebyshev method and the conjugate gradient method. We show that a p-term Neumann series can reduce the number of iterations required by the conjugate gradient algorithm by at most a factor of p. We present comparisons of the conjugate gradient method using different numbers of terms in the Neumann series and using incomplete Cholesky decomposition, with regard to number of iterations and time required on the 7600 and the STAR.

2. First Order Methods

First order methods (cf. [8]) for solving $A\,x=b$ are obtained by first defining the appropriate splitting $A=M-N$ and then iterating according to

$$M\,\underline{x}^{(k+1)}=N\,\underline{x}^{(k)}+\underline{b},\quad \underline{x}^{(0)}=\text{given}, \tag{2.1}$$

or equivalently,

$$\underline{x}^{(k+1)}=\underline{x}^{(k)}+M^{-1}(\underline{b}-A\,\underline{x}^{(k)}). \tag{2.2}$$

The matrix M is chosen so that the above iterative method converges to the solution of $A\,x=b$ and the solution of the system in (2.1) is simple. Alternatively, M^{-1} can be defined a-priori in which case the iterative method given by (2.2) can be used.

Either method converges for every starting value $\underline{x}^{(0)}$ if and only if (see [7])

$$\rho(M^{-1}N)=\max\{|\lambda| : \lambda \text{ an eigenvalue of } M^{-1}N\}<1.$$

Since $A = M(I - M^{-1}N)$, we can write in this case

$$\begin{aligned} A^{-1} &= (I - M^{-1}N)^{-1} M^{-1} \\ &= \left(\sum_{i=0}^{\infty} (M^{-1}N)^i\right) M^{-1} \\ &= \left(\sum_{i=0}^{\infty} (I - M^{-1}A)^i\right) M^{-1}. \end{aligned}$$

We might consider improving the approximation M^{-1} to A^{-1} by instead using

$$M_p^{-1} = \left(\sum_{i=0}^{p-1} (I - M^{-1}A)^i\right) M^{-1}, \quad p > 1. \tag{2.3}$$

We need not explicitly calculate M or M_p in order to calculate M_p^{-1}; we need only have some initial approximation M^{-1}, such that $\rho(I - M^{-1}A) < 1$. If M^{-1} is taken to be $(\text{diagonal}(A))^{-1}$, then this spectral radius can be shown to be less than one if A is strictly or irreducibly diagonally dominant [7, p. 73], or if the off-diagonal elements of A are nonpositive, A is nonsingular, and $A^{-1} > 0$ (element-wise) [7, p. 84]. A more generally applicable algorithm might be obtained by using a different M^{-1}, and this possibility is currently being investigated.

Note also that the matrix M_p^{-1} need not actually be calculated and stored, since at each iteration we simply form the product $M_p^{-1}\underset{\sim}{d} = M_p^{-1}(\underset{\sim}{b} - A\underset{\sim}{x}^{(k)})$. This can be accomplished as follows:

```
$\underset{\sim}{w} := M^{-1}\underset{\sim}{d}$;
product$:= \underset{\sim}{w}$;
For $i := 1$ step 1 until $p-1$ do
product$:= \underset{\sim}{w} + (I - M^{-1}A)\cdot$product.
```

How, then, does the first order iteration with $M^{-1} = M_1^{-1}$ compare to that with M_p^{-1}, for $p > 1$? Suppose $\underset{\sim}{x}^{(0)} = \underset{\sim}{x}_p^{(0)}$ is given, and for $k = 0, 1, 2, \ldots$, we have

$$\underset{\sim}{x}^{(k+1)} = (I - M^{-1}A)\underset{\sim}{x}^{(k)} + M^{-1}\underset{\sim}{b}$$

$$\underset{\sim}{x}_p^{(k+1)} = (I - M_p^{-1}A)\underset{\sim}{x}_p^{(k)} + M_p^{-1}A.$$

From (2, 3), one can see that $(I - M_p^{-1}A) = (I - M^{-1}A)^p$ so that $\underset{\sim}{x}^{(kp)} = \underset{\sim}{x}_p^{(k)}$, for $k = 0, 1, 2, \ldots$. Thus, one step of the iteration with M_p^{-1} is precisely equivalent to p steps of the iteration with M^{-1}.

Using the form (2.2) for the iteration, we find that one step using M^{-1} requires that we multiply A times a vector and M^{-1} times a vector, and then do two vector additions. Hence p steps requires:

$A \times$ vector	p times
$M^{-1} \times$ vector	p times
vector + vector	$2p$ times.

Using M_p^{-1} we have

$$\underset{\sim}{x}_p^{(k+1)} = \underset{\sim}{x}_p^{(k)} + \left(\sum_{i=0}^{p-1} (I - M^{-1}A)^i\right) M^{-1}(\underset{\sim}{b} - A\underset{\sim}{x}_p^{(k)}),$$

which requires exactly the same amount of work. Thus, using M_p^{-1} instead of M^{-1} does not save any work and is actually somewhat disadvantageous, since the standard algorithm may satisfy the convergence criterion on a step which is not a multiple of p.

3. Second Order Methods

When used in a second order method such as the Chebyshev method or the conjugate gradient method, one step of the algorithm using M_p^{-1} is no longer equivalent to p steps of the algorithm using M^{-1}. In fact, for the conjugate gradient method, we will show that this factor of p reduction in the number of iterations is the best possible improvement and, in general, will not actually be achieved; though experiments have shown that for $p=2$ it is often nearly achieved. One step of the conjugate gradient method using M_p^{-1}, however, may be significantly faster than p steps of the algorithm using M^{-1}. Hence, even if the maximum reduction in iterations is not obtained, there may be enough of a reduction to justify the increased time for an iteration using M_p^{-1}.

For A and M^{-1} real $n \times n$ symmetric positive-definite matrices, the generalized conjugate gradient algorithm may be stated as follows (see [2]):

Let $\underline{x}^{(0)}$ be given. For $k=0, 1, \ldots,$

$$\begin{aligned}
&\begin{cases} \underline{r}^{(0)} = \underline{b} - A\,\underline{x}^{(0)}, & k=0 \\ \underline{r}^{(k)} = \underline{r}^{(k-1)} - a_{k-1}\,\underline{q}^{(k-1)}, & k \geq 1, \end{cases}\\
&\underline{z}^{(k)} = M^{-1}\,\underline{r}^{(k)},\\
&b_0 = 0, \; k=0\\
&\hat{b}_k = \underline{z}^{(k)T}\,\underline{r}^{(k)}\\
&b_k = \hat{b}_k/\hat{b}_{k-1}, \; k \geq 1,\\
&\underline{p}^{(k)} = \underline{z}^{(k)} + b_k\,\underline{p}^{(k-1)},\\
&\underline{q}^{(k)} = A\,\underline{p}^{(k)},\\
&a_k = \hat{b}_k/\underline{p}^{(k)T}\,\underline{q}^{(k)},\\
&\underline{x}^{(k+1)} = \underline{x}^{(k)} + a_k\,\underline{p}^{(k)}.
\end{aligned}$$

A single iteration, after the first, requires (ignoring single scalar operations):

1	$M^{-1} \times$ vector
1	$A \times$ vector
3	scalar $\times$ vector
3	vector additions
2	vector inner products.

If M^{-1} is replaced by M_p^{-1}, an iteration requires:

p	$M^{-1}\times$ vector
p	$A\times$ vector
3	scalar $\times$ vector
$3+2(p-1)$	vector additions
2	vector inner products.

Thus, the time for an iteration with M_p^{-1} is greater than the time for an iteration with M^{-1} but less than the time for p iterations with M^{-1}. If A and/or M^{-1} are dense matrices, then most of the time will be spent on the operations involving A and M^{-1}, and an iteration with M_p^{-1} will take almost as much time as p iterations with M^{-1}. In this case, little can be gained and much can be lost by using $p>1$. If A and M^{-1} are sparse, however, an iteration with M_p^{-1} may be much faster than p iterations with M^{-1} because it avoids the additional vector sums and inner products. If the number of iterations is reduced by almost a factor of p, this can give significant savings.

To insure that the conjugate gradient algorithm with M_p^{-1} converges, we must first show that M_p^{-1} is symmetric and positive definite.

Theorem 3.1: *Let A be a real $n\times n$ symmetric positive definite matrix and let M^{-1} be any real $n\times n$ symmetric positive definite matrix such that $\rho(I-M^{-1}A)<1$. If M_p^{-1} is as defined in (2.3), for $p>1$, then M_p^{-1} is symmetric and positive definite.*

Proof:

$$M_p^{-1}=\left(\sum_{i=0}^{p-1}(I-M^{-1}A)^i\right)M^{-1}=\left(\sum_{i=0}^{p-1}\left(\sum_{j=0}^{i}(-1)^j\binom{i}{i-j}(M^{-1}A)^j\right)\right)M^{-1}=$$

$$=\left(\sum_{i=0}^{p-1}\alpha_i(M^{-1}A)^iM^{-1}\right)=M^{-1}\left(\sum_{i=0}^{p-1}\alpha_i(AM^{-1})^i\right),$$

for some coefficients α_i.

$$(M_p^{-1})^T=(M^{-1})^T\left(\sum_{i=0}^{p-1}\alpha_i(A^TM^{-1^T})^i\right)=M^{-1}\left(\sum_{i=0}^{p-1}\alpha_i(AM^{-1})^i\right)=M_p^{-1}.$$

Thus, M_p^{-1} is symmetric.

Now $I-M_p^{-1}A=(I-M^{-1}A)^p$. Since $\rho(I-M^{-1}A)<1$, we have $\rho(I-M_p^{-1}A)<1$, so the eigenvalues of $M_p^{-1}A$ are positive. Since A is symmetric and positive definite, we can write $A=LL^T$ for some nonsingular lower triangular matrix L. Then $M_p^{-1}A=M_p^{-1}LL^T$ is similar to $L^TM_p^{-1}L$ since $L^TM_p^{-1}L=L^T(M_p^{-1}A)L^{-T}$.

Since $L^TM_p^{-1}L$ is symmetric and has positive eigenvalues, we can write $L^TM_p^{-1}L=UU^T$, for some nonsingular upper triangular matrix U. Hence $M_p^{-1}=(L^{T^{-1}}U)(L^{T^{-1}}U)^T$, so M_p^{-1} is also positive definite.

It can be shown, as in [2], that the k-th iterate, $\underline{x}^{(k)}$, from the conjugate gradient algorithm can be written as

$$\underline{x}^{(k)} = \underline{x}^{(0)} + P_{k-1}(M^{-1}A)\,\underline{z}^{(0)},$$

where P_{k-1} is the $(k-1)$-st degree polynomial in $M^{-1}A$ which minimizes the A-norm of the error, $\| A^{-1}\underline{b} - \underline{x}^{(k)} \|_A = (A^{-1}\underline{b} - \underline{x}^{(k)})^T (\underline{b} - A\,\underline{x}^{(k)})^{1/2}$.

We use this fact to show that using M_p^{-1} instead of M^{-1} can reduce the number of iterations for the conjugate gradient method by no more than a factor of p.

Theorem 3.2: *Let A be a real $n \times n$ symmetric positive definite matrix and $\underline{b}$ a vector of length n, and let M^{-1} be any real $n \times n$ symmetric positive definite matrix such that $\rho(I - M^{-1}A) < 1$. Let M_p^{-1} be as defined in* (2.3), *for $p > 1$. Suppose $\underline{x}^{(0)} = \underline{x}_p^{(0)}$ is given and $\underline{x}_p^{(k)}$ is the k-th conjugate gradient iterate using M_p^{-1} and $\underline{x}^{(kp)}$ is the $(k\,p)$-th conjugate gradient iterate using M^{-1}. Then*

$$\| A^{-1}\underline{b} - \underline{x}_p^{(k)} \|_A \geq \| A^{-1}\underline{b} - \underline{x}^{(kp)} \|_A.$$

Proof: We have that

$$\underline{x}_p^{(k)} = \underline{x}_p^{(0)} + P_{k-1}(M_p^{-1}A)\,M_p^{-1}(\underline{b} - A\,\underline{x}_p^{(0)}),$$

where P_{k-1} is the $(k-1)$-st degree polynomial in $M_p^{-1}A$ which minimizes $\| A^{-1}\underline{b} - \underline{x}_p^{(k)} \|_A$.

$$M_p^{-1}A = \left(\sum_{i=0}^{p-1} (I - M^{-1}A)^i\right) M^{-1}A$$

is a p-th degree polynomial in $M^{-1}A$, so $P_{k-1}(M_p^{-1}A)$ is a $((k-1)\cdot p)$-th degree polynomial in $M^{-1}A$. Writing

$$M_p^{-1}(\underline{b} - A\,\underline{x}_p^{(0)}) = \left(\sum_{i=0}^{p-1} (I - M^{-1}A)^i\right) M^{-1}(\underline{b} - A\,\underline{x}_p^{(0)}),$$

we have

$$\underline{x}_p^{(k)} = \underline{x}_p^{(0)} + Q_{kp-1}(M^{-1}A)\,M^{-1}(\underline{b} - A\,\underline{x}_p^{(0)}),$$

where Q_{kp-1} is a $(k\,p-1)$-st degree polynomial in $M^{-1}A$. But

$$\underline{x}^{(kp)} = \underline{x}^{(0)} + P_{kp-1}(M^{-1}A)\,M^{-1}(\underline{b} - A\,\underline{x}^{(0)}),$$

where P_{kp-1} is the best $(k\,p-1)$-st degree polynomial in $M^{-1}A$ for minimizing $\| A^{-1}\underline{b} - \underline{x}^{(kp)} \|_A$. Therefore, we must have $\| A^{-1}\underline{b} - \underline{x}_p^{(k)} \|_A \geq \| A^{-1}\underline{b} - \underline{x}^{(kp)} \|_A$.

To get an estimate of how much, if any, the number of iterations for the conjugate gradient method is reduced by using M_p^{-1} instead of M^{-1}, we can look at the eigenvalues of the corresponding iteration matrices, $M_p^{-1}A$ and $M^{-1}A$. It is shown in [1] that the error in the k-th step of the conjugate gradient algorithm is bounded by:

$$\begin{aligned} \frac{\| \underline{x} - \underline{x}^{(k)} \|_A}{\| \underline{x} - \underline{x}^{(0)} \|_A} &\leq 2\left[\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^k - \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k\right]^{-1} \\ &\leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k, \end{aligned} \tag{3.1}$$

where κ is the condition number of the iteration matrix.

Hence, if the condition number of $M_p^{-1} A$ is smaller than that of $M^{-1} A$, we get a smaller upper bound for the error in each iterate of the conjugate gradient algorithm.

Since $\rho(I - M^{-1} A) < 1$, the eigenvalues, $\lambda(M^{-1} A)$, of $M^{-1} A$ lie between 0 and 2. Also,

$$\lambda(M_p^{-1} A) = \lambda[I - (I - M^{-1} A)^p] = 1 - (1 - \lambda(M^{-1} A))^p \tag{3.2}$$

so that $0 < \lambda(M_p^{-1} A) < 2$. Now, if $\alpha \le \lambda(M^{-1} A) \le 2 - \alpha$ where $\alpha = \min \lambda(M^{-1} A) \ll 1$, then

$$\begin{aligned} \min \lambda(M_p^{-1} A) &= 1 - (1-\alpha)^p \\ &= \sum_{i=1}^{p} (-1)^{i-1} \binom{p}{p-i} \alpha^i \\ &\simeq p\,\alpha. \end{aligned}$$

Let $\mu \in \lambda(M^{-1} A)$ be such that

$$|\mu - 1| = \min_{\lambda(M^{-1} A)} |\lambda - 1|$$

Then for p even,

$$\max \lambda(M_p^{-1} A) = 1 - (1-\mu)^p \simeq 1$$

Also, if $\gamma = \max \lambda(M^{-1} A) \gg 1$, then for p odd

$$\begin{aligned} \max \lambda(M_p^{-1} A) &= 1 - (1-\gamma)^p \\ &= 1 - (1 - (2-\delta))^p \\ &= 2 - \sum_{i=1}^{p} (-1)^{i-1} \binom{p}{p-i} \delta^i \\ &\simeq 2 - p\,\delta, \ \delta \text{ small}. \end{aligned}$$

Using the above analysis, we see that if $M^{-1} A$ has eigenvalues near the values 0, 1, and 2, then

$$\kappa(M_p^{-1} A) \simeq \frac{1}{2p} \cdot \kappa(M^{-1} A), \quad \text{for } p \text{ even}$$

$$\kappa(M_p^{-1} A) \simeq \frac{1}{p} \cdot \kappa(M^{-1} A), \quad \text{for } p \text{ odd}.$$

Using the error bound (3.1), we can obtain a bound on the number of iterations k necessary to reduce the initial error by a factor of ε^{-1}:

$$k \le \log\left(\tfrac{\varepsilon}{2}\right) / \log \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}.$$

As an example, let us take $\varepsilon = 10^{-6}$, and consider matrices $M^{-1} A$, whose largest eigenvalue is fixed, say 1.9, and whose smallest eigenvalue is 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}. Let us also assume that each matrix has an eigenvalue of 1. Using the expression (3.2), we can calculate the condition number of $M_p^{-1} A$,

$p=1, 2, 3, 4$. Table 1 gives bounds on the number of iterations (for the Chebyshev algorithm or the conjugate gradient algorithm) with M^{-1} ($p=1$) and with M_p^{-1} for $p=2, 3, 4$, necessary to reduce the error by a factor of 10^6.

For $p=2$, the number of iterations is reduced by approximately a factor of 2, though this factor is slightly smaller for large values of κ. For $p=3$, the larger value of $\lambda_{\max}$ outweighs the increase in $\lambda_{\min}$ to give a larger condition number than for $p=2$. Consequently, it takes more steps with $p=3$ to reduce the error bound by the same factor. This would indicate that the even values of p (odd maximum power, $p-1$, in the Neumann series), are generally better than the odd ones, as has been borne out in experiments. For $p=4$, the number of iterations is reduced by slightly less than a factor of 3, over the case where $p=1$.

These, of course, are only error bounds, not actual numbers of iterations required by the conjugate gradient algorithm. In general, they are pessimistic, with the algorithm actually taking considerably fewer iterations than the bounds would indicate. If the number of distinct eigenvalues of the iteration matrix is smaller than the number of steps indicated to reduce the error bound, for example, then clearly the bound is a poor one, since the exact solution is obtained in fewer steps. Still, the *ratios* of the number of iterations required with different values of p, seem approximately like those found in practice.

Table 1. *Bounds on the number of iterations necessary to reduce the error by a factor of* 10^6, *using* 1, 2, 3, *or* 4 *terms in the Neumann series*

$p=1$				$p=2$			
$\lambda_{\min}$	$\lambda_{\max}$	κ	k	$\lambda_{\min}$	$\lambda_{\max}$	κ	k
10^{-1}	1.9	1.9×10	32	1.9×10^{-1}	1	$.53\times 10$	16
10^{-2}	1.9	1.9×10^{2}	100	2×10^{-2}	1	$.5\times 10^{2}$	51
10^{-3}	1.9	1.9×10^{3}	317	2×10^{-3}	1	$.5\times 10^{3}$	163
10^{-4}	1.9	1.9×10^{4}	1000	2×10^{-4}	1	$.5\times 10^{4}$	513
10^{-5}	1.9	1.9×10^{5}	3163	2×10^{-5}	1	$.5\times 10^{5}$	1623
10^{-6}	1.9	1.9×10^{6}	10000	2×10^{-6}	1	$.5\times 10^{6}$	5130
$p=3$				$p=4$			
$\lambda_{\min}$	$\lambda_{\max}$	κ	k	$\lambda_{\min}$	$\lambda_{\max}$	κ	k
2.71×10^{-1}	1.729	$.64\times 10$	18	3.439×10^{-1}	1	$.29\times 10$	11
3×10^{-2}	1.729	$.576\times 10^{2}$	55	4×10^{-2}	1	$.25\times 10^{2}$	36
3×10^{-3}	1.729	$.576\times 10^{3}$	175	4×10^{-3}	1	$.25\times 10^{3}$	115
3×10^{-4}	1.729	$.576\times 10^{4}$	551	4×10^{-4}	1	$.25\times 10^{4}$	363
3×10^{-5}	1.729	$.576\times 10^{5}$	1742	4×10^{-5}	1	$.25\times 10^{5}$	1147
3×10^{-6}	1.729	$.576\times 10^{6}$	5506	4×10^{-6}	1	$.25\times 10^{6}$	3628

What, then, is the optimal value of p, and how does this approximation compare with others, such as incomplete Cholesky decomposition, in terms of number of iterations and total time required for the conjugate gradient algorithm? This question is highly machine and problem dependent. As indicated before, if almost all of the time for an iteration is spent in multiplying A and M^{-1} against a vector, then the higher powers of p offer little advantage. If the eigenvalues of $M^{-1}A$ satisfy the hypotheses used in making Table 1, then we can expect a reduction of almost a factor of 2 in the number of iterations for $p=2$ over $p=1$. If an iteration with $p=2$ takes significantly less than twice the time of an iteration with $p=1$, it may be advantageous to use $p=2$. Similarly, in comparing the total time using a Neumann series expansion with that using an incomplete Cholesky decomposition, depending on the machine and problem being solved, a backsolve in incomplete Cholesky decomposition may be much faster or much slower than the matrix multiplications in the Neumann series.

4. Numerical Results

The conjugate gradient method was used to solve the linear system generated by a 5-point discretization of the elliptic differential equation (cf. [7])

$$\frac{\partial}{\partial x}\alpha(x,y)\frac{\partial u}{\partial x}+\frac{\partial}{\partial y}\beta(x,y)\frac{\partial u}{\partial y}=f(x,y)$$

where $\alpha, \beta>0$ are piecewise continuous functions and u is defined on the unit square $[0,1]\times[0,1]$.

Test I: The Dirichlet problem, $\alpha=\beta=1$, was used as the first test problem. In this case, the linear system $Ax=b$ generated, is given by

$$A=\begin{bmatrix} D_1 & -I & & 0 \\ -I & D_2 & -I & \\ & \ddots & \ddots & -I \\ 0 & & -I & D_n \end{bmatrix},$$

$$D_i=\begin{bmatrix} 4 & -1 & & 0 \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ 0 & & -1 & 4 \end{bmatrix}, \quad I=m\times m \text{ identity matrix.}$$

This is the usual five point difference approximation to the Dirichlet problem on a square $m\times m$ grid for the elliptic differential equation

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=f.$$

A random initial guess was used in all tests, with a right hand side consisting of all ones. Table 2 shows the number of iterations necessary to reduce the A-norm of the error by a factor of 10^6. Table 3 shows the time required on the 7600, using STACKLIB wherever practical, and Table 4 shows the time required on the STAR.

Table 2. *Number of iterations required to reduce the error by a factor of* 10^6

		No. of Iterations				
		Neumann	with	$(M^{-1} = (\text{diagonal } (A))^{-1})$		Incomplete
m	$n=m^2$	$p=1$	$p=2$	$p=3$	$p=4$	Cholesky
10	100	28	14	16	10	11
20	400	53	27	30	20	17
30	900	76	40	44	28	24
40	1600	91	52	53	37	30
50	2500	120	65	70	46	37

Table 3. *Time required on the* 7600 *to reduce the error by factor of* 10^6

		7600 Time in Milliseconds				
		Neumann	with	$(M^{-1} = (\text{diagonal } (A))^{-1})$		Incomplete
m	$n=m^2$	$p=1$	$p=2$	$p=3$	$p=4$	Cholesky
10	100	21	16	25	21	22
20	400	89	74	113	96	119
30	900	230	202	306	251	360
40	1600	459	435	617	555	780
50	2500	914	820	1218	1036	1591

Table 4. *Time required on the STAR* 100 *to reduce the error by a factor of* 10^6

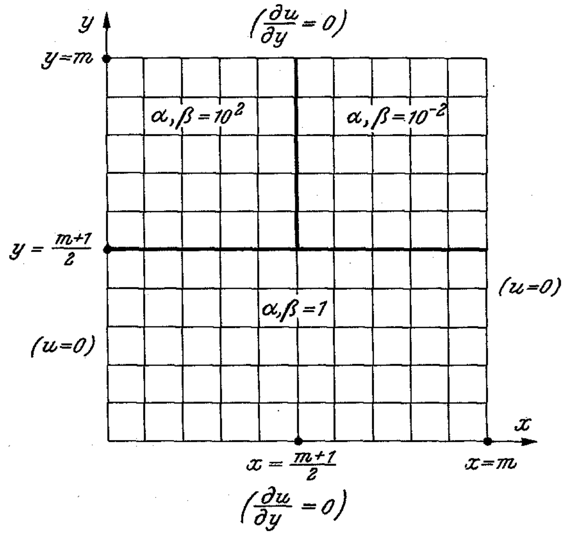
		STAR Time in Milliseconds				
		Neumann	with	$(M^{-1} = (\text{diagonal } (A))^{-1})$		Incomplete
m	$n=m^2$	$p=1$	$p=2$	$p=3$	$p=4$	Cholesky
10	100	13	9	13	10	42
20	400	49	34	47	37	170
30	900	128	92	127	98	384
40	1600	289	221	297	244	780
50	2500	584	445	606	493	1295

From Table 2, we see that the number of iterations is almost halved in going from $p=1$ to $p=2$, is slightly larger for $p=3$, and is reduced by almost a factor of 3 in going from $p=1$ to $p=4$; this is just what was predicted from the change in condition number. The incomplete Cholesky decomposition starts out taking approximately the same number of iterations as the 4-term Neumann series, but then takes fewer iterations as n increases.

In obtaining the timings for Table 3, the backsolving section of the iteration using incomplete Cholesky decomposition, was coded in scalar Fortran. For the timings on the STAR, in Table 4, it was advantageous to partially vectorize this backsolving process, as described in [3]. From these tables we see that using $p=2$ offers some advantage over $p=1$. In going to $p=4$, the extra time for an iteration begins to outweigh the reduction in iterations, and the total time increases. In all cases, the fully vectorized Neumann series is faster than the incomplete Cholesky decomposition, despite the fact that the incomplete Cholesky decomposition takes far fewer iterations.

From these figures we see, then, that on the 7600 and the STAR, for problems in which A has five diagonals, M^{-1} is taken to be (diagonal $(A))^{-1}$, and $M^{-1}A$ has some eigenvalues near 0, 1, and 2, the two-term Neumann series offers an advantage over the standard one-term approximation, M^{-1}. Using higher powers in the series is disadvantageous. As the number of diagonals in A and/or M^{-1} increases, the speedup obtained from using $p=2$, decreases. For this problem, the two-term Neumann series cannot complete with incomplete Cholesky decomposition in terms of number of iterations, but it is significantly faster in total time, especially on the STAR.

Test II: For the second test problem, the functions α and β were defined as follows:



y
y=m
$(\frac{\partial u}{\partial y}=0)$
α,β = 10²
α,β = 10⁻²
y = (m+1)/2
(u=0)
α,β = 1
(u=0)
x
x = (m+1)/2
x=m
$(\frac{\partial u}{\partial y}=0)$

Table 5

Algorithm		No. of Iterations	STAR-time	7600-time
Neumann $M^{-1}=\text{diag}(A)^{-1}$	$p=1$	133	611	987
	$p=2$	75	468	955
	$p=3$	77	605	1334
	$p=4$	53	504	1217
Incomplete Cholesky		43	1505	2158

The conjugate gradient method was used on the CDC-STAR and the CDC-7600 to solve the associated linear system. (In this case $m=50$ so that the order of the matrix is 2500.) The pertinent results are listed in Table 5 where again the number of iterations pertain to those required to reduce the A-norm of the error by a factor of 10^6 and the time is the amount of milliseconds expended for the execution of these iterations. A random initial guess was used with a right hand source vector consisting of all ones.

As in the first test problem, the number of iterations is almost halved in going from $p=1$ to $p=2$, is slightly larger for $p=3$, and is reduced by almost a factor of 3 in going from $p=1$ to $p=4$. The results using the incomplete Cholesky conjugate gradient method are included for purposes of comparison. It should be noted that the classical conjugate gradient method as proposed by Hestenes and Stiefel (i.e. $M=I$) was also tried but failed to converge after 2500 iterations (i.e. the A-norm of the error had only been reduced to 10^{-3}), indicating that at least diagonal scaling in such problems is essential.

5. Conclusions

The idea of using an approximation M^{-1} to A^{-1}, for iterative algorithms on vector processors, is a very attractive one, since it avoids the time-consuming scalar operation of backsolving. The two-term Neumann series combined with the conjugate gradient algorithm offers some improvement over the standard algorithm, for the five-point operator. What would be even better, however, would be an approximation M^{-1} to A^{-1} which has the same number of nonzero elements as the incomplete Cholesky decomposition, LL^T, and which would reduce the number of iterations required by the conjugate gradient algorithm by the same amount. It remains to be seen whether such an approximation exists and can be calculated in a reasonable amount of time.

Acknowledgements

The authors would like to thank Paul Saylor for many helpful discussions.

References

[1] Cline, A. K.: Several observations on the use of conjugate gradient Methods. ICASE, NASA Langley Research Center (Hampton, Va.).

[2] Concus, P., Golub, G. H., O'Leary, D. P.: A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations. Computer Science Dept., Stanford University, STAN-CS-76-533 (January, 1976).

[3] Greenbaum, A., Rodrigue, G. H.: The incomplete Cholesky conjugate gradient method for the Star (5-Point Operator). UCID-Lawrence Livermore Laboratory (Livermore, CA., July 1977).

[4] McMahon, F. H., Sloan, L. J., Long, G. A.: Stacklibe — a vector function library of optimum stack-loops for the CDC 7600. UCID-30083, Lawrence Livermore Laboratory (Livermore, CA., 1976).

[5] Meijerink, J. A., van der Vorst, H. A.: An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. Math. of Comp. *1977,* 148—162.

[6] Nelson, H.: TIMTBL-STAR instruction times. Lawrence Livermore Laboratory (Livermore, CA., November 1976).

[7] Varga, R. S.: Matrix iterative analysis. Prentice-Hall 1962.

[8] Young, D.: Iterative solution of large linear systems. Academic Press 1971.

A. Greenbaum, P. Dubois, G. H. Rodrigue
Lawrence Livermore Laboratory
University of California
P.O. Box 808
Livermore, CA 94550, U.S.A.