SIAM J. SCI. COMPUT.
Vol. 18, No. 3, pp. 838–853, May 1997

009

PARALLEL PRECONDITIONING WITH SPARSE APPROXIMATE INVERSES*

MARCUS J. GROTE† AND THOMAS HUCKLE‡

Abstract. A parallel preconditioner is presented for the solution of general sparse linear systems of equations. A sparse approximate inverse is computed explicitly and then applied as a preconditioner to an iterative method. The computation of the preconditioner is inherently parallel, and its application only requires a matrix-vector product. The sparsity pattern of the approximate inverse is not imposed a priori but captured automatically. This keeps the amount of work and the number of nonzero entries in the preconditioner to a minimum. Rigorous bounds on the clustering of the eigenvalues and the singular values are derived for the preconditioned system, and the proximity of the approximate to the true inverse is estimated. An extensive set of test problems from scientific and industrial applications provides convincing evidence of the effectiveness of this approach.

Key words. preconditioning, approximate inverses, parallel algorithms, sparse matrices, sparse linear systems, iterative methods

AMS subject classifications. 65F10, 65F35, 65F50, 65Y05

PII. S1064827594276552

1. Introduction.

We consider the linear system of equations

$$Ax = b, \quad x, b \in \mathbb{R}^n \ . \tag{1}$$

Here A is a large, sparse, and nonsymmetric matrix. Due to the size of A, direct solvers become prohibitively expensive because of the amount of work and storage required. As an alternative we consider iterative methods such as generalized minimal residual (GMRES), biconjugate gradient (BCG), biconjugate gradient stabilized (Bi-CGSTAB), and conjugate gradient (CG) applied to the normal equations [6]. Given the initial guess x_0, these algorithms compute iteratively new approximations x_k to the true solution $x = A^{-1}b$. The iterate x_m is accepted as a solution if the residual $r_m = b - Ax_m$ satisfies $\|r_m\|/\|b\| \leq \text{tol}$. In general, the convergence is not guaranteed or may be extremely slow. Hence, the original problem (1) must be transformed into a more tractable form. To do so, we consider a preconditioning matrix M and apply the iterative solver either to the right or to the left preconditioned system

$$AMy = b \ , \quad x = My \ , \quad \text{or} \quad MAx = Mb \ . \tag{2}$$

Therefore, M should be chosen such that AM (or MA) is a good approximation of the identity. As the ultimate goal is to reduce the total execution time, both the computation of M and the matrix-vector product My should be evaluated in parallel. Since the matrix-vector product must be performed at each iteration, the number of nonzero entries in M should not exceed that in A.

*Received by the editors May 6, 1994; accepted for publication (in revised form) September 14, 1995.

http://www.siam.org/journals/sisc/18-3/27655.html

†Scientific Computing and Computational Mathematics, Building 460, Stanford University, Stanford, California, 94305 (grote@cims.nyu.edu). The research of this author was supported by an IBM fellowship.

‡Institut für Angewandte Mathematik und Statistik, Universität Würzburg, D-97074 Würzburg, Germany (huckle@informatik.tu-muenchen.de). The research of this author was supported by a research grant of the Deutsche Forschungsgemeinschaft.

The most successful preconditioning methods in terms of reducing the number of iterations, such as incomplete LU factorizations or symmetric successive overrelaxation (SSOR), are notoriously difficult to implement on a parallel architecture, especially for unstructured matrices. Indeed, the application of the preconditioner in the iteration phase requires the solution of triangular systems at each step, which is difficult to parallelize because of the recursive nature of the computation (see [2, section 4.4.4]). Our aim is to find an inherently parallel preconditioner which retains the convergence properties of incomplete LU.

A natural way to achieve parallelism is to compute an approximate inverse M of A, such that $AM \simeq I$ in some sense. The evaluation of My is then easy to parallelize and will be cheap if M is sparse. Although such a sparse approximate inverse does not always exist, it often occurs in applications that most entries in A^{-1} are very small [5], [1]. For instance, if the problem results from the discretization of a partial differential equation, it is generally meaningful to look for a sparse approximate inverse. Polynomial preconditioners with $M = p(A)$ are inherently parallel but do not lead to as much improvement in the convergence as incomplete LU (see [2, section 3.5]).

A different approach is to minimize $\|AM - I\|$. Yet we cannot confine the whole spectrum of AM to the vicinity of 1, or else we would be minimizing the 1-norm or the 2-norm. This would be too expensive, since $M = 0$ is a better solution as long as $\|AM - I\| > 1$. To ensure fast convergence at a reasonable cost, we need to cluster most eigenvalues and singular values about 1 but give a few outliers additional freedom. We achieve this by minimizing $\|AM - I\|$ in the Frobenius norm. Moreover, this choice naturally leads to inherent parallelism because the columns m_k of M can be computed independently of one another. Indeed, since

$$\|AM - I\|_F^2 = \sum_{k=1}^{n} \|(AM - I)e_k\|_2^2 \ , \tag{3}$$

the solution of (3) separates into n independent least squares problems

$$\min_{m_k} \|Am_k - e_k\|_2 \ , \quad k = 1, \ldots, n \ , \tag{4}$$

where $e_k = (0, ..., 0, 1, 0, ..., 0)^T$. Thus, we can solve (4) in parallel and obtain an explicit approximate inverse M of A. If M is sparse, (4) reduces to n small least squares problems, which can be solved very quickly [9], [13]. The difficulty lies in determining a good sparsity structure of the approximate inverse, or else the solution of (4) will not yield an effective preconditioner. Therefore, we seek a method that captures the sparsity pattern of the main entries of A^{-1} automatically, yet at a reasonable cost. We start with a given sparsity pattern, such as diagonal, and augment M progressively until the 2-norm of the residual is small enough or a maximal sparsity has been reached. The key to the algorithm lies in the strategy used to determine the locations of the entries in M. Our selection criterion is simple and cheap to compute, yet it yields an effective preconditioner without generating excessive fill-in. The extensive set of difficult test problems we consider at the end shows that our algorithm produces a sparse and effective preconditioner.

The computation of approximate inverses, based on minimizing (4), has been proposed by several authors. Kolotilina and Yeremin [11]; Kolotilina, Nikishin, and Yeremin [12]; and Lifshitz, Nikishin, and Yeremin [13] compute a factorized sparse approximate inverse but only consider fixed sparsity patterns. Grote and Simon solve

(4) explicitly but only allow for a banded sparsity pattern in M [9], [10]. The approach of Cosgrove, Diaz, and Griewank [4] is similar to ours but differs in the criteria used for augmenting M. Chow and Saad [3] use an iterative method to compute an approximate solution of (4). Their method automatically generates new entries, to which they apply a dropping strategy to remove the excessive fill-in appearing in M.

In section 2 we introduce the SPAI algorithm, which computes a sparse approximate inverse of A. In section 3 we derive theoretical properties of the spectrum of the preconditioned system (2). In sections 4 and 5 we present a wide range of numerical experiments using test matrices from engineering and scientific applications.

2. Computation of the sparse approximate inverse. We shall first show how to compute a sparse approximate inverse M for a given sparsity structure. The matrix M is the solution of the minimization problem (4). Since the columns of M are independent of one another, we only need to present the algorithm for one of them, and we denote it by m_k. Now let $\mathcal{J}$ be the set of indices j such that $m_k(j) \neq 0$. We denote the reduced vector of unknowns $m_k(\mathcal{J})$ by $\hat{m}_k$. Next, let $\mathcal{I}$ be the set of indices i such that $A(i, \mathcal{J})$ is not identically zero. This enables us to eliminate all zero rows in the submatrix $A(., \mathcal{J})$. We denote the resulting submatrix $A(\mathcal{I}, \mathcal{J})$ by $\hat{A}$. Similarly, we define $\hat{e}_k = e_k(\mathcal{I})$. If we now set $n_1 = |\mathcal{I}|$ and $n_2 = |\mathcal{J}|$, we see that solving (4) for m_k is equivalent to solving

$$\min_{\hat{m}_k} \|\hat{A}\hat{m}_k - \hat{e}_k\|_2 \tag{5}$$

for $\hat{m}_k$. The $n_1 \times n_2$ least squares problem (5) is extremely small because A and M are very sparse matrices. If A is nonsingular, the submatrix $\hat{A}$ must have full rank. Thus, the QR decomposition of $\hat{A}$ is

$$\hat{A} = Q \begin{pmatrix} R \\ 0 \end{pmatrix}, \tag{6}$$

where R is a nonsingular upper triangular $n_2 \times n_2$ matrix. If we let $\hat{c} = Q^T \hat{e}_k$, the solution of (5) is

$$\hat{m}_k = R^{-1}\, \hat{c}\,(1 : n_2). \tag{7}$$

We solve (7) for each $k = 1, \ldots, n$ and set $m_k(\mathcal{J}) = \hat{m}_k$. This yields an approximate inverse M, which minimizes $\|AM - I\|_F$ for the given sparsity structure.

Our aim is now to improve upon M by augmenting its sparsity structure to obtain a more effective preconditioner. To do so, we shall reduce the current error $\|AM - I\|_F$, that is reduce $\|Am_k - e_k\|_2$ for each $k = 1, \ldots, n$. We recall that m_k is the optimal solution of the least squares problem (4), and we denote its residual by

$$r = A(., \mathcal{J})\, \hat{m}_k - e_k. \tag{8}$$

If $r = 0$, m_k is exactly the kth column of A^{-1} and cannot be improved upon. We now assume that $r \neq 0$ and demonstrate how to augment the set of indices $\mathcal{J}$ to reduce $\|r\|_2$. Since A and m_k are sparse, most components of r are zero, and we denote by $\mathcal{L}$ the remaining set of indices ℓ for which $r(\ell) \neq 0$. Typically $\mathcal{L}$ is equal to $\mathcal{I}$, since $\hat{r}$ does not have exact zero entries in finite precision. But if $\mathcal{I}$ does not contain k, it must be included in $\mathcal{L}$ since $r(k)$ is then equal to -1. To every $\ell \in \mathcal{L}$ corresponds an index set $\mathcal{N}_\ell$, which consists of the indices of the nonzero elements of $A(\ell, .)$ that are

not in $\mathcal{J}$ yet. The potential new candidates that might be added to $\mathcal{J}$ are contained in

$$\tilde{\mathcal{J}} = \bigcup_{\ell \in \mathcal{L}} \mathcal{N}_\ell. \tag{9}$$

We must now select new indices j that will lead to the most profitable reduction in $\|r\|_2$. To do so in a cheap but effective way, we consider for each $j \in \tilde{\mathcal{J}}$ the one-dimensional minimization problem

$$\min_{\mu_j} \|r + \mu_j A e_j\|_2. \tag{10}$$

The solution of (10) is

$$\mu_j = -\frac{r^T A e_j}{\|A e_j\|_2^2}. \tag{11}$$

For each j we compute the 2-norm ρ_j of the new residual $r + \mu_j A e_j$ with μ_j given by (11):

$$\rho_j^2 = \|r\|_2^2 - \frac{(r^T A e_j)^2}{\|A e_j\|_2^2}. \tag{12}$$

There is at least one index $j \in \tilde{\mathcal{J}}$ such that $r^T A e_j \neq 0$, which will lead to a smaller residual in (12). Otherwise

$$0 = r(\mathcal{L})^T A(\mathcal{L}, \tilde{\mathcal{J}}) = r(\mathcal{L})^T A(\mathcal{L}, \mathcal{J} \cup \tilde{\mathcal{J}}) \ , \tag{13}$$

which would imply that $r(\mathcal{L})$ is zero, since $A(\mathcal{L}, .)$ has full rank. We note that $\mathcal{J} \cup \tilde{\mathcal{J}}$ contains the column indices of all nonzero elements of $A(\mathcal{L}, .)$ and that $\mathcal{J} \cap \tilde{\mathcal{J}} = \emptyset$. We reduce $\tilde{\mathcal{J}}$ to the set of the most profitable indices j with smallest ρ_j and add it to $\mathcal{J}$. We note that equation (9) was also used in [4] to estimate the reduction in the residual and can already be found in [8].

Using the augmented set of indices $\mathcal{J}$, we solve the sparse least squares problem (4) again. This yields a better approximation m_k of the kth column of A^{-1}. We repeat this process for each $k = 1, \ldots, n$ until the residual satisfies a prescribed tolerance or a maximum amount of fill-in has been reached in m_k. The numerical study in section 4 shows that this iterative procedure captures the main entries of A^{-1} extremely well.

Every time we augment the set of nonzero entries in m_k, we solve the least squares problem exactly. We shall now demonstrate how one can easily update the QR decomposition and greatly reduce the amount of work. We recall that $\mathcal{J}$ is the current set and that $\tilde{\mathcal{J}}$ is the set of new indices that will be added to m_k. We denote by $\tilde{\mathcal{I}}$ the new rows, which correspond to the nonzero rows of $A(., \mathcal{J} \cup \tilde{\mathcal{J}})$ not contained in $\mathcal{I}$ yet, and by $\tilde{n}_1$ and $\tilde{n}_2$ the number of indices in $\tilde{\mathcal{I}}$ and $\tilde{\mathcal{J}}$. Thus, we need to replace in (5) the submatrix $\hat{A}$ by the larger submatrix $A(\mathcal{I} \cup \tilde{\mathcal{I}}, \mathcal{J} \cup \tilde{\mathcal{J}})$. To solve (5), we use the known QR decomposition of $\hat{A}$ to update the QR decomposition of the augmented matrix $A(\mathcal{I} \cup \tilde{\mathcal{I}}, \mathcal{J} \cup \tilde{\mathcal{J}})$:

$$A(\mathcal{I} \cup \tilde{\mathcal{I}}, \mathcal{J} \cup \tilde{\mathcal{J}}) = \begin{pmatrix} \hat{A} & A(\mathcal{I}, \tilde{\mathcal{J}}) \\ 0 & A(\tilde{\mathcal{I}}, \tilde{\mathcal{J}}) \end{pmatrix} \tag{14}$$

$$= \begin{pmatrix} Q & \\ & I_{\tilde{n}_1} \end{pmatrix} \begin{pmatrix} R & Q_1^T A(\mathcal{I}, \tilde{\mathcal{J}}) \\ 0 & Q_2^T A(\mathcal{I}, \tilde{\mathcal{J}}) \\ 0 & A(\tilde{\mathcal{I}}, \tilde{\mathcal{J}}) \end{pmatrix} = \begin{pmatrix} Q & \\ & I_{\tilde{n}_1} \end{pmatrix} \begin{pmatrix} R & B_1 \\ 0 & B_2 \end{pmatrix}. \tag{15}$$

This requires only the computation of the QR decomposition of B_2. We note that $A(\tilde{\mathcal{I}}, \mathcal{J}) = 0$, because $\mathcal{I}$ already contains the indices of all nonzero entries present in columns $\mathcal{J}$. We let

$$B_2 = \tilde{Q} \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}, \tag{16}$$

where B_2 is a $\tilde{n}_1 + n_1 - n_2 \times \tilde{n}_2$ matrix. Using (16) we rewrite (15) as

$$A(\mathcal{I} \cup \tilde{\mathcal{I}}, \mathcal{J} \cup \tilde{\mathcal{J}}) = \begin{pmatrix} Q & \\ & I_{\tilde{n}_1} \end{pmatrix} \begin{pmatrix} I_{n_2} & \\ & \tilde{Q} \end{pmatrix} \begin{pmatrix} R & B_1 \\ 0 & \tilde{R} \\ 0 & 0 \end{pmatrix}. \tag{17}$$

This procedure enables us to add new indices to $\mathcal{J}$ and solve the least squares problem for the optimal solution without recomputing the full QR decomposition at each step. It generalizes the updating strategy proposed in [4] for a single entry by allowing for several new entries at a time.

If we do not stop the process, the algorithm will compute the kth column of A^{-1}. In practice, however, we stop the process once the prescribed tolerance is met or a maximal amount of fill-in has been reached. We now present the full SPAI algorithm, where SPAI stands for SParse Approximate Inverse.

THE SPAI ALGORITHM.

For every column m_k of M:

- **(a)** Choose an initial sparsity $\mathcal{J}$.
- **(b)** Compute the row indices $\mathcal{I}$ of the corresponding nonzero entries and the QR decomposition (6) of $A(\mathcal{I}, \mathcal{J})$. Then compute the solution m_k of the least squares problem (4) and its residual r given by (8).
  While $\|r\|_2 > \varepsilon$:
- **(c)** Set $\mathcal{L}$ equal to the set of indices ℓ for which $r(\ell) \neq 0$.
- **(d)** Set $\tilde{\mathcal{J}}$ equal to the set of all new column indices of A that appear in all $\mathcal{L}$ rows but not in $\mathcal{J}$.
- **(e)** For each $j \in \tilde{\mathcal{J}}$ solve the minimization problem (10).
- **(f)** For each $j \in \tilde{\mathcal{J}}$ compute ρ_j given by (12) and delete from $\tilde{\mathcal{J}}$ all but the most profitable indices.
- **(g)** Determine the new indices $\tilde{\mathcal{I}}$ and update the QR decomposition using (17). Then solve the new least squares problem, compute the new residual $r = Am_k - e_k$, and set $\mathcal{I} = \mathcal{I} \cup \tilde{\mathcal{I}}$ and $\mathcal{J} = \mathcal{J} \cup \tilde{\mathcal{J}}$.

Remarks.

1. The initial sparsity structure of M is arbitrary and may be chosen empty or diagonal if no a priori information about the sparsity of A^{-1} is available. Yet to solve a sequence of problems with similar sparsity patterns but varying entries in A, a clever initial guess for the initial sparsity is to choose the sparsity of the previously computed approximate inverse. This greatly reduces the computational cost of M, since the initial sparsity structure would be almost optimal. In all our numerical examples the initial sparsity of M was chosen to be diagonal.

2. In addition to the stopping criterion on $\|r\|_2$, we constrain the loop to a maximal number of iterations to limit the maximal fill-in per column in M. This threshold was almost never reached and the total number of nonzero entries in M is usually comparable with the amount in A.

3. In (f) we first reduce $\tilde{\mathcal{J}}$ to the set of indices j such that ρ_j is less than or equal to the mean value of all ρ_j. From the remaining indices we keep at most s indices

with smallest ρ_j. Here s should be a small integer to avoid excessive fill-in, and we have set s equal to 5 in most numerical calculations. This criterion is very cheap to compute and removes useless indices effectively. It is parameter free since it uses a dynamic mean value criterion and does not require a threshold input by the user. A more sophisticated weighting could be applied to the distribution of the ρ_j to control the rate at which M is being filled.

4. When we update the QR decomposition (17), we store the Householder matrices resulting from the factorizations of the matrices B_2 separately and never construct Q explicitly.

5. The selection process in (c) may be restricted to the most easily accessible rows to minimize communications and data flow. It can also be restricted only to the largest elements in r.

6. The one-dimensional minimization can be replaced by another minimization method such as steepest descent or the exact minimization problem related to $\mathcal{J} \cup \{j\}$. From our experience, the former is not accurate enough, and the latter is too expensive.

7. The approximate inverse M computed with the SPAI algorithm is permutation invariant. If A is replaced by $P_1 A P_2$, where P_1 and P_2 are permutation matrices, we obtain $P_2^T M P_1^T$ instead of M.

The SPAI algorithm may also be applied to compute a sparse approximate left inverse of A, which can be used as a left preconditioner in (2). This may yield a better result if all the rows of A^{-1} are sparse but a few columns in A^{-1} are full. Such a case is discussed in section 5.

If the iterative solver preconditioned with the current M does not converge, it is easy to improve upon M using the sparsity of M as initial sparsity for the SPAI algorithm. The iteration will then proceed with the new preconditioner M. Moreover, since we compute the residual for each individual column m_k of M, it is easy to single out the most difficult columns and concentrate on them to improve the convergence of the iterative solver. This may prove useful in connection with flexible preconditioning, where the preconditioner is adapted during the iterative process (see [17], [14], [3]).

3. Theoretical properties of M. We shall now derive rigorous bounds on the spectrum of the preconditioned matrix AM. Furthermore, we shall estimate the difference between M and A^{-1} and derive conditions that guarantee that M is nonsingular. Let M be an approximate inverse of A obtained from the SPAI algorithm, and let m_k be its kth column. We denote by r_k the residual for every m_k and assume that it satisfies

$$\|r_k\| = \|Am_k - e_k\| < \varepsilon. \tag{18}$$

THEOREM 3.1. *Let* $p = \max_{1 \le k \le n}\{$*number of nonzero elements of* $r_k\}$. *Then*

$$\|AM - I\|_F \le \sqrt{n}\,\varepsilon\ , \quad \|M - A^{-1}\|_F \le \|A^{-1}\|_2 \sqrt{n}\,\varepsilon\ , \tag{19}$$

$$\|AM - I\|_2 \le \sqrt{n}\,\varepsilon\ , \quad \|M - A^{-1}\|_2 \le \|A^{-1}\|_2 \sqrt{n}\,\varepsilon\ , \tag{20}$$

$$\|AM - I\|_1 \le \sqrt{p}\,\varepsilon\ , \quad \|M - A^{-1}\|_1 \le \|A^{-1}\|_1 \sqrt{p}\,\varepsilon. \tag{21}$$

We note that p is usually much smaller than n because A and M are sparse. Thus, some of the bounds derived in this section are tighter than those discussed in [4] and in [1, Chapter 8]). They directly apply to the computed approximate inverse and the preconditioned system (2) because the basic assumption (18) coincides with the stopping criterion used in the SPAI algorithm.

Proof. Since

$$\|AM - I\|_F^2 = \sum_{k=1}^{n} \|(AM - I)\, e_k\|_2^2 \le \sum_{k=1}^{n} \varepsilon^2 = n\,\varepsilon^2 \ , \tag{22}$$

we immediately obtain the left inequality in (19). To derive the left inequality in (20), we use the definition of the 2-norm

$$\begin{aligned} \|AM - I\|_2 &= \max_{\|x\|_2=1} \|(AM - I)\, x\|_2 \\ &= \max_{\|x\|_2=1} \|\sum_{k=1}^{n} x_k\, (AM - I)\, e_k\|_2 \le \max_{\|x\|_2=1} \|x\|_1\, \varepsilon \le \sqrt{n}\,\varepsilon. \end{aligned} \tag{23}$$

We get the right inequalities in (19) and (20) using (23) and

$$\|M - A^{-1}\| \le \|A^{-1}\|_2\, \|AM - I\| \le \|A^{-1}\|_2 \sqrt{n}\,\varepsilon \ , \tag{24}$$

which holds in both the 2-norm and the Frobenius norm. As $\|r_k\|_1$ is bounded by $\sqrt{p}\,\|r_k\|_2$, we immediately get (21). ∎

If we apply Gershgorin's theorem to AM, we see that all eigenvalues lie inside a disk of radius $\sqrt{p}\,\varepsilon$ centered at 1. Therefore, M is nonsingular if $\sqrt{p}\,\varepsilon < 1$. We summarize this result as a corollary.

COROLLARY 3.1. *If $\sqrt{p}\,\varepsilon < 1$, then M is nonsingular.*

As we generally do not know p in advance, (21) is only useful after having computed M. Since $p \le n$, we see that if $\sqrt{n}\,\varepsilon < 1$, M must be nonsingular. This gives a criterion for choosing ε for a given n, although in practice it is often too costly to run the algorithm with such a small ε.

Even if A is symmetric, M will be nonsymmetric in general. It may then be appropriate to use the symmetrized preconditioner $(M + M^T)/2$ instead of M. To derive some estimates we begin with the following inequality, which holds in the 2-norm and the Frobenius norm if A is symmetric:

$$\|M - M^T\| = \|M - A^{-1} + A^{-T} - M^T\| \le 2\,\|M - A^{-1}\| \le 2\sqrt{n}\,\varepsilon\,\|A^{-1}\|_2. \tag{25}$$

Therefore, the symmetrized preconditioner $(M + M^T)/2$ satisfies

$$\begin{aligned} \|A(M + M^T)/2 - I\| &\le \|AM - I\| + \|A\|_2\, \|M - M^T\|/2 \\ &\le (1 + \mathrm{cond}_2(A))\sqrt{n}\,\varepsilon. \end{aligned} \tag{26}$$

In general, equation (26) is a very pessimistic estimate and not of much practical use. This is because the SPAI algorithm does not take advantage of the symmetry of A and does not yield a symmetric approximate inverse. It is easy, however, to reformulate the algorithm to compute only the lower triangular part of M. This yields a symmetric preconditioner, but the algorithm then loses its inherent parallelism. An interesting alternative would be to compute a factorized sparse approximate inverse, as in [11], but to leave the sparsity open as in the SPAI algorithm.

The convergence of most iterative methods heavily depends on the distribution of the eigenvalues or the singular values of the preconditioned matrix [6]. Indeed, if most eigenvalues are clustered about 1 and only a few outliers are present, the convergence will generally be very fast. Thus, it is crucial to derive estimates on the spectrum of

AM to determine the theoretical effectiveness of the preconditioner. Such estimates are summarized in the following two theorems.

THEOREM 3.2. *Let* $p = \max_{1\le k\le n}\{$*number of nonzero elements of* $r_k\}$. *Then, the eigenvalues* λ_k *of* AM *are clustered at* 1 *and lie inside a circle of radius* $\sqrt{p}\,\varepsilon$. *Furthermore, if* $\sqrt{p}\,\varepsilon < 1$, *then* λ_{max} *and* λ_{min} *satisfy*

$$\left|\frac{\lambda_{max}}{\lambda_{min}}\right| \le \frac{1+\sqrt{p}\,\varepsilon}{1-\sqrt{p}\,\varepsilon}. \tag{27}$$

Proof. Let QRQ^T be a Schur decomposition of $AM - I$. Then

$$\sum_{k=1}^{n} |1-\lambda_k|^2 = \|\mathrm{diag}(R)\|_2^2 \le \|R\|_F^2 = \|AM-I\|_F^2 \le n\,\varepsilon^2. \tag{28}$$

Since

$$\frac{1}{n}\sum_{k=1}^{n} |1-\lambda_k|^2 \;\le\; \varepsilon^2\,, \tag{29}$$

the eigenvalues λ_k of AM are clustered at 1. Next, we use Gershgorin's theorem and (21) to conclude that all λ_k lie inside a disk of radius $\sqrt{p}\,\varepsilon$ centered at 1. ∎

As an immediate consequence of (28), we see that minimizing $\|AM - I\|_F$ also reduces A's departure from normality $\|N\|_F^2$, where $Q^T AMQ = \mathrm{diag}(\lambda_i) + N$ is a Schur decomposition of AM.

We now derive bounds for the singular values and the condition number of AM.

THEOREM 3.3. *The singular values of* AM *are clustered at* 1 *and lie inside the interval* $[1-\delta, 1+\delta]$, *with* $\delta = \sqrt{n}\,\varepsilon\,(2+\sqrt{n}\,\varepsilon)$. *Furthermore, if* $\delta < 1$, *then the condition number of* AM *satisfies*

$$\mathrm{cond}_2(AM) \le \sqrt{\frac{1+\delta}{1-\delta}}. \tag{30}$$

Proof. Since the singular values of AM are the square roots of the eigenvalues of AMM^TA^T, we consider

$$\|I - AMM^TA^T\|_2 = \|I + (I-AM)\,M^TA^T - M^TA^T\|_2. \tag{31}$$

We then bound (31) from above by

$$\sqrt{n}\,\varepsilon\,(1+\|AM\|_2) \le \sqrt{n}\,\varepsilon\,(1+\|AM-I+I\|_2) \le \sqrt{n}\,\varepsilon\,(2+\sqrt{n}\,\varepsilon). \tag{32}$$

Next, we apply (28) to AMM^TA^T instead of AM and conclude that the singular values must be clustered at 1. ∎

Essential properties for the convergence of iterative methods are the clustering of eigenvalues and singular values, the condition number, and the departure from normality of the preconditioned linear system. In this section we have shown that minimizing $\|AM - I\|$ in the Frobenius norm produces a preconditioner M, which improves on all four points.

4. A first numerical example. We begin with a detailed study of orsirr2, the smallest among the orsx oil reservoir simulation problems in the Harwell–Boeing matrix collection. Convergence results for orsreg1 and orsirr1 are presented at the end of this section. Here is a brief description of the problems:

TABLE 1
Convergence results for orsirr2: unpreconditioned ($M = I$), and preconditioned ($0.2 \le \varepsilon \le 0.6$).

	BCG	CGS	Bi-CGSTAB	GMRES(20)	CGNE
$M = I$	848	653	> 1000	> 1000	> 1000
$\varepsilon = 0.6$	276	200	223	458	> 1000
$\varepsilon = 0.5$	112	70	68	146	600
$\varepsilon = 0.4$	72	41	47	84	296
$\varepsilon = 0.3$	51	32	32	52	168
$\varepsilon = 0.2$	30	18	18	29	68

TABLE 2
Orsirr2: $\|AM - I\|$ and the condition number of AM for different values of ε.

	$\|AM - I\|_F$	$\|AM - I\|_2$	$\|AM - I\|_1$	$\mathrm{cond}_2(AM)$	$\frac{nz(M)}{nz(A)}$
$M = I$	1.6×10^6	4.6×10^5	5.7×10^5	6.3×10^4	0.15
$\varepsilon = 0.6$	14.3	1.02	1.22	1991	0.32
$\varepsilon = 0.5$	11.3	1.08	1.56	211	0.61
$\varepsilon = 0.4$	8.98	1.05	1.56	74.2	0.89
$\varepsilon = 0.3$	7.12	1.00	1.78	35.1	1.53
$\varepsilon = 0.2$	4.82	0.98	1.74	13.5	3.39

orsreg1: an oil reservoir simulation matrix for a $21 \times 21 \times 5$ full grid of size $n = 2205$ and with $nz = 14133$ nonzero entries.

orsirr1: an oil reservoir simulation matrix for an $21 \times 21 \times 5$ irregular grid, $n = 1030$ and $nz = 6858$.

orsirr2: an oil reservoir simulation matrix for a $21 \times 21 \times 5$ irregular grid, $n = 886$ and $nz = 5970$.

Because of the rather small size of orsirr2, we can compute its true inverse and all the eigenvalues and singular values of the preconditioned system. All the numerical experiments presented in this section were computed in double precision using a MATLAB implementation. The initial guess was always $x_0 = 0$, $\tilde{r}_0 = r_0 = b$, and the stopping criterion

$$\frac{\|b - Ax_m\|_2}{\|b\|_2} < 10^{-8}\ , \quad x_m = My_m. \tag{33}$$

In Table 1 we present the convergence results for a variety of iterative methods and decreasing values of ε. The algorithms for the various iterative methods were obtained from [2]. The convergence results without preconditioning are listed under $M = I$. We denote by CGNE the CG algorithm applied to the preconditioned normal equations $AMM^TA^Ty = b$ with $x = MM^TA^Ty$. As we reduce ε and $\|AM - I\|_F$, the number of iterations drops significantly for all iterative methods. The results in Table 1 exemplify the robustness of the preconditioner with respect to ε.

In Table 2 we compare different norms of $AM - I$ as a function of ε. As we reduce ε, both the 1-norm and the 2-norm remain nearly constant. Thus, $M = 0$ is a better solution of the minimization problem $\|AM - I\|_1$ than is the computed M for all $\varepsilon \ge 0.2$. This is also true for the 2-norm as long as $\varepsilon \ge 0.3$ and indicates that neither the 1-norm nor the 2-norm is a good measure of the proximity of AM to the identity if the Frobenius norm is being minimized. As we reduce ε the singular values

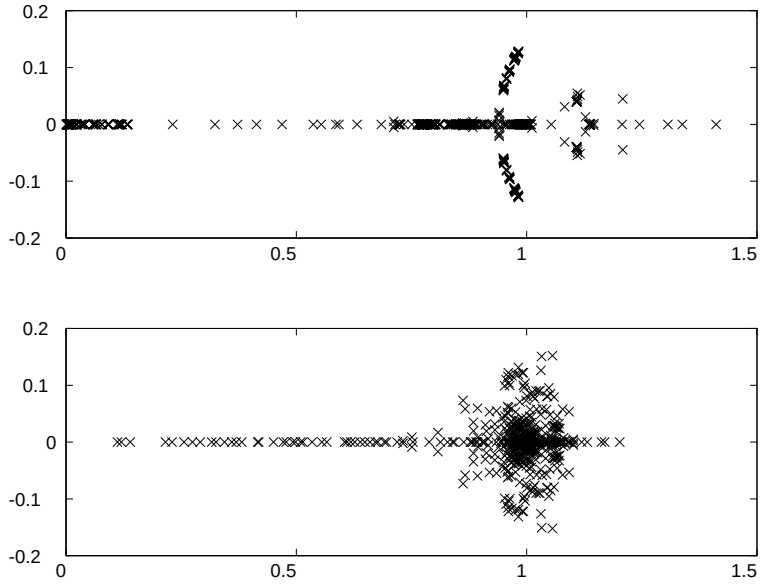

FIG. 1. *Orsirr2: eigenvalues of AM with* $\varepsilon = 0.6$ (*top*) *and with* $\varepsilon = 0.2$ (*bottom*).

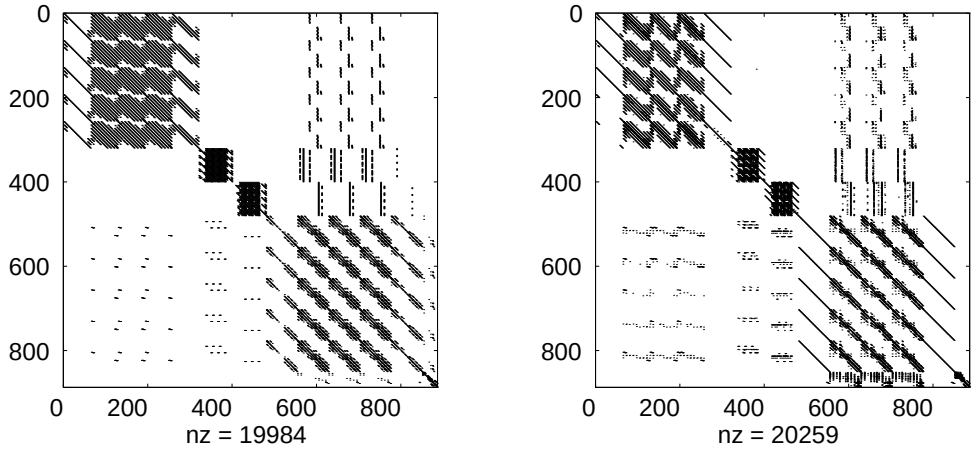

FIG. 2. *Orsirr2: sparsity pattern of entries in* A^{-1} *larger than* 0.001 *in absolute value* (*left*) *and sparse approximate inverse* M *with* $\varepsilon = 0.2$ (*right*).

cluster at 1, as demonstrated in Theorem 3.3. This is clearly shown in Table 2, since the condition number of AM is the ratio of the largest to the smallest singular value.

In Figure 1 we verify the improvement in the clustering of the eigenvalues predicted by Theorem 3.2 as we reduce ε.

Next, we compare the approximate inverse M with the true inverse A^{-1}. We compute A^{-1} and discard all entries whose absolute value is less than or equal to 0.001. Then, we compute the approximate inverse M with $\varepsilon = 0.2$. It is quite striking how well the sparsity patterns of both matrices agree with each other qualitatively in Figure 2.

TABLE 3
Convergence results for orsreg1 and orsirr1: unpreconditioned ($M = I$) and preconditioned ($\varepsilon = 0.4$).

	BCG	CGS	Bi-CGSTAB	GMRES(20)	GMRES(50)
orsreg1, $M = I$	430	254	689	753	563
$\varepsilon = 0.4$	86	48	55	82	77
orsirr1, $M = I$	> 1000	948	> 1000	> 1000	> 1000
$\varepsilon = 0.4$	71	41	45	81	67

TABLE 4
Convergence results for shermanx: unpreconditioned (top) and preconditioned (bottom).

	sh1		sh2		sh3		sh4		sh5	
Bi-CGSTAB	405		> 1000		> 1000		97		> 1000	
GMRES(20)	> 1000		> 1000		> 1000		792		> 1000	
Bi-CGSTAB	41		4 ($< 10^{-5}$)		72		28		41	
GMRES(20)	89		7 ($< 10^{-5}$)		264		86		173	
$nz(M)/nz(A)$	1.34		1.14		2.42		2.45		1.34	
ε | max. $nz(m_k)$	0.4	100	0.4	50	0.2	100	0.2	50	0.2	50

We conclude this section with the convergence results for both orsreg1 and orsirr1, given in Table 3. The relative sparsity $nz(M)/nz(A)$ was 0.94 for orsreg1 and 0.88 for orsirr1.

5. Numerical experiments. In this section we consider a wide spectrum of problems coming from scientific and industrial applications. We shall demonstrate the effectiveness of the preconditioner for two standard but very different iterative methods: Bi-CGSTAB [18] and GMRES(m) with restart. We recall that the former requires two matrix-vector multiplications per iteration, whereas the latter requires only one matrix-vector multiplication per iteration. In all numerical calculations the initial guess was $x_0 = 0$, $\tilde{r}_0 = r_0 = b$, and unless specified the stopping criterion was as in (33). All the computations were done in double precision FORTRAN, partly on a Sparc10 Sun station and partly on an Iris 4D/35 SGI station. The FORTRAN code provided in [2] was used for GMRES.

We begin with the convergence results for the five shermanx black oil simulators. This set consists of

sherman1: a black oil simulator, shale barrier, $10 \times 10 \times 10$ grid, 1 unknown, of size $n = 1000$, and with $nz = 3750$ nonzero elements.

sherman2: a thermal simulation, steam injection, $6 \times 6 \times 5$ grid, 5 unknowns, $n = 1080$, and $nz = 23094$.

sherman3: a black oil, impes simulation, $35 \times 11 \times 13$ grid, 1 unknown, $n = 5005$, and $nz = 20033$.

sherman4: a black oil, impes simulation, $16 \times 23 \times 3$ grid, 1 unknown, $n = 1104$, and $nz = 3786$.

sherman5: a fully implicit black oil simulator, $16 \times 23 \times 3$ grid, 3 unknowns, $n = 3312$, and $nz = 20793$.

The right-hand side was always provided. Table 4 shows that preconditioning clearly improves the convergence for all considered problems. Here "max. $nz(m_k)$" denotes the upper limit on the number of nonzero elements per column in M. For sherman2, both Bi-CGSTAB and GMRES(20) reduced the relative residual below 10^{-5} after

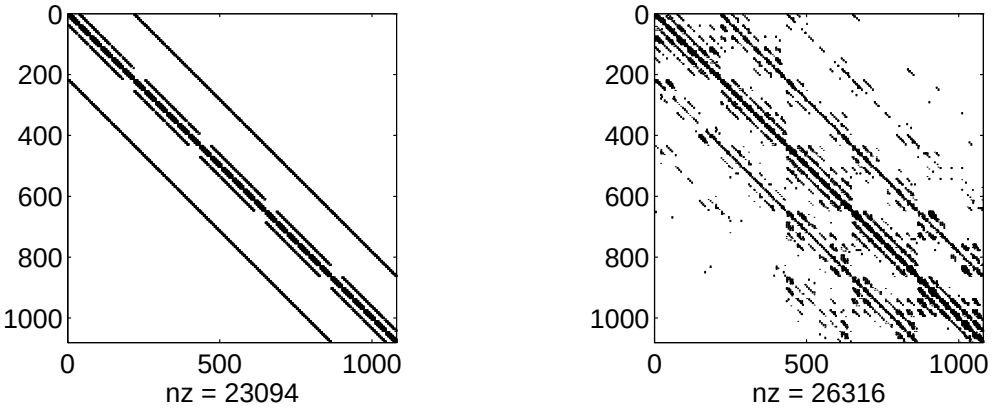

FIG. 3. *Sherman2: original matrix A (left) and sparse approximate inverse M with* $\varepsilon = 0.4$ *(right).*

four and seven steps, respectively, but never reached 10^{-8}. This may be due to the very large condition number 9.64×10^{11} of sherman2 and could not be improved upon. A different implementation of GMRES [16] might further reduce the residual, but we did not pursue this matter.

In Figure 3 we have displayed both the original matrix A and the approximate inverse M for sherman2. This picture clearly shows why we cannot simply set M equal to a banded matrix like in [10]. Indeed the sparsity structures of A and M are totally different in this particular case, whereas the number of nonzero entries in both matrices are comparable.

The next set of examples consists of the larger problems in the poresx and saylorx collections:

pores2: a nonsymmetric matrix, $n = 1224$, and $nz = 9613$.

pores3: a nonsymmetric matrix, $n = 532$, and $nz = 3474$.

saylor3: a nonsymmetric problem, $n = 1000$, and $nz = 3750$.

saylor4: a nonsymmetric problem, $n = 3564$, and $nz = 22316$.

The numerical results are shown in Table 5; here the right-hand side was randomly chosen. Since pores2 did not generate a very sparse approximate inverse and the iteration never reached the relative tolerance of 10^{-8}, we opted for left instead of right preconditioning. Thus in the case of pores2, we considered $\|MA - I\|_F$ instead of (4). Yet we still computed the exact residual of the original problem to check convergence. This approach proved to be more efficient, because the rows of A^{-1} could be approximated more effectively than its columns. In Figure 4 we compare both pores2 and its left approximate inverse. The number of nonzero entries is similar, but the sparsity patterns are quite different. For pores2, GMRES(20) reduced the relative residual below 10^{-6} after 106 iterations but did not improve any further. Again, a different implementation of GMRES [16] might mitigate this problem. In saylor3 we discovered in columns 988, 989, 998, and 999 two independent 2×2 singular submatrices. We ignored those four columns in the computation of M and simply replaced the two submatrices by 2×2 identity matrices. To guarantee the existence of a solution, we set the right-hand side to A times a random vector. The orsx, shermanx, poresx, and saylorx problems were all taken from the Harwell–Boeing matrix collection. A comparative study of these problems for different iterative methods using incomplete LU preconditioning can be found in [15].

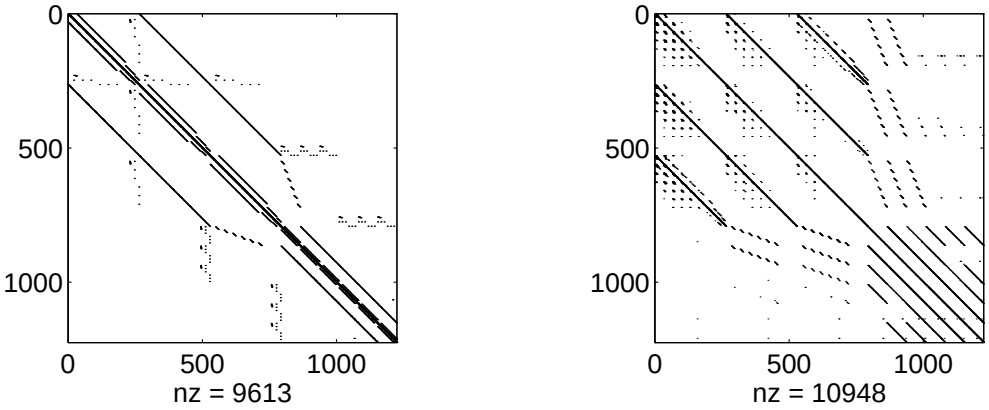

FIG. 4. *Pores2: original matrix A (left) and sparse approximate inverse M with $\varepsilon = 0.4$ (right).*

TABLE 5

Convergence results for poresx and saylorx: unpreconditioned (top) and preconditioned (middle and bottom). Left preconditioning is used for pores2.

	pores 2	pores 3	saylor 3	saylor 4
Bi-CGSTAB	> 1000	> 1000	369	> 1000
GMRES(20)	> 1000	> 1000	> 1000	> 1000
Bi-CGSTAB	78	118	104	285
$nz(M)/nz(A)$	1.14	1.17	1.29	2.02
ε | max. $nz(m_k)$	0.4 | 50	0.4 | 50	0.4 | 50	0.3 | 100
Bi-CGSTAB	27	118	67	67
GMRES(20)	103 ($< 10^{-6}$)	599	69	> 1000
$nz(M)/nz(A)$	5.16	4.82	2.00	3.8
ε | max. $nz(m_k)$	0.2 | 150	0.2 | 50	0.3 | 100	0.2 | 150

In the final part of this section we shall consider several large problems. The right-hand side was always provided, the initial guess $x_0 = 0$, and the tolerance as in (33). We start with four typical problems from Centric Engineering:

P1: incompressible flow in a pressure driven pipe, $T = 05$, $n = 3242$, and $nz = 293409$.

P2: incompressible flow in a pressure driven pipe, $T = 25$, $n = 3242$, and $nz = 293551$.

P3: landing hydrofoil airplane FSE model, $n = 6316$, and $nz = 167178$.

P4: slosh tank model, $n = 3402$, $nz = 130371$.

The numerical results are shown in Table 6. We remark that we have used the actual number of nonzero elements in the Px matrices, since about 0.3% of the entries were zero in the original data files. It is quite remarkable that although the Px matrices are rather full, we obtain such a big improvement in convergence with much sparser approximate inverses. Moreover, because of the periodic pattern in Figure 5, one could determine the sparsity pattern of the first few columns of M and then slide the pattern about the diagonal down to the lower right corner to get the full sparsity structure.

To conclude this series of numerical experiments, we consider three problems coming from an implicit two-dimensional Euler solver for an unstructured grid [19]. The

TABLE 6
Convergence results for Px: unpreconditioned (top) and preconditioned (bottom).

	P1		P2		P3		P4	
Bi-CGSTAB	221		323		108		> 1000	
GMRES(30)	> 1000		> 1000		67		> 1000	
Bi-CGSTAB	79		83		5		6	
GMRES(30)	866		685		10		12	
$nz(M)/nz(A)$	0.10		0.33		0.20		0.37	
ε | max. $nz(m_k)$	0.3	50	0.25	100	0.4	100	0.4	150

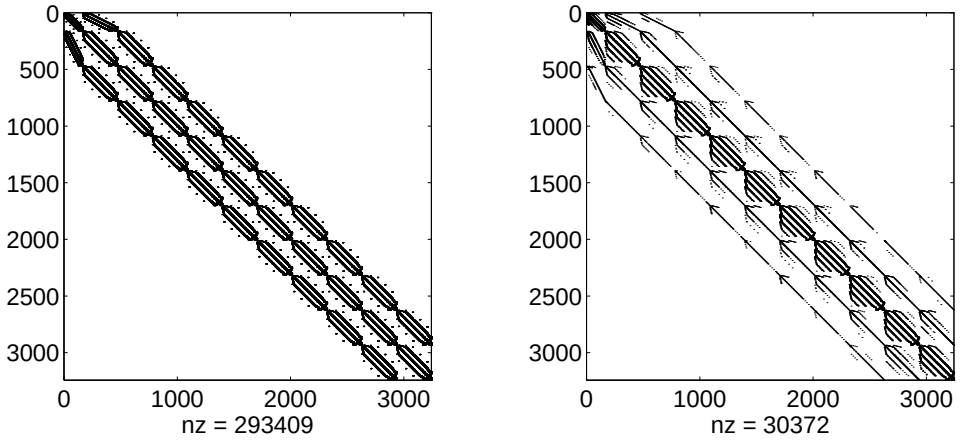

FIG. 5. *P1: original matrix A (left) and sparse approximate inverse M with $\varepsilon = 0.3$ (right).*

matrices are of order $n = 62424$ with $nz = 1717792$ nonzero elements and correspond to the initial and later stages in the flow simulation. None of the problems converged without preconditioning. As shown in Figure 6, the first problem T01 was much easier, since the flow was still in its initial stage. It is also discussed in [7]. Again, we see with T01 that we obtain a considerable improvement in convergence with a very sparse approximate inverse. For T01 we set $\varepsilon = 0.4$ and allowed for at most 50 nonzero entries per column in M, which led to a relative sparsity of $nz(M)/nz(A) = 0.3$. For both T25 and T50 we set $\varepsilon = 0.3$ and allowed for at most 100 nonzero entries per column in M; this led to a relative sparsity of $nz(M)/nz(A) = 1.3$.

6. Conclusion. The SPAI algorithm computes a sparse approximate inverse M of a general sparse matrix A. It is inherently parallel, since the columns of M are calculated independently of one another. The matrix M gives valuable insight into A^{-1}, and provides a measure on the proximity of M to A^{-1}. Instead of imposing an a priori sparsity pattern upon M, we let the algorithm capture automatically the relevant entries in the inverse. Thus, we minimize the number of nonzero entries in M and concentrate the computational effort where it is needed. The algorithm generates a robust and flexible preconditioner for iterative solvers, as it gives full control over the sparsity and the quality of M. We have shown both from a theoretical and a practical point of view that the preconditioner generated by the SPAI algorithm is very effective in improving the convergence of iterative solvers.

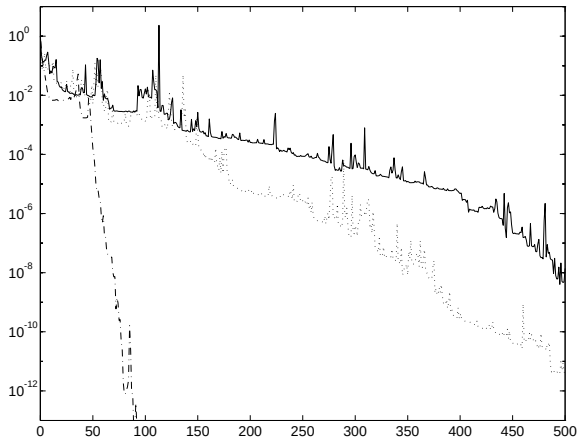

FIG. 6. *Convergence history of Tx with Bi-CGSTAB: relative residual vs. number of iterations for* $T01$(*dash-dot*), $T25$(*dotted*), $T50$(*solid*).

It is possible to minimize the total execution time for a particular problem and architecture by choosing an optimal M. It is clear that a very sparse preconditioner is very cheap but may not lead to much improvement in convergence and that if M becomes too dense, it becomes too expensive to compute. The optimal preconditioner lies somewhere between these two extremes and is problem and architecture dependent. In a parallel environment with very slow communication capabilities, such as a cluster of workstations, it may be advantageous to compute a fairly dense approximate inverse. This will increase the local floating-point intensive computations and reduce the number of iterations; hence, the amount of communications required at each iteration by inner products and matrix-vector products will be diminished.

The implementation of this method on a parallel computer is straightforward, since all calculations are done independently. Yet, each processor must have access to the data required to solve its subproblem. Although the preconditioner is invariant under permutation, the ordering of the unknowns can affect the amount of interprocessor communication involved in the computation of M and may be optimized for a particular application. This approach will prove particularly effective when a linear system needs to be solved repeatedly, such as in implicit time-marching schemes or the solution of nonlinear equations.

Acknowledgments. We thank Horst Simon for providing us with the matrices and for his helpful comments. This paper was written while the second author was visiting the SCCM program at Stanford University. He would like to thank Gene Golub for his kind hospitality.

REFERENCES

[1] O. AXELSSON, *Iterative Solution Methods*, Cambridge University Press, Cambridge, U.K., 1994.
[2] R. BARRET ET AL., *Templates for the Solution of Linear Systems*, SIAM, Philadelphia, PA, 1994.
[3] E. CHOW AND Y. SAAD, *Approximate inverse preconditioners via sparse-sparse iterations*, SIAM J. Sci. Comput., to appear.
[4] J. D. F. COSGROVE, J. C. DIAZ, AND A. GRIEWANK, *Approximate inverse preconditionings for sparse linear systems*, Internat. J. Comput. Math., 44 (1992), pp. 91–110.
[5] S. DEMKO, W. F. MOSS, AND P. W. SMITH, *Decay rates for inverses of band matrices*, Math. Comp., 43 (1984), pp. 491–499.

[6] R. W. Freund, G. H. Golub, and N. N. Nachtigal, *Iterative solution of linear systems*, Acta Numerica, 1991, pp. 57–100.
[7] R. W. Freund and N. N. Nachtigal, *Implementation details of the coupled QMR algorithm*, in Numerical Linear Algebra, L. Reichel, A. Ruttan, and R. S. Varga, eds., W. de Gruyter, Berlin, 1993, pp. 123–140.
[8] G. Golub, *Numerical methods for solving linear least squares problems*, Numer. Math., 7 (1965), pp. 206–216.
[9] M. Grote and H. Simon, *Parallel preconditioning and approximate inverses on the Connection Machine*, in Proceedings of the Scalable High Performance Computing Conference (SHPCC), IEEE Press, Piscataway, NJ, 1992, pp. 76–83.
[10] M. Grote and H. Simon, *Parallel preconditioning and approximate inverses on the Connection Machine*, in Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing, vol. II, R. Sincovec et al., eds., SIAM, Philadelphia, PA, 1993, pp. 519–523.
[11] L. Yu. Kolotilina and A. Yu. Yeremin, *Factorized sparse approximate inverse preconditionings*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 45–58.
[12] L. Yu. Kolotilina, A. A. Nikishin, and A. Yu. Yeremin, *Factorized sparse approximate inverse (FSAI) preconditionings for solving 3D FE systems on massively parallel computers* II, Iterative Methods in Linear Algebra, in Proceedings of the IMACS International Symposium, R. Beauwens and P. de Groen, eds., Brussels, 1992, pp. 311–312.
[13] Ju. B. Lifshitz, A. A. Nikishin, and A. Yu. Yeremin, *Sparse approximate inverse preconditionings for solving 3D CFD problems on massively parallel computers*, Iterative Meth. in Lin. Alg., Proc. of the IMACS Internat. Sympos., R. Beauwens and P. de Groen, eds., Brussels, 1992, pp. 83–84.
[14] Y. Saad, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM J. Sci. Statist. Comput., 14 (1993), pp. 461–469.
[15] C. H. Tong, *A comparative study of preconditioned Lanczos methods for nonsymmetric linear systems*, Sandia report SAND91-8240, Sandia National Laboratories, Albuquerque, NM, 1992.
[16] K. Turner and H. F. Walker, *Efficient high accuracy solutions with GMRES(m)*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 815–825.
[17] H. A. Van der Vorst and C. Vuik, *GMRESR: a family of nested GMRES methods*, Numer. Linear Alg. Appl., 1 (1993), pp. 1–7.
[18] H. A. Van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 631–644.
[19] V. Venkatakrishnan and T. J. Barth, *Unstructured grid solvers on the iPSC/860*, in Proceedings of the Conference on Parallel Computational Fluid Dynamics 1992, R. B. Pelz, A. Ecer, and J. Hauser, eds., North–Holland, New York, 1993.