Numerical Algorithms 16 (1997) 1–15

Sparse approximate inverse preconditioning for dense linear systems arising in computational electromagnetics

Guillaume Alléon [a], Michele Benzi [b,*] and Luc Giraud [b]

[a] *Parallel Computing Group, Physics and Mathematics Department, Aerospatiale, 12 Rue Pasteur, BP 76, F-92150 Suresnes Cedex, France*
E-mail: guillaume.alleon@siege.aerospatiale.fr

[b] *Parallel Algorithms Project, CERFACS, 42 Ave. G. Coriolis, F-31057 Toulouse Cedex, France*
E-mail: benzi@lanl.gov

We investigate the use of sparse approximate inverse preconditioners for the iterative solution of linear systems with dense complex coefficient matrices arising in industrial electromagnetic problems. An approximate inverse is computed via a Frobenius norm approach with a prescribed nonzero pattern. Some strategies for determining the nonzero pattern of an approximate inverse are described. The results of numerical experiments suggest that sparse approximate inverse preconditioning is a viable approach for the solution of large-scale dense linear systems on parallel computers.

Keywords: dense linear systems, preconditioning, sparse approximate inverses, complex symmetric matrices, scattering calculations, Krylov subspace methods, parallel computing

AMS subject classification: 65F10, 65F50, 65R20, 65N38, 78-08, 78A50, 78A55

1. Introduction

In the last decade, a significant amount of effort has been spent on the simulation of electromagnetic wave propagation phenomena to address various topics ranging from electromagnetic compatibility, to stealth, to absorbing materials, and antenna design. Two complementary approaches based on the solution of the Maxwell equations are often adopted for tackling these problems. The first approach utilizes finite differencing in the time domain. The second operates in the frequency domain. The latter approach offers two main advantages: (a) it does not require truncating the infinite spatial domain surrounding the scatterer and using approximate boundary conditions, and (b) it requires discretizing only the surface of the scatterer. On the other hand, the frequency domain approach leads to singular integral equations of the first kind, the discretization of which results in linear systems with complex and dense matrices which are quite challenging to solve. With the advent of parallel processing, both

* Present address: Scientific Computing Group (CIC-19), MS B256, Los Alamos National Laboratory, Los Alamos, NM 87545, USA.

time and frequency domain methods have witnessed important developments and the typical electromagnetic problem size in industry has increased by one order of magnitude.

In this paper, we consider the solution of linear systems of the form

$$Ax = b, \tag{1}$$

where the coefficient matrix $A = [a_{ij}]$ is a large, dense, complex matrix of order n arising from the boundary element method in the frequency domain. Incidentally, we note that in the engineering literature these equations are usually written as $Zj = e$, where Z is the impedance matrix, j the vector of currents, and the right-hand side e corresponds to the discretization of the incident electromagnetic field. The coefficient matrix can be symmetric non-Hermitian or nonsymmetric, depending on the geometry of the object being analyzed. For example, A is symmetric in the case of a three-dimensional axisymmetric object, but nonsymmetric in the case of a three-dimensional cyclic object. In this paper, we will assume that A is symmetric, but the techniques considered here can be applied equally well to nonsymmetric matrices.

Direct solution methods based on (out-of-core) block LU factorizations are often preferred in an industrial environment because they are deemed more reliable [1], but this reliability comes at the price of O(n^3) arithmetic operations. Iterative methods could be an attractive alternative, but they require preconditioning in order to be effective. The purpose of this paper is to investigate a class of parallel preconditioners for problems of this kind. It is worth mentioning that iterative methods become particularly attractive in those situations where the matrix–vector products involving the coefficient matrix A can be computed using "fast" techniques, which can reduce the cost of each iteration from O(n^2) to O($n \log n$) or even O(n), depending on the problem. However, we will not consider such techniques in the present study.

Typically, system (1) must be solved for many right-hand sides b simultaneously, and it is therefore natural to use block iterative methods. A well suited block algorithm for complex symmetric systems with multiple right-hand sides is block QMR (see, e.g., [7,12,13]). If both A and the preconditioner are symmetric, QMR can be simplified to exploit symmetry, resulting in lower computational complexity. As we shall see, requiring the preconditioner to be symmetric can be restrictive in certain situations, in which case block GMRES [23] can also be used. Another popular method in the engineering community is CGNR (conjugate gradients on the normal equations). Whatever the iterative method chosen, preconditioning is the key ingredient for the successful use of iterative methods.

The remainder of this paper is organized as follows. In section 2, the general idea of preconditioning for linear systems of equations is briefly rewieved. Sections 3 and 4 are concerned with the special class of preconditioners considered in this paper, sparse approximate inverses. Numerical experiments with four model problems arising in scattering calculations are presented in section 5. A few concluding remarks are given in section 6.

2. Generalities on preconditioning

It is well known that the rate of convergence of iterative methods for solving (1) is strongly influenced by the spectral properties of A. Preconditioning amounts to transforming the original system into one having the same solution but more favorable spectral properties, such as a clustering of the eigenvalues around 1. A *preconditioner* is a matrix that can be used to accomplish such a transformation. If M is a nonsingular matrix which approximates A^{-1} ($M \approx A^{-1}$), the transformed linear system

$$MAx = Mb \tag{2}$$

will have the same solution as system (1) but the convergence rate of iterative methods applied to (2) may be much higher. Problem (2) is preconditioned from the left, but *right* preconditioning is also possible. Preconditioning on the right leads to the transformed linear system

$$AMy = b. \tag{3}$$

Once the solution y of (3) has been obtained, the solution of (1) is given by $x = My$. The choice between left or right preconditioning often depends on the choice of the iterative method, and on properties of the coefficient matrix (the side of the preconditioner can be important for nonsymmetric problems).

Loosely speaking, the closer M is to the exact inverse of A, the higher the rate of convergence of the iterative method. Choosing $M = A^{-1}$ yields convergence in one step, but of course constructing such a preconditioner is equivalent to solving the original problem. In practice, the preconditioner M should be easily computed and applied, so that the total time for the solution of the preconditioned system is less than the time for the unpreconditioned one. In addition, the preconditioner should not require too much storage. It is therefore natural to consider sparse preconditioners.

Most existing preconditioners can be broadly classified as being either of the *implicit* or of the *explicit* kind. A preconditioner is implicit if its application, within each step of the chosen iterative method, requires the solution of a linear system. Perhaps the most important example is provided by preconditioners based on an Incomplete LU (ILU) decomposition. Here M is implicitly defined by $M = (\overline{L}\overline{U})^{-1}$, where $\overline{L}$ and $\overline{U}$ are triangular matrices which approximate the exact L and U factors of A. Applying the preconditioner requires the solution of two triangular systems (the forward and backward solves). Matrices $\overline{L}$ and $\overline{U}$ are computed by performing the LU factorization of A incompletely, dropping elements of the triangular factors according to some selection rule in the course of the computation. These techniques have been developed with sparse matrices in mind, but they may also be applied to dense linear systems by extracting a sparse matrix B from A (usually by setting entries of A below a prescribed threshold to zero) and performing an incomplete factorization of B, see [19]. Implicit preconditioners are quite robust and often give fast convergence of the preconditioned iteration, but are difficult to implement in parallel. In particular, the triangular solves involved in ILU-type preconditioning represent a serial bottleneck

(due to the recursive nature of the computation), thus limiting the effectiveness of this approach on vector and parallel computers.

In contrast, with explicit preconditioning a matrix $M \approx A^{-1}$ is explicitly computed and the preconditioning operation reduces to forming a matrix–vector product, or a matrix–matrix product if multiple right-hand sides are present. If M is sparse, this is an O(n) operation, a small overhead as compared to the O(n^2) cost of a straightforward implementation of matrix–vector products involving the coefficient matrix A.

These computations are easier to parallelize than the sparse triangular solves. Furthermore, the construction of some types of approximate inverse preconditioners can be performed in parallel. For these reasons, sparse approximate inverse preconditioners are becoming increasingly popular as an alternative to more traditional implicit techniques for the parallel solution of large and sparse problems. See [6] for further discussion of the relative advantages and disadvantages of implicit and explicit preconditioners. Thorough treatments of preconditioning for sparse matrices can be found, e.g., in [3,22]. Preconditioning techniques for dense linear systems have received less attention; see, e.g., [8] and the references therein. To our knowledge, sparse approximate inverse preconditioning for dense matrices was first considered in [20]. See also [24] and the internal report [10].

3. Methods based on Frobenius norm minimization

A good deal of work has been devoted to explicit preconditioning based on the following approach: the sparse approximate inverse is computed as the matrix M which minimizes $\|I - MA\|$ (or $\|I - AM\|$ for right preconditioning) subject to some sparsity constraint (see [3,5,9,15–17,20,22]). Here the matrix norm is usually the Frobenius norm or a weighted variant of it, for computational reasons. With this choice, the constrained minimization problem decouples into n independent linear least squares problems (one for each row, or column of M), the number of unknowns for each problem being equal to the number of nonzeros allowed in each row (or column) of M. This immediately follows from the identity

$$\|I - AM\|_F^2 = \sum_{j=1}^{n} \|e_j - Am_j\|_2^2,$$

where e_j is the jth unit vector and m_j is the jth column of M. Clearly, there is considerable scope for parallelism in this approach. The resulting least squares problems can be solved, in principle, independently of each other, for instance by direct methods.

The main issue is the selection of the nonzero pattern of M. The idea is to keep M reasonably sparse while trying to capture the "large" entries of the inverse, which are expected to contribute the most to the quality of the preconditioner. In early papers (e.g., [5,17]) the sparsity constraint was imposed a priori, and the minimizer was found relative to a class of matrices with a predetermined sparsity pattern. For

instance, when A is a band matrix with a good degree of diagonal dominance, a banded approximation to A^{-1} is often justified (see [11]). However, in the context of general sparse matrices it is not known how to find a good sparsity pattern for the approximate inverse a priori, and several papers have addressed the problem of adaptively defining the nonzero pattern of M in order to capture "large" entries of the inverse [9,15,16].

The use of sparse approximate inverse preconditioners for *dense* linear systems was first proposed by Kolotilina. In her paper [20], a few heuristics were given in order to prescribe a nonzero pattern for M. The simplest criterion is to choose a threshold parameter $\varepsilon \in (0, 1)$ and to include the position (i, j) in the nonzero pattern of M if

$$|a_{ij}| > \varepsilon \cdot \max_{1 \leqslant k,l \leqslant n} |a_{kl}|. \tag{4}$$

As reported in [20], this simple approach gave good results for some relevant problems. Other heuristics, which are refinements of this simple criterion, are described in [20]. In this paper we consider other variants of this idea. As we shall see, good results can be obtained using a prescribed sparsity pattern, and therefore we do not consider the much more expensive adaptive strategies which have been devised for general sparse matrices.

Even with a prescribed sparsity pattern, the serial cost for the construction of this type of preconditioner is relatively high, especially compared with the cost of standard implicit techniques like ILU. An exact operation count is not possible in general, since it depends on the choice of the sparsity pattern. In any case, substantial speed-ups can be obtained when the parallelism inherent in the preconditioner construction is exploited.

4. Construction of the preconditioner

In this section we describe how to compute the sparse approximate inverse $M = [m_{ij}]$, and discuss criteria for determining the nonzero pattern of M. For the sake of brevity, we restrict ourselves to right preconditioning. The construction of left preconditioners is analogous.

Assume first that the nonzero pattern of M has been prescribed. The nonzero pattern is a subset $\mathcal{G} \subseteq \{(i, j);\ 1 \leqslant i, j \leqslant n\}$ such that $m_{ij} = 0$ if $(i, j) \notin \mathcal{G}$. As before, we denote by m_j the jth column of M ($1 \leqslant j \leqslant n$). For a fixed j, consider the set $J = \{i;\ (i, j) \in \mathcal{G}\}$, which specifies the nonzero pattern of m_j. Clearly, the only columns of A that enter into the definition of m_j are those whose index is in J. Let $A(:, J)$ be the submatrix of A formed from such columns, and let I be the set of indices of nonzero rows of $A(:, J)$. Then we can restrict our attention to the matrix $\widehat{A} = A(I, J)$, to the unknown vector $\widehat{m}_j = m_j(J)$ and to the right-hand side $\hat{e}_j = e_j(J)$. The nonzero entries in m_j can be computed by solving the (small) unconstrained least squares problem

$$\min \left\| \hat{e}_j - \widehat{A}\widehat{m}_j \right\|_2.$$

This least squares problem can be solved, for instance, by orthogonal factorization. Again, each column m_j can be computed, at least in principle, independently of the other columns of M. Note that if A is sparse, then $\widehat{A}$ will contain only a few nonzero rows and columns, so each least squares problem has small size and can be solved efficiently by dense matrix techniques.

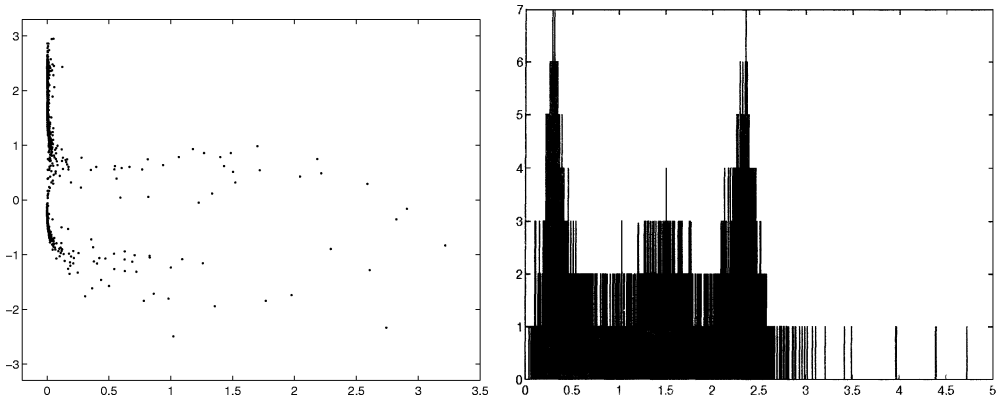
We now describe four simple heuristics that can be used to prescribe $\mathcal{G}$. These heuristics are motivated by the empirical observation that for the kind of problems considered here, the large entries of A^{-1} tend to be in positions that correspond to large entries in A, or nearby (see the examples in the next section).

1. For a fixed positive integer k, with $k \ll n$, find the k largest entries (in modulus) in each column of A. Then $\mathcal{G}$ is the set of positions (i, j) of the resulting $k \cdot n$ entries.
2. For each column, find the row indices of the k largest entries in modulus; then, for each row index i found, the same search is performed on column i. The new row indices found are added to the previous ones to determine the nonzero structure of the column. This is the *neighbours of neighbours* heuristic proposed in [10].
3. The previous heuristic can be extended in an obvious way by performing several iterations instead of just one. However, to preserve sparsity in the approximate inverse, the number of largest entries to be located is halved at each iteration. In practice, two iterations are enough.
4. This is Kolotilina's heuristic (4).

We compare these heuristics on four test matrices in the next section. The first and the last heuristics are the simplest, and have obvious advantages from the point of view of parallel implementation (no interprocessor communication is needed if the matrix is distributed by columns among the processors). The first one has the additional advantage that the amount of nonzeros in the approximate inverse is controlled entirely by the user; furthermore, it allows one to achieve a perfect load balancing in a parallel implementation. A drawback common to all the heuristics is the need to find good values of the parameters involved. However, most algebraic preconditioning strategies are likely to share the same limitation. Some experience with these methods and with particular applications is needed in order to make a good choice of the parameters involved.

5. Experimental results

In this section, we report on the results of numerical experiments aimed at assessing the viability of sparse approximate inverse techniques for preconditioning large, dense, complex symmetric matrices arising from scattering calculations. The spatial discretization method adopted here relies on the Stratton–Chu boundary integral formulation of the Maxwell equations in the frequency domain, and the discrete element



(a) Eigenvalue distribution (b) Singular value histogram

Figure 1. Eigenvalues and singular values of CETAF1101.

of Raviart–Thomas. Four different matrices are generated, corresponding to four different scatterers. While these matrices are relatively small, they are by no means easy to solve and capture many of the features of real-life problems.

- Example 1: CETAF1101, a matrix of order $n = 1101$.
- Example 2: STR599, a matrix of order $n = 599$.
- Example 3: SPH624, a matrix of order $n = 624$.
- Example 4: PL1825, a matrix of order $n = 1825$.

Each matrix corresponds to a different metallic structure: an aircraft wing (with an antenna), a strip, a sphere and a square plate, respectively.

In order to gain some insight into the nature of these problems, we performed a complete analysis of the four matrices, including the determination of singular values and eigenvalues and the computation of the inverse using standard LAPACK [2] subroutines. For CETAF1101, the largest and smallest singular values were found to be $\sigma_{\max} \approx 4.727$ and $\sigma_{\min} \approx 3.610 \cdot 10^{-2}$, respectively, so that the spectral condition number is ≈ 131. A plot of the eigenvalues of this matrix is shown in figure 1(a), and a histogram of the singular values is shown in figure 1(b). The singular value information is reported here because of its relevance for the CGNR method. Even if the condition number in this case is not particularly large, it should be clear from the eigenvalue distribution that preconditioning is necessary. Plots of the eigenvalues of STR599, SPH624 and PL1825 are shown in figures 8(a), 9(a) and 10(a), respectively.

As we already observed, the large entries of A^{-1} tend to be located in positions that correspond to large entries of A, which is the rationale for the heuristics described in the previous section. Figures 2 and 3 provide an illustration of this fact for the matrix CETAF1101.

In figure 2 we have plotted the nonzero pattern of the matrix B obtained from A by dropping all entries less than $\varepsilon = 0.075$ in absolute value (A was first rescaled so that $\max_{ij} |a_{ij}| = 1$). This matrix is more than 98% sparse. The inverses A^{-1}

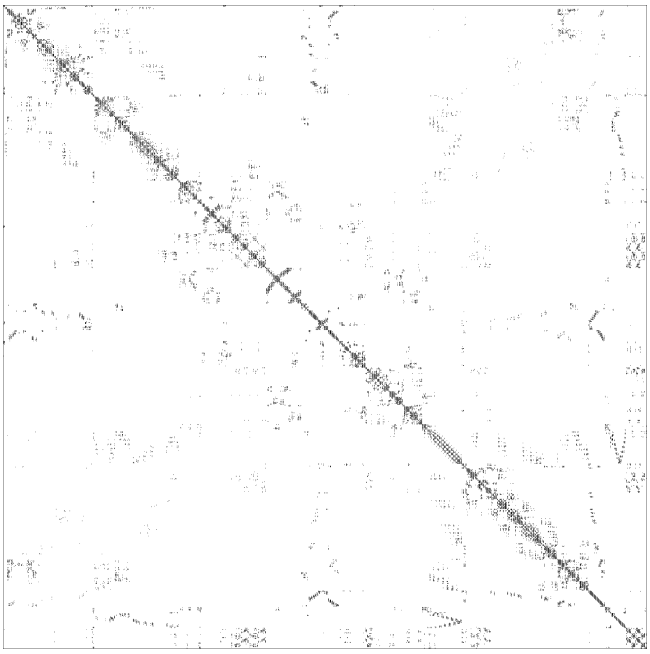


Figure 2. Pattern of $B = \textit{sparsified}(A)$, CETAF1101.

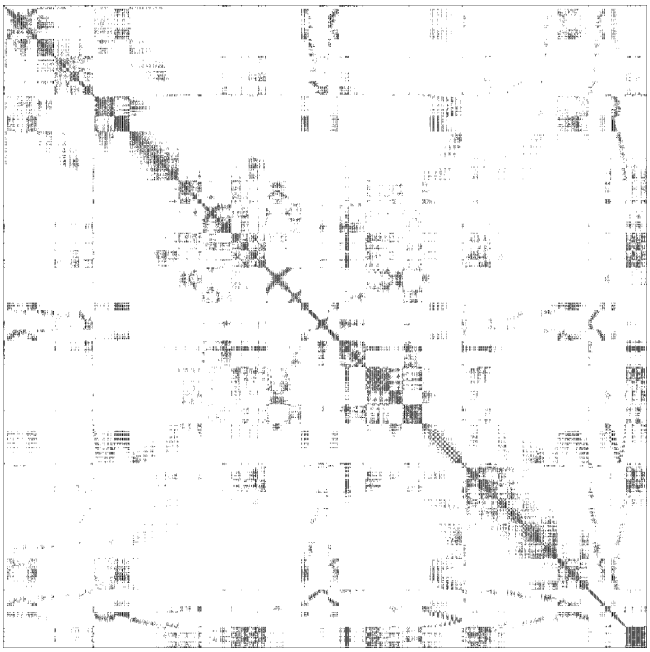
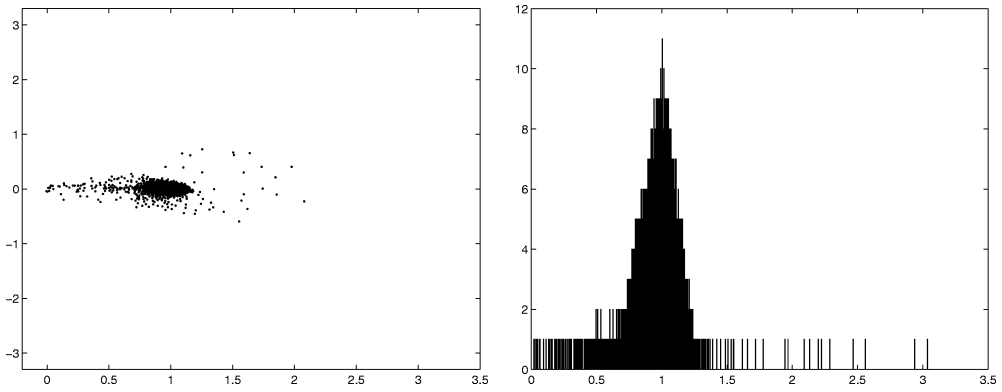


Figure 3. Pattern of $\textit{sparsified}(B^{-1})$, CETAF1101.



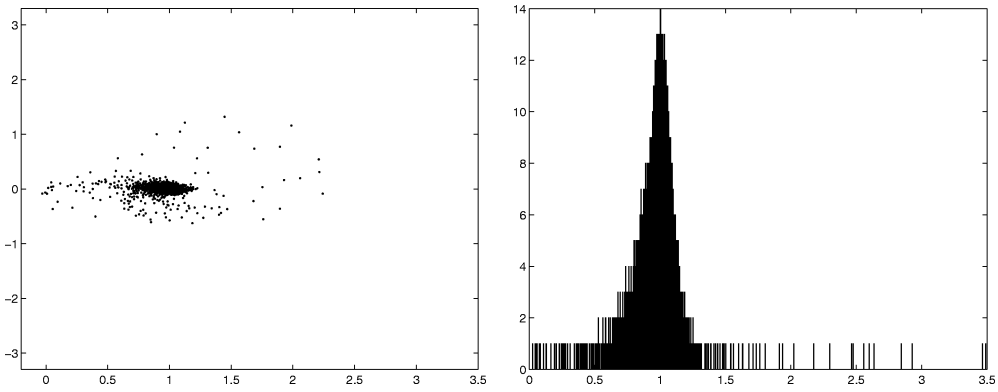
(a) Eigenvalue distribution (b) Singular value histogram

Figure 4. Using heuristic 1 on CETAF1101.

and B^{-1} were computed (using LAPACK) and then "sparsified" by discarding all entries smaller than ε in absolute value. The resulting nonzero patterns were found to be virtually identical, both similar to the nonzero pattern of B, although not quite as sparse (about 95%). Figure 3 shows the nonzero pattern obtained from sparsification of B^{-1}. This observation suggests that when computing a sparse approximate inverse preconditioner, it is sufficient to work with a sparse approximation of the coefficient matrix A rather than with A itself, an important observation from a practical standpoint.

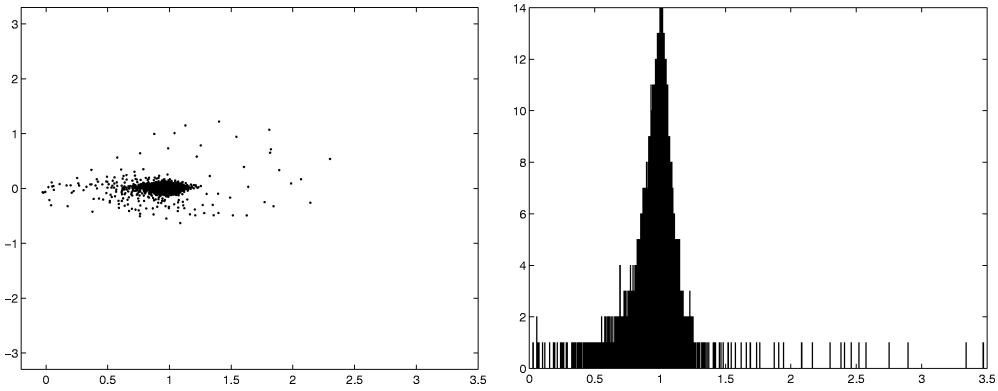
We computed different approximate inverses using the four heuristics described in the previous section, for different values of the parameters k and ε. This took some tuning of the parameters before satisfactory approximate inverses were obtained. For CETAF1101, it was found that preconditioners of acceptable quality required between 50 and 70 nonzero entries in each column, corresponding to a sparsity around 93–96%. We do not show the corresponding nonzero patterns because they are virtually indistinguishable from that of figure 3, a fact that demonstrates that the heuristics are effective at capturing the important entries of the inverse. We constructed both left and right approximate inverses (to be used as left and right preconditioners) and we found that the results were similar in all cases. Also, it was found that very little is gained if A is used instead of a sparse approximation to it in the construction of the approximate inverse. Figures 4–7 show the effects of approximate inverse preconditioning on the singular value and eigenvalue distribution for CETAF1101 using the four heuristics described in the previous section.

From these figures it appears that the four heuristics are roughly equivalent, in the sense that they produce a similar clustering of the eigenvalues and singular values. It is interesting to observe that the condition number of the preconditioned matrix is virtually the same as the condition number of the original matrix (for some heuristics it is actually slightly larger). Figures 8–10 show the effect of preconditioning with heuristic 1 for the matrices STR599, SPH624 and PL1825, respectively.



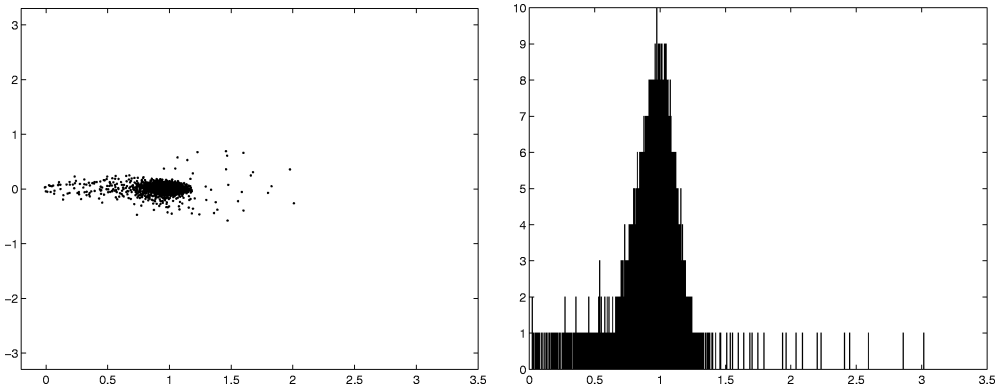
(a) Eigenvalue distribution (b) Singular value histogram

Figure 5. Using heuristic 2 on CETAF1101.



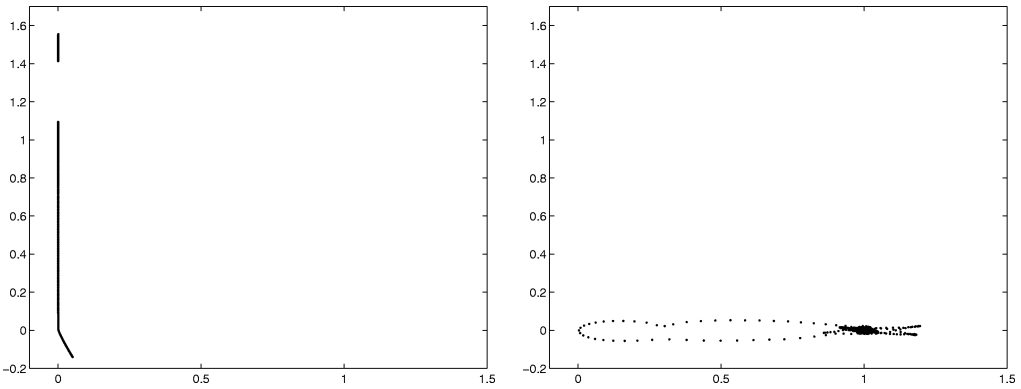
(a) Eigenvalue distribution (b) Singular value histogram

Figure 6. Using heuristic 3 on CETAF1101.



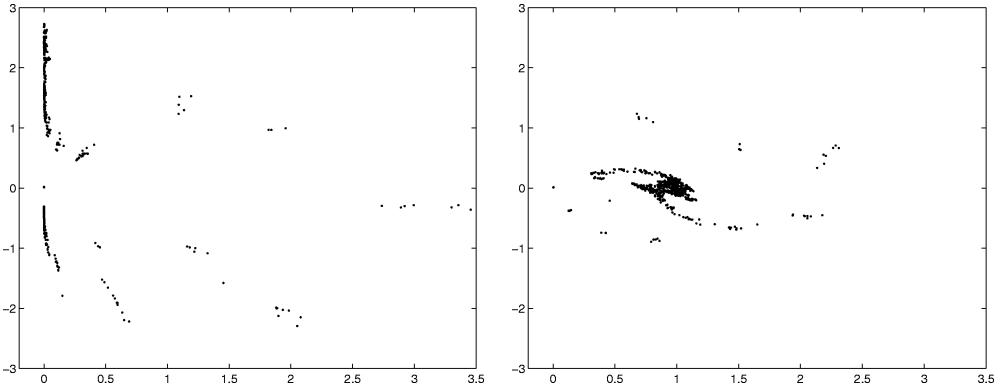
(a) Eigenvalue distribution (b) Singular value histogram

Figure 7. Using heuristic 4 on CETAF1101.



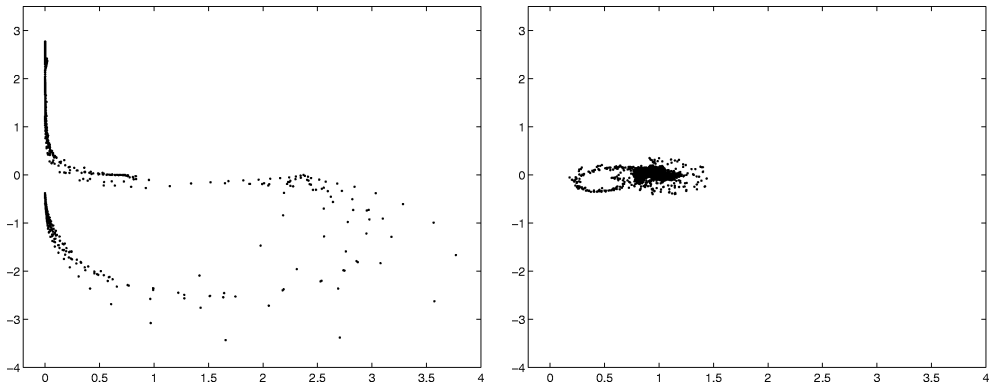
(a) Unpreconditioned matrix (b) Preconditioned using heuristic 1

Figure 8. Eigenvalue distributions, STR599.



(a) Unpreconditioned matrix (b) Preconditioned using heuristic 1

Figure 9. Eigenvalue distributions, SPH624.



(a) Unpreconditioned matrix (b) Preconditioned using heuristic 1

Figure 10. Eigenvalue distributions, PL1825.

Numerical experiments were performed with the aim of assessing the effectiveness of these sparse approximate inverses as preconditioners for iterative linear solvers. As noted before, QMR is the Krylov subspace method which is usually recommended for the complex symmetric case, because this method is able to exploit symmetry, leading to a more efficient algorithm. As shown in [13], this desirable property is preserved when preconditioning is used, provided that the preconditioner M is itself symmetric. The preconditioner computed via the Frobenius minimization approach is, in general, nonsymmetric. Symmetry can be restored by replacing M with $(M + M^T)/2$. Unfortunately, this operation requires global communication among the processors in a distributed memory environment, where it is assumed that the columns (or rows) of M are distributed among the processors. An alternative would be to compute a factored approximate inverse, see [6,21], which is automatically symmetric. We performed some experiments with the method in [6], which is quite effective for general sparse matrices, but the results were unsatisfactory. For these reasons we used (restarted) GMRES and a standard implementation of QMR as the Krylov subspace methods and we found that, for our model problem, QMR required more matrix–vector products than GMRES to achieve convergence. Therefore we restrict our attention to GMRES, but in fairness it should be stressed that simplified QMR should be used if a symmetric preconditioner is available.

The restarted GMRES method, GMRES(m), without preconditioning was applied to four linear systems $Ax = b$, where A is one of our four test matrices and b is chosen so that the exact solution of the system was known. The initial guess was taken to be $x_0 = \mathbf{0}$ (the null vector), and the iteration was stopped when the current approximation x_k was found to satisfy the following inequality:

$$\frac{||b - Ax_k||_2}{||A||_F \, ||x_k||_2 + ||b||_2} \leqslant 10^{-7}, \tag{5}$$

which can be justified by backward stability considerations (see, e.g., [14,18]). In all cases, convergence was found to be slow, thus confirming that preconditioning is necessary for this kind of problem.

In tables 1–4 we show the results of runs with GMRES(m) for the four test matrices with various choices of m and four different sparse approximate inverse preconditioners, each corresponding to one of the four heuristics described in the previous section. For completeness, we also include results for full GMRES ($m = \infty$). The parameters governing the various heuristics were adjusted to produce approximate inverses with roughly the same number of nonzeros (sparsity around 94–96%). For comparison purposes, we also show the number of GMRES iterations without preconditioning.

It can be seen from these results that, on average, the various heuristics all gave comparable results. Note that the simplest heuristic, the first one, gives results that are about as good as the other, more complicated ones. We also note that it is best to use GMRES with a high value of the restart parameter or no restart, except for matrix PL1825 where excellent results are obtained with a restart parameter as low as $m = 5$.

Table 1
CETAF1101: number of matrix–vector products for GMRES with various approximate inverse preconditioners and different values of the restart parameter. The second column shows the number of nonzeros in the approximate inverse.

Heuristic	$NZ(M)$	$m = 10$	$m = 20$	$m = 30$	$m = 40$	$m = 50$	$m = \infty$
1	78,172	651	256	219	177	165	87
2	77,872	568	338	270	210	163	80
3	77,795	626	330	239	200	168	81
4	78,526	2990	493	289	237	216	101
no precond.	–	4903	3245	2637	2186	1794	300

Table 2
Test results for STR599.

Heuristic	$NZ(M)$	$m = 10$	$m = 20$	$m = 30$	$m = 40$	$m = 50$	$m = \infty$
1	14,976	172	120	124	80	94	55
2	15,454	108	88	84	68	80	54
3	14,476	172	120	124	80	94	55
4	15,398	137	123	95	73	85	54
no precond.	–	1116	984	863	790	668	163

Table 3
Test results for SPH624.

Heuristic	$NZ(M)$	$m = 10$	$m = 20$	$m = 30$	$m = 40$	$m = 50$	$m = \infty$
1	15,601	312	92	93	75	71	63
2	16,913	638	142	111	104	93	70
3	15,719	552	119	112	97	92	70
4	15,545	273	97	103	81	78	54
no precond.	–	2756	2230	2152	2129	2053	262

Table 4
Test results for PL1825.

Heuristic	$NZ(M)$	$m = 5$	$m = 10$	$m = 15$	$m = \infty$
1	142,351	22	21	22	21
2	144,329	19	19	18	18
3	142,971	21	20	21	20
4	142,351	28	27	26	24
no precond.	–	503	345	266	172

We conclude this section on numerical experiments by showing in table 5 the dependence of the convergence rate of full GMRES on the number of nonzeros in the approximate inverse preconditioner M, limited to the two smaller matrices preconditioned with heuristic 1. Note that this dependency is non-monotonic for STR599.

Table 5
Effect of number of nonzeros in M on the convergence of GMRES(∞).

STR599							
$NZ(M)$	–	5,991	8,986	11,981	14,976	17,971	20,966
GMRES(∞)	163	220	77	80	55	63	39
SPH624							
$NZ(M)$	–	6,241	9,361	12,481	15,601	18,721	21,841
GMRES(∞)	262	180	108	72	63	58	57

6. Conclusions

Based on the results of numerical experiments we conclude that sparse approximate inverses can be effective preconditioners for large, dense, complex linear systems. For the model problems considered in this paper, approximate inverse preconditioning resulted in a good clustering of the eigenvalues, which translated in a substantial reduction of the number of GMRES iterations required to attain convergence. We emphasize that the additional cost incurred by one iteration of GMRES due to application of the preconditioner is negligible, being an O(n) operation.

The computation and the application of the approximate inverse preconditioner can be performed in parallel. It is important to stress that the nonzero structure of the preconditoner can be determined a priori: this makes the construction of the preconditioner relatively straightforward.

In conclusion, the use of Krylov subspace methods preconditioned with sparse approximate inverses is expected to result in efficient parallel solvers for large-scale, dense linear systems of equations of the type arising in computational electromagnetics.

Acknowledgements

We would like to thank Iain Duff and Valeria Simoncini for their comments on an earlier version of the manuscript, and Serge Gratton for helping with the implementation of complex GMRES. This work has been partly funded by the European Commission under contract EP20170.

References

[1] G. Alléon, S. Amram, N. Durante, P. Homsi, D. Pogarieloff and C. Farhat, Massively parallel processing boosts the solution of industrial electromagnetic problems: high-performance out-of-core solution of complex dense systems, in: *Proceedings of 8th SIAM Conference on Parallel Processing for Scientific Computing*, eds. M. Heath et al. (SIAM, Philadelphia, PA, 1997).

[2] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov and D. Sorensen, *LAPACK User's Guide* (SIAM, Philadelphia, PA, 1992).

[3] O. Axelsson, *Iterative Solution Methods* (Cambridge University Press, Cambridge, 1994).
[4] R. Barret, M. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine and H. van der Vorst, *Templates for the Solution of Linear Systems* (SIAM, Philadelphia, PA, 1994).
[5] M. Benson, J. Krettmann and M. Wright, Parallel algorithms for the solution of certain large sparse linear systems, Internat. J. Comput. Math. 16 (1984) 245–260.
[6] M. Benzi and M. Tůma, A sparse approximate inverse preconditioner for nonsymmetric linear systems, CERFACS Technical Report TR/PA/96/15 (1996), to appear in SIAM J. Sci. Comput. 19 (1998).
[7] W.E. Boyse and A.A. Seidl, A block QMR method for computing multiple simultaneous solutions to complex symmetric systems, SIAM J. Sci. Comput. 17 (1996) 263–274.
[8] K. Chen, Efficient iterative solution of linear systems from discretizing singular integral equations, Electr. Trans. Numer. Anal. 2 (1994) 76–91.
[9] J.D.F. Cosgrove, J.C. Diaz and A. Griewank, Approximate inverse preconditionings for sparse linear systems, Internat. J. Comput. Math. 44 (1992) 91–110.
[10] A. Cosnuau, Etude d'un préconditionneur pour les matrices complexes dense issues des équations de Maxwell en formulation intégrale, Note technique ONERA No. 142328.96/DI/MT (1996).
[11] S. Demko, W.F. Moss and P.W. Smith, Decay rates for inverses of band matrices, Math. Comp. 43 (1984) 491–499.
[12] R. Freund, Conjugate gradient-type methods for linear systems with complex symmetric coefficient matrices, SIAM J. Sci. Statist. Comput. 13 (1992) 425–448.
[13] R. Freund and M. Malhotra, A block-QMR algorithm for non-Hermitian linear systems with multiple right-hand sides, Linear Algebra Appl. 254 (1997) 119–157.
[14] L. Giraud and S. Gratton, Solveurs linéaires performants pour la résolution de systèmes complexes non hermitiens creux de grande taille, CERFACS Technical Report TR/PA/95/24 (1995).
[15] N.I.M. Gould and J.A. Scott, On approximate-inverse preconditioners, Technical Report RAL-95-026, Rutherford Appleton Laboratory, Chilton, England (1995).
[16] M. Grote and T. Huckle, Parallel preconditioning with sparse approximate inverses, SIAM J. Sci. Comput. 18 (1997) 838–853.
[17] M. Grote and H. Simon, Parallel preconditioning and approximate inverses on the Connection Machine, in: *Proceedings of 6th SIAM Conference on Parallel Processing for Scientific Computing*, eds. R. Sincovec et al. (SIAM, Philadelphia, PA, 1993) pp. 519–523.
[18] E.M. Kasenally, GMBACK: A generalised minimum backward error algorithm for nonsymmetric linear systems, SIAM J. Sci. Comput. 16 (1995) 698–719.
[19] S. Kharchenko, P. Kolensnikov, A. Nikishin, A. Yeremin, M. Heroux and Q. Sheikh, Iterative solution methods on the Cray YMP/C90, Part II: Dense linear systems, in: *Simulation Multiconference: High Performance Computing Symposium* (Washington, DC, 1993).
[20] L.Yu. Kolotilina, Explicit preconditioning of systems of linear algebraic equations with dense matrices, J. Sov. Math. 43 (1988) 2566–2573. English translation of a paper first published in Zapiski Nauchnykh Seminarov Leningradskogo Otdeleniya Matematicheskogo Instituta im. V.A. Steklova AN SSSR 154 (1986) 90–100.
[21] L.Yu. Kolotilina and A.Yu. Yeremin, Factorized sparse approximate inverse preconditioning I: Theory, SIAM J. Matrix Anal. Appl. 14 (1993) 45–58.
[22] Y. Saad, *Iterative Methods for Sparse Linear Systems* (PWS Publishing Co., Boston, 1996).
[23] V. Simoncini and E. Gallopoulos, Convergence properties of block GMRES and matrix polynomials, Linear Algebra Appl. 247 (1996) 97–119.
[24] S.A. Vavasis, Preconditioning for boundary integral equations, SIAM J. Matrix Anal. Appl. 13 (1992) 905–925.