Using spectral low rank preconditioners for large electromagnetic calculations[1]

Iain S. Duff[2], Luc Giraud[3,4], Julien Langou[3,5], and Emeric Martin[3,6]

ABSTRACT

For solving large dense complex linear systems that arise in electromagnetic calculations, we perform experiments using a general purpose spectral low rank update preconditioner in the context of the GMRES method preconditioned by an approximate inverse preconditioner. The goal of the spectral preconditioner is to improve the convergence properties by shifting by one the smallest eigenvalues of the original preconditioned system. Numerical experiments on parallel distributed memory computers are presented to illustrate the efficiency of this technique on large and challenging real–life industrial problems.

Keywords: electromagnetic scattering problems, Frobenius–norm minimization preconditioner, large dense complex linear systems, spectral low rank update preconditioner.

AMS(MOS) subject classifications: 65F05, 65F50.

---

[1]Also appeared as CERFACS Report TR/PA/03/95. Current RAL reports available by anonymous ftp to `ftp.numerical.rl.ac.uk` in directory pub/reports. This report is in file dglmRAL2003023.ps.gz. The report also available through the URL `www.numerical.rl.ac.uk/reports/reports.html`.

[2]i.s.duff@rl.ac.uk. This work was supported by the EPSRC Grant GR/R46441.

[3] CERFACS, 42 Ave G. Coriolis, 31057 Toulouse Cedex, France.

[4]giraud@cerfacs.fr.

[5]langou@cs.utk.edu. Now postdoc at University of Tennessee.

[6]emartin@cerfacs.fr.

Computational Science and Engineering Department
Atlas Centre
Rutherford Appleton Laboratory
Oxon OX11 0QX

28th November 2003

Contents

1 Introduction

In electromagnetic calculations, a classic problem is to compute the currents generated on the surface of an object illuminated by an given incident plane wave. Such calculations, relying on the Maxwell's equations, are required in the simulation of many industrial processes coming from antenna design, electromagnetic compatibility, computation of back-scattered fields, and so on. Recently the Boundary Element Method (BEM) has been successfully used in the numerical solution of this class of problems. The formulation considered in this paper is the EFIE (Electric Field Integral Equation) as it is the most general and does not require any assumption about the geometry of the object. The matrices associated with the resulting linear systems are large dense, non–Hermitian and complex. With the advent of parallel processing, this approach has become viable for large problems and the typical problem size in the electromagnetics industry is continually increasing. Nevertheless, nowadays, problems with a few hundred thousand variables can no longer be solved by direct solvers even if they are parallel and out–of–core because they require too much memory, CPU and disk resources. Iterative solvers appear as the only viable alternative since techniques based on multipole expansion (Darve 2000*a*, Darve 2000*b*) have been developed to perform fast matrix–vector products without forming all the entries of the dense matrices. In particular, the fast multipole method (FMM) performs the matrix–vector product in $\mathcal{O}(n \log n)$ floating–point operations and can efficiently be implemented on parallel distributed platforms with some out–of–core techniques in order to tackle huge industrial problems (Sylvand 2002). The industrial problem we focus on in this paper is the monostatic radar cross section calculation of an object. The procedure consists in considering a set of waves with the same wavelength but different incident angles that illuminate the object. For each of these waves we compute the electromagnetic field backscattered in the direction of the incident wave. This requires the solution of one linear system per incident wave. For a complete radar cross section calculation, from a few tens up to a few hundred waves have to be considered. We have then to solve a sequence of linear systems having the same coefficient matrix but different right-hand sides. The problem can be written

$$A(x_1, \ldots, x_p) = (b_1, \ldots, b_p).$$

In this context it is particularly important to have a numerically efficient and easily parallelizable preconditioner. A preconditioner suitable for implementation in a multipole framework on parallel distributed platforms has been proposed by Alléon, Benzi and Giraud (1997) and Carpentieri (2002). It is based on a sparse approximate inverse using a Frobenius norm minimization with an a priori sparsity pattern selection strategy. It has been shown (Carpentieri, Duff and Giraud 2000, Carpentieri, Duff, Giraud and Magolu monga Made 2001) that on medium size problems this technique gives rise to an effective preconditioner that

outperforms more classical approaches like incomplete factorizations or other general purpose approximate inverses (Benzi, Meyer and Tůma 1996, Benzi and Tůma 1998, Grote and Huckle 1997). However, for large problems the preconditioner becomes sparser and sparser when the problem size increases and eventually performs poorly. Unfortunately, making it denser is not feasible due to memory and disk constraints (Carpentieri, Duff, Giraud and Sylvand 2003*b*). In this paper, we investigate the use of a spectral low rank update (Carpentieri, Duff and Giraud 2003*a*) that attempts to shift by one the smallest eigenvalues of the preconditioned systems resulting in a faster convergence rate. The paper is organized as follows. In Section 2, we recall the main features of the spectral low rank update preconditioner and describe the real-life test problems we consider for the numerical experiments. Then, in Section 3, we illustrate the effect of the size of the update on the convergence rate of GMRES (Saad and Schultz 1986) for one right-hand side and show the gain for a complete radar cross section in the following section. Since the spectral low rank update compensates for some possible weaknesses of the approximate inverse, we illustrate in Section 5 that a balance has to be found between the two components of the resulting preconditioner. In Section 6, we show that the gain induced by the low rank update becomes larger as the size of the restart in GMRES decreases. Finally, since many right-hand sides have to be solved, we illustrate in Section 7 the benefit of using the preconditioner in the context of seed GMRES that is a Krylov solver designed to deal with multiple right-hand sides. We conclude with some conclusions and comments on future work in Section 8.

2 Background and test examples

For the solution of large linear systems using fast multipole techniques on parallel distributed memory computers, we consider an approximate inverse preconditioner based on a Frobenius norm minimization procedure and we denote it by M_{Frob}. Information from the underlying physical problem is exploited to prescribe in advance the sparsity pattern of the preconditioner (Alléon et al. 1997, Carpentieri et al. 2000). We note that its density in terms of number of nonzero entries per column can be adjusted through a simple parameter. For further details on its implementation in the fast multipole code as well as its behaviour on large problems we refer to Carpentieri et al. (2003*b*) and Sylvand (2002). In this paper, we investigate the behaviour of the low rank update preconditioner described by Carpentieri et al. (2003*a*) that attempts to improve the convergence rate of the Krylov solver by shifting by one the smallest eigenvalues that M_{Frob} leaves close to the origin. On these linear systems, GMRES has shown itself to be the most robust Krylov solver (Carpentieri et al. 2003*b*, Sylvand 2002). We note that other related approaches that attempt to remove the effect of the smallest eigenvalues are available in the literature. In the context of adaptive preconditioners for restarted

GMRES, we refer the reader to Baglama, Calvetti, Golub and Reichel (1999), Erhel, Burrage and Pohl (1996), and Kharchenko and Yeremin (1995).

Let us describe the spectral low rank update for the right preconditioner that we use later in our experiments. Given the right preconditioned linear system:

$$AM_{Frob}u = b \text{ with } x = M_{Frob}u,$$

where the matrix A is $n \times n$ complex nonsingular and M_{Frob} is the right preconditioner, the spectral low rank update is as follows. For the sake of simplicity, we assume that the preconditioned matrix AM_{Frob} is diagonalizable. That is,

$$AM_{Frob} = V\Lambda V^{-1},$$

with Λ the diagonal matrix formed by the eigenvalues $\{\lambda_i\}_{i\in\{1,n\}}$ ordered by increasing magnitude, and V the matrix whose columns correspond to the right eigenvectors. Let us consider the k smallest eigenvalues of AM_{Frob} denoted by λ_i with $i = 1, ..., k$. We denote by V_k the set of the right eigenvectors associated with these k smallest eigenvalues. In Carpentieri et al. (2003*a*), it is shown that, if W is a m by k matrix such that $A_0 = W^H AM_{Frob}V_k$ has full rank, then setting

$$M_0 = M_{Frob}V_k A_0^{-1} W^H \text{ and } M_{SLRU(k)} = M_{Frob} + M_0,$$

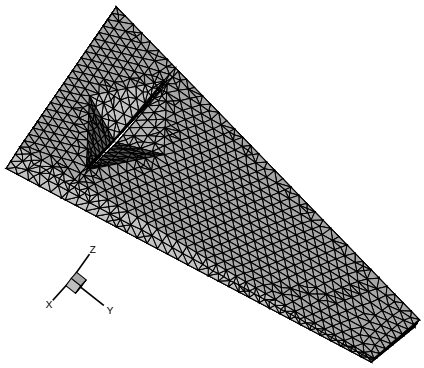
we have that $AM_{SLRU(k)}$ is similar to a matrix whose eigenvalues are:

$$\begin{cases} \eta_i = \lambda_i & \text{if} \quad i > k, \\ \eta_i = 1 + \lambda_i & \text{if} \quad i \leq k. \end{cases} \tag{2.1}$$

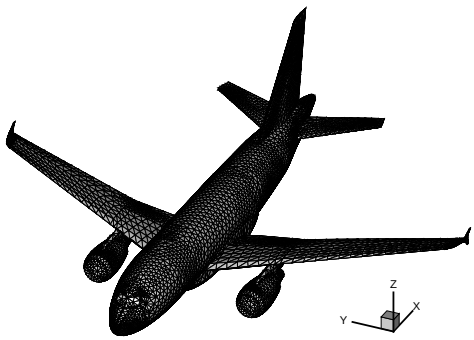
The matrix M_0 is a rank-k correction of M_{Frob}, which ensures that the new system:

$$AM_{SLRU(k)}u = b \text{ with } x = M_{SLRU(k)}u.$$

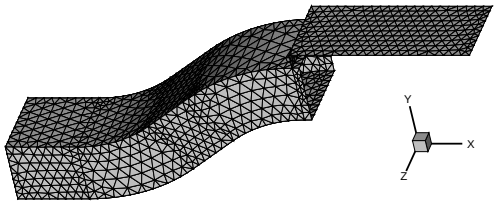
no longer has eigenvalues with magnitude smaller than $|\lambda_{k+1}|$. Note that we can choose other shifts in other contexts. In our case, as most of eigenvalues are already close to one, this shift makes sense. Furthermore, we set $W = V_k$ in our experiments. In Carpentieri et al. (2003*a*), some promising results in the context of BEM are shown on small examples with a few hundred unknowns; the spectral low rank update is combined with the GMRES method preconditioned by a Frobenius norm minimizer preconditioner. In this work, we investigate the same solver using the same combination of preconditioners but on large real industrial applications. The test geometries are shown in Figure 2.1. They consist of a wing with a hole referred to as Cetaf, an Airbus aircraft, an air intake referred to as Cobra, and finally an Almond. The Cetaf and Almond cases are classic test problems in the computational electromagnetics community; the other two have been kindly provided to us by EADS–CCR. Given these four geometries, the sets of angles of interest for the monostatic radar cross section vary. In Table 2.1 we indicate, for each geometry, the number of unknowns associated with each linear system, the density of M_{Frob}, the



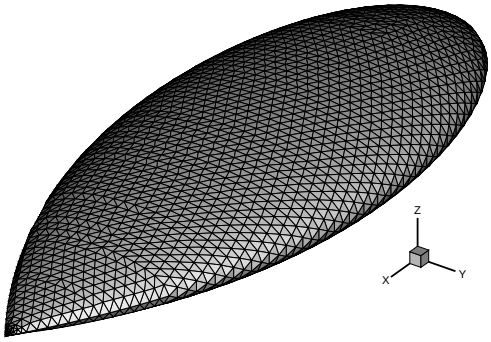
(a) Cetaf



(b) Airbus



(c) Cobra



(d) Almond

Figure 2.1: Various geometries used in the numerical experiments

Geometry	# unknowns	density	θ	φ	# rhs
Cetaf	5 391	3.3 %	(-90) - 90	0	181
Airbus	23 676	0.94 %	90	0 - 180	181
Cobra	60 695	0.24 %	0 - 90	0	91
Almond	104 793	0.19 %	90	0 - 180	181

Table 2.1: Angular section of interest for each geometry; the discretization step is one degree.

angular section considered (given in spherical coordinates, see Figure 2.2) as well as # rhs, the number of right-hand sides to be solved for the complete monostatic calculation. The numerical experiments reported in this paper have been performed on a Compaq Alpha (EV 68, 1.3 Gflops peak) server that is a cluster of symmetric

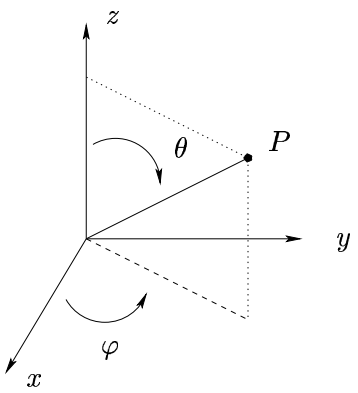


Figure 2.2: Spherical coordinates

multiprocessors. The stopping criterion consists in reducing the original residual by 10^{-3} that can then be related to a norm-wise backward error. Although this value may appear to be large, it is sufficient to obtain accurate radar cross section results. Finally, the initial guess is set to the zero vector.

3 Efficiency of the preconditioner with respect to the rank of the update

In Figure 3.1, we plot using the symbol "x" the spectrum of M_{Frob}, the matrix preconditioned with the Frobenius preconditioner, for the Cetaf case. As can be observed, M_{Frob} succeeds in clustering most of the eigenvalues around $(1.0, 0.0)$. Such a distribution is highly desirable to get fast convergence of Krylov solvers. Nevertheless the remaining eigenvalues nearest to zero can potentially slow down the convergence. Using the symbol "o" we plot, in Figure 3.1, the spectrum of the matrix preconditioned with $M_{SLRU(20)}$. We observe that the 20 smallest eigenvalues of the matrix AM_{Frob} have been shifted close to one, in agreement with (2.1). Consequently, we expect the Krylov solver to perform better with $M_{SLRU(20)}$ than with M_{Frob}. In Figure 3.2, we plot the convergence histories obtained by varying the size of the low rank update. It can be observed that the larger the rank, the faster the convergence of full GMRES. However, in going from 15 to 20 the gain is negligible and going beyond 20 does not give further improvements.

As we mentioned earlier, we have used this technique for monostatic calculation. As many linear systems with the same coefficient matrix but different right-hand sides have to be solved, some are easier to solve than others. In Figure 3.3, we illustrate an important feature of the low rank update by showing the number of iterations for convergence for a difficult right–hand side and an easy one. It can be observed that, when the number of shifted eigenvalues increases, the number of iterations to reach convergence decreases. Furthermore, there is not much difference

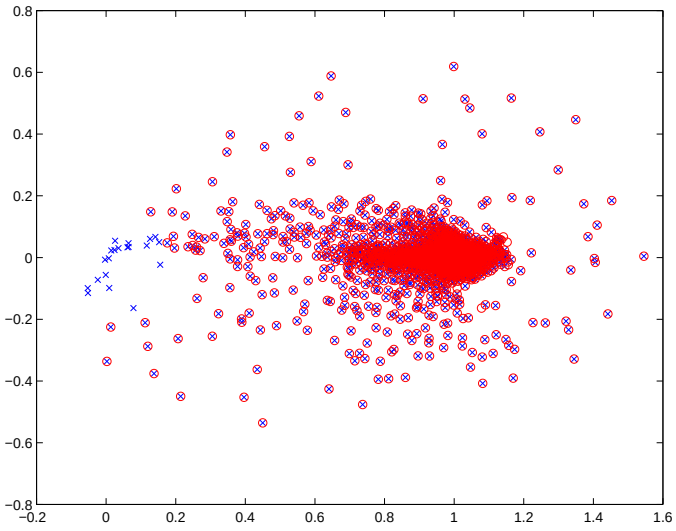


Figure 3.1: Spectrum of AM_{Frob} denoted by x and $AM_{SLRU(20)}$ depicted with o on the Cetaf test problem.

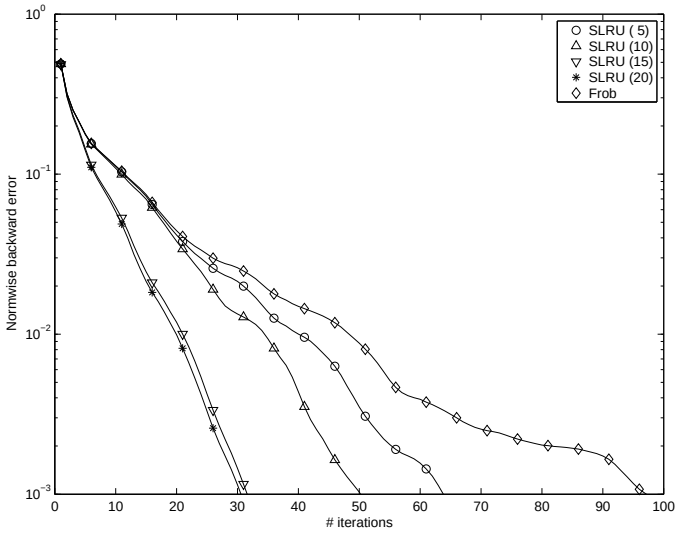


Figure 3.2: Convergence history when varying the rank of the update on the Cetaf example.

between a difficult right–hand side and an easy one when the rank of the update increases. Later in this paper, we illustrate the advantage of this feature in the context of restarted GMRES and seed GMRES.

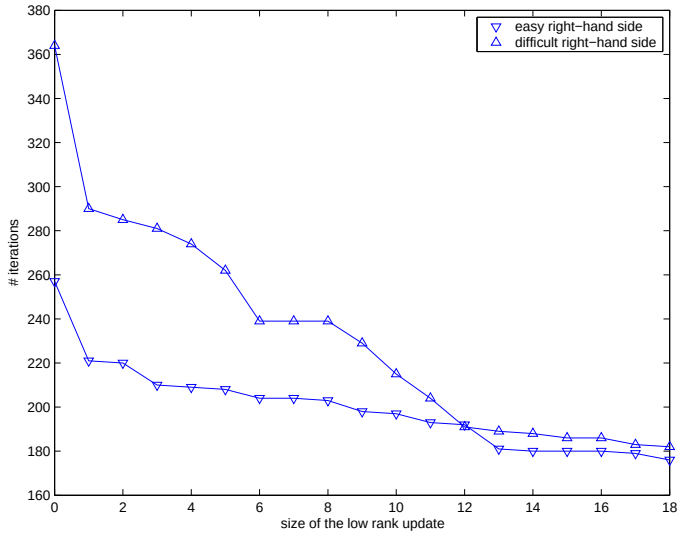


Figure 3.3: Number of full GMRES iterations when varying k in $M_{SLRU(k)}$ on the Cobra test problem.

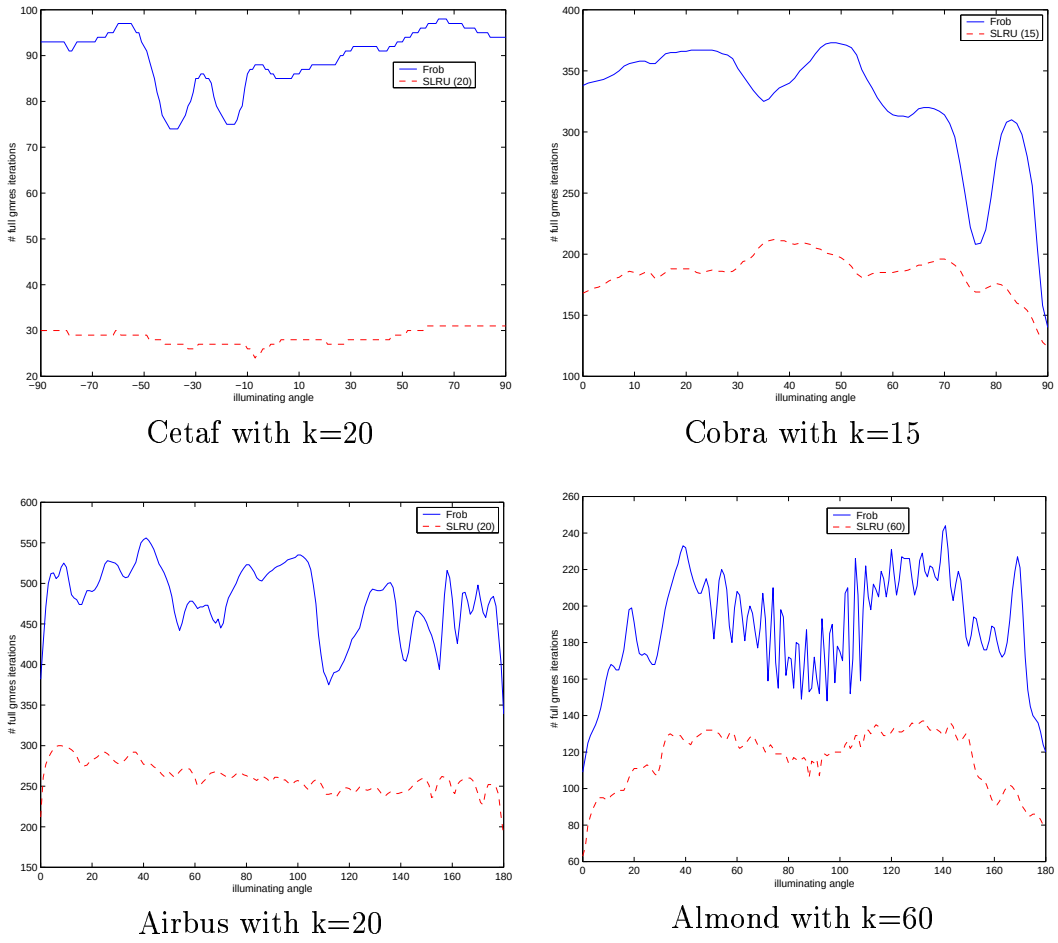
4 Results on a complete monostatic calculation

The use of a preconditioner is often beneficial, however its usefulness depends not only on its effect on convergence but also its construction time and the time spent in applying it at each step. In Table 4.1, we give the construction time for the FMM operator, for M_{Frob} and for $M_{SLRU(k)}$. In our experiments, the eigenvalue calculation is performed in a preprocessing phase using ARPACK (Lehoucq, Sorensen and Yang 1998), that represents the main part of the time required to setup $M_{SLRU(k)}$. In that table, we also display the average time spent in one FMM operation, one product by M_{Frob}, and one application of $M_{SLRU(k)}$. We expect to reduce the overall number of iterations, but each new iteration is more expensive. One application of $M_{SLRU(k)}$ compared with one application of M_{Frob} introduces around $4.k.n$ additional flops, where k is the chosen number of eigenvalues and n the size of the problem. On our test examples the extra cost per iteration in elapsed time ranges from 6% (Cobra case) to 35% (Almond case), but it remains small in comparison to the FMM application times.

In Figure 4.1, we show the number of full GMRES iterations for each right-hand side using either M_{Frob} (solid line) or $M_{SLRU(k)}$ (dashed line). For each geometry the value of k is given in the caption. While without the spectral preconditioner, the numbers of iterations from one right–hand side to another vary a lot, with the spectral preconditioner the number of iterations per right–hand side is more similar and almost constant for some geometries. This behaviour was already observed in Figure 3.3.

			Construction Times			Application Times		
geometry	# proc.	k	FMM	M_{Frob}	$M_{SLRU(k)}$	FMM	M_{Frob}	$M_{SLRU(k)}$
Cetaf	8	20	13 s	25 s	45 s	0.208 s	0.035 s	0.041 s
Airbus	32	20	27 s	51 s	19 m	0.830 s	0.118 s	0.147 s
Cobra	32	15	36 s	73 s	28 m	1.259 s	0.110 s	0.117 s
Almond	32	60	59 s	3 m	2 h	1.914 s	0.138 s	0.187 s

Table 4.1: Average elapsed time per matrix-vector product.



Cetaf with k=20 Cobra with k=15

Airbus with k=20 Almond with k=60

Figure 4.1: Number of full GMRES iterations with M_{Frob} and $M_{SLRU(k)}$ for the different incident angles for each geometry.

Table 4.2 summarizes the cumulated numbers of matrix–vector products and the total elapsed solution time for a complete radar cross section calculation for each geometry using full GMRES. For all geometries except the Cetaf, 181 linear systems are solved; only 91 are considered for the Cetaf test problem. Depending on the geometry, the overall gain ranges from a factor of 1.6 to 3 for both CPU time and

		M_{Frob}		$M_{SLRU(k)}$	
Geometry	# procs.	# iter	times	# iter	times
Cetaf	8	16 391	1 h 40 m	5 349	47 m
Airbus	32	87 121	46 h	47 385	18h 40 m
Cobra	32	29 777	21 h	16 921	8h 30 m
Almond	32	34 375	25 h 30 m	21 273	14h 40 m

Table 4.2: Cost for a complete monostatic calculation.

total number of GMRES iterations. It should be pointed out that this could be improved on some examples if more eigenvalues were shifted. Our purpose in these experiments is to illustrate the potential of $M_{SLRU(k)}$, and we did not try to find the best values of k for each geometry. For example, by shifting 10 more small eigenvalues for the Cobra case, we move from a factor 1.6 to a factor 3.

The extra cost for computing the eigenspace during the preprocessing phase in terms of matrix–vector products by AM_{Frob} as well as the corresponding elapsed time is displayed in Table 4.3. In that table, we also give the number of right-hand sides in the monostatic calculation, denoted # rhs, for which the gain introduced by $M_{SLRU(k)}$ compensates for the cost of computing the preconditioner. It can be seen that the preprocessing calculation is quickly amortized when a few right-hand sides need to be solved.

Geometry	# procs.	# mat-vec	times	# rhs
Cetaf	8	170	45 s	3
Airbus	32	1 000	19 m	6
Cobra	32	1 000	28 m	6
Almond	32	2 200	2 h	32

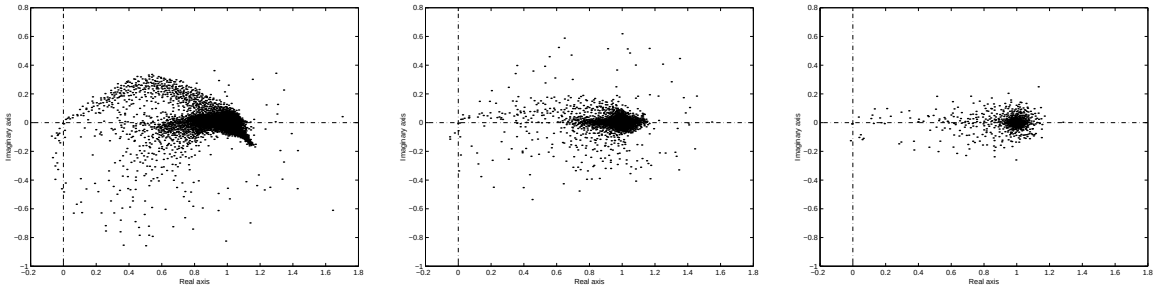
Table 4.3: Cost of the eigencomputation preprocessing phase.

In addition, we should mention that the quality of the eigenvectors in terms of backward error does not needs to be very accurate: using eigenvectors with a backward error of less than 10^{-3} does not significantly improve convergence if at all.

5 Balancing the two components of the preconditioner

Although we save a significant number of iterations using $M_{SLRU(k)}$, we might ask whether it could be more effective to use a better M_{Frob}, by allowing the preconditioner to have more nonzero entries, combined with a migration of only

some smallest eigenvalues, or a worse M_{Frob}, but combined with a migration of many smallest eigenvalues. To deal with this, we first investigate the effect of the quality of M_{Frob} on the calculation time of the smallest eigenvalues of AM_{Frob}. In that respect, we vary the number of nonzeros per column leading to different densities of M_{Frob}. In Figure 5.1, we display the spectrum of AM_{Frob} on the Cetaf example for various values of the density. As we might expect, the denser M_{Frob} is, the better the clustering around one and the fewer eigenvalues close to zero; moreover these are better separated. A consequence for the eigensolver is that the few well separated eigenvalues that are near zero for the largest density of the preconditioner are more easily computed. When the density is relaxed, the eigenvalue cluster close to zero becomes wider and the eigensolver has more difficulty in computing those near zero. Indeed, it needs extra effort to identify the eigenvalues in a cluster. To illustrate this claim we show, in Table 5.1, the number of matrix–vector products and the corresponding elapsed time required by ARPACK to find the eigenvectors associated with the 60 smallest eigenvalues as the density increases. As expected, we observe that the denser the preconditioner is, the easier it is for the eigensolver to find the smallest eigenvalues.



(a) 0.7 % (b) 3.3 % (c) 15.7 %

Figure 5.1: Spectrum of AM_{Frob} for various density of M_{Frob}

density	# mat-vec	times
0.7%	365	477 s
3.3%	240	281 s
15.7%	152	172 s

Table 5.1: Number of matrix–vector products and elapsed times (one processor) required to compute the 60 smallest eigenvalues of AM_{Frob} for the Cetaf when the density of M_{Frob} is varied.

A question that seems natural to raise is: how many eigenvalues should be shifted for these three densities of M_{Frob} to get convergence in the same number of

full GMRES iterations ? On the Cetaf for a difficult angle and a density equal to 3.3 %, full GMRES needs 98 iterations to converge. Using $M_{SLRU(20)}$ GMRES needs 31 iterations to converge. The number 31 is taken as a reference value to compare the performance with the three densities. Table 5.2 shows the cost for ARPACK to compute the corresponding number of eigenvalues for each density and the total cost of computing the preconditioner.

To obtain the same number of full GMRES iterations, we need to shift more and more eigenvalues as we decrease the density. On the other hand, the construction cost of M_{Frob} with a low density is cheaper than with a higher density. There is a trade-off to be found between a cheap M_{Frob} that requires shifting many eigenvalues that might be difficult to compute, and a more expensive M_{Frob} where only a few eigenvalues need to be shifted to get a similar convergence behaviour. As Table 5.2 shows, the medium density 3.3% offers the best trade-off among the three densities considered. The preconditioner already works well and only a few eigenvalues need to be shifted. In each case shown in Table 5.2, 31 iterations of preconditioned full GMRES are needed for convergence. The symbol "k" is the number of eigenvalues computed by ARPACK.

M_{Frob} construction		eigensolver		
density	times	k	times	Total
0.7%	41 s	54	469 s	510 s
3.3%	143 s	20	161 s	304 s
15.7%	1114 s	2	64 s	1178 s

Table 5.2: Construction times when varying the M_{Frob} density on the Cetaf test problem.

Let us illustrate, on another example, the advantage of using $M_{SLRU(k)}$ rather than increasing the density of M_{Frob}. We consider now the Almond test problem, that is the biggest, using two densities for M_{Frob}. The targeted number of iterations of full GMRES is 157 that is obtained with a density of 0.19% and by shifting 30 eigenvalues using $M_{SLRU(30)}$. To get the same number of iterations without shifting eigenvalues, we need to increase the density of M_{Frob} to 1.76 %. In Table 5.3, we show the computation cost for both preconditioners. It can be seen that the eigencalculation in the preprocessing phase of $M_{SLRU(30)}$ combined with the low cost of its M_{Frob} component is significantly less expensive than the denser M_{Frob} that exhibits the same convergence property. We compare the application times for these two approaches and the time to obtain solution. The second approach is four times as expensive needing 1.21 s per application as opposed to the first with 0.38 s. The use of M_{Frob} with a density of 0.19% in the first case would have cost 0.36 s; it means that $M_{SLRU(30)}$ yields an extra cost of only 0.02 s per application. Moreover, it gives a smaller solution time and a much cheaper setup time. Setup time appears

to be too much important towards solution time, but these results are obtained on just one angle. For solutions on an angular section, we will pay this Setup time for just one time.

Setup					Solution		
M_{Frob}		Eigencalculation		Total	Application Times		Total
Density	Times	# Eigen.	Times	Times	M_{Frob}	$M_{SLRU(30)}$	Times
0.19%	6 m	30	3 h	3 h 06 m	-	0.38 s	510 s
1.76%	5 h 40 m	0	-	5 h 40 m	1.21 s	-	648 s

Table 5.3: Comparison of a denser M_{Frob} with a sparser $M_{SLRU(30)}$ on 8 processors on the Almond test example to obtain 157 iterations with full GMRES.

6 Sensitivity of the restarted GMRES

All the numerical experiments reported on so far have been obtained with full GMRES. In this section we will investigate the effect of restarted GMRES on the efficiency of the preconditioners. For each value of the restart, we show in Table 6.1 the number of GMRES iterations of M_{Frob} and $M_{SLRU(k)}$ on a easy and a hard right-hand side. The symbol "-" means that convergence is not obtained within 5000 iterations. As can be seen, the smaller the restart is, the larger the improvement with $M_{SLRU(k)}$. With $M_{SLRU(20)}$ on the Cetaf example, we see that GMRES converges quickly and that GMRES(10) behaves very similarly to GMRES(∞). This observation is no longer true for the other geometries. It might depend on the number of eigenvalues that are computed: the choice is the best for the Cetaf example but not for the other geometries.

Easy case								
Restart	Cetaf		Airbus		Cobra		Almond	
	M_{Frob}	$M_{SLRU(20)}$	M_{Frob}	$M_{SLRU(20)}$	M_{Frob}	$M_{SLRU(15)}$	M_{Frob}	$M_{SLRU(60)}$
10	271	30	-	639	514	378	492	95
30	128	27	-	439	280	196	149	67
50	100	27	-	390	264	188	118	68
∞	74	27	421	242	222	172	109	63
Difficult case								
Restart	Cetaf		Airbus		Cobra		Almond	
	M_{Frob}	$M_{SLRU(20)}$	M_{Frob}	$M_{SLRU(20)}$	M_{Frob}	$M_{SLRU(15)}$	M_{Frob}	$M_{SLRU(60)}$
10	669	37	-	1200	2624	481	-	1497
30	275	31	-	688	1031	232	429	163
50	197	31	-	608	760	207	334	144
∞	98	31	490	283	367	187	232	126

Table 6.1: Sensitivity to the GMRES restart parameter.

The $M_{SLRU(k)}$ preconditioner allows us to use a smaller restart than usual; this might be a significant asset on very large problems where using a large restart becomes a severe bottleneck because of the prohibitive memory requirements. There are some problems where M_{Frob} does not converge, for instance on the Airbus case with a restart between 10 and 50, or on the Almond example with a restart of 10, while $M_{SLRU(k)}$ does converge in a reasonable number of iterations. On that later example, as we see in Figure 6.1, the convergence rate of GMRES increases with the restart whatever the chosen preconditioner. Furthermore, the slope of the convergence history for $M_{SLRU(k)}$ becomes quickly comparable to that for full GMRES, while for M_{Frob} this phenomenon takes more time to appear. Although not reported here, the same behaviour was observed on the other geometries.

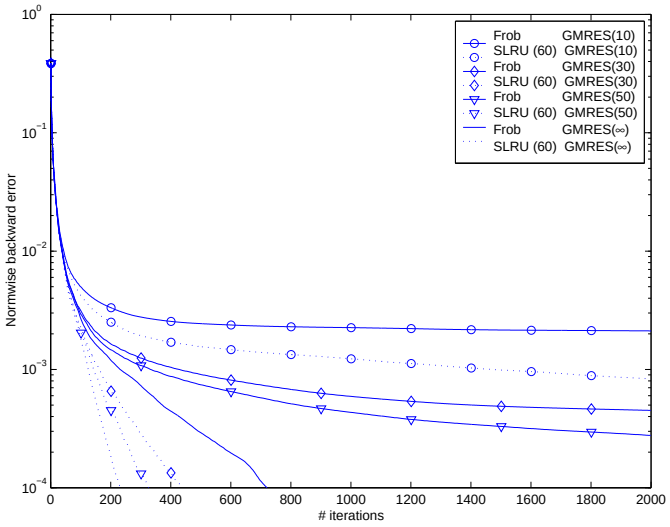


Figure 6.1: Convergence history varying the restart for a "difficult" angle of the Almond test problem.

7 Complementarity of $M_{SLRU(k)}$ and the seed GMRES algorithm

As mentioned already for a complete monostatic calculation, several linear systems with the same coefficient matrix but different right-hand sides have to be solved. In that framework, it is crucial not only to use an efficient preconditioner but also a suitable Krylov solver. There are basically two classes of techniques designed for this purpose. These are the seed techniques and the block approaches (Freund and Malhotra 1997, O'Leary 1980, Vital 1990). In this section, we investigate the benefit of using $M_{SLRU(k)}$ in the context of seed GMRES.

For the sake of simplicity of the notation, we describe below the method for the solution of only two right-hand sides. The problem can be written:

$$\begin{cases} A(x_1, x_2) = (b_1, b_2), \text{ with} \\ \quad (x_1^0, x_2^0) \text{ as the initial guesses.} \end{cases}$$

The seed GMRES method starts by solving the first system $Ax_1 = b_1$ using the GMRES algorithm. Once that is converged, it computes an orthonormal basis of the Krylov space (V_{m+1}^1) and the upper-Hessenberg matrix ($\bar{H}_m^1$) such that $AV_m^1 = V_{m+1}^1 \bar{H}_m^1$. The idea consists in updating the initial guess x_2^0, by minimizing the norm of the residual $r_2^0 = b_2 - Ax_2^0$ of the second system on the basis V_{m+1}^1. We search a new initial guess $\hat{x}_2^0$ such as:

$$\begin{cases} \hat{x}_2^0 = x_2^0 + V_m^1 y_2 \\ y_2 = \arg\min_y ||b_2 - A(x_2^0 + V_m^1 y)||_2 \end{cases}$$

Using the fact that:

$$\begin{aligned} ||b_2 - A(x_2^0 + V_m^1 y)||_2^2 &= ||r_2^0 - AV_m^1 y||_2^2, \\ &= ||(I_n - V_{m+1}^1 (V_{m+1}^1)^T) r_2^0 + V_{m+1}^1 (V_{m+1}^1)^T r_2^0 - AV_m^1 y||_2^2, \\ &= ||(I_n - V_{m+1}^1 (V_{m+1}^1)^T) r_2^0 + V_{m+1}^1 ((V_{m+1}^1)^T r_2^0 - \bar{H}_m^1 y)||_2^2, \\ &= ||(I_n - V_{m+1}^1 (V_{m+1}^1)^T) r_2^0||_2^2 + ||V_{m+1}^1 ((V_{m+1}^1)^T r_2^0 - \bar{H}_m^1 y)||_2^2, \\ &= ||(I_n - V_{m+1}^1 (V_{m+1}^1)^T) r_2^0||_2^2 + ||(V_{m+1}^1)^T r_2^0 - \bar{H}_m^1 y||_2^2, \end{aligned}$$

we finally should solve the least-squares problem: $\arg\min_y ||(V_{m+1}^1)^T r_2^0 - \bar{H}_m^1 y||_2$, where we already have a QR factorization of $\bar{H}_m^1$ given by the GMRES algorithm. Once the new initial guess is computed, we simply run a new GMRES method for the second-right hand side. When more than two right-hand sides are considered, the same algorithm can be applied but we have to solve a least-squares problem for each previous right-hand side after the first.

In order to illustrate the advantage of using $M_{SLRU(k)}$ in this context, we consider eleven right–hand sides: $\phi = 15^o : 1^o : 25^o$ for $\theta = 90^o$, involved in the radar cross section calculation of the Airbus. On these right–hand sides, seed GMRES combined with M_{Frob} behaves rather poorly. To illustrate this, we first compare the numerical behaviour of GMRES with three strategies for defining the initial guess: first using the zero vector, second using the solution of the previous linear system, and finally the initial guess computed by the seed technique. In Table 7.1, we display the number of iterations for each right–hand side for the three initial guess strategies. In the case of zero as initial guess, the initial backward error $\|r_0\|_2/\|b\|_2 = \|b - Ax_0\|_2/\|b\|_2$ is equal to one. In Table 7.1, we see that the seed GMRES method does a good job of decreasing the initial residual norm; it is always by far the smallest. Unfortunately, starting from the seed initial guess that is the closest (in the backward error sense) to the solution does not guarantee fast convergence. That

M_{Frob} preconditioner					
	zero guess	simple strat.		seed GMRES	
(θ, φ)	# iter	# iter	$\|r_0\|_2/\|b\|_2$	# iter	$\|r_0\|_2/\|b\|_2$
(90,15)	474	474	1	474	1
(90,16)	474	443	0.284	440	0.13
(90,17)	483	435	0.298	338	0.05
(90,18)	491	442	0.311	387	0.023
(90,19)	491	438	0.325	458	0.008
(90,20)	490	438	0.338	543	0.004
(90,21)	492	436	0.352	605	0.003
(90,22)	497	418	0.365	599	0.003
(90,23)	499	424	0.379	589	0.003
(90,24)	499	431	0.392	601	0.003
(90,25)	499	406	0.406	573	0.003
# iterations	5389	4785		5607	
elapsed time (s)	17400 s	15438 s		28804 s	

Table 7.1: Number of iterations per right–hand side using three strategies for the initial guess on the Airbus example with M_{Frob} preconditioner on 8 processors.

is, from that good initial guess, GMRES performs rather poorly. It performs only slightly better than starting from zero and is outperformed by the approach that starts from the initial guess provided by the simple strategy of using the solution to the previous system. In other words and surprisingly enough, the approach that gives the smallest initial residual norm is not the method that gives the smallest number of iterations.

We have observed this behaviour of the seed GMRES method on some other difficult problems. Intuitively, it seems to us that an analogy exists between this behaviour and the observed stagnation of the classical restarted GMRES method. In the two cases, an initial guess is extracted from a Krylov space to generate a new Krylov space. As illustrated in the previous section, one possible remedy for the restarted GMRES method is to replace M_{Frob} by $M_{SLRU(k)}$.

In Table 7.2, we investigate this possibility. We keep the same strategies as in Table 7.1 but now use $M_{SLRU(20)}$ rather than M_{Frob}. The initial guess computed by the seed GMRES method becomes the best performing strategy. In Figure 7.1, for each right–hand side, $\phi = 15^o : 1^o : 25^o$, we plot the convergence history of the seed GMRES method with the two preconditioners. It can be seen that although the norm of the initial residuals are about the same for the two preconditioners, the rate of convergence is significantly improved by $M_{SLRU(20)}$. The seed GMRES method provides a small initial residual and $M_{SLRU(k)}$ ensures a fast rate of convergence of

GMRES iterations: in a race starting close to the finish (seed strategy) and running fast ($M_{SLRU(k)}$ preconditioner) this ensures we finish first!

$M_{SLRU(20)}$ preconditioner					
	zero guess	simple strat.		seed GMRES	
(θ, φ)	# iter	# iter	$\|r_0\|_2/\|b\|_2$	# iter	$\|r_0\|_2/\|b\|_2$
(90,15)	280	280	1	280	1
(90,16)	275	198	0.284	201	0.14
(90,17)	275	188	0.298	145	0.052
(90,18)	276	190	0.311	167	0.025
(90,19)	280	198	0.325	165	0.009
(90,20)	283	205	0.338	171	0.006
(90,21)	284	208	0.352	171	0.005
(90,22)	286	212	0.365	170	0.005
(90,23)	289	214	0.379	172	0.005
(90,24)	291	218	0.392	176	0.005
(90,25)	292	219	0.406	171	0.005
# iterations	3111	2330		1989	
elapsed time (s)	8348 s	6252 s		5518 s	

Table 7.2: Number of iterations per right–hand side using three strategies for the initial guess on the Airbus example with $M_{SLRU(20)}$ preconditioner on 8 processors.

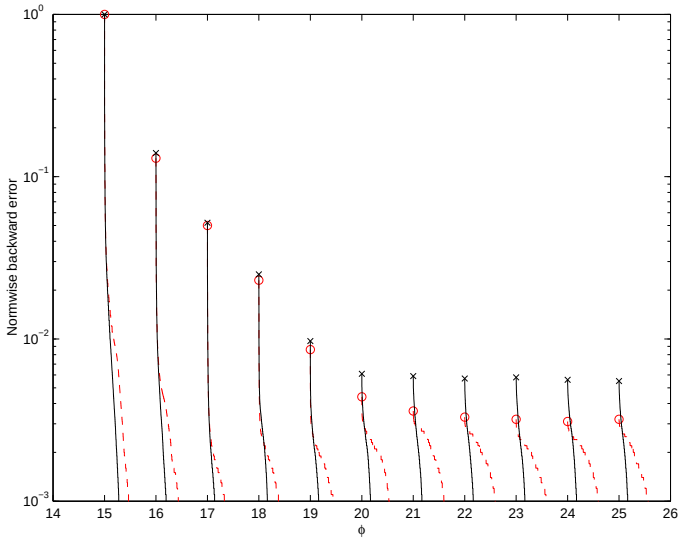


Figure 7.1: Convergence history of seed GMRES for the right-hand sides associated with $\phi = 15^o : 1^o : 25^o$ using M_{Frob} (○, in dashed line) and $M_{SLRU(20)}$ (×, in solid line) on the Airbus test problem.

8 Conclusions and prospectives

In this paper, we have shown that the $M_{SLRU(k)}$ preconditioner based on the M_{Frob} preconditioner is effective for solving large linear systems arising in challenging real-life electromagnetics applications. It is important to point out that, to be effective, the spectral low rank update should be built on top of a good enough preconditioner that already succeeds in clustering most of the eigenvalues close to a point far from the origin (one in our case). There are two main reasons that motivate this observation. Firstly, if only a few eigenvalues are left close to the origin, a small rank update will be sufficient to significantly improve the convergence. Secondly, these few isolated eigenvalues will be easily found by an eigensolver. Of course a trade-off between the two components of $M_{SLRU(k)}$ should be found as the low rank update might be unnecessary if M_{Frob} is very dense, or it might be too expensive to improve a poor M_{Frob} because too many eigenvalues, potentially difficult to compute, have to be shifted.

We observe that the convergence of GMRES using $M_{SLRU(k)}$ only weakly depends on the choice of the initial guess. This is particularly useful in the seed GMRES or restarted GMRES contexts.

When several right-hand sides have to be solved with the same coefficient matrix, the extra cost of the eigencalculation preprocessing phase is quickly amortized by the reduction in the number of iterations as the extra cost per iteration is negligible. However, the cost of the preprocessing phase can be decreased in different ways. The first approach would consist in constructing the preconditioner as we solve different right-hand sides by extracting the spectral information from previous GMRES solutions. Another possibility, as the accuracy of the eigenvectors in terms of backward error does not need to be very good, would be to still compute them in a preprocessing phase but with a less accurate FMM. Implementing this idea for the Cobra example leads to the plot reported in Figure 8.1. In that figure, it can be seen that using either a high or the medium accuracy FMM leads to the same quality of the $M_{SLRU(k)}$ preconditioner, while using a low accuracy only deteriorates the efficiency by less than 20% in terms of iteration number. Consequently, using a low accurate FMM for the eigencalculation might also be a way of reducing the cost for the preprocessing phase.

A question that remains open is the a priori identification of the optimal number of eigenvalues to be shifted. We have seen that increasing this number is always beneficial, but the relative gain tends to vanish when this number becomes large. If the eigenvalue information was extracted at run-time, a possible strategy might be to increase the size of the rank from one right-hand side to the next as long as an improvement is observed. Further investigations deserve to be undertaken to study this possibility in the framework of the monostatic radar cross section calculations.

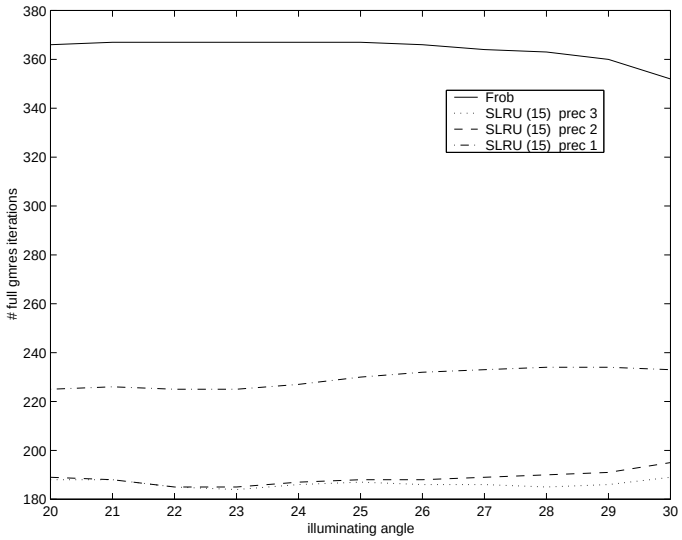


Figure 8.1: Influence of the multipole precision (prec) in ARPACK on the iterations on a difficult range for the Cobra test problems; the most accurate precision is prec 3.

Acknowledgements

We would like to thank Guillaume Alléon from EADS-CCR, for providing us with some test examples considered in this paper. We wish to express our gratitude to Guillaume Sylvand from CERMICS-INRIA, for his friendly and permanent support and his advice in using the FMM components of the code.

References

Alléon, G., Benzi, M. and Giraud, L. (1997), 'Sparse approximate inverse preconditioning for dense linear systems arising in computational electromagnetics', *Numerical Algorithms* **16**(1), 1–15.

Baglama, J., Calvetti, D., Golub, G. H. and Reichel, L. (1999), 'Adaptively preconditioned GMRES algorithms', *SIAM J. Scientific Computing* **20**(1), 243–269.

Benzi, M. and Tůma, M. (1998), 'Numerical experiments with two sparse approximate inverse preconditioners', *BIT* **38**, 234–241.

Benzi, M., Meyer, C. D. and Tůma, M. (1996), 'A sparse approximate inverse preconditioner for the conjugate gradient method', *SIAM J. Scientific Computing* **17**(5), 1135–1149.

Carpentieri, B. (2002), Sparse preconditioners for dense complex linear systems in electromagnetic applications, Ph.D. dissertation, INPT. TH/PA/02/48.

Carpentieri, B., Duff, I. S. and Giraud, L. (2000), 'Sparse pattern selection strategies for robust Frobenius-norm minimization preconditioners in electromagnetism', *Numerical Linear Algebra with Applications* **7**(7-8), 667–685.

Carpentieri, B., Duff, I. S. and Giraud, L. (2003*a*), 'A class of spectral two-level preconditioners', *SIAM J. Scientific Computing* **25**(2), 749–765. Was Report TR/PA/02/55.

Carpentieri, B., Duff, I. S., Giraud, L. and Magolu monga Made, M. (2001), Sparse symmetric preconditioners for dense linear systems in electromagnetism, Technical Report TR/PA/01/35, CERFACS, Toulouse, France.

Carpentieri, B., Duff, I. S., Giraud, L. and Sylvand, G. (2003*b*), Combining fast multipole techniques and an approximate inverse preconditioner for large parallel electromagnetics calculations., Technical Report TR/PA/03/77, CERFACS, Toulouse, France.

Darve, E. (2000*a*), 'The fast multipole method (i) : Error analysis and asymptotic complexity', *SIAM J. Numerical Analysis* **38**(1), 98–128.

Darve, E. (2000*b*), 'The fast multipole method: Numerical implementation', *J. Comp. Phys.* **160**(1), 195–240.

Erhel, J., Burrage, K. and Pohl, B. (1996), 'Restarted GMRES preconditioned by deflation', *J. Comput. Appl. Math.* **69**, 303–318.

Freund, R. W. and Malhotra, M. (1997), 'A block QMR algorithm for non-Hermitian linear systems with multiple right-hand sides', *Linear Algebra and its Applications* **254**(1–3), 119–157.

Grote, M. J. and Huckle, T. (1997), 'Parallel preconditioning with sparse approximate inverses', *SIAM J. Scientific Computing* **18**(3), 838–853.

Kharchenko, S. A. and Yeremin, A. Y. (1995), 'Eigenvalue translation based preconditioners for the GMRES(k) method', *Numerical Linear Algebra with Applications* **2(1)**, 51–77.

Lehoucq, R. B., Sorensen, D. C. and Yang, C. (1998), *ARPACK User's guide: Solution of large-scale problem with implicitly restarted Arnoldi methods*, SIAM, Philadelphia.

O'Leary, D. P. (1980), 'The block conjugate gradient algorithm and related methods', *Linear Algebra and its Applications* **29**, 293–322.

Saad, Y. and Schultz, M. H. (1986), 'GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems.', *SIAM J. Scientific and Statistical Computing* **7**, 856–869.

Sylvand, G. (2002), Résolution Itérative de Formulation Intégrale pour Helmholtz 3D : Applications de la Méthode Multipôle à des Problèmes de Grande Taille, PhD thesis, Ecole Nationale des Ponts et Chaussées.

Vital, B. (1990), Etude de quelques méthodes de résolution de problèmes linéaires de grande taille sur multiprocesseur, Ph.D. dissertation, Université de Rennes.