Noname manuscript No.
(will be inserted by the editor)

Application of the Jacobi-Davidson method for spectral low-rank preconditioning in computational electromagnetics problems

J. Cerdán · N. Malla · J. Marín · J. Mas

Received: date / Accepted: date

Abstract We consider the numerical solution of linear systems arising from computational electromagnetics applications. For large scale problems the solution is usually obtained iteratively with a Krylov subspace method. It is well known that for ill conditioned problems the convergence of these methods can be very slow or even it may be impossible to obtain a satisfactory solution. To improve the convergence a preconditioner can be used, but in some cases additional strategies are needed. In this work we study the application of spectral low-rank updates (SLRU) to a previously computed sparse approximate inverse preconditioner. The updates are based on the computation of a small subset of the eigenpairs closest to the origin. Thus, the performance of the SLRU technique depends on the method available to compute the eigenpairs of interest. The SLRU method was first used using the IRA's method implemented in ARPACK. In this work we investigate the use of a Jacobi-Davidson method, in particular its JDQR variant . The results of the numerical experiments show that the application of the JDQR method to obtain the spectral low-rank updates can be quite competitive compared with the IRA's method.

Keywords Iterative methods · Preconditioning · Spectral low-rank updates · Jacobi-Davidson · Computational electromagnetics

This work was supported by Spanish Grant MTM 2010-18674

J. Cerdán
Instituto de Matemática Multidisciplinar.Universitat Politècnica de València
E-mail: jcerdan@imm.upv.es

N. Malla

J. Marín
Instituto de Matemática Multidisciplinar.Universitat Politècnica de València
E-mail: jmarinma@imm.upv.es

J. Mas
Instituto de Matemática Multidisciplinar.Universitat Politècnica de València
Tel.: +34-963877000
Fax: +34-963877669
E-mail: jmasm@imm.upv.es

1 Introduction

Computational electromagnetics applications include the computation of the antenna radiation pattern, electromagnetic interference and compatibility studies in aircraft industry and electromagnetic scattering problems as the computation of the radar cross-section of a complex body. A good understanding of these phenomena is crucial to the design of many industrial devices like radars, computer microprocessors and optical fibre systems. All these simulations can be very demanding in terms of computer resources. Therefore new algorithms and the use of high performance computers are required to afford a rigorous numerical solution.

Maxwell's equations [13] are used to model Electromagnetism phenomena. A numerical solution can be computed by solving either the differential formulation of these equations or their integral counterpart. Differential equation methods apply classical discretization techniques, like the finite-element or the finite-difference method [12,19], to obtain very large and sparse systems of linear equations which must be solved. Recently, integral equation solvers have become more popular. The integral equations relate the electric and magnetic fields to the equivalent electric and magnetic currents on the surface of the object. The electric-field integral equation (EFIE) formulation is widely used in industrial simulations since it can handle general geometries in open domains without truncating and formulating any artificial boundary as it is the case with the differential methods. In this work we focus on the solution of the linear systems obtained after applying the Method of Moments to the integral equations, see [11,17]. The coefficient matrix of the system is dense, complex, non-hermitian for EFIE and ill-conditioned. In real industrial applications the system matrix can be so large that it can not be explicitly stored in memory. Indeed, only a sparse approximation of the system matrix corresponding to the near-field interactions is stored and the rest of the entries are computed on the fly whenever they are needed. In this context direct methods are not applicable and preconditioned Krylov iterative methods must be employed. The application of these methods requires the computation of matrix-vector products which requires $\mathscr{O}(n^2)$ operations. This complexity can be improved to $\mathscr{O}(n \log^2 n)$ by applying the Fast Multipole Method [7,9].

The success of an iterative method, specially for ill-conditioned problems, depends on the preconditioner that is applied to the system matrix. In [2,3] the authors show that most general purpose preconditioners fail to produce good converge rates for the iterative method. The best results were obtained with variants of the Sparse Approximate Inverse Preconditioner (SPAI) [10], that computes a sparse approximation to the inverse of the coefficient matrix. Nevertheless, for many applications the number of nonzeros of the SPAI preconditioner necessary to get convergence is quite large. Then, it can be prohibitive in terms of both memory requirements and computational time. To solve these problems, the authors proposed the application of low-rank updates to the preconditioned matrix, such that a set of the smallest eigenvalues in magnitude are shifted to one, which have a beneficial effect on the convergence of Krylov solvers. The method, called spectral low-rank updates (SLRU), depends on the computation of a set of the interior eigenpairs closest to the origin. To this end, the Implicitly Restarted Arnoldi (IRA) method, implemented in the Arnoldi Package (ARPACK) [16], was used [2–6]. The results showed that the SLRU technique can improve considerably the performance, specially when multiple right hand sides have to be solved, as is the case for most of the problems mentioned above. In this work we investigate the use of a Jacobi-Davidson method [20], in particular the JDQR variant proposed in [8]. The results of the numerical experiments show that with the application of the JDQR method the spectral low-rank updates technique can be quite competitive.

The paper is organized as follows. In section 2 we recall the solution of linear systems with preconditioned iterative methods and the fundamentals of the SPAI preconditioner. In section 3 the SLRU technique is described. Then, in section 4 the Jacobi-Davidson method and its variant JDQR are revised. The numerical results are presented in section 5. Finally, the main conclusions are outlined in section 6.

2 Iterative solution of linear systems

Let

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

be a linear system of n equations with n unknowns obtained after applying a boundary element discretization method to the Maxwell's equations. The matrix A is large, dense and non-hermitian with complex entries. An approximate solution of (1) is usually obtained by using a preconditioned iterative Krylov method, as the Generalized Minimal Residual algorithm (GMRES) [18] or the Biconjugate Gradient Stabilized (BiCGSTAB) method [21].

In general, to obtain good convergence rates, or even to converge, these methods are applied to the left preconditioned linear system

$$MAx = Mb \; ,$$

where the matrix M is the preconditioner. Right preconditioning and two-side preconditioning are also possible, see [18]. The matrix M should be chosen in such a way that the preconditioned matrix MA has a smaller condition number and a better spectral distribution than A. Typically, since $MA \approx I_n$ is desired, the eigenvalues should be clustered around 1, but other distributions of the eigenvalues may be satisfactory. If the eigenvalues of MA are clustered away from the origin, one can expect a good performance of the iterative solver. Moreover, the preconditioner should be easily and inexpensively computed and applied. Clearly, both requirements are difficult to fulfill for a general purpose preconditioner and considerable efforts have been made to develop suitable methods which perform well for a wide range of problems. In [2,3] the authors show that most preconditioners, both direct and approximate inverse preconditioners type, fail to converge in electromagnetics applications, while the best results were obtained with the SPAI [10] preconditioner.

The SPAI technique computes and stores explicitely a matrix M that approximates the inverse of A. Its computation is based on the solution of the Frobenius norm minimization problem $\|I - AM\|_F$. This problem is formulated equivalently as

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

where e_j is the unitary vector with all zero entries except the jth one which is equal to 1. Then, by (2), M can be obtained solving n independent linear least squares problems, one for each column m_j of M. This characteristic has made the SPAI preconditioning technique quite popular in parallel computing environments. To obtain a sparse preconditioner at a moderate computational cost only some elements of the column m_j are computed. Usually a static sparsity pattern of nonzero elements is used [10]. The pattern is chosen from a sparse matrix $\hat{A}$ obtained applying an sparsifying process to A that keeps only the biggest entries. Figures 1 and 2 show the sparse patterns of the matrix $\hat{A}$ and its inverse for the matrix CETAF1178 (see Table 1). These matrices are obtained considering the distance between mesh elements in terms of wavelength and discarding the entries that are not strongly coupled. Comparing both figures we observe that their patterns present some similarities (see [2] for a detailed study). Therefore, the SPAI technique is applied to a sparsified system matrix $\hat{A}$ which is explicitly stored in memory and corresponds to the near-field part of A.

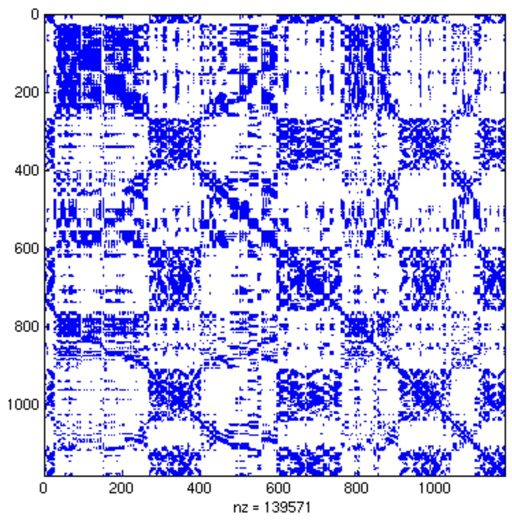

Fig. 1 Sparse pattern of the sparsified matrix $\hat{A}$.

In Figures 3 and 4 we can see the effect of the preconditioner on the spectrum of the system matrix CETAF1178. Even though most eigenvalues are shifted to one, there are still some of them clustered around the origin which can slow down the convergence of the iterative method. In the next section we will see that applying low-rank updates to the preconditioned matrix, a small subset of these eigenvalues can be shifted to 1, improving the convergence of the Krylov iterative solvers.

3 Spectral low rank updates

As already stated, the purpose of the SLRU technique is to shift a subset of the smallest eigenvalues in magnitude of a given matrix from the origin to one. Let M_1A be a left preconditioned matrix and assume it is diagonalizable, that is:

$$M_1A = V\Lambda V^{-1},$$

where $\Lambda = \text{diag}(\lambda_i),\ |\lambda_1| \leq \cdots \leq |\lambda_n|$ are the eigenvalues of M_1A, and $V = (v_i)$ is the matrix whose columns are the associated right eigenvectors. Let V_ε be the matrix whose columns are the set of right eigenvectors associated with the set of eigenvalues λ_i with $|\lambda_i| < \varepsilon$. We have the following result from [4].

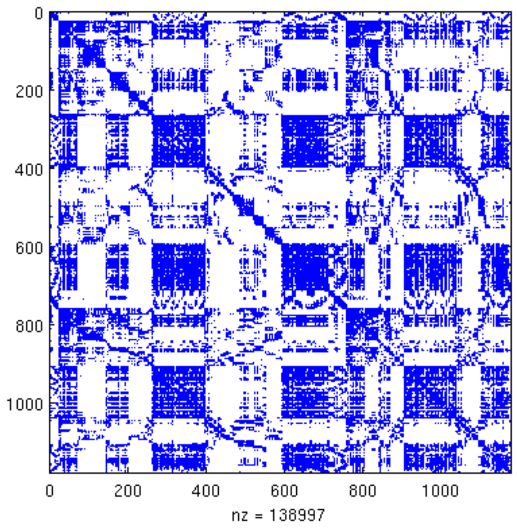

Fig. 2 Sparse pattern of $\hat{A}^{-1}$

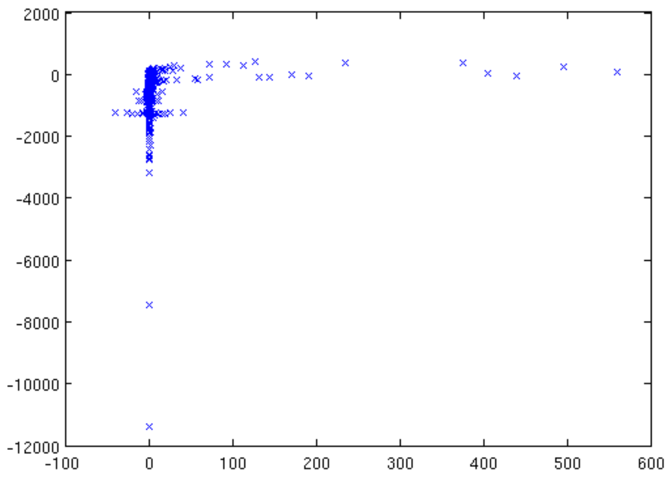

Fig. 3 Eigenvalues of A.

Theorem 1 (Proposition 2 of [4]) *Let W be a matrix such that the matrix $\tilde{A}_c = W^H A V_\varepsilon$ has full rank, $\tilde{M}_c = V_\varepsilon \tilde{A}_c^{-1} W^H$ and*

$$\tilde{M} = M_1 + \tilde{M}_c.$$

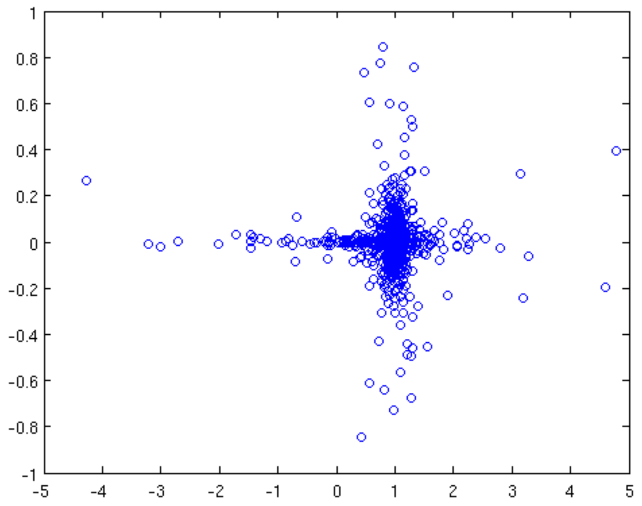

Fig. 4 Eigenvalules of *MA*

Then the matrix $\tilde{M}A$ is similar to a matrix whose eigenvalues are

$$\begin{cases} \eta_i = \lambda_i & \textit{if } |\lambda_i| > \varepsilon, \\ \eta_i = 1 + \lambda_i & \textit{if } |\lambda_i| \leq \varepsilon. \end{cases}$$

A similar result also holds for right preconditioned matrices.

To reduce the computational cost the matrix W is chosen equal to V_ε. The result implies that some of the eigenvalues closest to the origin can be shifted near to one by computing the associated right eigenvectors. We recall that for most of the problems considered in computational electromagnetics the system matrix is very ill-conditioned and indefinite. Therefore the computation of these eigenpairs can be a challenge since interior eigenvalues computation is a difficult task. In previous work the Implicit Restarted Arnoldi method implemented in ARPACK [16] has been employed [2–6]. The results showed that the SLRU technique can improve considerably the performance, specially when multiple right hand sides have to be solved, as it is the case for many scattering problems, for instance the computation of the radar cross section of a complex body. In this work we investigate the use of a Jacobi-Davidson method. In particular the JDQR variant [8].

4 Jacobi-Davidson method

The Jacobi-Davidson [20] method is an excellent method for the computation of a subset of eigenpairs of a matrix. Because its basic ingredients are matrix-vector products, vector updates and inner products it can be parallelized easily [1]. Its JDQR variant [8], which is based on the computation of a partial Schur form of the system matrix, uses deflation techniques to compute a number of interior eigenpairs. The application of the method involves the solution of a small projected eigenproblem. Given an approximate eigenpair (u,θ) the correction equation

$$(I-uu^*)(A-\theta I)(I-uu^*)s=-r, \quad s\perp u\ , \tag{3}$$

where $r=Au-\theta u$, must be approximately solved to expand the search subespace. This is the most expensive phase of the algorithm.

In our case to obtain good eigenpairs estimates to apply the low-rank updates, it was enough to solve equation (3) iteratively using GMRES(10) without preconditioning and stopping the iteration at a small accuracy (10^{-2}). Indeed, we found that on the average for medium size and large problems the computational cost of the spectral low-rank updates is moderate if compared with the overall solution time.

5 Numerical experiments

In this section we show the results of the numerical experiments obtained for a set of model problems which are listed in Table 1. All matrices were kindly provided to us by the EADS-CASA company.

Table 1 Tested matrices

Matrix	size
CETAF1178	1178
CETAF3000	3097
CETAF5000	5021
CETAF10000	10022
CN5000	5005

The BiCGSTAB [15] method with left preconditioning was used to solve the linear system. In [14] the authors found that the GMRES(m) method performs similarly to the BiCGSTAB with a restart parameter value around 10 percent of the size of the matrix. We choosed the BiCGSTAB method because its smaller memory requirements, that can become a critical point in large scale computational electromagnetics computations. All codes developed for the tests were written in FORTRAN 95 in double precision complex arithmetic, and have been compiled with Intel Fortran Composer XE 2013 and linked with the Intel Math Kernel Library. For the experiments we used one AMD Opteron CPU of a Sun Fire X2200 M2 Server. The experiments were made with a SPAI preconditioner of moderate density, i.e., the ratio between the number of nonzeros and the square of its dimension is less than 4 percent in all cases. Table 2 reports relative densities, under ρ, and the time needed to compute the SPAI preconditioners used. The iterative method was stopped when the initial residual was reduced by a factor of 10^{-6}. The JDQR method was run to compute the eigenpairs for the low-rank updates until the norm of the residual vector was reduced to $\varepsilon_1=10^{-1}$ or to $\varepsilon_2=10^{-2}$. These parameters are indicated in tables as superscripts of the SLRU dimension. To speed-up the computations, instead of computing the eigenpairs of the full matrix the JDQR was applied to a preconditioned near-field matrix $M\hat{A}$. Tables also the parameter δ that indicates the relative density of the matrix $\hat{A}$. We note that the sparsified matrix $\hat{A}$ used to compute the SLRU subspace may be different from that used for the computation of the SPAI preconditioner M. We found that it is better, from the point of view of

Table 2 SPAI preconditioner density and computation time

Matrix	ρ	T. prec (s)
CETAF1178	0.04	3.3
CETAF3000	0.03	31.3
CETAF5000	0.02	21.1
CETAF10000	0.01	85.2
CN5000	0.01	15.4

Table 3 Matrix CETAF1178. SLRU subspace computed with ARPACK and JDQR.

SLRU method	SLRU size	Iter.	CPU SLRU time (s)	CPU solution time (s)	VA
ARPACK	0	297	0	4.21	
	$5^{(\varepsilon_2)}$	273	27.9	3.93	100
	$10^{(\varepsilon_2)}$	191	30.6	2.77	21
	$15^{(\varepsilon_2)}$	138	34.19	2.06	16
	$20^{(\varepsilon_2)}$	81	29.72	1.21	10
JDQR	0	297	0	5	
	$5^{(\varepsilon_2)}$	223	3.82	2.97	3
	$10^{(\varepsilon_2)}$	95	5.96	1.26	2
	$15^{(\varepsilon_2)}$	65	9.37	0.88	3
	$20^{(\varepsilon_2)}$	58	11.86	0.79	4

total computational time, to keep M as sparse as possible and then compute the SLRU subspace for a slightly denser matrix $\hat{A}$. Table 9 shows some results concerning this idea for the matrix CETAF3000. SLRU size is the number of eigenpairs computed to apply the updates; Iter., the number of iterations performed by the BiCGSTAB; and VA, the number of linear systems that amortize the extra computational cost of computing the SLRU subspace.

We want to compare the performance for computing the SLRU subspace of the previously used, [2–6], IRA method, with the performance of the Jacobi-Davidson method. The balance between the quality of the SLRU subspace and its computational cost for different densities of the matrix $\hat{A}$ is also of interest. In addition, some remarks concerning to the performance for different sizes of the subspace SLRU and the density of the SPAI preconditioner are given.

Tables 3 to 5 show the performance differences observed when computing the SLRU subspace using the ARPACK and JDQR methods. We observe that the computational cost reduces quickly for a relative small size of the SLRU subspace. Nevertheless, with JDQR the performance of the SLRU method seems to be higher compared with ARPACK. Tables show that the JDQR method is quite competitive for the relative small number of eigenpairs, around 1% of the size of the matrix, needed to be computed in order to accelerate the convergence of the iterative method. For the matrices not listed above the same behaviour was observed. We recall the moderate accuracy estimation and small number of eigenpairs required in order to successfully apply the spectral correction, and in this situation JDQR performed significantly better.

Tables 6, 7 and 8 show that computing the eigenpairs with the JDQR method the number of vectors VA can be reduce to 1, showing a very good performance of the algorithm. As it has been stated before, the use of ARPACK often pays off when more than 3 systems with the same coefficient matrix are solved. In general, for the tested problems an SLRU subspace of size around 0.5–1 percent of the size of the matrix is enough to obtain the best

Table 4 Matrix CETAF3000. SLRU subspace computed with ARPACK and JDQR.

SLRU method	SLRU size	Iter.	CPU SLRU time (s)	CPU solution time (s)	VA
ARPACK	0	326	0	19.7	
	$15^{(\varepsilon_1)}$	106	18.1	11.1	3
	$30^{(\varepsilon_1)}$	77	16.9	8.1	2
	$60^{(\varepsilon_1)}$	66	34.8	7.1	3
JDQR	0	326	0	19.7	
	$15^{(\varepsilon_1)}$	119	5.3	12.4	1
	$30^{(\varepsilon_1)}$	90	8.62	9.52	1
	$60^{(\varepsilon_1)}$	77	28.7	10.4	3

Table 5 Matrix CETAF10000. SLRU subspace computed with ARPACK and JDQR.

SLRU method	SLRU size	Iter.	CPU SLRU time (s)	CPU solution time (s)	VA
ARPACK	0	922		601.0	
	$50^{(\varepsilon_1)}$	208	384.3	226.9	2
	$100^{(\varepsilon_1)}$	150	689.4	159.8	2
	$200^{(\varepsilon_1)}$	104	1774.5	116.0	3
JDQR	0	922		601.0	
	$50^{(\varepsilon_1)}$	151	131.9	164.0	1
	$100^{(\varepsilon_1)}$	91	287.8	100.5	1
	$200^{(\varepsilon_1)}$	71	1188.5	79.7	3
	$200^{(\varepsilon_2)}$	126	723.6	142.0	2

results. Larger subspace sizes increase the overall computational time in such a way that the number of amortizing vectors may become larger. In addition, it is important to note that, for large scale problems, the memory storage requirements can also be a limiting factor and therefore it must be taken into account.

With respect to the density of the sparsified matrix $\hat{A}$ used to compute the eigenpairs, a density from 2 to 5 percent is usually good. Higher densities improve the quality of the SLRU subspace and the number of iterations decreases. But, as the computational time also grows, the extra cost only pays off for applications where many linear system with the same coefficient matrix is going to be solved for different right hand sides, without mentioning the extra memory requirements. We do not recommend densities larger than 10 percent.

Another important question is the precision recommended to estimate the eigenpairs of the SLRU subspace. From the results we can see that reducing the initial error to 10^{-1} or to 10^{-2} can be a good compromise between the overall computational time and the number of iterations required to obtain a solution.

Finally, Table 9 shows the results for the matrix CETAF3000 where both, the effect of the density of the SPAI preconditioner and the matrix $\hat{A}$ used for the SLRU computation, was studied. In this table, ρ indicates the density of the preconditioner relative to the number of elements of the system matrix. The column Prec. time corresponds to the preconditioner computational time. As pointed above, the computation of the SPAI preconditioner has a big impact in the total solution time. Thus, we found that it is better to keep the preconditioner matrix M as sparse as possible and then, if needed, performing the SLRU computations with a slightly denser matrix $\hat{A}$. That implies that the sparsified matrix $\hat{A}$ used to compute M may be different from that used for the SLRU subspace. For instance, considering only the

Table 6 Matrix CETAF5000. SLRU subspace computed with JDQR.

SLRU size	Iter.	CPU SLRU time (s)	CPU solution time (s)	δ	VA
0	940		128.5		
$25^{(\varepsilon_2)}$	653	12.2	178.2	0.02	1
$25^{(\varepsilon_1)}$	426	19.4	143.3	0.02	1
$25^{(\varepsilon_2)}$	269	9.5	94.0	0.04	1
$25^{(\varepsilon_1)}$	130	22.1	35.6	0.04	1
$25^{(\varepsilon_2)}$	138	20.0	38.7	0.10	1
$25^{(\varepsilon_1)}$	89	33.3	24.4	0.10	1
$50^{(\varepsilon_1)}$	1110	55.0	410.0	0.01	†
$50^{(\varepsilon_2)}$	163	26.4	51.1	0.02	1
$50^{(\varepsilon_1)}$	141	40.5	47.2	0.02	1
$50^{(\varepsilon_2)}$	81	39.4	22.4	0.04	1
$50^{(\varepsilon_1)}$	59	48.0	16.4	0.04	1
$50^{(\varepsilon_2)}$	89	30.5	24.6	0.10	1
$50^{(\varepsilon_1)}$	46	53.3	12.8	0.10	1
$100^{(\varepsilon_2)}$	56	140.3	15.8	0.04	1
$100^{(\varepsilon_1)}$	45	145.5	13.0	0.04	1
$100^{(\varepsilon_2)}$	27	100.6	7.7	0.10	1
$100^{(\varepsilon_1)}$	25	148.8	9.7	0.10	1

Table 7 Matrix CETAF10000. SLRU subspace computed with JDQR.

SLRU size	Iter.	CPU SLRU time (s)	CPU solution time (s)	δ	VA
0	922		601.1		
$25^{(\varepsilon_1)}$	240	64.5	324.0	0.05	1
$25^{(\varepsilon_2)}$	319	49.5	365.8	0.05	1
$50^{(\varepsilon_1)}$	260	75.5	282.6	0.02	1
$50^{(\varepsilon_2)}$	319	58.5	347.4	0.02	1
$50^{(\varepsilon_1)}$	164	131.9	164.0	0.05	1
$50^{(\varepsilon_2)}$	234	75.3	306.7	0.05	1
$100^{(\varepsilon_1)}$	91	287.8	100.5	0.05	1
$100^{(\varepsilon_2)}$	105	198.7	155.3	0.05	1
$200^{(\varepsilon_1)}$	137	1188.5	173.0	0.05	2
$200^{(\varepsilon_2)}$	126	723.6	142.0	0.05	2

Table 8 Matrix CN5000. SLRU subspace computed with JDQR.

SLRU size	Iter.	CPU SLRU time (s)	CPU solution time (s)	δ	VA
0	3451		465.5		
$25^{(\varepsilon_1)}$	1385	37.0	380.5	0.04	1
$25^{(\varepsilon_1)}$	793	52.6	213.7	0.10	1
$50^{(\varepsilon_1)}$	1072	56.5	296.9	0.04	1
$50^{(\varepsilon_1)}$	538	81.4	126.6	0.10	1
$100^{(\varepsilon_1)}$	270	180.4	76.4	0.04	1
$100^{(\varepsilon_1)}$	73	185.0	20.4	0.10	1

Table 9 Matrix CETAF3000. ρ indicates the density of the preconditioner SPAI relative to the number of elements of the matrix. CPU time in seconds.

ρ	Prec. time	SLRU size	Iter.	SLRU time	Solution time	δ	VA
0.1	589.5		166		15.5		
0.06	142.2		229		13.5		
0.03	38.3		326		19.7		
		$15^{(\varepsilon_1)}$	390	7.0	41.4	0.02	†
		$15^{(\varepsilon_1)}$	119	5.3	12.4	0.03	1
		$15^{(\varepsilon_2)}$	181	2.6	19.2	0.03	1
		$15^{(\varepsilon_1)}$	100	5.3	10.7	0.06	1
		$15^{(\varepsilon_2)}$	126	4.4	13.5	0.06	1
		$15^{(\varepsilon_1)}$	82	8.2	8.7	0.10	1
		$15^{(\varepsilon_2)}$	113	5.9	12.0	0.10	1

solution time, Table 9 shows that the best result for SPAI preconditioning without the SLRU technique is achieved with a density of 6 percent. But the application of the SLRU technique allows us to almost achieve this result with a preconditioner of half size. In this case, we compute the SLRU subspace with a slightly denser matrix $\hat{A}$ ($\delta = 0.06$). Also observe that, for a given preconditioner size, the solution time decreases for increasing densities of $\hat{A}$ although the overall solution time is obtained for intermediate values. We also note that there are not big differences between the results obtained for different δ values. Therefore, fine tuning can be avoided by keeping the same matrix $\hat{A}$ for computing both, the SPAI preconditioner and the SLRU subspace without an important loose of efficiency. Densities from 2 to 10 percent have been good enough for all the problems considered.

6 Conclusions

In this work we consider the iterative solution of the linear systems arising from different computational electromagnetics applications. We studied spectral low-rank updates (SLRU) of the preconditioned system matrix. The SLRU relies on the computation of a small subset of the eigenvalues closest to the origin which have a negative effect on the convergence of the Krylov solvers. The SLRU subspace is used to update an existing SPAI preconditioner. Previous work computed these updates by using an Implicit Restarted Arnoldi method, in particular ARPACK's implementation. In this work we experimented with a Jacobi-Davidson method, the JDQR variant. From the numerical experiments presented we conclude that the combination of the SLRU method with the JDQR algorithm provides a robust and fast algorithm to reduce the number of iterations and overall computational cost. In order to speed-up the SLRU set-up time it is convenient to apply the JDQR method to a sparsified matrix $\hat{A}$, the near-field part of the system matrix. Moreover, estimating the eigenvalues with precision 0.1 or 0.01 is enough to obtain good low-rank updates. And finally, the use of the JDQR algorithm reduces the computational cost of the updates such that multiple right hand sides for the same coefficient matrix are not needed to be solved.

References

1. Bergamaschi, L., Pini, G., Sartoretto, F.: Computational experience with sequential, and parallel, preconditioned Jacobi Davidson for large sparse symmetric matrices. J. Comut. Phys., 188(1), 318–331 (2003)

2. Carpentieri, B.: Sparse preconditioners for dense linear systems from electromagnetics applications. PhD thesis, Institut National Polytechnique de Toulouse, CERFACS (2002)
3. Carpentieri, B., Duff, I. S., Giraud, L.: Sparse pattern selection strategies for robust Frobenius-norm minimization preconditioners in electromagnetism. Numer. Linear Algebr. Appl., 7(7-8), 667–685 (2000)
4. Carpentieri, B., Duff, I. S., Giraud, L.: A class of spectral two-level preconditioners. SIAM J. Sci. Comput., 25(2), 749–765 (2003)
5. Carpentieri, B., Duff, I. S., Giraud, L., Magolu monga Made, M.: Sparse symmetric preconditioners for dense linear systems in electromagnetism. Numer. Linear Algebr. Appl., 11(8-9), 753–771 (2004)
6. Carpentieri, B., Duff, I. S., Giraud, L., Sylvand, G.: Combining fast multipole techniques and an approximate inverse preconditioner for large electromagnetism calculations. SIAM J. Sci. Comput., 27(3), 774–792 (2005)
7. Darve, E.: The Fast Multipole Method I: Error Analysis and Asymptotic Complexity. SIAM J. Numer. Anal., 38(1), 98–128 (2000)
8. Fokkema, D. R., Sleijpen, G. L., Van der Vorst, H. A.: Jacobi-Davidson style QR and QZ algorithms for the reduction of matrix pencils. SIAM J. Sci. Comput., 20(1), 94–125 (1998)
9. Greengard, L., Rokhlin, V.: A fast algorithm for particle simulations. J. Comput. Phys., 73(3), 325–348 (1987)
10. Grote, M., Huckle, T.: Parallel preconditioning with sparse approximate inverses. SIAM J. Sci. Comput., 18(3), 838–853 (1997)
11. Harrington, R.: Origin and development of the Method of Moments for field computation. IEEE Antenn. Propag. M. (1990)
12. Kunz, K. S., Luebbers, R. J.: The Finite Difference Time Domain Method for Electromagnetics. SIAM J. Sci. Comput., 18(3), 838–853 (1997)
13. Maxwell, J. C.: A dynamical theory of the electromagnetic field. Roy. S. Trans., CLV, (1864). Reprinted in Tricker, R. A. R., The Contributions of Faraday and Maxwell to Electrial Science, Pergamon Press (1966)
14. Marín, J., Malla M.: Some experiments preconditioning via spectral low rank updates for electromagnetism applications. Proceedings of the International Conference on Preconditioning Techniques for Large Sparse Matrix Problems in Scientific and Industrial Applications (Preconditioning 2007), Toulouse, France, 2007.
15. Meijerink, J. A., van der Vorst, H. A.: An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. Math. Comp., 31, 148–162, (1977)
16. Sorensen, D. C., Lehoucq, R. B., Yang, C.: ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods. SIAM, Philadelphia (1998)
17. Rao, S. M., Wilton, D. R., Glisson, A. W.: Electromagnetic scattering by surfaces of arbitrary shape. IEEE Trans. Antenn. Propag., AP-30, 409–418 (1982)
18. Saad, Y.: Iterative Methods for Sparse Linear Systems. PWS Publishing Company, Boston (1996)
19. Silvester, P. P., Ferrari, R. L.: Finite Elements for Electrical Engineers. Cambridge University Press, Cambridge (1990)
20. Sleijpen, S. L., van der Vorst, H. A.: A Jacobi-Davidson iteration method for linear eigenvalue problems. SIAM J. Matrix Anal. Appl., 17, 401–425 (1996)
21. van der Vorst, H. A. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of non-symmetric linear systems. SIAM J. Sci. Stat. Comput., 12(6), 631–644 (1992)