A Parallel Implementation of the Jacobi-Davidson Eigensolver for Unsymmetric Matrices*

Eloy Romero[1], Manuel B. Cruz[2], Jose E. Roman[1], and Paulo B. Vasconcelos[3]

[1] Instituto I3M, Universidad Politécnica de Valencia,
Camino de Vera s/n, 46022 Valencia, Spain
{eromero,jroman}@upvnet.upv.es
[2] Laboratório de Engenharia Matemática
Instituto Superior de Engenharia do Porto,
Rua Dr. Bernardino de Almeida, 431, 4200-072 Porto
mbc@isep.ipp.pt
[3] Centro de Matemática da Universidade do Porto and
Faculdade de Economia da Universidade do Porto,
Rua Dr. Roberto Frias s/n, 4200-464 Porto, Portugal
pjv@fep.up.pt

Abstract. This paper describes a parallel implementation of the Jacobi-Davidson method to compute eigenpairs of large unsymmetric matrices. Taking advantage of the capabilities of the PETSc library —Portable Extensible Toolkit for Scientific Computation—, we build an efficient and robust code adapted either for traditional serial computation or parallel computing environments. Particular emphasis is given to the description of some implementation details of the so-called correction equation, responsible for the subspace expansion, and crucial in the Jacobi-Davidson algorithm. Numerical results are given and the performance of the code is analyzed in terms of serial and parallel efficiency. The developments achieved in the context of this work will be incorporated in future releases of SLEPc —Scalable Library for Eigenvalue Problem Computations—, thus serving the scientific community and guaranteeing dissemination.

Keywords: Message-passing parallel computing, eigenvalue computations, Jacobi-Davidson.

1 Introduction

The Jacobi-Davidson method is a popular technique to compute a few eigenpairs of large sparse matrices, i.e., (λ, x) pairs that satisfy $Ax = \lambda x$, $x \neq 0$. This problem arises in many scientific and engineering areas such as structural dynamics, electrical networks, quantum chemistry and control theory, among others.

* This work was partially supported by Fundação para a Ciência e a Tecnologia - FCT, through Centro de Matemática da Universidade do Porto - CMUP, and by the Spanish Ministerio de Educación y Ciencia under project TIN2009-07519.

Its introduction, about 15 years ago, was motivated by the fact that standard iterative eigensolvers often require an expensive factorization of the matrix to compute interior eigenvalues, *e.g.*, shift-and-invert Arnoldi with a direct linear solver. Jacobi-Davidson tries to reduce the cost by solving linear systems only approximately (usually with iterative methods) without compromising the robustness.

General purpose parallel Jacobi-Davidson eigensolvers are currently available in the PRIMME [1], JADAMILU [2] and Anasazi [3] software packages. However, currently PRIMME and JADAMILU can only cope with standard Hermitian eigenproblems, and Anasazi only implements a basic block Davidson method for generalized Hermitian eigenproblems, $Ax = \lambda Bx$, where the user is responsible for implementing the correction equation. There are no freely available parallel implementations for the non-Hermitian case.

There are several publications dealing with parallel Jacobi-Davidson implementations employed for certain applications. For instance, the design of resonant cavities needs to solve real symmetric generalized eigenproblems arising from the Maxwell equations [4]. Other cases result in non-Hermitian problems, as they occur in linear magnetohydrodynamics (MHD): in [5] a complex generalized non-Hermitian problem is solved using the shift-and-invert spectral transformation for searching for the interior eigenvalues closest to some target, and a variant with harmonic Ritz values is presented in [6].

Our aim is to fill the lack of a parallel general purpose non-Hermitian Jacobi-Davidson eigensolver providing a robust and efficient implementation in the context of SLEPc, the Scalable Library for Eigenvalue Problem Computations [7]. Our previous work addresses the Generalized Davidson method for symmetric-definite (and indefinite) generalized problems [8].

In this work we focus on important details of the non-Hermitian version of Jacobi-Davidson: searching for interior eigenvalues using harmonic Ritz values and the real arithmetic management of complex conjugate eigenpairs in problems with real matrices. These improvements will be added to a future fully-fledged Jacobi-Davidson solver in SLEPc. The description of the particular features of the solver implemented in this work are described in Section 2. The analysis of the performance of the code is detailed in Section 3. We conclude with some final remarks.

2 The Jacobi-Davidson method

The Jacobi-Davidson algorithm combines ideas from Davidson's and Jacobi's methods (see [9]). As all other methods based on projection techniques, it has an extraction phase and another one of subspace expansion (see Algorithm 1).

Regarding the extraction phase, A is projected on a low-dimensional subspace $\mathcal{K}$ of size m with Ritz values and Ritz vectors, respectively, θ_j and $u_j = Vs_j$, $j = 1, ..., m$, being $V = [v_1 v_2 \dots v_m]$ the $n \times m$ matrix whose columns form an orthonormal basis of $\mathcal{K}$.

Algorithm 1 Jacobi-Davidson algorithm

procedure JD($A, u_0, tol, itmax$)
  $u \leftarrow u_0/\|u_0\|_2$, $V \leftarrow [u]$, $\theta \leftarrow u^*Au$, $r \leftarrow Au - \theta u$
  $m \leftarrow 1$
  while $\|r\|_2/|\theta| > tol$ & $m < itmax$ **do**
    Solve approximately $(I - uu^*)(A - \theta I)(I - uu^*)t = -r, \quad t \perp u$
    $V \leftarrow$ Orthonormalize$[V, t]$, and $m \leftarrow m + 1$
    Compute a desired eigenpair (θ, s) from the projected eigensystem
      using standard or harmonic techniques
    $u \leftarrow Vs$, $r \leftarrow Au - \theta u$
  end while
  return θ, u
end procedure

In the expansion phase, a new direction should be considered in such a way that $\mathcal{K}$ is extended providing better information to obtain a new approximation of the selected eigenpair (θ, u). Jacobi [10] proposed to correct u by a vector t, the Jacobi orthogonal component correction (JOCC)

$$A(u + t) = \lambda(u + t), \quad u \perp t. \tag{1}$$

Pre-multiplying (1) by u^*, and considering $\|u\| = 1$, results in

$$\lambda = u^*A(u + t). \tag{2}$$

Projecting (1) onto the orthogonal complement of u, which is done by pre-multiplying by $I - uu^*$, and replacing λ, which is unknown, by the available approximation θ results in the Jacobi-Davidson correction equation,

$$(I - uu^*)(A - \theta I)(I - uu^*)t = -r, \tag{3}$$

being $r = Au - \theta u$ the eigenpair residual. Far from convergence, θ can be set to a target τ.

Usually, iterative linear solvers are used to solve the equation (3), for instance using Krylov methods. Regarding the required precision for the resolution of (3), a low one is generally accepted. In practice, the performance of the method largely depends on the appropriate tunning of this parameter.

2.1 Computation of eigenvalues at the periphery of the spectrum

For exterior eigenvalues, the Jacobi-Davidson method uses the Rayleigh-Ritz approach to extract the desired approximate eigenpair $(\theta, u = Vs)$, by imposing the Ritz-Galerkin condition on the associated residual,

$$r = Au - \theta u \perp \mathcal{K}, \tag{4}$$

which leads to the low-dimensional projected eigenproblem

$$V^*AVs = \theta s. \tag{5}$$

At each iteration the implementation updates matrix $M = V^*AV$, by computing the jth row and column. Then it computes the Schur decomposition of M and sorts it in order to have the desired Ritz value and vector as the first one.

However, the Rayleigh-Ritz method generally gives poor approximate eigenvectors for interior eigenvalues, that is, the Galerkin condition (4) does not imply that the residual norm is small. Moreover, spurious Ritz values can appear and it is difficult to distinguish them from the ones we seek. In that case, the Ritz vector can contain large components in the direction of eigenvectors corresponding to eigenvalues from all over the spectrum, so adding this vector to the search subspace will hinder convergence. The harmonic projection methods are an alternative to solve this problem [11, 12].

2.2 Computation of interior eigenvalues

For interior eigenpairs, Jacobi-Davidson uses, in general, the harmonic Rayleigh-Ritz extraction, requiring the Petrov-Galerkin condition

$$(A - \tau I)^{-1}u - (\theta - \tau)^{-1}u \perp \mathcal{L}, \tag{6}$$

where $\mathcal{L} \equiv (A - \tau I)^*(A - \tau I)\mathcal{K}$.

This extraction technique exploits the fact that the harmonic Ritz values closest to the target τ are the reciprocals of the largest magnitude Ritz values of $(A-\tau I)^{-1}$, and avoids working with the inverse of a large matrix. The resulting projected generalized eigenvalue problem is then

$$V^*(A - \tau I)^*(A - \tau I)Vs = (\theta - \tau)V^*(A - \tau I)^*Vs. \tag{7}$$

Our implementation of the harmonic extraction is based on the algorithm described in [13, §7.12.3], which maintains W as an orthogonal basis of $(A-\tau I)V$, and updates the matrices $N = W^*W$ and $M = W^*V$ in each iteration. Thus, in this case, step 7 in Algorithm 1 has to solve the eigenproblem associated to the (N, M) matrix pair (see Algorithm 2). The new vectors added to W, as in the case of the V vectors, are orthogonalized with a variant of the Gram-Schmidt process available in SLEPc, which is based on classical Gram-Schmidt with selective reorthogonalization, providing both numerical robustness and good parallel efficiency [14].

2.3 Computing complex eigenvalues with real arithmetic

Real unsymmetric matrices may have complex eigenvalues and corresponding eigenvectors. It is possible to avoid the complex arithmetic by using real Schur vectors instead of eigenvectors in step 7 of Algorithm 1. The real Schur decomposition consists of an orthogonal matrix and a block upper triangular matrix, which has scalars or two by two blocks on the diagonal. The eigenvalues of two by two blocks correspond to two complex conjugate Ritz values of the original

Algorithm 2 Harmonic extraction in Jacobi-Davidson algorithm

1: $w \leftarrow (A - \tau I)v_m$, where v_m is the last column of V
2: $W \leftarrow$ Orthonormalize$[W, w]$, with h the orthogonalization coefficients and ρ the norm prior to normalization
3: Update $N \leftarrow \begin{bmatrix} N & h \\ 0 & \rho \end{bmatrix}$,
4: Update M such that $M = W^*V$
5: Compute generalized Schur decomposition $NQ = Z\tilde{N}$, $MQ = Z\tilde{M}$ such that $|\tilde{n}_{1,1}/\tilde{m}_{1,1}|$ is the minimum
6: $s \leftarrow q_1$ and $\theta \leftarrow \overline{\tilde{m}_{1,1}}\tilde{n}_{1,1} + \tau$

matrix. The real Schur form requires less storage, since these Schur vectors are always real. Another advantage is that complex conjugate pairs of eigenvalues always appear together.

Our implementation is based on RJDQZ [15], that adapts the extraction process and the correction equation to work with the real Schur form. The changes to the original algorithm are only significant when a complex Ritz pair is selected. In that case, the method works with a real basis $U = [u_1\ u_2]$ of the invariant subspace associated to the selected complex conjugate pair of Ritz values, $(\theta, \bar{\theta})$, where the corresponding Ritz vectors are $u = u_1 \pm u_2\, i$. The residual is now computed as

$$[r_1\ r_2] = A[u_1\ u_2] - [u_1\ u_2] \begin{bmatrix} \Re(\theta) & \Im(\theta) \\ -\Im(\theta) & \Re(\theta) \end{bmatrix}. \tag{8}$$

Apart from the residual, the correction equation also contains the Ritz value, so it has to be rearranged as

$$P \begin{bmatrix} A - \Re(\theta)I & \Im(\theta)I \\ -\Im(\theta)I & A - \Re(\theta)I \end{bmatrix} P \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} = - \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}. \tag{9}$$

For this correction equation, we propose three different projectors P, that can be selected at runtime in our solver. The first option, P_0, implements the same projector that would be used in complex arithmetic,

$$P_0 = I - \begin{bmatrix} u_1 & u_2 \\ u_2 & -u_1 \end{bmatrix} \begin{bmatrix} u_1^T & u_2^T \\ u_2^T & -u_1^T \end{bmatrix}. \tag{10}$$

P_1 represents the orthogonal projector onto the orthogonal complement of the subspace $\mathcal{U} = \text{span}\{u_1, u_2\}$,

$$P_1 = I - \begin{bmatrix} \hat{u}_1 & \hat{u}_2 & 0 & 0 \\ 0 & 0 & \hat{u}_1 & \hat{u}_2 \end{bmatrix} \begin{bmatrix} \hat{u}_1^T & 0 \\ \hat{u}_2^T & 0 \\ 0 & \hat{u}_1^T \\ 0 & \hat{u}_2^T \end{bmatrix}, \tag{11}$$

where $\{\hat{u}_1, \hat{u}_2\}$ is an orthogonal basis of $\mathcal{U}$. This basis is cheaply obtained from the Schur decomposition of the projected problem. Finally, P_2 implements the

following lighter projector

$$P_2 = I - \begin{bmatrix} \hat{u}_1 & 0 \\ 0 & \hat{u}_2 \end{bmatrix} \begin{bmatrix} \hat{u}_1^T & 0 \\ 0 & \hat{u}_2^T \end{bmatrix}. \tag{12}$$

2.4 Preconditioning

In some cases, the convergence of the iterative linear solver for the correction equation (3) can be very slow. Therefore, it is convenient to use a preconditioner. The preconditioner can be chosen as

$$M = (I - uu^*)M_\theta(I - uu^*), \tag{13}$$

where M_θ is some approximation of $A - \theta I$. Since θ changes in each outer iteration, it would be necessary to rebuild M_θ every time. This high overhead can be avoided by using M_τ, where τ is the constant target value, but probably losing effectiveness in the preconditioner. The application of M^{-1} to a given vector is done without building M explicitly, in such a way that only one preconditioner application with M_τ^{-1} is done per inner iteration [9].

2.5 Implementation details

Our Jacobi-Davidson implementation is intended to be executed on distributed memory parallel machines using PETSc. The problem matrix and the vectors of size n are distributed by blocks of rows and the corresponding operations are parallelized. These include the computation of the matrices of the projected system, the selected Ritz vector and its residual, the solution of the correction equation and the orthogonalization of V and W. The projected problem decomposition and other minor computations are replicated in all nodes.

The algorithm is initialized with a randomly generated vector in parallel, taking care that each processor generates different random sequences. This initial vector must have a nonzero component in the desired eigenvector. To further ensure this feature, and to seek the eigenvalue of largest magnitude, a few iterations of the power method may be applied.

The real Schur decomposition of the projected problem is computed using appropriate LAPACK routines, since they are dense and small matrices. Also, LAPACK is used for sorting the Schur decomposition, keeping the complex conjugate pairs together.

The stopping criterion of the algorithm is currently based on the simple evaluation of the normalized residual $\|Au - \theta u\|_2/|\theta|$. When the convergence is achieved then there exists an exact eigenvalue λ such that $|\lambda - \theta|$ must be sufficiently small, although λ may not be the desired eigenvalue. In the symmetric case, a small relative residual guarantees a small error in the eigenvalue; for the unsymmetric case, however, the norm of the residual may give an indication of the error but it is not guaranteed, especially for highly non-normal matrices.

The correction equations, (3) and (9), are solved using a PETSc Krylov solver (commonly GMRES) without explicitly storing the coefficient matrices of

the system, and only defining the matrix-vector product. Also, only applying the left projector is sufficient to guarantee the condition $t \perp u$, provided that a Krylov solver is used with a zero starting vector, as shown in [16].

As mentioned before, the tunning of the stopping criterion for the correction equation is crucial since it greatly influences the convergence behavior of the method. It is widely referred that the required tolerance for the residual on this inner iteration phase does not need to be tight, yet a loose one can provoke a large number of outer iterations or even lead to non-convergence of the Jacobi-Davidson method. A good balance between the required precision for the correction equation and the tolerance of the overall process is thus mandatory. Generally, one can bound the time spent in the correction equation by limiting both the number of iterations and the required precision for the solution. We tested several approaches trying to fulfill the above mentioned ideas as well as to deliver a default option, which can be used by the user as a first approach to solve his eigenproblem. We tested several mechanisms, namely, a fixed small number of GMRES iterations (5, 20 or 40) and a coarse tolerance (10^{-2} or 10^{-3}). We also implemented a dynamic criteria such as a variable tolerance, function of the outer iteration number; for instance, a tolerance of α^{-Its+1} (among others), being Its the outer iteration number, for $\alpha = 1.5$, 2.0, 3.0. The approach proposed in the JDQR code [9] is 0.7^{Its}.

In this work restarting techniques were not considered in order to concentrate on other crucial algorithmic parts and parameters. Details about a restarted variant, together with numerical results on a plasma physics application, can be found in [17]. In general terms, the amount of information remaining after a restart should be neither too small nor too large, in order not to lose too much information and to prevent many time consuming restarts, respectively. A related issue is deflation, required for continuing the computation of other eigenpairs after some of them have converged. Deflation can be implemented by locking converged vectors at restart time [17], so it has not been covered here either.

3 Computational results

Several experiments have been performed for testing our implementation. The sequential tests use a collection of 136 real unsymmetric matrices of moderate size and density (see Figure 1) chosen from the MatrixMarket[4] database and the University of Florida Sparse Matrix Collection[5]. The sequential tests are primarily intended to evaluate the performance and robustness of our implementation. Also, using such a large collection of matrices allows us to extend previous results comparing standard versus harmonic extraction [6, 9] as well as complex versus real arithmetic versions. In order to avoid biased results, the collection is very heterogeneous, and cover the case of large condition number and norm (see Figure 1), clustered eigenvalues and complex conjugate eigenpairs.

[4] http://math.nist.gov/MatrixMarket/
[5] http://www.cise.ufl.edu/research/sparse/matrices/

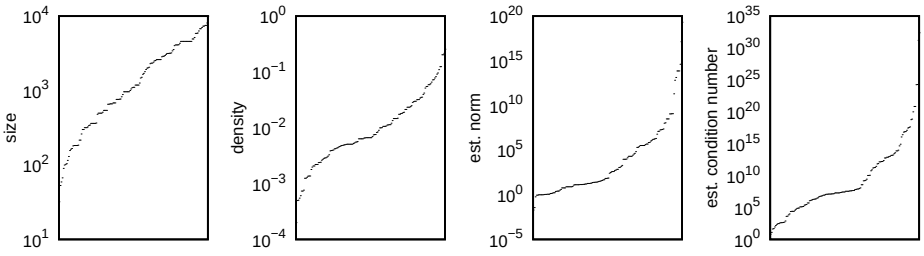

Fig. 1. Size, density, estimated 2-norm and estimated condition number for each matrix used in the sequential tests.

Also the Krylov-Schur eigensolver, currently the most powerful method available in SLEPc, is employed and the results will be compared with Jacobi-Davidson.

In the tests to be presented, only one eigenpair is requested, since locking is not available, as mentioned earlier. The convergence criterion is based on the residual norm divided by the absolute value of the approximate eigenvalue, and a tolerance of 10^{-7} was considered. A problem is flagged as unconverged if the desired tolerance was not reached within a maximum of 500 (outer) iterations.

In order to study the parallel performance of our implementation, some large-scale tests were run on two distributed memory machines: (i) Odin, from Universidad Politécnica de Valencia, which is a cluster with 55 dual-processor nodes (however only one process runs per node in the tests), Pentium Xeon processors at 2.8 GHz and 1 GB per node, connected by a high-speed SCI network with 2D torus topology; and (ii) CaesarAugusta, from the Barcelona Supercomputing Center, consisting of 256 JS20 blade computing nodes, each of them with two 64-bit PowerPC 970FX processors running at 2.2 GHz, interconnected with a low latency Myrinet network, where only 90 processors (i.e., 45 nodes running up to 90 processes) are used due to account limitations.

3.1 The exterior case

For the case of exterior eigenvalues, the developed method was able to compute, within the specified tolerance, the largest eigenvalue in magnitude and associated eigenvector in nearly all cases, as it is shown in Table 1. Except for the real arithmetic version with P_0 projector, any other configuration achieves (at least) 132 converged problems out of 136, and increasing the number of inner iterations or the power iterations helps convergence in especially difficult problems.

On the other hand, the real arithmetic version with P_0 was clearly the worst one. Since the P_1 and P_2 projectors obtained good results, P_0 will be discarded for subsequent analyses within this subsection.

At this stage we should remark that JDQR[6], a Matlab implementation of a Jacobi-Davidson algorithm as described in [9], solved around 82% of the prob-

[6] http://www.math.uu.nl/people/sleijpen/JD_software

Table 1. Number of converged problems for different combinations of the variant, the projector, the number of power iterations and the maximum number of iteration for solving the linear equation system.

Power its.	0			5		
Max. inner its.	5	20	40	5	20	40
complex arith.	134	135	135	134	136	136
real arith. P_0	134	96	103	118	129	129
real arith. P_1	132	133	132	133	133	134
real arith. P_2	134	133	134	134	133	134

lems and that Matlab's function *eigs* failed to converge in around 9% of the problems (using default parameters). Although not directly comparable, these results should be read solely to emphasize the degree of difficulty in the numerical solution of the selected testbed.

When considering the time spent by the solvers, Figure 2 shows that the real arithmetic version can be twice as fast as the complex arithmetic counterpart, especially for problems where complex Ritz pairs do not show up often. In contrast, there is no significant difference between the projector P_1 and P_2. However, the plots comparing the number of outer iterations show the penalty of real arithmetic variants. This can be attributed to the fact that the P_1 and P_2 projectors are not mathematically equivalent to the complex case. Still, each iteration is much faster. These conclusions are statistically significant.

Finally, the influence of the maximum number of linear iterations (MLIts) was analyzed. Figure 3 compares the performance of solving the Jacobi-Davidson correction equation with 5, 20 and 40 GMRES iterations at most. It is observed that the value of this parameter does not seem to influence the number of outer iterations. As a consequence, the higher the value of this parameter, the more overall GMRES iterations are required, and the execution time is thus increased significantly. For the computation of eigenvalues at the periphery of the spectrum, taking into account the mechanisms implemented to date in the code and in light of the successful use of a few linear iterations, extensive tests with a dynamic criterion were not done.

One could draw the conclusion that a very small value of MLIts is to be preferred, although this is application dependent, and may not be the case in a restarted implementation.

3.2 The interior case

As a test for evaluating the computation of interior eigenvalues, we looked for the smallest magnitude eigenvalue of the matrices in the test battery, that is, we run the harmonic solver with $\tau = 0$. This choice leads to convergence difficulties in some matrices of the test battery, mainly attributable to a large condition number with respect to inversion, see Fig. 1. The Jacobi-Davidson eigensolver is configured to perform 50 iterations of GMRES for solving the correction equation, with or without a preconditioner.

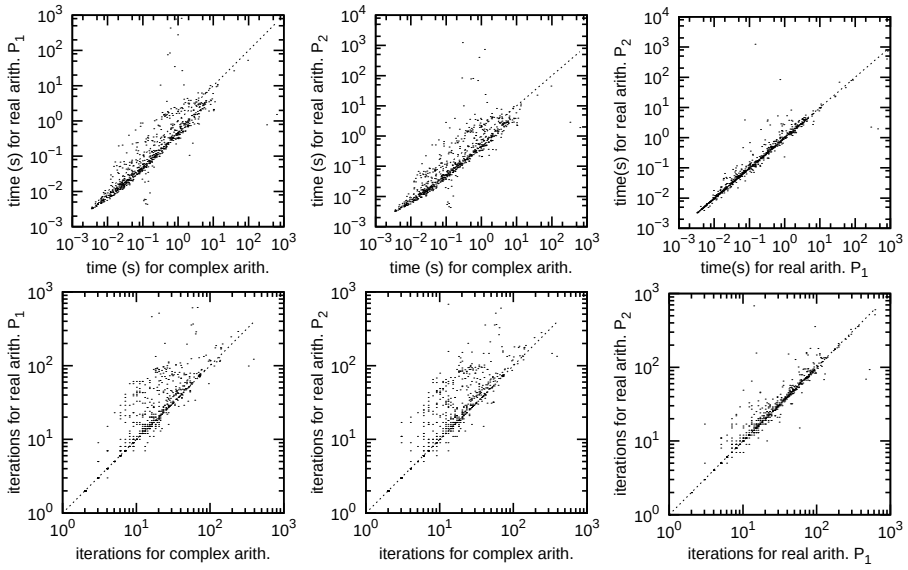

Fig. 2. Plots comparing the execution time (top) and outer iterations (bottom) when solving the problems with the complex arithmetic version and the real arithmetic version with projectors P_1 and P_2.

The standard Rayleigh-Ritz extraction version of Jacobi-Davidson is able to reach convergence in 11 problems only, as expected from the theoretical properties previously discussed. The harmonic extraction version (proposed for this case) successfully solved up to 60 problems without using any preconditioning at the correction equation solution, up to 46 problems with a diagonal preconditioner, up to 39 problems with an ILU(0) preconditioner and up to 100 problems using an LU factorization. These results highlight the impact of the preconditioner in the global convergence of the method. However, a more powerful preconditioner is not always better. We have to recall here that the preconditioner is built from matrix $A - \tau I$ (A in this case) and not from $A - \theta I$, as discussed in section 2.4, so its effectiveness may vary widely depending on the problem. Even a complete factorization such as LU should be considered a preconditioner in this case.

Just for reference, JDQZ (the companion to JDQR with harmonic extraction) without preconditioning solved only 15% of the problems.

Regarding the performance, Fig. 4 shows the hard influence of limiting the maximum number of iterations for solving the correction equation in two sample problems. It is observed the rapidly decreasing of the spent time while allowing more iterations for solving the linear system. Moreover, from a certain point this benefit ends abruptly. A dynamic criteria combining the usual maximum number of iterations and a prescribed tolerance with the requirement of improving the solution quality of the correction equation as the number of outer iterations

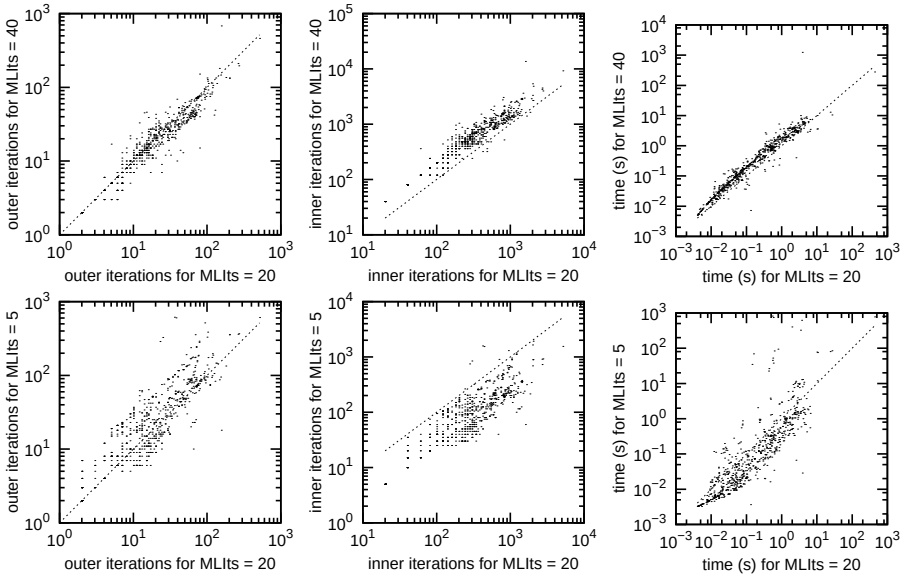

Fig. 3. Plots comparing outer iterations (left), inner iterations (middle), and execution time (right) when solving the problems with 5, 20 and 40 maximum linear iterations (MLIts) of GMRES for solving the Jacobi-Davidson correction equation.

progresses is, thus, essential. The criteria for leaving the iterative process to improve the solution of the linear systems should be more demanding for the interior case than for the exterior.

3.3 Parallel performance

The parallel performance is tested with the matrix *tmt_unsym* from the University of Florida Sparse Matrix Collection, arising from a computational electromagnetics application. The dimension of the matrix is 917,825 and it has 4,584,801 nonzero elements, all of them contained in a narrow band of half-width around 2000.

Generally, the parallel performance of the matrix-vector product and the application of the preconditioner can significantly influence the global performance of an eigensolver. However, the chosen matrix avoids those problems because of the fact that its band structure makes these two operations scalable.

Figure 5 (first row) shows the speedup of the different Jacobi-Davidson versions (real arithmetic using P_0, P_1 and P_2 projectors, and complex arithmetic) and the Krylov-Schur solver in SLEPc, when computing the largest magnitude eigenpair. In both machines it can be observed that all tested implementations show good parallel performance, although the complex arithmetic version and Krylov-Schur are slightly worse than the rest.

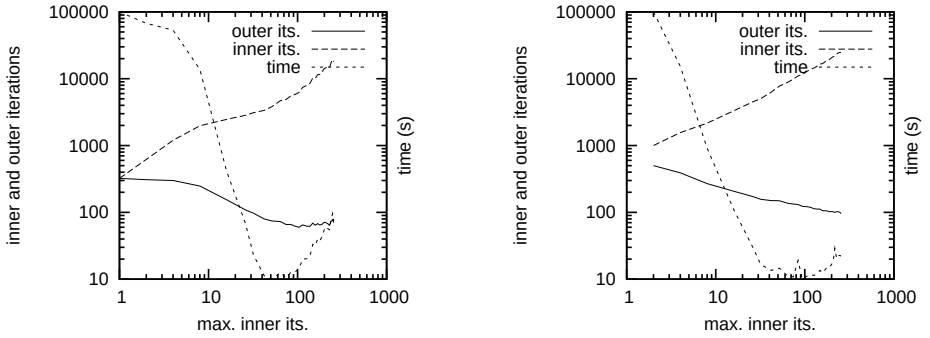

Fig. 4. Plots comparing outer iterations, inner iterations and execution time when obtaining the smallest eigenpairs for the problems *qc324* (left) and *cavity06* (right) with different maximum linear iterations of GMRES for the correction equation.

Regarding the computation of interior eigenvalues, we tried the harmonic Jacobi-Davidson without preconditioner as well as with Jacobi and block Jacobi preconditioners. The latter uses an ILU(0) decomposition in each diagonal block (one per process), although in this particular matrix this preconditioner does not improve over Jacobi. We also compared with SLEPc's Krylov-Schur with shift-and-invert, wherein the associated LU factorization is handled via MUMPS[7]. Figure 5 (second row) shows the obtained speedups. Again all tested implementations reveal speedups not far from the ideal one, except shift-and-invert Krylov-Schur that shows the poor parallel performance of the triangular backward and forward solves, which is particularly bad in this matrix.

4 Conclusions and future work

We have presented a parallel implementation of the Jacobi-Davidson method for distributed memory machines for non-Hermitian matrices, using the PETSc and SLEPc libraries. We discuss the design decisions that influence the convergence and the performance of the method, such as the stopping criterion for solving the correction equation, the eigenpair extraction method and the parallelization. The presented results suggest that it is sufficient to perform only a few GMRES iterations when searching for largest eigenvalues, whereas for interior eigenvalues it is necessary to use the harmonic extraction technique combined with much more iterations of the correction equation solver.

Furthermore, a real arithmetic version for real unsymmetric matrices has been implemented also. The results show the sequential and parallel performance improvement of this version with a convergence rate comparable to the complex arithmetic version using our proposed projectors P_1 and P_2.

At this stage of the development, the code is already able to compute exterior as well as interior eigenvalues with a reasonable efficiency despite the lack of a

[7] http://graal.ens-lyon.fr/MUMPS/

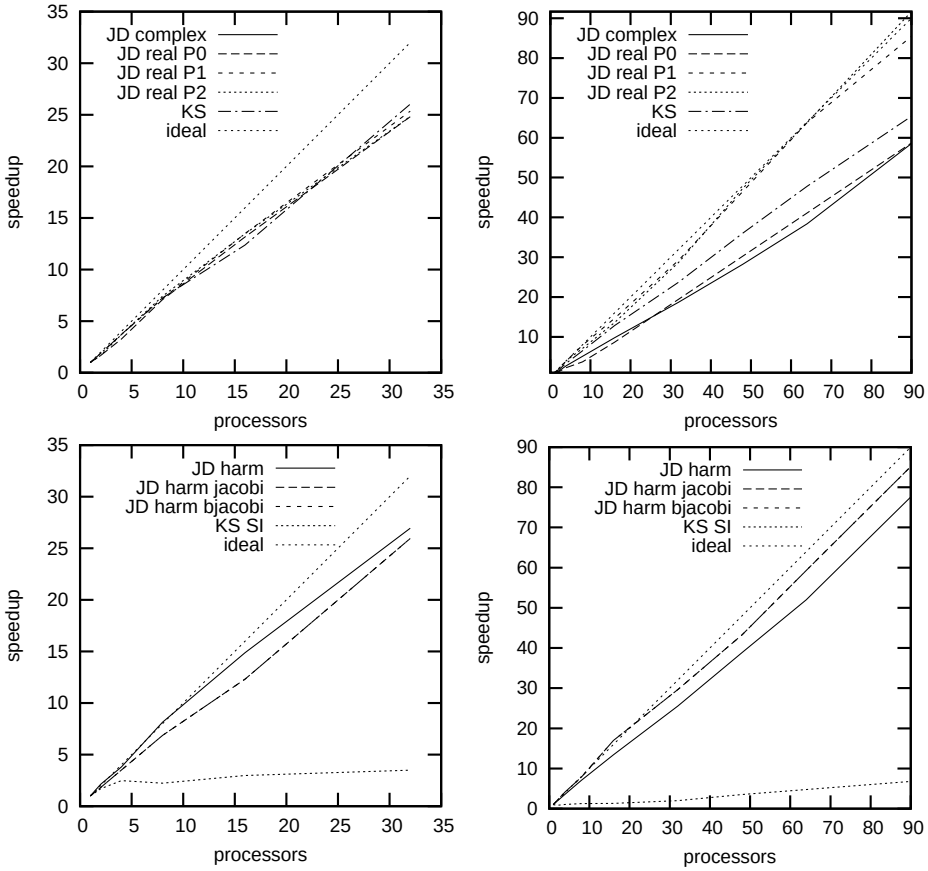

Fig. 5. Speedup on Odin (left column) and CaesarAugusta (right column) when finding the largest magnitude eigenpairs of *tmt_unsym* using the complex and real arithmetic (with projectors P_0, P_1 and P_2) versions of Jacobi-Davidson, and Krylov-Schur (first row); and when finding the eigenpairs closest to 0 of *tmt_unsym* using the harmonic version of Jacobi-Davidson without preconditioner and with Jacobi and block Jacobi preconditioners, and Krylov-Schur using shift-and-invert with MUMPS (second row).

restarting technique for the search subspace. It is able to solve a wide range of difficult test problems, showing good speedups for their parallel performance.

These developments will be added to a fully featured Jacobi-Davidson solver in SLEPc, that will include restarting, locking of converged eigenpairs, and a more elaborate adaptive stopping criterion for the correction equation solver. The possibility of estimating, inexpensively, the residual norm of the eigenvalue problem from the one of the inner linear systems, as presented in [18], is attractive and will be address in a future work.

References

1. Stathopoulos, A., McCombs, J.R.: PRIMME: PReconditioned Iterative Multi-Method Eigensolver: Methods and software description. ACM Trans. Math. Software **37**(2) (2010) 21:1–21:30
2. Bollhöfer, M., Notay, Y.: JADAMILU: a software code for computing selected eigenvalues of large sparse symmetric matrices. Comput. Phys. Commun. **177**(12) (2007) 951–964
3. Baker, C.G., Hetmaniuk, U.L., Lehoucq, R.B., Thornquist, H.K.: Anasazi software for the numerical solution of large-scale eigenvalue problems. ACM Trans. Math. Software **36**(3) (2009) 13:1–13:23
4. Arbenz, P., Becka, M., Geus, R., Hetmaniuk, U., Mengotti, T.: On a parallel multilevel preconditioned Maxwell eigensolver. Parallel Comput. **32**(2) (2006) 157–165
5. Nool, M., van der Ploeg, A.: A parallel Jacobi-Davidson-type method for solving large generalized eigenvalue problems in magnetohydrodynamics. SIAM J. Sci. Comput. **22**(1) (2000) 95–112
6. Nool, M., van der Ploeg, A.: Parallel Jacobi-Davidson for solving generalized eigenvalue problems. In: Vector and Parallel Processing – VECPAR'98. Lect. Notes Comp. Sci. (1999) 58–70
7. Hernandez, V., Roman, J.E., Vidal, V.: SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems. ACM Trans. Math. Software **31**(3) (2005) 351–362
8. Romero, E., Roman, J.E.: A parallel implementation of the Davidson method for generalized eigenproblems. In: Parallel Computing: Architectures, Algorithms and Applications. Advances in Parallel Computing, IOS Press to appear (2010).
9. Sleijpen, G.L.G., van der Vorst, H.A.: A Jacobi-Davidson iteration method for linear eigenvalue problems. SIAM Review **42**(2) (2000) 267–293
10. Jacobi, C.G.J.: Über ein leichtes verfahren die in der theorie der Säculärstörungen vorkommenden gleichungen numerisch aufzulösen. Crelle's J. **30** (1846) 51–94
11. Morgan, R.B.: Computing interior eigenvalues of large matrices. Linear Algebra Appl. **154** (1991) 289–309
12. Paige, C.C., Parlett, B.N., van der Vorst, H.A.: Approximate solutions and eigenvalue bounds from Krylov subspaces. Num. Linear Algebra Appl. **2** (1995) 115–134
13. Bai, Z., Demmel, J., Dongarra, J., Ruhe, A., van der Vorst, H., eds.: Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide. Society for Industrial and Applied Mathematics, Philadelphia, PA (2000)
14. Hernandez, V., Roman, J.E., Tomas, A.: Parallel Arnoldi eigensolvers with enhanced scalability via global communications rearrangement. Parallel Comput. **33**(7–8) (2007) 521–540
15. van Noorden, T., Rommes, J.: Computing a partial generalized real Schur form using the Jacobi-Davidson method. Num. Linear Algebra Appl. **14** (2007) 197–215
16. Sleijpen, G.L.G., van der Vorst, H.A., Meijerink, E.: Efficient expansion of subspaces in the Jacobi-Davidson method for standard and generalized eigenproblems. Electron. Trans. Numer. Anal. **7** (1998) 75–89
17. Romero, E., Roman, J.E.: A parallel implementation of the Jacobi-Davidson eigensolver and its application in a plasma turbulence code. In: EuroPar'10. to appear (2010).
18. Hochstenbach, M.E., Notay, Y.: Controlling inner iterations in the Jacobi-Davidson method. SIAM J. Matrix Anal. Appl. **31**(2) (2009) 460–477