A Parallel Implementation of the Jacobi-Davidson Eigensolver and Its Application in a Plasma Turbulence Code[⋆]

Eloy Romero and Jose E. Roman

Instituto ITACA, Universidad Politécnica de Valencia,
Camino de Vera s/n, 46022 Valencia, Spain
{eromero,jroman}@itaca.upv.es

Abstract. In the numerical solution of large-scale eigenvalue problems, Davidson-type methods are an increasingly popular alternative to Krylov eigensolvers. The main motivation is to avoid the expensive factorizations that are often needed by Krylov solvers when the problem is generalized or interior eigenvalues are desired. In Davidson-type methods, the factorization is replaced by iterative linear solvers that can be accelerated by a smart preconditioner. Jacobi-Davidson is one of the most effective variants. However, parallel implementations of this method are not widely available, particularly for non-symmetric problems. We present a parallel implementation to be released in SLEPc, the Scalable Library for Eigenvalue Problem Computations, and test it in the context of a highly scalable plasma turbulence simulation code. We analyze its parallel efficiency and compare it with Krylov-type eigensolvers.

Keywords: Message-passing parallelization, eigenvalue computations, Jacobi-Davidson, plasma simulation.

1 Introduction

We are concerned with the partial solution of the standard eigenvalue problem defined by a large, sparse matrix A of order n, $Ax = \lambda x$, where the scalar λ is called the eigenvalue, and the n-vector x is called the eigenvector. Many iterative methods are available for the partial solution of the above problem, that is, for computing a subset of the eigenvalues. The most popular ones are Krylov projection methods such as Lanczos, Arnoldi or Krylov-Schur, and Davidson-type methods such as Generalized Davidson or Jacobi-Davidson. Details of these methods can be found in [2]. Krylov methods achieve good performance when computing extreme eigenvalues, but usually fail to compute interior eigenvalues. In that case, the convergence can be improved by combining the method with a spectral transformation technique, i.e., to solve $(A - \sigma)^{-1}x = \theta x$ instead of $Ax = \lambda x$. The drawback of this approach is the added high computational cost

[⋆] This work was partially supported by the Spanish Ministerio de Ciencia e Innovación under project TIN2009-07519.

of solving large linear systems at each iteration of the eigensolver. Moreover, for stability reasons these systems must be solved very accurately (normally with direct methods). Davidson-type methods aim at reducing the cost by solving linear systems approximately, without compromising the robustness, usually with iterative methods.

Davidson methods are becoming an excellent alternative due to the possibility of tuning the balance between numerical behaviour and computational performance. A powerful preconditioner (close to the matrix inverse), if available, can usually reduce the number of iterations significantly. However, in practice its use is normally too expensive computationally and difficult to parallelize, thus dominating the cost of the eigensolver. Otherwise, depending on the performance of the matrix-vector product, the preconditioner and the orthogonalization, there exist Davidson-type variants that can be competitive with respect to Krylov-type eigensolvers. This paper illustrates an example of this.

Despite their potential benefit, it is still difficult to find freely available parallel implementations of Davidson-type eigensolvers, especially for the non-symmetric case, although there are some publications dealing with parallel implementations of these methods employed for certain applications (for instance, the Jacobi-Davidson described in [1]). Some Davidson-type methods can be found in PRIMME [19], Anasazi [3] and JADAMILU [5]. PRIMME implements almost all Davidson-type variants, Anasazi only implements a basic block Generalized Davidson method, and JADAMILU implements Jacobi-Davidson. However, none of them support non-Hermitian problems. Implementation of non-Hermitian solvers gets complicated because of the need to work with invariant subspaces rather than eigenvectors, as well as to consider both right and left eigenspaces. Our aim is to provide a robust and efficient parallel implementation of the Jacobi-Davidson method in the context of SLEPc, the Scalable Library for Eigenvalue Problem Computations [9], that can address standard and generalized problems, both Hermitian and non-Hermitian, with either real or complex arithmetic. The solver will also provide different Davidson variants other than Jacobi-Davidson. Some preliminary results have been presented in [14], where a simple non-restarted variant with real arithmetic is discussed. In this work, we focus on the restarted Jacobi-Davidson method for complex non-Hermitian problems.

The eigenvalue problem is the main algebraic problem in many areas such as structural dynamics, quantum chemistry and control theory. In this work, we show results for the eigenvalue calculation that takes place in the plasma physics application GENE, that solves a set of non-linear partial integro-differential equations in five-dimensional phase space by means of the method of lines. Because of the shape of the spectrum (see Fig. 1), computing the largest magnitude eigenvalues of the linearized operator is not particularly difficult, despite the unfavorable characteristics of the problem (complex non-Hermitian with matrix in implicit form). However, the case of computing the rightmost eigenvalues is much more difficult from the numerical point of view, since these eigenvalues are much smaller in magnitude compared to the dominant ones. This makes the computational problem challenging and suitable as a testbed for our new parallel eigensolver running on distributed memory architectures.

In section 2 we describe the Jacobi-Davidson method and several relevant variants such as harmonic extraction. The implementation details, including how the method is parallelized, are discussed in section 3. In section 4 we provide a brief description of the application. The performance of the parallel eigensolver in this application is presented in section 5.

2 The Jacobi-Davidson Method

Davidson-type methods belong to the class of subspace methods, i.e., approximate eigenvectors are sought in a search subspace $\mathcal{V}$. Each iteration of these methods has two phases: the subspace extraction, which selects the best approximated eigenpair from $\mathcal{V}$ in the desired spectrum region, and the subspace expansion, which adds a correction for the selected eigenvector to $\mathcal{V}$.

The subspace expansion distinguishes a Davidson-type variant from others. Jacobi-Davidson computes a correction t orthogonal to the selected approximate eigenvector u as an approximate solution of the so-called Jacobi orthogonal component correction (JOCC) [10] equation

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

There are ways in which (1) derives into different linear systems. For our purpose, we implement the Jacobi-Davidson correction equation studied in [7]

$$\left(I - \frac{uz^*}{z^*u}\right)(A - \theta I)\left(I - \frac{uz^*}{z^*u}\right) t = -r \ , \tag{2}$$

being $r = Au - \theta u$ the residual associated to the selected approximate eigenpair (θ, u), and $z \in \operatorname{span}\{Au, u\}$.

If (2) is solved exactly, one step of the algorithm turns out to be one step of the Rayleigh Quotient Iteration, which converges almost quadratically [7]. Otherwise, i.e., (2) is solved approximately, this high convergence rate may get lost. There is a trade-off between speed of convergence and the amount of work one is willing to spend for solving the equation, that is easily tuned if an iterative method is used. In practice, the performance of the eigensolver depends dramatically on a suitable stopping criterion for the iterative method.

In the subspace extraction phase, Davidson-type methods classically impose the Ritz-Galerkin condition to the eigenpair that will be selected (θ, u),

$$r = Au - \theta u \perp \mathcal{V} \ . \tag{3}$$

Since $u \in \mathcal{V}$, it is possible to express $u = V\tilde{u}$, being V a basis of $\mathcal{V}$. Then this leads to the low-dimensional projected eigenproblem $V^*AV\tilde{u} = \theta\tilde{u}$.

In practice this extraction technique, named Rayleigh-Ritz, obtains good convergence rates when exterior eigenvalues are desired. However, it gives poor approximate eigenvectors for interior eigenvalues. The harmonic Rayleigh-Ritz method was proposed in [11,12] as an alternative extraction technique for this case.

Assuming that interior eigenvalues close to a given target τ are desired, harmonic Rayleigh-Ritz imposes the Petrov-Galerkin condition to the selected eigenpair

$$(A-\tau)u-\xi u \perp \mathcal{W} \equiv (A-\tau)\mathcal{V}, \quad \xi=\theta-\tau \ . \tag{4}$$

For stability reasons V and W (a basis of $\mathcal{W}$) are both constructed to be orthonormal: W is such that $(A-\tau I)V=WS$, where S is upper triangular. In the same way, this leads to the projected eigenproblem

$$W^*(A-\tau)V\tilde{u}=\xi W^*V\tilde{u} \ . \tag{5}$$

Then the smallest magnitude pairs $(\xi,\tilde{u})$ of (5) correspond to the pairs $(\xi+\tau, V\tilde{u})$ in $\mathcal{V}$ closest to the target τ.

Finally, instead of directly computing the eigenpairs of the projected problem, the implementation performs the Schur decomposition and works with Schur vectors along the method. At the end, the solver has obtained a partial Schur decomposition from which it is possible to compute the corresponding approximate eigenpairs. This variant, called JDQZ [7], facilitates restarting (when the subspaces are full) and locking (when eigenpairs converge).

Due to memory limitations and in order to improve efficiency, the maximum size of the search and test subspaces have to be bounded. The thick restart technique resets the subspace with the best m_{min} approximate eigenvectors when its size achieves m_{max}. JDQZ replaces the eigenvectors by an orthogonal basis of the corresponding eigenspace, which means updating V and W with the first m_{min} right and left Schur vectors of the projected problem, respectively.

When an eigenpair converges, it is removed from the subspace bases V and W, forcing a restart without it and in the following iterations orthogonalizing the new vectors t also against all locked vectors.

Algorithm 1 summarizes the scheme of a Jacobi-Davidson method with harmonic Rayleigh-Ritz extraction and the correction equation (2). For a more detailed description, the reader is referred to [16,15,7].

3 Implementation Description

3.1 Overview of SLEPc

SLEPc, the Scalable Library for Eigenvalue Problem Computations [9][1], is a software library for the solution of large, sparse eigenvalue and singular value problems on parallel computers. It was designed to solve problems formulated in either standard or generalized form, both Hermitian and non-Hermitian, with either real or complex arithmetic.

[1] `http://www.grycap.upv.es/slepc/`

Algorithm 1. *Block Jacobi-Davidson with harmonic Rayleigh-Ritz extraction*

```
Input:   matrix $A$ of size $n$, target $\tau$, number of wanted eigenpairs $p$,
         block size $s$, maximum size of $V$ $m_{max}$, restart with $m_{min}$ vectors
Output:  resulting eigenpairs $(\widetilde{\Theta}, \widetilde{X})$

Choose an $n \times s$ full rank matrix $V$ such that $V^*V = I$
While size$(\widetilde{\Theta}) < p$
    1. Compute $W \leftarrow (A - \tau)V$
    2. Compute $G \leftarrow W^*AV$ and $H \leftarrow W^*V$
    3. Compute the eigenpairs $(\Xi, U)$ of the eigenproblem $(G, H)$ and
            sort them ascendantly
    4. Compute the first $s$ Ritz pairs, $X \leftarrow VU_{1:s}$ and $\theta_i \leftarrow \xi_i + \tau$
    5. Compute the residual vectors, $r_i \leftarrow AVu_i - Vu_i\theta_i$
    6. Test for convergence
    7. Compute the corrections, solving
            $\left(I - \frac{x_i z_i^*}{z_i^* x_i}\right)(A - \theta_i I)\left(I - \frac{x_i z_i^*}{z_i^* x_i}\right) t_i = -r_i$, with $z_i = (\bar{\tau} A + I)x_i$
    8. If size$(V) \geq m_{max}$, $V \leftarrow VU_{1:m_{min}}$
       Else if $k$ pairs are converged
            Add eigenvalues $\theta_1, \ldots, \theta_k$ to $\widetilde{\Theta}$
            $\widetilde{X} \leftarrow [\widetilde{X} \quad VU_{1:k}]$
            $V \leftarrow VU_{k+1:k'}$, where $k' = $ size$(V)$
       Else, $V \leftarrow [V \quad$ orthonormalize$([\widetilde{X} \quad V], T)]$
End while
```

SLEPc provides a collection of eigensolvers on top of PETSc (Portable, Extensible Toolkit for Scientific Computation, [4]), including Krylov-Schur, Arnoldi, Lanczos, Subspace Iteration and Power/RQI. However, SLEPc lacks Davidson-type solvers, which motivates the development of our implementation.

PETSc is a parallel framework for the numerical solution of partial differential equations, whose approach is to encapsulate mathematical algorithms using object-oriented programming techniques in order to be able to manage the complexity of efficient numerical message-passing codes.

PETSc is object-oriented in the sense that all the code is built around a set of data structures and algorithmic objects. The application programmer works directly with these objects rather than concentrating on the underlying data structures. The three basic abstract data objects are index sets, vectors and matrices. Built on top of this foundation are various classes of solver objects, including linear, nonlinear and time-stepping solvers. Many different iterative linear solvers are provided, including GMRES, BiCGstab and BiCGstab(ℓ) [18].

SLEPc eigensolvers rely on the parallel implementation of vector operations, the matrix-vector product and linear equation solvers. Basic implementations of these operations are supplied by PETSc objects. However, certain time-consuming, critical operations have custom implementation in SLEPc in order to improve the overall performance, as explained below.

3.2 Parallelization Details

The problem matrix A and the vectors of size n, such as V, W, X, $\tilde{X}$ and R are distributed by blocks of rows. The rest of vectors and matrices of size bounded by $m_{max} \ll n$, such as G, H, U, Ξ, $\widetilde{\Theta}$, are replicated in all nodes.

Operations involving distributed operands are parallelized. These include updating W, computing the projected eigensystem (G, H), the selected Ritz vectors X and their residuals R, solving the correction equation (2) and orthogonalizing V. The orthogonalization is based on a variant of classical Gram-Schmidt with selective reorthogonalization, providing both numerical robustness and good parallel efficiency [8]. The eigendecomposition of the projected problem and other minor computations are replicated in all nodes.

PETSc only provides basic support for multivectors (a multivector can be seen as a thin tall matrix, or a set of vectors that should be stored contiguously for memory efficiency). In order to develop an optimized version of Jacobi-Davidson it is necessary to implement basic multivector operations using BLAS to perform the local calculations. We provide an implementation in which individual vectors in a multivector can be used in common PETSc functions.

The most time-consuming operations are the multivector inner product W^*V and the updating VU. However, PETSc only implements the level 2 BLAS operations W^*v_i and Vu_i. For W^*V, our implementation (i) performs the level 3 BLAS matrix-matrix product of the locally stored parts of V and W on each processor, and then (ii) sums up all of them with a single call to an MPI routine. For VU, it performs the BLAS matrix-matrix product of the locally stored part of V and the whole U.

3.3 Solution of the Correction Equation

The correction equation (2) is solved using PETSc's Krylov linear solvers, which need to compute matrix-vector products with the coefficient matrix. In this case, performing the shifting and the projections implicitly is more efficient than explicitly building the coefficient matrix. Also, applying only 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 [17].

The performance and the global convergence trade-off is controlled in two ways. First, the maximum number of iterations and the relative residual tolerance can be tuned for the linear system solver. Generally, increasing the number of linear solver iterations (inner iterations) causes a decrease of the global method iterations (outer iterations). In this work, only the static stopping criterion is tested.

Secondly, the convergence behaviour of different Krylov solvers depends on the properties of the problem. We have tested two important solvers of this family: GMRES and BiCGstab(ℓ). They have different parallel behaviour because GMRES generally requires less matrix-vector products than BiCGstab(ℓ), but in contrast it has to explicitly maintain an orthonormalized basis whereas BiCGstab does not.

Furthermore, our implementation incorporates the strategy described in [7] for computing z_i as the result of normalizing $\bar{\tau}Ax_i + x_i$.

Finally, it is noteworthy that in the first outer steps the pairs resulting from the (harmonic) Rayleigh-Ritz procedure are usually poor approximations of the desired eigenpairs, and the target τ may be a relatively better approximation. Therefore, when the selected eigenpair's associated residual norm is greater than a threshold value *fix*, the correction equation (2) is solved with $\theta = \tau$ instead [7, Section 4.0.1].

The preconditioning of the equation is desirable. However, this issue is not addressed in this work because efficient preconditioners are not available in PETSc that support implicit matrices like in the GENE code case.

4 GENE

One of the main goals of plasma simulation is to study the micro-instabilities that drive turbulence which in turn produces anomalous transport. This analysis must be done to determine the energy confinement time, a crucial parameter for the design of a fusion reactor.

GENE [6] is a massively parallel plasma simulation code written in Fortran 90/95, which is based on the numerical solution of the gyrokinetic equations. These equations stem from a simplification of the Maxwell-Boltzmann equations by eliminating the fast gyration of ions and electrons in strongly magnetized, dilute plasmas. This periodic motion is not relevant for most investigations, usually focusing on observables related to much slower time scales such as the net particle transport.

The linearized gyrokinetic equation can be written schematically as

$$\frac{\partial g}{\partial t} = \mathcal{L}[g], \tag{6}$$

where $\mathcal{L}$ is a time independent, complex, non-Hermitian integro-differential operator. It describes the time evolution of the modified distribution function of the gyrocentres g, which is a (scalar) function of the perpendicular spatial wave vector $(k_x,\ k_y)$, the coordinate z parallel to the magnetic field, the velocity parallel to the magnetic field $v_{\|}$, the magnetic moment μ, and the species label j. The GENE code follows an Eulerian approach. In particular, an explicit Runge-Kutta scheme is used for time integration, while the semi-discretization of phase space variables is done with a fixed grid and a combination of spectral and finite difference techniques. In GENE, the operator matrix is never computed explicitly, but implemented in a highly parallelized and efficient matrix-free form. For reasonably accurate models, the size of the problem ranges from several hundred thousand for linear simulations up to a few billion for nonlinear problems.

There is the need of computing some selected eigenvalues of the linearized operator. In [13], the Krylov eigensolvers available in SLEPc are used for this. One scenario is the computation of the largest magnitude eigenvalue in order to estimate the optimal timestep of the initial value solver. In this case, Krylov

Table 1. GENE test configuration: very unstable kinetic ballooning mode with growth rate of 0.2055 and frequency of 0.2872 and another unstable mode $(0.1227 - 0.4494i)$

GENE

Direction	Par. divisions	Resol.	Boxsize	
s	SCAN	2		
x	1	12	l_x	125.628
y	1	1	$k_{y,min}$	0.25
z	SCAN	16		
v	SCAN	48	l_v	3.0
μ	SCAN	8	l_μ	9.0

Other params.		*Geometry*		Param.	Ions	Electrons
β	0.001	$\hat{s}$	0.8	R/L_n	2.0	2.0
hyp_z	2	q_0	1.4	R/L_T	3.125	3.375
hyp_v	0.5	trpeps	0.18	*mass*	1.0	0.00027
				charge	1.0	-1.0
				T	1.0	1.5
				dens	1.0	1.0

SLEPc

which	closest to target
target	1
nev	2
ncv	64
tol	10^{-5}
harmonic	yes

SLEPc JD

initv	5
minv	8
block size	1
fix	10^{-3}
KSP type	gmres/bcgsl
KSP max its	110

solvers converge very fast. In a different context, SLEPc is also used for computing the subdominant unstable modes, i.e., the rightmost eigenvalues. Due to the shape of the spectrum (see Fig. 1 for an example), these eigenvalues are much more difficult to compute. In [13], it is shown that the Krylov-Schur method with harmonic extraction has a reasonably good performance, compared to plain Krylov-Schur with spectral transformation. In the next section, we will show that our Jacobi-Davidson implementation performs even better.

5 Results

This section summarizes the experiments carried out in order to evaluate the convergence behaviour and the parallel performance of our implementation.

The tests are executed on CaesarAugusta, a machine 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. For the tests, only 128 processors are used due to account limitations.

The following software is employed: GENE 1.4, PETSc 3.0.0-p9, SLEPc development version (the new Jacobi-Davidson will be included in the next release) and LAPACK 3.2.1. All of them are built with the IBM compilers XL C and Fortran, and linked with the BLAS routines in ESSL and MPICH 1.2.7.

The set of parameters specified in the GENE input file is detailed in Table 1 and can be considered a real use scenario. The impact of the *Par. divisions* parameters, which determine how many groups of processors there are in each direction, is studied in [13].

The tested eigensolvers are the Jacobi-Davidson method described previously and harmonic Krylov-Schur. Both of them are configured for computing the two largest real (rightmost) eigenvalues, with a tolerance of 10^{-5} for the residual

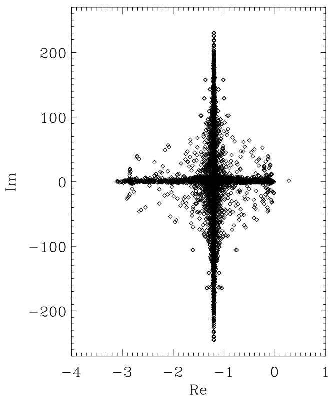

Fig. 1. Spectrum of the linearized operator of a GENE problem similar to the one described in Table 1

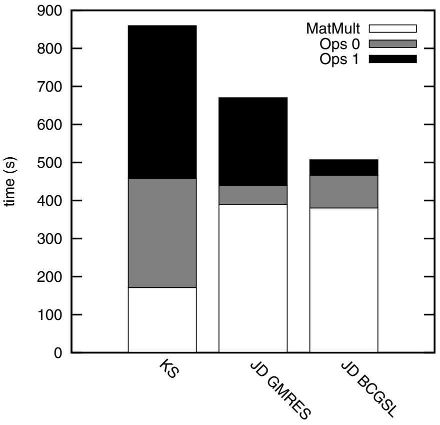

Fig. 2. Sequential time (in seconds) using Krylov-Schur (KS) and Jacobi-Davidson (JD) with GMRES and BiCGstab(ℓ). The total time is split into matrix-vector products (MatMult) and vector operations without (Ops 0) and with (Ops 1) communication.

norm relative to the eigenvalue. The search subspace is limited to 64 vectors, but Krylov-Schur restarts with 32 vectors and Jacobi-Davidson with 8. Regarding the linear solvers, GMRES(30) and BiCGstab(2) are tested for solving the correction equation (2). Both of them are set to perform 110 iterations per outer iteration. The values of these parameters have been chosen in order to maximize the performance of the solver.

The sequential experiments show that Jacobi-Davidson is faster than Krylov-Schur. Figure 2 summarizes the time spent by matrix-vector products and vector operations requiring communications or not. Krylov-Schur performs less matrix-vector products but spends more time orthogonalizing than Jacobi-Davidson. The GMRES solver also orthogonalizes, but it has a smaller search subspace besides using classical Gram-Schmidt without reorthogonalization. This may explain why Jacobi-Davidson with GMRES is faster than Krylov-Schur but slower than using BiCGstab(2), which does not orthogonalize at all.

Figure 3 illustrates the speedup of Jacobi-Davidson with BiCGstab (left top plot) and Krylov-Schur (right top plot) when solving the test case with different distribution of the problem domain among the processors, that significantly affects the matrix-vector performance (left bottom plot). This means that not all the directions of the five-dimensional phase space are equally parallelizable. We found that the best results are obtained when putting as many processors as possible in the μ direction, the same conclusion as in [13]. It is also observed that Jacobi-Davidson is slightly more sensitive to this variability of the matrix-vector product speedup, due to its greater use. It can be noticed that the sequential

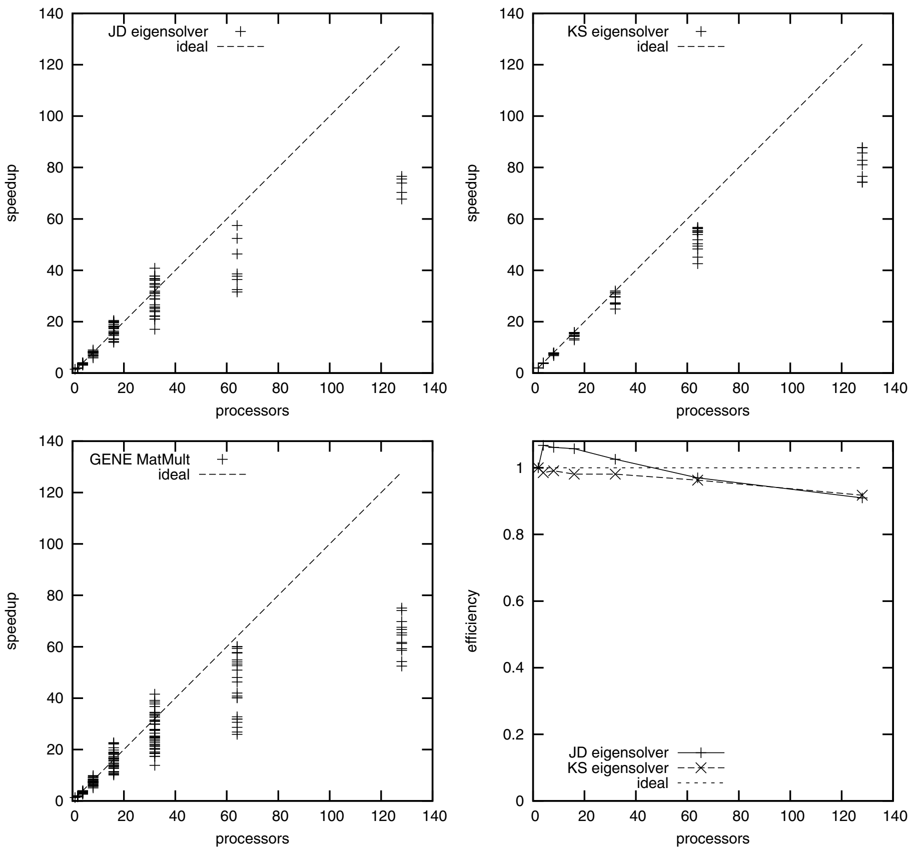

Fig. 3. Speedups when solving the problem of Table 1 with Jacobi-Davidson (left top) and Krylov-Schur (right top) with different matrix-vector product speedups (left bottom). Scaled efficiency of both eigensolvers (right bottom) also on CaesarAugusta.

gain factor of Jacobi-Davidson gets exhausted when increasing the number of processors because of the fact that Krylov-Schur relies on operations performing better than the matrix-vector product. However, none of the executed tests had a negative gain factor.

The first three plots show speedup in the hard scaling sense, that is, with a fixed problem size. In order to illustrate the weak scaling scenario, an example of scaled efficiency (pE_p where E_p is the standard efficiency for p processes) when increasing the resolution in the v direction, is also shown (right bottom plot in Fig. 3), where both eigensolvers are comparably good.

6 Conclusions

We have presented a parallel implementation of the Jacobi-Davidson eigensolver for complex non-Hermitian matrices. In particular, the proposed solver has interesting features for finding interior eigenvalues, such as harmonic extraction and special attention to the Jacobi-Davidson correction equation (the linear system method, the projectors and the fix parameter). The implementation has been carried out in the context of SLEPc, where the user is able to easily adjust the different parameters for the best performance.

In order to analyze the performance of the new solver, we have addressed a relevant scientific computing application, namely the plasma turbulence simulation as implemented in the GENE code. This application requires computing the rightmost eigenvalues (unstable modes) of a discretized advection dominated partial integro-differential equation, where the matrix has almost pure imaginary eigenvalues. This problem is a challenge for iterative eigensolvers.

The comparison, in terms of sequential computing time, with the harmonic Krylov-Schur method is very favorable, with an improvement of about 40%. However, this gain is mitigated when it comes to parallel efficiency, where both methods become nearly equivalent for a large number of processors, although in absolute terms Jacobi-Davidson is still faster. The reason for this behaviour is that the Jacobi-Davidson eigensolver, in order to be competitive with respect to Krylov-Schur, needs to perform much more inner iterations than outer iterations. In this way, the cost is dominated by the matrix-vector product operation. In the considered application, the parallel efficiency of the matrix-vector product is rather variable, depending on which direction the parallelization is done.

The newly developed solver will be included in a future version of SLEPc. The final version of the solver will include even more interesting features such as real arithmetic for non-symmetric problems and an adaptive stopping criterion for the correction equation solver. The latter will be especially important for GENE and similar applications. Another topic for further research is the design of a specific preconditioner for the GENE linearized operator.

Acknowledgements. The authors thankfully acknowledge the computer resources and assistance provided by the Barcelona Supercomputing Center (BSC). We also thank Florian Merz for his suggestions on an early draft of the paper.

References

1. Arbenz, P., Becka, M., Geus, R., Hetmaniuk, U., Mengotti, T.: On a parallel multilevel preconditioned Maxwell eigensolver. Parallel Computing 32(2), 157–165 (2006)
2. 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 (2000)
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 Transactions on Mathematical Software 36(3) 13:1–13:23 (2009)
4. Balay, S., Buschelman, K., Eijkhout, V., Gropp, W., Kaushik, D., Knepley, M., McInnes, L.C., Smith, B., Zhang, H.: PETSc users manual. Tech. Rep. ANL-95/11 - Revision 3.0.0, Argonne National Laboratory (2008)

5. Bollhöfer, M., Notay, Y.: JADAMILU: a software code for computing selected eigenvalues of large sparse symmetric matrices. Computer Physics Communications 177(12), 951–964 (2007)
6. Dannert, T., Jenko, F.: Gyrokinetic simulation of collisionless trapped-electron mode turbulence. Physics of Plasmas 12(7), 072309 (2005)
7. Fokkema, D.R., Sleijpen, G.L.G., van der Vorst, H.A.: Jacobi–Davidson style QR and QZ algorithms for the reduction of matrix pencils. SIAM Journal on Scientific Computing 20(1), 94–125 (1999)
8. Hernandez, V., Roman, J.E., Tomas, A.: Parallel Arnoldi eigensolvers with enhanced scalability via global communications rearrangement. Parallel Computing 33(7-8), 521–540 (2007)
9. Hernandez, V., Roman, J.E., Vidal, V.: SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems. ACM Transactions on Mathematical Software 31(3), 351–362 (2005)
10. Jacobi, C.G.J.: Über ein leichtes Verfahren die in der Theorie der Säculärstörungen vorkommenden Gleichungen numerisch aufzulösen. Crelle's J. 30, 51–94 (1846)
11. Morgan, R.B.: Computing interior eigenvalues of large matrices. Linear Algebra and its Applications, 154–156, 289–309 (1991)
12. Paige, C.C., Parlett, B.N., van der Vorst, H.A.: Approximate solutions and eigenvalue bounds from Krylov subspaces. Numerical Linear Algebra with Applications 2(2), 115–133 (1995)
13. Roman, J.E., Kammerer, M., Merz, F., Jenko, F.: Fast eigenvalue calculations in a massively parallel plasma turbulence code. In: Parallel Computing (to appear, 2010), doi:10.1016/j.parco.2009.12.001
14. Romero, E., Cruz, M.B., Roman, J.E., Vasconcelos, P.B.: A Jacobi-Davidson parallel implementation for unsymmetric eigenproblems. In: Vector and Parallel Processing – VECPAR 2010 (to appear, 2010)
15. Sleijpen, G.L.G., Booten, A.G.L., Fokkema, D.R., van der Vorst, H.A.: Jacobi-Davidson type methods for generalized eigenproblems and polynomial eigenproblems. BIT Numerical Mathematics 36(3), 595–633 (1996)
16. Sleijpen, G.L.G., van der Vorst, H.A.: A Jacobi–Davidson iteration method for linear eigenvalue problems. SIAM Journal on Matrix Analysis and Applications 17(2), 401–425 (1996)
17. 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. Electronic Transactions on Numerical Analysis 7, 75–89 (1998)
18. Sleijpen, G., Fokkema, D.: BiCGstab(ℓ) for linear equations involving unsymmetric matrices with complex spectrum. Electronic Transactions on Numerical Analysis 1, 11–32 (1993)
19. Stathopoulos, A., McCombs, J.R.: PRIMME: PReconditioned Iterative Multi-Method Eigensolver: Methods and software description. Tech. Rep. WM-CS-2006-08, College of William & Mary (2006)