A Parallel Scalable PETSc-Based Jacobi-Davidson Polynomial Eigensolver with Application in Quantum Dot Simulation

Zih-Hao Wei[1], Feng-Nan Hwang[1], Tsung-Ming Huang[2], and Weichung Wang[3]

[1] Department of Mathematics, National Central University, Jhongli 320, Taiwan,
socrates.wei@gmail.com; hwangf@math.ncu.edu.tw
[2] Department of Mathematics, National Taiwan Normal University, Taipei 116, Taiwan,
min@math.ntnu.edu.tw
[3] Department of Mathematics, National Taiwan University, Taipei 106, Taiwan,
wwang@math.ntu.edu.tw

Summary. The Jacobi-Davidson (JD) algorithm recently has gained popularity for finding a few selected interior eigenvalues of large sparse polynomial eigenvalue problems, which commonly appear in many computational science and engineering PDE based applications. As other inner–outer algorithms like Newton type method, the bottleneck of the JD algorithm is to solve approximately the inner correction equation. In the previous work, [Hwang, Wei, Huang, and Wang, A Parallel Additive Schwarz Preconditioned Jacobi-Davidson (ASPJD) Algorithm for Polynomial Eigenvalue Problems in Quantum Dot (QD) Simulation, Journal of Computational Physics (2010)], the authors proposed a parallel restricted additive Schwarz preconditioner in conjunction with a parallel Krylov subspace method to accelerate the convergence of the JD algorithm. Based on the previous computational experiences on the algorithmic parameter tuning for the ASPJD algorithm, we further investigate the parallel performance of a PETSc based ASPJD eigensolver on the Blue Gene/P, and a QD quintic eigenvalue problem is used as an example to demonstrate its scalability by showing the excellent strong scaling up to 2,048 cores.

1 Introduction

Many applications in computational science and engineering modeled by partial differential equations (PDEs) requires fast, accurate numerical solutions to the large-scale polynomial eigenvalue problems (EVPs), e.g., generalized EVPs in the linear stability analysis of incompressible flows and magnetohydrodynamics [4, 12, 13], quadratic EVPs in the vibration analysis of a fast train or the acoustic problem with damping [3, 5], and cubic or quintic EVPs in the estimate of discrete energy states and wave functions of the semiconductor quantum dot with non-parabolic band structure [9, 10].

The Jacobi-Davidson (JD) algorithm originally proposed by Sleijpen and Van der Vorst for linear EVPs, now has gained popularity for solving polynomial EVPs due to several advantages. For examples, without recasting the polynomial EVPs as an enlarged linearized EVPs, one only needs to deal with the problem as the same size of the original one and the interior eigenvalues are targeted without using computational expensive shift-and-invert techniques. Moreover, the JD algorithm is parallelizable, hence it is suitable for large-scale eigenvalue computations.

The JD algorithm belongs to a class of subspace methods, which consists of two key steps: one first enlarges a subspace or a so-called search space by adding a new basis vector and then extract an approximate eigenpair from the search space through the Rayleigh-Ritz procedure. To obtain a new basis vector for the search space, at each JD iteration, one needs to solve approximately a large sparse linear system of equations, which is known as the correction equation, by an iterative method. In [8] the authors proposed a new algorithm, namely the additive Schwarz preconditioned Jacobi-Davidson algorithm (ASPJD) that imports an idea from the area of parallel Schwarz-Krylov solver to enhance the parallel scalability of the JD algorithms. The Schwarz methods [14] have been widely used and is well-understood for solving a variety of linear systems arising from the discretization of PDEs and is applied to nonlinear systems as a linear preconditioner for the Jacobian system in the Newton-Krylov-Schwarz algorithm [2] or as a nonlinear preconditioner in the additive Schwarz preconditioned inexact Newton algorithm [7]. On the other hand, however only a few studies are available in the literature for solving eigenvalue problems using Schwarz methods, e.g., Schwarz methods employed as the action of the spectral transformation in the Arnoldi methods for generalized EVPs [13] or as a preconditioner in the locally optimal block preconditioned conjugate gradient method [11].

In this paper, we continue the previous work investigating how the ASPJD algorithm performs on a parallel machine with a large number of processors, e.g., the Blue Gene/P. One of our target applications is a quintic polynomial EVPs arising from the semiconductor quantum dot simulation [7].

2 A Description of the ASPJD Algorithm

In this section, we briefly describe the ASPJD algorithm for solving polynomial eigenvalue problems of degree τ, which take the form of

$$\mathcal{A}(\lambda)x = \sum_{i=0}^{\tau} \lambda^i A_i x = 0, \tag{1}$$

where $A_i \in \mathbb{R}^{n\times n}$ are the large sparse matrices arising from some discretization of certain PDEs, $\lambda \in \mathbb{C}$ is an eigenvalue and $x \in \mathbb{C}^n$ is the corresponding eigenvector. The detailed algorithm in conjunction with other techniques, such as locking and restarting can be found in [8]. Let V be the current search space. Assume that (λ, u) is current the approximate eigenpair, which is not close enough to the exact one, (λ^*, u^*). Then the next eigenpair (λ_{new}, u_{new}) can be obtained through the following two steps:

Step 1 Update the search space $V = [V, v]$ by solving the **correction equation**.

$$\left(I - \frac{pu^*}{u^*p}\right)\mathcal{A}(\lambda)(I - uu^*)t = -r$$

approximately for $t \perp u$ by a Krylov subspace method with a preconditioner B_d^{-1} defined as

$$B_d = \left(I - \frac{pu^*}{u^*p}\right)B(I - uu^*) \approx \left(I - \frac{pu^*}{u^*p}\right)\mathcal{A}(\lambda)(I - uu^*)$$

Here $r = \mathcal{A}(\lambda)$ and $p = \mathcal{A}'(\lambda)u$, where $\mathcal{A}'(\theta) = \sum_{i=1}^{\tau} i\theta^{i-1}A_i$. Then t is orthogonalized against V, and v is defined as $v = t/\|t\|_2$.

Step 2 Perform the **Rayleigh-Ritz procedure** to extract (λ_{new}, u_{new}) from the search space V by solving the small projected PEP, $(V^T\mathcal{A}(\theta)V)s = 0$. Then set $\lambda_{new} = \theta$ and compute $u_{new} = Vs$.

In practice, one does not explicitly form B_d to perform the preconditioning operation, $z = B_d^{-1}y$ with $z \perp u$ for a given y, as it can be done equivalently by computing

$$z = B^{-1}y - \eta B^{-1}p, \text{ with } \eta = \frac{u^*B^{-1}y}{u^*B^{-1}p}$$

Note that the preconditioning operation $B^{-1}p$ and inner product $u^*B^{-1}p$ need to be computed only once for solving each correction equation and there is no need to re-compute them in the Krylov subspace iteration. Furthermore, in the ASPJD algorithm, the construction of the preconditioner B^{-1} is based on an additive Schwarz framework defined as follows.

Let $S = \{1, 2, ..., n\}$ be an index set and let each integer corresponds to one component of the eigenvector. Let $S_1, S_2, ..., S_N$ be an non-overlapping partition of S, i.e.

$$\cup_{i=1}^{N} S_i = S \quad \text{and} \quad S_i \cap S_j = \emptyset \quad i \neq j$$

To obtain an overlapping partition of S, we extend each S_i to a larger subset S_i^δ with the size of n_i, i.e. $S_i \subset S_i^\delta$. Here δ is a positive integer indicating the degree of overlap and in general $\sum_{i=1}^{N} n_i \geq n$. Using the overlapping partitions of S we define a subspace of $\mathbb{R}^n$, V_i^δ as

$$V_i^\delta = \{v | v = (v_1, ..., v_n)^T \in \mathbb{C}^n, v_k = 0 \text{ if } k \notin S_i^\delta\},$$

and the corresponding restriction operators, R_i^δ, which transfers data from $\mathbb{C}^n$ to V_i^δ. Then, the interpolation operator $(R_i^\delta)^T$ can be defined as the transpose of R_i^δ. Using the restriction operator, we define the one-level restricted additive Schwarz (RAS(δ)) preconditioner with the degree of overlapping δ as

$$B^{-1} = \sum_{i=1}^{N_s} (R_i^0)^T B_i^{-1} R_i^\delta,$$

where B_i^{-1} is the subspace inverse of B_i and $B_i = R_i^\delta \mathcal{A}(\lambda)(R_i^\delta)^T$. Note that the block Jacobi preconditioner can be considered as a special case of the RAS preconditioner by setting the level of overlap equal to 0.

In the second step, we compute the eigenpair of the projected eigenvalue problem, $(V^T\mathcal{A}(\theta)V)s = 0$, by solving the corresponding linearized projected eigenvalue problem,

$$M_A z = \theta M_B z, \tag{2}$$

where

$$M_A = \begin{bmatrix} 0 & I & 0 & \dots & 0 \\ 0 & 0 & I & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & I \\ M_0 & M_1 & M_2 & \dots & M_{\tau-1} \end{bmatrix},$$

$$M_B = \begin{bmatrix} I & 0 & 0 & \dots & 0 \\ 0 & I & 0 & \dots & 0 \\ 0 & 0 & I & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & -M_\tau \end{bmatrix}, z = \begin{bmatrix} s \\ \theta s \\ \theta^2 s \\ \vdots \\ \theta^{\tau-1} s \end{bmatrix}.$$

Here $M_i = V^T A_i V$. Note that the dimension of $V^T\mathcal{A}(\theta)V$ is usually small and not larger than a user defined restarting number.

3 A PETSc-Based ASPJD Polynomial Eigensolver

The ASPJD algorithm was implemented using two powerful scientific software libraries, namely the PETSc [1] and the SLEPc [6]. As shown in Fig. 1, the design of PETSc adopts the principle of *software layering*. As an application code of PETSc, the major component in our ASPJD polynomial eigensolver, the JD object, is built on top of the KSP, a Linear Equation Solver. All PETSc libraries are based on Message Passing Interface (MPI) and two modules of linear algebra libraries: Basic Linear Algebra Subproblems (BLAS) and Linear Algebra Packages (LAPACK) library. The vector (Vec) and matrix (Mat) are two basic objects in PETSc. The eigenvector x and other working vectors are created as parallel vectors in the Vec object. The column vectors of V are stored as an array of parallel vectors. The coefficient matrices A_i and the matrix $\mathcal{A}(\theta)$ are created in a parallel sparse matrix format. We do explicitly form $\mathcal{A}(\theta)$ using parallel matrix–matrix addition and it is used in the construction of a RAS type preconditioner.

The fully parallel correction equation solve as described in Step 1 is the kernel of the JD algorithm. The ASPJD eigensolver employs a Krylov subspace method, such as GMRES or CG, which is provided by PETSc, in conjunction with the preconditioner, B_d^{-1}, where the RAS preconditioner B^{-1} is set to be a default one. For

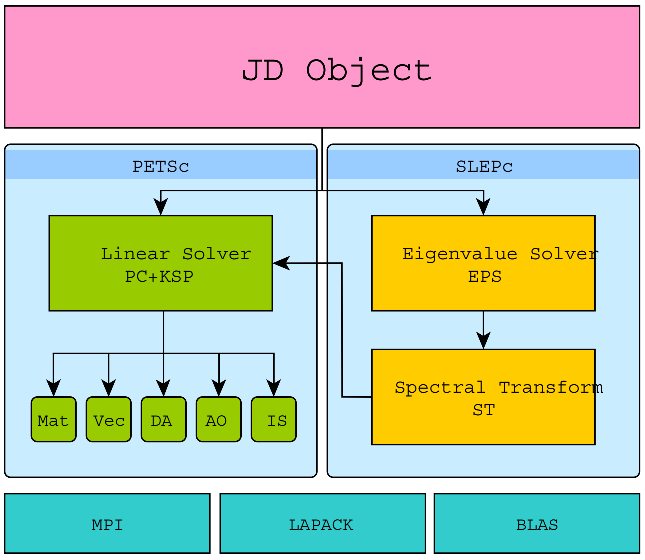

Fig. 1. The organization of PETSc, SLEPc, and the ASPJD eigensolver.

simplicity, in our current implementation both of the construction and the application of RAS are done internally by PETSc.

On each processor, the sequential QZ routine, called ZGGEVX in LAPACK, is employed to redundantly solve the same linearized projected eigenvalue problem, $M_A z = \theta M_B z$, through an interface provided by SLEPc [6]. Here, the matrices M_A and M_B, as well as M_i, are stored in the sequential dense matrix format and their sizes increase as ASPJD iterates.

4 Numerical Results

To demonstrate the scalability of our newly developed ASPJD eigensolver, we consider a quintic QD eigenvalue problem as a test case. The eigenvalue problem is derived from the second order finite volume discretization of the time-independent Schrödinger equation with non-parabolic effective mass, which is used to model a pyramidal InAs dot embedded in a cuboid GaAs matrix. The size of the resulting quintic QD eigenvalue problem is about 32 millions.

The numerical experiment was performed on the Blue Gene/P and all computation were done in double precision complex arithmetic. We claim that the JD iterations converge to an eigenpair if the absolute or the relative residuals $\|\mathcal{A}(\lambda)x\|$ is less than 10^{-10}. $V_{ini} = (1, 1, \ldots, 1)^T$ is normalized and set to be in the initial search space. We report the numerical results obtained by using the ASPJD algorithm, where the correction equation is solved by right 20 (or 40) steps RAS(0) preconditioned GMRES incorporate with the ILU(0) as a subdomain solver for finding the smallest positive eigenvalue.

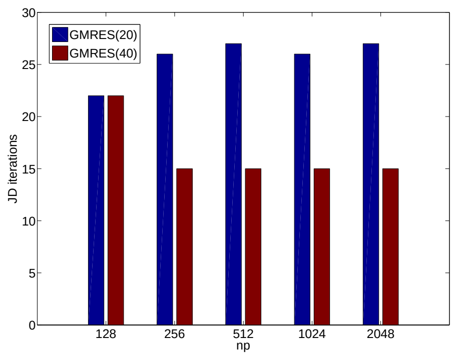

Fig. 2. The number of JD iterations with respect to np for the case of GMRES(20) and GMRES(40) as the correction equation solver.

Figure 2 shows the number of JD iterations of the ASPJD eigensolver with respect to the number of processor np, ranging from $np = 128$ to $np = 2{,}048$. We observe that except for the case of $np = 128$, the ASPJD eigensolver is quite algorithmically scalable, i.e., while the number of inner correction equation iterations is kept constant, the number of outer JD iterations remains almost the same with 26 and 15 JD iterations required to achieve convergence for the cases of GMRES(20) and GMRES(40), respectively. We may conclude that for this particular case, the number of JD iterations only depends on the number of GMRES iterations to be employed. A similar observation is made in [8] for the same test case but with a small size (about 1.5 M) and solved by the smaller number of processors (about $np = 320$).

It should be noted that the QD eigenvalue problem we consider has a special structure such that the eigenvectors corresponding to the eigenvalues of interest are localized to the dot. That is, the components of the eigenvector corresponding to the matrix (outside of the QD) are mostly zero. In our simulations, the ratio of the cuboid matrix to the pyramidal dot is about $35 : 1$ in the computational domain. Consequently, that is why we are able to decouple the original pyramidal QD eigenvalue problem problem into many subproblems using RAS(0) without a penalty in terms of an increased number of the JD iterations.

Figure 3 exhibits a very good strong scaling result for our ASPJD eigensolver for up to 2,048 processors. Note that by the definition, strong scaling means the execution time decreases in inverse proportion to the number of processors, provided that the problem size is fixed. In the ideal case, the slope of the curve is expected to be -1. The parallel efficiency for the case of GMRES(40) is about 80% based on the timing result obtained by using $np = 256$. Using a better grid partitioning and taking the design of the network topology of the BG/P into account to reduce the communication cost might further improve the parallel scalability of the ASPJD eigensolver.

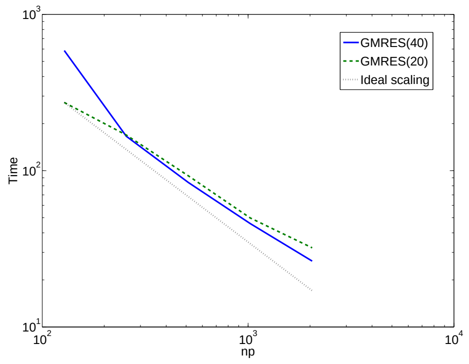

Fig. 3. Strong scalability of ASPJD on BG/P.

Acknowledgement. The authors are grateful to the BG/P computer sources provided by IBM during the workshop on computational science: IBM research and BG/P held at National Taiwan University during summer 2009. This work is partially supported by the National Science Council, the Taida Institute of Mathematical Sciences, and the National Center for Theoretical Sciences in Taiwan.

References

1. S. Balay, K. Buschelman, W.D. Gropp, D. Kaushik, M.G. Knepley, L.C. McInnes, B.F. Smith, and H. Zhang. PETSc webpage, 2010. http://www.mcs.anl.gov/petsc.
2. X.-C. Cai, W.D. Gropp, D.E. Keyes, R.G. Melvin, and D.P. Young. Parallel Newton-Krylov-Schwarz algorithms for the transonic full potential equation. *SIAM J. Sci. Comput.*, 19(1):246–265, 1998.
3. K.-W.E. Chu, T.-M. Hwang, W.-W. Lin, and C.-T. Wu. Vibration of fast trains, palindromic eigenvalue problems and structure-preserving doubling algorithms. *J. Comput. Appl. Math.*, 219:237–252, 2008.
4. K. Cliffe, H. Winters, and T. Garratt. Is the steady viscous incompressible two-dimensional flow over a backward-facing step at Re= 800 stable? *Int. J. Numer. Methods Fluids*, 17:501–541, 1993.
5. M.B. Van Gijzen. The parallel computation of the smallest eigenpair of an acoustic problem with damping. *Int. J. Numer. Methods Eng.*, 45:765–777, 1999.
6. V. Hernandez, J.E. Roman, and V. Vidal. SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems. *ACM Trans. Math. Softw.*, 31:351–362, 2005.

7. F.-N. Hwang and X.-C. Cai. A parallel nonlinear additive Schwarz preconditioned inexact Newton algorithm for incompressible Navier-Stokes equations. *J. Comput. Phys.*, 204:666–691, 2005. URL http://www.sciencedirect.com/science/article/B6WHY-4DVW0FD-3/2/17056653526b99d086bd799b21da26e4.
8. F.-N. Hwang, Z.-H. Wei, T.-M. Huang, and W. Wang. A parallel additive Schwarz preconditioned Jacobi-Davidson algorithm for polynomial eigenvalue problems in quantum dot simulation. *J. Comput. Phys.*, 229:2932–2947, 2010.
9. T.-M. Hwang, W.-W. Lin, J.-L. Liu, and W. Wang. Jacobi-Davidson methods for cubic eigenvalue problems. *Numer. Linear Algebra Appl.*, 12:605–624, 2005. URL http:// dx.doi.org/10.1002/nla.423.
10. T.M. Hwang, W.C. Wang, and W. Wang. Numerical schemes for three-dimensional irregular shape quantum dots over curvilinear coordinate systems. *J. Comput. Phys.*, 226(1):754–773, 2007.
11. A.V. Knyazev. Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM J. Sci. Comput.*, 23:517–541, 2001.
12. M. Nool and A. van der Ploeg. A parallel Jacobi-Davidson-type method for solving large generalized eigenvalue problems in magnetohydrodynamics. *SIAM J. Sci. Comput.*, 22:95–112, 2000.
13. R.P. Pawlowski, A.G. Salinger, J.N. Shadid, and T.J. Mountziaris. Bifurcation and stability analysis of laminar isothermal counterflowing jets. *J. Fluid Mech.*, 551:117–139, 2006.
14. B.F. Smith, P.E. Bjørstad, and W. Gropp. *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press, Cambridge, 1996.