A Domain Decomposition Based Jacobi-Davidson Algorithm for Quantum Dot Simulation

Tao Zhao[1], Feng-Nan Hwang[2] and Xiao-Chuan Cai[3]

Abstract In this paper, we develop an overlapping domain decomposition (DD) based Jacobi-Davidson (JD) algorithm for a polynomial eigenvalue problem arising from quantum dot simulation. Both DD and JD have several adjustable components. The goal of the work is to figure out if it is possible to choose the right components of DD and JD such that the resulting approach has a near linear speedup for a fine mesh calculation. Through experiments, we find that the key is to use two different coarse meshes. One is used to obtain a good initial guess that helps to achieve quadratic convergence of the nonlinear JD iterations. The other guarantees scalable convergence of the linear solver of the correction equation. We report numerical experiments carried out on a supercomputer with over 10,000 processors.

1 Introduction

Quantum dot (QD) is a semiconducting nanostructure where electrons are confined in all three spatial dimensions [8], as shown in Fig. 1. The quantum states of the pyramidal quantum dot with a single electron can be described by the time-independent 3D Schrödinger equation

$$-\nabla \cdot \left(\frac{\hbar^2}{2m(\mathbf{r}, \lambda)} \nabla u \right) + V(\mathbf{r}) u = \lambda u, \tag{1}$$

defined on a cuboid Ω, where λ is called an energy state or eigenvalue, and u is the corresponding wave function or eigenvector. In (1), $\hbar$ is the reduced

Department of Computer Science, University of Colorado Boulder, CO 80309, USA, tao.zhao@colorado.edu · Department of Mathematics, National Central University, Jhongli 320, Taiwan, hwangf@math.ncu.edu.tw · Department of Computer Science, University of Colorado Boulder, CO 80309, USA, cai@cs.colorado.edu

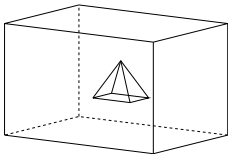

Fig. 1 Structure of a pyramidal quantum dot embedded in a cuboid. The size of the cuboid is 24.8nm×24.8nm×18.6nm; the width of the pyramid base is 12.4nm and the height of the pyramid is 6.2nm.

Plank constant, $\mathbf{r}$ is the space variable, $m(\mathbf{r}, \lambda)$ is the effective electron mass, and $V(\mathbf{r})$ is the confinement potential.

The Ben Daniel-Duke interface condition

$$\left(\frac{1}{m(\mathbf{r},\lambda)}\frac{\partial u}{\partial \mathbf{n}}\right)\Big|_{\partial D_-} = \left(\frac{1}{m(\mathbf{r},\lambda)}\frac{\partial u}{\partial \mathbf{n}}\right)\Big|_{\partial D_+}$$

is imposed on the interface, where D denotes the domain of the pyramid dot and $\mathbf{n}$ is the unit outward normal of ∂D. We impose the homogeneous Dirichlet boundary condition $u = 0$ on the boundary of the cuboid. For details, see [3, 8] and references therein.

A cell-centered finite volume method on an uniform mesh in Cartesian coordinates is applied to discretize the Schrödinger equation with non-parabolic effective mass model [3]. Then we obtain the polynomial eigenvalue problem

$$(\lambda^5 A_5 + \lambda^4 A_4 + \lambda^3 A_3 + \lambda^2 A_2 + \lambda A_1 + A_0)x = 0, \tag{2}$$

where $\lambda \in \mathbb{C}$, $x \in \mathbb{C}^N$, $A_i \in \mathbb{R}^{N\times N}$, and N is the total number of unknowns. The matrices A_0 and A_1 are diagonal, and all other matrices are nonsymmetric.

The rest of the paper is organized as follows. In Section 2, we first recall the convergence of the single-vector version of Jacobi-Davidson (JD) based on the residual of the approximate eigenpair for solving the general polynomial eigenvalue problem of degree m. Then we propose a three-grid parallel domain decomposition based JD algorithm for computing the relevant quantum dot eigenvalues and the corresponding eigenvectors. Numerical results are reported in Section 3. Some finial remarks are given in Section 4.

2 Jacobi-Davidson algorithm and domain decomposition based preconditioners

For given $A_i \in \mathbb{C}^{N\times N}$, $i = 0, 1, \cdots, m$, we define $\mathcal{A}_\phi = \sum_{i=0}^{m} \phi^i A_i$ as a matrix polynomial of $\phi \in \mathbb{C}$. If there exist $\lambda \in \mathbb{C}$ and $x \in \mathbb{C}^N$ such that $\mathcal{A}_\lambda x = 0$, then λ is called an eigenvalue of $\mathcal{A}_\phi$ and x is the eigenvector of $\mathcal{A}_\phi$

associated with the eigenvalue λ. There are several versions of JD for solving eigenvalue problems; see [4**? ?**] and references therein. A relatively simple version referred to as JD1 in this paper is summarized in Algorithm 1 below.

Algorithm 1 JD1 for polynomial eigenvalue problems

Input: A_i for $i = 0, \cdots, m$, and the maximum number of iterations k.

1: Choose an initial eigenvector u_0 with $\|u_0\|_2 = 1$.
For $n = 0, \cdots, k$
2: Solve $u_n^* \mathcal{A}_{\phi_n} u_n = 0$ for a new eigenvalue approximation ϕ_n.
3: Compute the residual $r_n = \mathcal{A}_{\phi_n} u_n$.
4: If the stopping criteria is satisfied, then stop.
5: Compute $p_n = \mathcal{A}'_{\phi_n} u_n = \left(\sum_{i=1}^{m} i\phi_n^{i-1} A_i\right) u_n$.
6: Solve $\|(I - (p_n u_n^*)/(u_n^* p_n))\, \mathcal{A}_{\phi_n} z_n + r_n\|_2 \le \varepsilon_n \|r_n\|_2,\ z_n \perp u_n$.
7: Compute a new eigenvector approximation $u_{n+1} = (u_n + z_n)/\|u_n + z_n\|_2$.
End for

Theorem 1. *Let $P_n = p_n u_n^*/(u_n^* p_n)$ and (λ, x) be an eigenpair of $\mathcal{A}_\phi$. There exists $\mathbb{D}(\lambda, x, d) = \{\phi \in \mathbb{C},\ u \in \mathbb{C}^N : \mathcal{A}_\phi$ is nonsingular and $\|\mathcal{A}_\phi u\|_2 < d\}$. If the initial eigenpair (ϕ_0, u_0) and any eigenpair (ϕ_n, u_n) generated by JD1 are all in $\mathbb{D}(\lambda, x, d)$, then the residuals satisfy*

$$\|(I - P_n) r_{n+1}\|_2 \le \varepsilon_n \|r_n\|_2 + \xi_n \|r_n\|_2^2 + \mathcal{O}(\|r_n\|_2^3) \tag{3}$$

if $\mathcal{A}_{\phi_n}$ is non-Hermitian, and

$$\|(I - P_n) r_{n+1}\|_2 \le \varepsilon_n \|r_n\|_2 + \zeta_n \|r_n\|_2^3 + \mathcal{O}(\|r_n\|_2^4) \tag{4}$$

if $\mathcal{A}_{\phi_n}$ is Hermitian. Here, ξ_n and ζ_n depend on ε_n, ϕ_n and A_i's.

Because of the page limit, the proof of the theorem is not shown. Theorem 1 implies that if r_{n+1} is orthogonal to u_n, then $I - P_n$ can be removed from the left-hand sides without changing the right-hands sides of (3) and (4). The authors of [4] suggest that a subspace $\mathcal{V}_{n+2}$ is built by all the correction vectors $u_n's$ and the initial vector u_0. Then a new approximate eigenvector u_{n+1} is extracted from the subspace with the Galerkin condition $r_{n+1} \perp \mathcal{V}_{n+2}$. We will refer the resulting method as the JD algorithm described in Algorithm 2 below. In the JD algorithm, r_{n+1} is orthogonal to any u_i for $i \le n + 1$, which leads to $(I - P_n) r_{n+1} = r_{n+1}$. Thus one can reasonably expect that the JD algorithm has a quadratic or cubic convergence if ε_n is sufficiently small relative to the residual.

Algorithm 2 JD for polynomial eigenvalue problems

Input: A_i for $i = 0, \cdots, m$, and the maximum number of iterations k.

1: Let $V = [v]$, where v is an initial eigenvector such that $\|v\|_2 = 1$.
For $n = 0, \cdots, k$
2: Compute $W_i = A_i V$ and $M_i = V^H W_i$ for $i = 0, \cdots, m$.

3: Solve the projected polynomial eigenvalue problem $\left(\sum_{i=0}^{m} \phi^i M_i\right) s = 0$, then obtain the desired eigenpair (ϕ, s) such that $\|s\|_2 = 1$.
4: Compute the Ritz vector $u = Vs$, and the residual vector $r = \mathcal{A}_\phi u$.
5: If the stopping criteria is satisfied, then stop.
6: Compute $p = \mathcal{A}'_\phi u = \left(\sum_{i=1}^{m} i\phi^{i-1} A_i\right) u$.
7: Solve $\|(I - (pu^*)/(u^*p))\, \mathcal{A}_\phi (I - uu^*)t + r\|_2 \le \varepsilon_n \|r\|_2,\ t \perp u$.
8: Orthogonalize t against V, set $v = v/\|t\|_2$, then expand $V \leftarrow [V, v]$.
End for

Remark 1. If the initial guess is good enough, and if the tolerance of the correction equation satisfies $\varepsilon_n \le \mathcal{O}(\|r_n\|_2)$, then the JD algorithm may converge quadratically, but we are unable to theoretically prove this.

Remark 2. If ε_n is chosen as a constant independent of n, then the convergence can only be linear in theory, however, in practice, if the constant tolerance is reasonably small, quadratic or near quadratic convergence has been observed in our numerical experiments.

In the entire JD approach, the linear correction equation is the most expensive part of the calculation since it is in the inner most loop. In earlier work, people often restrict the number of iterations to be carried out for the correction equation to be a small number (5 or 10) without considering how large the residual is when the iteration is stopped. This does cut down the computational cost per iteration, but as a result, the outer JD iteration may not have a quadratic convergence. In this paper, we make sure the correction equation is solved to a certain accuracy. A two-level preconditioner with a sufficiently fine coarse grid is used to control the number of iterations and scalability of the correction equation solver.

In JD, the preconditioner is of the form $\tilde{M} = (I - (pu^*)/(u^*p))\, M(I - uu^*)$, where M is an approximation of $\mathcal{A}_\phi$. We assume that the Krylov subspace method starts with an initial vector $t = 0$, and is preconditioned by $\tilde{M}$ from the right with a fixed ϕ. In the Krylov solver, we have to compute $x = \tilde{M}^{-1} y$ at each iteration. To avoid forming $\tilde{M}^{-1}$ explicitly, we solve a linear system $\tilde{M}x = y$ for x. Assume that x is orthogonal to u. It is straightforward to show that x takes the following form $x = M^{-1}y - (u^*M^{-1}y)/(u^*M^{-1}p)M^{-1}p$. Thus, for solving each correction equation, we need to compute $s = M^{-1}y$ at each iteration of Krylov subspace method, while compute $M^{-1}p$ only once.

In our method, we let M^{-1} be a two-level multiplicative type Schwarz preconditioner [1, 2]. Its multiplication with a vector y requires two steps:

$$
\begin{aligned}
s &\leftarrow I_c^h M_c^{-1} R_h^c y, \\
s &\leftarrow s + M_f^{-1}(y - \mathcal{A}_\phi s).
\end{aligned}
$$

Here, M_c is a preconditioner defined on the coarse mesh Ω_c. To obtain M_c, we discretize the Schrödinger equation (1) on Ω_c by the finite volume method mentioned in Section 1 and then obtain a coarse mesh polynomial eigenvalue

problem $\left(\sum_{i=0}^{m} \lambda^i B_i\right) x = 0$. For this particular quantum dot simulation that we are interested, m is equal to 5. The matrix B_i $(i = 0, \cdots, m)$ is much smaller than A_i in (2), but has the same nonzero structure pattern as A_i. Then we define $M_c = \sum_{i=0}^{m} \phi^i B_i$, where ϕ is the Ritz value computed on the fine mesh Ω_f.

In the first step of the two-level Schwarz preconditioner, I_c^h is an interpolation from Ω_c to Ω_f, and R_h^c is a restriction from Ω_f to Ω_c. As before, computing $w = M_c^{-1}(R_h^c y)$ is equivalent to solving a linear system $M_c w = R_h^c y$. In practice, we solve it approximately using a Krylov subspace method preconditioned by a one-level RAS preconditioner defined on the coarse mesh Ω_c using the same number of processors as on the fine mesh. In the second step of the two-level preconditioner, M_f is the RAS preconditioner defined on the fine mesh Ω_f.

To build the RAS preconditioners, we partition the cuboid into non-overlapping subdomains ω_i, $i = 1, \cdots, np$, then generate the overlapping subdomain ω_i^δ by including the δ layers of mesh cells in the neighboring subdomains of ω_i, i.e., $\omega_i \subset \omega_i^\delta$. Here, np is the number of processors that is the same as the number of subdomains, and δ is the size of overlap. Let R_i^0 and R_i^δ be restriction operators to non-overlapping and overlapping subdomains, respectively. With R_i^δ, we define the matrix $\mathcal{J}_i = R_i^\delta \mathcal{A}_\phi (R_i^\delta)^T$. Then the one-level RAS preconditioner reads as $M_{RAS}^{-1} = \sum_{i=1}^{np} (R_i^0)^T \mathcal{J}_i^{-1} R_i^\delta$. In practice, $\mathcal{J}_i^{-1}$ is not formed explicitly, instead it is approximated by ILU factorization.

In theory, a good initial guess implies good convergence of JD, but in practice, it is a nontrivial issue to find the right initial guess, especially when both the accuracy and the computational cost need to be balanced since our goal is to achieve near linear speedup measured by the total compute time. For convenience (less coding, less memory required, and computationally cheaper), the coarse mesh for finding the initials is usually chosen to be the coarse mesh of the two-level Schwarz method. However, as is shown in the next section, the coarse mesh of the two-level Schwarz preconditioner in this paper is not suitable to generate the initial guess since it is still very large. Note that only several eigenpairs around the ground state are of interests in this simulation. As a result, we have to generate another much coarser mesh Ω_o for computing the initials.

On Ω_o, we discretize the Schrödinger equation using the finite volume method mentioned in Section 1. Next, the resulting polynomial eigenvalue problem is solved using, for instance, the QZ method with linearization. Once we obtain the desired eigenpair (ϕ_o, v_o), ϕ_o is used as the initial eigenvalue and v_o is interpolated to the fine mesh Ω_f to generate the initial eigenvector on the fine mesh $v_h \leftarrow I_o^h v_o$, where I_o^h is an interpolation operator from Ω_o to Ω_f. Due to the small size, (ϕ_o, v_o) is computed redundantly on all processors.

With the coarse and fine grids, we describe the three-grid parallel domain decomposition based JD algorithm in Algorithm 3.

Algorithm 3 Three-grid parallel domain decomposition based JD algorithm

for polynomial eigenvalue problems

Input: Coefficient matrices on Ω_o, Ω_c and Ω_f for $i = 0, \cdots, m$.

1: On Ω_c, solve the polynomial eigenvalue problem roughly and obtain the desired eigenpair (ϕ_o, v_o).
2: Obtain the initial eigenvector on the fine mesh $v_h \leftarrow I_o^h v_o$.
3: Solve the polynomial eigenvalue problem on Ω_f by Algorithm 2. At each iteration, the correction equation is solved to a modest accuracy by Krylov subspace method with either one-level or two-level preconditioner.

3 Numerical results

We use Algorithm 3 to compute 6 smallest positive eigenvalues and the corresponding eigenvectors of the pyramidal quantum dot problem as shown in Fig. 1. The physical parameters in the non-parabolic effective mass model are the same as described in [3]. The software is implemented using PETSc [5], SLEPc [7] and PJDPack [3].

The fine mesh Ω_f is 600×600×450 with 161,101,649 unknowns. The coarse mesh Ω_o to generate the initial guess is 12×12×9 with 968 unknowns. Due to the small size of Ω_o, the Schrödinger equation discretized on Ω_o is solved redundantly on all processors using JD with the one-vector as the initial guess. The JD iteration is stopped when either the absolute or the relative residual norm is below 10^{-8}. The eigenvectors on Ω_o are interpolated to the fine mesh by trilinear interpolation.

On Ω_f, we stop the JD iteration when either the absolute or the relative residual norm is below 10^{-10}. The correction equation is solved by the flexible GMRES (FGMRES) without restarting [6] preconditioned by either one-level or two-level preconditioners. The stopping criteria of FGMRES on Ω_f is 10^{-4}. For the two-level Schwarz preconditioner, we solve the linear system on Ω_c by FGMRES with the RAS preconditioner. The stopping criteria of FGMRES on Ω_c is 10^{-1}. For the RAS preconditioners on Ω_c and Ω_f, ILU(0) is applied to solve the linear system on each subdomain; the size of overlap is 1.

Tables 1-3 show the numerical performance of JD with the one-level and two-level preconditioners in terms of the number of JD iterations, the average number of FGMRES for solving the correction equations and the compute time. Since the imaginary parts of the computed eigenvalues are less than 10^{-13}, we report the real parts only. Consider the compute time and the average number of FGMRES iterations, the two-level preconditioner is much better than the one-level preconditioner.

Figures 2 and 3 plot the speedup curves of JD with one-level and two-level preconditioners. Obviously, JD with both preconditioners are scalable, and the two-level approach is faster in terms of the total compute time.

Table 1 The ground state $e_0 = 0.4162094856604$ and Ω_c is $56 \times 56 \times 42$.

	One-level			Two-level		
np	JD	FGMRES	Time	JD	FGMRES	Time
5120	4	185.25	42.48	4	38.75	7.48
7168	4	185.25	29.20	4	39.50	6.36
9216	4	186.00	23.23	4	38.75	5.34
10240	4	186.00	22.46	4	39.75	4.84

Table 2 The first excited state $e_1 = 5.990754117523$ and Ω_c is $80 \times 80 \times 60$.

	One-level			Two-level		
np	JD	FGMRES	Time	JD	FGMRES	Time
5120	4	251.75	71.10	4	34.75	9.47
7168	4	255.75	47.96	4	34.25	6.20
9216	4	254.50	39.29	4	34.50	5.64
10240	4	256.50	37.37	4	34.00	5.29

Table 3 The second excited state $e_2 = 0.5990754117522$ and Ω_c is $80 \times 80 \times 60$.

	One-level			Two-level		
np	JD	FGMRES	Time	JD	FGMRES	Time
5120	4	258.25	71.90	4	34.25	9.99
7168	4	243.25	45.42	4	34.25	6.71
9216	4	258.00	40.99	4	35.50	5.92
10240	4	250.50	35.42	4	34.50	5.44

Table 4 Residual norms of the first six eigenpairs at each Jacobi-Davidson iteration using 10240 processors. The correction equations are preconditioned by the two-level Schwarz preconditioner. "it" is the index for the JD iteration.

it	e_0	e_1	e_2	e_3	e_4	e_5
0	$2.590e+00$	$4.249e+00$	$4.249e+00$	$9.430e+00$	$5.457e+00$	$7.339e+00$
1	$7.614e-02$	$2.236e-01$	$2.233e-01$	$3.926e+00$	$5.959e-01$	$1.465e+00$
2	$1.911e-04$	$3.433e-04$	$3.505e-04$	$1.211e+00$	$6.696e-03$	$8.375e-02$
3	$1.640e-08$	$3.657e-08$	$3.780e-08$	$1.418e-01$	$1.044e-06$	$8.642e-05$
4	$1.780e-12$	$8.054e-12$	$8.216e-12$	$3.547e-04$	$1.036e-11$	$3.270e-08$
5				$6.083e-08$		$7.659e-12$
6				$8.533e-12$		

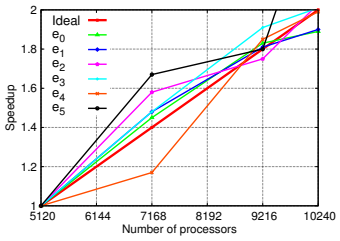

Fig. 2 Speedup with the one-level preconditioner.

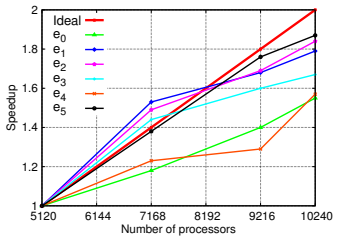

Fig. 3 Speedup with the two-level Schwarz preconditioner.

4 Conclusions

A parallel domain decomposition based Jacobi-Davidson algorithm with three meshes was introduced and studied for the pyramidal quantum dot simulation. The proposed method requires three meshes; one fine mesh that determines the accuracy of the solution and two coarse meshes to accelerate the convergence of the inner and outer iterations. Numerical results confirmed that our method converges quadratically with the proposed strategy for computing the initial guess, and also is scalable for problems with over 160 millions unknowns on a parallel computer with over 10,000 processors.

References

[1] P. E. Bjørstad B. F. Smith and W. D. Gropp. *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations.* Cambridge University Press, New York, 1996.

[2] X.-C. Cai and M. Sarkis. A restricted additive Schwarz preconditioner for general sparse linear systems. *SIAM Journal on Scientific Computing*, 21:239–247, 1999.

[3] T.-M. Huang F.-N. Hwang, Z.-H. Wei and W. Wang. A parallel additive schwarz preconditioned jacobi-davidson algorithm for polynomial eigenvalue problems in quantum dot simulation. *J. Comput. Phys.*, 229: 2923–2947, 2010.

[4] D. R. Fokkema G. L. G. Sleijpen, A. G. L. Booten and H. van der Vorst. Jacobi-davidson type methods for generalized eigenproblems and polynomial eigenproblems. *BIT*, 36:595–633, 1996.

[5] K. Buschelman V. Eijkhout W. D. Gropp D. Kaushik M. G. Knepley L. C. McInnes B. F. Smith S. Balay, J. Brown and H. Zhang. Petsc users manual, 2013. Argonne National Laboratory.

[6] Y. Saad. *Iterative Methods for Sparse Linear Systems.* SIAM, Philadelphia, 2nd edition, 2003.

[7] J. E. Roman V. Hernandez and V. Vidal. SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems. *ACM Trans. Math. Software*, 31:351–362, 2005.

[8] N. Vukmirović and L.-W. Wang. Quantum dots: theory. Technical report, Lawrence Berkeley National Laboratory, 2009.