TAIWANESE JOURNAL OF MATHEMATICS
Vol. 14, No. 3A, pp. 999-1021, June 2010
This paper is available online at http://www.tjm.nsysu.edu.tw/

AN EFFICIENCY STUDY OF POLYNOMIAL EIGENVALUE PROBLEM SOLVERS FOR QUANTUM DOT SIMULATIONS

Tsung-Ming Huang, Weichung Wang and Chang-Tse Lee

Abstract. Nano-scale quantum dot simulations result in large-scale polynomial eigenvalue problems. It remains unclear how these problems can be solved efficiently. We fill this gap in capability partially by proposing a polynomial Jacobi-Davidson method framework, including several varied schemes for solving the associated correction equations. We investigate the performance of the proposed Jacobi-Davidson methods for solving the polynomial eigenvalue problems and several Krylov subspace methods for solving the linear eigenvalue problems with the use of various linear solvers and preconditioning schemes. This study finds the most efficient scheme combinations for different types of target problems.

1. INTRODUCTION

A standard matrix polynomial eigenvalue problem can be written as

$$\mathcal{P}(\lambda)\, u \equiv \left(\sum_{i=0}^{\tau} \lambda^i A_i \right) u = 0, \tag{1}$$

where (λ, u) is the corresponding eigenpair with $\lambda \in \mathbb{C}$ and $u \in \mathbb{C}^{\mathcal{N}}$, integer τ is the degree of the matrix polynomial, and $A_i \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}}$ are the coefficient matrices. Solving large-scale polynomial eigenvalue problems has a long term history and still remains an active research topic due to its computational challenges and wide applications. Scientific and engineering studies that lead to polynomial eigenvalue problems include nano-scale quantum mechanism of degree 1 [7], 3 [34], or 5 [15]; high speed railway vibration of degree 2 [3, 14]; and plasma physics of degree 3 [13].

Received August 11, 2009, revised December 1, 2009, accepted January 4, 2010.
2000 *Mathematics Subject Classification*: 65F15, 65F50.
Key words and phrases: Polynomial eigenvalue problems, Jacobi-Davidson methods, Correction equations, Krylov subspace methods, Schrödinger equation, Quantum dot.

In this article, we focus on the polynomial eigenvalue problems arising in simulations of nano-scale quantum dots (QDs). In particular, we consider how the Jacobi-Davidson methods and Krylov subspace methods may be used to solve these problems efficiently. This study aims to do the following:

- derive a framework of the Jacobi-Davidson method for solving polynomial eigenvalue problems;
- propose various schemes for solving the correction equation in the Jacobi-Davidson method;
- provide a set of large-scale polynomial eigenvalue benchmark problems arising in the numerical simulation of quantum dots; and
- to perform intensive numerical performance comparisons for the various schemes against the benchmark problems.

The paper is organized as follows. We first introduce the model problems arising in quantum dot simulations in Section 2. Several variants of Jacobi-Davidson methods are discussed in Section 3. Several Krylov subspace methods for solving linear eigenvalue problems are discussed in Section 4. Numerical experiments are presented and analyzed in Section 5. We conclude the paper in Section 6.

2. Model Problems

Nano-scale semiconductor quantum dots are materials in which the carriers are confined within the dots in all three dimensions. These carries consequently have wavelike properties with discrete energy levels that are induced. Over the past few years, numerous studies regarding nano-scale quantum dots have been conducted to examine their physical properties [4, 10, 20, 25] and applications [2, 8, 9, 19, 24]. Other than theoretical and experimental methods, numerical simulations also play important roles to investigate insights into a QD's electronic and optical properties [26, 28, 35].

We focus on a single particle conduction band model in this article. As shown in Figure 1, which is a structure scheme of the model, a QD is embedded in a matrix. The governing equation of this model can be described by the Schrödinger equation in general

$$-\nabla \cdot (\frac{\hbar^2}{2m(\mathbf{x})}\nabla u) + c(\mathbf{x})u = \lambda u, \tag{2}$$

or, in cylindrical coordinates,

$$-\frac{1}{r}\frac{\partial}{\partial r}(r\frac{\hbar^2}{2m(\mathbf{x})}\partial_r u) - \frac{1}{r}\partial_\theta(\frac{1}{r}\frac{\hbar^2}{2m(\mathbf{x})}\partial_\theta u) - \partial_z(\frac{\hbar^2}{2m(\mathbf{x})}\partial_z u) + c(\mathbf{x})u = \lambda u. \tag{3}$$

Here, $\hbar$ is the reduced Plank constant, the eigenvalue λ is the total electron energy, and the corresponding eigenvector u denotes the wave function. The variable $m(\mathbf{x})$ represents the electron effective mass, and $c(\mathbf{x})$ is the confinement potential.

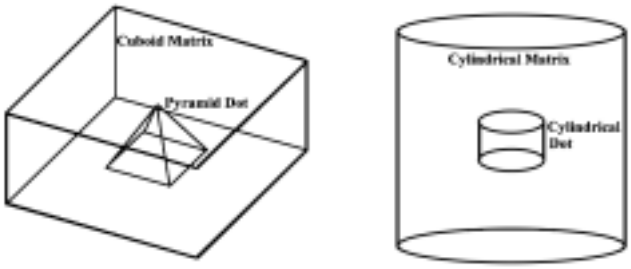


Fig. 1. Structure schema of a pyramidal and a cylindrical quantum dot. Each of the quantum dots is embedded in a hetero-structure matrix.

In hetero-structures, both $m(\mathbf{x})$ and $c(\mathbf{x})$ are discontinuous at the interface. Particularly, in a constant effective mass model, $m(\mathbf{x})$ and $c(\mathbf{x})$ are piecewise constant functions with respect to the space variable $\mathbf{x}$:

$$m(\mathbf{x}) = \begin{cases} m_1 & \text{in the dot,} \\ m_2 & \text{in the matrix,} \end{cases} \qquad c(\mathbf{x}) = \begin{cases} c_1 & \text{in the dot,} \\ c_2 & \text{in the matrix.} \end{cases}$$

The Ben Daniel-Duke interface conditions associated with the discontinuity in m are imposed as follows

$$\begin{aligned} u|_{D_+} &= u|_{D_-}\,, \\ \frac{\hbar^2}{2m_2}\frac{\partial u}{\partial n}\bigg|_{\partial D_+} &= \frac{\hbar^2}{2m_1}\frac{\partial u}{\partial n}\bigg|_{\partial D_-}\,, \end{aligned} \tag{4}$$

where D stands for the QD domain. The normal direction n with the subscripts $+$ and $-$ denotes the corresponding outward normal derivatives of the interface defined for the matrix and dot regions, respectively. If the non-parabolicity of the electron's dispersion relation is taken into account [5], the effective mass model for the interface conditions (4) becomes

$$\frac{1}{m_\ell(\lambda)} = \frac{P_\ell^2}{\hbar^2}\left(\frac{2}{\lambda + g_\ell - c_\ell} + \frac{1}{\lambda + g_\ell - c_\ell + \delta_\ell}\right), \tag{5}$$

where P_ℓ, g_ℓ, and δ_ℓ are the momentum, main energy gap, and spin-orbit splitting in the ℓth region, respectively. The following parameters are used in our numerical experiments: c_1=0.000, g_1=0.235, δ_1=0.81, P_1=0.2875, c_2=0.350, g_2=1.590, δ_2= 0.80, and $P_2 = 0.1993$. Additionally, we apply homogeneous Dirichlet conditions on the boundary of the quantum matrix.

To simulate QDs fabricated in laboratories, various discretization schemes have been developed for various QD geometries such as a cylinder [16, 18, 34], cone [23], pyramid [6, 15, 27, 33], or irregular shape [17]. Among these QD geometries and discretizations, we pick the following five settings and use the induced polynomial eigenvalue problems as the test problems. We omit detailed derivations of the discretizations and the resulting eigen-systems here as they can be found in the corresponding references.

- Cylindrical QD with the constant effective mass model [18]:

$$\mathcal{P}_{cc}(\lambda)u \equiv (A_0^{cc} + \lambda I)u, \tag{6}$$

  where A_0^{cc} is unsymmetric.

- Cylindrical QD with the non-parabolic effective mass model [34]:

$$\mathcal{P}_{cn}(\lambda)u \equiv \left(\sum_{i=0}^{3} \lambda^i A_i^{cn}\right) u. \tag{7}$$

- Pyramidal QD with the constant effective mass model [15]:

$$\mathcal{P}_{pc}(\lambda)u \equiv (A_0^{pc} - \lambda I)u, \tag{8}$$

  where A_0^{pc} is a symmetric positive definite matrix.

- Pyramidal QD with non-parabolic effective mass model [15]:

$$\mathcal{P}_{pn}(\lambda)u \equiv \left(\sum_{i=0}^{5} \lambda^i A_i^{pn}\right) u. \tag{9}$$

- Irregular QD with the constant effective mass model over the skewed coordinate system [17]:

$$\mathcal{P}_{ic}(\lambda)u \equiv (A_0^{ic} + \lambda A_1^{ic})u, \tag{10}$$

  where A_0^{ic} is symmetric positive definite matrix and A_1^{ic} is a diagonal matrix.

The above test problems have different degrees in the polynomial eigenvalue problems and different distributions of matrix entry magnitudes. We use the eigensolvers proposed in Sections 3 and 4 to solve these problems, to investigate the efficiency, and to justify the performance.

3. Polynomial Jacobi-davidson Algorithm

Jacobi-Davidson type methods are competitive numerical methods when interior eigenvalues are of interest, especially for large-scale eigenvalue problems. Such attractive properties mainly result from the fact that: (i) the coefficient matrices are used implicitly in the matrix-vector multiplication forms; (ii) no inverse of the matrix is needed even to compute interior eigenvalues; and (iii) the desired eigenpairs are approximated iteratively by a gradually expanding subspace, and then

corrector vectors can be approximately computed and preconditioned to achieve efficiency. However, there is only sparse literature that addresses how variants of Jacobi-Davidson methods perform on large-scale polynomial eigenvalue problems.

We describe the main idea of the Jacobi-Davidson method for the polynomial eigenvalue problem as follows from the viewpoint of Taylor expansion. We see that the following derivation is a straightforward generation of the standard linear eigenvalue problem discussed in [29]. Suppose $\mathcal{V}_k$ is a k-dimensional subspace that has an orthogonal unitary basis $v_1, v_2, \ldots, v_k$. Let (θ_k, u_k) be a Ritz pair (an approximate eigenpair) of $\mathcal{P}(\lambda)$ and (θ_k, s_k) be an eigenpair of $V_k^* \mathcal{P}(\lambda) V_k s = 0$, where $\|s_k\|_2 = 1$, $u_k = V_k s_k$, and $V_k = [v_1, \cdots, v_k]$. To expand the subspace $\mathcal{V}_k$ successively, the Jacobi-Davidson method finds the orthogonal complement for the current approximation u_k. In other words, starting from the Ritz pair (θ_k, u_k), we intend to find a corrector $t \perp u_k$ such that

$$\mathcal{P}(\lambda)(u_k + t) = 0. \tag{11}$$

Obviously, it is impractical to solve Eq. (11) directly as it is another polynomial eigenvalue problem that is equivalent to the original target problem (1). Instead, we may expand Eq. (11) and rewrite $\mathcal{P}(\lambda)$ in terms of θ_k by using a Taylor expansion to obtain an approximation to the corrector t. First, we rewrite Eq. (11) as

$$\mathcal{P}(\lambda)t = -\mathcal{P}(\lambda)u_k = -r_k + (\mathcal{P}(\theta_k) - \mathcal{P}(\lambda))u_k,$$

where the residual vector

$$r_k = \mathcal{P}(\theta_k)u_k.$$

Furthermore, by assuming θ_k is close to λ, we may use Taylor's Theorem to obtain

$$\begin{aligned}(\mathcal{P}(\theta_k) - \mathcal{P}(\lambda))u_k &= \left[(\theta_k - \lambda)\mathcal{P}'(\theta_k) - \frac{1}{2}(\theta_k - \lambda)^2\mathcal{P}''(\xi_k)\right] u_k \\ &= (\theta_k - \lambda)p_k - \frac{1}{2}(\theta_k - \lambda)^2\mathcal{P}''(\xi_k)u_k,\end{aligned}$$

where ξ_k is between λ and θ_k and

$$p_k = \mathcal{P}'(\theta_k)u_k \equiv \frac{d}{d\lambda}\mathcal{P}(\lambda)\Big|_{\lambda=\theta_k} u_k = \left(\sum_{i=1}^{\tau} i\theta_k^{i-1}A_i\right) u_k.$$

Consequently, Eq. (11) is equivalent to

$$\mathcal{P}(\lambda)t = -r_k + (\theta_k - \lambda)p_k - \frac{1}{2}(\theta_k - \lambda)^2\mathcal{P}''(\xi_k)u_k. \tag{12}$$

We can further manipulate the terms containing $(\theta_k - \lambda)$ to derive practical computational schemes for finding t. First, by using the fact that $r_k \perp u_k$, or

$$u_k^* r_k = u_k^* \mathcal{P}(\theta_k) u_k = s_k^* V_k^* \mathcal{P}(\theta_k) V_k s_k = 0,$$

we multiply $\left(I - \frac{p_k u_k^*}{u_k^* p_k}\right)$ on both sides of Eq. (12) to eliminate the term containing $(\theta_k - \lambda)$ and to obtain

$$\left(I - \frac{p_k u_k^*}{u_k^* p_k}\right) \mathcal{P}(\lambda) t = -r_k - \frac{1}{2}(\theta_k - \lambda)^2 \left(I - \frac{p_k u_k^*}{u_k^* p_k}\right) \mathcal{P}''(\xi_k) u_k.$$

Second, we neglect the second order term containing $(\theta_k - \lambda)^2$ on the right hand side to obtain a linear (in terms of λ) approximation of $\mathcal{P}(\lambda)t$ satisfying

$$\left(I - \frac{p_k u_k^*}{u_k^* p_k}\right) \mathcal{P}(\lambda) t = -r_k \quad \text{and} \quad t \perp u_k. \tag{13}$$

Practically, we further apply the orthogonal projection $(I - u_k u_k^*)$ and approximate $\mathcal{P}(\lambda)$ by $\mathcal{P}(\theta_k)$ and then form the following correction equation

$$\left(I - \frac{p_k u_k^*}{u_k^* p_k}\right) \mathcal{P}(\theta_k)(I - u_k u_k^*) \tilde{t} = -r_k \quad \text{and } \tilde{t} \perp u_k. \tag{14}$$

Based on the above discussions, a general polynomial Jacobi-Davidson method designed to compute all the desired eigenvalues for the problem (1) is shown in Algorithm 1.

Algorithm 1. *Polynomial Jacobi-Davidson Algorithm for* $(\sum_{i=0}^{\tau} \lambda^i A_i)u = 0$.

Input: Coefficient matrices A_i for $i = 0, \ldots, \tau$, the number of desired eigenvalues k and an initial orthonormal vector V_{ini}.

Output: The desired eigenpairs (λ_j, u_j) for $j = 1, \ldots, k$.

1. Set $V = [V_{ini}]$, $V_u = [\ ]$, and $\Lambda = \emptyset$.
2. **for** $j = 1$ to k **do**
3. Compute $W_i = A_i V$ and $M_i = V^* W_i$ for $i = 0, \ldots, \tau$.
4. **while** (user defined stopping criteria are not satisfied)
5. Compute the eigenpairs (θ, s) of $(\sum_{i=0}^{\tau} \theta^i M_i)s = 0$.
6. Select the desired eigenpair (θ, s) with $\|s\|_2 = 1$ and $\theta \notin \Lambda$.
7. Compute $u = Vs$, $p = \mathcal{P}'(\theta)u$, $r = \mathcal{P}(\theta)u$.
8. Solve the correction equation

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

approximately for $t \perp u$.

9. Orthogonalize t against V; set $v = t/\|t\|_2$.
10. Compute $w_i = A_i v$,

$$M_i = \begin{bmatrix} M_i & V^*w_i \\ v^*W_i & v^*w_i \end{bmatrix}$$

for $i = 0, \ldots, \tau$.

11. Expand $V = [V, v]$ and $W_i = [W_i, w_i]$ for $i = 1, \ldots, \tau$.
12. **end while**
13. Set $\lambda_j = \theta$, $u_j = u$, $\Lambda = \Lambda \cup \{\lambda_j\}$.
14. Perform locking by orthogonalizing u_j against V_u; Compute $u_j = u_j/\|u_j\|_2$; Update $V_u = [V_u, u_j]$.
15. Choose an orthonormal matrix $V_{ini} \perp V_u$; Set $V = [V_u, V_{ini}]$.
16. **end for**

Now, we focus on three schemes for approximately solving Eq. (14). This is an essential step in the polynomial Jacobi-Davidson method that may affect the overall performance significantly. We approximately solve Eq. (14) by a preconditioned iterative method, e.g., GMRES with SSOR preconditioner. We call this preconditioned iterative process corresponding to line 1 of Algorithm 1 as the "inner loop" of the algorithm. On the other hand, we call the while-loop in lines 1 to 1 as the "outer loop" of the algorithm.

Three schemes, $\mathcal{S}_{OneLS}$, $\mathcal{S}_{TwoLS}$, and $\mathcal{S}_{OneStep}$, are proposed below to solve Eq. (14) approximately. $\mathcal{S}_{OneLS}$ solves one linear system by a preconditioned iteration method. $\mathcal{S}_{TwoLS}$ and $\mathcal{S}_{OneStep}$ solve two linear systems by preconditioned iterations, but $\mathcal{S}_{OneStep}$ conducts only one step in the preconditioned iterations.

- **Scheme** $\mathcal{S}_{OneLS}$. In each step of the preconditioned iterations for solving correction equation (14), we need to solve a linear system in a form such that

$$\mathcal{M}_p z = y, \quad z \perp u_k \tag{15}$$

where y is a certain given vector that is orthogonal to u_k and

$$\mathcal{M}_p \equiv \left(I - \frac{p_k u_k^*}{u_k^* p_k}\right)\mathcal{M}\,(I - u_k u_k^*)$$

with preconditioner $\mathcal{M}$ of $\mathcal{P}(\theta_k)$. Under the requirement $z \perp u_k$, the solution of Eq. (15) is given by

$$z = \mathcal{M}^{-1}y + \eta_k \mathcal{M}^{-1}p_k \quad \text{with } \eta_k = -\frac{u_k^*\mathcal{M}^{-1}y}{u_k^*\mathcal{M}^{-1}p_k}. \tag{16}$$

It is clear that the vector $\mathcal{M}^{-1}p_k$ and the inner product $u_k^*\mathcal{M}^{-1}p_k$ need to be computed only once in the first step of the preconditioned iteration. Consequently, other iterative steps need only the preconditioning operations in the form of $\mathcal{M}^{-1}y$.

- **Scheme** $\mathcal{S}_{TwoLS}$. By (14) and $t \perp u_k$, it follows that

$$\mathcal{P}(\theta_k)t = \frac{u_k^*\mathcal{P}(\theta_k)t}{u_k^*p_k}p_k - r_k \equiv \eta_k(t)p_k - r_k. \tag{17}$$

We can then solve the two linear systems

$$\mathcal{P}(\theta_k)z = -r_k \quad \text{and} \quad \mathcal{P}(\theta_k)z = p_k \tag{18}$$

approximately by a preconditioned iterative method to obtain the approximate solution z_1 and z_2, respectively. Then we compute

$$\tilde{t} = z_1 + \eta_k z_2 \quad \text{for} \quad \eta_k = -\frac{u_k^*z_1}{u_k^*z_2}. \tag{19}$$

Here $\tilde{t}$ as an approximate solution to Eq. (17) that satisfies the requirement $\tilde{t} \perp u_k$. In $\mathcal{S}_{TwoLS}$, two linear systems (18) need to be solved approximately by a preconditioned iteration method to obtain the approximated solution $\tilde{t}$.

- **Scheme** $\mathcal{S}_{OneStep}$. We may reduce the cost for computing $\tilde{t}$ that solves Eq. (17) in the inner loop. Here we only conduct one preconditioned iteration. Namely, $\tilde{t}$ is computed by

$$\tilde{t} = -\mathcal{M}^{-1}r_k + \eta_k\mathcal{M}^{-1}p_k \quad \text{for} \quad \eta_k = \frac{u_k^*\mathcal{M}^{-1}r_k}{u_k^*\mathcal{M}^{-1}p_k}, \tag{20}$$

when we have a suitable preconditioner $\mathcal{M} \approx \mathcal{P}(\theta_k)$.

4. Krylov Subspace Methods

While the polynomial eigenvalue problems (6)-(10) can be solved by the Jacobi-Davidson methods presented in Section 3, the linear eigenvalue problems (6), (8), and (10) can also be solved by the the so-called Krylov subspace methods, such as the Lanczos, Arnoldi, and Krylov-Schur methods. In this section, we briefly describe how the Krylov subspace methods can be used to solve the standard eigenvalue problem $A_0u = \lambda u$. Similar ideas can be generalized to general eigenvalue problems $A_0u = \lambda A_1u$ by the A_1-inner product.

First, we define the Krylov decomposition [30] as follows. Let $V_{k+1} = [V_k, v_{k+1}] \in \mathbb{R}^{\mathcal{N}\times(k+1)}$ be an orthonormal matrix, where $V_k \in \mathbb{R}^{\mathcal{N}\times k}$ and $v_{k+1} \in \mathbb{R}^{\mathcal{N}\times 1}$. An orthonormal Krylov decomposition of order k is a relation of the form

$$A_0 V_k = V_k B_k + v_{k+1} b_{k+1}^T, \tag{21}$$

where $B_k \in \mathbb{R}^{k\times k}$ is a Rayleigh quotient $R(A_0; V_k) = V_k^T A_0 V_k$ and $b_{k+1} \in \mathbb{R}^{k\times 1}$. Note that if A_0 is symmetric, B_k is tridiagonal, and $b_{k+1} = \beta_k e_k$, then (21) is a Lanczos decomposition of order k. If B_k is upper Hessenberg, $b_{k+1} = \beta_k e_k$, then (21) is an Arnoldi decomposition of order k. Here $\beta_k \in \mathbb{R}$ and $e_k \in \mathbb{R}^{k\times 1}$ is the standard unit vector (kth column of identity matrix).

Consequently, the Ritz pairs associated with U_k can be computed as follows. Let (θ_i, v_i) be an eigenpair of B_k and let $(\theta_i, y_i) = (\theta_i, U_k v_i)$ be the Ritz pair of A. By (21), we have

$$\begin{aligned}\|Ay_i - \theta_i y_i\|_2 &= \|AU_k v_i - \theta_i U_k v_i\|_2 = \|(U_k B_k + u_{k+1} b_{k+1}^T) v_i - \theta_i U_k v_i\|_2 \\ &= \|U_k(B_k v_i - \theta_i v_i) + (b_{k+1}^T v_i) u_{k+1}\|_2 = |b_{k+1}^T v_i|.\end{aligned}$$

As k increases, some of these Ritz pairs will approach the eigenpairs of A. That is, $|b_{k+1}^T v_i| \to 0$ for some i.

Theoretically, we can keep expanding the Krylov decomposition until the Ritz eigenpairs converge to the desired eigenpairs. Practically, however, the expanding process is limited to avoid loss of numerical orthogonality of V_k and to use a reasonable amount of memory for storing V_k. A general idea of restarting is that, after V_p has been computed, a new Krylov process is performed to compute a different Krylov decomposition of order p with "better" initial vectors. For example, (i) an "explicit restart" strategy [11, 12] reruns the Krylov decomposition by using the approximate Schur vectors associated with the first not-yet-converged eigenvalue as an initial vector. (ii) An "implicit restart" [21] combines the Krylov decomposition process with the implicitly shifted QR algorithm. This implicit restart process is more efficient and numerically stable than explicit restart. (iii) A "Krylov-Schur method" [31, 32] that can be seen as an improvement on traditional Krylov subspace methods. We sum up all the processes in Algorithm 2.

Algorithm 2. *Restarting Krylov Subspace Algorithm for* $A_0 u = \lambda u$.

Input: Coefficient matrix A_0; initial orthonormal vector v_1; number of desired eigenpairs k; size of subspace for restarting p.

Output: The desired k eigenpairs of A_0.

1. Generate the Krylov decomposition of order j ($j \geq k$), by starting from v_1:

$$A_0 V_j = V_j B_j + v_{j+1} b_{j+1}^T.$$

2. Compute the Ritz pairs of A_0 from B_j and V_j.
3. **while** (the desired k eigenpairs of A_0 are not convergent)
4. Extend the Krylov decomposition from order j to p:

$$AV_p = V_p B_p + v_{p+1} b_{p+1}^T.$$

5. Compute the Ritz pairs of A_0 from B_p and V_p.
6. Reformat a new Krylov decomposition with order j by a restarting process.
7. **end while**

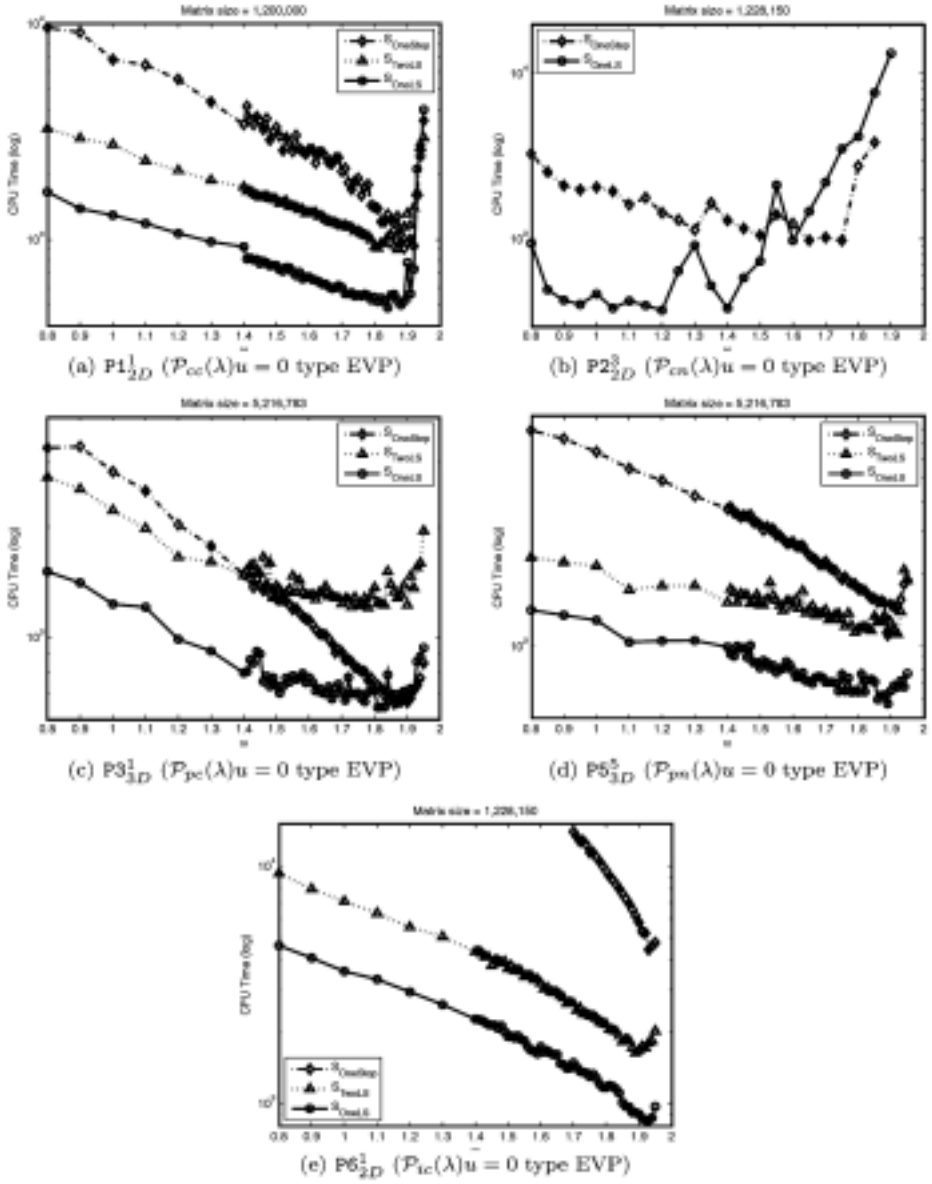
5. Numerical Results

We study how various Jacobi-Davidson and Krylov subspace methods perform when solving the polynomial eigenvalue problems arising in quantum dot simulations. The properties of the test problems are shown in Table 1. All of the test problems are solved by the Jacobi-Davidson methods. Only problems of degree 1 (linear or generalized) problems are solved by the Krylov type methods. Note that, as the eigenvector solutions of the cylindrical and irregular QDs are periodical in the azimuthal direction, the 3D problems are transformed to a sequence of 2D problems by the truncated Fourier series. Consequently, the discretization domain dimensions of problems $\texttt{P1}_{2D}^{1}$, $\texttt{P2}_{2D}^{3}$, and $\texttt{P6}_{2D}^{1}$ are over the two dimensional radial-longitude planes. Except for the results shown on Figure 2, all of the numerical experiments are conducted on an HP BL460c workstation composed of two Intel Dual-Core 5160 3.0 GHz CPUs and 32 GB main memory.

In Section 5.1 and 5.2, we study how the correction equation solution schemes and the preconditioners affect the performance of the Jacobi-Davidson methods presented in Algorithm 1. We implement the Jacobi-Davidson methods with Fortran 90. In Section 5.3, we investigate the performance of various Krylov subspace methods. In Section 5.4, we make an overall comparison of all methods and conclude the most efficient scheme combinations for each of the problems.

5.1. Correction Equation Solution Schemes

To compare the efficiency of the three schemes ($\mathcal{S}_{OneLS}$, $\mathcal{S}_{TwoLS}$, and $\mathcal{S}_{OneStep}$) for solving the correction equation (14), we use GMRES to solve the linear systems in (14) and (18) with the SSOR (symmetric successive over-relaxation) preconditioner $\mathcal{M} = (D + \omega L)D^{-1}(D + \omega U)$. Here $\mathcal{P}(\theta_k) = L + D + U$ and L, D, and U are the strict lower triangular, diagonal, and strict upper triangular matrices, respectively. The parameter ω is chosen from 0.8 to 1.95. The timing results for problems $\texttt{P1}_{2D}^{1}$, $\texttt{P2}_{2D}^{3}$, $\texttt{P3}_{3D}^{1}$, $\texttt{P5}_{3D}^{5}$, and $\texttt{P6}_{2D}^{1}$ are shown in Figure 2. The CPU timing results are computed by summing up the total cost for computing the smallest five positive eigenvalues and their corresponding eigenvectors.



(a) $\mathrm{P1}^{1}_{2D}$ ($\mathcal{P}_{cc}(\lambda)\mathrm{u} = 0$ type EVP)

(b) $\mathrm{P2}^{3}_{2D}$ ($\mathcal{P}_{cs}(\lambda)\mathrm{u} = 0$ type EVP)

(c) $\mathrm{P3}^{1}_{3D}$ ($\mathcal{P}_{pc}(\lambda)\mathrm{u} = 0$ type EVP)

(d) $\mathrm{P5}^{5}_{3D}$ ($\mathcal{P}_{ps}(\lambda)\mathrm{u} = 0$ type EVP)

(e) $\mathrm{P6}^{1}_{2D}$ ($\mathcal{P}_{ic}(\lambda)\mathrm{u} = 0$ type EVP)

Fig. 2. Timing results for eigenvalue problems with a matrix size larger than 1.2 million. The numerical experiments are conducted on a workstation equipped with an Intel 1.6 GHz Itanium II CPU, 32-gigabyte main memory, and an HP Unix operating system.

Table 1. Test problem properties. The table shows the names of the eigenvalue problems, QD geometry, QD geometric symmetry, dimension of discretization domains, QD effective mass, degree of the eigenvalue problems, properties of the coefficient matrices, and coefficient matrix size of the eigenvalue problems. The superscript and subscript of problem names denotes the degree and the discretization domain dimension, respectively. Unsym. and S.P.D. stands for unsymmetric and symmetric positive definite, respectively.

EVP	$\texttt{P1}^1_{2D}$	$\texttt{P2}^3_{2D}$	$\texttt{P3}^1_{3D}$	$\texttt{P4}^1_{3D}$	$\texttt{P5}^5_{3D}$	$\texttt{P6}^1_{2D}$
QD Geo.	cylinder	cylinder	pyramid	pyramid	pyramid	irregular
QD Geo. Sym.	radial	radial	non-rad.	non-rad.	non-rad.	radial
Disc. Dom.	2D	2D	3D	3D	3D	2D
QD Eff. Mass	constant	non-para.	constant	constant	non-para.	constant
Mtx. Degree (τ)	1	3	1	1	5	1
Mtx. Prop.	Unsym.	Unsym.	S.P.D.	S.P.D.	Sym.	S.P.D.
Mtx. Size	$1,200,000$	$1,228,150$	$5,216,783$	$77,315$	$5,216,783$	$1,228,150$

Remark. As mentioned in [29], the scheme involving (17) is not efficient for solving linear eigenvalue problems. We have similar observations for higher order polynomial eigenvalue problems and we believe the reasons behind this are similar to the linear cases. Using the definition of residual $r_k = \mathcal{P}(\theta_k)u_k$, we can rewrite (17) as

$$t = \eta_k \mathcal{P}(\theta_k)^{-1} p_k - u_k.$$

Such choice is actually equivalent with $t = \mathcal{P}(\theta_k)^{-1}p_k$, as t is made orthogonal to u_k afterwards. Let $\tilde{t}$ be an approximated solution of $\mathcal{P}(\theta_k)t = p_k$. The angle between $\tilde{t}$ and u_k may be small. Consequently, we do not expect that the subspace expansion would be efficient. Our numerical experiments of the five test problems verify this conjecture. The norm of residuals can only be reduced to 10^{-2}.

5.2. Effects of Preconditioning

We have observed that $\mathcal{S}_{OneLS}$ outperforms another two schemes while using the preconditioner SSOR(ω). In this subsection, we further compare the performance of preconditioners, including SSOR(ω), ILU(ℓ) (incomplete LU factorization), ICC(ℓ) (incomplete Cholesky factorization), and Jacobi when using the the $\mathcal{S}_{OneLS}$ scheme. The numerical results for solving problems $\texttt{P1}^1_{2D}$, $\texttt{P2}^3_{2D}$, $\texttt{P3}^1_{3D}$, $\texttt{P5}^5_{3D}$, and $\texttt{P6}^1_{2D}$ are shown in Table 2. Note that we have scanned the parameter ω of SSOR(ω) from 0.8 to 1.98 and the fill-in level ℓ of ILU(ℓ) and ICC(ℓ) from 0 to 12 for each of the test problems. However, Table 2 presents only the particular parameters that achieve better timing results. The table suggests the following observations.

Table 2. The total Jacobi-Davidson iteration numbers (Itno) and CPU times in second (Time) used for solving the test problems $\texttt{P1}^1_{2D}$, $\texttt{P2}^3_{2D}$, $\texttt{P3}^1_{3D}$, $\texttt{P4}^1_{3D}$, $\texttt{P5}^5_{3D}$, and $\texttt{P6}^1_{2D}$ by Algorithm 1, scheme $\mathcal{S}_{OneLS}$ and different preconditioners. The column "ω or ℓ" shows the parameter ω in the preconditioner SSOR(ω) or the fill-in level ℓ in the preconditioners ILU(ℓ) and ICC(ℓ). The timing results are computed by summing the five smallest positive eigenvalues in each of the test problems

(a) 2D Problems

	$\texttt{P1}^1_{2D}$			$\texttt{P2}^3_{2D}$			$\texttt{P6}^1_{2D}$		
Precond.	ω or ℓ	Itno	Time	ω or ℓ	Itno	Time	ω or ℓ	Itno	Time
SSOR(ω)	1.80	285	1,706	1.25	367	1,440	1.85	458	3,897
	1.85	256	1,536	1.30	292	1,077	1.90	392	3,304
	1.90	232	1,388	1.35	395	1,734	1.95	388	3,258
	1.95	1,213	11,206	1.40	377	1,584	1.98	530	4,548
ILU(ℓ)	5	148	1,415	5	157	867	5	161	1,798
	6	126	1,309	**6**	**85**	**483**	**6**	**150**	**1,767**
	7	**112**	**1,253**	7	100	570	7	143	1,782
	8	106	1,278	8	102	631	8	136	1,792
ICC(ℓ)	-	-	-	-	-	-	0	446	3,148
Jacobi	-	2,427	11,982	-	837	2,588	-	8,580	48,590

(b) 3D Problems

	$\texttt{P3}^1_{3D}$			$\texttt{P4}^1_{3D}$			$\texttt{P5}^5_{3D}$		
Precond.	ω or ℓ	Itno	Time	ω or ℓ	Itno	Time	ω or ℓ	Itno	Time
SSOR(ω)	1.80	86	1,905	1.55	50	11	1.80	98	4,087
	1.85	**82**	**1,818**	**1.60**	**48**	**11**	**1.85**	**98**	**3,831**
	1.90	93	2,090	1.65	51	12	1.90	101	4,028
	1.95	114	2,659	1.70	50	12	1.95	97	4,036
ILU(ℓ)	0	141	3,273	0	53	12	0	102	3,981
	1	106	3,196	1	52	17	1	83	4,008
	2	89	3,284	2	52	24	2	72	4,231
ICC(ℓ)	0	143	2,990	0	54	12	0	130	4,662
	1	101	2,627	1	51	14	1	99	4,394
	2	85	2,887	2	50	20	2	94	5,093
Jacobi	-	388	6,342	-	106	16	-	312	9,633

- For $\texttt{P1}^1_{2D}$, $\texttt{P2}^3_{2D}$, and $\texttt{P6}^1_{2D}$, the preconditioner ILU(6) or ILU(7) results in the best timing results. But, for $\texttt{P3}^1_{3D}$ and $\texttt{P5}^5_{3D}$, SSOR(1.85) achieves the best timing results.
- For ILU and ICC, the best ℓ for $\texttt{P1}^1_{2D}$, $\texttt{P2}^3_{2D}$, and $\texttt{P6}^1_{2D}$ is either 6 or 7. However, the best ℓ for $\texttt{P3}^1_{3D}$ and $\texttt{P5}^5_{3D}$ is 0 or 1.

The above behaviors are mainly due to the bandwidths of the corresponding coefficient matrices. In $\texttt{P1}^1_{2D}$, $\texttt{P2}^3_{2D}$, and $\texttt{P6}^1_{2D}$, the discretizations are associated

with two-dimensional planes and thus have smaller bandwidths. In contrast, the matrices associated with $\texttt{P3}^1_{3D}$ and $\texttt{P5}^5_{3D}$ have larger bandwidths due to the three-dimensional discretization. Figure 3-(a), Figure 4, and Figure 3-(c) illustrate the sparsity of the coefficient matrices associated with $\texttt{P1}^1_{2D}$, $\texttt{P2}^3_{2D}$, and $\texttt{P6}^1_{2D}$ with smaller matrix sizes, respectively. Figure 3-(b) and Figure 5 illustrate the sparsity of the coefficient matrices associated with $\texttt{P3}^1_{3D}$ and $\texttt{P5}^5_{3D}$ with smaller matrix sizes, respectively.

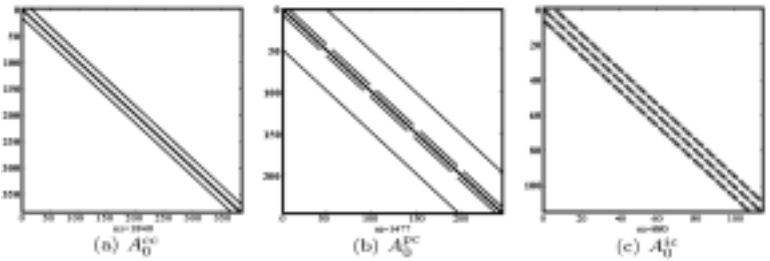


Fig. 3. Sparsity of the coefficient matrices A_0^{cc}, A_0^{pc}, and A_0^{ic}. (a) A_0^{cc} with matrix size 384, (b) A_0^{pc} with matrix size 245, and (c) A_0^{ic} with matrix size 114.

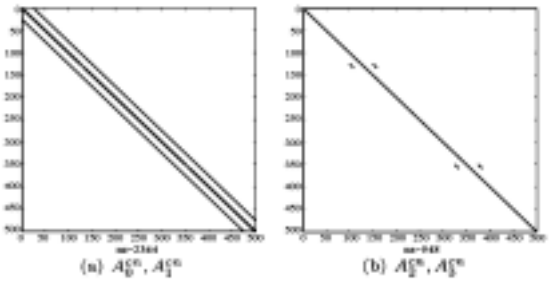


Fig. 4. Sparsity of the coefficient matrices for $\mathcal{P}_{cn}(\lambda)u = 0$ with matrix size 500. (a) A_0^{cn} and A_1^{cn}, (b) A_2^{cn} and A_3^{cn}.

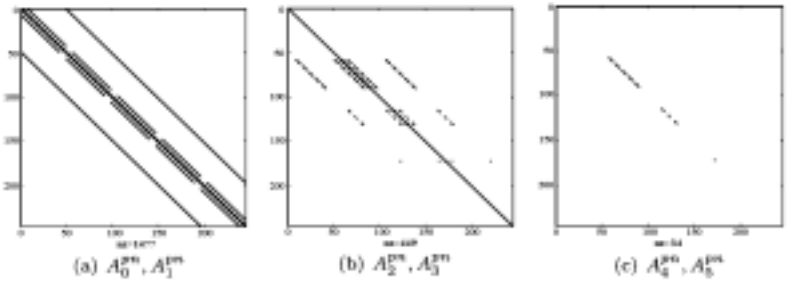


Fig. 5. Sparsity of the coefficient matrices for $\mathcal{P}_{pn}(\lambda)u = 0$ with matrix size 245. (a) A_0^{pn} and A_1^{pn}, (b) A_2^{pn} and A_3^{pn}, (c) A_4^{pn} and A_5^{pn}.

Table 3. Numerical results for solving $\texttt{P1}^1_{2D}$. The notation "*" means that the method does not converge. Since the coefficient matrices are unsymmetric, the Cholesky-based direct solver and ICC preconditioners are not used

(a) Explicit restarting

	Direct		GMRES			
	LU		ILU(6)		SSOR(1.85)	
Rstno	Itno	Time	Itno	Time	Itno	Time
10	29	291	-	-	-	-
15	12	251	-	-	-	-
20	6	216	-	-	-	-
25	3	186	4	8,690	-	-
30	3	203	3	7,870	3	8,386
35	2	189	2	6,395	2	6,721
40	1	164	1	3,934	1	4,073
45	1	171	1	4,273	1	4,558
50	1	178	1	4,748	1	5,031

(b) Implicit restarting

	Direct		GMRES			
	LU		ILU(6)		SSOR(1.85)	
Rstno	Itno	Time	Itno	Time	Itno	Time
10	11	165	14	5,251	*	*
15	5	164	5	4,465	*	*
20	3	166	3	4,469	*	*
25	2	165	3	5,723	*	*
30	2	176	2	5,162	*	*
35	2	188	2	6,015	*	*
40	1	166	1	3,903	*	*
45	1	173	1	4,369	*	*
50	1	180	1	4,830	*	*

(c) Krylov Schur restarting

	Direct		GMRES			
	LU		ILU(6)		SSOR(1.85)	
Rstno	Itno	Time	Itno	Time	Itno	Time
10	10	163	12	5,334	12	5,527
15	5	161	6	4,669	5	4,264
20	3	159	4	4,620	4	4,745
25	2	159	3	1,656	3	4,825
30	2	169	2	4,164	2	4,436
35	2	179	2	3,819	2	5,128
40	1	165	1	3,819	1	4,043
45	1	172	1	4,275	1	4,527
50	1	180	1	4,738	1	5,017

Table 4. Numerical results for solving $\texttt{P4}^1_{3D}$. The notation "*" means that the method does not converge.

(a) Explicit restarting

	Direct				GMRES					
	LU		Cholesky		ILU(2)		SSOR(1.30)		ICC(2)	
Rstno	Itno	Time	Itno	Time	Itno	Time	Itno	Time	Itno	Time
10	61	415	68	4,520	147	581	49	143	69	183
15	26	376	18	4,046	27	191	1981	11,003	19	91
20	12	338	12	4,027	13	131	252	1,871	13	87
25	10	346	10	4,056	23	308	*	*	378	3,444
30	8	344	8	4,040	7	113	8	87	7	76
35	6	335	18	4,700	6	114	6	79	6	77
40	4	320	4	3,913	9	199	17	260	8	117
45	5	341	5	4,026	15	377	439	7,599	17	285
50	5	349	5	4,087	115	3,160	10	194	68	1,264

(b) Implicit restarting

	Direct				GMRES					
	LU		Cholesky		ILU(2)		SSOR(1.30)		ICC(2)	
Rstno	Itno	Time	Itno	Time	Itno	Time	Itno	Time	Itno	Time
10	17	290	19	3,782	20	47	20	33	20	31
15	7	290	8	3,773	9	46	8	30	9	31
20	5	290	5	3,773	5	43	5	30	5	29
25	3	287	4	3,786	4	47	4	32	4	31
30	3	292	3	3,779	3	45	3	32	3	31
35	2	287	3	3,804	3	52	3	36	3	35
40	2	291	2	3,786	2	43	2	30	2	29
45	2	294	2	3,973	2	49	2	34	2	34
50	2	298	2	3,809	2	55	2	39	2	38

(c) Krylov Schur restarting

	Direct				GMRES					
	LU		Cholesky		ILU(2)		SSOR(1.55)		ICC(2)	
Rstno	Itno	Time	Itno	Time	Itno	Time	Itno	Time	Itno	Time
10	15	289	15	3,788	17	48	16	32	17	33
15	8	288	8	3,778	9	40	9	29	9	28
20	6	288	6	3,791	6	40	6	28	6	27
25	4	287	4	3,807	5	42	5	30	5	29
30	4	291	4	3,797	4	43	4	30	4	29
35	3	290	3	3,789	3	40	3	29	3	28
40	3	293	2	3,769	3	46	3	33	3	32
45	2	289	2	3,783	2	39	2	29	2	27
50	2	292	2	3,800	2	44	2	33	2	30

Table 5. Numerical results for solving $\texttt{P6}^1_{2D}$. Results of explicit restarting are not shown here as the method does not converge in most of the cases. The results of Cholesky-based direct method cannot be completed in our computer due to the out of memory errors. The results of GMRES with SSOR are not shown here as the method does not converge. Note that the fill-ins of the Cholesky factorizations in $\texttt{P6}^1_{2D}$ cause out of memory storage errors in our experiments. Therefore, no Cholesky results are listed here even though the coefficient matrix is symmetric positive definite

(a) Implicit restarting

	Direct		GMRES			
	LU		ILU(6)		ICC(0)	
Rstno	Itno	Time	Itno	Time	Itno	Time
10	19	298	22	5,553	22	15,165
15	6	275	7	4,281	7	11,924
20	4	284	5	4,832	5	13,244
25	3	288	3	4,339	3	11,965
30	2	279	3	5,230	3	14,836
35	2	296	2	4,423	2	12,474
40	2	312	2	5,100	2	14,623
45	1	273	2	5,730	2	16,847
50	1	282	1	3,564	1	10,143

(b) Krylov Schur restarting

	Direct		GMRES			
	LU		ILU(6)		ICC(0)	
Rstno	Itno	Time	Itno	Time	Itno	Time
10	12	274	18	5,323	18	148,803
15	5	263	8	4,343	8	12,284
20	4	272	5	4,015	5	10,956
25	3	277	4	4,201	4	11,848
30	2	274	3	4,116	3	11,783
35	2	288	3	4,740	2	11,875
40	1	268	3	5,423	2	11,877
45	1	278	2	4,712	2	13,410
50	1	289	1	3,613	1	10,262

5.3. Performance of Krylov subspace methods

We also solve the linear eigenvalue problems $\texttt{P1}^1_{2D}$, $\texttt{P4}^1_{3D}$ and $\texttt{P6}^1_{2D}$ by the explicit restarting Lanczos/ Arnoldi method (ER), implicit restarting the Lanc-

zos/Arnoldi method (IR), and Krylov-Schur method (KS). Note that another linear problem $\texttt{P3}^1_{3D}$ is skipped as the matrix size is too large for the direct solvers in our computers.

We use the ER and KS provided by the software package SLEPc (Scalable Library for Eigenvalue Problem Computations) [11, 12]. For IR, we use the ARPACK [22] wrapper included in SLEPc. We use PETSc (Portable, Extensible Toolkit for Scientific Computation) [1] to solve the linear system solvers and to perform the preconditionings within ER, KS, and IR. In particular, to solve the associated linear systems within these three methods, we consider (i) direct solvers based on LU or Cholesky factorization and (ii) GMRES iterative solvers with preconditioners SSOR, ILU, or ICC.

The results for computing the five smallest positive eigenvalues of $\texttt{P1}^1_{2D}$, $\texttt{P4}^1_{3D}$ and $\texttt{P6}^1_{2D}$ are summarized in Tables 3, 4 and 5. In the tables, "Rstno", "Itno", and "Time" stand for the restarting number p in Algorithm 2, the number of the while-loop starting from line 2 to line 2 in Algorithm 2, and total CPU time in seconds for computing five target eigenpairs, respectively. Note that we have scanned the parameter ω in SSOR from 0.8 to 1.98 and the fill-in level parameter ℓ of ILU and ICC from 0 to 12 in our numerical experiments. The tables only present the results with better timing results. We highlight some observations from the tables as follows.

- KS outperforms ER and IR in almost all cases. In particular, KS and IR are more efficient and numerically stable than ER. Furthermore, the performance of KS is slightly better than that of IR.

- The bandwidths play an important role in determining efficiency of the linear system solvers. In particular, for $\texttt{P1}^1_{2D}$ and $\texttt{P6}^1_{2D}$, the Krylov subspace methods with direct linear solvers are better than the Krylov subspace methods with iterative linear solvers. For $\texttt{P4}^1_{3D}$, the Krylov subspace methods with iterative linear solvers perform better.

  Such behavior is again due to the bandwidths of the coefficient matrices. As the discretizations of these two problems involve only two-dimensional planes, the corresponding bandwidths of matrices A_0^{cc} and A_0^{ic} in (6) and (10) are small. Direct solvers remain efficient even when dense band matrices are introduced after LU or Cholesky factorizations have been performed. In contrast, the discretization of $\texttt{P4}^1_{3D}$ involves all three dimensions and the corresponding bandwidth is large. In such cases, direct solvers are not efficient due to the fill-ins of LU or Cholesky factorizations. See Figure 3 for sparsity examples A_0^{cc}, A_0^{pc}, and A_0^{ic}.

We conclude this subsection with the following remarks.

Table 6. Numerical results for solving $\texttt{P3}^1_{3D}$ by Krylov subspace methods "without" shift-and-invert

	Krylov Schur restarting		Implicit restarting	
Rstno	Itno	Time	Itno	Time
20	1052	19,055	449	9,646
25	409	**11,430**	250	7,956
30	305	11,734	198	8,645
35	268	13,771	118	**6,770**
40	183	12,055	95	7,132

Table 7. The most efficient scheme combinations for different test problems

	3D	2D
Higher order EVP	JD + $\mathcal{S}_{OneLS}$ + SSOR ($\texttt{P5}^5_{3D}$)	JD + $\mathcal{S}_{OneLS}$ + ILU ($\texttt{P2}^3_{2D}$)
Standard/General EVP	JD + $\mathcal{S}_{OneLS}$ + SSOR ($\texttt{P3}^1_{3D}$, $\texttt{P4}^1_{3D}$)	KS + LU-based Direct ($\texttt{P1}^1_{2D}$, $\texttt{P6}^1_{2D}$)

Table 8. Timing in seconds and the corresponding percentage (shown within parentheses) breakdown of the key components in the LU- and Cholesky-based direct solvers for solving $\texttt{P4}^1_{3D}$

	LU	Cholesky
Time for Symbolic factorization	3 (1%)	2,378 (63%)
Time for numerical factorization	273 (90%)	1,257 (33%)
Total CPU time	304	3,773
Fill-in ratio	102%	309%

1. For the problems with symmetric positive definite coefficient matrices (e.g. $\texttt{P3}^1_{3D}$ and $\texttt{P6}^1_{2D}$), the eigenvalues are positive and the five smallest positive eigenvalues can be computed "*without*" using shift-and-invert. This approach involves matrix-vector multiplications and does not need to solve any linear system. However, our numerical experiments suggest that such an approach is not efficient.

   In particular, if we do not perform shift-and-invert in the Krylov subspace methods, the methods either converge slowly ($\texttt{P3}^1_{3D}$, as shown in Table 7) or fail to convergence ($\texttt{P6}^1_{2D}$). Table 7 also suggests that the methods take much more CPU time as compared to the Jacobi-Davidson methods shown Table 2.

2. Direct solvers with LU factorization outperforms Cholesky factorization, as shown in Table 4. This is because a Cholesky factorization spends much

more time in symbolic factorization and leads to a much larger fill-in ratio in order to maintain the symmetry of the matrices. See Table 6 for a detailed breakdown analysis for LU- and Cholesky-based direct solvers.

5.4. Overall Comparisons

Observing the results shown in Tables 2, 3, 4, 5, we can make an overall comparison between the Jacobi-Davidson methods and the Krylov subspace methods and conclude the best scheme combinations for the different types of problems in Table 8. As discussed above, the bandwidth of the coefficient matrices is the key component that affects the choice of schemes.

6. Conclusion

We consider degree 1, 3, and 5 eigenvalue problems arising in numerical simulations of nano-scale quantum dots. We have shown that a polynomial Jacobi-Davidson method can solve all of these problems without linearizing the higher degree problems. As the efficiency of the polynomial Jacobi-Davidson method mainly relies on solving the correction equation, we have discussed three schemes regarding how to compute the approximate solutions of the correction equations. We have also conducted intensive numerical experiments by using the Jacobi-Davidson and Krylov subspace methods with various linear solvers and preconditioners. The numerical results suggest the most efficient scheme combinations for different types of problems, which are shown in Table 8. We also find that the bandwidth of the coefficient matrices is the key component that affects the choice of schemes.

It is possible to further improve the solver efficiencies by parallel computing. We then need to find efficient preconditioners that are suitable to the target problems and the particular parallel architectures of interest. The best scheme combinations for different types of problems may also change consequently on parallel computers.

Acknowledgment

We are grateful to Wen-Wei Lin for many helpful discussions and comments. 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, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith and H. Zhang, PETSc users manual, *Argonne National Laboratory, Tech. Rep. ANL-95/11-Revision*, **2(5)** (2002).

2. G. Burkard, D. Loss and D. P. DiVincenzo, Couple quantum dots as quantum gates, *Phys. Rev. B*, **59** (1999), 2070-2078.
3. E. K.-W. 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** (2008), 237-252.
4. M. A. Cusack, P. R. Briddon and M. Jaros, Electronic structure of InAs/GaAs self-assembled quantum dots, *Phys. Rev. B*, **54** (1996), 2300-2303.
5. E. A. de Andrada e Silva, G. C. La Rocca and F. Bassani, Spin-orbit splitting of electronic states in semiconductor asymmetric quantum wells, *Phys. Rev. B*, **55(24)** (1997), 16293.
6. D. El-Moghraby, R. G. Johnson and P. Harrison, Calculating modes of quantum wire and dot systems using a finite differencing technique, *Computer Physics Communications*, **150** (2003), 235-246.
7. J. L. Fattebert and M. B. Nardelli, Finite difference methods for ab initio electronic structure and quantum transport calculations of nanostructures, *Handbook of Numerical Analysis*, **10** (2003), 571-612.
8. J. W. Gray, D. Childs, S. Malik, P. Siverns, C. Roberts, P. N. Stavrinou, M. Whitehead, R. Murray and G. Parry, Quantum dot resonant cavitylight emitting diode operating near 1300 nm, *Electron. Lett.*, **35** (1999), 242.
9. L. Harris, D. J. Mowbray, M. S. Skolnick, M. Hopkinson and G. Hill, Emission spectra and mode structure of InAs/GaAs self-organized quantum dot lasers, *Appl. Phys. Lett.*, **73** (1998), 969-971.
10. R. Heitz, M. Veit, N. N. Ledentsov, A. Hoffmann, D. Bimberg, V. M. Ustinov, P. S. Kopév and Zh. I. Alferov, Energy relaxation by multiphonon processes in InAs/GaAs quantum dots, *Phys. Rev. B*, **56** (1997), 10435-10445.
11. V. Hernandez, J. E. Roman, A. Tomas and V. Vidal, *SLEPc Users Manual*, Tech. Report DSIC-II/24/02 - Revision 2.3.2, D. Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, 2006.
12. V. Hernandez, J. E. Roman and V. Vidal, SLEPc: A Scalable and Flexible Toolkit for the Solution of Eigenvalue Problems, *ACM Trans. Math. Software*, **31** (2005), 351-362.
13. M. Hochbruck and D. Lochel, *A multilevel Jacobi-Davidson method for polynomial pde eigenvalue problems arising in plasma physics*, Technical Report, Math. Institut HHU Dusseldorf, 2009.
14. T.-M. Huang, W.-W. Lin and J. Qian, Structured Algorithms for Palindromic Quadratic Eigenvalue Problems Arising in Vibration of Fast Trains, *SIAM J. Matrix Anal. Appl.*, **30** (2009), 1566-1592.
15. T.-M. Hwang, W.-W. Lin, W.-C. Wang and W. Wang, Numerical Simulation of Three Dimensional Pyramid Quantum Dot, *Journal of Computational Physics*, **196** (2004), 208-232.

16. T.-M. Hwang and W. Wang, Energy states of vertically aligned quantum dot array with nonparabolic effective mass, *Comput. Math. Appl.*, **49** (2005), 39-51.

17. T.-M. Hwang, W.-C. and W. Wang, Numerical schemes for three-dimensional irregular shape quantum dots over curvilinear coordinate systems, *J. Comput. Phys.*, **226** (2007), 754-773.

18. T.-M. Hwang, W.-H. Wang and W. Wang, Efficient Numerical Schemes for Electronic States in Coupled Quantum Dots, *J. Nanosci. Nanotechnol.*, **8** (2008), 3695-3709.

19. G. Iannaccone, A. Trellakis and U. Ravaioli, Simulation of a quantum-dot flash memory, *J. Appl. Physi.*, **84(9)** (1998), 5032-5036.

20. L. Jacak, P. Hawrylak and A. Wójs, *Quantum Dots*, Springer, Berlin, 1998.

21. R. B. Lehoucq and D. C. Sorensen, Deflation techniques for an implicitly restarted Arnoldi iteration, *SIAM J. Matrix Anal. Appl.*, **17** (1996), 789-821.

22. R. B. Lehoucq and D. C. Sorensen and C. Yang, *ARPACK USERS GUIDE: Solution of large scale eigenvalue problems with implicitely restarted Arnoldi methods*, SIAM, Philadelphia, 1998.

23. Y. Li, O. Voskoboynikov, C. P. Lee and S. M. Sze, Computer simulation of electron energy levels for different shape InAs/GaAs semiconductor quantum dots, *Computer Physics Communications*, **141** (2001), 66-72.

24. S. Maimon, E. Finkman, G. Bahir, S. E. Schacham, J. M. Garcia and P. M. Petroff, Intersublevel transitions in InAs/GaAs quantum dots infrared photodetectors, *Appl. Phys. Lett.*, **73** (1998), 2003-2005.

25. G. Medeiros-Ribeiro, J. M. Garcia and P. M. Petroff, Charging dynamics of InAs self-assembled quantum dots, *Phys. Rev. B*, **56** (1997), 3609-3612.

26. F. M. Peeters and V. A. Schweigert, Two-electron quantum disks, *Phys. Rev. B*, **53** (1996), 1468-1474.

27. C. Pryor, Eight-band calculations of strained InAs/GaAs quantum dots compared with one-, four-, and six-band approximations, *Phys. Rev. B*, **57** (1998), 7190-7195.

28. J. Shumway, L. R. C. Fonseca, J. P. Leburton, R. M. Martin and D. M. Ceperley, Electronic structure of self-assembled quantum dots: comparison between density functional theory and diffusion quantum Monte Carlo, *Physica E*, **8** (2000), 260-268.

29. G. L. G. Sleijpen and H. A. van der Vorst, A Jacobi-Davidson iteration method for linear eigenvalue problems, *SIAM J. Matrix Anal. Appl.*, **17(2)** (1996), 401-425.

30. G. W. Stewart, *Matrix algorithms, Volume II: eigensystems*, SIAM, Philadelphia, 2001.

31. G. W. Stewart, A Krylov-Schur Algorithm for Large Eigenproblems, *SIAM J. Matrix Anal. Appl.*, **23** (2001), 601-614.

32. ———, Addendum to "A Krylov-Schur Algorithm for Large Eigenproblems", *SIAM J. Matrix Anal. Appl.*, **24** (2002), 599-601.

33. W. Wang, T.-M. Hwang and J.-C. Jang, A second-order finite volume scheme for three dimensional truncated pyramidal quantum dot, *Comput. Phys. Comm.*, **174** (2006), 371-385.

34. W. Wang, T.-M. Hwang, W.-W. Lin and J.-L. Liu, Numerical methods for semiconductor heterostructures with band nonparabolicity, *J. Comput. Phys.*, **190** (2003), 141-158.

35. A. J. Williamson and A. Zunger, InAs quantum dots: Predicted electronic structure of free-standing versus GaAs-embedded structures, *Phys. Rev. B*, **59** (1999), 15819-15824.

Tsung-Ming Huang
Department of Mathematics,
National Taiwan Normal University,
Taipei 116, Taiwan
E-mail: min@math.ntnu.edu.tw

Weichung Wang
Department of Mathematics,
National Taiwan University,
Taipei 106, Taiwan
E-mail: wwang@math.ntu.edu.tw

Chang-Tse Lee
Department of Mathematics,
National Taiwan Normal University,
Taipei 116, Taiwan
E-mail: otherchang@yahoo.com.tw