A Parallel Two-level Polynomial Jacobi-Davidson Algorithm for Large Sparse PDE Eigenvalue Problems

Feng-Nan Hwang
Department of Mathematics
National Central University, Jhongli, Taiwan

Xiao-Chuan Cai (CU-Boulder, USA)
Yu-Fen Cheng (NCHC, Taiwan)
Tsung-Min Huang (NTNU, Taiwan)
Weichung Wang (NTU, Taiwan)
Tao Zhao (CU-Boulder, USA)

Supported by National Science Council of Taiwan
Outline

1. Polynomial eigenvalue problem
2. Two-level polynomial Jacobi-Davidson (JD) algorithm
   - Nested iterations (grid sequencing)
   - Two-level hybrid Schwarz-based preconditioner
3. Parallel implementation:
   - PJDPack: PETSc based JD polynomial eigensolver
4. Numerical results
5. Concluding remarks.
Polynomial eigenvalue problems (EVPs)

- Large sparse polynomial eigenvalue problems

\[ A(\lambda)x = \sum_{i=0}^{\tau} \lambda^i A_i x = (\lambda^{\tau} A_\tau + \cdots + \lambda^2 A_2 + \lambda A_1 + A_0) x = 0, \]

where

- \( \tau \) is the degree of the polynomial.
- \( A_i \) are coefficient matrices arising from some discretization of PDEs.
- \( \lambda \) is an eigenvalue, and \( x \) is a corresponding eigenvector.

- Computational challenges:
  - Nonlinearity: No simple canonical form is available
    - Generalized/linear EVPs: (generalized) Schur form
      helpful for developing new algorithms,
      e.g. locking, purging, or implicit restart, etc.
  - Large dimension (\( 1M \sim 100M \)) of coefficient matrix.
  - Only few interior or exterior spectrum of interests in applications.
Special cases of poly. EVPs and their applications

- Generalized EVP ($\tau = 1$): $Jx = \lambda Mx$.
  Bifurcation analysis of incompressible flows. Huang and Hwang, '10.

- Quadratic EVPs ($\tau = 2$): $\lambda^2 Mx + \lambda Dx + Kx = 0$.
  Vibration analysis of a fast train. E.K.-W. Chu et al., '08.

- Cubic EVPs ($\tau = 3$). *Our target application*
  Noise control in acoustical engineering
Acoustic eigenvalue problem

\[ \begin{align*}
\lambda^2 p &= c^2 \Delta p \quad \text{in } \Omega, \\
p &= 0 \quad \text{in } \Gamma_O, \\
\frac{\partial p}{\partial n} &= 0 \quad \text{in } \Gamma_R, \\
\frac{\partial p}{\partial n} &= \frac{-\rho \lambda^2}{\alpha + \lambda \beta} p \quad \text{in } \Gamma_A.
\end{align*} \]

- \( p \): eigenvector (pressure perturbation).
- \( \lambda \): eigenvalue (\( \text{Im}(\lambda) \): vibration angular frequency).
- \( \rho \): fluid density, assumed to be constant.
- \( c \): sound speed in the air.
- \( \alpha \): elastic coefficient.
- \( \beta \): damping coefficient.
- \( n \): outward normal vector.
Cubic acoustic algebraic eigenvalue problem

- Acoustic eigenvalue problem with B.C.:

\[ \mathcal{R}(\lambda) \equiv (\lambda^2 M + \frac{\lambda^2}{\alpha + \lambda\beta} D + K)u = 0, \]

\[ \times (\alpha + \lambda\beta)I_n \Rightarrow \mathcal{A}(\lambda)u \equiv (\sum_{i=0}^{\tau} \lambda^i A_i)u = 0, \quad (\tau = 3), \]

where \( A_0 = \alpha K, \ A_1 = \beta K, \ A_2 = \alpha M + C, \) and \( A_3 = \beta M. \)

Some remarks:
- \( \mathcal{A}(\lambda) = 0 \) has extraneous roots \( \lambda = -\alpha/\beta \) w/ algebraic multiplicity \( n. \)
- In the case that \( \Gamma_O = \emptyset \), a zero eigenvalue exists for both forms.
- It can be shown that \( \text{Re}(\lambda) < 0 \) for all eigenvalues where \( \lambda \neq 0, \) \Rightarrow acoustic vibrations are damped due to the effect of the viscoelastic materials.
Classical approach for solving polynomial EVPs

- Reformulate the original poly. EVP as the linearized companion EVP:

\[
\begin{bmatrix}
0 & I & 0 & \cdots & 0 \\
0 & 0 & I & \cdots & 0 \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & I \\
A_0 & A_1 & A_2 & \cdots & A_{\tau-1}
\end{bmatrix}
\begin{bmatrix}
\lambda S \\
\lambda^2 S \\
\vdots \\
\lambda^{\tau-1} S \\
\lambda S \\
\lambda^2 S \\
\vdots \\
\lambda^{\tau-1} S
\end{bmatrix}
= \lambda
\begin{bmatrix}
I & 0 & 0 & \cdots & 0 \\
0 & I & 0 & \cdots & 0 \\
0 & 0 & I & \cdots & 0 \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & -A_\tau \\
\lambda S \\
\lambda^2 S \\
\vdots \\
\lambda^{\tau-1} S
\end{bmatrix}
\]

and solved by some Krylov subspace method or QZ method.

- Pros
  - Several state-of-the-art packages available, e.g. polyeig in Matlab, parallel ARPACK, SLEPc etc.

- Cons
  - Increase the size of the problem \((dim(A) \times \tau)\).
  - The system is more ill-conditioned
    - Introduce the computational issues on accuracy loss and convergence of the algorithm.
  - Require an expensive invert operator and hard to be preconditioned.
  - Do not preserve any good structure of \(A_i\).
An alternative: Jacobi-Davidson (JD) algorithm

- **Linear EVPs**: Sleijpen and van der Vorst, ’96.
- **Polynomial EVPs**: Huang ’03, ’04, ’05, Meerbergen ’01, and Tisseur. ’01.
- **Pros**
  - Deal with the problem as the same size of $A_i$.
  - Target directly interior spectrum without using shift-and-invert. (Only requires some low cost inexact linear solves)
  - Preserve the structure of $A_i$.
  - Enjoy fast convergence.
  - Can be parallelized
- **Cons**
  - However, there was no parallel general-purpose JD based package for solving polynomial EVPs.
Polynomial JD algorithm

- Given an **initial search space** and let $V$ be the current search space.
- Assume that $(\lambda, u)$ is the current approximate eigenpair, which is not close to the exact one.
- Then the new eigenpair, $(\lambda_{\text{new}}, u_{\text{new}})$, can be obtained through the following two steps:
  1. Expand the search space $V = [V, v]$ by solving the **correction eq.**
     \[ (I - \frac{pu^*}{u^*p})A(\lambda)(I - uu^*)t = -r \text{ such that } t \perp u \]
     Here $r = A(\lambda)u$ and $p = A'(\lambda)u$.
     Then $t$ is orthogonalized against $V$, and set $v = t/\|t\|$.
  2. Perform the **Rayleigh-Ritz procedure**: find a proper Ritz pair $(\theta, s)$ by solving the projected poly. EVP
     \[ (V^T A(\theta) V) s = 0 \]
     Then set $\lambda_{\text{new}} = \theta$ and compute $u_{\text{new}} = V s$. 

Factors affect robustness and efficiency of PJD

- Finding an initial basis in the search space
  - Randomized vector: unpredictable
  - Fourier mode: $V = \exp(kij\theta)$:
    only available for simple geometry, and bc
  - Multilevel approach: Interpolation of coarse grid eigenvector

- Solution quality of the correction equation
  - Designing an efficient solver is the current research focus
    - Heuveline and Bertsch '00, non-selfadjoint elliptic EVPs,
      Multigrid (ILU(0) and Gauss-Seidel smoother)
    - Feng '01, Generalized EVPs for structural dynamics,
      Multigrid (ILU(0) smoother) preconditioner + FGMRES
    - Arbenz '06, Generalized EVPs for Maxwell eq.
      Algebraic multigrid preconditioner + CG
    - Hwang et al. '10, Cubic and quintic EVPs for quantum dot simulation,
      one-level additive Schwarz preconditioner + GMRES

- Ritz-pair selection strategies
  - Hochbruck and Löchel., '10. Similarity measure, using eigenpairs to avoid picking up the unphysical Ritz value.
Construction of initial search space: two-level approach

Algorithm 1 Two-level Polynomial JD algorithm

Input: Coefficient matrices $A_{i}^{(l)}$ for $i = 0, \ldots, \tau$, $l = H$ (coarse) or $h$ (fine), the number of desired eigenvalues $k$, targets $\mu$.

Output: the desired eigenpairs $(\lambda_j, f_j)$ for $j = 1, \ldots, k$.

1. Solve $A^H(\lambda^H)\phi^H = 0$ by PJD algorithm on the coarse grid
2. Select $m$ eigenpairs close to the target
3. Set $\phi_i^h = I_H^h \phi_i^H$ for $i = 1, \ldots, m$.
4. Normalize $\hat{\phi}_i^h$ by $\tilde{\phi}_i^h = \hat{\phi}_i^h / \|\phi_i^h\|_2$
5. Set the initial search space for fine level, $V_{int}^h = [\tilde{\phi}_1^h, \ldots, \tilde{\phi}_m^h]$.
6. Solve $A^h(\lambda)f = 0$ by PJD algorithm on fine grid
Parallel correction equation solve

- The left preconditioned correction equation

\[ B_d^{-1} \left( I - \frac{pu^*}{u^*p} \right) A(\lambda)(I - uu^*)t = -B_d^{-1}r, \]

where the preconditioner, \( B_d^{-1} \) is defined as

\[ B_d = \left( I - \frac{pu^*}{u^*p} \right) B(I - uu^*) \]

Here \( B \) is an approximation of \( A(\lambda) \) in some sense.

- A parallel Krylov subspace type (KSP) method, such as GMRES or CG can be used to solve the correction equation. In general, only few KSP iterations, say 10 or 20, are sufficient for most of problems.
In practice, there is no need to explicitly form $B_d$, only preconditioning operation $z = B_d^{-1} y$ with $z \perp u$ for a given $y$, is required.

This can be done equivalently by solving $B_d y = z$.

Since $z \perp u$ and suppose $B$ is invertible, it can be shown

$$z = B^{-1} y - \eta B^{-1} p$$

where $\eta = \frac{u^* B^{-1} y}{u^* B^{-1} p}$

The operations $B^{-1} p$ and $u^* B^{-1} p$ need to be computed only once for each correction equation, and no need to recompute them in KSP iterations.
Additive Schwarz framework

\( \Omega_i^{H,0} \)

\( R_i^{H,\delta} : \Omega^H \rightarrow \Omega_i^{H,\delta} \)

\( I_h^H : \Omega^H \rightarrow \Omega_i^h \)

\( I_h^H : \Omega^h \rightarrow \Omega^H \)

\( \Omega_i^{h,0} \)

\( R_i^{h,\delta} : \Omega^h \rightarrow \Omega_i^{h,\delta} \)
Schwarz framework based $B^{-1}$

- **Fine** restricted additive Schwarz (RAS$_f$) preconditioner:

  $$B^{-1} \approx B^{-1}_{RAS_f} \equiv \left( \sum_{i=1}^{N_s} \left( R_i^{h,0} \right)^T \left( A_i^h \right)^{-1} R_i^{h,\delta} \right).$$

- Hybrid two-level restricted additive Schwarz (HRAS) preconditioner:

  $$B^{-1}_{HRAS} = I_H^h \left( A^H \right)^{-1} I_H^h + \left( I - I_H^h \left( A^H \right)^{-1} I_H^h A^h \right) B^{-1}_{RAS_f}$$

  $$\left( A^H \right)^{-1} \approx B^{-1}_{RAS_c} \equiv \sum_{i=1}^{N_s} \left( R_i^{H,0} \right)^T \left( B_i^H \right)^{-1} R_i^{H,\delta}.$$  

  $$\left( A^H \right)^{-1} \approx A^H z^c = w^c$$ solved by $B^{-1}_{RAS_c}$ preconditioned GMRES,

  where $B_i^H$ could be ILU($k$) of $A_i^H = R_i^{H,\delta} A^H (\lambda) (R_i^{H,\delta})^T$.  

F.-N. Hwang (Dept. of Math, NCU) Two-level PJD for PDE Poly. EVPs
The projected poly. EVP $M(\theta)s = \sum_{i=0}^{T} \theta^i M_i s = 0$ can be solved as a linearized projected EVP

$$
\begin{bmatrix}
0 & I & 0 & \cdots & 0 \\
0 & 0 & I & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & I \\
M_0 & M_1 & M_2 & \cdots & M_{T-1}
\end{bmatrix}
\begin{bmatrix}
s \\
\theta s \\
\theta^2 s \\
\vdots \\
\theta^{T-1} s
\end{bmatrix}
= \theta
\begin{bmatrix}
I & 0 & 0 & \cdots & 0 \\
0 & I & 0 & \cdots & 0 \\
0 & 0 & I & \cdots & 0 \\
0 & 0 & 0 & \cdots & -M_T \\
\theta s \\
\theta^2 s \\
\vdots \\
\theta^{T-1} s
\end{bmatrix}
\begin{bmatrix}
s \\
\theta s \\
\theta^2 s \\
\vdots \\
\theta^{T-1} s
\end{bmatrix}
$$

- The size is much smaller than the original $A_i$.
- On each processor, it is solved redundantly by QZ method in SLEPC.
PETSc-based PJDPack
Software developments

- **PJDPack** *(Parallel Jacobi-Davidson Package)*
  - based on PJD algorithm, along with restarting, locking, and preconditioning techniques.
  - built on top of PETSc and SLEPc and inherits their important features, portability and extensibility.

- **PETSc** *(Portable, Extensible Toolkit for Scientific Computation)*
  - Parallel package for the solution of PDEs.
  - Developed by Argonne National Lab in USA.
  - Support a variety of storage format for parallel matrices and provide many parallel Krylov subspace type methods as well as domain decomposed based preconditioners.
  - Open source for free download.

- **SLEPc** *(Scalable Library for Eigenvalue Problem Computations)*
  - Parallel package for standard, generalized, quadratic EVPs, and SVD.
  - Developed by Universidad Politecnica de Valencia in Spain.
  - Based on PETSc and provide several eigensolvers.
The JD algorithm can be classified into three main parts:

1. Linearized projected eigensolve (SLEPc).
2. Correction equation solve (PETSc). (Most expensive part.)
3. A sequence of basic linear algebra operations. (PETSc)
Numerical example 1: Cubic acoustic dissipative EVP

Physical parameters:
- \( a = 1 \text{m}, b = 0.75 \text{m} \)
- \( \rho = 1 \text{kg/m}^3, c = 340 \text{m/s} \)
- \( \alpha = 5 \times 10^4 \text{ N/m}^3, \beta = 200 \text{ Ns/m}^3 \)

PJD parameters:
- JD RTOL: 1e-10; JD ATOL: 1e-8
- JD target: 0.0 + 1900.0i
- Projected linearized eps solver: QZ (LAPACK)

Top bc: absorbing type;
Rest bcs: Neumann type

Wanted eigenvalues:
\[
0 = \frac{\text{Im}(\lambda)}{2\pi} < 600 \text{Hz}
\]
\[
\Rightarrow 0 < \text{Im}(\lambda) < 1200\pi \approx 3770
\]
The spectrum of mesh size $h = 1/32$
PJD residual history: Computing 10 eigs, $h = 1/256$
## Fixed-subdomain-size scalability

<table>
<thead>
<tr>
<th>Fine grid</th>
<th>np</th>
<th>1st coarse</th>
<th>2nd coarse</th>
<th>JD.lts (Ave.Corr.lts)</th>
<th>Time (s)</th>
<th>Scaled Ef (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>One-level PJD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>256×192</td>
<td>12</td>
<td></td>
<td></td>
<td>30 (30.0)</td>
<td>11.9</td>
<td>100</td>
</tr>
<tr>
<td>512×384</td>
<td>48</td>
<td></td>
<td></td>
<td>30 (30.0)</td>
<td>14.8</td>
<td>80</td>
</tr>
<tr>
<td>1024×768</td>
<td>192</td>
<td></td>
<td></td>
<td>30 (30.0)</td>
<td>26.8</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Two-level PJD</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Coarse grid: 32×24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>256×192</td>
<td>12</td>
<td>2.30</td>
<td>0.50</td>
<td>2 (16.0)</td>
<td>0.3</td>
<td>42</td>
</tr>
<tr>
<td>512×384</td>
<td>48</td>
<td>0.96</td>
<td>0.40</td>
<td>2 (27.0)</td>
<td>0.7</td>
<td>18</td>
</tr>
<tr>
<td>1024×768</td>
<td>192</td>
<td>5.85</td>
<td>2.13</td>
<td>3 (30.0)</td>
<td>1.6</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Coarse grid: 64×48</td>
<td></td>
<td></td>
</tr>
<tr>
<td>512×384</td>
<td>12</td>
<td>5.80</td>
<td>0.01</td>
<td>2 (25.0)</td>
<td>2.5</td>
<td>100</td>
</tr>
<tr>
<td>1024×768</td>
<td>48</td>
<td>6.07</td>
<td>1.10</td>
<td>2 (29.0)</td>
<td>3.3</td>
<td>73</td>
</tr>
<tr>
<td>2048×1536</td>
<td>192</td>
<td>18.98</td>
<td>0.67</td>
<td>3 (30.0)</td>
<td>7.1</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Coarse grid: 128×96</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1024×768</td>
<td>12</td>
<td>1.63</td>
<td>0.49</td>
<td>3 (30.0)</td>
<td>20.3</td>
<td>100</td>
</tr>
<tr>
<td>2048×1536</td>
<td>48</td>
<td>6.29</td>
<td>1.18</td>
<td>3 (30.0)</td>
<td>22.0</td>
<td>90</td>
</tr>
<tr>
<td>4096×3072</td>
<td>192</td>
<td>20.84</td>
<td>2.09</td>
<td>4 (30.0)</td>
<td>37.6</td>
<td>50</td>
</tr>
</tbody>
</table>
Numerical example 2: Quintic quantum dot EVP

- Mathematical model: Schrödinger equation for QDs
  \[-\nabla \cdot \left( \frac{\hbar^2}{2m_{\ell}(r, \lambda)} \nabla u \right) + V_{\ell}(r)u = \lambda u\]
  
  where \( \ell = 1 \) (in the dot), 2 (in the matrix)

  - \( \lambda \): eigenvalue, total electron energy
  - \( u \): eigenfunction
  - \( \hbar \): reduced Plank constant
  - \( V_{\ell}(r) \): confinement potential

- Non-parabolic model for effective electron mass:
  \[
  \frac{1}{m_{\ell}(\lambda)} = \frac{P_{\ell}^2}{\hbar^2} \left( \frac{2}{\lambda + g_{\ell} + V_{\ell}} + \frac{1}{\lambda + g_{\ell} - V_{\ell} + \delta_{\ell}} \right) \quad \ell = 1, 2
  \]

  - \( P_{\ell} \): momentum matrix element
  - \( g_{\ell} \): conduction
  - \( \delta_{\ell} \): spin-orbit spin-off band gap

- Interface condition:
  \[
  \left( \frac{\hbar^2}{2m_1(r, \lambda)} \frac{\partial u}{\partial n} \right)_{D_-} = \left( \frac{\hbar^2}{2m_2(r, \lambda)} \frac{\partial u}{\partial n} \right)_{D_+}
  \]

- Homogeneous Dirichlet boundary conditions: \( u = 0 \).
A typical quantum dot is an InAs dot embedded in a GaAs matrix.

Applications: Single-electron transistors (SET); Quantum dot laser; Quantum dot device.

Use finite volume method on a uniform grid for pyramid QD.

After the complicated algebraic manipulations, the discretization can be reduced to quintic polynomial EVP

\[(\lambda^5 A_5 + \lambda^4 A_4 + \lambda^3 A_3 + \lambda^2 A_2 + \lambda^1 A_1 + A_0)u = 0\]
Large scale eigen-computation

- Fine mesh $600 \times 600 \times 450$, (about 161 million unknowns); Coarse meshes: $56 \times 56 \times 42$ for the two-level preconditioners and $12 \times 12 \times 9$ for the initial search space construction.
- Numerical results obtained by running the Yellowstone cluster (72,288 processors) at NCAR.
- Correction equation solvers: Flexible GMRES (with $\text{rtol}=1.0\times10^{-4}$)
  - One-level RAS vs Two-level Cascade preconditioners
### Ground state case

<table>
<thead>
<tr>
<th>np</th>
<th>JD</th>
<th>One-level</th>
<th>Two-level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>FGMRES</td>
<td>Time</td>
</tr>
<tr>
<td>5120</td>
<td>4</td>
<td>185.25</td>
<td>42.48</td>
</tr>
<tr>
<td>7168</td>
<td>4</td>
<td>185.25</td>
<td>29.20</td>
</tr>
<tr>
<td>9216</td>
<td>4</td>
<td>186.00</td>
<td>23.23</td>
</tr>
<tr>
<td>10240</td>
<td>4</td>
<td>186.00</td>
<td>22.46</td>
</tr>
</tbody>
</table>

![Image of heatmaps](image-url)
Concluding remarks

- With geometric information available, in general, the two-level PJD method is 10 times faster than the one-level PJD method.

- PJD is quite scalable for the problems with over 160 millions unknowns on a parallel computers with over 10,000 processors.

- It is worthy to explore its applicability of the two-level PJD algorithm for finding multiple eigenvalues. Then several issues are needed to further study, such as the optimal stopping condition or the more effective coarse PJD solver.

- It is interesting to compare the two-level PJD algorithm with some other popular alternative approach for solving cubic eigenvalue problems, e.g. Shift-and-invert linearization approach with the Krylov-Schur method.