A Parallel Polynomial Jacobi-Davidson Approach for Dissipative Acoustic Eigenvalue Problems

Tsung-Ming Huang[a], Feng-Nan Hwang[b,*], Sheng-Hong Lai[b], Weichung Wang[c], Zih-Hao Wei[b]

[a] *Department of Mathematics, National Taiwan Normal University, Taipei 106, Taiwan*
[b] *Department of Mathematics, National Central University, Jhongli 320, Taiwan*
[c] *Department of Mathematics, National Taiwan University, Taipei 106, Taiwan*

Abstract

We consider a rational algebraic large sparse eigenvalue problem arising in the discretization of the finite element method for the dissipative acoustic model in the pressure formulation. The presence of nonlinearity due to the frequency-dependent impedance poses a challenge in developing an efficient numerical algorithm for solving such eigenvalue problems. In this article, we reformulate the rational eigenvalue problem as a cubic eigenvalue problem and then solve the resulting cubic eigenvalue problem by a parallel restricted additive Schwarz Preconditioned Jacobi-Davidson algorithm (ASPJD). To validate the ASPJD-based eigensolver, we numerically demonstrate the optimal convergence rate of our discretization scheme and show that ASPJD converges successfully to all target eigenvalues. The extraneous root introduced by the problem reformulation does not cause any observed side effect that produces an undesirable oscillatory convergence behavior. By performing intensive numerical experiments, we identify an efficient correction-equation solver, an effective algorithmic parameter setting, and an optimal mesh partitioning. Furthermore, the numerical results suggest that the ASPJD-based eigensolver with an optimal mesh partitioning results in superlinear scalability on a distributed and parallel computing cluster scaling up to 192 processors.

Key words: Acoustic wave equation, cubic eigenvalue problems, Jacobi-Davidson methods, domain decomposition, additive Schwarz preconditioner, parallel computing.

1. Introduction

Acoustics has been widely studied due to a broad range of scientific and engineering applications [14, 15, 18]. One typical example is to achieve optimal designs that reduce noise and vibration in a car, a commercial airplane, or a building. In the computations regarding acoustics, efficient and accurate numerical solutions of large and sparse

*Corresponding author. Tel: +886-3-422-7151 Ext. 65110; Fax +886-3-425-7379
Email addresses: min@math.ntnu.edu.tw (Tsung-Ming Huang), hwangf@math.ncu.edu.tw (Feng-Nan Hwang), ironbox360@gmail.com (Sheng-Hong Lai), wwang@math.ntu.edu.tw (Weichung Wang), socrates.wei@gmail.com (Zih-Hao Wei)

nonlinear algebraic eigenvalue problems (EVP) are often required. In this article, we particularly focus on a rational acoustic EVP with an absorbing wall in the pressure formulation, which was introduced by Bermúdez et al. [2, 3]. The reasons to consider this particular pressure formulation are twofold. First, due to the nonlinearity arising from the frequency-dependent impedance, developing numerical algorithms to solve this nonlinear EVP effectively and efficiently remains a computational challenge. Second, once the primitive variable pressure is determined, other additionally introduced variables, e.g., fluid displacement or potential in the pressure-fluid displacement formulation and pressure-potential formulation [2, 3, 4, 17], can be recovered through postprocessing.

One way to solve the target acoustic EVP is to reformulate the original rational EVP in the pressure formulation as a cubic polynomial EVP and then employ a polynomial Jacobi-Davidson (JD) approach to solve the resulting eigenproblem. The JD algorithm [23] was originally proposed by Sleijpen and Van der Vorst for linear EVPs. Various generalizations of the linear JD have continued to gain their popularity for solving polynomial EVPs with diverse applications. Examples include generalized EVP in the stability analysis of magnetohydrodynamics [20], cubic EVP in the study of instability in the plasma edge of a magnetic fusion device [7], and cubic or quintic EVP in the estimation of discrete energy states and wave functions of a semiconductor quantum dot with non-parabolic band structure [12, 13]. The popularity of the JD algorithm is mainly due to its promising numerical advantages. For example, without recasting the polynomial EVPs as enlarged linearized EVPs, one only needs to deal with a problem the same size as the original one. JD can target the interior spectrum directly without using computationally expensive shift-and-invert techniques. Moreover, the JD algorithm is parallelizable and thus suitable for large-scale eigenvalue computations.

Recently, Hwang et al. proposed the additive Schwarz preconditioned Jacobi-Davidson algorithm (ASPJD), which exports the idea from the parallel Schwarz-Krylov solver for the correction equation to enhance the overall parallel scalability of the JD algorithms [11, 26]. The Schwarz methods [24] are well understood and have been been widely used for solving a variety of linear systems arising from the discretization of partial differential equations (PDEs). They have further been extended to nonlinear systems as a linear preconditioner for the Jacobian system in the Newton-Krylov-Schwarz algorithm and as a nonlinear preconditioner in the additive Schwarz preconditioned inexact Newton algorithm [10]. However, only comparatively few publications are available on addressing eigenvalue problems using Schwarz methods. Among those few, Schwarz methods have been employed as a preconditioner for the Arnoldi method with the spectral transformation [21] or for the locally optimal block preconditioned conjugate gradient method [16].

In this article, we investigate the applicability of ASPJD for solving the acoustic cubic EVP and conduct intensive numerical experiments to study how ASPJD can perform efficiently on parallel computers. It is worth mentioning that (i) the extraneous roots introduced by the reformulation (9) do not lead to oscillatory convergence behavior in ASPJD and (ii) superlinear speedups can be achieved by an optimal mesh partitioning. All these numerical findings suggest that the combination of polynomial eigenvalue problem reformulation and the ASPJD eigensolver forms a promising parallel method for solving the target acoustic eigenvalue problem.

The rest of this paper is organized as follows. Section 2 introduces the acoustic mathematical model, its finite element discretization, and the corresponding cubic EVP.

Section 3 briefly describes the ASPJD. Section 4 examines the parallel performance of the proposed algorithms. Section 5 concludes the paper.

2. Acoustic polynomial eigenvalue problem

Under the time-harmonic assumption, an acoustic EVP is derived from the acoustic wave equation [14, 15], which is a simplified mathematical model of inviscid, compressible, and barotropic fluids with small perturbation. As the model equation is in the pressure formulation, a finite element discretization leads to a rational EVP due to the frequency-dependent impedance. This rational EVP is then converted to the model cubic polynomial EVP by multiplying the common denominators, as shown in the following subsections.

2.1. Dissipative acoustic eigenvalue problem

We consider the following dissipative acoustic EVP [15]:

$$\frac{\lambda^2}{c^2} p = \Delta p, \tag{1}$$

which is defined on a computational domain $\Omega \subset \mathbb{R}^2$ and equipped with three types of boundary conditions:

$$p = 0 \text{ on } \Gamma_O, \tag{2}$$

$$\frac{\partial p}{\partial \mathbf{n}} = 0 \text{ on } \Gamma_R, \text{ and} \tag{3}$$

$$\frac{\partial p}{\partial \mathbf{n}} = -\frac{\lambda \rho}{\mathcal{Z}(\lambda)} p \text{ on } \Gamma_A. \tag{4}$$

Here, (λ, p) forms the complex eigenpair of (1). The imaginary part and the real part of λ represent the angular frequency and the decay rate of sound disturbances, respectively, and p is the pressure perturbation. Other notations include: ρ is the density of fluids, c is the speed of sound, $\mathcal{Z}(\lambda)$ is the frequency-dependent acoustic impedance, and $\mathbf{n}$ is an outward normal vector. Further, the boundaries $\Gamma = \Gamma_O \cup \Gamma_R \cup \Gamma_A$ and Γ_O, Γ_R, and Γ_A stand for the open, reflecting, and absorbing boundary conditions, respectively.

2.2. Finite element discretizations

To discretize (1), we employ the standard Galerkin finite element method on a given quadrilateral mesh $\mathcal{T}^h = \{\mathcal{K}\}$ with a mesh diameter h. Let P^h be a continuous bilinear finite element space for p. We define

$$P^h = \{p \in H^1(\Omega) \cap C^0(\Omega) : p|_K \in P_1(\mathcal{K}), \mathcal{K} \in \mathcal{T}^h\},$$

where $H^1(\Omega)$ is a Hilbert space, $C^0(\Omega)$ is the set of all continuous functions defined on Ω, and $P_1(\mathcal{K})$ is the set of all bilinear functions defined on the element $\mathcal{K}$. Note that

$$P_0^h = \{p \in P^h |\ p = 0 \text{ on } \Gamma_O\}$$

is used for both the trial and test spaces. Next, we want to find an eigenpair (λ_h, p_h), where $\lambda_h \in \mathbb{C}$ and $p_h \in P_0^h$ such that

$$\frac{\lambda_h^2}{c^2}\int_\Omega p_h q\,dx + \frac{\lambda_h \rho}{\mathcal{Z}}\int_{\Gamma_A} p_h q\,ds + \int_\Omega \nabla p_h \cdot \nabla q\,dx = 0 \;\forall\; q \in P^h. \tag{5}$$

Let $\Phi = [\phi_1, \phi_2, ..., \phi_n]^T$ be a vector of global bilinear shape functions, where n is the number of total interior nodes, including all boundary nodes on Γ_A and Γ_R. Then the numerical approximation of the eigenfunction p_h can be expressed as a linear combination of the shape functions

$$p_h = \sum_{i=1}^{n} u_i \phi_i. \tag{6}$$

where $u := [u_1, u_2, ..., u_n]^T$ represents the nodal values of p_h, which is to be determined.

Choosing $q = \phi_i$ (for $i = 1, ..., n$) and substituting (6) into the Galerkin finite element formulation (5), we obtain a PDE-based nonlinear EVP

$$R(\lambda)u = (\lambda^2 M + \frac{\lambda}{\mathcal{Z}}C + K)u = 0.$$

Here, the coefficient matrices

$$M = \frac{1}{c^2}\int_\Omega \Phi\Phi^T\,dx,\; C = \rho\int_{\Gamma_A}\Phi\Phi^T\,dx, \text{ and } K = \int_\Omega \nabla\Phi\cdot\nabla\Phi^T\,dx$$

stand for mass, damping, and stiffness, respectively. In general, M and K are both real symmetric positive definite matrices, and C is a real semi-positive definite matrix containing diagonal blocks. Note that we drop the subscript h of λ throughout the paper to simplify the notation.

2.3. Algebraic eigenvalue problem

Using different mathematical impedance models, we can derive different rational EVP accordingly. In this article, we particularly consider the Kelvin-Voigt model [19]. The impedance $\mathcal{Z}(\lambda)$ for the aborting material can be written as

$$\mathcal{Z}(\lambda) = \frac{\alpha}{\lambda} + \beta,$$

where α is the elastic coefficient, and β is the damping coefficient due to viscous effect. Because

$$\frac{1}{\mathcal{Z}(\lambda)} = \frac{1}{\frac{\alpha}{\lambda}+\beta} = \frac{\lambda}{\alpha+\lambda\beta},$$

we can rewrite the absorbing boundary condition (4) as

$$\frac{\partial p}{\partial \mathbf{n}} = \frac{-\rho\lambda^2}{\alpha+\lambda\beta}\,p \text{ on } \Gamma_A \tag{7}$$

and obtain a nonlinear algebraic large-scale EVP in the rational form

$$\mathcal{R}(\lambda)u = (\lambda^2 M + \frac{\lambda^2}{\alpha + \lambda\beta}C + K)u = 0. \tag{8}$$

We can further rewrite Equation (8) in a cubic polynomial matrix form by multiplying both sides of the equation by $(\alpha + \lambda\beta)I_n$. Here, I_n is an identity matrix of order n. Therefore, we have

$$\mathcal{A}(\lambda)u = (\lambda^3 A_3 + \lambda^2 A_2 + \lambda A_1 + A_0)u = 0, \tag{9}$$

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

Some remarks:

1. By comparing Equations (8) and (9), it is obvious that the later has extraneous roots with algebraic multiplicity n and that those roots are all equal to $\lambda = -\alpha/\beta$.
2. In the case that $\Gamma_O = \emptyset$, a zero eigenvalue exists for both Equations (8) and (9).
3. As suggested in [2] for the quadratic EVP arising in the pressure-displacement formulation, $\mathrm{Re}(\lambda) < 0$ for all eigenvalues where $\lambda \neq 0$, which is consistent with the numerical or physical observation that the acoustic vibrations are damped due to the effect of the viscoelastic materials. We conjecture that the cubic EVP has similar phenomena. This conjecture is numerically confirmed in Figure 2 in Section 4.

3. A parallel polynomial Jacobi-Davidson eigensolver

The JD-type algorithms belong to a class of iterative subspace methods, which consists of two key steps in each iteration: (i) to enlarge a subspace or so-called search space by adding a new basis vector and (ii) to 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 *outer* iteration, it is necessary to solve the so-called correction equations approximately via *inner* iterations. As in other inner-outer algorithms such as the Newton-type method, the bottleneck of JD algorithms is the approximate solution of the correction equations in the inner iterations.

In this section, we briefly describe the ASPJD algorithm, which is used to solve the cubic EVP (9). ASPJD is a Jacobi-Davidson based algorithm that iteratively looks for better approximate eigenpairs in a growing search space. Details of the algorithm can be found in [11]. Here, we simply highlight how the next eigenpair (λ_{new}, u_{new}) can be obtained from the current approximate eigenpair (λ, u).

Suppose (λ, u) has been extracted from the current search space V, and assume it is not close enough to the exact eigenpair (λ^*, u^*). We can find the next approximate eigenpair (λ_{new}, u_{new}) by the following two steps.

Step 1. Expand the search space V by redefining it as $[V, v]$. The vector v is computed as follows. We solve the *correction equation*

$$\left(I - \frac{pu^*}{u^*p}\right)\mathcal{A}(\lambda)(I - uu^*)t = -r \tag{10}$$

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

$$B_d = \left(I - \frac{pu^*}{u^*p}\right) B(I - uu^*) \approx \left(I - \frac{pu^*}{u^*p}\right) \mathcal{A}(\lambda)(I - uu^*), \tag{11}$$

$r = \mathcal{A}(\lambda)u$, $p = \mathcal{A}'(\lambda)u$, and $\mathcal{A}'(\theta) = \sum_{i=1}^{\tau} i\theta^{i-1}A_i$. Furthermore, 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 polynomial EVP $(V^T\mathcal{A}(\theta)V)s = 0$. Then set $\lambda_{new} = \theta$ and compute $u_{new} = Vs$.

In practice, one does not explicitly form B_d defined as (11) to perform the preconditioning operation, $z = B_d^{-1}y$ with $z \perp u$ for a given y. Instead, the equivalent can be done 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 to solve each correction equation, and there is no need to re-compute them in the Krylov subspace iterations. Furthermore, in the ASPJD, the construction of the preconditioner B^{-1} based on an additive Schwarz framework is described as follows.

Let $\{\Omega_i^h, i = 1, ..., N_s\}$ be a non-overlapping subdomain partition with the boundary $\partial\Omega_i^h$. Assume further that the union of these non-overlapping subdomains covers the entire domain Ω and the corresponding mesh $\mathcal{T}^h$. We use $\mathcal{T}_i^h$ to denote the collection of mesh points in Ω_i^h. To obtain *overlapping* subdomains, we expand each subdomain Ω_i^h to a larger subdomain $\Omega_i^{h,\delta}$ with the boundary $\partial\Omega_i^{h,\delta}$. Here, δ is an integer indicating the level of overlap. We assume that neither $\partial\Omega_i^h$ nor $\partial\Omega_i^{h,\delta}$ cut any elements of $\mathcal{T}^h$. Similarly, we use $\mathcal{T}_i^{h,\delta}$ to denote the collection of mesh points in $\Omega_i^{h,\delta}$. Then we define the overlapping subdomain space as $P_i^h = P^h \cap H_0^1(\Omega^{h,\delta})$ and the restriction operator R_i, which transfers data from P^h to P_i^h. In the matrix representation, R_i is an $n_i \times n$ matrix with values of either 0 or 1, where n and n_i are the total numbers of *interior* mesh points in $\mathcal{T}^h$ and $\mathcal{T}_i^{h,\delta}$, respectively, and $\sum_{i=1}^{N_s} n_i \geq n$. 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 preconditioner RAS(δ) with the degree of overlapping δ [5] 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 is considered as a special case of the RAS preconditioner by setting the level of overlap equal to 0.

In the aforementioned Step 2, we compute the eigenpair of the projected EVP, $(V^T\mathcal{A}(\theta)V)s = 0$, by solving the corresponding linearized projected EVP,

$$M_A z = \theta M_B z, \tag{12}$$

where

$$M_A = \begin{bmatrix} 0 & I & 0 \\ 0 & 0 & I \\ M_0 & M_1 & M_2 \end{bmatrix}, M_B = \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & -M_3 \end{bmatrix}, \text{ and } z = \begin{bmatrix} s \\ \theta s \\ \theta^2 s \end{bmatrix},$$

and $M_i = V^T A_i V$. Note that the dimension of $V^T \mathcal{A}(\theta) V$ is usually small, so that a direct eigensolver such as the QZ algorithm, is suitable for this purpose.

We use the following strategy to choose the wanted eigenvalues. Let Θ be the set of the Ritz values computed from (12), and let μ be a given target value that is supposed to be near the desired eigenvalues. Let Λ be a certain set containing the unwanted targets. We select one of the Ritz values, say θ_j, from the set $\Theta \backslash \Lambda$ so that $|\mu - \theta_j| = \min_{\theta \in \Theta \backslash \Lambda} |\mu - \theta|$. Depending on the application, we have different choices of Λ to avoid choosing unwanted Ritz values as the candidate approximate eigenvalues. In particular, in the model acoustic EVP, we define Λ as the extraneous roots, the trivial solutions, or both.

Some practical techniques such as restarting and locking are also included in the AS-PJD eigensolver. In restarting, the eigenpair search is started over, and a new orthogonal V is chosen after a certain number of iterations. The restarting technique can keep the projected EVP (12) to a manageable size and avoid the loss of the numerical orthogonality of basis vectors in the search space. Very often, such a technique can accelerate the convergence of the JD algorithm. In addition, the locking technique is useful when multiple eigenvalues are of interest. By performing locking, the convergent eigenvalues can be included in the set Λ.

4. Numerical results

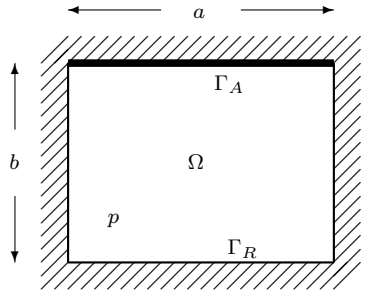

Figure 1: The benchmark problem that models fluid in a cavity with an absorbing wall.

We considered a test case that is often used to study acoustics in a cavity as a benchmark problem to validate the correctness of our parallel eigensolver and to evaluate its parallel performance. In this problem, one wall of the cavity is coated with some sound absorbing material. As shown in Figure 1, a rectangular computational domain is defined on $\Omega = [0, a] \times [0, b]$ with $a > 0$ and $b > 0$. A zero homogeneous Neumann condition is imposed on all boundaries, except for the top boundary, in which the absorbing type

boundary condition (7) is imposed. The analytical solutions for this particular problem can be derived from the technique of separation variables [3], in which (η_m, λ) satisfy

$$\eta_m^2 = \frac{\lambda^2}{c^2} + \frac{m^2\pi^2}{a^2} \text{ and } \eta_m \tanh \eta_m b = \frac{-\rho\lambda^2}{\alpha + \lambda\beta}$$

for a particular $m \in \mathbb{N}$. We obtain (η_m, λ) numerically by Newton's method and refer to them as the semi-analytical solutions. Figure 2 shows a typical spectrum of the benchmark problem .

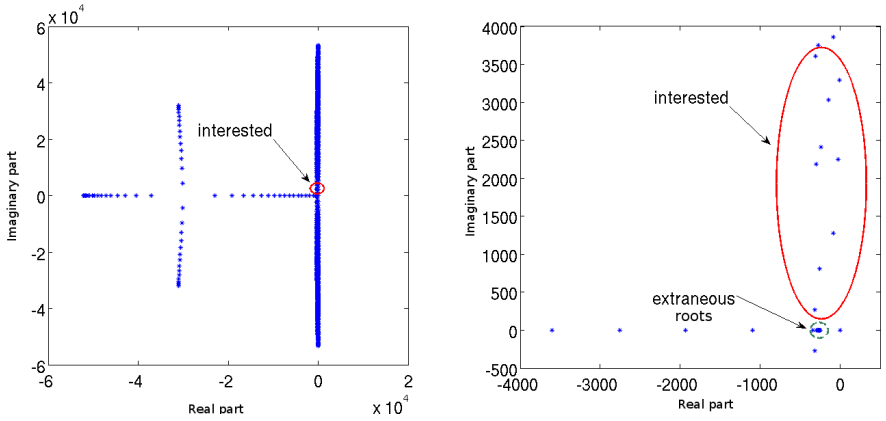

Figure 2: The distribution of the entire spectrum of the benchmark problem with $h = 1/32$ (left). A magnification of the selected spectrum corresponding to the lowest positive vibration frequencies with the highlight of the most interesting eigenvalues highlighted (right).

The setup for the numerical experiments is summarized as follows. Table 1 lists the physical parameters used for the numerical simulation. A sequence of uniformly-refined regular bilinear finite element meshes is employed. We took the ASPJD eigensolver implemented on top of the Portable, Extensible Toolkit for Scientific Computation (PETSc) [1] and the Scalable Library for Eigenvalue Problem Computation (SLEPs) [6] in [11] as the parallel polynomial JD eigensolver for the target problem. Aiming at the target acoustic problem, we conducted intensive numerical experiments to validate the ASPJD eigensolver (Section 4.1), to compare the efficiency of three possible correction-equation solvers (Section 4.2), to tune the algorithmic parameters (Section 4.3), and to investigate the scalability (Section 4.4).

Parameter	Symbol	Value	Unit	Parameter	Symbol	Values	Unit
domain width	a	1.00	m	domain height	b	0.75	m
fluid density	ρ	1.00	kg/m^3	sound speed	c	340.00	m/s
elasticity	α	50000.00	N/m^3	damping	β	200.00	Ns/m^3

Table 1: Physical parameters used for the numerical experiments.

The ASPJD eigensolver is declared to be convergent, if $\|r\|_2 < atol$ or $\|r\|_2 < rtol\|r_0\|_2$, where $rtol$ and $atol$ are set to be 10^{-10} and 10^{-8}, respectively. The eigenvalue solver restarts every 30 steps and selects 5 original best vectors with the smallest magnitudes to be retained in the search space during the restarting procedure. We selected the initial search space to be $V = [g]_{n\times 1}$, where $g = normalize(v + normalize(s))$, as a constant vector is a known trivial solution for $\Gamma_O = \emptyset$. Here, the normalized constant, $v = (\frac{1}{n}, \frac{1}{n}, ..., \frac{1}{n})$, with a small random perturbation s generated from a random variable in $(0, 1]$. For the acoustics problem in industrial and scientific applications, the eigenvalues of interest are often interior ones, which correspond to low frequency modes within the range of $0 < \frac{\mathrm{Im}(\lambda)}{2\pi} < 600$Hz. Hence, the target eigenvalue for the ASPJD eigensolver was set to be $0 + 1900i$. All computations were performed in complex arithmetic double precision and were run on a cluster of computers with the dual Intel Woodcrest 3.0G CPUs with dual cores communicating via Infiniband switch. Various numerical results and our findings will be reported and discussed in the following subsections.

4.1. Parallel eigensolver validation

To validate the parallel ASPJD eigensolver, we computed 10 eigenvalues that were close to the target for different mesh sizes by using 48 processors. We employed a fixed 20 Generalized Minimal Residual Method (GMRES) [22] iterations as the stopping criterion while solving the correction equation. The GMRES solver was incorporated with a left RAS(1) preconditioner, where the subdomain problems were solved by a direct incomplete LU decomposition with three-level of fill-ins. Careful algorithmic parameter tuning for achieving the optimal performance will be presented in Sections 4.3 and 4.4.

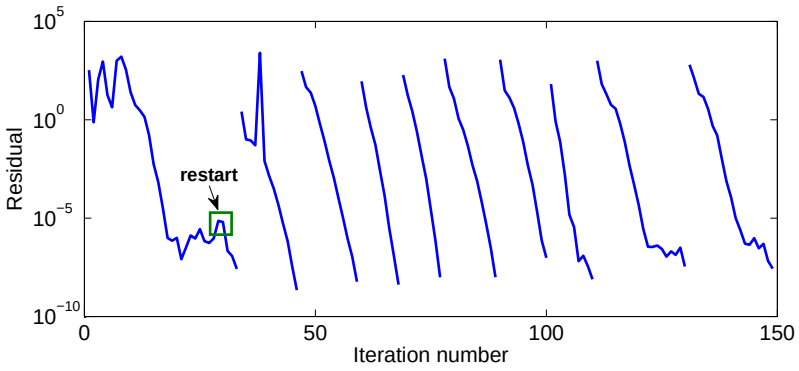

Figure 3: History of residuals for computing 10 eigenvalues for $h = 1/256$.

Figure 3 shows the history of the residuals for computing 10 eigenvalues for the case of $h = 1/256$. The results show that the ASPJD eigensolver converged successfully to all target eigenvalues. The figure also suggests that ASPJD took many more iterations to converge to the first eigenvalue, due mainly to the randomness of the initial basis in the search space. The residual oscillated at the beginning and then was gradually reduced. The results indicate that the restarting technique can save memory usage and improve the quality of the basis in the search space so that the ASPJD converges

rapidly after restarting. In addition, Figure 3 shows that the residuals decreased almost in a monotone manner, except for the ones corresponding to the first eigenvalue. This observation suggests that the extraneous roots introduced by reformulating (8) as (9) do not cause oscillatory convergence behavior as reported in [8]. This is another justification for the proposed numerical approach.

The accuracy results reported in Table 2 indicate that the computed eigenvalues and semi-analytical eigenvalues matched each other quite well. All of the relative differences were less than 10^{-6}. Additionally, the convergence behavior of our numerical discretization scheme shown in Tables 2 and 3 closely followed the theoretical predictions available in the literature. An error estimate [25] for the standard elliptic EVP obtained by linear finite elements is given as

$$\lambda_i \leq \lambda_i^{(h)} \leq \lambda_i + \mathbf{C}h^2\lambda_i^2,$$

where $\mathbf{C}$ is a constant, and λ_i $(0 < \lambda_1 \leq \lambda_2 \leq \lambda_3 \leq ...)$ and $\lambda_i^{(h)}$ are the ith exact eigenvalue and the ith computed eigenvalues, respectively. We highlight some observations from these two tables.

- The optimal quadratic rate of convergence for the 10 computed eigenvalues was achieved.
- The quality of the approximate eigenvalues for the high frequency modes was degraded, as the error bound became wider for higher frequency modes.
- For a fixed i, the computed eigenvalue $|\lambda_i^{(h)}|$ was bounded from below by $|\lambda_i|$ and the sequence $|\lambda_i^{(h)}|$ was decreasing as the mesh was refined, i.e., $|\lambda_i| \leq \ldots \leq |\lambda_i^{(\frac{h}{4})}| \leq |\lambda_i^{(\frac{h}{2})}| \leq |\lambda_i^{(h)}|$.

i	$\lambda_i^{(h)}$ (by ASPJD)	λ_i (semi-analytical)	$\lvert\lambda_i^{(h)} - \lambda_i\rvert$	$\frac{\lvert\lambda_i^{(h)} - \lambda_i\rvert}{\lvert\lambda_i\rvert}$	k
1	-320.70879 + 267.64787i	-320.70845 + 267.64791i	3.43234e-04	8.21687e-07	2.00
2	-259.20827 + 813.28676i	-259.20818 + 813.28664i	1.49098e-04	1.74670e-07	2.00
3	-89.95380 + 1281.34546i	-89.95380 + 1281.34506i	4.04794e-04	3.15138e-07	2.00
4	-297.21018 + 2181.14878i	-297.20938 + 2181.14545i	3.42330e-03	1.55512e-06	2.00
5	-27.36523 + 2250.40916i	-27.36529 + 2250.40594i	3.22123e-03	1.43130e-06	2.00
6	-236.70537 + 2409.20862i	-236.70459 + 2409.20544i	3.26946e-03	1.35057e-06	2.00
7	-143.16372 + 3023.68912i	-143.16330 + 3023.68443i	4.71118e-03	1.55635e-06	2.00
8	-12.69358 + 3282.07988i	-12.69366 + 3282.06881i	1.10723e-02	3.37354e-06	2.00
9	-302.60555 + 3588.44995i	-302.60332 + 3588.43448i	1.56315e-02	4.34068e-06	2.00
10	-275.41285 + 3737.82898i	-275.41058 + 3737.81401i	1.51402e-02	4.03961e-06	2.00

Table 2: A comparison of the computed eigenvalues obtained by the ASPJD eigensolver with the semi-analytical solutions that $h = 1/1024$. The convergence rate $k = \log_2 \frac{|\lambda_i^{(h=1/512)} - \lambda_i|}{|\lambda_i^{(h=1/1024)} - \lambda_i|}$.

4.2. A comparison of three correction-equation solvers

The numerical solution of the correction equation is a kernel of the ASPJD (and other JD-type algorithms), as it is the most expensive step in the algorithm and solving that

i	$\lvert\lambda_i^{(\frac{1}{64})}\rvert$	$\lvert\lambda_i^{(\frac{1}{128})}\rvert$	$\lvert\lambda_i^{(\frac{1}{256})}\rvert$	$\lvert\lambda_i^{(\frac{1}{512})}\rvert$	$\lvert\lambda_i^{(\frac{1}{1024})}\rvert$	$\lvert\lambda_i\rvert$
1	417.78005	417.73440	417.72299	417.72013	417.71942	417.71918
2	853.63067	853.60375	853.59702	853.59533	853.59491	853.59477
3	1284.60203	1284.52451	1284.50513	1284.50029	1284.49908	1284.49867
4	2202.17357	2201.51959	2201.35612	2201.31526	2201.30504	2201.30163
5	2251.39666	2250.77839	2250.62384	2250.58520	2250.57554	2250.57232
6	2421.63431	2421.01277	2420.85742	2420.81858	2420.80887	2420.80563
7	3028.27694	3027.37299	3027.14704	3027.09055	3027.07643	3027.07173
8	3284.92850	3282.80199	3282.27050	3282.13764	3282.10442	3282.09335
9	3605.16700	3602.16952	3601.42043	3601.23317	3601.18635	3601.17075
10	3751.81292	3748.91295	3748.18825	3748.00710	3747.96181	3747.94671

Table 3: The computed eigenvalues for different mesh sizes and the semi-analytical solutions.

equation efficiently is often a significant factor in the overall performance. We compared three correction equation solvers (10). Table 4 demonstrates the comparison results for a RAS iterative method (RAS), GMRES with RAS preconditioning (GMRES+RAS) (the one employed in ASPJD), and a fixed-step GMRES without preconditioning (GMRES-only). For the purpose of comparison, the maximum of number of iterations for all methods was set to be 25, the degree of RAS overlapping was set to be 1, and the incomplete LU with 0, 1, 2, or 3 levels of fill-ins was used as a subdomain solver. The test problem of mesh size $h = 1/512$ is considered and 48 processors were used.

We found that although the computational costs per outer iteration in JD iteration of both the GMRES-only method and the RAS method for solving the correction equation were much cheaper than the cost in GMRES with RAS, the GMRES+RAS method always performed the best. Note that the correction-equation formulation in the form of Equation (10) is the most efficient compared to the other two available formulations (not shown here); see [9].

	RAS		GMRES+RAS		GMRES-only	
Subdomain solvers	JD ite	Time (s)	JD ite	Time (s)	JD ite	Time (s)
ILU(0)	301	25.0	52	5.6	256	18.3
ILU(1)	156	15.0	34	4.3		
ILU(2)	108	12.1	30	4.3		
ILU(3)	88	11.4	30	4.6		
LU	68	18.9	33	9.4		

Table 4: An efficiency comparison of the JD-based algorithm on three types of correction-equation solver: a RAS iterative method, fixed 25-step RAS preconditioned GMRES, and GMRES without preconditioning.

4.3. ASPJD algorithmic parameter tuning

To achieve optimal parallel performance, several algorithmic parameters involved in the ASPJD algorithm needed to be well tuned. Here, we focus on how some factors

related to the Krylov-Schwarz solver for the correction equation affect the overall convergence. These factors include the maximum number of GMRES iterations allowed, denoted by GMRES(k) for $k = 20,\ 25,\ 30$, the degree of overlapping for restricted additive Schwarz preconditioners, denoted by RAS(δ) for $\delta = 0,\ 1,\ 2,\ 3$, and the quality of subdomain solution, where a direct sparse LU decomposition or an incomplete sparse LU decomposition with different levels of fill-ins ILU(l) for $l = 0,\ 1,\ 2,\ 3$ was employed.

We summarize the all possible test runs for the case of $h = 1/512$ with the number of processors $np = 12$ and $np = 48$ in Tables 5 and 6, respectively. Observing the values from these two tables, we found that in general, the number of the outer iterations in ASPJD was typically reduced when we enhanced the solution quality of the correction equation by increasing the levels of ILU fill-ins, the degrees of RAS overlapping, or the number of GMRES iterations. However, in practice, ILU(0) and LU, which are either too inaccurate or too expensive, are not recommended for use as a subdomain solver. For the remaining cases, if mild levels of ILU fill-ins and mild degrees of RAS overlapping were chosen, the number of outer iterations in ASPJD was nearly independent of the number of processors.

		RAS(0)		RAS(1)		RAS(2)		RAS(3)	
		JD ite	Time	JD ite	Time	JD ite	Time	JD ite	Time
GMRES(20)	ILU(0)	79	40.32	69	37.93	65	35.78	64	35.71
	ILU(1)	49	27.18	44	25.96	40	23.68	38	22.94
	ILU(2)	40	23.96	30	19.36	30	19.53	30	19.76
	ILU(3)	35	22.74	30	21.01	30	21.28	30	21.77
	LU	37	47.77	33	45.13	23	31.75	31	43.76
GMRES(25)	ILU(0)	55	35.92	51	35.28	48	33.74	48	33.44
	ILU(1)	37	25.99	30	22.69	30	22.89	30	22.95
	ILU(2)	34	26.15	30	24.40	30	24.70	30	24.86
	ILU(3)	30	24.76	30	26.58	30	26.75	33	29.59
	LU	35	54.56	32	52.17	31	51.31	38	53.32
GMRES(30)	ILU(0)	47	37.88	43	36.78	43	37.01	42	36.45
	ILU(1)	30	26.30	30	27.97	30	28.08	30	28.44
	ILU(2)	30	28.30	30	30.05	30	30.39	30	30.56
	ILU(3)	30	30.47	32	34.44	32	34.76	33	36.19
	LU	32	58.32	31	59.16	33	63.86	20	39.07

Table 5: Parametric study for $h = 1/512$ and the number of processors $np = 12$.

4.4. Parallel scalability analysis

We investigated how mesh partitioning affects the overall parallel scalability performance of the ASPJD algorithm on a parallel computing cluster. Mesh partitioning is a way to map the mesh data into processors on a distributed-memory computer. The main goal of mesh partitioning is to balance computational loading and to minimize point-to-point processor communications. For the case of structured meshes we are currently considering, it can be done by using the DA object in PETSc internally. As shown in Figure 4, we considered two types of mesh partitioning: partitioning in both the $x-$ and y-directions (Partition I) and partitioning in the x-direction only (Partition II). In addition, we took the strong scalability, known as fixed-problem-size scalability, as a

		RAS(0)		RAS(1)		RAS(2)		RAS(3)	
		JD ite	Time	JD ite	Time	JD ite	Time	JD ite	Time
GMRES(20)	ILU(0)	83	6.45	70	6.17	66	6.05	65	6.21
	ILU(1)	53	4.58	45	4.49	39	4.18	38	4.10
	ILU(2)	46	4.40	34	3.86	30	3.61	30	3.76
	ILU(3)	42	4.49	30	3.84	30	3.91	33	4.47
	LU	35	7.90	32	7.79	33	8.54	34	9.11
GMRES(25)	ILU(0)	53	4.58	52	5.64	50	5.58	49	5.70
	ILU(1)	43	4.41	34	4.34	30	3.85	30	3.99
	ILU(2)	34	4.07	30	4.32	33	4.68	30	4.57
	ILU(3)	31	4.11	30	4.64	32	5.16	30	5.10
	LU	33	8.54	33	9.39	33	9.90	31	9.82
GMRES(30)	ILU(0)	51	5.65	45	5.80	45	5.93	42	5.73
	ILU(1)	34	4.31	30	4.54	30	4.64	30	4.81
	ILU(2)	31	4.45	30	4.89	30	5.19	29	5.27
	ILU(3)	32	5.07	31	5.75	30	5.99	33	6.54
	LU	34	10.30	32	10.85	31	11.37	31	11.63

Table 6: Parametric study for $h = 1/512$ and the number of processors $np = 48$.

measurement for evaluating the parallel performance of our ASPJD eigensolver. To conduct the numerical experiments, we employed GMRES(30) preconditioned by RAS(1), where each subdomain problem is solved by ILU(3), as the correction equation solver.

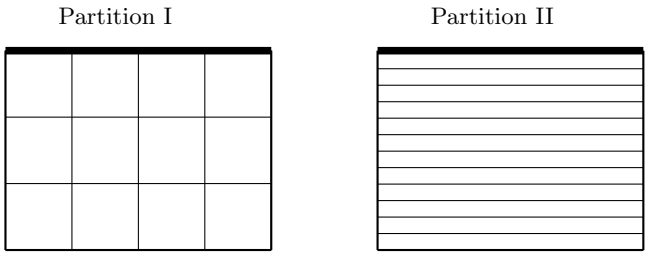

Figure 4: Two types of mesh partitioning: 4×3 (left) and 1×12 (right)

Table 7 shows the performance summary for computing both the first wanted eigenvalue and the ten wanted eigenvalues of the problem with about $800,000$ unknowns. This table suggests the following findings. (i) The influence of mesh partitioning on the scalability of the algorithm increases with the number of processors. (ii) The number of outer iterations in ASPJD remains almost constant for Partition I, but increases significantly for Partition II. (iii) ASPJD based on Partition I achieves, remarkably, a superlinear (or an excellent parallel efficiency) on up to 192 processors. Note that Partition I keeps the number of interfacial nodes as small as possible so that the communications between processors can be minimized. Also, that partition allows the solution data to be propagated in two directions (instead of in only one direction for Partition II) simultaneously, so that data exchange can be completed rapidly.

Partition Type	$n_x \times n_y$	Results for the first eigenvalue					Results for the ten eigenvalues			
		np	JD ite	Time (s)	Sp	Ef (%)	JD ite	Time (s)	Sp	Ef (%)
I	4×3	12	30	157.7	1.0	100.0	233	1031.1	1.0	100.0
	8×6	48	30	33.2	4.8	118.8	266	224.2	4.1	102.9
	16×12	192	38	7.8	20.2	126.5	263	47.8	21.0	131.0
II	1×12	12	43	158.7	1.0	100.0	188	1019.3	1.0	100.0
	1×48	48	54	51.4	3.6	89.9	244	276.1	3.7	92.3
	1×192	192	96	25.5	6.9	43.1	493	129.4	7.8	49.0

Table 7: Parallel scalability for computing the first eigenvalue and the first ten desired eigenvalues using different mesh partitioning schemes for the case of $h = 1/1024$. The mesh partitioning scheme is classified by n_x and n_y, which are the number of process partitions in the x- and y-direction, respectively. The total number of processors is $np = n_x n_y$. Sp and Ef stands for speedup and parallel efficiency, respectively.

5. Conclusions

This work is the first attempt, to the best of the authors' knowledge, to solve the acoustic EVP in the pressure formulation by reformulating it as a cubic polynomial eigenvalue problem. The resulting problem has been successfully and efficiently solved by the ASPJD algorithm. The optimal convergence rate of linear finite element discretizations for the computed eigenvalues was numerically observed and its convergence behavior is similar to that predicted theoretically for the standard elliptic EVP [25]. The convergence analysis of finite elements for the nonlinear acoustic problem, of course, is worth further investigation. We have also identified the most efficient correction equation solver, i.e., GMRES with restricted additive Schwarz preconditioning, among the other options. Algorithmic parameter-tuning studies have led to the suggested parameter choices that are able to achieve an excellent strong scalability on a cluster of parallel computers, scaling up to 192 processors. We should emphasize that the proposed polynomial JD approach is quite general; hence, it has potential applications for acoustic EVPs in different formulations. For example, in the pressure-displacement formulation of an acoustic wave [2, 3], a quadratic EVP needs to be solved. As the problem size would be triple for two-dimensional cases and four times for three-dimensional cases compared to the pressure formulation, and the size of the resulting quadratic EVP is even larger, classical linearization methods are not suitable for large-scale eigen-computation, as these methods usually introduce numerical difficulties in efficiency and robustness. On the other hand, based on our numerical experiments, the ASPJD-based polynomial eigensolver is shown to be a promising alternative for solving such EVPs efficiently, and it is interesting to explore its possible applicability to more complex acoustic-structure EVPs [18].

Acknowledgements

The authors are grateful to the anonymous reviewers for their valuable comments, to So-Hsiang Chou for helpful discussions, and to the Computer and Information Networking Center, National Taiwan University for providing high-performance computing

resource. 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, . C. McInnes, B. F. Smith, and H. Zhang. PETSc webpage, 2010. http://www.mcs.anl.gov/petsc.

[2] A. Bermúdez, R. G. Durán, and J. Rodríguez, R. Solomin. Finite element analysis of a quadratic eigenvalue problem arising in dissipative acoustics. *SIAM J. Numer. Anal.*, 38:267–291, 2000.

[3] A. Bermúdez and R. Rodríguez. Modelling and numerical solution of elastoacoustic vibrations with interface damping. *Int. J. Numer. Meth. Engng.*, 46:1763–1779, 1999.

[4] A. Bermúdez and R. Rodríguez. Analysis of a finite element method for pressure/potential formulation of elastoacoustic spectral problem. *Math. Comp.*, 71:537–552, 2002.

[5] X.-C. Cai and M. Sarkis. A restricted additive Schwarz preconditioner for general sparse linear systems. *SIAM J. Sci. Comput*, 21:792–797, 1999.

[6] V. Hernandez, J.E. Roman, A. Tomas, and V. Vidal. SLEPc webpage, 2010. http://www.grycap.upv.es/slepc.

[7] M Hochbruck and D. Löchel. A multilevel Jacobi-Davidson method for polynomial PDE eigenvalue problems arising in plasma physics. *SIAM J. Sci. Comput.*, 32:3151–3169, 2010.

[8] T.-M. Huang, W.-J. Chang, Y.-L. Huang, W.-W. Lin, W.-C. Wang, and W. Wang. Preconditioning bandgap eigenvalue problems in three dimensional photonic crystals simulations. *J. Comput. Phys.*, 229:8684–8703, 2010.

[9] T.-M. Huang, W. Wang, and C.-T. Lee. An efficiency study of polynomial eigenvalue problem solvers for quantum dot simulations. *Taiwanese J. Math.*, 14:999–1021, 2010.

[10] 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.

[11] 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.

[12] 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.

[13] T.-M. Hwang, W.-W. Lin, W.-C. Wang, and W. Wang. Numerical simulation of three dimensional pyramid quantum dot. *J. Comput. Phys.*, 196:208–232, 2004.

[14] F Ihlenburg. *Finite Element Analysis of Acoustic Scattering*. Springer, 1998.

[15] L.E. Kinsler, A.R. Frey, A.B. Coppens, and Sanders J.V. *Fundamentals of Acoustics*. John Wiley & Sons, 2000.

[16] A.V. Knyazev. Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM J. Sci. Comput*, 23:517–541, 2001.

[17] W. Larbi, J.F. Deü, and R. Ohayon. A new finite element formulation for internal acoustic problems with dissipative walls. *Int. J. Numer. Meth. Engng.*, 68:381–399, 2006.

[18] S Marburg. Developments in structural-acoustic optimization for passive noise control. *Arch. Comput. Meth. Engng.*, 9:292–370, 2002.

[19] M.A. Meyers and K.K. Chawla. *Mechanical Behavior of Materials*. Cambridge Univ Press, 2009.

[20] 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.

[21] 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.

[22] Y. Saad and M.H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Comput.*, 7(3):856–869, 1986.

[23] 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):401–425, 1996.

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

[25] G. Strang and G.J. Fix. *An Analysis of the Finite Element Method*. Prentice-Hall, 1973.

[26] Z.-H. Wei, F.-N. Hwang, T.-M. Huang, and W. Wang. A parallel scalable PETSc-based Jacobi-Davidson polynomial eigensolver with application in quantum dot simulation. *Lect. Notes Comput Sci Eng*, 78:157–164, 2011.