Spectral Schur complement techniques for symmetric eigenvalue problems

Vassilis Kalantzis

Computer Science and Engineering Department
University of Minnesota - Twin Cities, USA

Oral Preliminary Exam, 05-27-2016

Acknowledgments

- To my advisor Prof. Yousef Saad
- My deepest gratitudes to the exam committee members
- Special thanks to Prof. Bernardo Cockburn for our discussions regarding the presented work
- This work is partly based on computations performed at the Minnesota Supercomputing Institute
- Work supported by NSF and DOE (DE-SC0008877)

UNIVERSITY OF MINNESOTA

Supercomputing Institute

Contents

Introduction

The symmetric eigenvalue problem

$$Ax^{(i)} = \lambda_i x^{(i)}, \quad i = 1, \ldots, n,$$

where $A \in \mathbb{R}^{n\times n}$ is symmetric, $x^{(i)} \in \mathbb{R}^n$ and $\lambda_i \in \mathbb{R}$. A pair $(\lambda_i, x^{(i)})$ is an *eigenpair* of A.

Focus

1. Compute, up to a tolerance `tol`, all eigenpairs $(\lambda_i, x^{(i)})$ located inside the interval $[\alpha, \beta]$ where $\alpha,\ \beta \in \mathbb{R}$ and $\lambda_1 \le \alpha,\ \beta \le \lambda_n$.
2. Given a shift $\zeta \in \mathbb{R}$, find the k (λ, x) pairs closest to ζ.

How to compute a few eigenpairs of A?

1. Typically: Lanczos on A or $\rho(A - \sigma I),\ \sigma \in \mathbb{C}$
2. In this talk we consider Schur complement eigenvalue solvers

Schur complement eigenvalue solvers (I)

Block matrix representation

Consider matrix A written as

$$A = \begin{pmatrix} B & E \\ E^\top & C \end{pmatrix}$$

where $B \in \mathbb{R}^{d \times d},\ E \in \mathbb{R}^{d \times s}$ and $C \in \mathbb{R}^{s \times s},\ n = d + s$.

Let $x = [u^T, y^T]^T,\ u \in \mathbb{R}^d,\ y \in \mathbb{R}^s$, assume $\lambda \notin \Lambda(B)$

Solution of $(A - \lambda I)x = 0$ is transformed to:

1. Solve $(B - \lambda I)u = -Ey$
2. Solve $S(\lambda)y = 0$

where

$$S(\lambda) = C - \lambda I - E^T (B - \lambda I)^{-1} E,$$

is the spectral Schur complement matrix

Schur complement eigenvalue solvers (II)

Some references on spectral Schur complements

- Component Mode Synthesis and Automated Multi-Level Substructuring (AMLS) [BeLe] for the analysis of frequency response
  - Substructuring techniques (domain decomposition)
  - Approximate y by linearizing $S(\lambda) \rightarrow$ Generalized eig. problem with pencil $(S(0), S'(0))$
  - Approximate u by approximating $-(B - \lambda I)^{-1} Ey$
  - Perform a Rayleigh-Ritz projection
  - AMLS is a multilevel extension of CMS (combined with mode truncation)
  - Typically: fast computation but modest accuracy
- Other work: Abramov and Chishov (spectral Schur complements), Lui (Kron's method),...

Not Schur complement based, but of similar nature

Hybridized Raviart-Thomas approximation of 2nd order elliptic eigenvalue problems [Coc]

- The original eigenvalue problem is condensed by hybridization
- Result: transformation to a non-linear, but smaller eigenvalue problem
- Lower energy modes can be recovered accurately
- Solving the non-linear eigenvalue problem:
  - Fist solve a perturbed (linearized) version to obtain accurate approximations
  - Exploit Newton's method (or RQI) to refine each approximate eigenvalue

HDG approximation of 2nd order elliptic eigenvalue problems [Gop]

- Similar framework with the Raviart-Thomas approximation
- Covers a narrower effective spectrum but can be benefited by postprocessing

Our contribution

In this talk we concentrate on Schur complement eigenvalue solvers from a domain decomposition viewpoint and contribute:

1. A numerical scheme based on Newton root-finding procedures
2. An extension of current understanding regarding Schur complement eigenvalue solvers
3. An implementation of the proposed scheme in distributed computing environments

Contents

Partitioning of the domain (Metis, Scotch,...)

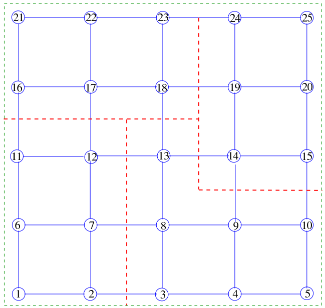


Figure: An edge-separator

- Goal: Partition a discretized domain in p subdomains
- Two main approaches to partition a graph:
  - Edge-separator (vertex-based partitioning)
  - Vertex-separator (edge-based partitioning)
- After partitioning, three types of unknowns:
  - interior unknowns
  - local interface unknowns
  - external interface unknowns
- The i'th subdomain has d_i interior unknowns and s_i local interface unknowns

The local viewpoint in each subdomain

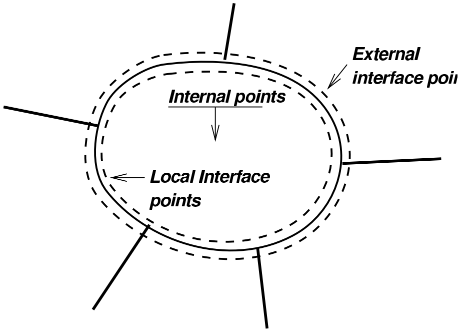


Figure: Local viewpoint

Equation $(A - \lambda I)x = 0$ can be written locally as

$$\underbrace{\begin{pmatrix} B_i - \lambda I & \hat{E}_i \\ \hat{E}_i^T & C_i - \lambda I \end{pmatrix}}_{A_i} \underbrace{\begin{pmatrix} u_i \\ y_i \end{pmatrix}}_{x_i} + \begin{pmatrix} 0 \\ \sum_{j \in N_i} E_{ij} y_j \end{pmatrix} = 0.$$

- $B_i \in \mathbb{R}^{d_i \times d_i}$: coupling between interior variables
- $\hat{E}_i \in \mathbb{R}^{d_i \times s_i}$: coupling between interior/loc. interface variables
- $C_i \in \mathbb{R}^{s_i \times s_i}$: coupling between local interface variables
- N_i: set of indices for subdomains that are neighbors to the i'th subdomain
- $E_{ij} \in \mathbb{R}^{s_i \times s_j}$: coupling among subdomains i and j

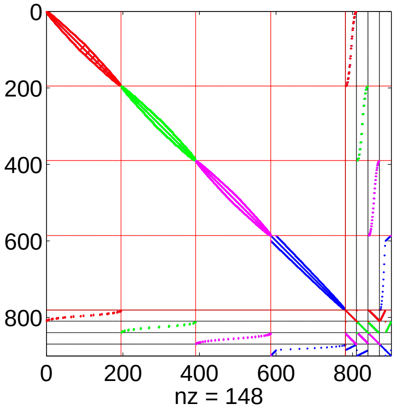
Global reordering of the eigenvalue problem

Stack interior variables $u_1, u_2, \ldots, u_p$ into u, then interface variables $y_1, y_2, \ldots, y_p$ to y, and reorder A so that interior variables are numbered before interface ones:

$$\begin{pmatrix} B_1 & & & & E_1 \\ & B_2 & & & E_2 \\ & & \ddots & & \vdots \\ & & & B_p & E_p \\ E_1^\top & E_2^\top & \cdots & E_p^\top & C \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_p \\ y \end{pmatrix} = \lambda \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_p \\ y \end{pmatrix}$$

where $E_i = [0_{d_i,\ell_i}, \hat{E}_i, 0_{d_i,\nu_i}]$, with $\ell_i = \sum_{j=1}^{j<i} s_j$, $\nu_i = \sum_{j>i}^{j=p} s_j$.

Example with $p = 4$ (different color $\rightarrow$ different subdomain):



Notation: write as

$$A = \begin{pmatrix} B & E \\ E^\top & C \end{pmatrix}$$

The spectral Schur complement

- Eliminating the u_i's we get

$$S(\lambda)y = \begin{pmatrix} S_1(\lambda) & E_{12} & \cdots & E_{1p} \\ E_{21} & S_2(\lambda) & \cdots & E_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ E_{p1} & E_{p2} & \cdots & S_p(\lambda) \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \end{pmatrix} = 0$$

with $S_i(\lambda) = C_i - \lambda I - E_i^\top (B_i - \lambda I)^{-1} E_i$.

- Interface problem (non-linear):

$$S(\lambda)y = 0$$

- Top parts can be recovered as $u_i = -(B_i - \lambda I)^{-1} E_i y$

Contents

Spectral Schur complement revisited

Our viewpoint:

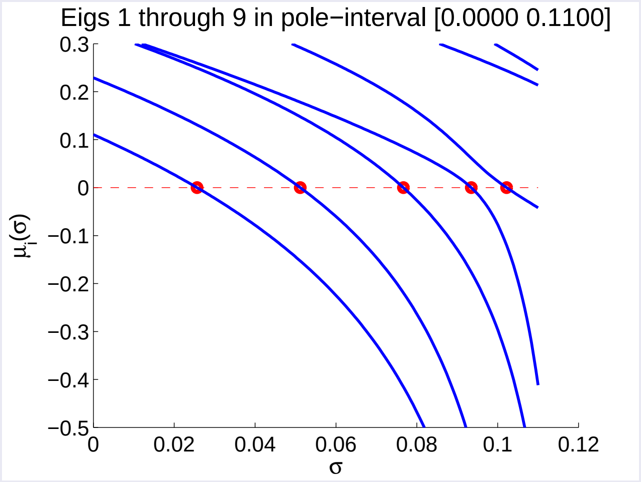
- Find $\sigma \in \mathbb{R}$ such that
$$\mu(\sigma) = 0,$$
where $\mu(\sigma)$ denotes the smallest $(|.|)$ eigenvalue of $S(\sigma)$.
- We can treat $\mu(\sigma)$ as a function $\rightarrow$ root-finding problem
- Bisection: slow. Newton: fast, but needs the derivative
- Other functions are possible, e.g. $\mu(\sigma) = \det[S(\sigma)]$ (Lui) $\rightarrow$ not very practical

Eigenbranches

- For each σ: $S(\sigma)y^{(i)}(\sigma) = \mu_i(\sigma)y^{(i)}(\sigma)$, $i = 1, \ldots, s$ (in sorted ascending algebraic order)
- Each $\mu_i(\sigma)$ is a function of σ (we call it an *eigenbranch*)
- The eigenvalues of B are the poles of the eigenbranches

Eigenbranches – illustration

An example for a discretized 2D Laplacian



Basic algorithm - Newton's scheme

Derivative of $\mu_i(\sigma)$

- Eigenbranch $\mu_i(\sigma)$ is analytic for any $\sigma \notin \Lambda(B)$ with

$$\frac{d\mu_i(\sigma)}{d\sigma} = \frac{(S'(\sigma)y^{(i)}(\sigma), y^{(i)}(\sigma))}{(y^{(i)}(\sigma), y^{(i)}(\sigma))} = -1 - \frac{\|(B-\sigma I)^{-1}Ey^{(i)}(\sigma)\|_2^2}{\|y^{(i)}(\sigma)\|_2^2}.$$

where

$$S'(\sigma) = -I - E^\top (B-\sigma I)^{-2}E.$$

Algorithm 3.1

```
1: Select $\sigma$, $\mathrm{tol}$
2: **repeat**
3:     Compute $\mu(\sigma)$ = Smallest eigenvalue in modulus of $S(\sigma)$
4:     along with associated unit eigenvector $y(\sigma)$
5:     Set $\eta := \|(B-\sigma I)^{-1}Ey(\sigma)\|_2$, and $\sigma := \sigma + \mu(\sigma)/(1+\eta^2)$
6: **until** $|\mu(\sigma)| \le \mathrm{tol}$
```

“Chasing” an eigenbranch using Newton’s iteration

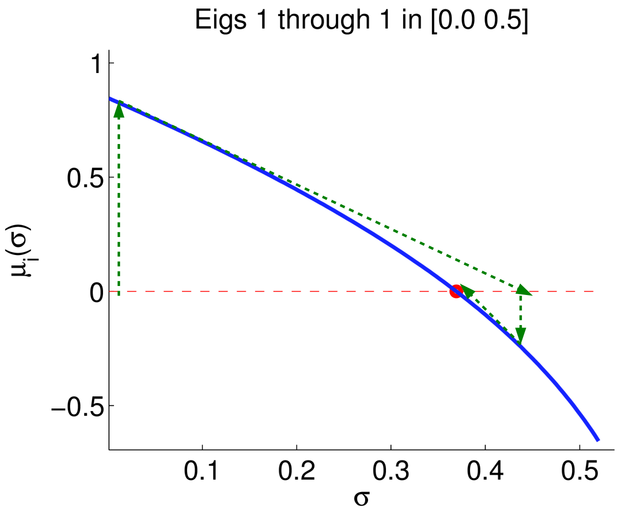


Figure: Computing the algebraically smallest eigenvalue of a 2D Laplacian

Practical details

Computing $(\mu(\sigma), y(\sigma))$ – the main computational kernel!

- Perfect setting for a form of (inexact) inverse iteration (MINRES, GMRES)
- Alternatively, the Lanczos algorithm can be used
- Note that $\mu'(\sigma)$ is upper-bounded by -1 and its computation is entirely local across the subdomains

Residual of the approximation + connection with RQ

It can be shown that

$$\|(A - \sigma I)\hat{x}(\sigma)\| = |\mu(\sigma)|$$

where $\hat{x}(\sigma) = [-(B - \sigma I)^{-1} E y(\sigma); y(\sigma)]$ is the approximate eigenvector, and the Newton update also is the Rayleigh quotient,

$$\sigma = \rho(A, \hat{x}(\sigma)).$$

Eigenbranches across the poles (I)

Let $(\theta_j, v_j),\ j = 1, \ldots, d$ be the eigenpairs of B. We re-write:

$S(\sigma) = C - \sigma I - E^T(B - \sigma I)E = C - \sigma I - \sum_{j=1}^{d} \frac{w_j w_j^T}{\theta_j - \sigma}$, with $w_j \equiv E^T v_j$.

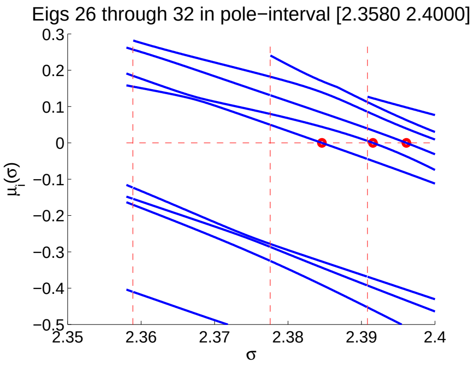
Analysis (assume $\theta_k,\ 1 \leq k \leq d$, is a simple pole)

We have

$$\mu_j(\sigma) = \min_{\mathcal{U}^j, \dim(\mathcal{U}^j) = j} \ \max_{r \in \mathcal{U}^j} \rho(\sigma, r), \qquad \rho(\sigma, r) = \frac{(S(\sigma)r, r)}{(r, r)}.$$

- As $\sigma \to \theta_k^-$, one eigenbranch $(\mu_1(\sigma))$ will descend to $-\infty$
- However, if $r \perp w_k \to$ eigenbranches $\mu_2(\sigma), \ldots, \mu_s(\sigma)$ are well defined and converge to the eigenvalues of an operator which is orthogonal to w_k

Eigenbranches across the poles (II)



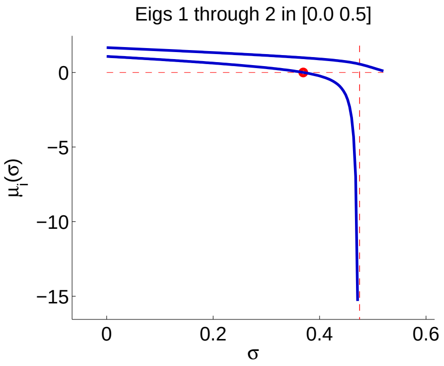


Figure: Behavior of a few eigenbranches as they cross a pole. Red circles: eigenvalues of A (roots). Dashed lines: eigenvalues of B (poles) Left: Eigenbranches $\mu_{26}, \ldots, \mu_{32}$ in an interval not containing their poles. Right: Eigenbranches $\mu_1(\sigma)$ and $\mu_2(\sigma)$ as σ approaches the pole of $\mu_1(\sigma)$.

A branch-hopping scheme

Algorithm 3.2 – "Chasing" more than one eigenvalues

```
1: Given $a, b$. Select $\sigma = a$
2: while $\sigma < b$ do
3:     Compute $S(\sigma)\mu(\sigma) = \mu(\sigma)y(\sigma)$
4:     if $|\mu(\sigma)| \leq \mathrm{tol}$ then
5:         Obtain $\mu(\sigma)$ = smallest positive eigenvalue of $S(\sigma)$
6:     end if
7:     Compute derivative and update $\sigma$ as in Algorithm 3.1
8: end while
```

- Algorithm 3.2 assumes that we move rightwards; can be trivially modified for the case where we move leftwards.

Short illustration - the branch-hopping scheme

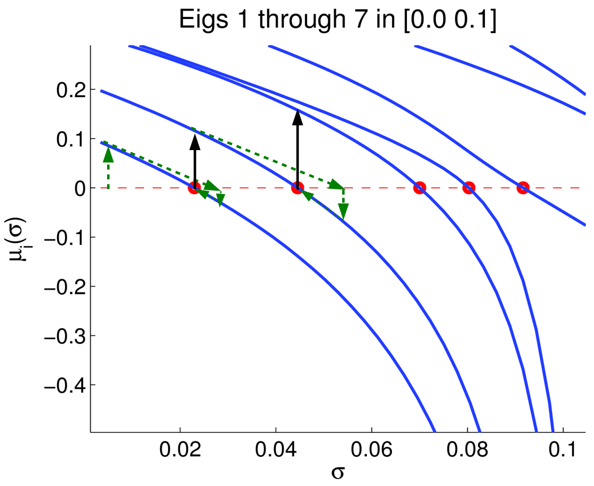
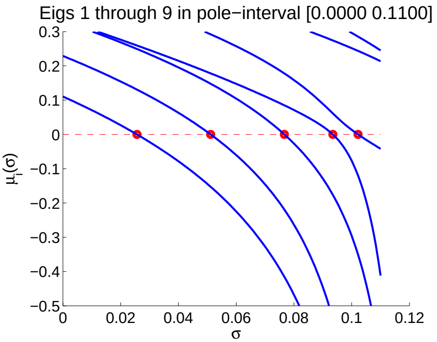


Figure: Computing the few smallest eigenvalues for the same 2D Laplacian

Effects of p in convergence – illustration



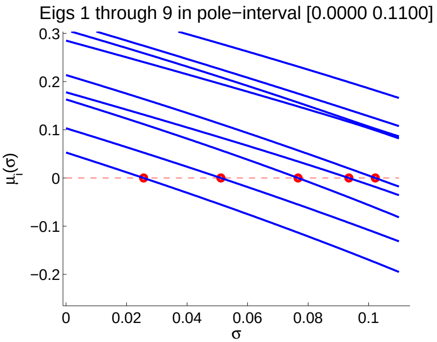


Figure: The shape of eigenbranches $\mu_1(\sigma), \ldots, \mu_9(\sigma)$. Left: $p = 4$. Right: $p = 16$.

Effects of p in convergence – eigenvalues of B as p varies

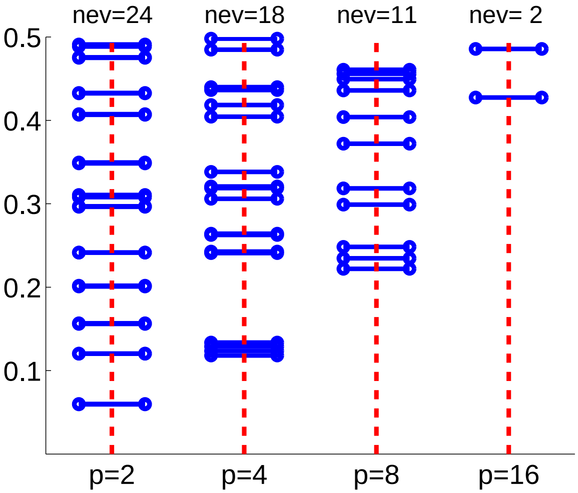


Figure: Monitoring the few smallest eigenvalues of B in a fixed interval as p varies.

A few more considerations

Multiple eigenvalues

Multiple eigenvalue $\lambda \to$ multiple (with the same degree of multiplicity) zero eigenvalue of $S(\lambda)$.

Misconvergence

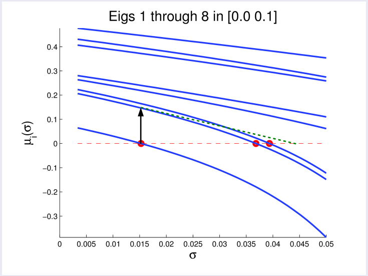


Figure: An example of misconvergence.

It is possible to misconverge when the root of the eigenbranch is close to its pole. To deal with this (potential) issue:

- Standard tool: inertia
- Have "better linear" eigenbranches
- Post-process new σ by inverse iteration

Contents

Experiments

Computational system

- Tests performed on Itasca Linux cluster @ MSI
- Each node is a two-socket, quad-core 2.8 GHz Intel Xeon X5560 "Nehalem EP" with 24 GB of system memory
- Interconnection: 40-gigabit QDR InfiniBand (IB)

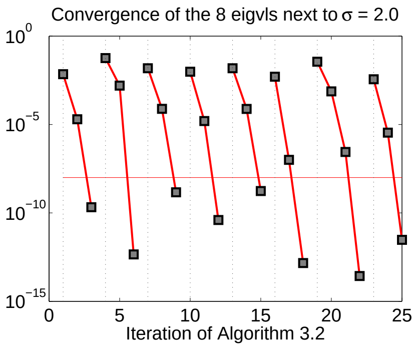
Test matrices

- Tests on 3D dicretized Laplacians (7pt. st. – FD) lying on the unit cube
- Dirichlet boundary conditions
- We use $n_x,\ n_y,\ n_z$ to denote the number of nodes along each dimension
- tol for each eigenpair set to $1e-8$
- *Software*: PETSc C++ (METIS+CHOLMOD)

Qualitative behavior of Alg 3.2 on a few 3D Laplacians

		$[\alpha,\beta] := [0, 0.5]$		$[\alpha,\beta] := [2, 2.2]$		$[\alpha,\beta] := [4.1, 4.2]$	
		#Eigvls	It	#Eigvls	It	#Eigvls	It
$\mathbf{n = 21 \times 20 \times 19}$							
# of subdomains (p)	2	35	60	82	152	127	360
	4		43		130		172
	8		35		116		152
	16		39		96		148
$\mathbf{n = 41 \times 20 \times 19}$							
# of subdomains (p)	2	72	210	154	342	209	424
	4		170		292		314
	8		154		273		310
	16		138		241		300
$\mathbf{n = 41 \times 40 \times 20}$							
# of subdomains (p)	2	160	385	319	703	472	802
	4		354		540		647
	8		296		502		592
	16		270		451		533

Convergence of Alg. 3.2



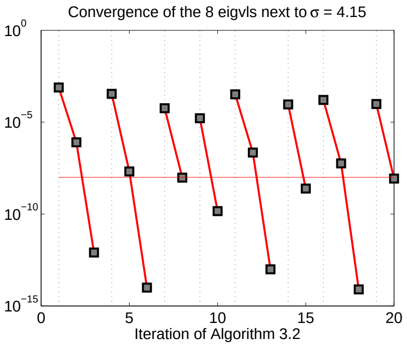


Figure: Rel. res. for a few consecutive eigenvalues. Left: $n_x = 21$, $ny = 20$ and $n_z = 19$. Right: $n_x = 41$, $ny = 20$ and $n_z = 19$.

Combining Alg. 3.1 with inverse iteration

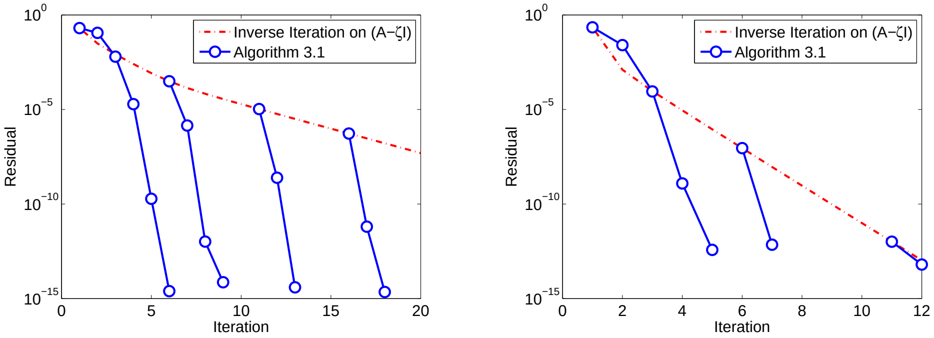


Figure: Convergence behavior for computing the eigenpair closest to ζ when combining inverse iteration and Newton's method. Left: $\zeta = 0.0$. Right: $\zeta = 3.0$.

Experiments in distributed computing environments

Performing the Matrix-Vector product with $S(\zeta)$

1. MV multiplication with $C - \sigma I$ (non-local),
2. MV multiplication with E and E^T (local),
3. system solution with $B - \sigma I$ (local),

Comparisons against residual inverse iteration (RI)

- RI: Similar to inverse iteration but with fixed inner tolerance
- Computation of $(\mu(\sigma), y(\sigma))$ is performed inexactly
- We use MINRES as the iterative solver
- Each subdomain is handled by a separate MPI process

Table: Computing $k = 1$ and $k = 5$ eigenvalues next to ζ for a $41 \times 40 \times 39$ 3D Laplacian. For the case where $\zeta \neq 0$, the starting shift for each particular eigenpair computation in Newton's scheme was provided by first performing three steps of inexact Inverse Iteration. Times are listed in seconds.

$\mathbf{n = 41 \times 40 \times 39}$							
		$\zeta = 0.0$			$\zeta = 0.1$ (19)		
(p, k)	s	T_{NT}	It	T_{RI}	T_{NT}	It	T_{RI}
(16,1)	15423	0.21	3	0.10	1.07	4	1.32
(16,5)	- -	1.39	15	0.62	5.85	19	7.77
(32,1)	20037	0.06	3	0.03	0.27	2	0.90
(32,5)	- -	0.32	14	0.19	1.52	14	4.86
(64,1)	24789	0.09	3	0.04	0.14	3	0.66
(64,5)	- -	0.44	14	0.21	1.01	15	3.51

Table: Computing $k = 1$ and $k = 5$ eigenvalues next to ζ for a $71 \times 70 \times 69$ 3D Laplacian. For the case where $\zeta \neq 0$, the starting shift for each particular eigenpair computation in Newton's scheme was provided by first performing three steps of inexact Inverse Iteration. Times are listed in seconds.

$n = 71 \times 70 \times 69$

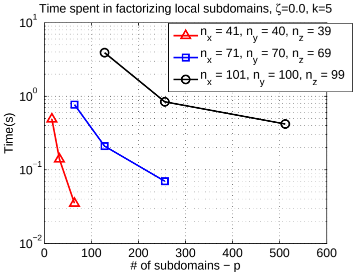
		$\zeta = 0.0$			$\zeta = 0.1$ (137)		
(p, k)	s	T_{NT}	It	T_{RI}	T_{NT}	It	T_{RI}
(64,1)	83358	0.80	3	0.61	15.4	2	15.9
(64,5)	- -	4.20	14	3.22	80.4	10	79.9
(128,1)	108508	0.19	3	0.32	3.12	2	8.41
(128,5)	- -	1.25	14	1.71	15.1	10	38.5
(256,1)	136159	0.10	3	0.27	5.99	2	12.7
(256,5)	- -	0.68	13	1.45	25.3	10	51.7

Table: Computing $k = 1$ and $k = 5$ eigenvalues next to ζ for a $101 \times 100 \times 99$ 3D Laplacian. For the case where $\zeta \neq 0$, the starting shift for each particular eigenpair computation in Newton's scheme was provided by first performing three steps of inexact Inverse Iteration. Times are listed in seconds.

$\mathbf{n = 101 \times 100 \times 99}$

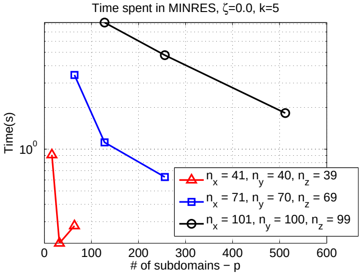
		$\zeta = 0.0$			$\zeta = 0.1$ (439)		
(p, k)	s	T_{NT}	It	T_{RI}	T_{NT}	It	T_{RI}
(128,1)	230849	2.73	3	2.02	48.1	3	93.3
(128,5)	- -	13.2	15	10.3	233.2	16	472.1
(256,1)	293626	1.10	3	1.61	23.4	3	62.4
(256,5)	- -	5.80	14	8.32	129.2	16	301.5
(512,1)	369663	0.62	2	0.99	32.4	2	75.3
(512,5)	- -	3.01	12	5.71	168.9	12	322.9

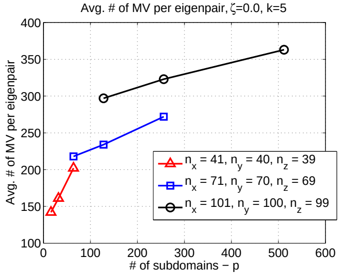
Time breakdown



For the case $\zeta = 0.0, \ k = 5$:

1. Time to factorize $B - \sigma I$
2. Time spent on solving linear systems
3. Avg. of MINRES iters per eigenpair





A comparison with ARPACK

Table: Computing $k = 1$ and $k = 5$ eigenvalues next to ζ with the proposed Newton scheme and ARPACK. The discretization selected as $n_x = 71$, $n_y = 70$, and $n_z = 69$. Times are listed in seconds. Newton's scheme is using 8 cores.

	$\zeta = 0.0$		$\zeta = 0.1$ (137)	
(p, k)	T_{NT}	T_{ARP}	T_{NT}	T_{ARP}
(64,1)	5.5	35.4	170.0	351.5
(128,1)	3.4	–	105.1	–
(256,1)	5.3	–	122.5	–
(64,5)	28.3	94.1	884.7	416.3
(128,5)	15.3	–	532.3	–
(256,5)	25.9	–	605.3	–

Contents

Conclusion

In this talk

- We presented a Schur complement eigenvalue solver that focuses solely on the interface problem region
- Eigenvalue branches of the SSC $\rightarrow$ Newton's method
- Potential for 2-D level parallelism
- Ultimately, k eigenpairs are computed at the cost of one

Considerations

- Use subspace acceleration
- Solution of generalized eigenvalue problems
- Combine with an efficient mechanism to obtain initial guesses (AMLS?)
- Use Newton's method on a higher-order approximation of $S(\lambda)$

Misconvergence

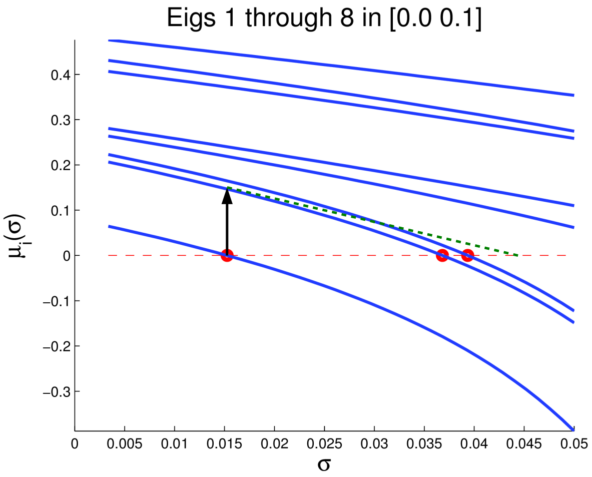


Figure: An example of misconvergence.

Table: Computing $k = 1$ and $k = 5$ eigenvalues next to ζ for a 801×800 2D Laplacian. For the case where $\zeta \neq 0$, the starting shift for each particular eigenpair computation in Newton's scheme was provided by first performing three steps of inexact Inverse Iteration. Times are listed in seconds.

$\mathbf{n = 801 \times 800}$							
		$\zeta = 0.0$			$\zeta = 0.01$ (488)		
(p, k)	s	T_{NT}	It	T_{RI}	T_{NT}	It	T_{RI}
(64,1)	24945	1.09	3	1.25	37.4	2	156.4
(64,5)	- -	5.95	15	6.98	198.7	12	775.1
(128,1)	36611	0.27	2	0.67	24.0	2	75.4
(128,5)	- -	1.31	9	3.82	125.0	11	382.1
(256,1)	52319	0.22	2	0.48	11.2	2	44.9
(256,5)	- -	1.59	9	2.73	61.3	10	231.6

Table: Computing $k = 1$ and $k = 5$ eigenvalues next to ζ for a 1001×1000 2D Laplacian. For the case where $\zeta \neq 0$, the starting shift for each particular eigenpair computation in Newton's scheme was provided by first performing three steps of inexact Inverse Iteration. Times are listed in seconds.

$\mathbf{n = 1001 \times 1000}$

		$\zeta = 0.0$			$\zeta = 0.01$ (764)		
(p, k)	s	T_{NT}	It	T_{RI}	T_{NT}	It	T_{RI}
(128,1)	46073	0.42	3	1.03	95.3	2	102.1
(128,5)	- -	2.84	15	5.33	482.7	10	532.1
(256,1)	65780	0.27	2	0.64	54.2	2	73.4
(256,5)	- -	1.35	10	3.32	281.3	9	381.4
(512,1)	93440	0.25	2	0.58	49.4	2	58.1
(512,5)	- -	1.42	10	3.21	256.7	10	312.8

Eigenbranches across the poles (III)

Let θ_k^- be a simple eigenvalue of B and define

$$S_k(\sigma) = C - \sigma I - \sum_{j=1, j\neq k}^{d} \frac{w_j w_j^T}{\theta_j - \sigma}, \quad \rho_k(\sigma, r) = \frac{(S_k(\sigma)r, r)}{(r, r)}$$

In the following we assume that $\sigma \in [\sigma_0, \theta_k]$ where $[\sigma_0, \theta_k)$ contains no eigenvalues of B.
As $\sigma \to \theta_k^-$, for any $j > 1$, we have

$$\mu_j(\sigma) = \rho(\sigma, y^{(j)}(\sigma)) = \rho_k(\sigma, y^{(j)}(\sigma)) - \frac{(y^{(j)}(\sigma)^T w_k)^2}{\theta_k - \sigma},$$

and since $\rho_k(\sigma, y^{(j)}(\sigma))$ is bounded, we get $\lim_{\sigma \to \theta_k^-} w_k^T y^{(j)}(\sigma) = 0$.

Eigenbranches across the poles (IV)

Let

$$P_k = I - w_k w_k^T/(w_k^T w_k) = I - \hat{w}_k \hat{w}_k^\top .$$

Multiplying $S(\sigma)y^{(j)}(\sigma) = \mu_j(\sigma)y^{(j)}(\sigma)$ by P_k from the left yields

$$P_k S_k(\sigma) P_k y^{(j)}(\sigma) - \mu_j(\sigma) P_k y^{(j)}(\sigma) = -P_k S_k(\sigma)(\hat{w}_k^T y^{(j)}(\sigma))\hat{w}_k .$$

Therefore, the eigenpair $(\mu_j(\sigma), P_k y^{(j)}(\sigma))$ converges to an eigenpair of $P_k S_k(\theta_k) P_k$, which is a trivial extension of $S_{k,|}(\sigma) = \left[P_k S_k(\sigma) P_k\right]_{|w_k{}^\perp}$.