

The FEAST Algorithm for Sparse Symmetric Eigenvalue Problems

Susanne Bradley
RPE Defense
University of British Columbia
November 2, 2016

Problem statement

- FEAST algorithm: find all eigenpairs (λ, \mathbf{x}) of a real symmetric matrix \mathbf{A} with
$$\lambda \in [\lambda_-, \lambda_+] =: \mathcal{L}.$$
- Our focus: \mathbf{A} is large and sparse
 - FEAST involves linear system solves – want to use iterative solvers

FEAST overview

Subspace iteration

Pick p random n -vectors

$$\mathbf{Q}_{(0)} = [\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_p]$$

For $k = 1, 2, \dots$

$$\tilde{\mathbf{Y}}_{(k)} \leftarrow \mathbf{A} \cdot \mathbf{Q}_{(k-1)}$$

Orthonormalize $\tilde{\mathbf{Y}}_{(k)}$ into $\mathbf{Y}_{(k)}$

Form $p \times p$ system $\hat{\mathbf{A}} = \mathbf{Y}_{(k)}^H \mathbf{A} \mathbf{Y}_{(k)}$

Compute eigenpairs $(\hat{\lambda}, \hat{\mathbf{X}})$ of $\hat{\mathbf{A}}$

$$\mathbf{Q}_{(k)} \leftarrow \mathbf{Y}_{(k)} \hat{\mathbf{X}}$$

$\mathbf{Q}_{(k)}$: converges linearly to the p dominant eigenvectors of \mathbf{A} (convergence factor to eigenvector i : $|\lambda_{p+1}/\lambda_i|$)

FEAST overview

Subspace iteration + filtering

Pick p random n -vectors

$$\mathbf{Q}_{(0)} = [\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_p]$$

For $k = 1, 2, \dots$

$$\tilde{\mathbf{Y}}_{(k)} \leftarrow \rho(\mathbf{A}) \cdot \mathbf{Q}_{(k-1)}$$

Orthonormalize $\tilde{\mathbf{Y}}_{(k)}$ into $\mathbf{Y}_{(k)}$

Form $p \times p$ system $\hat{\mathbf{A}} = \mathbf{Y}_{(k)}^H \mathbf{A} \mathbf{Y}_{(k)}$

Compute eigenpairs $(\hat{\lambda}, \hat{\mathbf{X}})$ of $\hat{\mathbf{A}}$

$$\mathbf{Q}_{(k)} \leftarrow \mathbf{Y}_{(k)} \hat{\mathbf{X}}$$

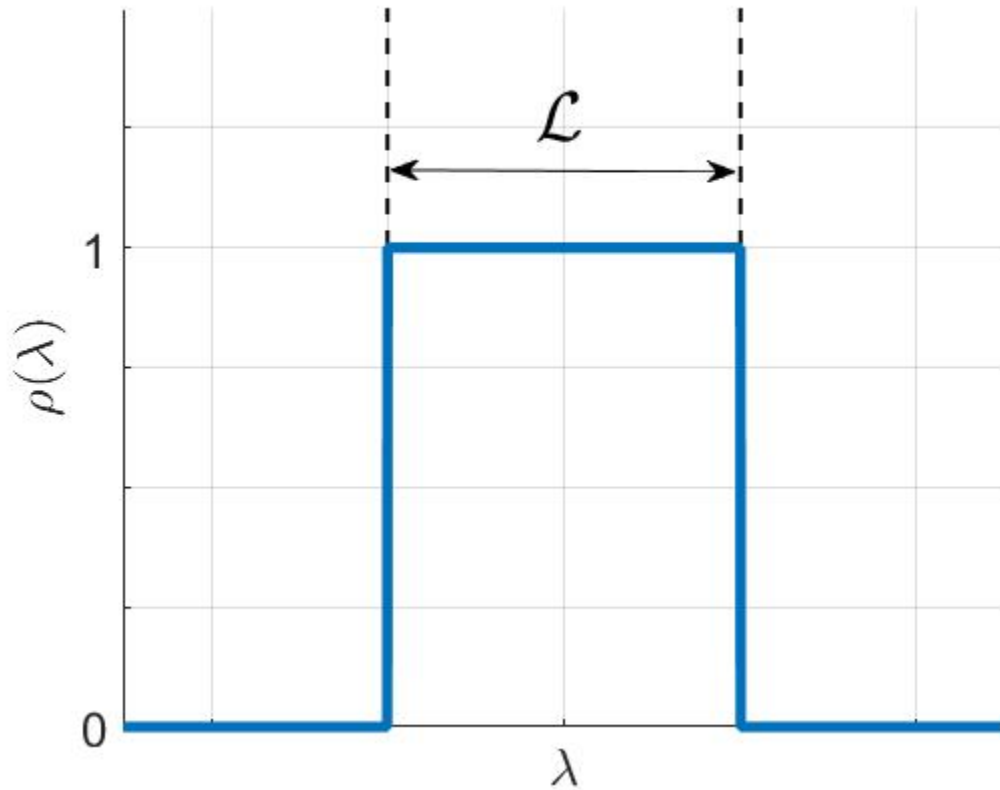
...where $\rho(\mathbf{A})$ preserves eigenvectors and maps eigenvalues λ to $\rho(\lambda)$.

$\mathbf{Q}_{(k)}$: converges linearly to the p dominant eigenvectors of $\rho(\mathbf{A})$ (convergence factors: $|\rho(\lambda_{p+1})/\rho(\lambda_i)|$)

Examples:

- $\rho(\mathbf{A}) = \mathbf{A}^k$
 $\rightarrow \rho(\lambda) = \lambda^k$
- $\rho(\mathbf{A}) = (\mathbf{A} - \sigma \mathbf{I})^{-1}$
 $\rightarrow \rho(\lambda) = \frac{1}{\lambda - \sigma}$

FEAST overview: choice of filter



$$\rho(\mathbf{A}) = ???$$

Ideal: indicator function over \mathcal{L}

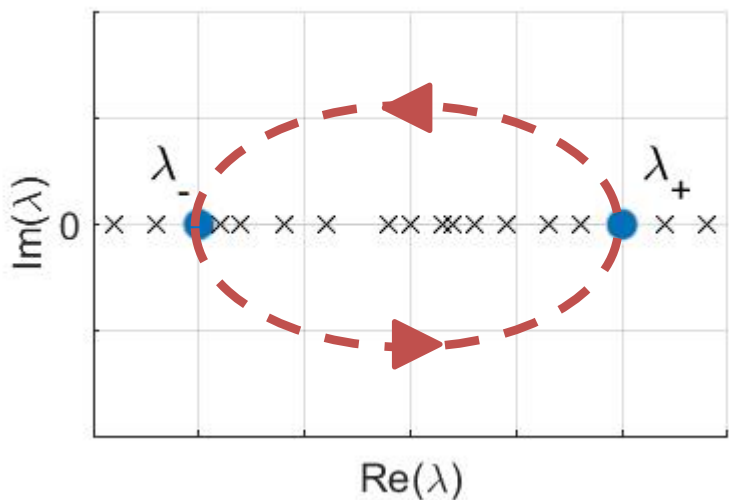
FEAST overview: choice of filter

Cauchy's Integral Theorem

Let Γ be a contour in the complex plane. Then the (counter-clockwise) contour integral

$$\pi(\lambda) = \frac{1}{2\pi i} \oint_{\Gamma} \frac{1}{z - \lambda} dz, \quad \lambda \notin \Gamma$$

is equal to 1 for λ enclosed in Γ , and 0 for λ not enclosed in Γ .

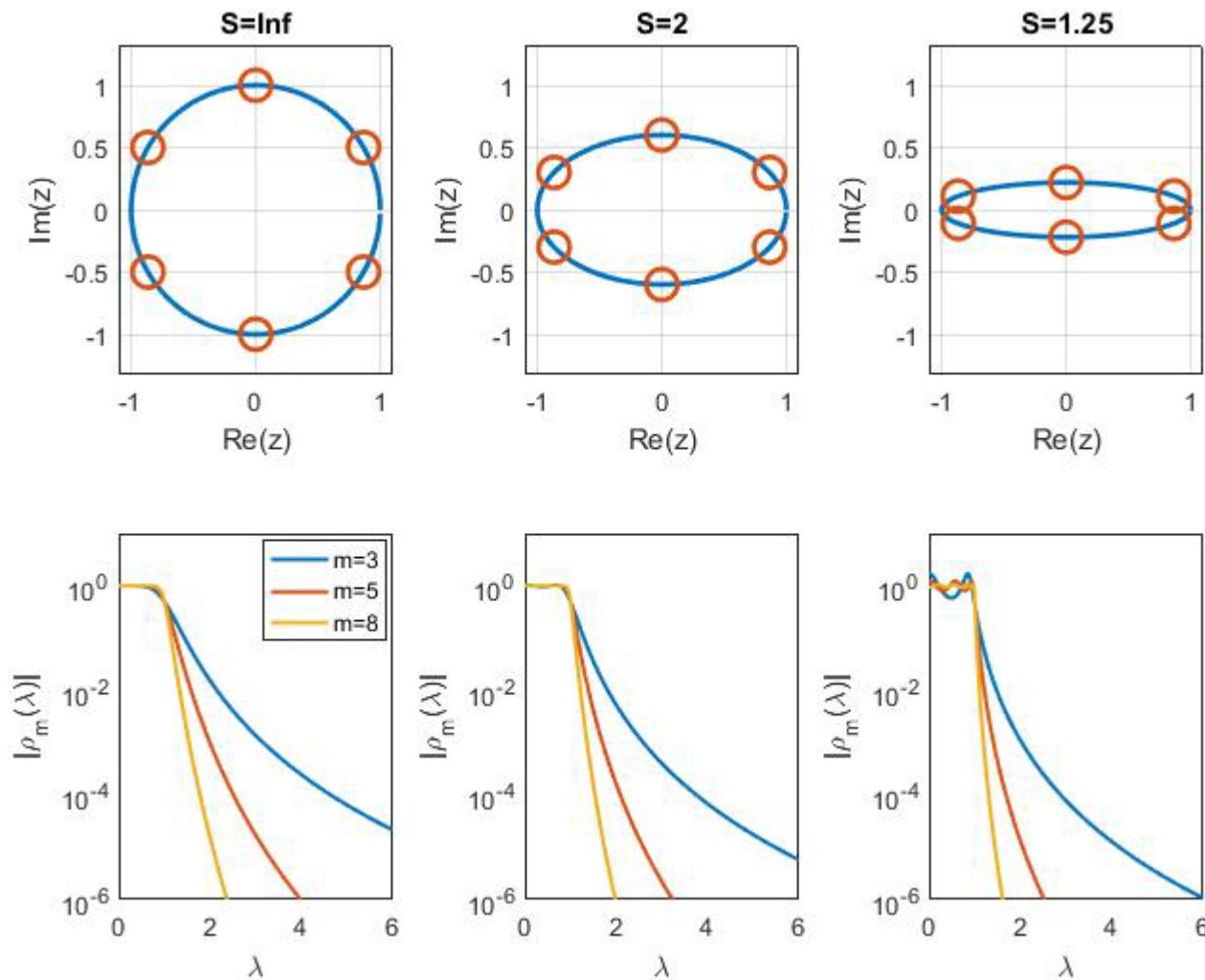


k -point
quadrature



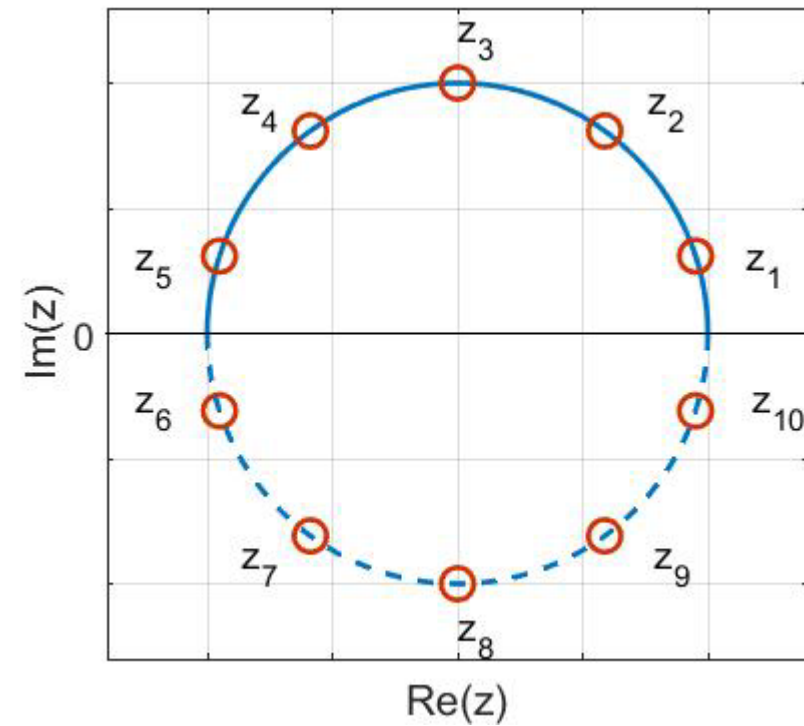
$$\pi(\lambda) \approx \rho(\lambda) := \sum_{i=1}^k \frac{w_i}{z_i - \lambda}$$
$$\rho(\mathbf{A}) := \sum_{i=1}^k w_i (z_i \mathbf{I} - \mathbf{A})^{-1}$$

Trapezoidal rule



On $[-1,1]$ for different (half) number of nodes m

Computing $\rho(\mathbf{A})\mathbf{Q}$



Trapezoidal quadrature points
 $S = \infty, m = 5$

- For $\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_p]$,
 $\rho(\mathbf{A})\mathbf{Q} = \sum_{i=1}^{2m} w_i (z_i \mathbf{I} - \mathbf{A})^{-1} \mathbf{Q}$
- **Note:** nodes z are in complex conjugate pairs
- For real \mathbf{A}, \mathbf{Q} :
 $(\bar{z}\mathbf{I} - \mathbf{A})^{-1} \mathbf{Q} = \overline{(z\mathbf{I} - \mathbf{A})^{-1} \mathbf{Q}}$
- Therefore: in real case, only need to solve mp linear systems to compute $\rho(\mathbf{A})\mathbf{Q}$ (instead of $2mp$)

Computing $\rho(\mathbf{A})\mathbf{Q}$

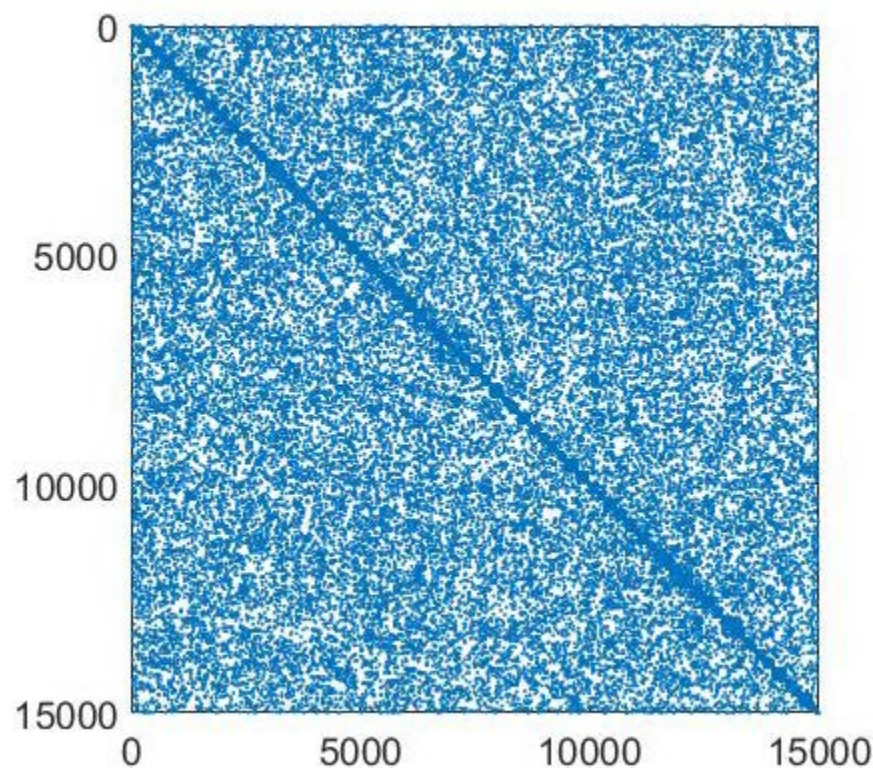
At each FEAST iteration: solve mp linear systems
 $(z_i \mathbf{I} - \mathbf{A})\mathbf{y}_{i,j} = \mathbf{q}_j, \quad 1 \leq i \leq m, 1 \leq j \leq p.$

- Usual approach in FEAST literature: use direct solvers
 - What if \mathbf{A} is large and sparse?
 - ...or we don't need highly accurate solutions?
- Our goal: use iterative solvers instead
 - Some challenges:
 - Systems are indefinite, non-Hermitian (even for Hermitian \mathbf{A})
 - Conditioning issues
 - Lots of right-hand sides

Numerical experiments

- Direct solvers vs. preconditioned GMRES
- Two matrices: finding 100 eigenpairs of each (with search space size $p = 130$)
- FEAST tolerance: 10^{-10}
- Quadrature: $S = 1.05$, $m = 8$
- Coded in MATLAB

Matrix 1: Andrews



$n = 60,000$; $nnz = 760,154$

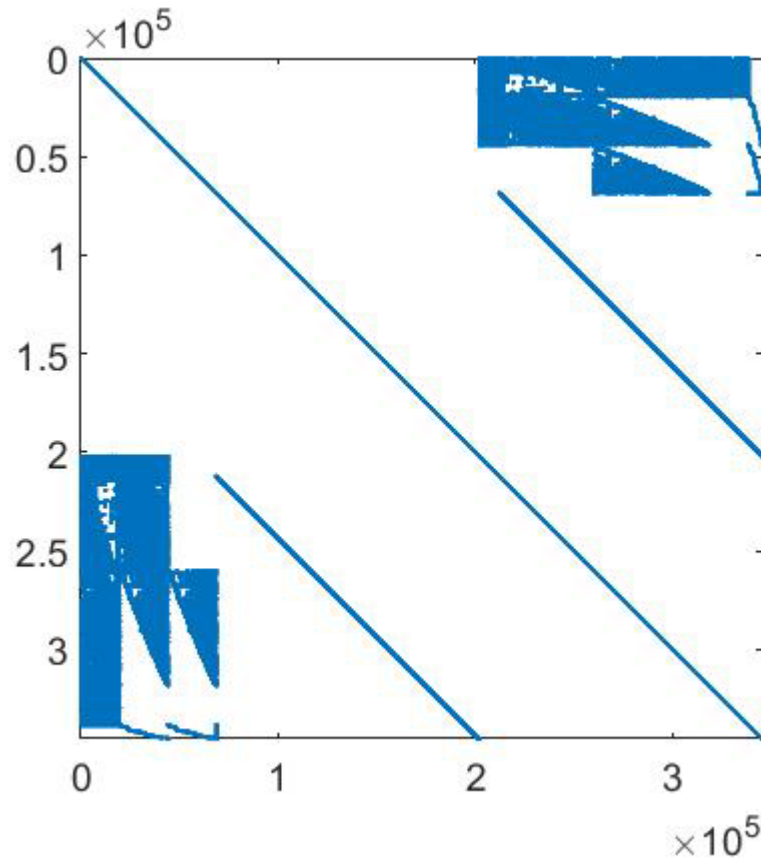
- $[\lambda_-, \lambda_+] = [0.0, 1.3]$
(100 smallest)
- GMRES solver:
 - Tolerance: 10^{-10}
 - Preconditioner: ILUTP
($\tau = 5.0 \times 10^{-3}$)
- Direct solver: backslash

Andrews matrix results

	GMRES with ILUTP	Direct solver
$nnz(L + U)$ (average over all shifted systems)	2,767,412	234,019,880
FEAST iterations to convergence	3	3
Average time per LS solve	7.9 s (119.0 iterations)	2.0 s
Total time	26,462 s*	6,530 s

*= includes time for preconditioner construction

Matrix 2: c-big



$n = 345,241$; $nnz = 2,340,859$

- $[\lambda_-, \lambda_+] = [45.2, 50.0]$ (interior)
- GMRES solver:
 - Tolerance: 10^{-10}
 - Preconditioner: ILUTP ($\tau = 1.0 \times 10^{-3}$)

c-big results

	GMRES with ILUTP	Direct solver
$nnz(L + U)$ (average over all shifted systems)	5,697,775	144,569,235
FEAST iterations to convergence	3	3
Average time per LS solve	2.5 s (8.5 iterations)	1.6 s
Total time	7,795 s*	4,979 s

*= includes time for preconditioner construction

Discussion/considerations

- Speed vs. memory
 - On our 60k matrix – per linear system: iterative solver takes 4x the time, $\sim 1/25^{\text{th}}$ of the memory
 - On 345k: 1.6x the time, $\sim 1/17^{\text{th}}$ of the memory
- Direct solver loses a level of parallelism
 - Can overcome this (to an extent) with a different direct solver
- Lots of improvements to explore for the iterative solvers...

Future work

- Krylov space recycling
- Preconditioned MINRES
- More parallelization
- Generalized and non-Hermitian problems

Thank you

- Chen
- Robert and Uri
- Ian
- Audience