FEAST v4.0 with Applications

Eric Polizzi
Department of Electrical and Computer Engineering
Department of Mathematics and Statistics
University of Massachusetts, Amherst

Sparse Days, CERFACS, Toulouse, 2019

FEAST for First-Principle Calculations

Ground-State Calculations
DFT/Kohn-Sham/All-electrons

$$\widehat{H} = -\frac{\hbar^2}{2m}\nabla^2 + v_H(r) + v_{ext}(r) + v_{xc}(r)$$

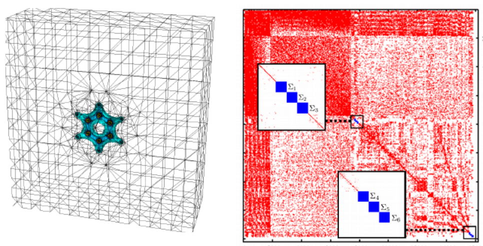
$$\widehat{H}\psi_i(r) = E_i\psi_i(r) \qquad n(r) = \sum_i |\psi_i(r)|^2$$

Excited-State Calculations
Time-dependent DFT (TDDFT)
ALDA/AGGA

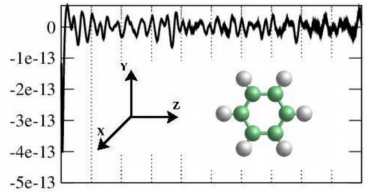
$$i\hbar\frac{\partial}{\partial t}\Psi(t) = \hat{H}(t)\Psi(t)$$

N E S S I E

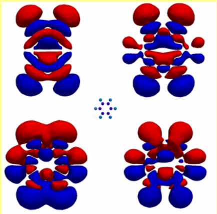
Real-Space Discretization

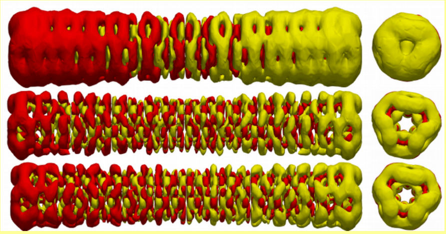


Real-Time Propagation



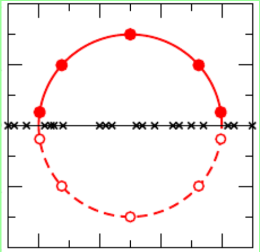
From Molecules to Nanostructures





FEAST Eigensolver

F E A S T



FEAST Solver Library

Design a robust, parallel and unified framework for solving the "interior" eigenvalue problems

Family of Eigenvalue Problems

- Hermitian $\mathbf{A}\mathbf{x} = \lambda \mathbf{B}\mathbf{x}$, $\mathbf{A}$ Herm., $\mathbf{B}$ spd/hpd
- non-Hermitian $\mathbf{A}\mathbf{x} = \lambda \mathbf{B}\mathbf{x}$, $\mathbf{A}, \mathbf{B}$ general
- Non-linear eigenvector $\mathbf{A}(\{\mathbf{x}\})\mathbf{x} = \lambda \mathbf{B}\mathbf{x}$, $\mathbf{A}$ Herm., $\mathbf{B}$ spd/hpd
- Non-linear eigenvalue $\mathbf{A}(\lambda)\mathbf{x} = \lambda \mathbf{B}\mathbf{x}$, $\mathbf{A}, \mathbf{B}$ general

FEAST Solver Library

Design a robust, parallel and unified framework for solving the "interior" eigenvalue problems

Family of Eigenvalue Problems

- Hermitian $\mathbf{Ax} = \lambda \mathbf{Bx}$, $\mathbf{A}$ Herm., $\mathbf{B}$ spd/hpd
- non-Hermitian $\mathbf{Ax} = \lambda \mathbf{Bx}$, $\mathbf{A}, \mathbf{B}$ general
- Non-linear eigenvector $\mathbf{A}(\{\mathbf{x}\})\mathbf{x} = \lambda \mathbf{Bx}$, $\mathbf{A}$ Herm., $\mathbf{B}$ spd/hpd
- Non-linear eigenvalue $\mathbf{A}(\lambda)\mathbf{x} = \lambda \mathbf{Bx}$, $\mathbf{A}, \mathbf{B}$ general

Release dates

- v1.0 (2009): Hermitian problem
- v2.0 (2012): SMP+MPI+RCI interfaces
- v2.1 (2013): **Adoption by Intel-MKL**
- v3.0 (2015): Support for non-Hermitan
- v4.0 (fall 2019): Residual inverse iter.

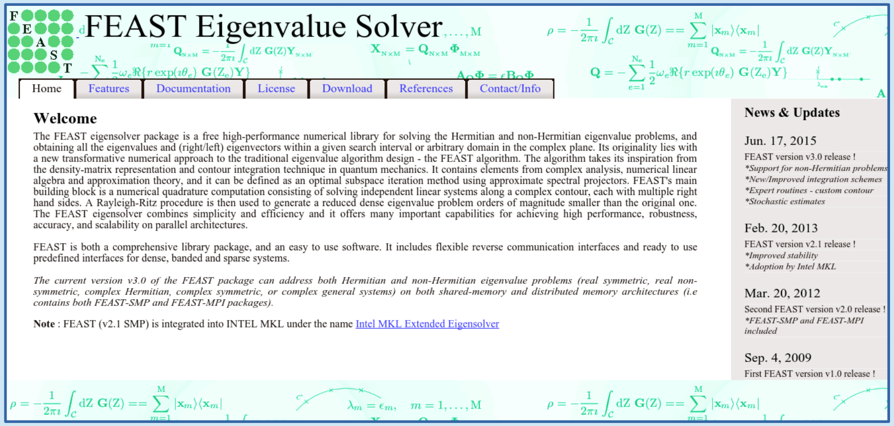
  -PFEAST (3 MPI levels)

  -IFEAST (FEAST w/o factorization)

  -mixed precision

  -non-linear (polynomial)

www.feast-solver.org


FEAST Eigenvalue Solver

Home | Features | Documentation | License | Download | References | Contact/Info

Welcome

The FEAST eigensolver package is a free high-performance numerical library for solving the Hermitian and non-Hermitian eigenvalue problems, and obtaining all the eigenvalues and (right/left) eigenvectors within a given search interval or arbitrary domain in the complex plane. Its originality lies with a new transformative numerical approach to the traditional eigenvalue algorithm design - the FEAST algorithm. The algorithm takes its inspiration from the density-matrix representation and contour integration technique in quantum mechanics. It contains elements from complex analysis, numerical linear algebra and approximation theory, and it can be defined as an optimal subspace iteration method using approximate spectral projectors. FEAST's main building block is a numerical quadrature computation consisting of solving independent linear systems along a complex contour, each with multiple right hand sides. A Rayleigh-Ritz procedure is then used to generate a reduced dense eigenvalue problem orders of magnitude smaller than the original one. The FEAST eigensolver combines simplicity and efficiency and it offers many important capabilities for achieving high performance, robustness, accuracy, and scalability on parallel architectures.

FEAST is both a comprehensive library package, and an easy to use software. It includes flexible reverse communication interfaces and ready to use predefined interfaces for dense, banded and sparse systems.

The current version v3.0 of the FEAST package can address both Hermitian and non-Hermitian eigenvalue problems (real symmetric, real non-symmetric, complex Hermitian, complex symmetric, or complex general systems) on both shared-memory and distributed memory architectures (i.e contains both FEAST-SMP and FEAST-MPI packages).

Note : FEAST (v2.1 SMP) is integrated into INTEL MKL under the name Intel MKL Extended Eigensolver

News & Updates

Jun. 17, 2015
FEAST version v3.0 release !
**Support for non-Hermitian problems*
**New/Improved integration schemes*
**Expert routines - custom contour*
**Stochastic estimates*

Feb. 20, 2013
FEAST version v2.1 release !
**Improved stability*
**Adoption by Intel MKL*

Mar. 20, 2012
Second FEAST version v2.0 release !
**FEAST-SMP and FEAST-MPI included*

Sep. 4, 2009
First FEAST version v1.0 release !

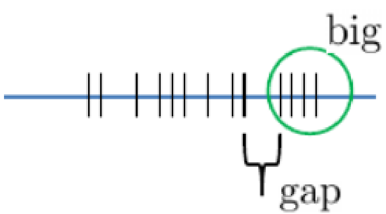
FEAST Algorithm- AX=BXΛ (Hermitian, Generalized)

Subspace iteration with RR

0. Start: Select random subspace $Y_{m_0} \equiv \{y_1, y_2, \ldots, y_{m_0}\}_{n \times m_0}$ $(n >> m_0 \geq m)$
1. Repeat until convergence
2. Compute $Q_{m_0} = \rho(B^{-1}A)Y_{m_0}$
3. Orthogonalize Q_{m_0}
4. Compute $A_Q = Q_{m_0}^H A Q_{m_0}$ and $B_Q = Q_{m_0}^H B Q_{m_0}$
5. Solve $A_Q W = B_Q W \Lambda_Q$ with $W^H B_Q W = I_{m_0 \times m_0}$
6. Compute $Y_{m_0} = Q_{m_0} W$
7. Check convergence of Y_{m_0} and $\Lambda_{Q_{m_0}}$ for the m wanted eigenvalues
8. End

Standard iteration (power method)

$$\rho(B^{-1}A) = B^{-1}A$$



linear CV rate: $|\lambda_{m_0+1}/\lambda_i|_{i=1,\ldots,m}$

Goal: $|\rho(\lambda_{m_0+1})/\rho(\lambda_i)|_{i=1,\ldots,m}$

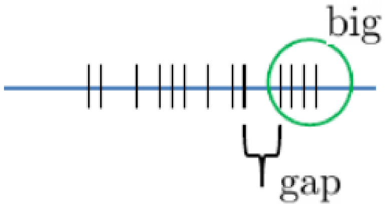
FEAST Algorithm- AX=BXΛ (Hermitian, Generalized)

Subspace iteration with RR

0. Start: Select random subspace $Y_{m_0} \equiv \{y_1, y_2, \ldots, y_{m_0}\}_{n \times m_0}$ $(n >> m_0 \geq m)$
1. Repeat until convergence
2. Compute $Q_{m_0} = \rho(B^{-1}A)Y_{m_0}$
3. Orthogonalize Q_{m_0}
4. Compute $A_Q = Q_{m_0}^H A Q_{m_0}$ and $B_Q = Q_{m_0}^H B Q_{m_0}$
5. Solve $A_Q W = B_Q W \Lambda_Q$ with $W^H B_Q W = I_{m_0 \times m_0}$
6. Compute $Y_{m_0} = Q_{m_0} W$
7. Check convergence of Y_{m_0} and $\Lambda_{Q_{m_0}}$ for the m wanted eigenvalues
8. End

Standard iteration (power method)

$$\rho(B^{-1}A) = B^{-1}A$$



linear CV rate: $|\lambda_{m_0+1}/\lambda_i|_{i=1,\ldots,m}$

Goal: $|\rho(\lambda_{m_0+1})/\rho(\lambda_i)|_{i=1,\ldots,m}$

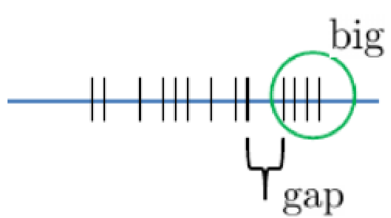
FEAST Algorithm- AX=BXΛ (Hermitian, Generalized)

Subspace iteration with RR

0. Start: Select random subspace $Y_{m_0} \equiv \{y_1, y_2, \ldots, y_{m_0}\}_{n \times m_0}$ $(n >> m_0 \geq m)$
1. Repeat until convergence
2. Compute $Q_{m_0} = \rho(B^{-1}A)Y_{m_0}$
3. Orthogonalize Q_{m_0}
4. Compute $A_Q = Q_{m_0}^H A Q_{m_0}$ and $B_Q = Q_{m_0}^H B Q_{m_0}$
5. Solve $A_Q W = B_Q W \Lambda_Q$ with $W^H B_Q W = I_{m_0 \times m_0}$
6. Compute $Y_{m_0} = Q_{m_0} W$
7. Check convergence of Y_{m_0} and $\Lambda_{Q_{m_0}}$ for the m wanted eigenvalues
8. End

Standard iteration (power method)

$$\rho(B^{-1}A) = B^{-1}A$$

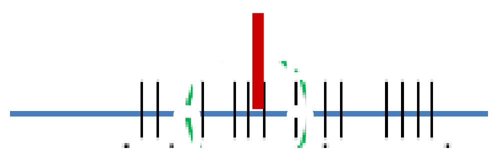


linear CV rate: $|\lambda_{m_0+1}/\lambda_i|_{i=1,\ldots,m}$

Goal: $|\rho(\lambda_{m_0+1})/\rho(\lambda_i)|_{i=1,\ldots,m}$

Shift-invert iteration

$$\rho(B^{-1}A) = (\sigma B - A)^{-1}B$$



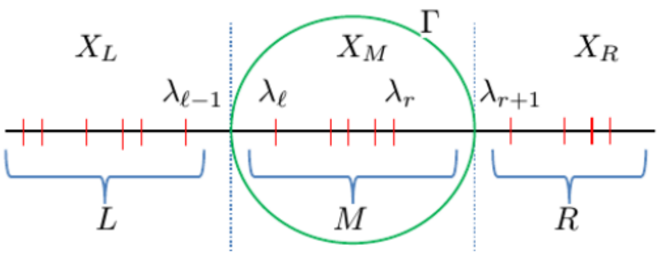
* 1 linear system solve by iteration
* fast CV near the shift
* slow CV elsewhere

FEAST Algorithm

Subspace iteration with RR

0. Start: Select random subspace $Y_{m_0} \equiv \{y_1, y_2, \ldots, y_{m_0}\}_{n \times m_0}$ $(n >> m_0 \geq m)$
1. Repeat until convergence
2. Compute $Q_{m_0} = \rho(B^{-1}A)Y_{m_0}$
3. Orthogonalize Q_{m_0}
4. Compute $A_Q = Q_{m_0}^H A Q_{m_0}$ and $B_Q = Q_{m_0}^H B Q_{m_0}$
5. Solve $A_Q W = B_Q W \Lambda_Q$ with $W^H B_Q W = I_{m_0 \times m_0}$
6. Compute $Y_{m_0} = Q_{m_0} W$
7. Check convergence of Y_{m_0} and $\Lambda_{Q_{m_0}}$ for the m wanted eigenvalues
8. End

Optimal filter for the M interior eigenpairs is given by the spectral projector



$$\rho(B^{-1}A) = X_m X_m^H B$$

FEAST Algorithm

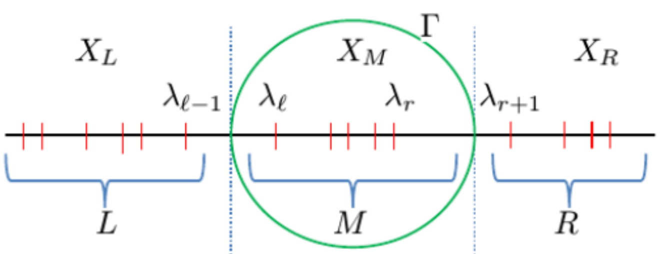
Subspace iteration with RR

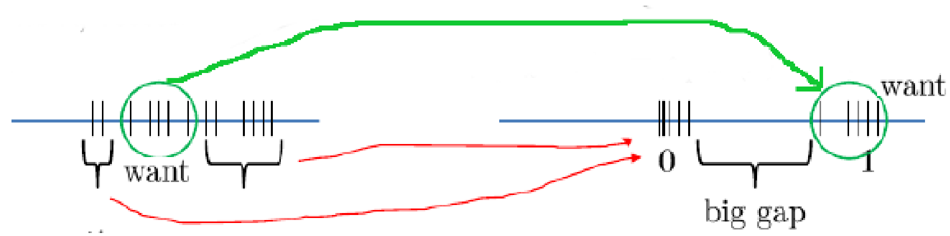
0. Start: Select random subspace $Y_{m_0} \equiv \{y_1, y_2, \ldots, y_{m_0}\}_{n \times m_0}$ $(n >> m_0 \geq m)$
1. Repeat until convergence
2. Compute $Q_{m_0} = \rho(B^{-1}A)Y_{m_0}$
3. Orthogonalize Q_{m_0}
4. Compute $A_Q = Q_{m_0}^H A Q_{m_0}$ and $B_Q = Q_{m_0}^H B Q_{m_0}$
5. Solve $A_Q W = B_Q W \Lambda_Q$ with $W^H B_Q W = I_{m_0 \times m_0}$
6. Compute $Y_{m_0} = Q_{m_0} W$
7. Check convergence of Y_{m_0} and $\Lambda_{Q_{m_0}}$ for the m wanted eigenvalues
8. End

Optimal filter for the M interior eigenpairs is given by the spectral projector

$$\rho(B^{-1}A) = X_m X_m^H B = \frac{1}{2\pi\imath}\oint_{\mathcal{C}} dz(zB - A)^{-1}B$$

$$\rho(\lambda) = \frac{1}{2\pi\imath}\oint_{\mathcal{C}} dz(z-\lambda)^{-1}$$



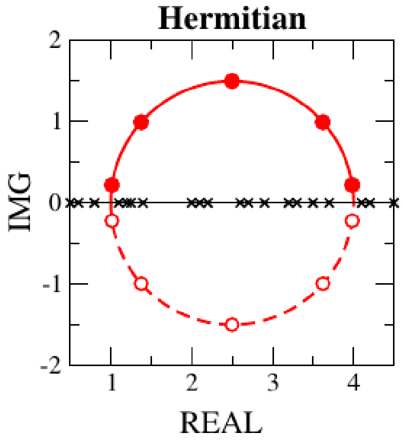


$$\rho(B^{-1}A) = X_m X_m^H B$$

FEAST Algorithm: Numerical Quadrature

Rational function filter

$$\rho_a(z) = \sum_{j=1}^{n_e} \frac{\omega_j}{z_j - z}$$



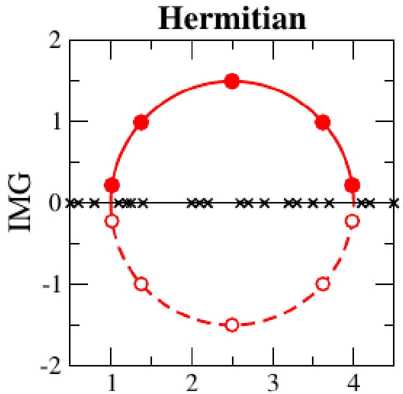
Solving independent linear systems (multiple shifts in complex plane)

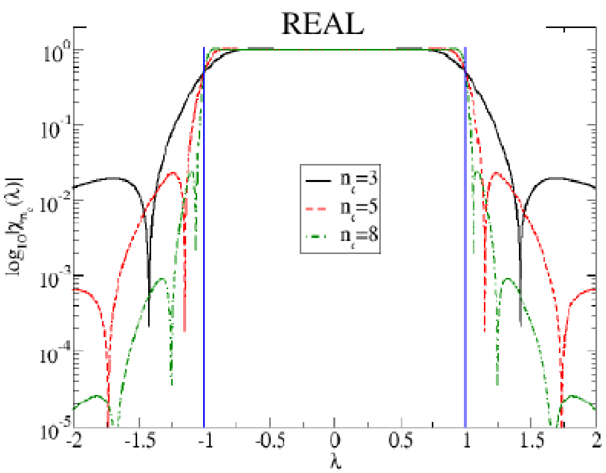
$$Q_{m_0} = \sum_{j=1}^{n_e} \omega_j Q_{m_0}^{(j)} \qquad (z_j B - A) Q_{m_0}^{(j)} = B Y_{m_0}$$

FEAST Algorithm: Numerical Quadrature

Rational function filter

$$\rho_a(z) = \sum_{j=1}^{n_e} \frac{\omega_j}{z_j - z}$$





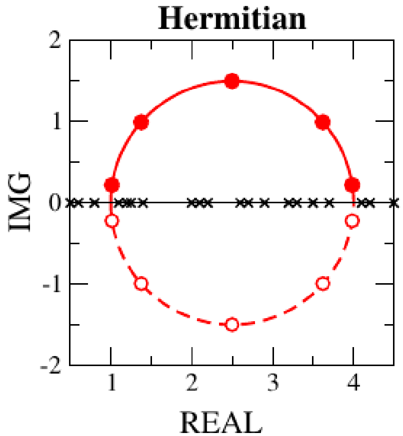
Solving independent linear systems (multiple shifts in complex plane)

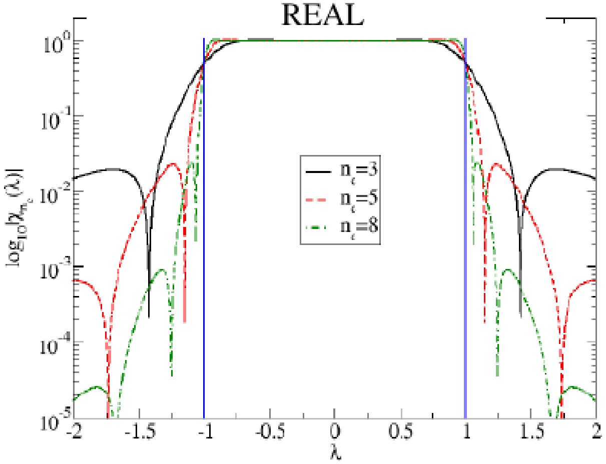
$$Q_{m_0} = \sum_{j=1}^{n_e} \omega_j Q_{m_0}^{(j)} \qquad (z_j B - A) Q_{m_0}^{(j)} = B Y_{m_0}$$

FEAST Algorithm: Numerical Quadrature

Rational function filter

$$\rho_a(z) = \sum_{j=1}^{n_e} \frac{\omega_j}{z_j - z}$$

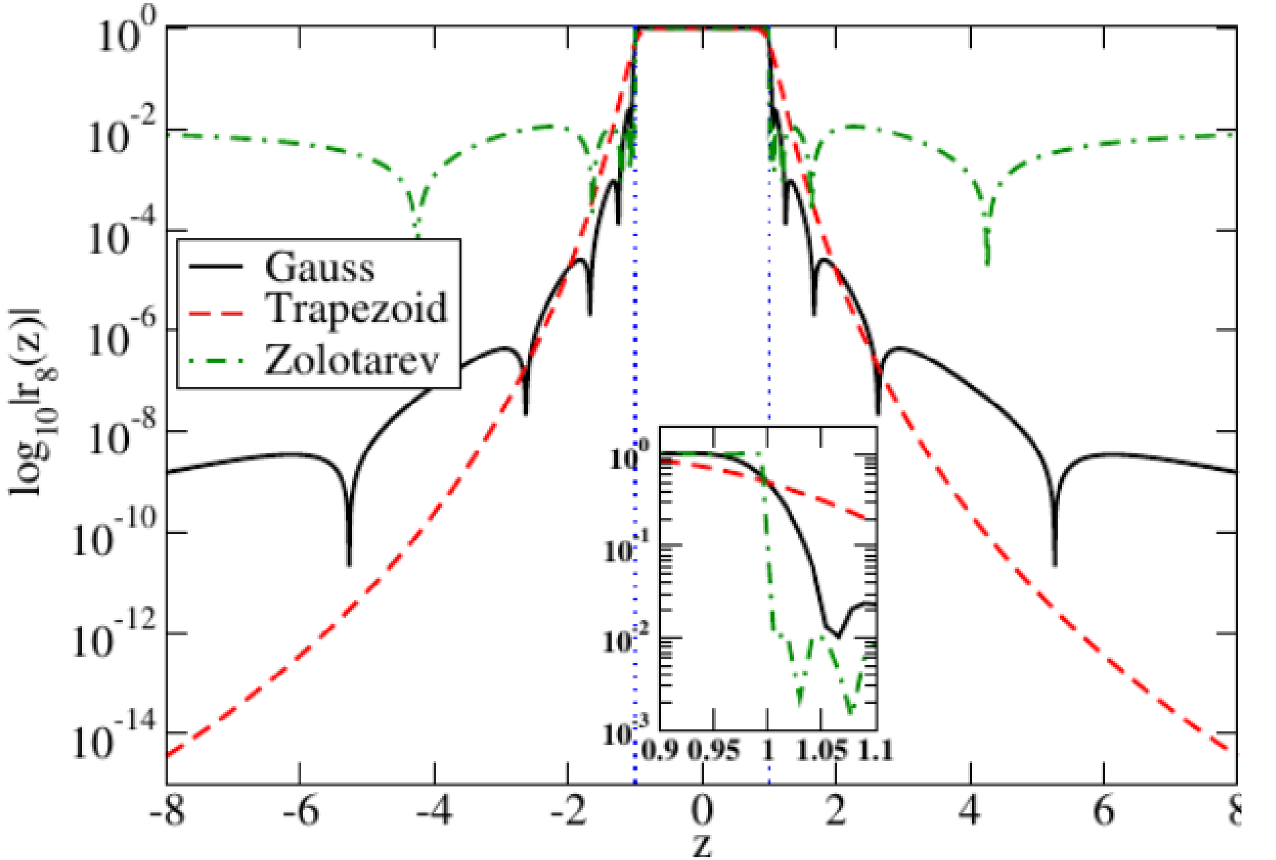




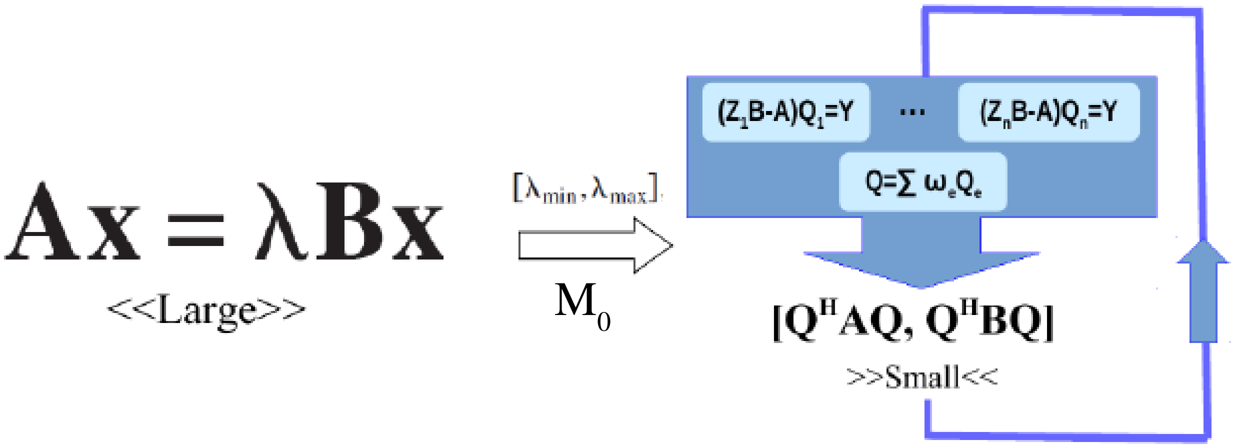
Polizzi, Phys. Rev. B. (2009)
Tang, Polizzi, SIAM SIMAX (2014)
Guettel, Polizzi, Tang, Viaud, SIAM SISC (2015)

Solving independent linear systems (multiple shifts in complex plane)

$$Q_{m_0} = \sum_{j=1}^{n_e} \omega_j Q_{m_0}^{(j)} \qquad (z_j B - A) Q_{m_0}^{(j)} = B Y_{m_0}$$

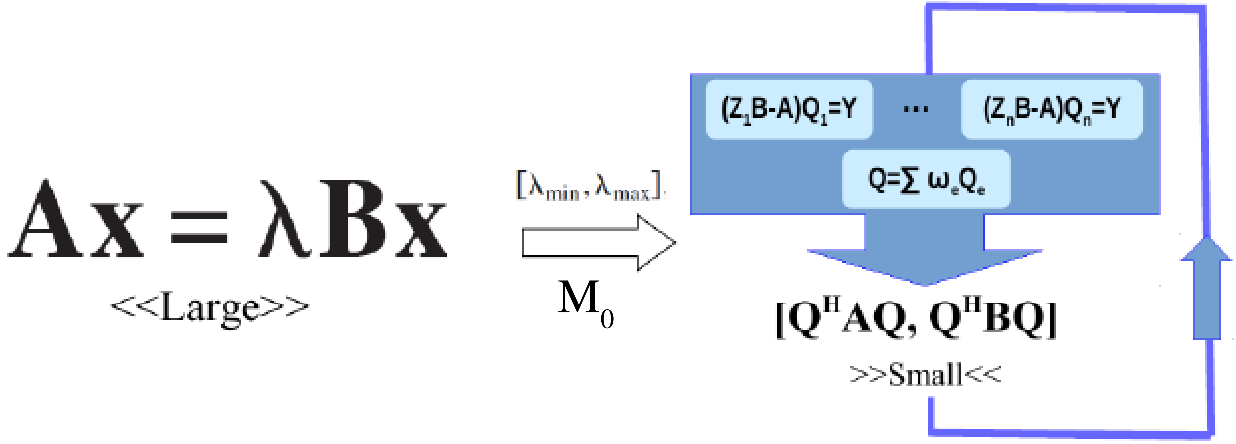


FEAST Algorithm at a glance



- Select search interval
- Select quadrature rule
- Select subspace size M_0

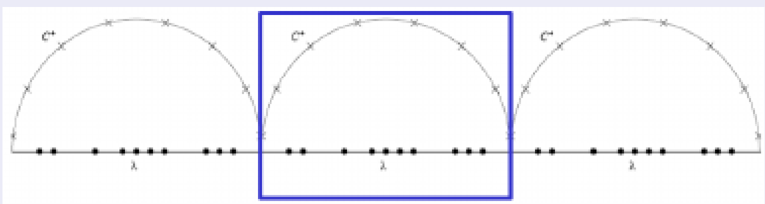
FEAST Algorithm at a glance

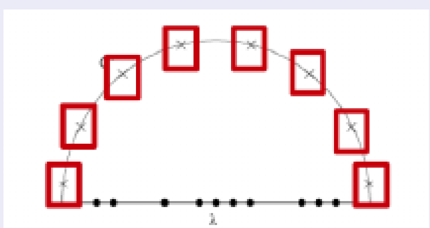


- Select search interval
- Select quadrature rule
- Select subspace size M_0

Properties

- "Fast" and systematic convergence
- Captures all multiplicities
- No explicit orthogonalization
- Reusable subspace
- Allow the use of iterative methods
- Applicable to non-Hermitian problem
- Natural parallelism at three levels





$$(z_e\mathbf{B} - \mathbf{A})\mathbf{Q_e} = \mathbf{Y}$$

FEAST non-Hermitian algorithm

$$\begin{aligned} AX &= BX\Lambda \\ A^H\widehat{X} &= B^H\widehat{X}\Lambda^* \end{aligned} \qquad \widehat{X}^H BX = I$$

Kestyn, Polizzi, Tang, SIAM, SISC (2015)

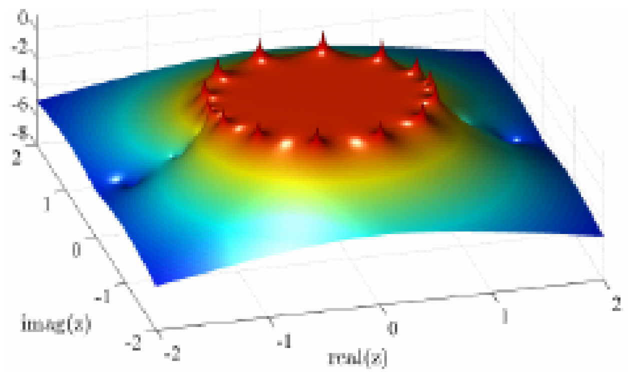
- Right projector

$$\rho(B^{-1}A) = \frac{1}{2\pi\imath}\oint_{\mathcal{C}} dz(zB - A)^{-1}B \equiv X_m\widehat{X}_m^H B.$$

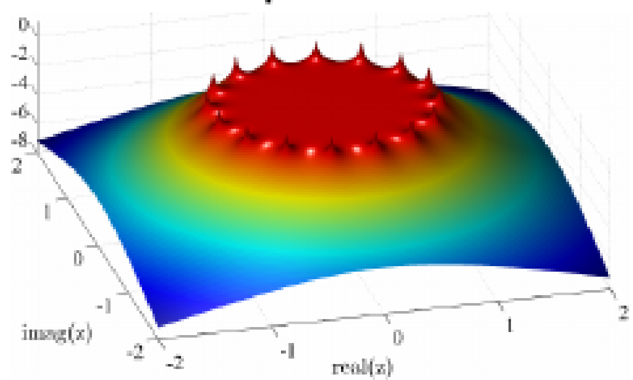
- Left projector

$$\rho(AB^{-1}) = \frac{1}{2\pi\imath}\oint_{\mathcal{C}} dzB(zB - A)^{-1} \equiv BX_m\widehat{X}_m^H$$

Gauss



Trapezoidal



FEAST non-Hermitian algorithm

$$AX = BX\Lambda \qquad \widehat{X}^H B X = I$$
$$A^H \widehat{X} = B^H \widehat{X}\Lambda^*$$

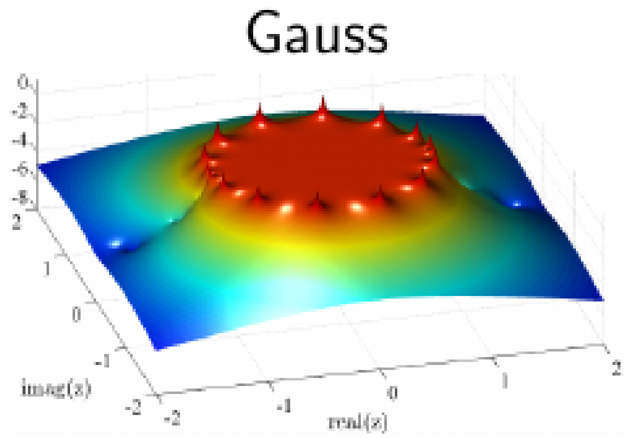
Kestyn, Polizzi, Tang, SIAM, SISC (2015)

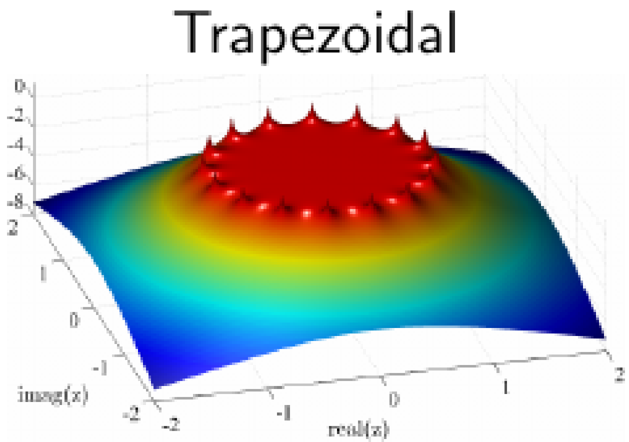
- Right projector

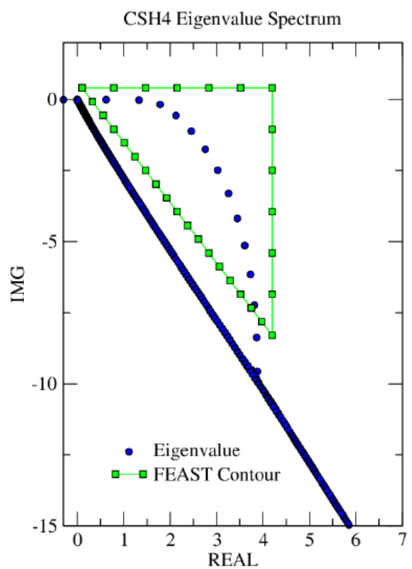
$$\rho(B^{-1}A) = \frac{1}{2\pi\imath}\oint_{\mathcal{C}} dz(zB - A)^{-1}B \equiv X_m \widehat{X}_m^H B.$$

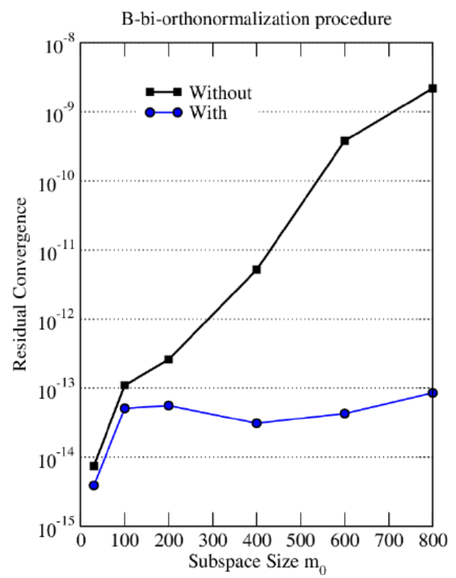
- Left projector

$$\rho(AB^{-1}) = \frac{1}{2\pi\imath}\oint_{\mathcal{C}} dzB(zB - A)^{-1} \equiv BX_m \widehat{X}_m^H$$



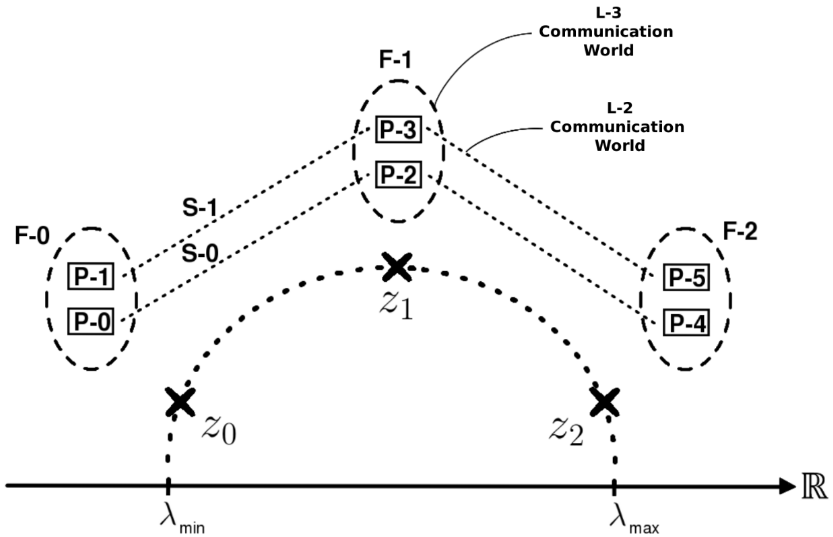






PFEAST (James Kestyn, PhD thesis 2018)

3 MPI communicators



New parallel FEAST interfaces local/global distributions

L1

Distribution of the spectrum (slicing)

L2

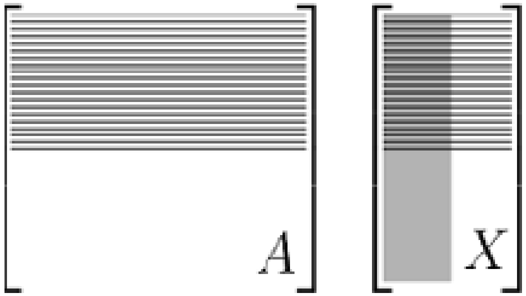
Ideal scalability - requires matrix copies

L3

(Row) Distributed direct solvers: Black-box (cluster pardiso,mumps) and DD custom solvers

L1 and L3 can be used to reduce memory and increase performances

Example with 2L1 and 2L3:



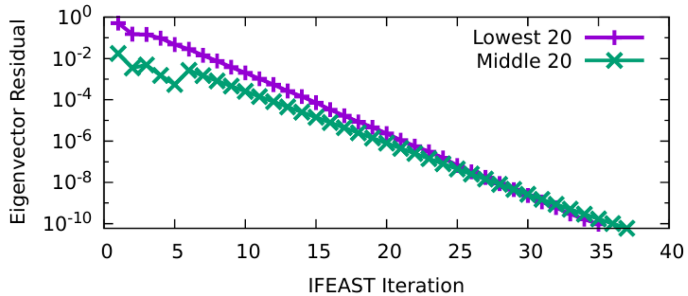
PFEAST
Kestyn, Kalantzis, Polizzi, Saad, supercomputing (2016)

FEAST-DD:
Kalantzis, Kestyn, Polizzi, Saad, NLAA (2018)

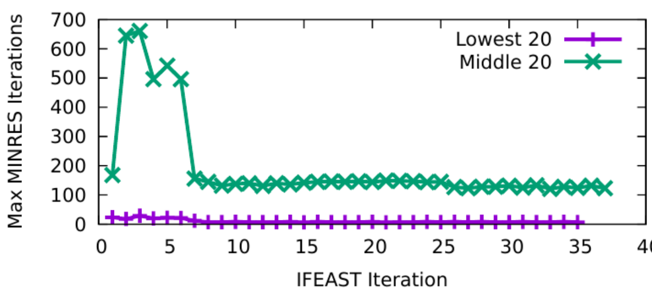
IFEAST- w/o factorization- (Brendan Gavin, PhD thesis 2018)

- FEAST using **inexact** iterative solves $(z_k B - A)\widetilde{y_k} = Bx + r_L$ $\quad \|r_L\| < \alpha\|r_E\|, \; 0 < \alpha < 1$

- *Example:* Parsec Si_2 (**B=I**)



α=0.5



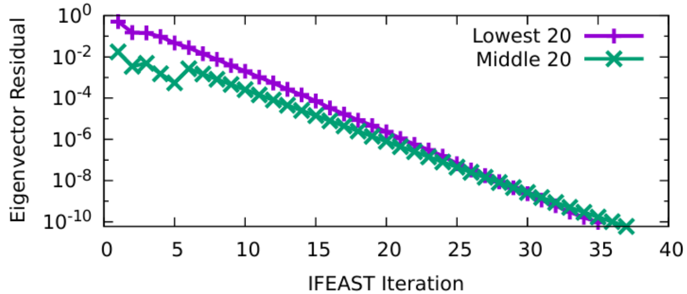
CV rate is still linear $e_{i+1} \leq \left(\frac{\rho(\lambda_{m+1}) + \alpha\Delta}{\rho(\lambda_j)}\right) e_i$

#inner iterations is constant!

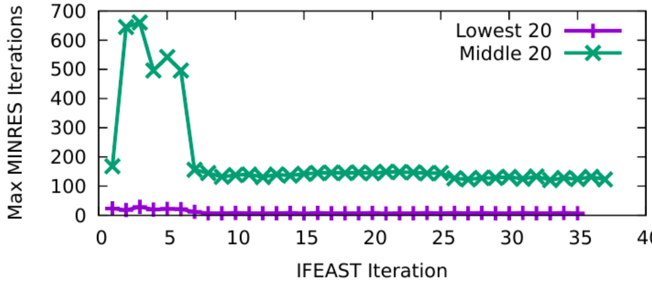
IFEAST- w/o factorization- (Brendan Gavin, PhD thesis 2018)

- FEAST using **inexact** iterative solves $(z_k B - A)\widetilde{y_k} = Bx + r_L$ $\quad \|r_L\| < \alpha \|r_E\|, \; 0 < \alpha < 1$

- *Example:* Parsec Si_2 **(B=I)**



α=0.5



CV rate is still linear $e_{i+1} \leq \left(\frac{\rho(\lambda_{m+1}) + \alpha\Delta}{\rho(\lambda_j)}\right) e_i$

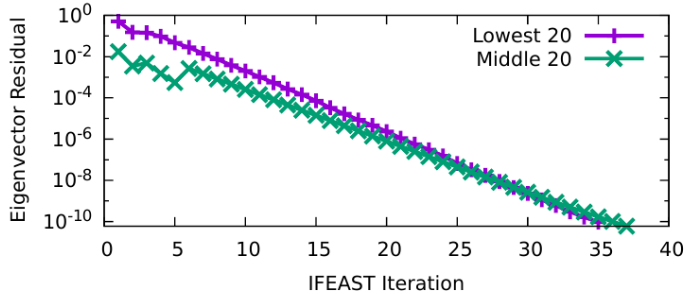
#inner iterations is constant!

- Generalization of previous work on inner-outer iterations with single real shift-invert. *Robbé, Sadkane, Spence, SIMAX, 31(1), p.92, (2009)*

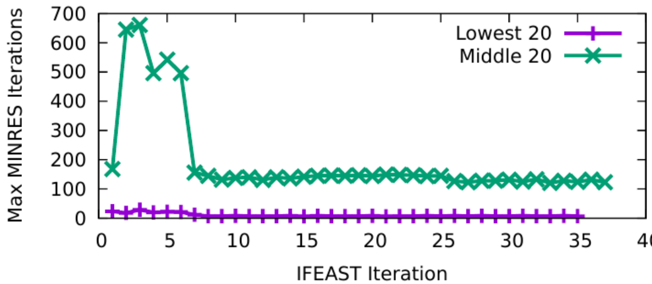
IFEAST- w/o factorization- (Brendan Gavin, PhD thesis 2018)

- FEAST using **inexact** iterative solves $(z_k B - A)\widetilde{y_k} = Bx + r_L$ $\quad \|r_L\| < \alpha\|r_E\|,\ 0 < \alpha < 1$

- *Example:* Parsec Si_2 **(B=I)**



α=0.5



CV rate is still linear $e_{i+1} \le \left(\frac{\rho(\lambda_{m+1}) + \alpha\Delta}{\rho(\lambda_j)}\right) e_i$

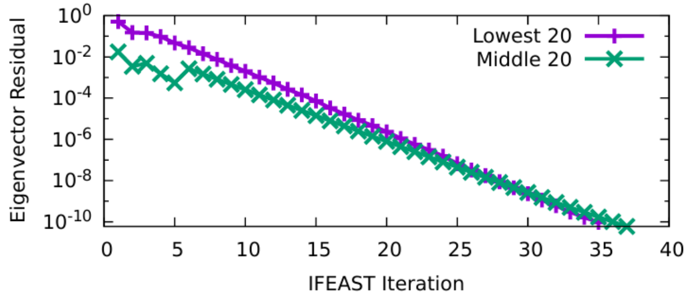
#inner iterations is constant!

- Generalization of previous work on inner-outer iterations with single real shift-invert. *Robbé, Sadkane, Spence, SIMAX, 31(1), p.92, (2009)*
- Formally equivalent to block restarted Krylov ideally suited for interior problem- *Krylov eigenvalue strategy using FEAST with inexact system solves, Gavin, Polizzi: NLAA, (2018)*

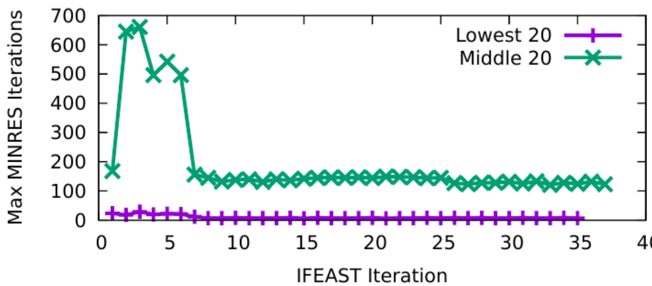
IFEAST- w/o factorization- (Brendan Gavin, PhD thesis 2018)

- FEAST using **inexact** iterative solves $(z_k B - A)\widetilde{y_k} = Bx + r_L$ $\quad \|r_L\| < \alpha \|r_E\|,\ 0 < \alpha < 1$

- *Example:* Parsec Si_2 **(B=I)**



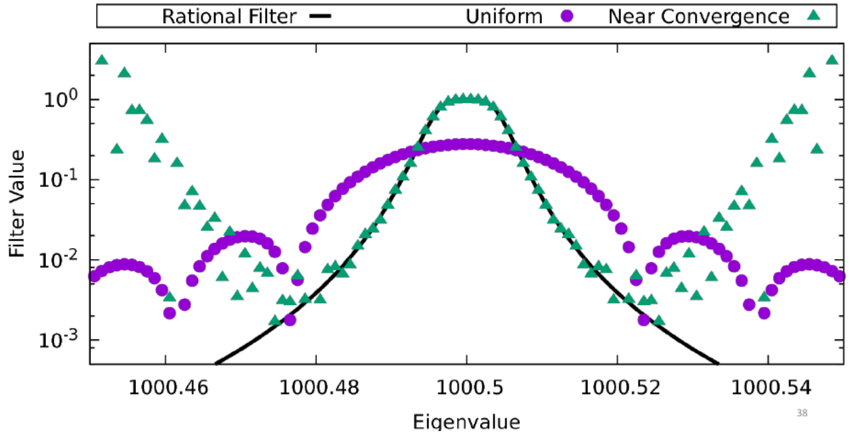
α=0.5



CV rate is still linear $e_{i+1} \leq \left(\frac{\rho(\lambda_{m+1}) + \alpha\Delta}{\rho(\lambda_j)}\right) e_i$

#inner iterations is constant!

- Generalization of previous work on inner-outer iterations with single real shift-invert. *Robbé, Sadkane, Spence, SIMAX, 31(1), p.92, (2009)*
- Formally equivalent to block restarted Krylov ideally suited for interior problem- *Krylov eigenvalue strategy using FEAST with inexact system solves, Gavin, Polizzi: NLAA, (2018)*
- Equivalence to Polynomial filtering



IFEAST- w/o factorization- (Brendan Gavin, PhD thesis 2018)

- **Example: Parsec standard Ga41As41H72,** n=268K, m=10 lowest,m_0=20,nc=3

IFEAST- w/o factorization- (Brendan Gavin, PhD thesis 2018)

- **Example: Parsec standard Ga41As41H72,** n=268K, m=10 lowest,m_0=20,nc=3

IFEAST- w/o factorization- (Brendan Gavin, PhD thesis 2018)

- **Example: Parsec standard Ga41As41H72,** n=268K, m=10 lowest,m_0=20,nc=3
  - FEAST/PARDISO: ~few hours

IFEAST- w/o factorization- (Brendan Gavin, PhD thesis 2018)

- **Example: Parsec standard Ga41As41H72,** n=268K, m=10 lowest,m_0=20,nc=3
  - FEAST/PARDISO: ~few hours
  - IFEAST/BiCGstab: ~few minutes **~100K mat-vec** (1 rhs), 20 feast iterations

IFEAST- w/o factorization- (Brendan Gavin, PhD thesis 2018)

- **Example: Parsec standard Ga41As41H72,** n=268K, m=10 lowest,m_0=20,nc=3
  - FEAST/PARDISO: ~few hours
  - IFEAST/BiCGstab: ~few minutes **~100K mat-vec** (1 rhs), 20 feast iterations
  - ARPACK: **~12K mat-vec** (1rhs), ~2K restarts,

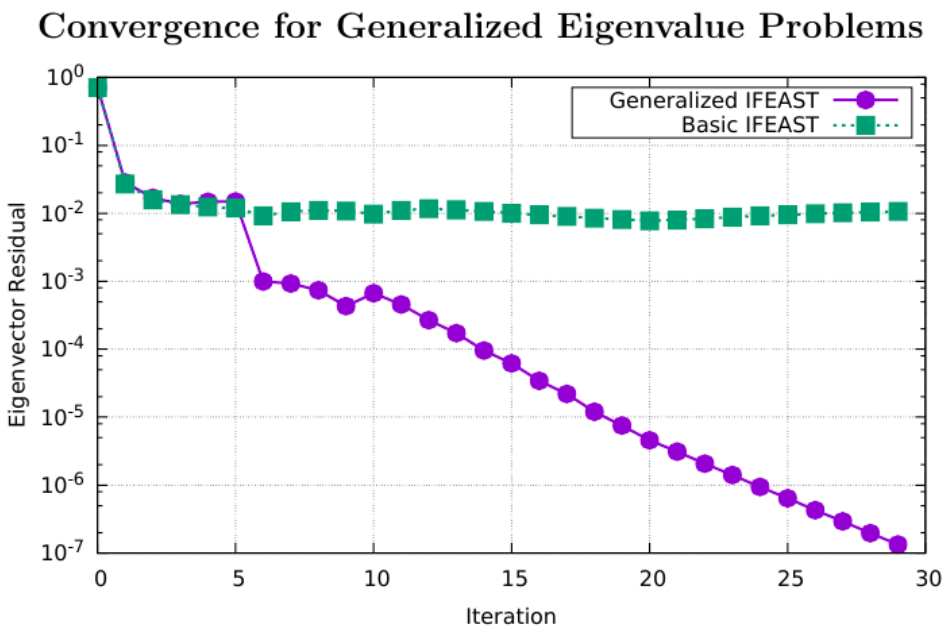
IFEAST- w/o factorization- (Brendan Gavin, PhD thesis 2018)

- **Example: Parsec standard Ga41As41H72,** n=268K, m=10 lowest,m_0=20,nc=3
  - FEAST/PARDISO: ~few hours
  - IFEAST/BiCGstab: ~few minutes **~100K mat-vec** (1 rhs), 20 feast iterations
  - ARPACK: **~12K mat-vec** (1rhs), ~2K restarts,
  - A lot more mat-vec than standard Krylov (Arnoldi) but can be performed in parallel => IFEAST fully in parallel, **1.7K mat-vec** (1rhs)

IFEAST- w/o factorization- (Brendan Gavin, PhD thesis 2018)

- **Example: Parsec standard Ga41As41H72,** n=268K, m=10 lowest, m_0=20,nc=3
  - FEAST/PARDISO: ~few hours
  - IFEAST/BiCGstab: ~few minutes **~100K mat-vec** (1 rhs), 20 feast iterations
  - ARPACK: **~12K mat-vec** (1rhs), ~2K restarts,
  - A lot more mat-vec than standard Krylov (Arnoldi) but can be performed in parallel => IFEAST fully in parallel, **1.7K mat-vec** (1rhs)

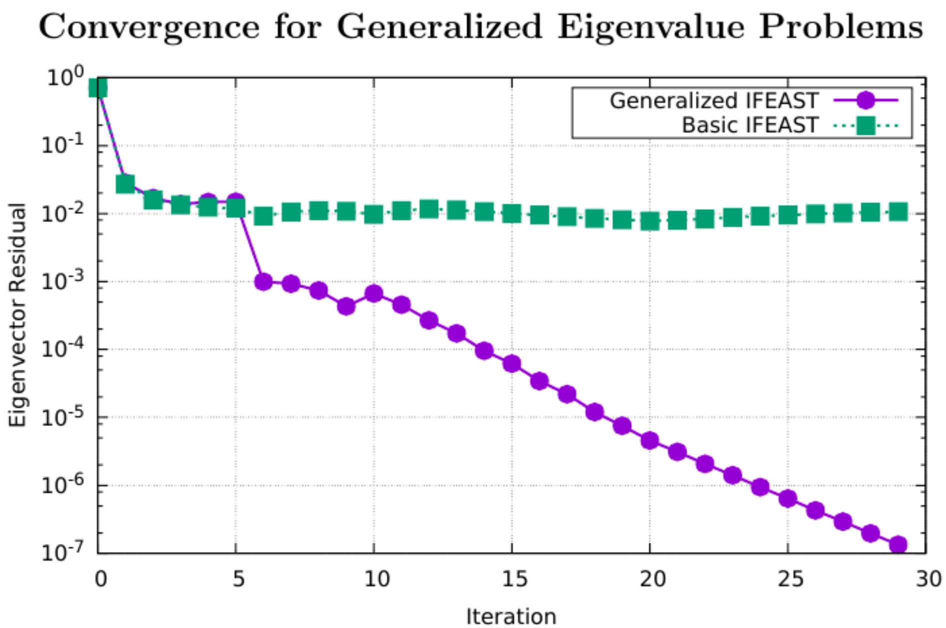
- **Difficulties**: inverse free generalized problems (B≠I) and preconditioners



IFEAST- w/o factorization- (Brendan Gavin, PhD thesis 2018)

- **Example: Parsec standard Ga41As41H72,** n=268K, m=10 lowest,m_0=20,nc=3
  - FEAST/PARDISO: ~few hours
  - IFEAST/BiCGstab: ~few minutes **~100K mat-vec** (1 rhs), 20 feast iterations
  - ARPACK: **~12K mat-vec** (1rhs), ~2K restarts,
  - A lot more mat-vec than standard Krylov (Arnoldi) but can be performed in parallel => IFEAST fully in parallel, **1.7K mat-vec** (1rhs)

- **Difficulties**: inverse free generalized problems (B≠I) and preconditioners



- **Solution:** Generalized IFEAST (based on Residual Inverse Iterations)

Residual Inverse Iterations

- **Generalization of previous work:**

  *Golub G., Ye Q. *Inexact Inverse Iteration for Generalized Eigenvalue Problems,* BIT p671 (2000)

  *See also (in the context of non-linear problems): A. Neumaier, *Residual inverse iteration for the nonlinear eigenvalue problem*, SIAM J. Numer. Anal. 22 (5) (1985)

Residual Inverse Iterations

- **Generalization of previous work:**

  *Golub G., Ye Q. *Inexact Inverse Iteration for Generalized Eigenvalue Problems*, BIT p671 (2000)

  *See also (in the context of non-linear problems): A. Neumaier, *Residual inverse iteration for the nonlinear eigenvalue problem*, SIAM J. Numer. Anal. 22 (5) (1985)

$$\text{Solve } (z_k B - A)Y = BX_k$$

$$\Rightarrow$$

$$R = BX_k\Delta - AX_k$$

$$\text{Solve } (z_k B - A)\gamma = -R$$

$$Y = (X_k + \gamma) * (z_k I - \Delta)^{-1}$$

Residual Inverse Iterations

- **Generalization of previous work:**

  *Golub G., Ye Q. *Inexact Inverse Iteration for Generalized Eigenvalue Problems*, BIT p671 (2000)

  *See also (in the context of non-linear problems): A. Neumaier, *Residual inverse iteration for the nonlinear eigenvalue problem*, SIAM J. Numer. Anal. 22 (5) (1985)

$$\boxed{\text{Solve } (z_k B - A)Y = BX_k} \;\Rightarrow\; \boxed{\begin{array}{c} R = BX_k\Delta - AX_k \\ \text{Solve } \ (z_k B - A)\gamma = -R \\ Y = (X_k + \gamma) * (z_k I - \Delta)^{-1} \end{array}}$$

$$T(z) = zB - A$$

$$Q = \frac{1}{2\pi i}\oint_{\mathcal{C}} dz T(z)^{-1} B X^{(k)} \Rightarrow Q = \frac{1}{2\pi i}\oint_{\mathcal{C}} dz \Big(X^{(k)} - T(z)^{-1} R(X^{(k)}, \Delta)\Big)(zI - \Delta)^{-1}$$

Residual Inverse Iterations

- **Generalization of previous work:**

*Golub G., Ye Q. *Inexact Inverse Iteration for Generalized Eigenvalue Problems*, BIT p671 (2000)

*See also (in the context of non-linear problems): A. Neumaier, *Residual inverse iteration for the nonlinear eigenvalue problem*, SIAM J. Numer. Anal. 22 (5) (1985)

$$\boxed{\text{Solve } (z_k B - A)Y = BX_k} \Rightarrow \boxed{\begin{array}{c} R = BX_k\Delta - AX_k \\ \text{Solve } (z_k B - A)\gamma = -R \\ Y = (X_k + \gamma) * (z_k I - \Delta)^{-1} \end{array}}$$

$$T(z) = zB - A$$

$$Q = \frac{1}{2\pi i}\oint_{\mathcal{C}} dz T(z)^{-1} B X^{(k)} \Rightarrow Q = \frac{1}{2\pi i}\oint_{\mathcal{C}} dz \Big(X^{(k)} - T(z)^{-1} R(X^{(k)}, \Delta)\Big)(zI - \Delta)^{-1}$$

Three main consequences

Residual Inverse Iterations

- **Generalization of previous work:**

  *Golub G., Ye Q. *Inexact Inverse Iteration for Generalized Eigenvalue Problems*, BIT p671 (2000)

  *See also (in the context of non-linear problems): A. Neumaier, *Residual inverse iteration for the nonlinear eigenvalue problem*, SIAM J. Numer. Anal. 22 (5) (1985)

$$\text{Solve } (z_k B - A)Y = BX_k$$

$$R = BX_k\Delta - AX_k$$

$$\text{Solve } (z_k B - A)\gamma = -R$$

$$Y = (X_k + \gamma) * (z_k I - \Delta)^{-1}$$

$$T(z) = zB - A$$

$$Q = \frac{1}{2\pi i}\oint_{\mathcal{C}} dz T(z)^{-1} B X^{(k)} \quad \Rightarrow \quad Q = \frac{1}{2\pi i}\oint_{\mathcal{C}} dz \Big(X^{(k)} - T(z)^{-1} R(X^{(k)}, \Delta)\Big)(zI - \Delta)^{-1}$$

Three main consequences

•IFEAST applicable to generalized systems and preconditioners

Residual Inverse Iterations

- **Generalization of previous work:**

*Golub G., Ye Q. *Inexact Inverse Iteration for Generalized Eigenvalue Problems*, BIT p671 (2000)

*See also (in the context of non-linear problems): A. Neumaier, *Residual inverse iteration for the nonlinear eigenvalue problem*, SIAM J. Numer. Anal. 22 (5) (1985)

$$\boxed{\text{Solve } (z_k B - A)Y = BX_k} \Longrightarrow \boxed{\begin{array}{c} R = BX_k\Delta - AX_k \\ \text{Solve } (z_k B - A)\gamma = -R \\ Y = (X_k + \gamma) * (z_k I - \Delta)^{-1} \end{array}}$$

$$T(z) = zB - A$$

$$Q = \frac{1}{2\pi i}\oint_{\mathcal{C}} dz T(z)^{-1} B X^{(k)} \Longrightarrow Q = \frac{1}{2\pi i}\oint_{\mathcal{C}} dz \Big(X^{(k)} - T(z)^{-1} R(X^{(k)}, \Delta)\Big)(zI - \Delta)^{-1}$$

Three main consequences

•IFEAST applicable to generalized systems and preconditioners

•Mixed-precision arithmetic (single precision direct/iterative solvers)

Residual Inverse Iterations

- **Generalization of previous work:**

*Golub G., Ye Q. *Inexact Inverse Iteration for Generalized Eigenvalue Problems*, BIT p671 (2000)
*See also (in the context of non-linear problems): A. Neumaier, *Residual inverse iteration for the nonlinear eigenvalue problem*, SIAM J. Numer. Anal. 22 (5) (1985)

$$\text{Solve } (z_k B - A)Y = BX_k$$

$$\Rightarrow$$

$$R = BX_k\Delta - AX_k$$

$$\text{Solve } (z_k B - A)\gamma = -R$$

$$Y = (X_k + \gamma) * (z_k I - \Delta)^{-1}$$

$$T(z) = zB - A$$

$$Q = \frac{1}{2\pi i}\oint_{\mathcal{C}} dz T(z)^{-1} B X^{(k)} \Rightarrow Q = \frac{1}{2\pi i}\oint_{\mathcal{C}} dz \Big(X^{(k)} - T(z)^{-1} R(X^{(k)}, \Delta)\Big)(zI - \Delta)^{-1}$$

Three main consequences

- IFEAST applicable to generalized systems and preconditioners
- Mixed-precision arithmetic (single precision direct/iterative solvers)
- Applicable to non-linear eigenvalue problem $T(\lambda)x = 0$

Residual Inverse Iterations: Applications (Generalized+mixed)

Example: C6H6 (P2-FEM generalized), n=49K, m=6 lowest, m_0=20 n_c=5

Solver precision	**FEAST (pardiso)**	• **IFEAST** • **(bicgstab 30 iter. max, jacobi prec.)**
double	7.94s (3 iter.)	51s (10 iter.)
single	5.18s (3 iter.)	33s (10 iter.)

Residual Inverse Iterations: Application to non-linear problem

0. Guess V (could be random)

1. Solve reduced eigenvalue problem

$$V^H T(\lambda) V x_v = 0$$

← reduced non-linear problem (reduced companion problem for polynomial eigenvalue)

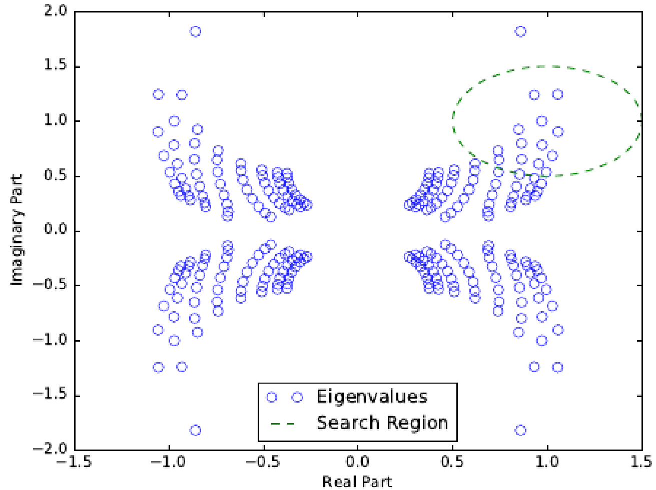
2. $X = VX_v$

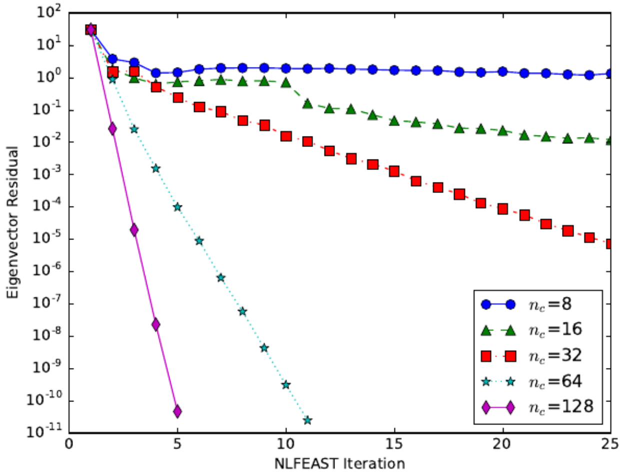
3. Check $\|T(\lambda)x\|$, stop if low enough

4. Generate subspace $V(x) = \sum_{k=1}^{n_c} \omega_k (x - T(z_k)^{-1} T(\lambda) x)(z_k - \lambda)^{-1}$

5. GOTO **step 1**

Example: Butterfly problem $P(\lambda)x = (\lambda^4 A_4 + \lambda^3 A_3 + \lambda^2 A_2 + \lambda A_1 + A_0)x = 0.$





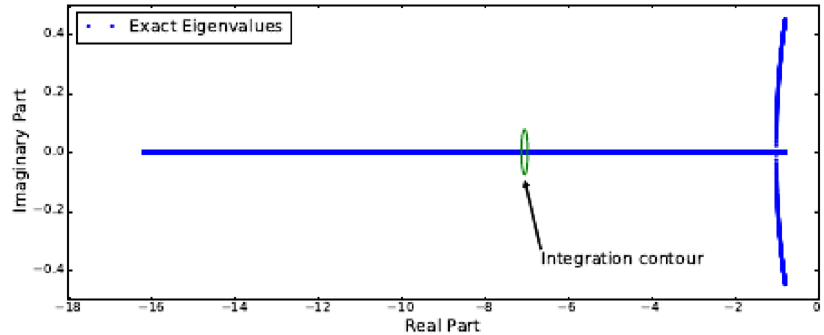
FEAST for nonlinear eigenvalue problems, Gavin, Miedlar, Polizzi, *JCS (2018)*

FEAST non-linear (FEAST and Beyn)

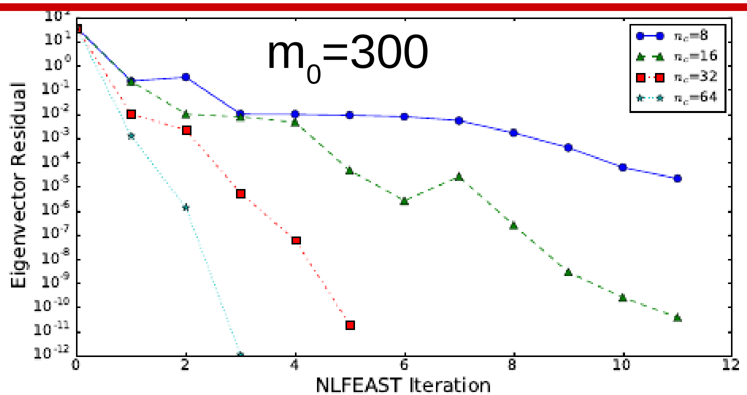
Rail Track Oscillations

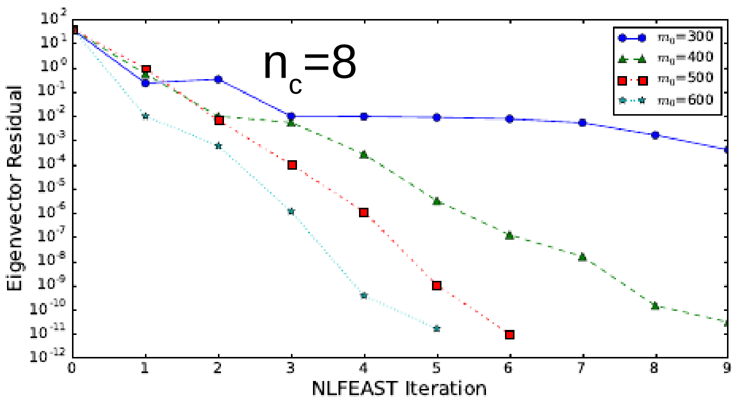
$$T(\lambda) = \lambda^2 I + \lambda(I + A^2) + A^2 + A + I$$

n=50K, m=250



FEAST



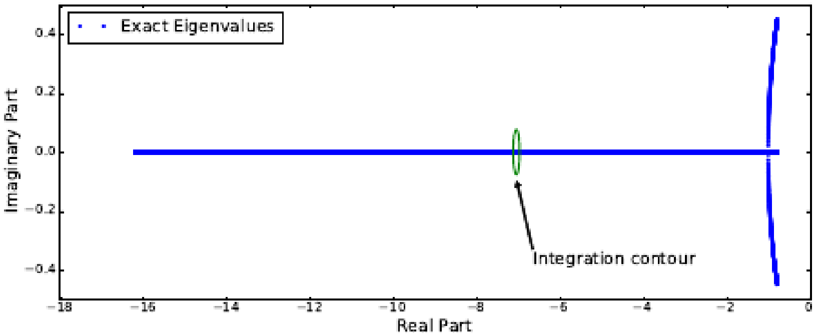


FEAST non-linear (FEAST and Beyn)

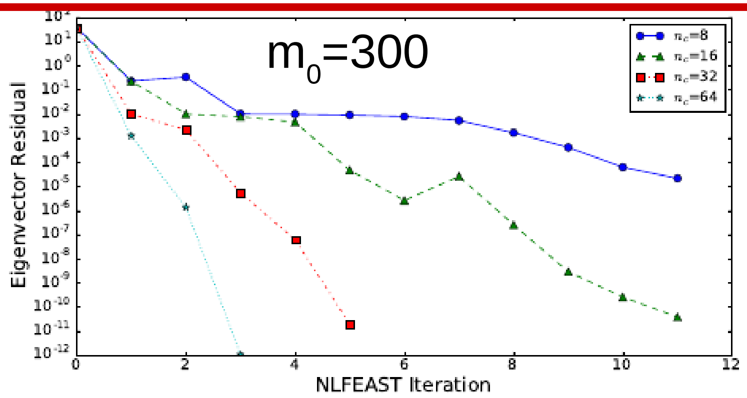
Rail Track Oscillations

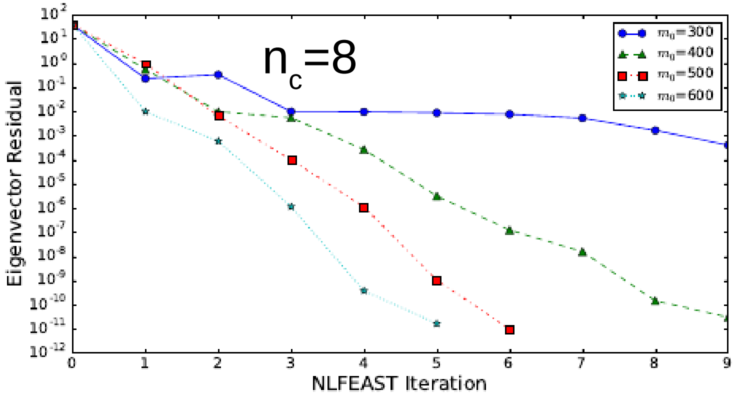
$$T(\lambda) = \lambda^2 I + \lambda(I + A^2) + A^2 + A + I$$

n=50K, m=250



FEAST





Beyn's Method Residuals for Rail Oscillations

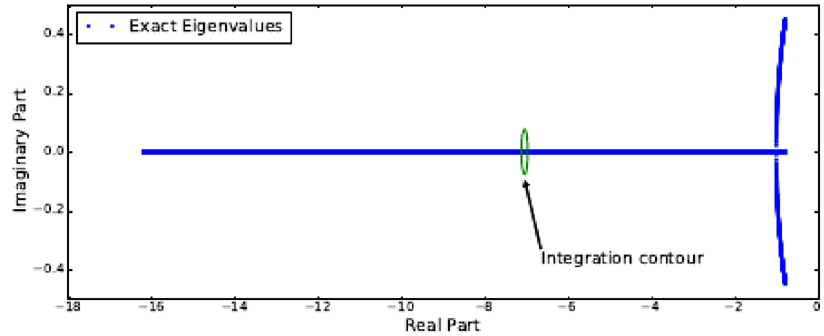
	$n_c = 8$	$n_c = 32$	$n_c = 64$	$n_c = 128$
$m_0 = 300$	2.4e-1	1.0e-2	3.8e-3	4.2e-9
$m_0 = 500$	9.4e-1	3.8e-8	2.5e-12	6.9e-12

FEAST non-linear (FEAST and Beyn)

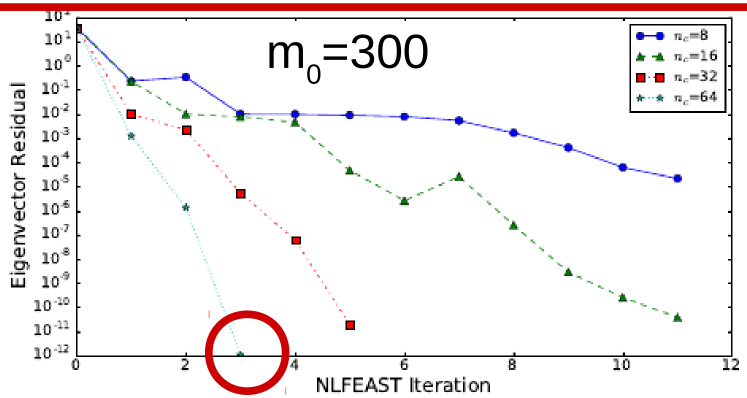
Rail Track Oscillations

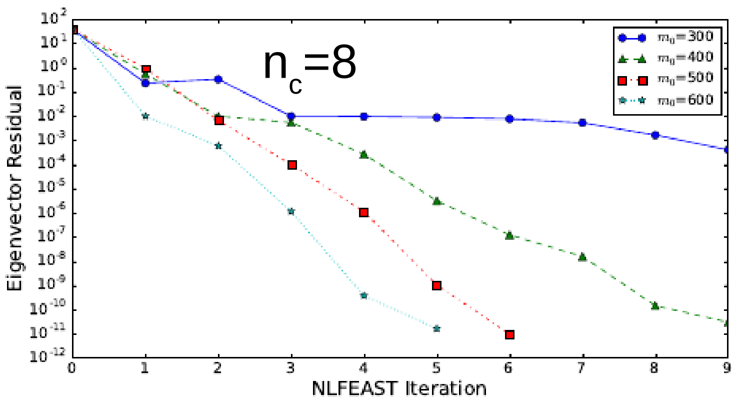
$$T(\lambda) = \lambda^2 I + \lambda(I + A^2) + A^2 + A + I$$

n=50K, m=250



FEAST





Beyn's Method Residuals for Rail Oscillations

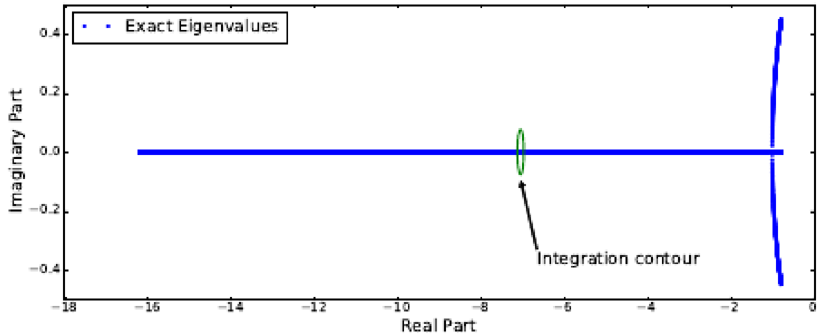
	$n_c = 8$	$n_c = 32$	$n_c = 64$	$n_c = 128$
$m_0 = 300$	2.4e-1	1.0e-2	3.8e-3	4.2e-9
$m_0 = 500$	9.4e-1	3.8e-8	2.5e-12	6.9e-12

FEAST non-linear (FEAST and Beyn)

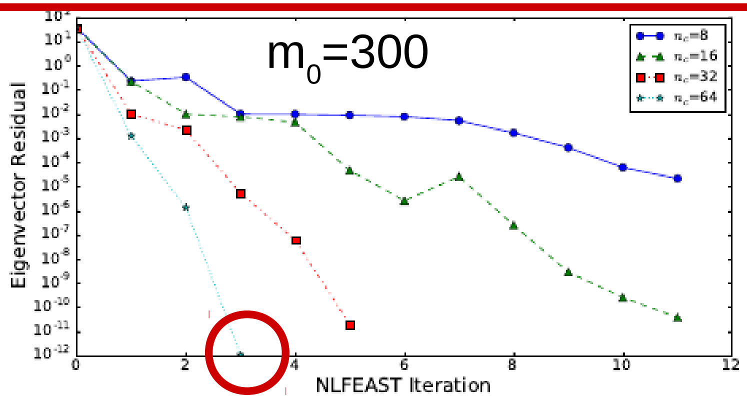
Rail Track Oscillations

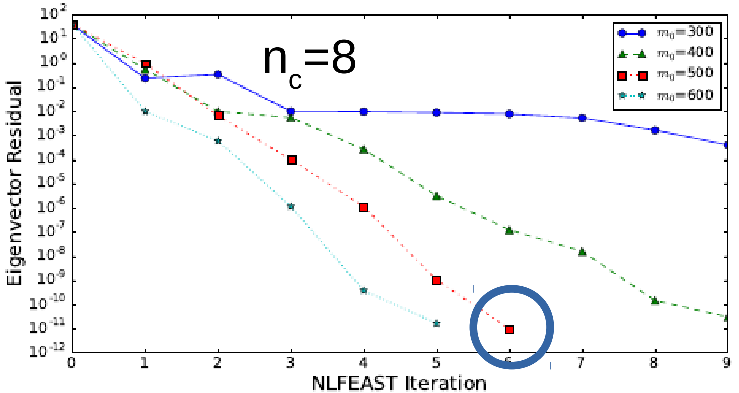
$$T(\lambda) = \lambda^2 I + \lambda(I + A^2) + A^2 + A + I$$

n=50K, m=250



FEAST





Beyn's Method Residuals for Rail Oscillations

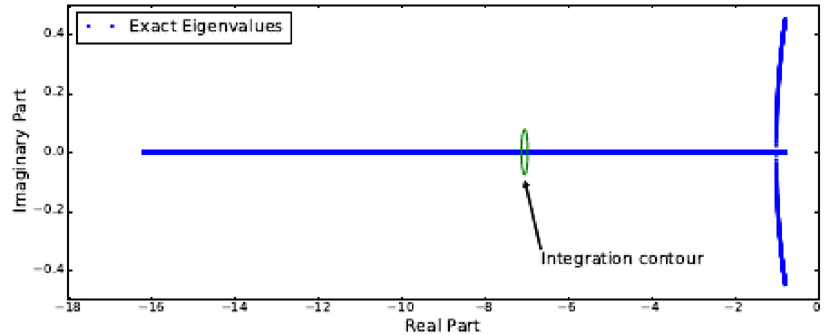
	$n_c = 8$	$n_c = 32$	$n_c = 64$	$n_c = 128$
$m_0 = 300$	2.4e-1	1.0e-2	3.8e-3	4.2e-9
$m_0 = 500$	9.4e-1	3.8e-8	2.5e-12	6.9e-12

FEAST non-linear (FEAST and Beyn)

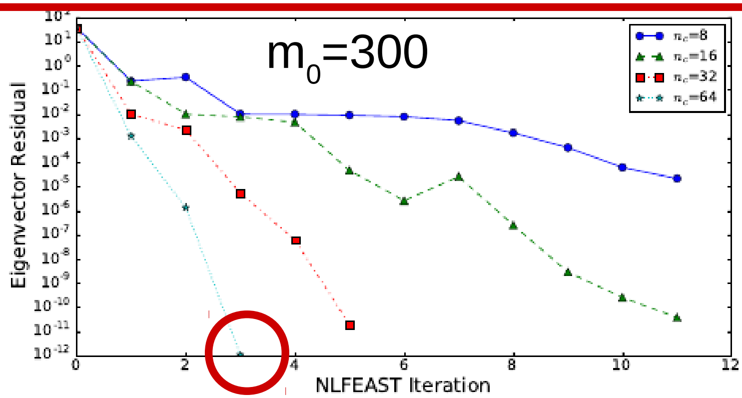
Rail Track Oscillations

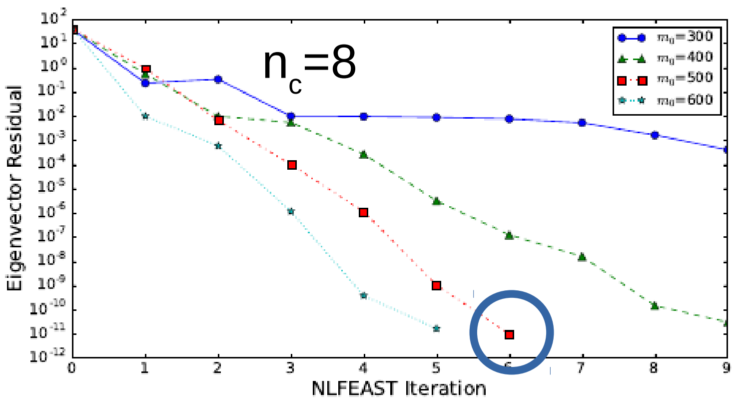
$$T(\lambda) = \lambda^2 I + \lambda(I + A^2) + A^2 + A + I$$

n=50K, m=250



FEAST





Beyn's Method Residuals for Rail Oscillations

	$n_c = 8$	$n_c = 32$	$n_c = 64$	$n_c = 128$
$m_0 = 300$	2.4e-1	1.0e-2	3.8e-3	4.2e-9
$m_0 = 500$	9.4e-1	3.8e-8	2.5e-12	6.9e-12

<u>FEAST proposed approach:</u> solve the projected non-linear reduced system

$$Q^H T(\lambda) Q y = 0$$

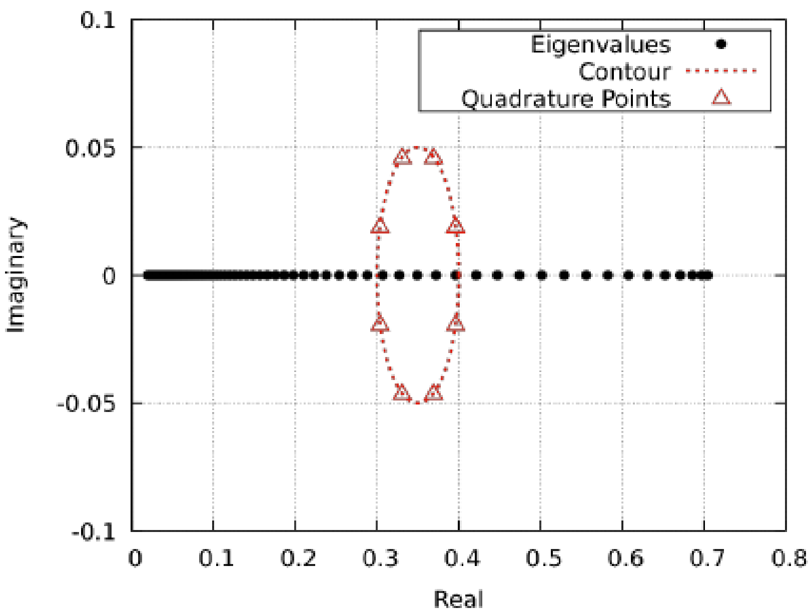
(i) using companion problem for reduced system, or

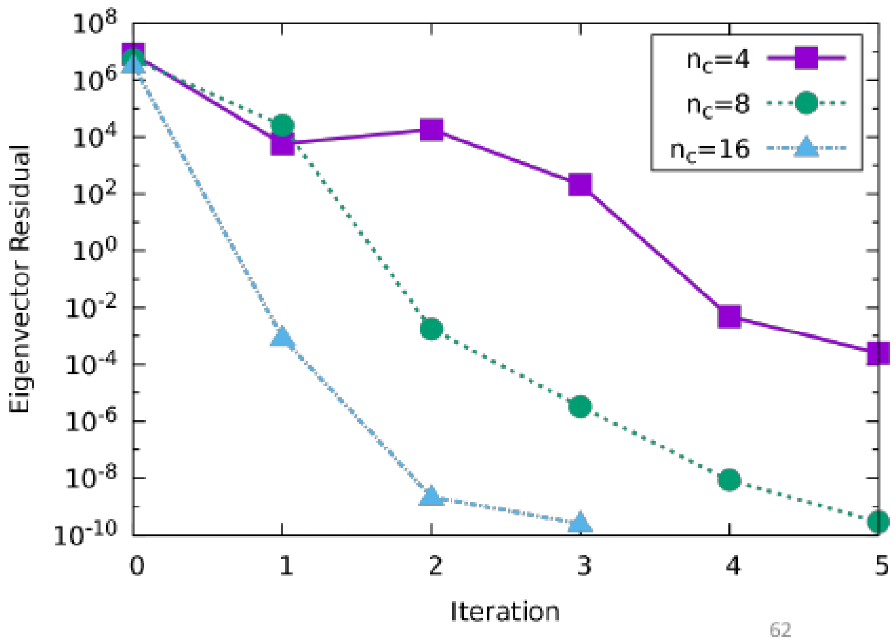
(ii) using Beyn's method (beyond v4.0)

FEAST non-linear (FEAST using Beyn)

Example: general nonlinear

$$T(\lambda) = \lambda^2 B_2 + (e^{\lambda} - 1)B_1 + B_0$$





Conclusion

<u>FEAST v4.0</u>
New implementation using Residual Inverse Iterations
PFEAST (MPI-MPI-MPI)
IFEAST (w/o factorization+basic preconditioners)
All linear system solves using single precisions
Non-linear problems (polynomial)

<u>New Direction (beyond 4.0)</u>: Hybrid solvers, svd, quaternions

Students: James Kestyn, Brendan Gavin, Braegan Spring, Julien Brenneck
Collaborators: Y. Saad, A. Miedlar, P. Tang
Funding: NSF #1510010, #1739423, #1813480, Intel