Residual Inverse Formulation of the FEAST Eigenvalue Algorithm Using Mixed-Precision and Inexact System Solves

Ivan Williams, Eric Polizzi

Department of Electrical and Computer Engineering

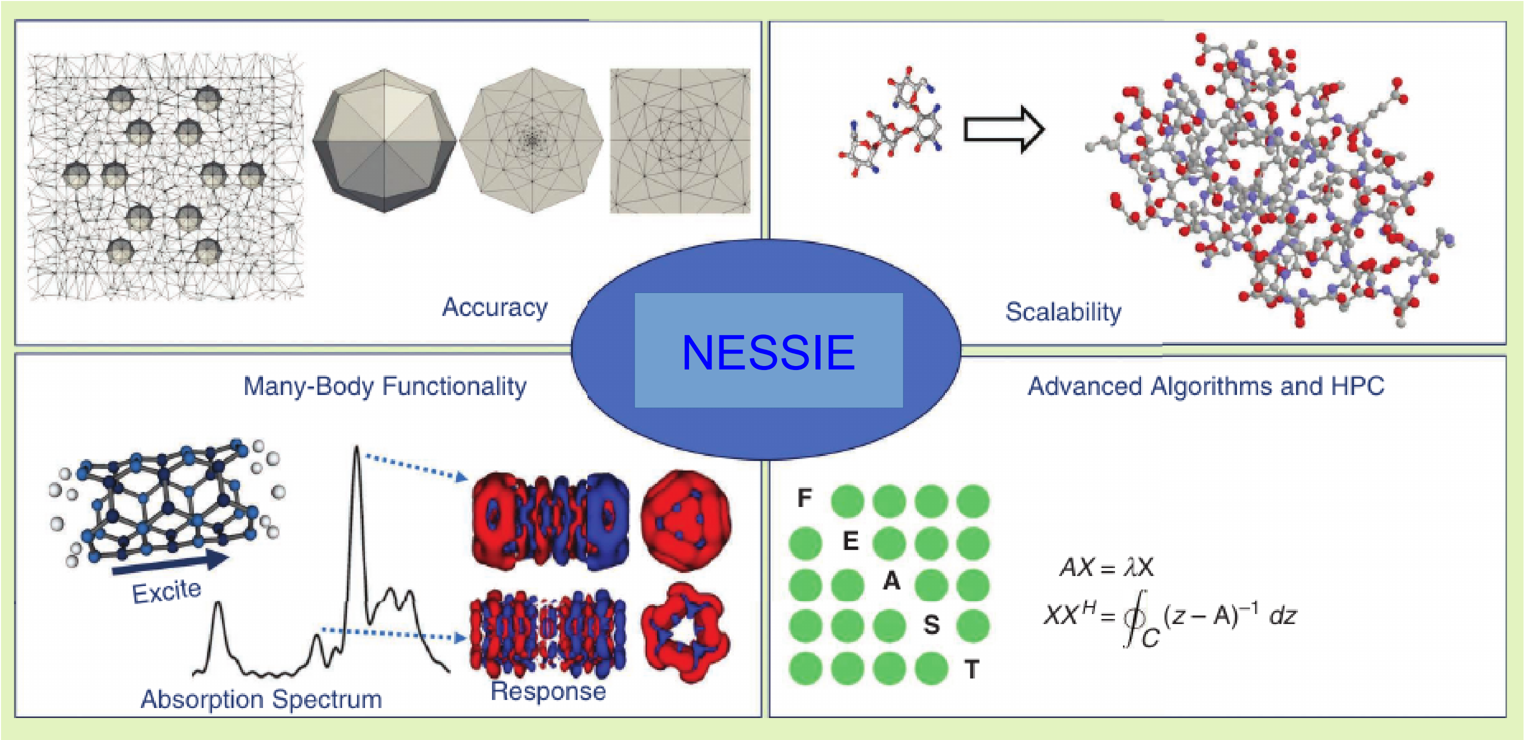
University of Massachusetts, Amherst

SIAM-PP 2024

Advancements in Sparse Linear Algebra: Hardware-Aware Algorithms and Optimization Techniques

March 7, 2024

NESSIE First-Principle Simulation Framework



Objectives:

- Nanoelectronics from the bottom-up
- Large-scale simulations targeted to nanoengineering applications
- *Predict "many-body-effects" operating principle of emerging devices (plasmonic, etc.)*
- *Operate the full range of electronic spectroscopy: UV-ViS, X-ray, and near IR*

www.nessie-code.org

From Fundamental First-Principle Calculations to Nanoengineering Applications: A review of the NESSIE project, J. Kestyn, E. Polizzi, IEEE nano magazine (Dec. 2020)

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

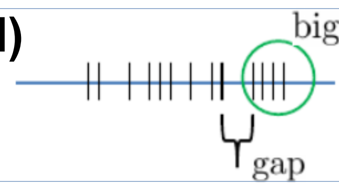
FEAST Algorithm- AX=BXΛ (Hermitian, Generalized)

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 $A_Q = Q^H_{m_0} A Q_{m_0}$ and $B_Q = Q^H_{m_0} B Q_{m_0}$
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)

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



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

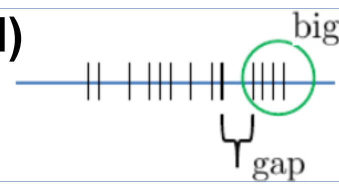
FEAST Algorithm- AX=BXΛ (Hermitian, Generalized)

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)

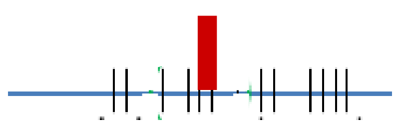
$$\rho(B^{-1}A) = B^{-1}A$$



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

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

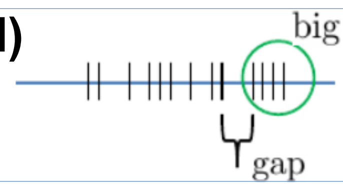
FEAST Algorithm- AX=BXΛ (Hermitian, Generalized)

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 $A_Q = Q^H_{m_0} A Q_{m_0}$ and $B_Q = Q^H_{m_0} B Q_{m_0}$
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)

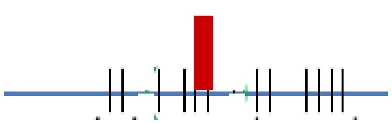
$$\rho(B^{-1}A) = B^{-1}A$$



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

Shift-invert iteration

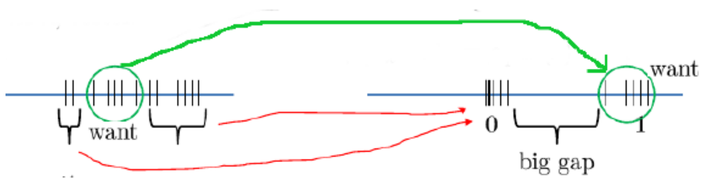
$$\rho(B^{-1}A) = (\sigma B - A)^{-1} B$$



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

Optimal filter: spectral projector

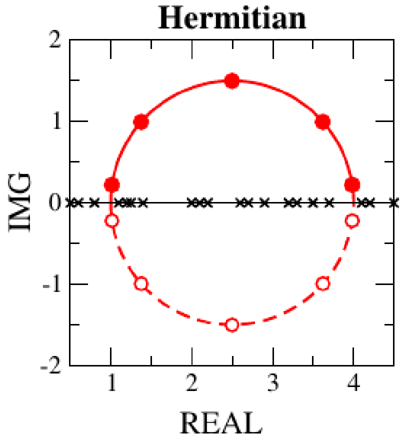
$$\rho(B^{-1}A) = X_m X_m^H B = \frac{1}{2\pi\imath} \oint_{\mathcal{C}} dz (zB - A)^{-1} B$$



FEAST Algorithm: Numerical Quadrature

Rational function filter

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



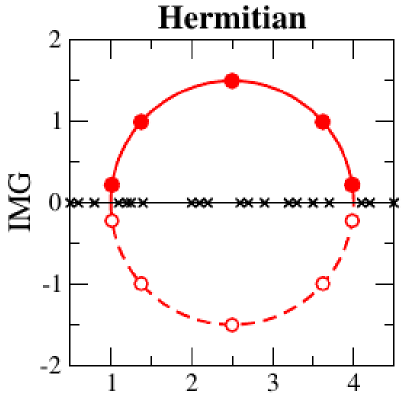
Solving independent linear systems (multiple shifts in complex plane)

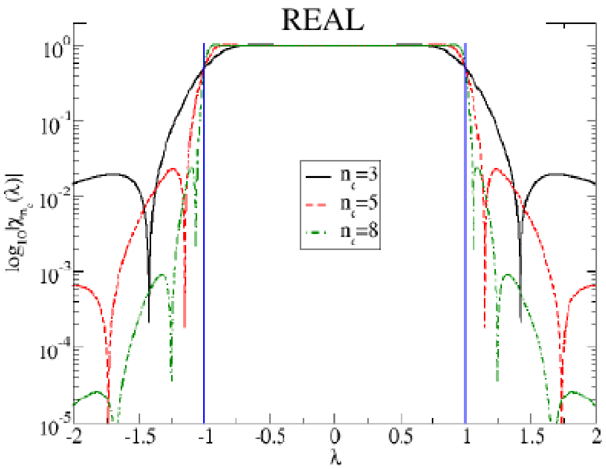
$$Q_{m_0} = \sum_{j=1}^{n_e} \omega_j Q_{m_0}^{(j)} \qquad (z_j B - A) Q_{m_0}^{(j)} = BY_{m_0}$$

FEAST Algorithm: Numerical Quadrature

Rational function filter

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





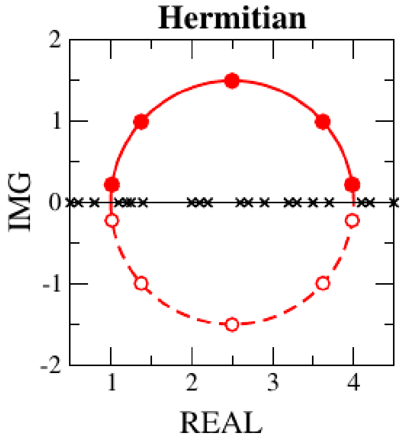
Solving independent linear systems (multiple shifts in complex plane)

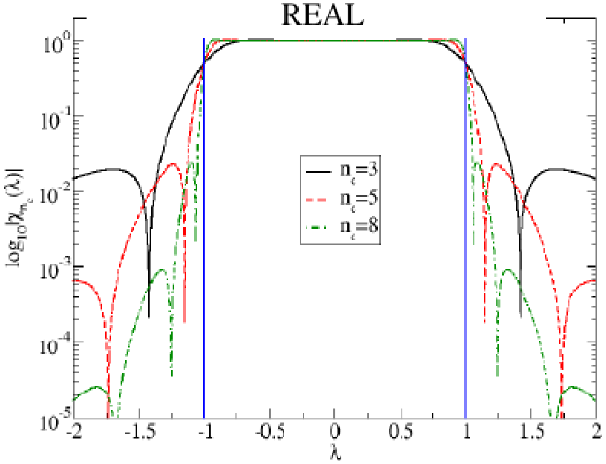
$$Q_{m_0} = \sum_{j=1}^{n_e} \omega_j Q_{m_0}^{(j)} \qquad (z_j B - A) Q_{m_0}^{(j)} = B Y_{m_0}$$

FEAST Algorithm: Numerical Quadrature

Rational function filter

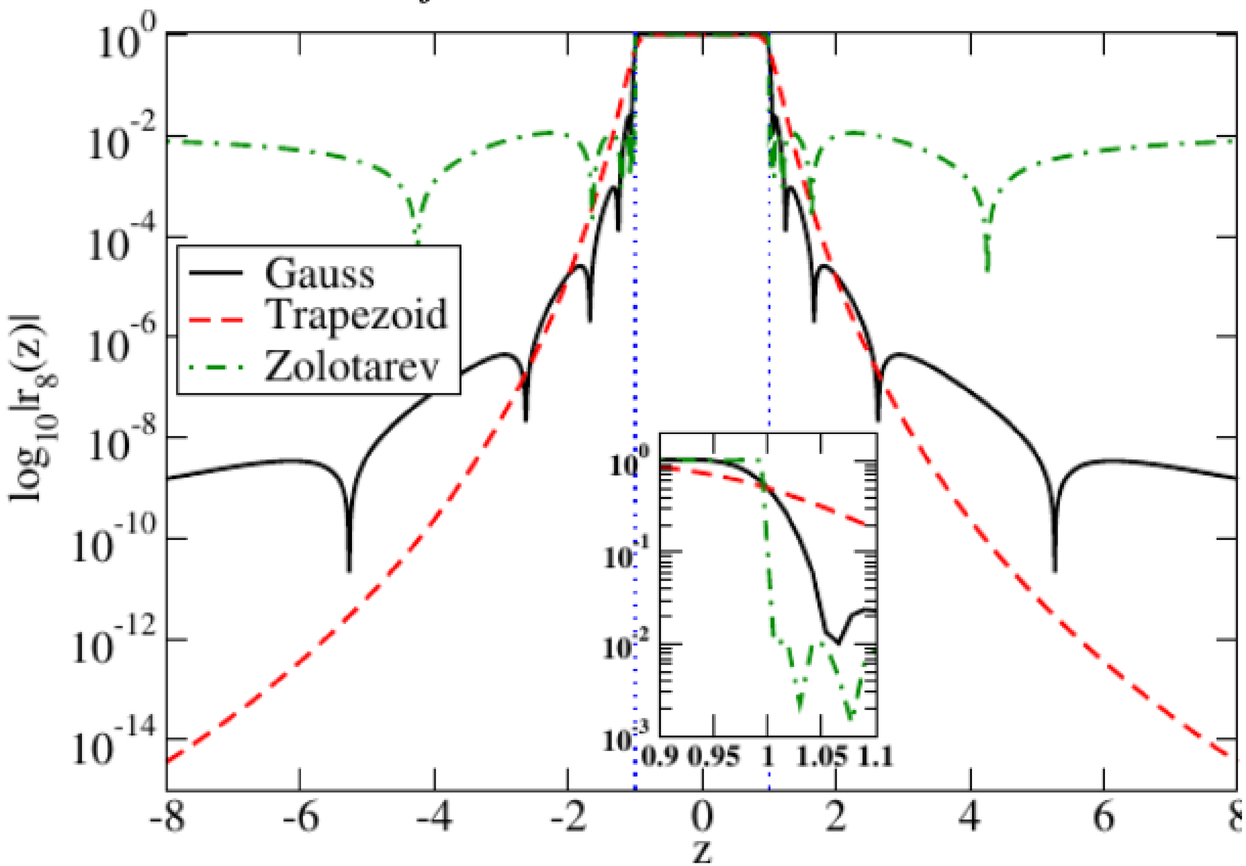
$$\rho_a(z) = \sum_{j=1}^{n_e} \frac{\omega_j}{z_j - z}$$





Solving independent linear systems (multiple shifts in complex plane)

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



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

FEAST non-Hermitian algorithm

$$\boxed{\begin{array}{cc} AX = BX\Lambda & \\ & \widehat{X}^H BX = I \\ A^H\widehat{X} = B^H\widehat{X}\Lambda^* & \end{array}}$$

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

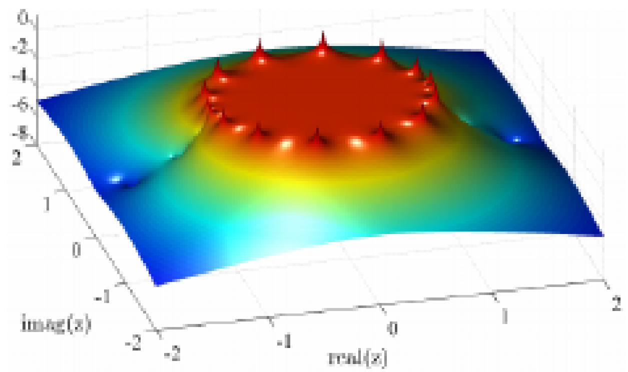
- Right projector

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

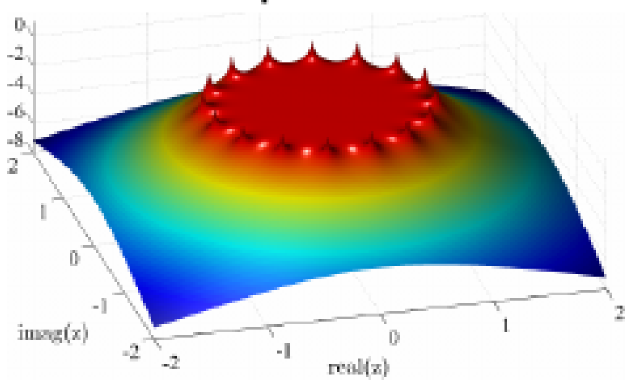
- Left projector

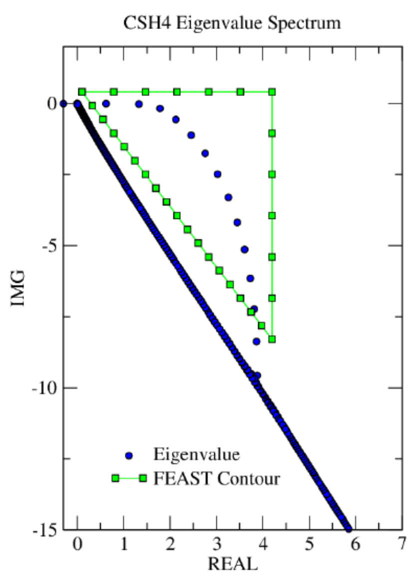
$$\rho(AB^{-1}) = \frac{1}{2\pi\imath}\oint_{\mathcal{C}} dzB(zB - A)^{-1} \equiv BX_m\widehat{X}_m^H$$

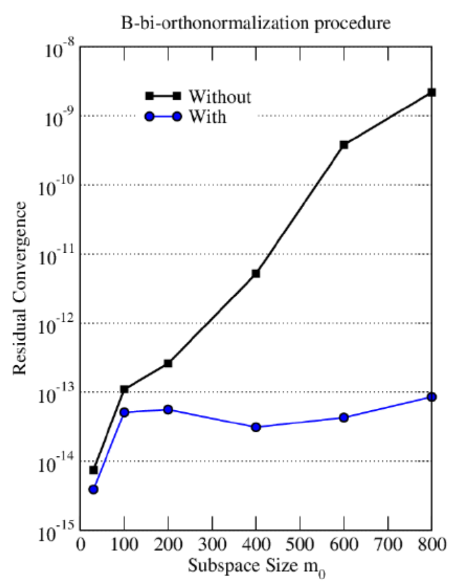
Gauss



Trapezoidal







FEAST Solver: Algorithm + Library

Robust, parallel and unified framework for solving various eigenvalue problems

Hermitian

- *Phys. Rev. B. Vol. 79, 115112 (2009)- Polizzi*
- *SIAM Journal on Matrix Analysis and Applications (SIMAX), 35, 354-390 (2014)- Tang, Polizzi*
- *SIAM Journal on Scientific Computing (SISC), 37 (4), pp2100-2122 (2015)- Guttel, Polizzi, Tang, Viaud*

non-Hermitian

- *SIAM Journal on Scientific Computing (SISC), 38-5, ppS772-S799 (2016) Kestyn, Polizzi, Tang*

Non-Linear Eigenvalue

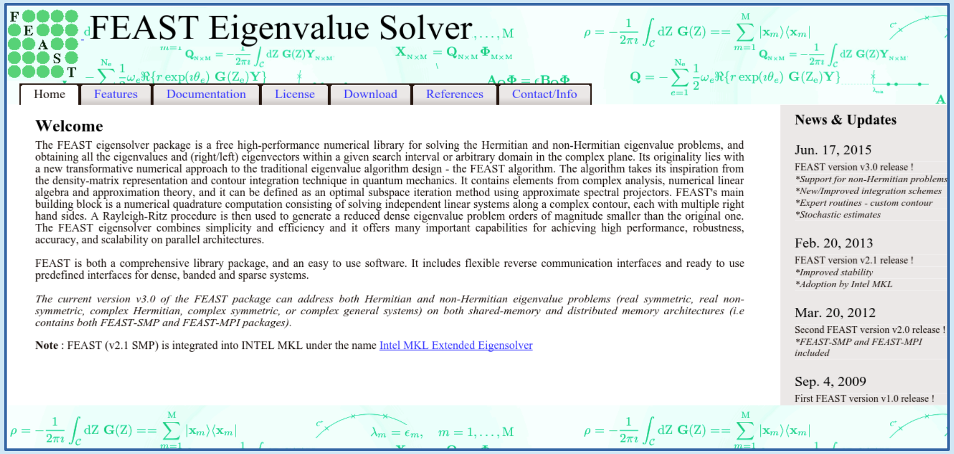
- *Journal of Computational Science, V. 27, 107, (2018), Gavin, Miedlar, Polizzi*
- *arxiv.org/abs/2007.03000, Breneck, Polizzi*

Non-Linear Eigenvector

- *J. Chem. Phys. 138, 194101 (2013), Gavin, Polizzi*

Release dates

www.feast-solver.org



FEAST Solver: Algorithm + Library

Robust, parallel and unified framework for solving various eigenvalue problems

Hermitian

- *Phys. Rev. B. Vol. 79, 115112 (2009)- Polizzi*
- *SIAM Journal on Matrix Analysis and Applications (SIMAX), 35, 354-390 (2014)- Tang, Polizzi*
- *SIAM Journal on Scientific Computing (SISC), 37 (4), pp2100-2122 (2015)- Guttel, Polizzi, Tang, Viaud*

non-Hermitian

- *SIAM Journal on Scientific Computing (SISC), 38-5, ppS772-S799 (2016) Kestyn, Polizzi, Tang*

Non-Linear Eigenvalue

- *Journal of Computational Science, V. 27, 107, (2018), Gavin, Miedlar, Polizzi*
- *arxiv.org/abs/2007.03000, Breneck, Polizzi*

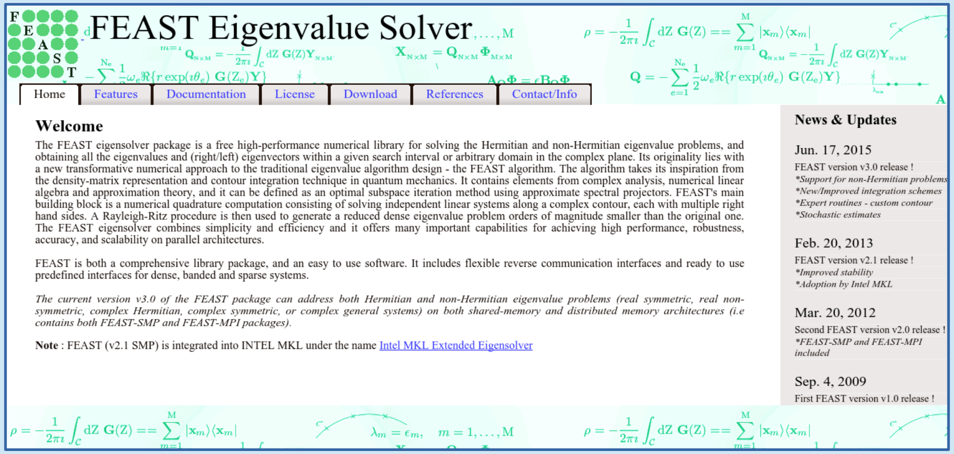
Non-Linear Eigenvector

- *J. Chem. Phys. 138, 194101 (2013), Gavin, Polizzi*

Release dates

- v1.0 (2009): Hermitian problem
- v2.0 (2012): SMP+MPI+RCI interfaces
- v2.1 (2013): **Adoption by Intel-MKL**

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 Solver: Algorithm + Library

Robust, parallel and unified framework for solving various eigenvalue problems

Hermitian

- *Phys. Rev. B. Vol. 79, 115112 (2009)- Polizzi*
- *SIAM Journal on Matrix Analysis and Applications (SIMAX), 35, 354-390 (2014)- Tang, Polizzi*
- *SIAM Journal on Scientific Computing (SISC), 37 (4), pp2100-2122 (2015)- Guttel, Polizzi, Tang, Viaud*

non-Hermitian

- *SIAM Journal on Scientific Computing (SISC), 38-5, ppS772-S799 (2016) Kestyn, Polizzi, Tang*

Non-Linear Eigenvalue

- *Journal of Computational Science, V. 27, 107, (2018), Gavin, Miedlar, Polizzi*
- *arxiv.org/abs/2007.03000, Breneck, Polizzi*

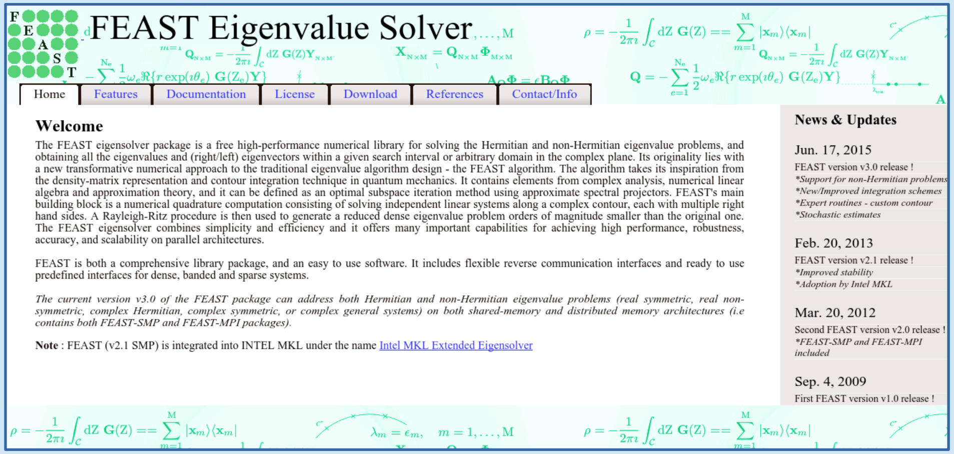
Non-Linear Eigenvector

- *J. Chem. Phys. 138, 194101 (2013), Gavin, Polizzi*

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

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 Solver: Algorithm + Library

Robust, parallel and unified framework for solving various eigenvalue problems

Hermitian

- *Phys. Rev. B. Vol. 79, 115112 (2009)- Polizzi*
- *SIAM Journal on Matrix Analysis and Applications (SIMAX), 35, 354-390 (2014)- Tang, Polizzi*
- *SIAM Journal on Scientific Computing (SISC), 37 (4), pp2100-2122 (2015)- Guttel, Polizzi, Tang, Viaud*

non-Hermitian

- *SIAM Journal on Scientific Computing (SISC), 38-5, ppS772-S799 (2016) Kestyn, Polizzi, Tang*

Non-Linear Eigenvalue

- *Journal of Computational Science, V. 27, 107, (2018), Gavin, Miedlar, Polizzi*
- *arxiv.org/abs/2007.03000, Breneck, Polizzi*

Non-Linear Eigenvector

- *J. Chem. Phys. 138, 194101 (2013), Gavin, Polizzi*

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 (2020): Residual inverse iterations

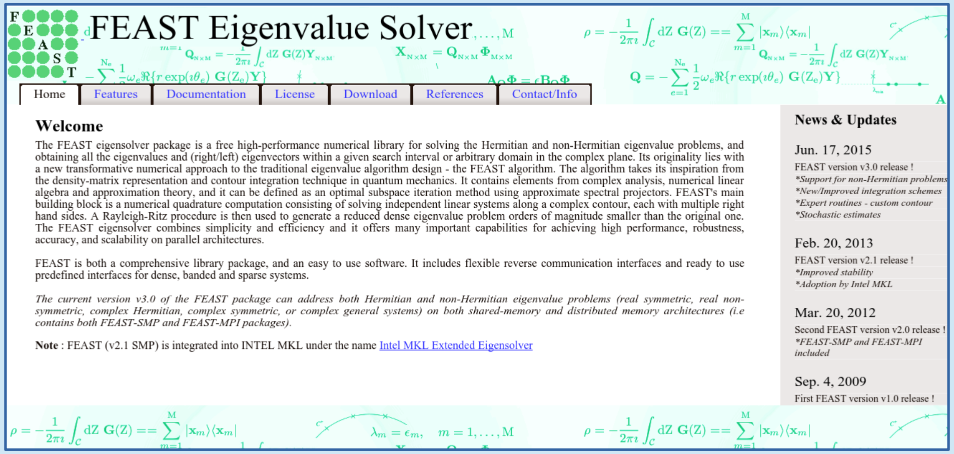
-PFEAST (3 MPI levels)

-IFEAST (FEAST w/o factorization)

-mixed precision

-non-linear (polynomial)

www.feast-solver.org



FEAST Solver: Algorithm + Library

Robust, parallel and unified framework for solving various eigenvalue problems

Hermitian

- *Phys. Rev. B. Vol. 79, 115112 (2009)- Polizzi*
- *SIAM Journal on Matrix Analysis and Applications (SIMAX), 35, 354-390 (2014)- Tang, Polizzi*
- *SIAM Journal on Scientific Computing (SISC), 37 (4), pp2100-2122 (2015)- Guttel, Polizzi, Tang, Viaud*

non-Hermitian

- *SIAM Journal on Scientific Computing (SISC), 38-5, ppS772-S799 (2016) Kestyn, Polizzi, Tang*

Non-Linear Eigenvalue

- *Journal of Computational Science, V. 27, 107, (2018), Gavin, Miedlar, Polizzi*
- *arxiv.org/abs/2007.03000, Breneck, Polizzi*

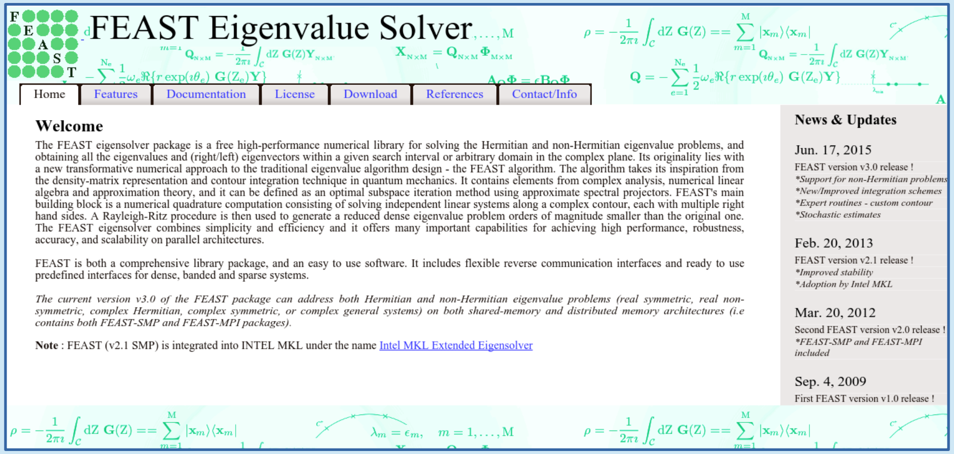
Non-Linear Eigenvector

- *J. Chem. Phys. 138, 194101 (2013), Gavin, Polizzi*

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 (2020): Residual inverse iterations
  - -PFEAST (3 MPI levels)
  - -IFEAST (FEAST w/o factorization)
  - *-mixed precision*
  - *-non-linear (polynomial)*
- v5.0 (2024): *non-linear (general) + hybrid MPI solvers*

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*

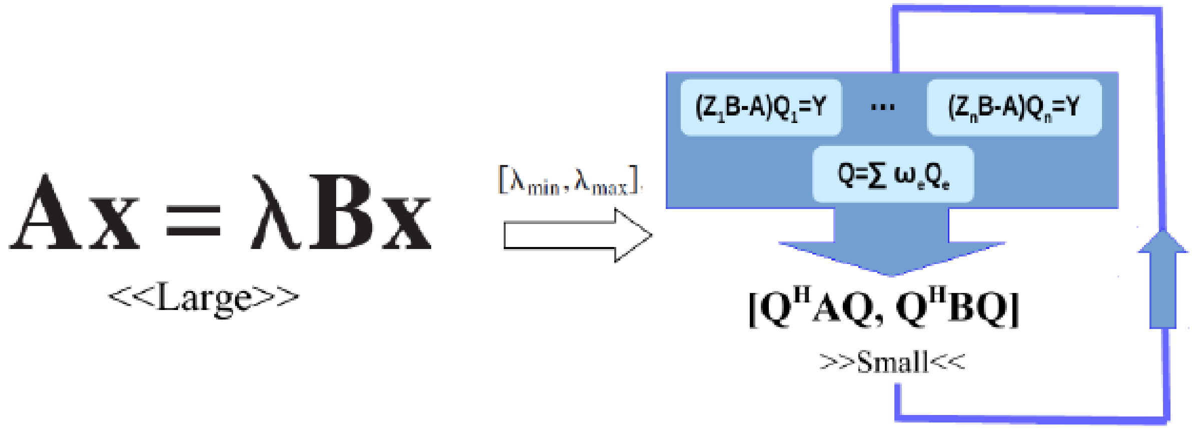
Mar. 20, 2012

Second FEAST version v2.0 release !
**FEAST-SMP and FEAST-MPI included*

Sep. 4, 2009

First FEAST version v1.0 release !

FEAST Algorithm/Solver at a glance



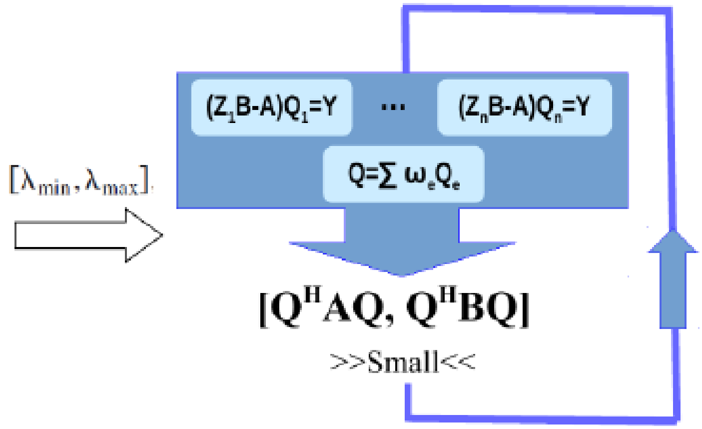
- **Select search interval**
- **Select subspace size M_0**

(good stochastic estimate possible)-
Di Napoli, Saad, Polizzi, NLAA (2015)

FEAST Algorithm/Solver at a glance

$$\mathbf{Ax} = \lambda \mathbf{Bx}$$

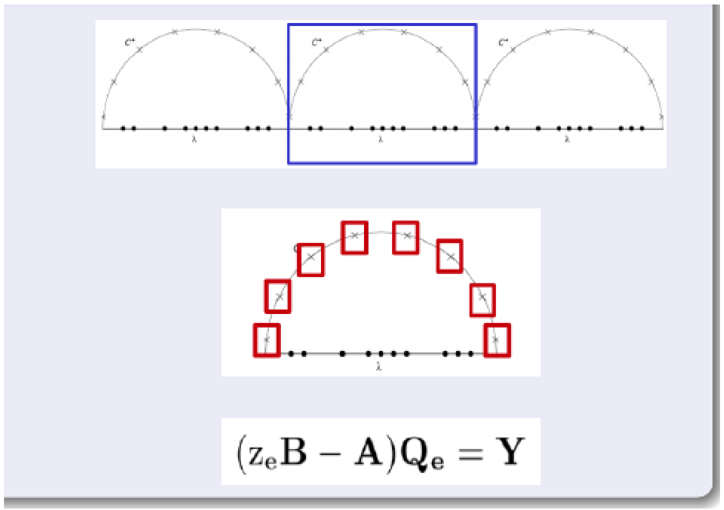
<<Large>>



- **Select search interval**
- **Select subspace size M_0**

(good stochastic estimate possible)-
Di Napoli, Saad, Polizzi, NLAA (2015)

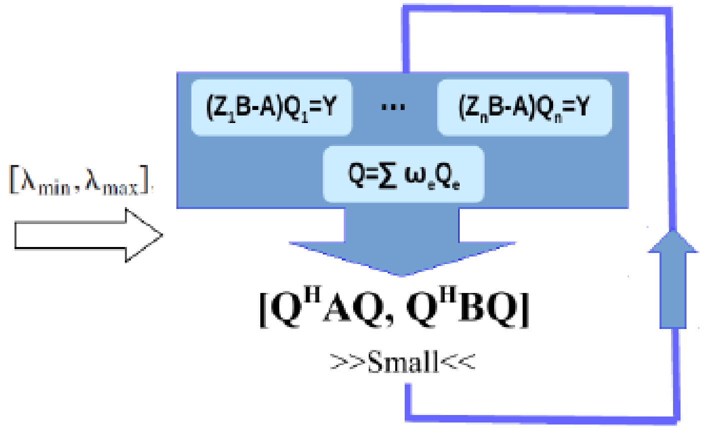
- **Robust and systematic convergence**
- **Standard and Generalized**
- **Hermitian and non-Hermitian**
- **Linear and Non-Linear**
- **3 levels of parallelism**
  - Multiple search intervals
  - Multiple shifts along the contour
  - Linear system solves



FEAST Algorithm/Solver at a glance

$$Ax = \lambda Bx$$

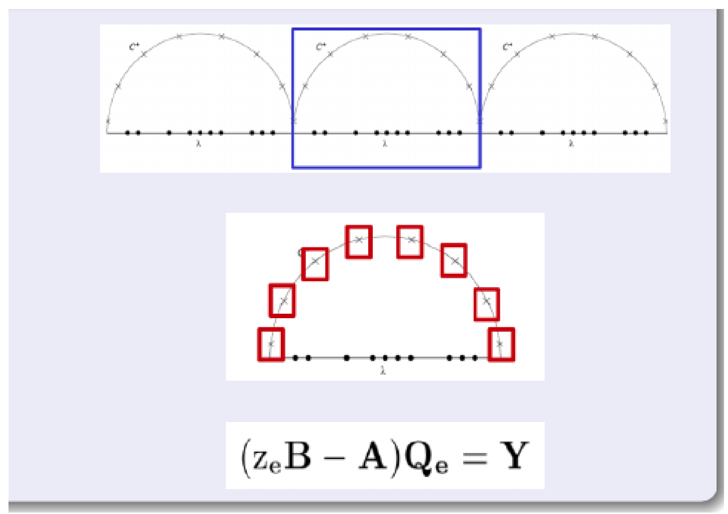
<<Large>>



- **Select search interval**
- **Select subspace size M_0**

(good stochastic estimate possible)-
Di Napoli, Saad, Polizzi, NLAA (2015)

- **Robust and systematic convergence**
- **Standard and Generalized**
- **Hermitian and non-Hermitian**
- **Linear and Non-Linear**
- **3 levels of parallelism**
  - <u>Multiple search intervals</u>
  - <u>Multiple shifts along the contour</u>
  - <u>Linear system solves</u>



- **Interfaces:**
  - <u>Predefined</u>: dense, banded, sparse storage using lapack/spike/pardiso/iterative solvers
  - <u>RCI:</u> independent of matrix format, mat-vec, and system solvers, can be customized by end users
- Used by many third-party Software and Libraries

Difficulties using the basic FEAST algorithm

Basic FEAST

$$Q = \frac{1}{2\pi i} \oint_{\mathcal{C}} T(z)^{-1} B \tilde{X} dz, \qquad T(z) = zB - A$$

Difficulties using the basic FEAST algorithm

Basic FEAST

$$Q = \frac{1}{2\pi i} \oint_{\mathcal{C}} T(z)^{-1} B \tilde{X} dz, \qquad T(z) = zB - A$$

Difficulties using the basic FEAST algorithm

Basic FEAST

$$Q = \frac{1}{2\pi i} \oint_{\mathcal{C}} T(z)^{-1} B \tilde{X} dz, \qquad T(z) = zB - A$$

1- IFEAST (inexact linear system solve using iterative solvers), residual tolerance needs to be gradually increased over the FEAST iterations.

Difficulties using the basic FEAST algorithm

Basic FEAST

$$Q = \frac{1}{2\pi i} \oint_{\mathcal{C}} T(z)^{-1} B \tilde{X} dz, \qquad T(z) = zB - A$$

1- IFEAST (inexact linear system solve using iterative solvers), residual tolerance needs to be gradually increased over the FEAST iterations.
Remark: convergence is still linear (α linear system solve residual)

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

Difficulties using the basic FEAST algorithm

Basic FEAST

$$Q = \frac{1}{2\pi i}\oint_{\mathcal{C}} T(z)^{-1} B\tilde{X} dz, \qquad T(z) = zB - A$$

1- IFEAST (inexact linear system solve using iterative solvers), residual tolerance needs to be gradually increased over the FEAST iterations.
Remark: convergence is still linear (α linear system solve residual)

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

2- basic FEAST **would not work** for non-linear systems (when T(z) is not linear)

Difficulties using the basic FEAST algorithm

Basic FEAST

$$Q = \frac{1}{2\pi i}\oint_{\mathcal{C}} T(z)^{-1} B\tilde{X} dz, \qquad T(z) = zB - A$$

1- IFEAST (inexact linear system solve using iterative solvers), residual tolerance needs to be gradually increased over the FEAST iterations.
Remark: convergence is still linear (α linear system solve residual)

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

2- basic FEAST **would not work** for non-linear systems (when T(z) is not linear)

Solution: Residual Inverse Iterations

$$Q = \frac{1}{2\pi i}\oint_{\mathcal{C}} \left(\tilde{X} - T(z)^{-1} R_E\right) (zI - \tilde{\Lambda})^{-1} dz$$

$$R_E = B\tilde{X}\tilde{\Lambda} - A\tilde{X}$$ FEAST residuals (at a given iteration / linear case)

Residual Inverse Iterations: Application to linear problems

- Generalization of previous works on residual inverse iterations
  - *Golub G., Ye Q. BIT p671 (2000)*
  - *A. Neumaier, SIAM J. Numer. Anal. 22 (5) (1985)*

Residual Inverse Iterations: Application to linear problems

- Generalization of previous works on residual inverse iterations
  - *Golub G., Ye Q. BIT p671 (2000)*
  - *A. Neumaier, SIAM J. Numer. Anal. 22 (5) (1985)*

- Reformulation of the contour integration solve (for the linear case)

Residual Inverse Iterations: Application to linear problems

- Generalization of previous works on residual inverse iterations
  - *Golub G., Ye Q. BIT p671 (2000)*
  - *A. Neumaier, SIAM J. Numer. Anal. 22 (5) (1985)*

- Reformulation of the contour integration solve (for the linear case)

$$\boxed{\text{Solve } (z_j B - A)Y = B\tilde{X}} \Rightarrow \boxed{\begin{array}{c} R_E = B\tilde{X}\tilde{\Lambda} - A\tilde{X} \\ \text{Solve } (z_j B - A)\gamma = -R_E \\ Y = (\tilde{X} + \gamma) * (z_j I - \tilde{\Lambda})^{-1} \end{array}}$$

Residual Inverse Iterations: Application to linear problems

- Generalization of previous works on residual inverse iterations
  - *Golub G., Ye Q. BIT p671 (2000)*
  - *A. Neumaier, SIAM J. Numer. Anal. 22 (5) (1985)*

- Reformulation of the contour integration solve (for the linear case)

$$\text{Solve } (z_j B - A)Y = B\tilde{X}$$

$$\Rightarrow$$

$$R_E = B\tilde{X}\tilde{\Lambda} - A\tilde{X}$$
$$\text{Solve } (z_j B - A)\gamma = -R_E$$
$$Y = (\tilde{X} + \gamma) * (z_j I - \tilde{\Lambda})^{-1}$$

- Theorem 1: The two formulations are equivalent using exact arithmetic

- Theorem 2: Solving $(z_j B - A)\gamma = -R_E$ with tolerance $\frac{||r_\Delta||}{||r_e||} \leq \epsilon$

  is equivalent to solving $(z_j B - A)Y = B\tilde{X}$ with tolerance $\frac{||r_l||}{||B\tilde{x}||} \leq \frac{1}{|z - \tilde{\lambda}|}\frac{||r_e||}{||B\tilde{x}||}\epsilon$
  that is ε below the convergence of the eigenvalue problem

Residual Inverse Iterations: Application to linear problems

$$R_E = B\tilde{X}\tilde{\Lambda} - A\tilde{X}$$

$$\text{Solve } (z_j B - A)\gamma = -R_E$$

$$Y = (\tilde{X} + \gamma) * (z_j I - \tilde{\Lambda})^{-1}$$

Residual Inverse Iterations: Application to linear problems

$$R_E = B\tilde{X}\tilde{\Lambda} - A\tilde{X}$$
$$\text{Solve } (z_j B - A)\gamma = -R_E$$
$$Y = (\tilde{X} + \gamma) * (z_j I - \tilde{\Lambda})^{-1}$$

- Consequences:

Residual Inverse Iterations: Application to linear problems

$$R_E = B\tilde{X}\tilde{\Lambda} - A\tilde{X}$$

$$\text{Solve } (z_j B - A)\gamma = -R_E$$

$$Y = (\tilde{X} + \gamma) * (z_j I - \tilde{\Lambda})^{-1}$$

- Consequences:
  - inexact solves with a fix ε (ε=10^{-1} in FEAST v4.0 by default)

Residual Inverse Iterations: Application to linear problems

$$\boxed{\begin{array}{c} R_E = B\tilde{X}\tilde{\Lambda} - A\tilde{X} \\ \text{Solve } (z_j B - A)\gamma = -R_E \\ Y = (\tilde{X} + \gamma) * (z_j I - \tilde{\Lambda})^{-1} \end{array}}$$

- Consequences:
  - inexact solves with a fix ε ($\varepsilon = 10^{-1}$ in FEAST v4.0 by default)
  - low accuracy (mixed-precision FEAST) for direct or iterative solvers

Residual Inverse Iterations: Application to linear problems

$$\boxed{\begin{array}{c} R_E = B\tilde{X}\tilde{\Lambda} - A\tilde{X} \\ \text{Solve } (z_j B - A)\gamma = -R_E \\ Y = (\tilde{X} + \gamma) * (z_j I - \tilde{\Lambda})^{-1} \end{array}}$$

- Consequences:
  - inexact solves with a fix ε ($\varepsilon=10^{-1}$ in FEAST v4.0 by default)
  - low accuracy (mixed-precision FEAST) for direct or iterative solvers
  - generalization to non-linear systems

Residual Inverse Iterations: Application to linear problems

$$R_E = B\tilde{X}\tilde{\Lambda} - A\tilde{X}$$

$$\text{Solve } (z_j B - A)\gamma = -R_E$$

$$Y = (\tilde{X} + \gamma) * (z_j I - \tilde{\Lambda})^{-1}$$

- Consequences:
  - inexact solves with a fix ε (ε=10^{-1} in FEAST v4.0 by default)
  - low accuracy (mixed-precision FEAST) for direct or iterative solvers
  - generalization to non-linear systems

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	8s (3 iter.)	51s (10 iter.)
single	5s (3 iter.)	33s (10 iter.)

Residual Inverse Iterations:Application to non-linear problems

- Examples of eigenvalue problems (from polynomial to general non-linear)

$$P(\lambda)x = (\lambda^4 A_4 + \lambda^3 A_3 + \lambda^2 A_2 + \lambda A_1 + A_0)x = 0. \qquad T(\lambda) = K - \lambda M + i\sqrt{\lambda - \sigma_1^2}W_1 + i\sqrt{\lambda - \sigma_2^2}W_2$$

Residual Inverse Iterations:Application to non-linear problems

- Examples of eigenvalue problems (from polynomial to general non-linear)

$$P(\lambda)x = (\lambda^4 A_4 + \lambda^3 A_3 + \lambda^2 A_2 + \lambda A_1 + A_0)x = 0. \qquad T(\lambda) = K - \lambda M + i\sqrt{\lambda - \sigma_1^2}W_1 + i\sqrt{\lambda - \sigma_2^2}W_2$$

<u>NLFEAST:</u> residual inverse iteration leads to a non-linear polynomial reduced system

*B. Gavin, A. Miedlar, E. Polizzi, *FEAST eigensolver for nonlinear eigenvalue problems, JCS (2018)*

**B.Gavin, UMass PhD thesis (2018)*

Remark: reduced eigenvalue problem becomes non-linear $Q^H T(\lambda) Q y = 0$
solved using companion problem for polynomial in v4.0

Residual Inverse Iterations:Application to non-linear problems

- Examples of eigenvalue problems (from polynomial to general non-linear)

$$P(\lambda)x = (\lambda^4 A_4 + \lambda^3 A_3 + \lambda^2 A_2 + \lambda A_1 + A_0)x = 0. \qquad T(\lambda) = K - \lambda M + i\sqrt{\lambda - \sigma_1^2}W_1 + i\sqrt{\lambda - \sigma_2^2}W_2$$

NLFEAST: residual inverse iteration leads to a non-linear polynomial reduced system

*B. Gavin, A. Miedlar, E. Polizzi, *FEAST eigensolver for nonlinear eigenvalue problems, JCS (2018)*

**B.Gavin, UMass PhD thesis (2018)*

Remark: reduced eigenvalue problem becomes non-linear $Q^H T(\lambda) Q y = 0$
solved using companion problem for polynomial in v4.0

Beyn's method: recent contour integration technique for general non-linear problem (beyond polynomial)- **non-iterative**

Residual Inverse Iterations: Application to non-linear problems

- Examples of eigenvalue problems (from polynomial to general non-linear)

$$P(\lambda)x = (\lambda^4 A_4 + \lambda^3 A_3 + \lambda^2 A_2 + \lambda A_1 + A_0)x = 0. \qquad T(\lambda) = K - \lambda M + i\sqrt{\lambda - \sigma_1^2}W_1 + i\sqrt{\lambda - \sigma_2^2}W_2$$

NLFEAST: residual inverse iteration leads to a non-linear polynomial reduced system

*B. Gavin, A. Miedlar, E. Polizzi, *FEAST eigensolver for nonlinear eigenvalue problems, JCS (2018)*

**B.Gavin, UMass PhD thesis (2018)*

Remark: reduced eigenvalue problem becomes non-linear $Q^H T(\lambda) Q y = 0$
solved using companion problem for polynomial in v4.0

Beyn's method: recent contour integration technique for general non-linear problem (beyond polynomial)- **non-iterative**

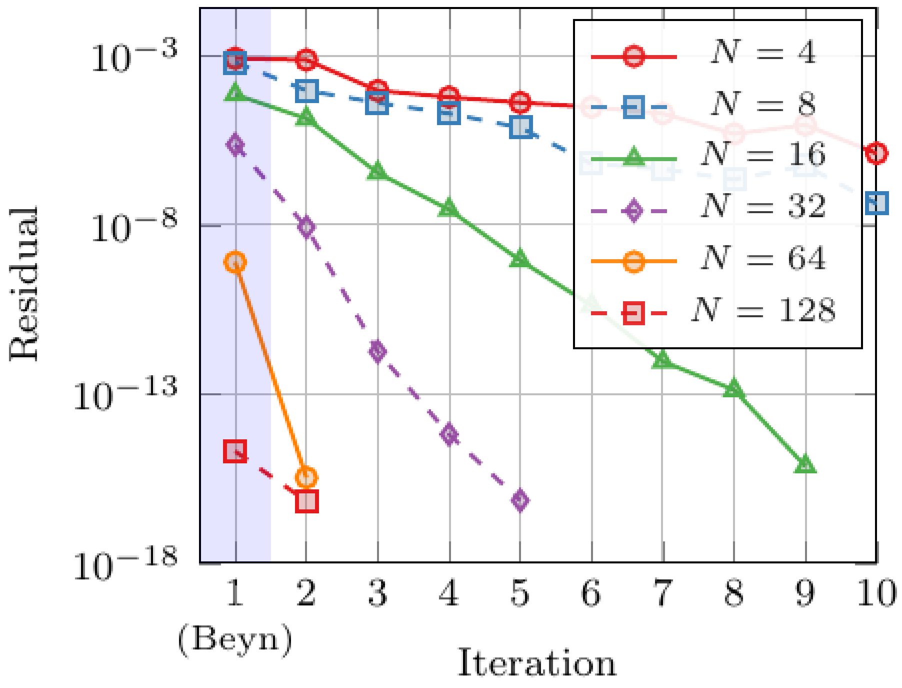
Upcoming NLFEAST v5: new hybrid NLFEAST-Beyn scheme for general non-linear problem- **iterative approach until convergence.**

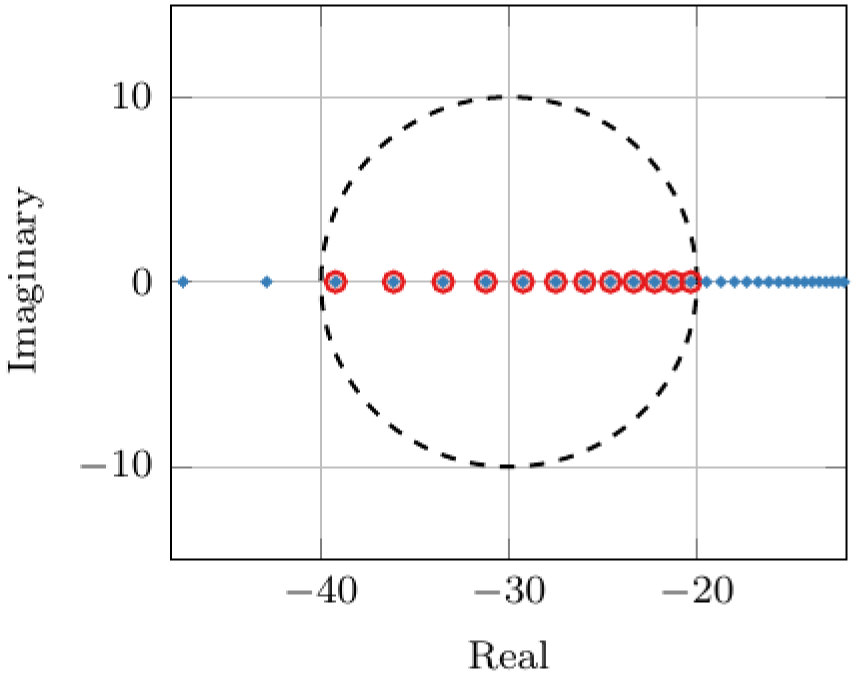
J. Brenneck, E. Polizzi, *An Iterative Method for Contour-Based Nonlinear Eigensolvers, arxiv (2022)*

Residual Inverse Iterations:Application to non-linear problems

Hadeler Problem

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





Conclusion

<u>FEAST v4.0</u>

New implementation using Residual Inverse Iterations

- PFEAST (MPI-MPI-MPI)
- IFEAST (w/o factorization, modest convergence residuals)
- All linear systems are solved inexactly using single precision
- Applicable to non-linear problems (polynomial)

<u>Upcoming v5.0</u>:

- non-linear problems (beyond polynomial)
- Hybrid low-accuracy spike-based MPI solvers (Braegan Spring)
- Single precision GPU for linear systems (Chenkai Zhang)

www.feast-solver.org