



FEAST eigenvalue algorithm and solver: review and perspectives

Eric Polizzi

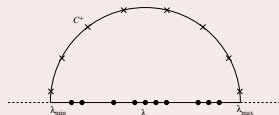
Department of Electrical and Computer Engineering
University of Massachusetts, Amherst, USA

Sparse Days, CERFACS, June 25, 2012

Overview of the FEAST Project

Algorithm

- New contour integration and density matrix-based algorithm for solving the eigenvalue problem
- Combines accuracy, robustness, performances, and exhibits natural parallelism at multiple levels



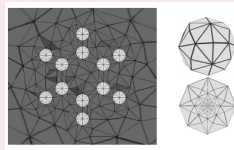
Software

- New “black-box” numerical library software
- Combines usability and flexibility



First-Principle Electronic Structure Framework

- Reformulation of the muffin-tin problem
- Direct solution of the DFT/Kohn-Sham problem
- Novel spectral time-dependent propagation schemes



FEAST- Algorithm

Algorithm- Big Picture

Contour Integration approach (Complex Analysis + Linear Algebra)

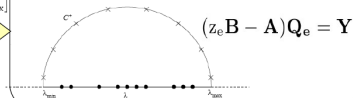
- SS projection method - *Sakurai-Sugiura*
- FEAST (density-matrix based)

Eigenvalue Problem

$$\mathbf{A}\mathbf{x}_j = \lambda_j \mathbf{B}\mathbf{x}_j$$



Set of Independent Linear Systems



Family of Eigenvalue Problems

- **Linear symmetric** $\mathbf{A}\mathbf{x} = \lambda \mathbf{B}\mathbf{x}$,
- **Linear non-symmetric** $\mathbf{A}\mathbf{x} = \lambda \mathbf{B}\mathbf{x}$,
- Non-linear eigenvalue $\mathbf{A}(\lambda)\mathbf{x} = \lambda \mathbf{B}\mathbf{x}$,
- **Non-linear eigenvector** $\mathbf{A}(\mathbf{x})\mathbf{x} = \lambda \mathbf{B}\mathbf{x}$,

\mathbf{A} sym./herm., \mathbf{B} spd/hpd

\mathbf{A}, \mathbf{B} general

\mathbf{A}, \mathbf{B} general

\mathbf{A} sym./herm., \mathbf{B} spd/hpd

Density Matrix

$$\rho \equiv -\frac{1}{2\pi i} \int_C dz \mathbf{G}(z) = \mathbf{X}\mathbf{X}^T$$

$$\mathbf{G}(z) = (z\mathbf{B} - \mathbf{A})^{-1}, \text{ and}$$

$$\mathbf{X}_{N \times M} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M\}$$

FEAST- basic idea- symmetric problem

- 1 Pick $\mathbf{Y}_{N \times M} = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_M\}$ random vectors
- 2 Postmultiply ρ by \mathbf{Y}

$$\mathbf{Q} = -\frac{1}{2\pi i} \int_C dz \mathbf{G}(z) \mathbf{Y} = \mathbf{X}(\mathbf{X}^T \mathbf{Y})$$

- 3 Solve (Rayleigh-Ritz)

$$(\mathbf{Q}^T \mathbf{A} \mathbf{Q}) \mathbf{p}_i = \mathbf{e}_i (\mathbf{Q}^T \mathbf{B} \mathbf{Q}) \mathbf{p}_i,$$

then $\mathbf{x}_i = \mathbf{Q} \mathbf{p}_i$, $\lambda_i = \mathbf{e}_i$

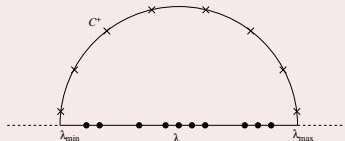
FEAST- Algorithm for the Symmetric Problem

Contour Integration- Gauss Quadrature (x_e, ω_e)

$$\mathbf{Q} = - \sum_{e=1}^{N_e} \frac{1}{2} \omega_e \Re \{ r \exp(i\theta_e) \mathbf{G}(z_e) \mathbf{Y} \} = (\mathbf{X} \tilde{\mathbf{I}}_M \mathbf{X}^T + \mathbf{X}_{l,r} \tilde{\mathbf{O}} \mathbf{X}_{l,r}^T) \mathbf{Y}$$

$$z_e = (\lambda_{\max} + \lambda_{\min})/2 + r \exp(i\theta_e),$$

$$\theta_e = -(\pi/2)(x_e - 1), r = (\lambda_{\max} - \lambda_{\min})/2$$

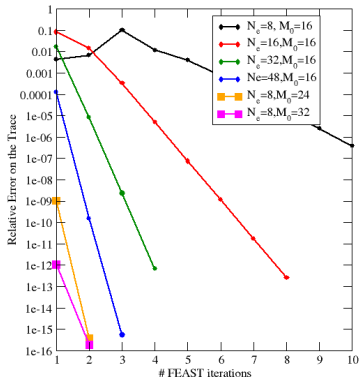
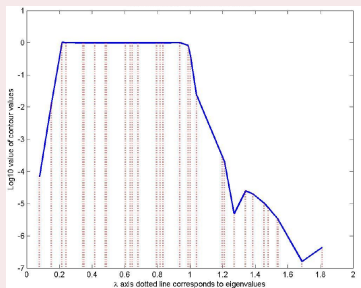


FEAST Algorithm - Subspace Iterations

- 1 Pick $\mathbf{Y}_{N \times M_0} = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_{M_0}\}$ random vectors ($M_0 > M$)
- 2 Compute $\mathbf{Q}_{N \times M_0} = \rho \mathbf{Y}_{N \times M_0}$,
- 3 Solve (Rayleigh-Ritz) $(\mathbf{Q}^T \mathbf{A} \mathbf{Q}) \mathbf{p}_i = \mathbf{e}_i (\mathbf{Q}^T \mathbf{B} \mathbf{Q}) \mathbf{p}_i$, then $\mathbf{x}_i = \mathbf{Q} \mathbf{p}_i$, $\lambda_i = \mathbf{e}_i$
- 4 Check convergence. Go back step 2 if needed using $\mathbf{Y} = \mathbf{B} \mathbf{X}$

FEAST - increase N_e or M_0 ?

Example $\mathbf{Ax} = \lambda \mathbf{Bx}$, $N=1671$, $\lambda \in [0.18, 1.0]$, 16 eigenvalues

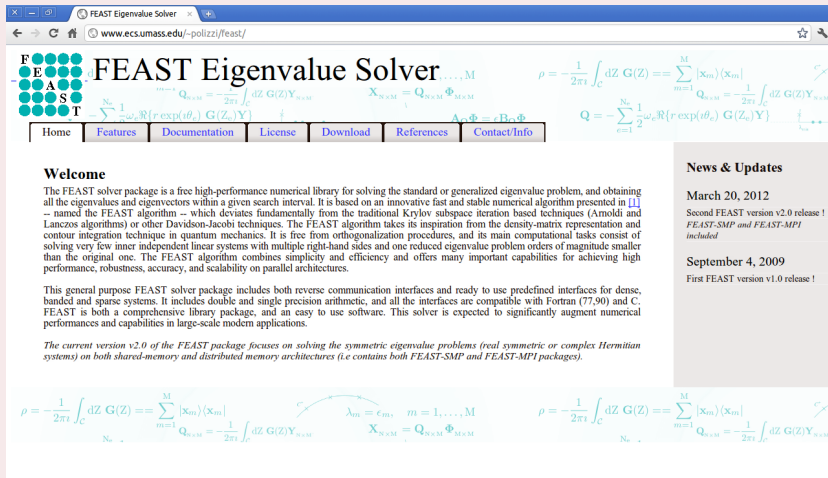


Properties

- 1 **Fast and systematic convergence** Using 8 to 16 contour points, FEAST converges in 2-3 iterations only to obtain up to thousands of eigenpairs with machine accuracy.
- 2 **Naturally captures all multiplicities**
- 3 **No (explicit) orthogonalization procedure**
- 4 **Reusable subspace** for fast convergence or as suitable initial guess for solving a series of eigenvalue problems that are close to one another (e.g. for bandstructure calculations, time-dependent propagation, etc.).
- 5 **Allow the use of iterative methods for solving the complex linear systems**
- 6 **Natural parallelism at three levels**
 - search interval
 - contour points
 - inner linear systems (also with multiple rhs)

Ultimately solving the eigenvalue problem of any sizes reduced to solving only one linear system

<http://www.ecs.umass.edu/~polizzi/feast>



The screenshot shows the homepage of the FEAST Eigenvalue Solver. The header features the project logo (a grid of colored dots) and the title "FEAST Eigenvalue Solver". Below the title is a navigation bar with links: Home, Features, Documentation, License, Download, References, and Contact/Info. The main content area includes a "Welcome" section with a paragraph describing the solver's capabilities, a "News & Updates" section with dates and release information, and a "The current version v2.0" section. Mathematical equations for the eigenvalue problem are displayed on the right side of the page.

Welcome

The FEAST solver package is a free high-performance numerical library for solving the standard or generalized eigenvalue problem, and obtaining all the eigenvalues and eigenvectors within a given search interval. It is based on an innovative fast and stable numerical algorithm presented in [1] – named the FEAST algorithm – which deviates fundamentally from the traditional Krylov subspace iteration based techniques (Arnoldi and Lanczos algorithms) or other Davidson-Jacobi techniques. The FEAST algorithm takes its inspiration from the density-matrix representation and contour integration technique in quantum mechanics. Its main computational tasks consist of solving very few inner independent linear systems with multiple right-hand sides and one reduced eigenvalue problem orders of magnitude smaller than the original one. The FEAST algorithm combines simplicity and efficiency and offers many important capabilities for achieving high performance, robustness, accuracy, and scalability on parallel architectures.

This general purpose FEAST solver package includes both reverse communication interfaces and ready to use predefined interfaces for dense, banded and sparse systems. It includes double and single precision arithmetic, and all the interfaces are compatible with Fortran (77,90) and C. FEAST is both a comprehensive library package, and an easy to use software. This solver is expected to significantly augment numerical performances and capabilities in large-scale modern applications.

The current version v2.0 of the FEAST package focuses on solving the symmetric eigenvalue problems (real symmetric or complex Hermitian systems) on both shared-memory and distributed memory architectures (i.e. contains both FEAST-SMP and FEAST-MPI packages).

News & Updates

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

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

$$\rho = -\frac{1}{2\pi i} \int_C dZ G(Z) = \sum_{m=1}^M |\mathbf{x}_m\rangle \langle \mathbf{x}_m|$$

$$\mathbf{Q}_{N \times M} = -\frac{1}{2\pi i} \int_C dZ G(Z) \mathbf{Y}_{N \times M}$$

$$\mathbf{X}_{N \times M} = \mathbf{Q}_{N \times M} \Phi_{M \times M}$$

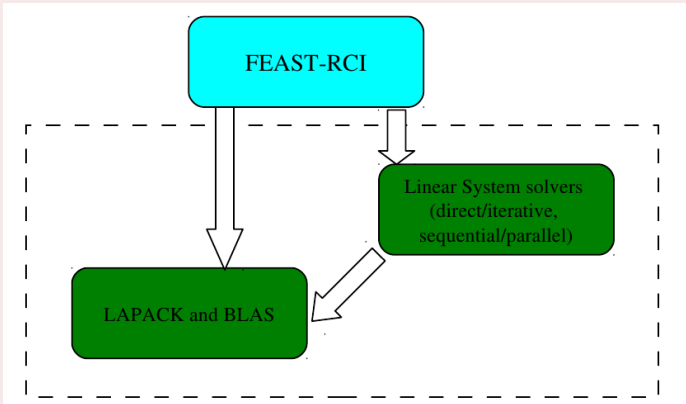
$$\mathbf{A}_{N \times N} \Phi = \epsilon \mathbf{B}_{N \times N} \Phi$$

$$\lambda_m = \epsilon_m, \quad m = 1, \dots, M$$

Features

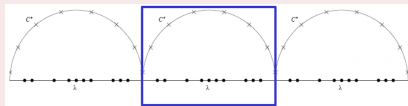
- ❶ Solving $\mathbf{Ax} = \lambda \mathbf{x}$ or $\mathbf{Ax} = \lambda \mathbf{Bx}$, \mathbf{A} is real symmetric or complex Hermitian, \mathbf{B} is spd or hpd.
- ❷ Two libraries: **SMP version** (one node), and **MPI version** (multi-nodes).
- ❸ Real/Complex and Single/Double precisions,
- ❹ All FEAST interfaces compatible with Fortran (77, 90), and C (and any MPI implementations). FEAST can be linked with any BLAS/LAPACK packages.
- ❺ **Two interface models:**
 - Reverse communication interfaces (RCI). Matrix format and linear system solver independent.
 - Predefined optimized drivers for dense, banded, and sparse (CSR); Requirement: SPIKE primitives (included)– (MKL)-Pardiso
- ❻ Detailed documentation and large number of examples provided,
- ❼ FEAST utility drivers for sparse systems for quick testing, timing, etc. .

FEAST- RCI (i.e. the FEAST kernel)

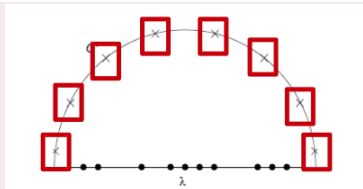


FEAST v2.0 can be linked with any linear system solvers

Three levels of Parallelism



Search intervals independent (no overlap)
FEAST-MPI selects a cluster of N nodes (user customized `MPI_COMM_WORLD`)



Contour points independent (solved simultaneously)
FEAST-MPI distributes the N_e contour points over N nodes (automatic)

$$(z_e B - A) Q_e = Y$$

Linear system solved in parallel
FEAST-MPI or **FEAST-SMP** use a shared memory solver (OpenMP)

- **FEAST v2.0** currently features MPI-MPI-OpenMP
- **FEAST vx.x** in preparation, calling MPI-solver (i.e. MPI-MPI-MPI approach for petascale/exascale computing)

FEAST-MPI Some Results

- Intel Nehalem cluster 8 nodes (64 cores), 48Gb per node, 2.66Ghz, Infiniband
- FEAST-MPI, using INTEL-MKL: Pardiso, Lapack, Blas
- 8 contour points, $M_0 = 1.5M$, convergence trace $< 10^{-13}$ (3 loops), residual $< 10^{-11}$

$Ax = \lambda Bx$, $N = 12,450$ $M = 100$ $nnz = 86,808$

Total Time (s)	1 node	2 nodes	4 nodes	8 nodes
FEAST	4.4	2.3	1.5	0.6
ARPACK	7			

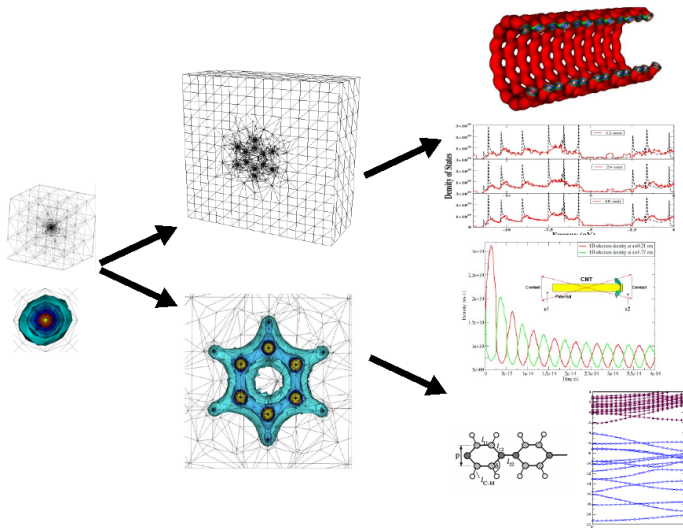
$Ax = \lambda Bx$, $N = 99,600$ $M = 800$ $nnz = 694,464$

Total Time (s)	1 node	2 nodes	4 nodes	8 nodes
FEAST	245	143	90	60
ARPACK	2580			
FEAST-40s*	205	105	50	20

* 40s==LAPACK Time for solving the reduced system on 1 node

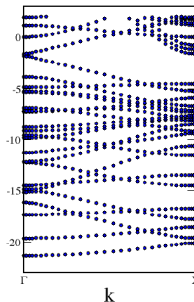
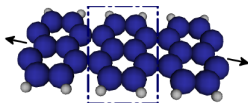
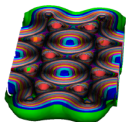
FEAST: From Atoms to Molecules and Nanostructures

Closed-Periodic system, DFT and TDDFT, All-electron, Real-space mesh (cubic FEM)

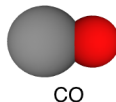
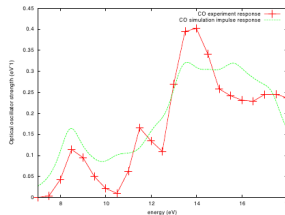
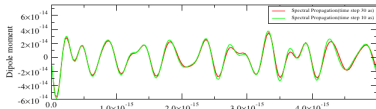


FEAST - Good Initial Subspace in Applications

- Bandstructure calculations (J. Kestyn, E. Polizzi)



- Time Dependent Propagation (Z. Chen, E. Polizzi and J. Jerome)



Density Functional Theory/Kohn-Sham Equations

$$\left(-\frac{1}{2m}\nabla^2 + V_{eff}[n(r)]\right)\psi_i(r) = E_i\psi_i(r),$$

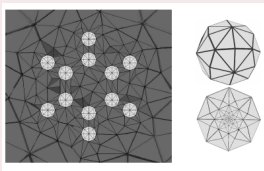
$$n(r) = 2 \sum_{i=1}^{N_E} |\psi_i(r)|^2.$$

$$V_{eff}[n(r)] = V_{ion}(r) + V_H[n(r)] + V_{XC}[n(r)],$$

$$-\nabla^2 V_H(r) = \frac{\rho(r)}{\epsilon}.$$

- Traditionally solved self-consistently
- Large linear eigenvalue problem to solve at each iteration
- Since the 1930s, several attempts to reduce the complexity of this eigenvalue problem by dissociating, screening or removing the effect of the core electrons

Muffin-tin approach



- Domain decomposition leads to a non-linear eigenvalue problem $H(E)\Psi = E\Psi$ in the interstitial region
- APW method originally introduced by Slater in 1937, he then stated: «Of course, we cannot solve this exactly, and we must look for methods of approximations »
- These limitations have historically motivated the development of a wide spectrum of approximation techniques ranging from direct linearization (LAPW, LMTO, etc.) to pseudopotential methods

Implicit Treatment of the non-linear eigenvalue problem using FEAST

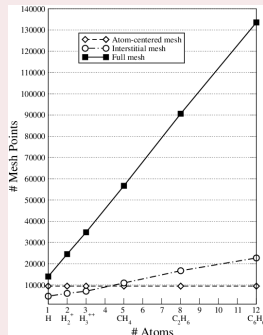
$$(ZS - H)Q^{(Z)} = Y, \quad \longleftrightarrow$$

$$(ZS_0 - H_0 + \sum_j \Sigma_j(Z))Q_0^{(Z)} = Y_0 + \sum_j F_j^{(Z)}$$

$$F_j^{(Z)} = \Sigma_j(Z)G_j(Z)Y_j$$

$$(Z - H_j)Q^{(Z)}(x) = Y(x), \quad x \in \Omega_j,$$

- Non-linear dependency on energy (eigenvalue) removed for the interstitial region
- Massively parallel (domain-decomposition)
- It is a numerically exact all-electron approach: bypass the motivations and needs for linear approximations and/or pseudopotentials
- A. Levin, D. Zhang, E. Polizzi, *Computer Physics Communications*, in press



Alternatives to traditional SCF methods (B. Gavin, E. Polizzi)

$$\left(-\frac{1}{2m}\nabla^2 + V_{eff}[n(r)]\right)\psi_i(r) = E_i\psi_i(r),$$

$$n(r) = 2 \sum_{i=1}^{N_E} |\psi_i(r)|^2.$$

$$V_{eff}[n(r)] = V_{ion}(r) + V_H[n(r)] + V_{XC}[n(r)],$$

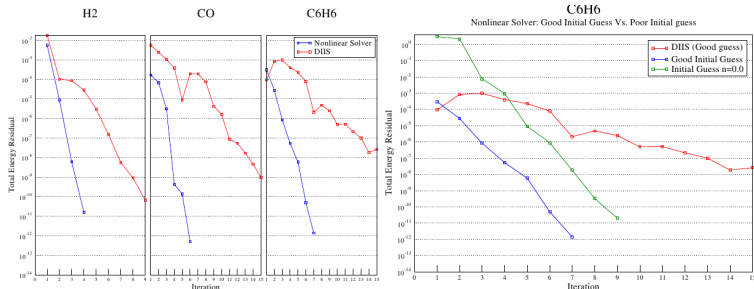
$$-\nabla^2 V_H(r) = \frac{\rho(r)}{\epsilon}.$$

- It is a non-linear eigenvector problem $H(\Psi)\Psi = E\Psi$
- Can be viewed as an optimization problem i.e. DCM algorithm from Chao Yang et al.
- Reduced system is now non-linear within FEAST i.e. $\mathbf{Q}^T \mathbf{H}[n] \mathbf{Q} \mathbf{p}_i = \mathbf{e}_i \mathbf{Q}^T \mathbf{B} \mathbf{Q} \mathbf{p}_i$

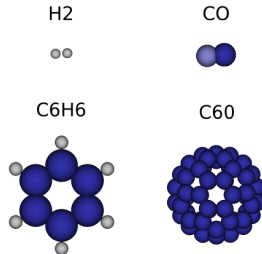
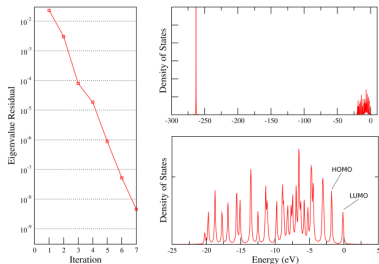
Advantages

- No initial guess needed
- Self-consistency on the reduced system more manageable (3 to 5 inner loops)
- High-converge rate (few contour loops <10): converges significantly faster than DIIS (Pulay) approaches, convergence rate remains high regardless of system size.
- General framework (i.e. black-box) but also fully-compatible with the muffin-tin real-space approach

FEAST Framework for Electronic Structure



C60 (Buckminsterfullerene):



Summary

FEAST

- Innovative Algorithm combines accuracy, robustness, performance, scalability
- State-of-the-art numerical library with different parallel programming models
- New Perspectives for Electronic Structure Calculations: Muffin-tin, Direct solution of the DFT/Kohn-Sham problem, Time-dependent propagation scheme

Current Goals

- Non-linear eigenvector problem $\mathbf{A}(\mathbf{x})\mathbf{x} = \lambda\mathbf{B}\mathbf{x}$ v2.0(!)+tutorial (in progress)
- Massively parallel implementation of the electronic structure problem using FEAST MPI-MPI-MPI
- Non-symmetric problem v3.0 (Spring 2013), (*see also S. Laux's recent work on complex bandstructure calculations*)

Acknowledgment

- Students: Z. Chen, B. Gavin, J. Kestyn, A. Levin, D. Zhang
- Support: Intel / NSF