Polynomial and rational filtering for eigenvalue problems and the EVSL project

Yousef Saad

Department of Computer Science and Engineering

University of Minnesota

ILAS 2017
July 25, 2017

Large eigenvalue problems in applications

➤ Challenge in eigenvalue problems: extract large number of eigenvalues & vectors of very large matrices (quantum physics/ chemistry, ...) - often in the middle of spectrum.

➤ Example: *Excited states* involve transitions $\longrightarrow$ much more complex computations than for DFT (ground states)

➤ Large matrices, *many* eigen-pairs to compute

Illustration:

'Hamiltonian of size $n \sim 10^6$ get 10% of bands'

Solving large interior eigenvalue problems

Three broad approaches:

1. Shift-invert (real shifts)
2. Polynomial filtering
3. Rational filtering (Cauchy, + others).

Issues with shift-and invert (and related approaches)

➤ Issue 1: factorization may be too expensive

- Can use iterative methods?

➤ Issue 2: Iterative techniques often fail –

- Reason: Highly indefinite problems.

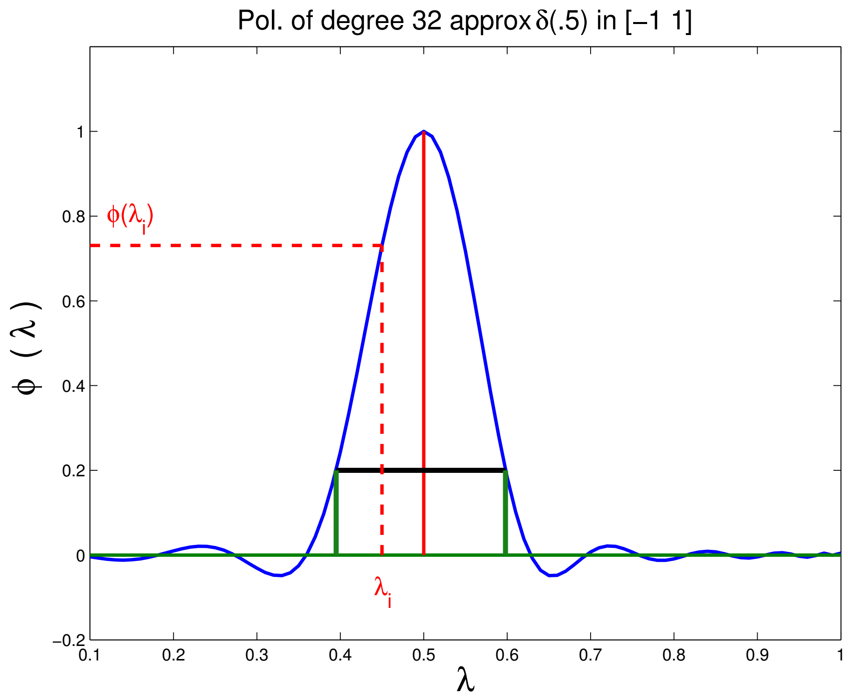
➤ First Alternative: ‘Spectrum slicing’ with Polynomial filtering

"Spectrum Slicing"

- Situation: very large number of eigenvalues to be computed
- Goal: compute spectrum by slices by applying filtering
- Apply Lanczos or Subspace iteration to problem:

$$\phi(A)u = \mu u$$

$\phi(t) \equiv$ a polynomial or rational function that enhances wanted eigenvalues



Rationale. Eigenvectors on both ends of wanted spectrum need not be orthogonalized against each other :

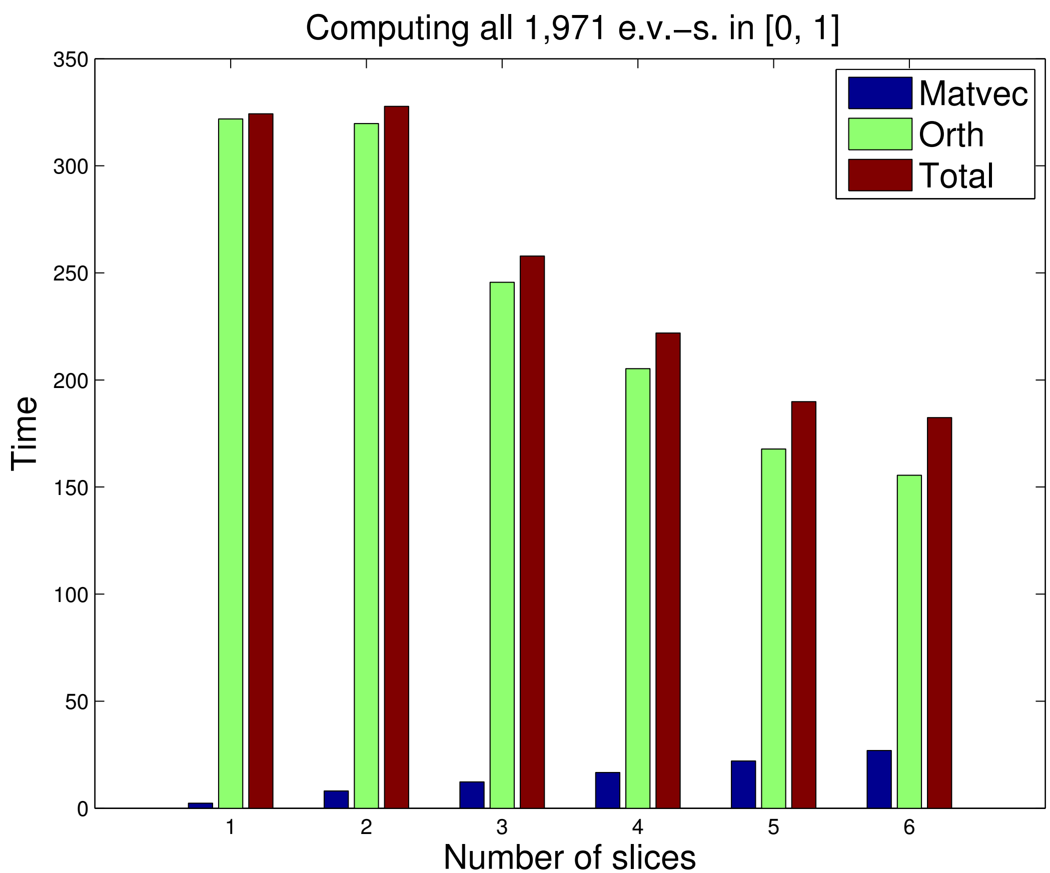
➤ Idea: Get the spectrum by ‘slices’ or ’windows’ [e.g., a few hundreds or thousands of pairs at a time]

➤ Can use polynomial or rational filters

*Hypothetical scenario: large A, *many* wanted eigenpairs*

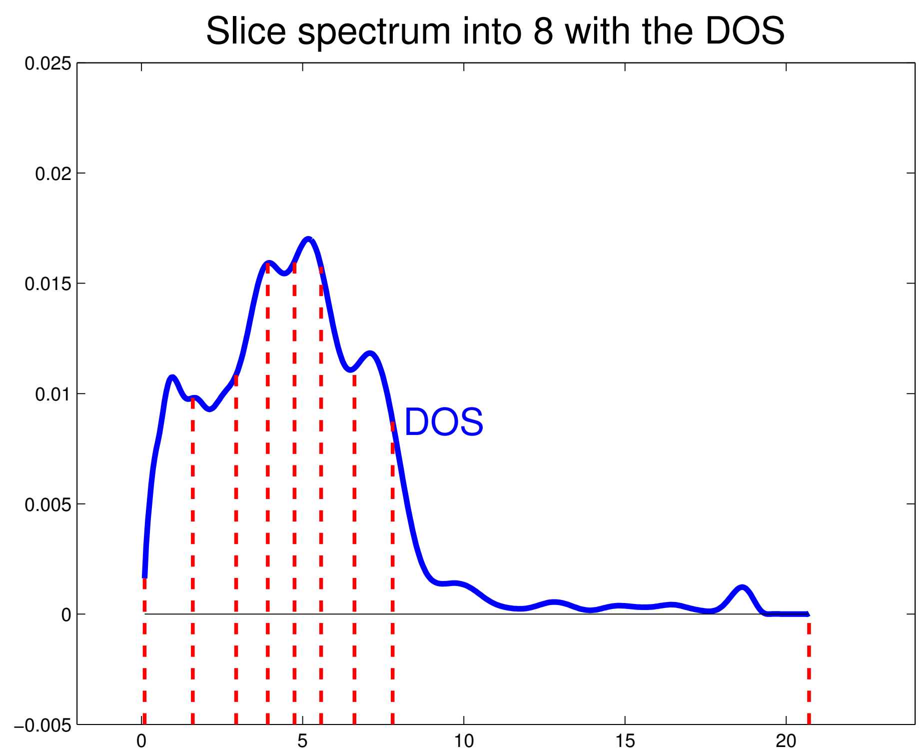
- Assume A has size $10M$
- ... and you want to compute 50,000 eigenvalues/vectors (huge for numerical analysits, not for physicists) ...
- ... in the lower part of the spectrum - or the middle.
- By (any) standard method you will need to orthogonalize at least 50K vectors of size 10M. Then:
  - Space needed: $\approx 4 \times 10^{12}$ b = 4TB *just for the basis*
  - Orthogonalization cost: 5×10^{16} = 50 PetaOPS.
  - At step k, each orthogonalization step costs $\approx 4kn$
  - This is $\approx 200,000n$ for k close to $50,000$.

Illustration: All eigenvalues in [0, 1] of a 49^3 Laplacean



Note: This is a small pb. in a scalar environment. Effect likely much more pronounced in a fully parallel case.

How do I slice my spectrum?



Answer: Use the DOS.

➤ We must have:

$$\int_{t_i}^{t_{i+1}} \phi(t)dt = \frac{1}{n_{slices}} \int_a^b \phi(t)dt$$

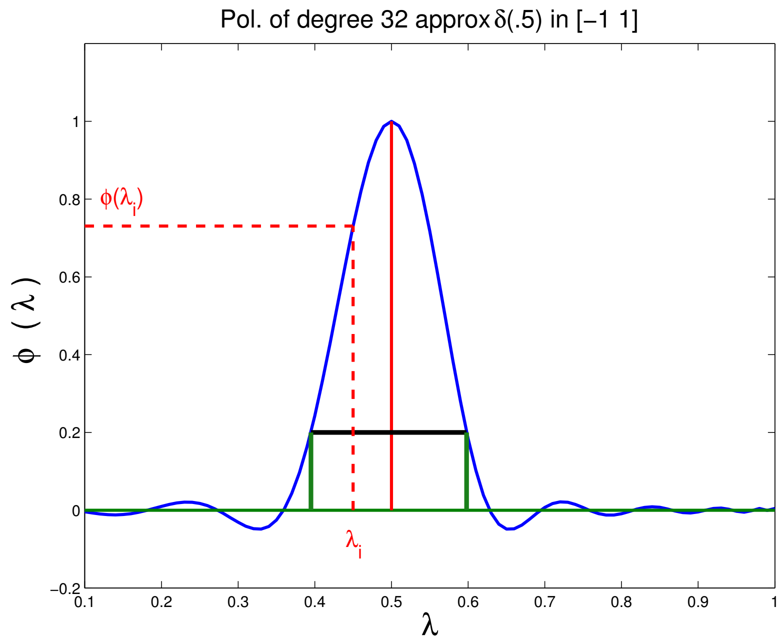
Polynomial filtering

- Apply Lanczos or Subspace iteration to: $\boxed{M = \rho(A)}$ where $\rho(t)$ is a polynomial
- Each *matvec* $y = Av$ is replaced by $y = \rho(A)v$.
- Eigenvalues in high part of filter will be computed first.
- Old (forgotten) idea. But new context is *very* favorable

What polynomials?

➤ LS approximations to δ-Dirac functions

➤ Obtain the LS approximation to the δ— Dirac function – Centered at some point (TBD) inside the interval. $\longrightarrow$



➤ W'll express everything in the interval $[-1, 1]$

Theory

The Chebyshev expansion of δ_γ is

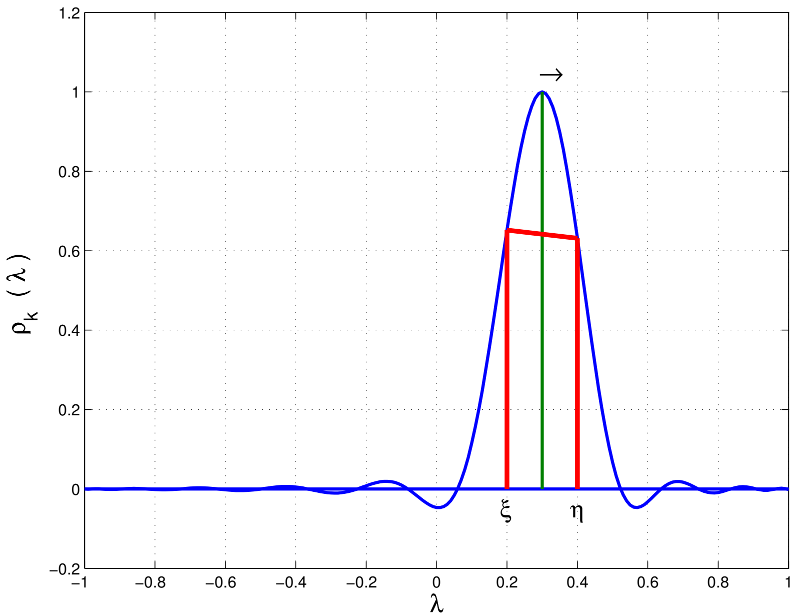
$$\rho_k(t) = \sum_{j=0}^{k} \mu_j T_j(t) \ \text{ with } \ \mu_j = \begin{cases} \frac{1}{2} & j = 0 \\ \cos(j \cos^{-1}(\gamma)) & j > 0 \end{cases}$$

➤ Recall: The delta Dirac function is not a function – we can't properly approximate it in least-squares sense. However:

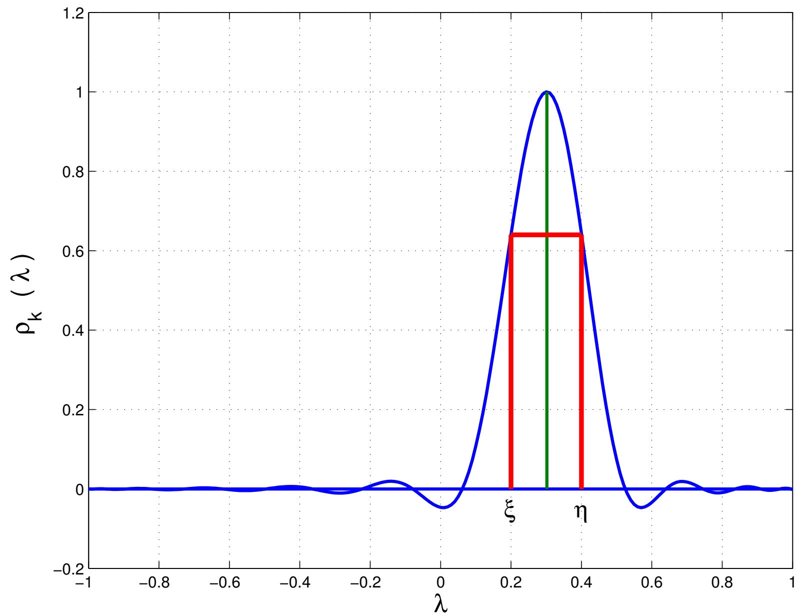
Proposition Let $\hat{\rho}_k(t)$ be the polynomial that minimizes $\|r(t)\|_w$ over all polynomials r of degree $\le k$, such that $r(\gamma) = 1$, where $\|.\|_w$ represents the Chebyshev L^2-norm. Then $\hat{\rho}_k(t) = \rho_k(t)/\rho_k(\gamma)$.

A few technical details. Issue # one: 'balance the filter'

- To facilitate the selection of *'wanted'* eigenvalues [Select λ's such that $\rho(\lambda) >$ bar] we need to ...
- ... find γ so that $\rho(\xi) == \rho(\eta)$



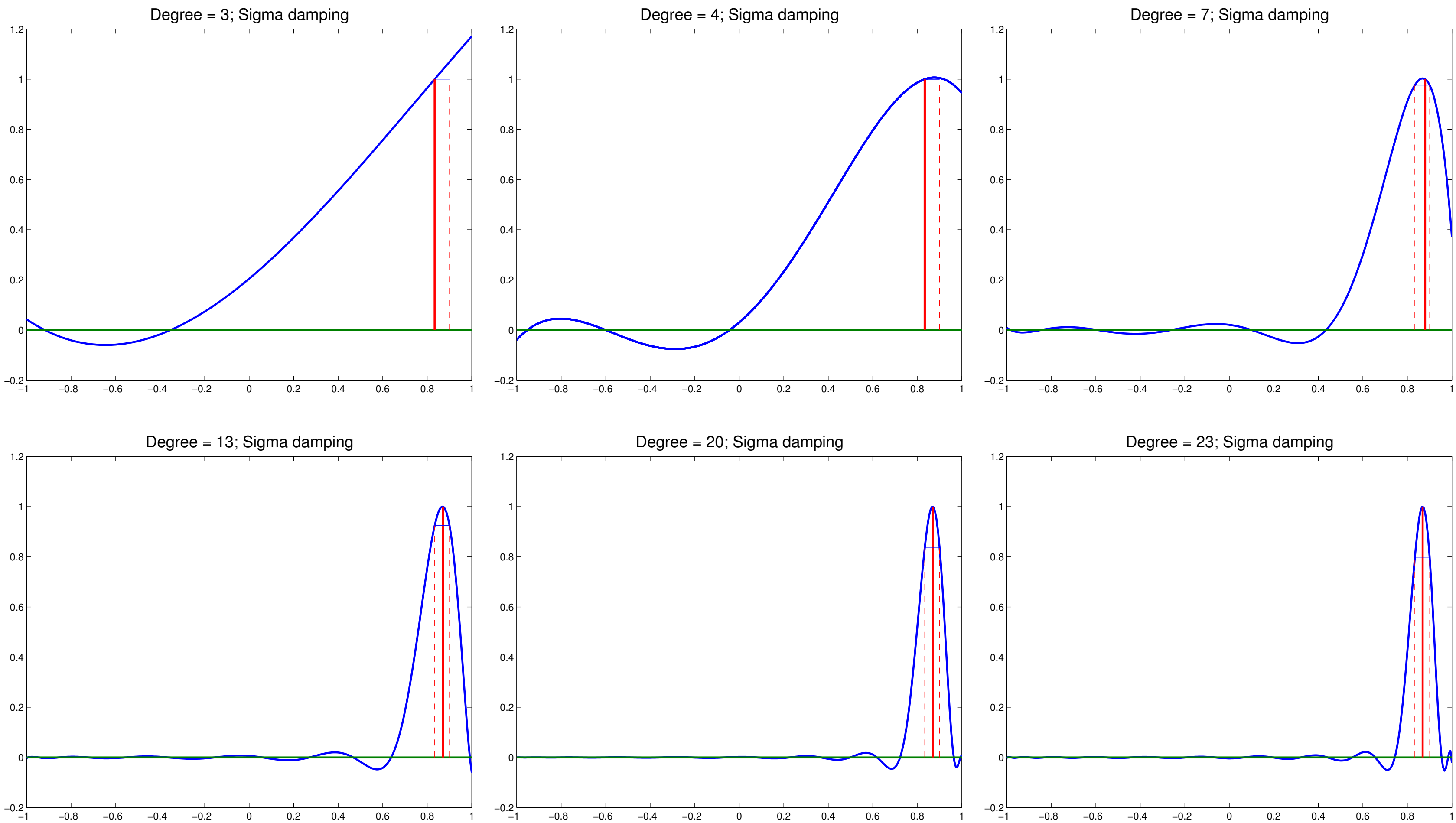
$\longrightarrow$



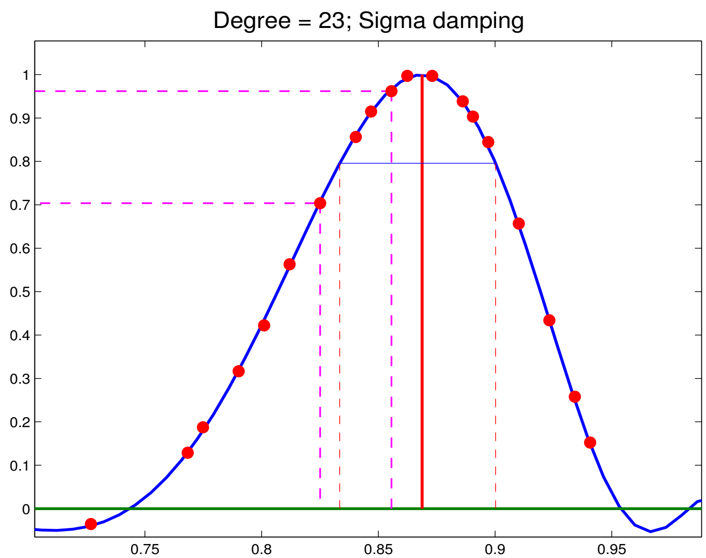
Procedure: Solve the equation $\rho_\gamma(\xi) - \rho_\gamma(\eta) = 0$ with respect to γ, accurately. Use Newton or eigenvalue formulation.

Issue # two: Determine degree & polynomial (automatically)

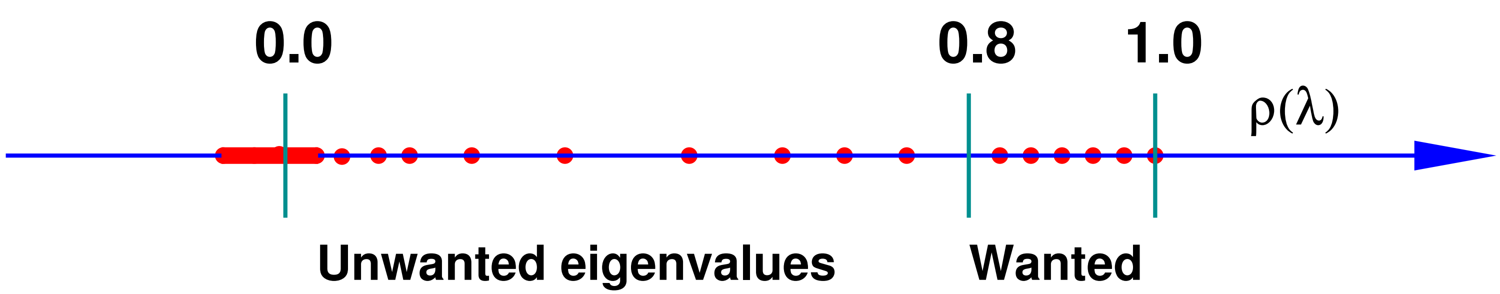
Start low then increase degree until value (s) at the boundary (ies) become small enough - Exple for [0.833, 0.907..]



Polynomial filtered Lanczos: No-Restart version

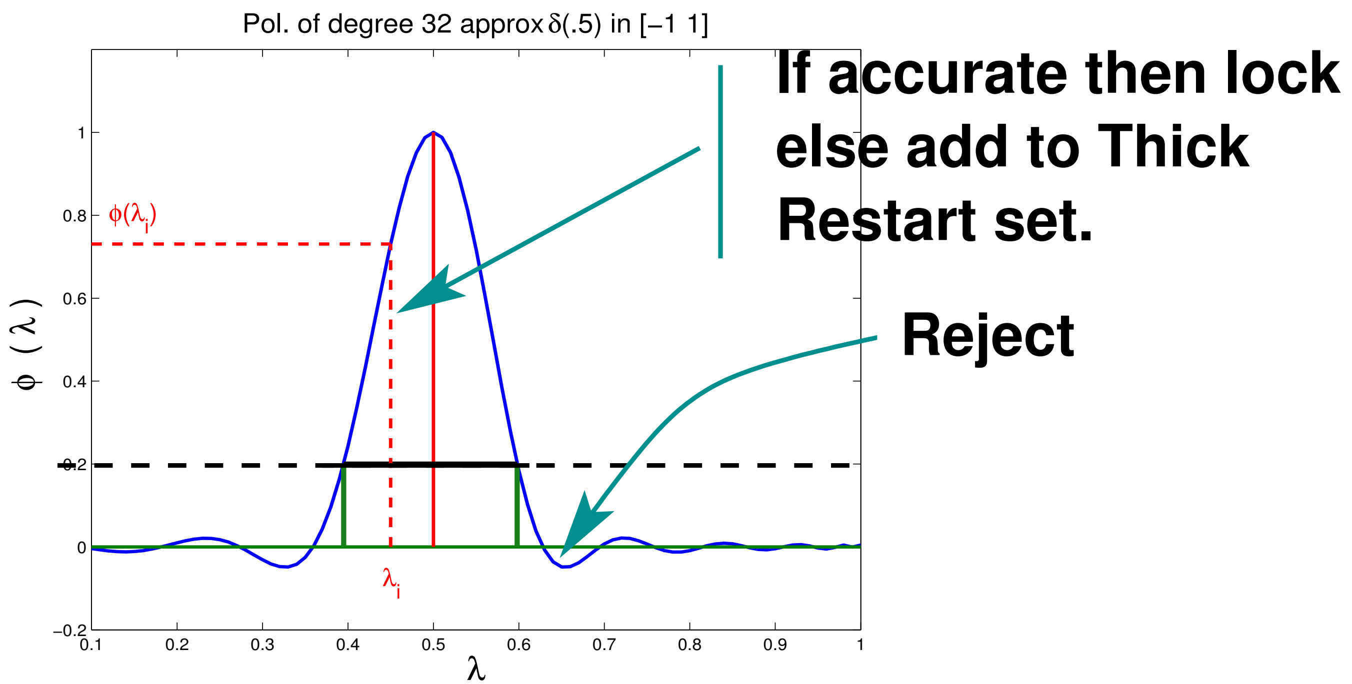


- Use Lanczos with full reorthogonalization on $\rho(A)$. Eigenvalues of $\rho(A)$: $\rho(\lambda_i)$
- Accept if $\rho(\lambda_i) \geq$ `bar`
- Ignore if $\rho(\lambda_i) <$ `bar`



Polynomial filtered Lanczos: Thick-Restart version

➤ PolFilt Thick-Restart Lanczos in a picture:

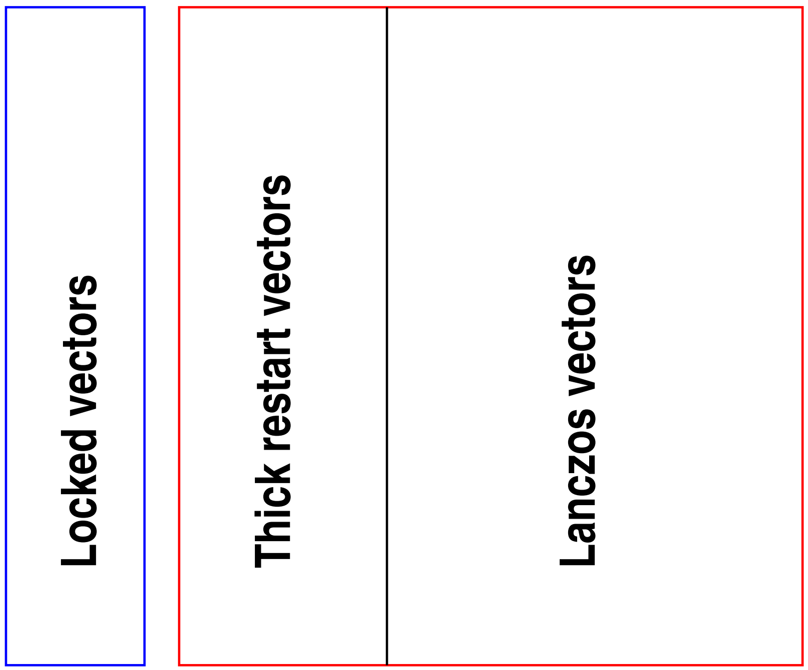


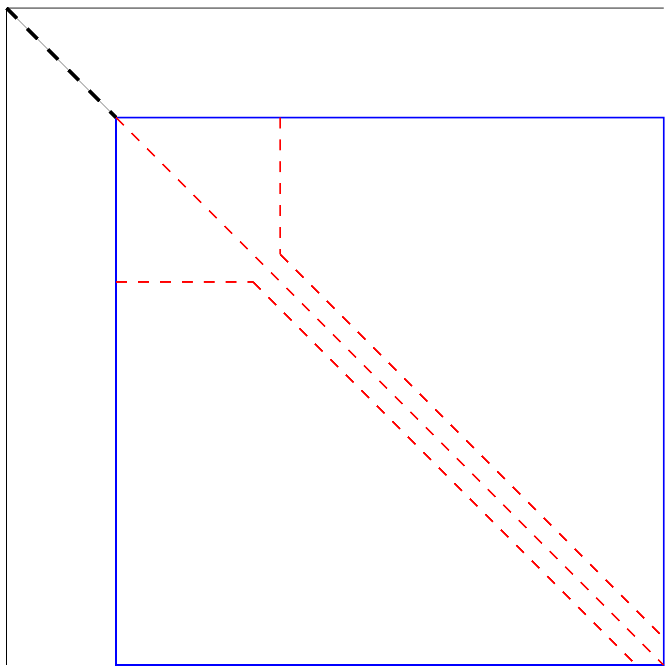
➤ Due to locking, no more candidates will show up in wanted area after some point $\rightarrow$ Stop.

TR Lanczos: The 3 types of basis vectors

Basis vectors

Matrix representation





Experiments: Hamiltonian matrices from PARSEC

Matrix	n	$\sim$ nnz	$[a, b]$	$[\xi, \eta]$	$\nu_{[\xi,\eta]}$
$\mathbf{Ge_{87}H_{76}}$	$\mathbf{112,985}$	$\mathbf{7.9}M$	$[-\mathbf{1.21, 32.76}]$	$[-\mathbf{0.64}, -\mathbf{0.0053}]$	212
$\mathbf{Ge_{99}H_{100}}$	$\mathbf{112,985}$	$\mathbf{8.5}M$	$[-\mathbf{1.22, 32.70}]$	$[-\mathbf{0.65}, -\mathbf{0.0096}]$	250
$\mathbf{Si_{41}Ge_{41}H_{72}}$	$\mathbf{185,639}$	$\mathbf{15.0}M$	$[-\mathbf{1.12, 49.82}]$	$[-\mathbf{0.64}, -\mathbf{0.0028}]$	218
$\mathbf{Si_{87}H_{76}}$	$\mathbf{240,369}$	$\mathbf{10.6}M$	$[-\mathbf{1.19, 43.07}]$	$[-\mathbf{0.66}, -\mathbf{0.0300}]$	213
$\mathbf{Ga_{41}As_{41}H_{72}}$	$\mathbf{268,096}$	$\mathbf{18.5}M$	$[-\mathbf{1.25, 1301}]$	$[-\mathbf{0.64}, -\mathbf{0.0000}]$	201

Results: (No-Restart Lanczos)

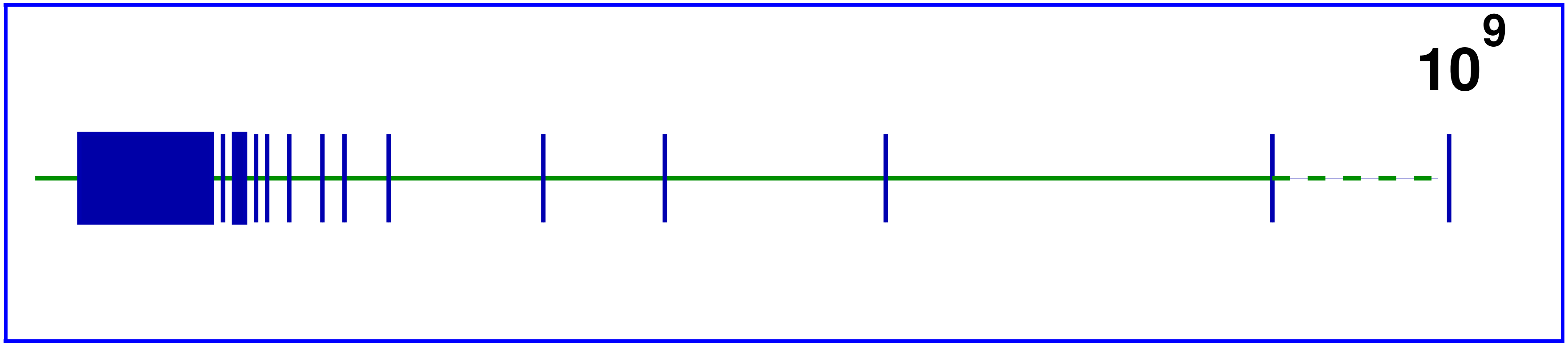
Matrix	deg	iter	matvec	CPU time (sec)			max residual
				matvec	orth.	total	
$Ge_{87}H_{76}$	26	1,020	26,784	48.58	18.67	74.45	1.20×10^{-12}
$Ge_{99}H_{100}$	26	1,090	28,642	60.11	20.44	86.52	7.20×10^{-12}
$Si_{41}Ge_{41}H_{72}$	32	950	30,682	105.05	28.25	144.19	1.20×10^{-10}
$Si_{87}H_{76}$	29	1,010	29,561	76.45	39.16	128.95	4.30×10^{-12}
$Ga_{41}As_{41}H_{72}$	174	910	158,889	693.5	34.16	759.99	3.70×10^{-12}

➤ Demo with Si10H16 [$n = 17,077, \ nnz(A) = 446,500$]

RATIONAL FILTERS

Why use rational filters?

- Consider a spectrum like this one:



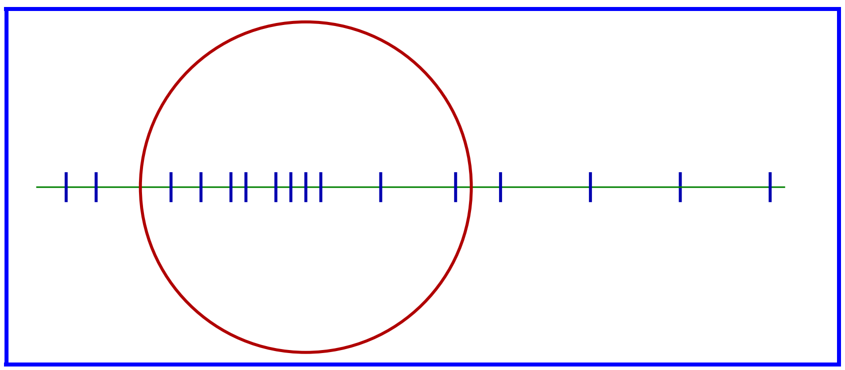
- Polynomial filtering utterly ineffective for this case
- Second issue: situation when Matrix-vector products are expensive
- Generalized eigenvalue problems.

- Alternative is to use rational filters:

$$\phi(z) = \sum_j \frac{\alpha_j}{z-\sigma_j}$$

$$\phi(A) = \sum_j \alpha_j (A - \sigma_j I)^{-1}$$ $\rightarrow$ We now need to solve linear systems

- Tool: Cauchy integral representations of spectral projectors

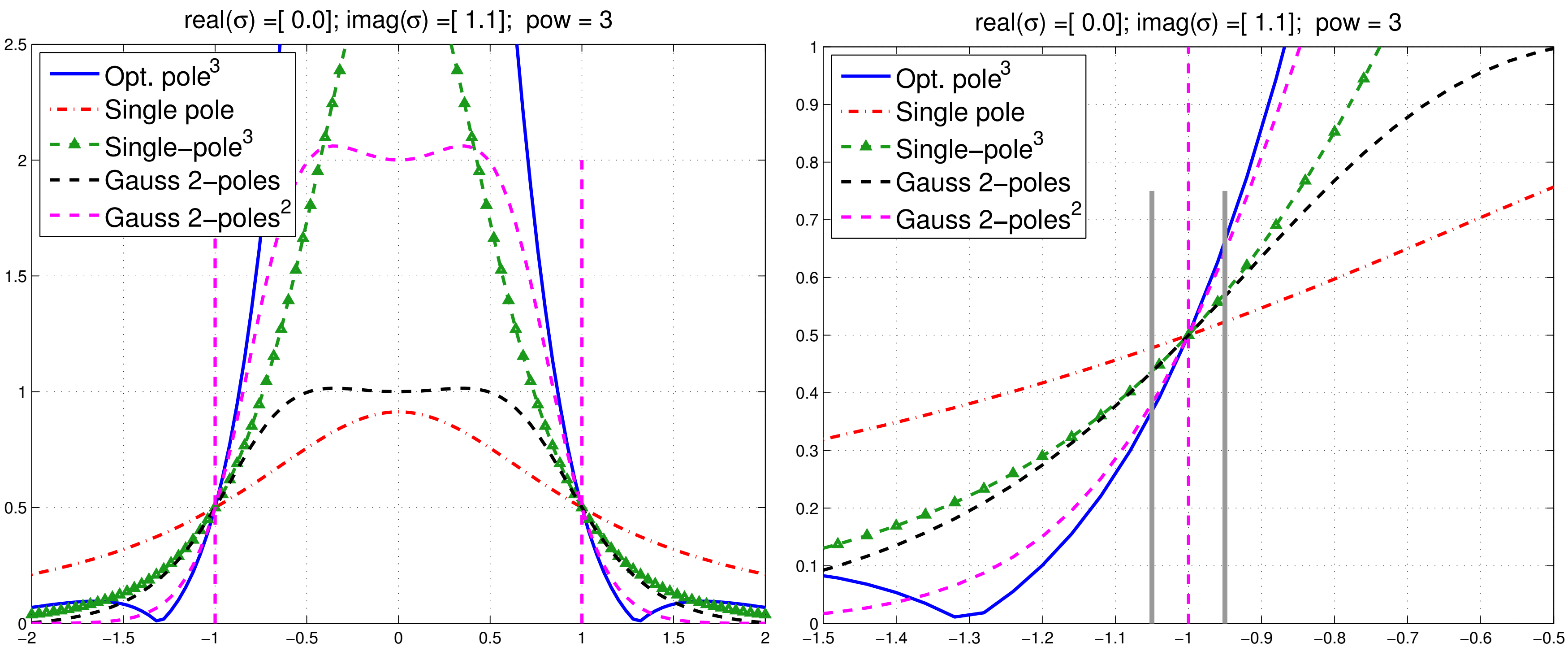


$$P = \frac{-1}{2i\pi} \int_\Gamma (A - sI)^{-1} ds$$

- Numer. integr. $P \rightarrow \tilde{P}$
- Use Krylov or S.I. on $\tilde{P}$

- Sakurai-Sugiura approach [Krylov]
- FEAST [Subs. iter.] (E. Polizzi)

What makes a good filter



- Assume subspace iteration is used with above filters. Which filter will give better convergence?

- Simplest and best indicator of performance of a filter is the magnitude of its derivative at -1 (or 1)

The Gauss viewpoint: Least-squares rational filters

- Given: poles $\sigma_1, \sigma_2, \cdots, \sigma_p$
- Related basis functions $\phi_j(z) = \frac{1}{z - \sigma_j}$

Find $\phi(z) = \sum_{j=1}^{p} \alpha_j \phi_j(z)$ that minimizes

$$\int_{-\infty}^{\infty} w(t)|h(t) - \phi(t)|^2 dt$$

- $h(t)$ = step function $\chi_{[-1,1]}$.
- $w(t)$= weight function.

For example $a = 10$,
$\beta = 0.1$

$$w(t) = \begin{cases} 0 \text{ if } & |t| > a \\ \beta \text{ if } & |t| \le 1 \\ 1 \text{ else} & \end{cases}$$

➤ Advantages:

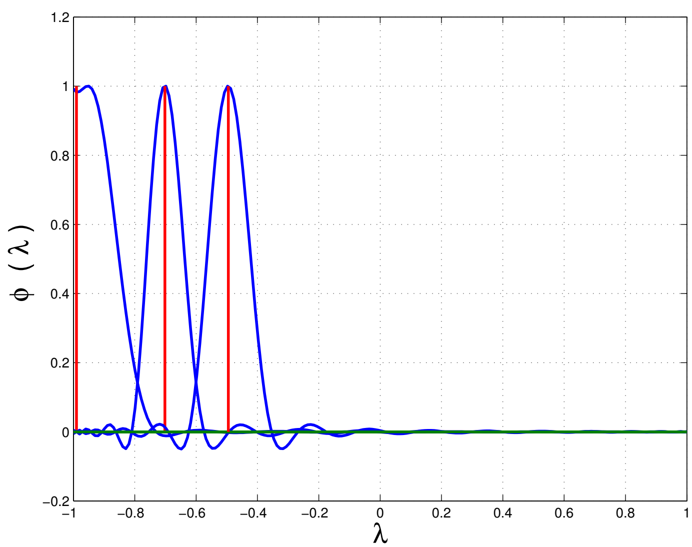
- Can select poles far away from real axis $\rightarrow$ faster iterative solvers
- Very flexible – can be adapted to many situations
- Can repeat poles (!)

➤ Implemented in EVSL.. [Interfaced to `SuiteSparse` as a solver]

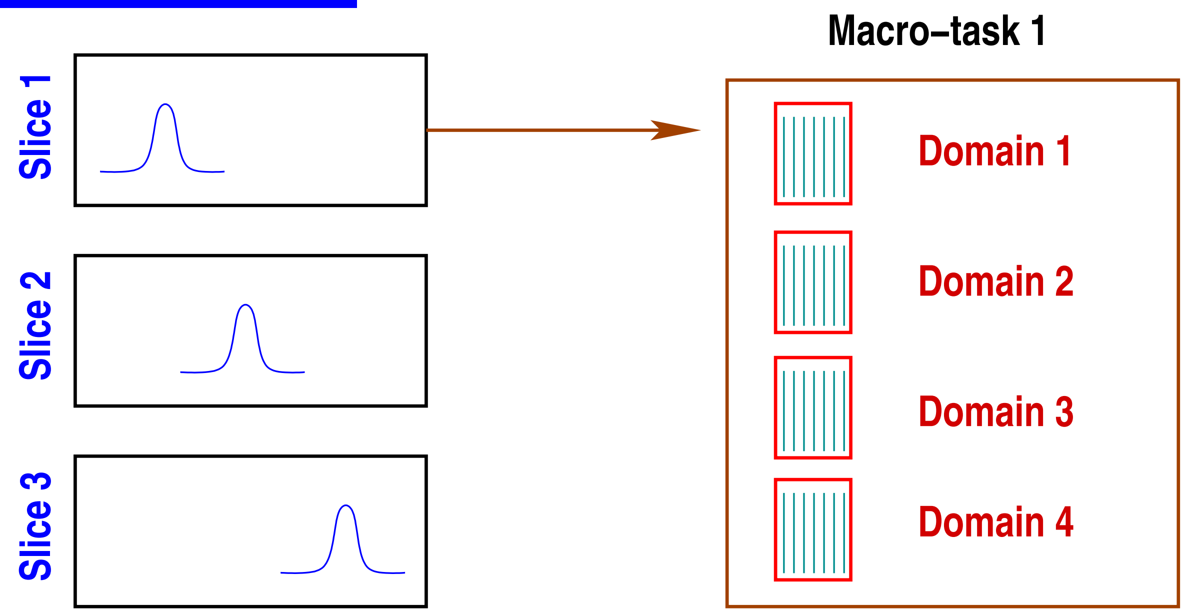
Spectrum Slicing and the EVSL project

- Newly released EVSL uses polynomial and rational filters
- Each can be appealing in different situations.

Spectrum slicing: cut the overall interval containing the spectrum into small sub-intervals and compute eigenpairs in each sub-interval independently.



Levels of parallelism



The two main levels of parallelism in EVSL

EVSL web-page - Mozilla Firefox

Yousef Saad -- SOFTWARE × EVSL web-page ×

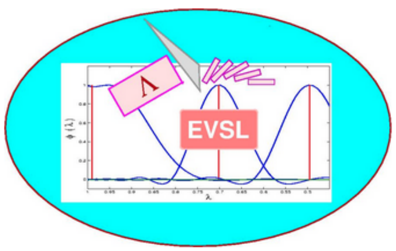
www-users.cs.umn.edu/~saad/software/EVSL/index.html 90% Ruipeng Li

Most Visited UoM Mail Home - Dropbox Research.gov - User L... Gmail OneStop : Welcome t... Google Accounts Live Statistics | Revolv... Setting up a repositor...

E V S L : *EigenValues Slicing Library* -- (Version 1.0)

This version dated : Thu Jan 26 15:24:02 CST 2017
New version to be posted (~ July 2017)

E . V . S . L .



Welcome to the first release of EVSL (the EigenValues Slicing Library). EVSL provides routines for computing eigenvalues located in a given interval, and their associated eigenvectors, of a real symmetric matrix. It also provides tools for spectrum slicing, i.e., the technique of subdividing a given interval into p smaller subintervals and computing the eigenvalues in each subinterval independently. EVSL implements a polynomial filtered Lanczos (thick restart, no restart) a rational filtered Lanczos (thick restart, no restart), and a polynomial filtered subspace iteration.
The technical reports listed below provide details on the techniques used in the package. **Online documentation** (based on Doxygen) is now available - see below.
The package will see frequent updates. We are currently working on various **interfaces to Fortran**.
Note: A new version of EVSL - with much added functionality - will be released in the next few weeks (Some time at the end of May/early June'17).

http://www.cs.umn.edu/~saad/software

Related publications

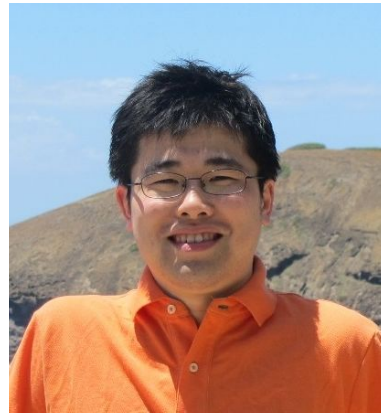
- **Ruipeng Li, Yuanzhe Xi, Eugene Vecharynski, Chao Yang, and Yousef Saad.**
  A Thick-Restart Lanczos algorithm with polynomial filtering for Hermitian eigenvalue problems. SIAM J. Sci. Comput., 38 (2016), pp. A2512-A2534. Preprint ys-2015-6, Dept. Computer Science and Engineering **Download: [PDF]**

Download EVSL

Before you download read the COPYRIGHT statement

Download: EVSL version 1.0 (EVSL_1.0.zip)

EVSL Main Contributors (version 1.1.0) + support



- Ruipeng Li
LLNL

- Yuanzhe Xi
Post-doc (UMN)

- Luke Erlandson
UG Intern (UMN)

➤ Work supported by DOE [ending this summer] ...

... and by NSF [going forward]

EVSL: current status & plans

Version _1.0 Released in Sept. 2016

- Matrices in CSR format (only)
- Standard Hermitian problems (no generalized)
- Spectrum slicing with KPM (Kernel Polynomial Meth.)
- Trivial parallelism across slices with OpenMP
- Methods:
  - Non-restart Lanczos – polynomial & rational filters
  - Thick-Restart Lanczos – polynomial & rational filters
  - Subspace iteration – polynomial & rational filters

Version _1.1.x V_1.1.0 Due for release end of July

- general `matvec` [passed as function pointer]
- $Ax = \lambda Bx$
- Fortran (03) interface.
- Spectrum slicing by Lanczos and KPM
- Efficient Spectrum slicing for $Ax = \lambda Bx$ (no solves with B).

Version _1.2.x V_1.2.0 Early 2018 (?)

- Fully parallel version [MPI + openMP]
- Challenge application in earth sciences [in progress]

Conclusion

- Polynomial Filtering appealing when # of eigenpairs to be computed is large and Matvecs are not too expensive
- Somewhat costly for generalized eigenvalue problems
- Will not work well for spectra with large outliers.
- Alternative: Rational filtering –
- Both approaches implemented in EVSL
- Current focus: provide as many interfaces as possible.
- `EVSL` code available here:

  `www.cs.umn.edu/~saad/software/EVSL`

- `EVSL` Also on github (development)