Polynomial and rational function filtering techniques for Hermitian eigenvalue problems

Yousef Saad
Department of Computer Science and Engineering

University of Minnesota

SIAM - CSE, Atlanta, GA
Feb. 28, 2017

Large eigenvalue problems in applications

➤ Some applications require the computation of a large number of eigenvalues and vectors of very large matrices. These are often found in quantum physics/ chemistry.

➤ Density Functional Theory in electronic structure calculations: *'ground states'*

➤ *Excited states* involve transitions and invariably lead to much more complex computations. $\longrightarrow$ Large matrices, *many* eigenpairs to compute

Illustration:

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

Soving large eigenvalue problems: Current state-of-the art

- Eigenvalues at one end of the spectrum:
  - Subspace iteration + filtering [e.g. FEAST, Cheb,...]
  - Lanczos+variants (no restart, thick restart, implicit restart, Davidson,..), e.g., ARPACK code, PRIMME.
  - Block Algorithms [Block Lanczos, TraceMin, LOBPCG, SLEPc,...]
  - + Many others - more or less related to above
- 'Interior' eigenvalue problems (middle of spectrum):
  - Combine shift-and-invert + Lanczos/block Lanczos. Used in, e.g., NASTRAN

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.

➤ Alternative to shift-and-invert: ‘Spectrum slicing’ with Polynomial filtering

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.

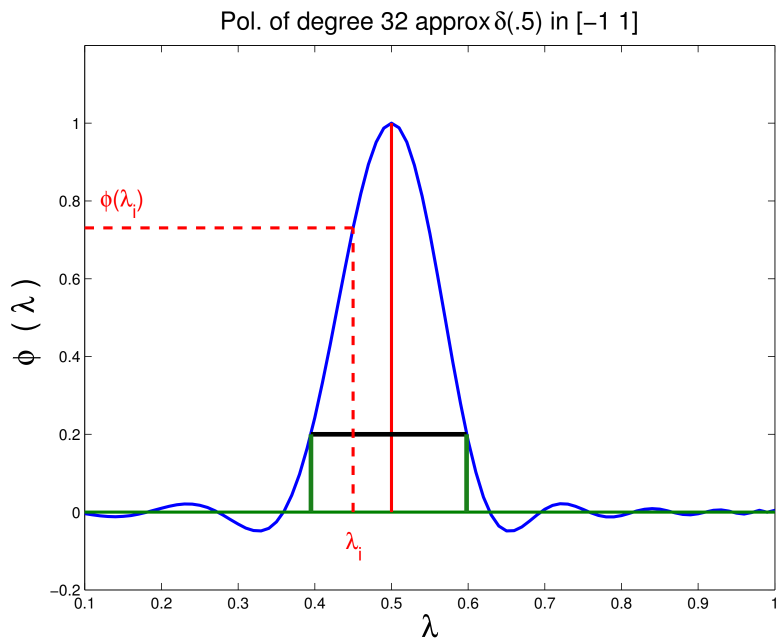
➤ Old (forgotten) idea. But new context is *very* favorable

What polynomials?

- For end-intervals: use standard Chebyshev polynomials
- For inside intervals several methods used:
  - Least- Squares (LS) approximation to a simple spline. [HR Fang and YS $\rightarrow$ *FILTLAN*]
  - Simpler approach: LS approximation to a step function [G. Schofield, J. R. Chelikowsky and YS, '11], [B. Lang et al, 2015]
  - In EVSL we use a simpler scheme yet.
- Observe: There is no real reason to seek to approximate the indicator function $\chi_{[a,b]}$
- Approximate the δ Dirac function instead

LS approximations to δ-Dirac functions

➤ Obtain the LS approximation to the $\delta-$ 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

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

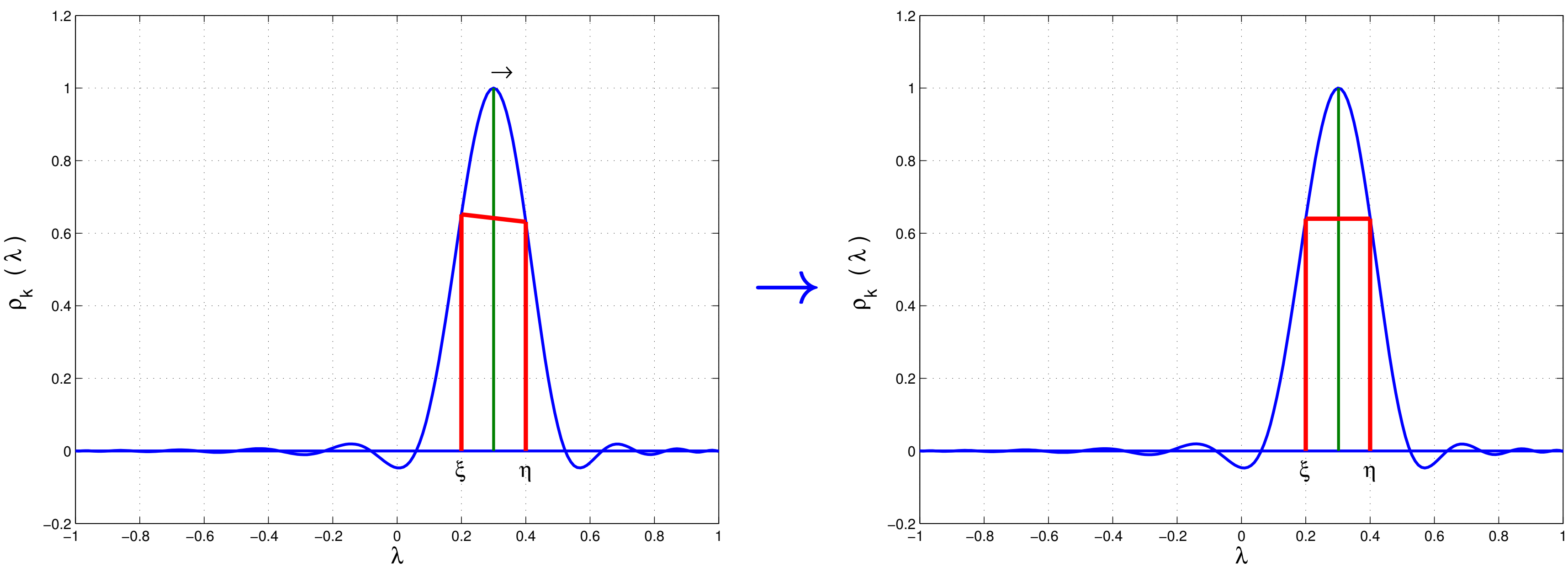
➤ Recall: The delta Dirac function is not a function – we can't properly approximate it in least-squares sense. However:

Proposition Let $\hat{\rho}_k(t)$ be the polynomial that minimizes $\|r(t)\|_w$ over all polynomials r of degree $\leq k$, 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'

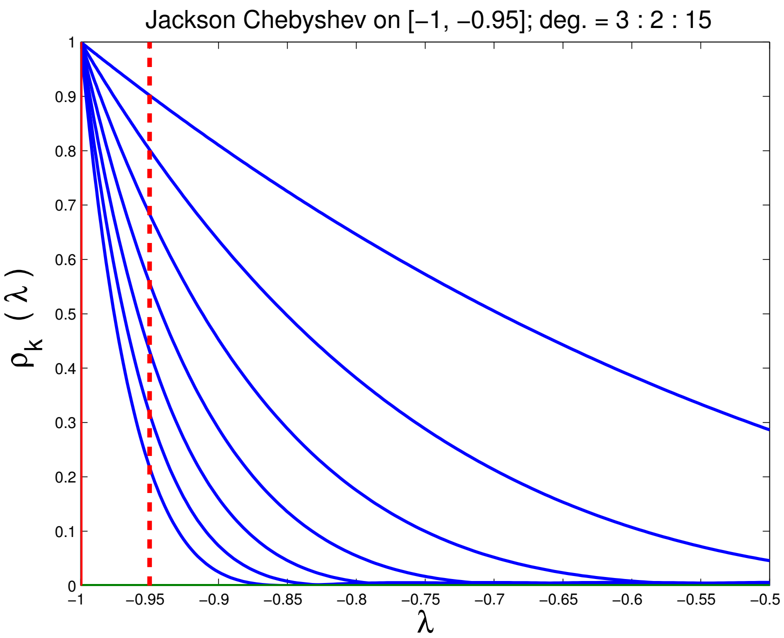
- To facilitate the selection of *'wanted'* eigenvalues [Select $\boldsymbol{\lambda}$'s such that $\rho(\boldsymbol{\lambda}) > \text{bar}$] we need to ...

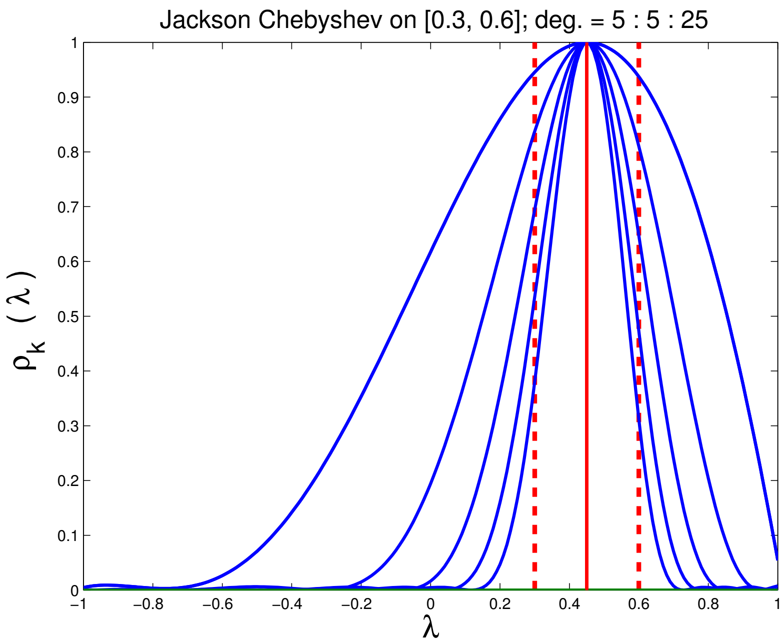
- ... find γ so that $\rho(\xi) == \rho(\eta)$



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)

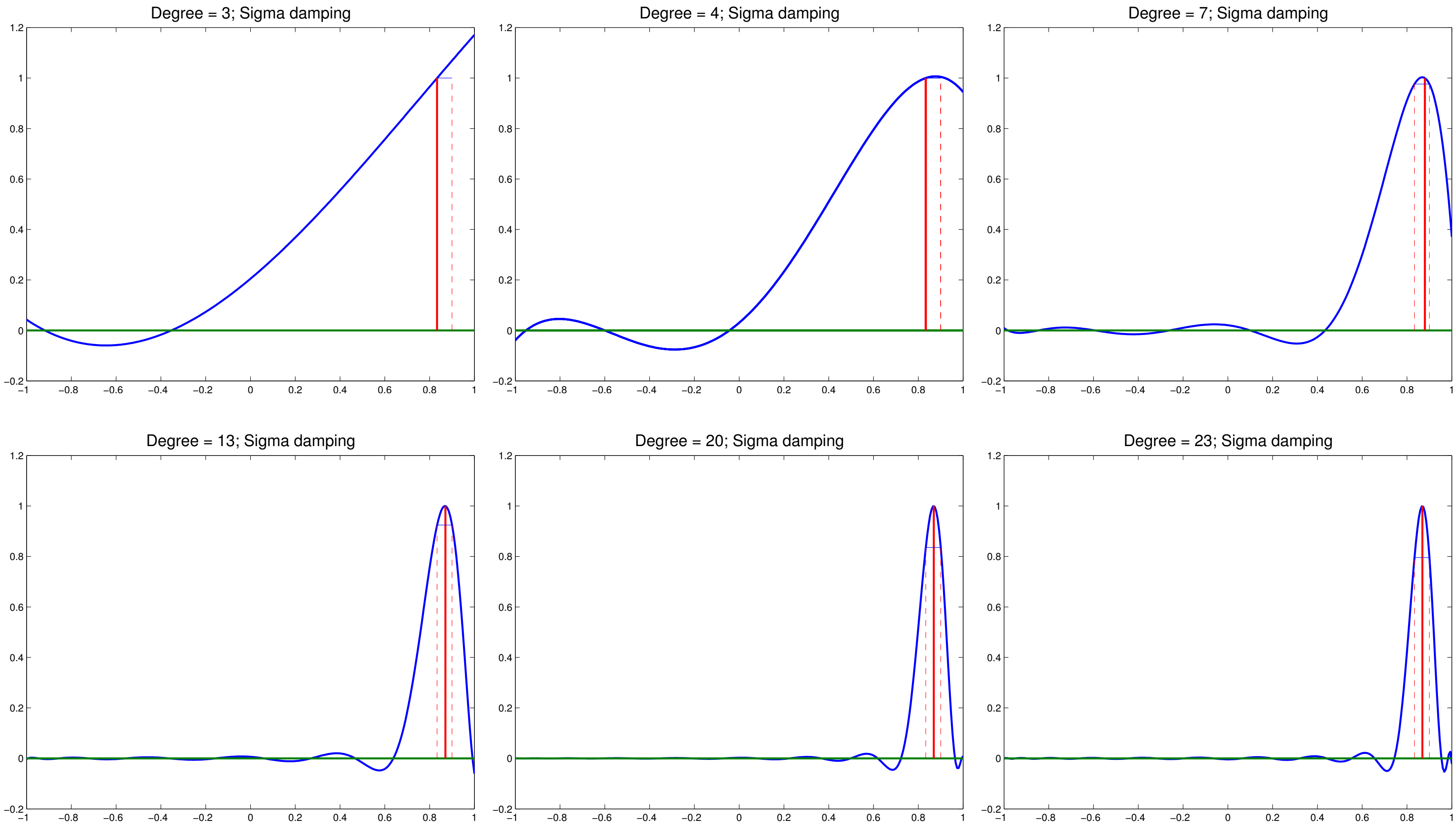




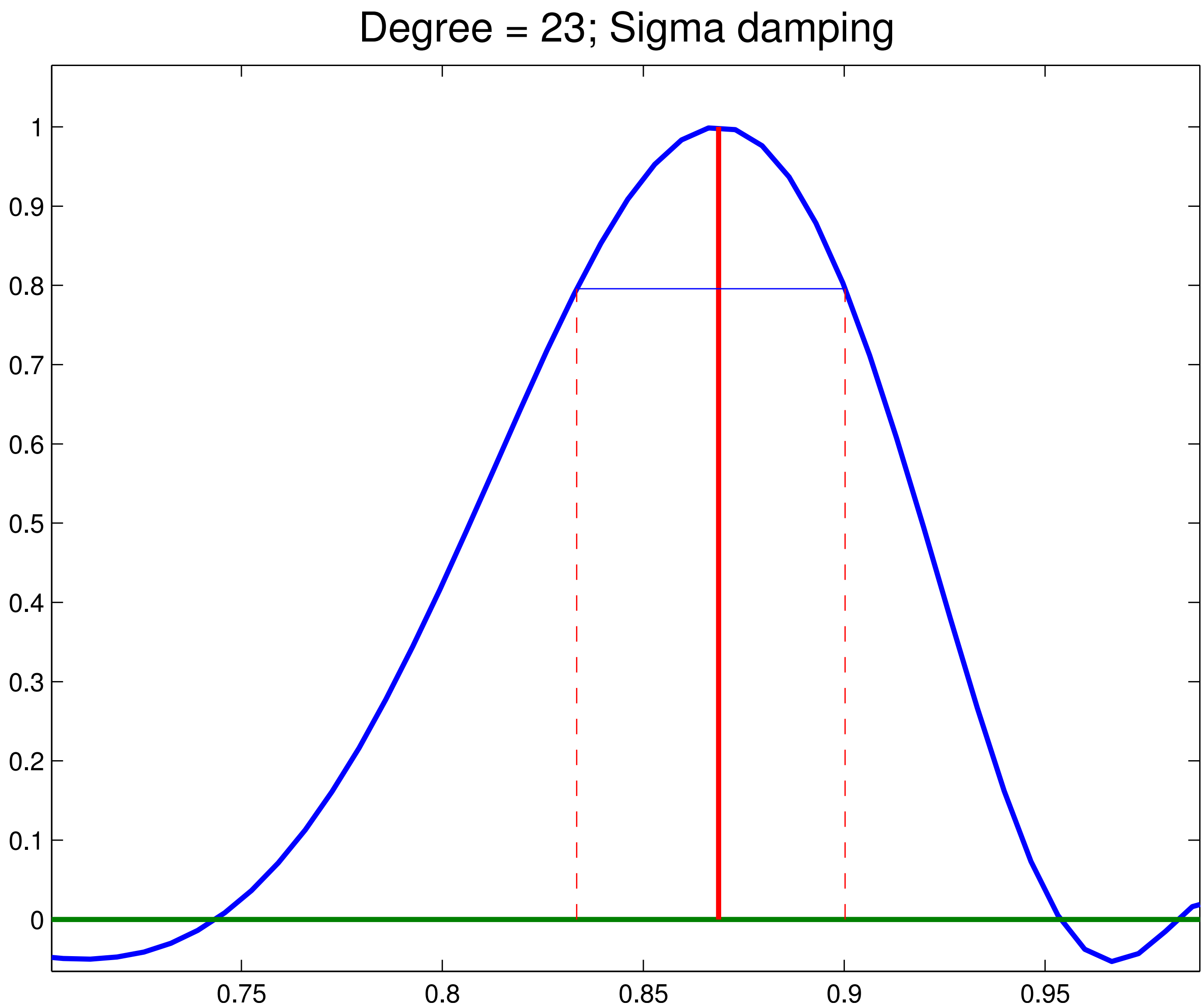
➤ 1) Start low (e.g. 2); 2) Increase degree until value (s) at the boundary (ies) become small enough ...

... and $\rho_k(\lambda)$ provides 'satisfactory' separation

➤ Eventually w'll use criterion based on derivatives at ξ & η



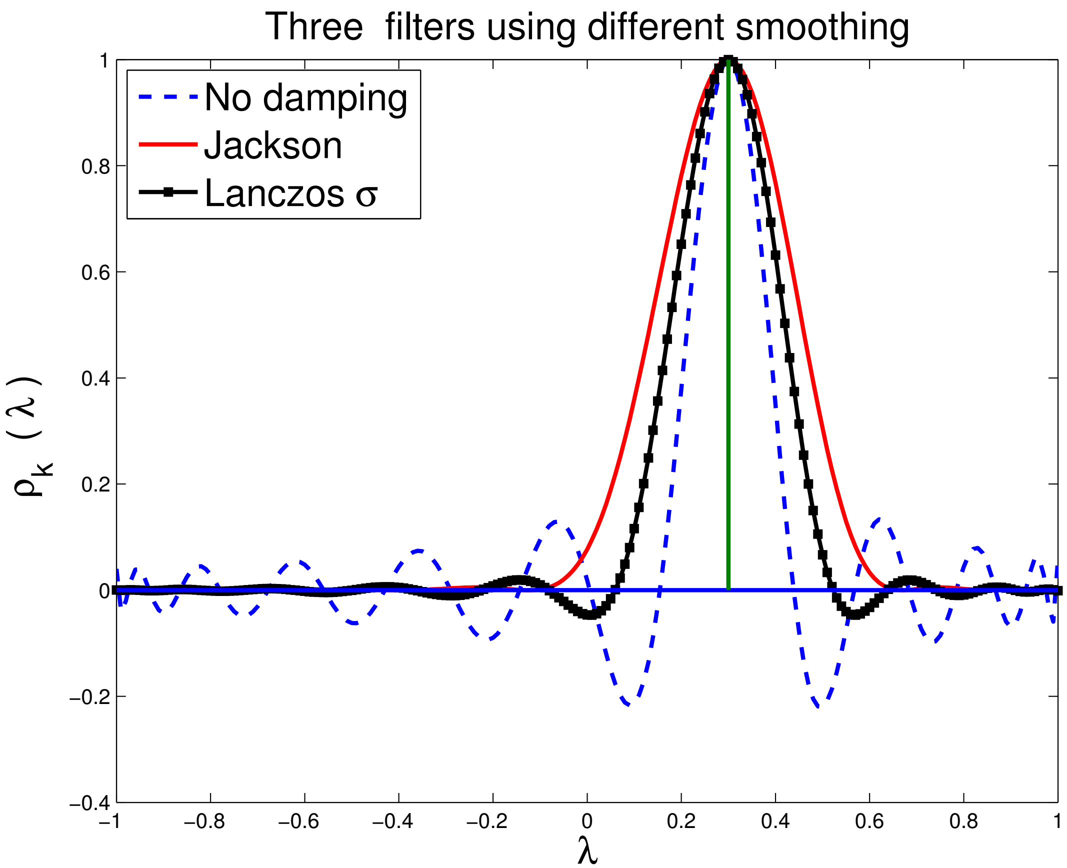
Progress of the *find_pol* algorithm. Interval is [0.833, 0.907..]



A zoom on the final polynomial found

Issue # Three : Gibbs oscillations

➤ Discontinuous 'function' approximated → Gibbs oscillations



➤ Three options:

- No damping
- Jackson damping
- Lanczos σ damping

➤ Good compromise: Lanczos σ damping

Polynomial filtered Lanczos

- Use the Lanczos algorithm with A replaced by $p_k(A)$, where $p_k(t)$ = polynomial of degree k
- Idea not new (and not too popular in the past)

What is new?

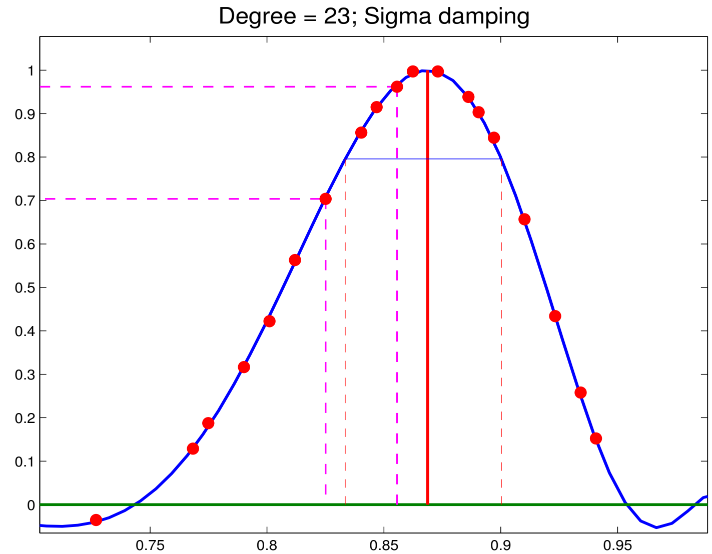
1. Very large problems;
2. (tens of) Thousands of eigenvalues;
3. Parallelism.

- Combine with spectrum slicing
- Main attraction: reduce cost of orthogonalization

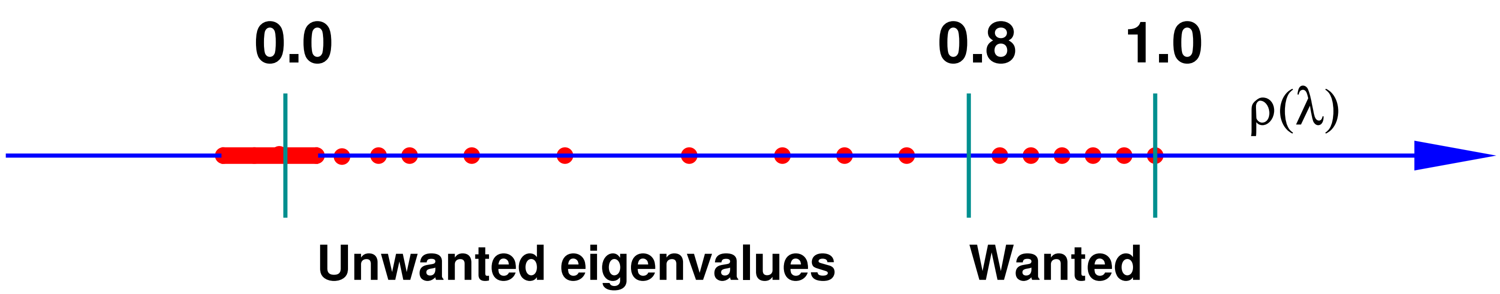
*Hypothetical scenario: large A, *many* wanted eigenpairs*

- 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 $50,000$.

Polynomial filtered Lanczos: No-Restart version

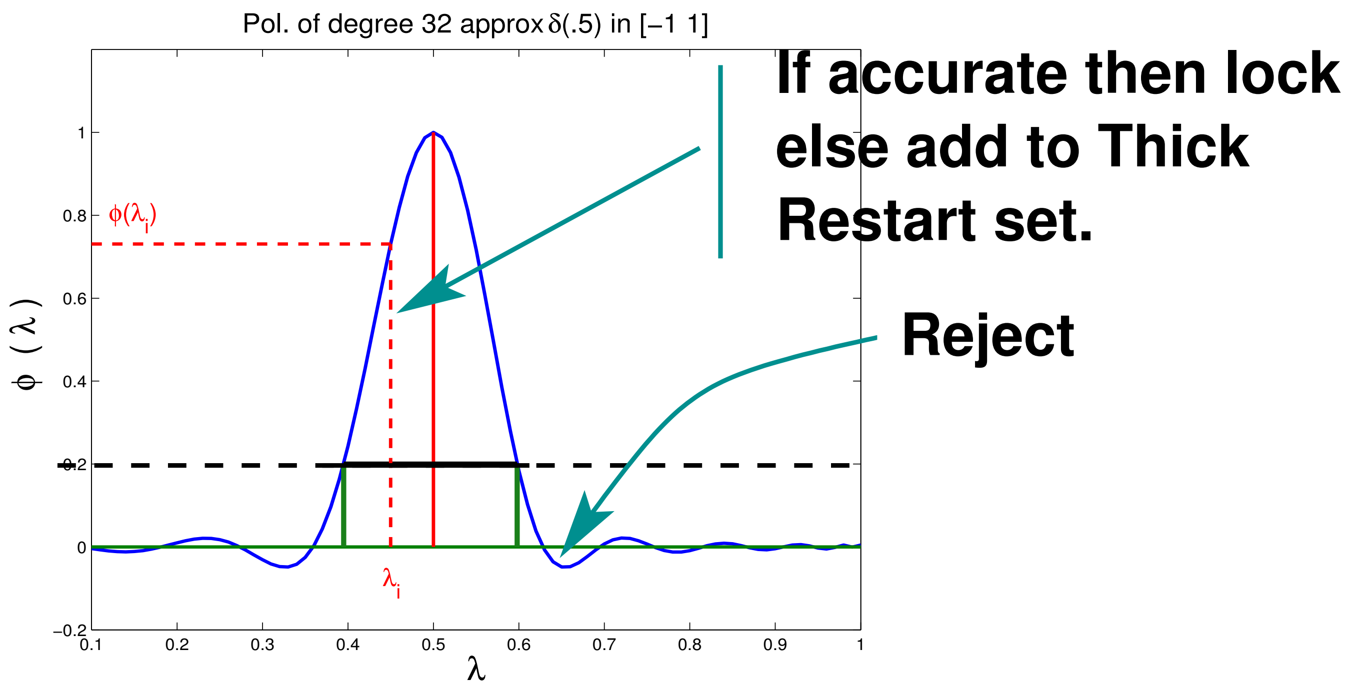


- Use Lanczos with full reorthogonalization on $\rho(A)$. Eigenvalues of $\rho(A)$: $\rho(\lambda_i)$
- Accept if $\rho(\lambda_i) \geq \texttt{bar}$
- Ignore if $\rho(\lambda_i) < \texttt{bar}$



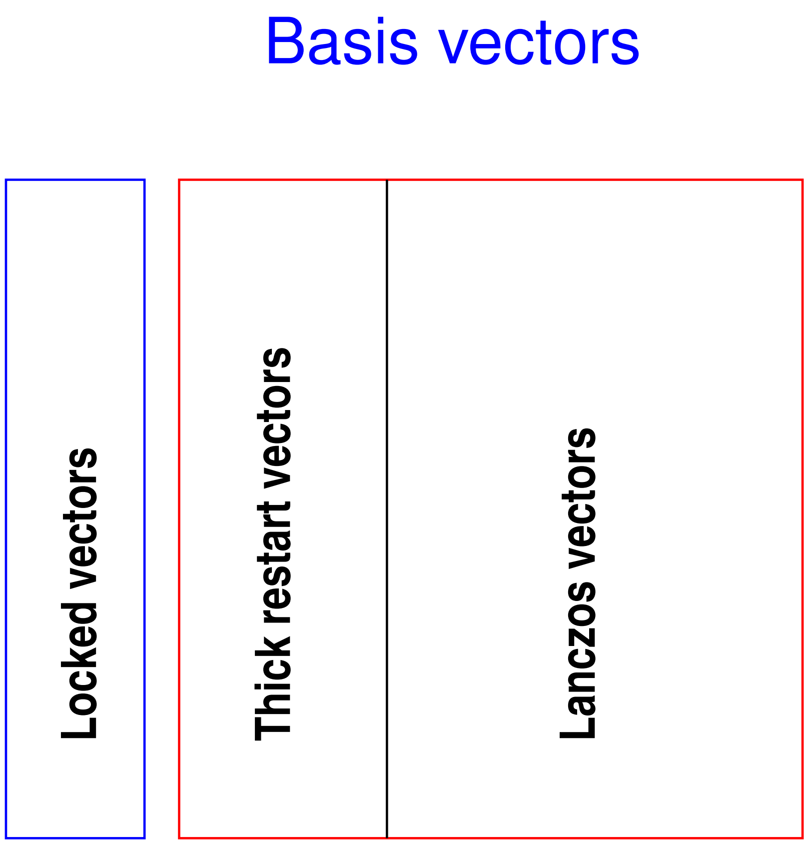
Polynomial filtered Lanczos: Thick-Restart version

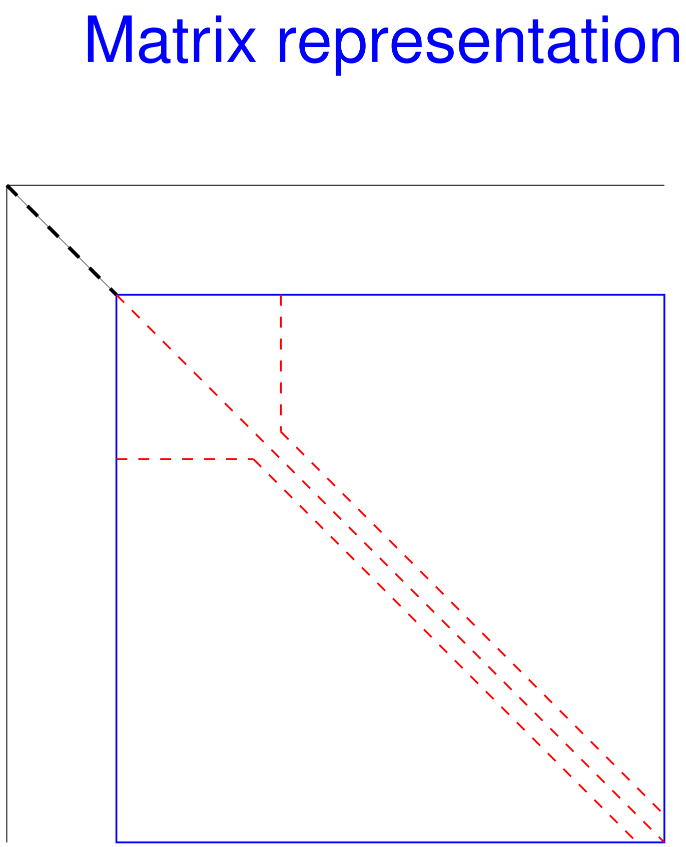
- PolFilt Thick-Restart Lanczos in a picture:



- 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





Experiments:Hamiltonian matrices from PARSEC

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

Results: (Thick-Restart Lanczos)

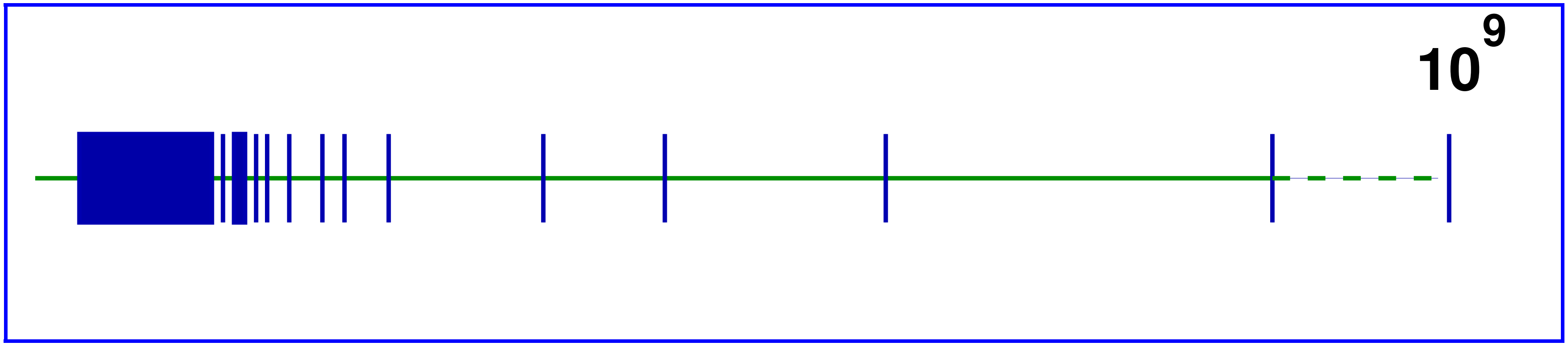
Matrix	deg	iter	matvec	CPU time (sec)		residual	
				matvec	total	max	avg
$\mathbf{Ge_{87}H_{76}}$	26	1431	37482	282.70	395.91	$\mathbf{9.40 \times 10^{-09}}$	$\mathbf{2.55 \times 10^{-10}}$
$\mathbf{Ge_{99}H_{100}}$	26	1615	42330	338.76	488.91	$\mathbf{9.10 \times 10^{-09}}$	$\mathbf{2.26 \times 10^{-10}}$
$\mathbf{Si_{41}Ge_{41}H_{72}}$	35	1420	50032	702.32	891.98	$\mathbf{3.80 \times 10^{-09}}$	$\mathbf{8.38 \times 10^{-11}}$
$\mathbf{Si_{87}H_{76}}$	30	1427	43095	468.48	699.90	$\mathbf{7.60 \times 10^{-09}}$	$\mathbf{3.29 \times 10^{-10}}$
$\mathbf{Ga_{41}As_{41}H_{72}}$	202	2334	471669	8179.51	9190.46	$\mathbf{4.20 \times 10^{-12}}$	$\mathbf{4.33 \times 10^{-13}}$

- Demo with Si10H16 [$\boldsymbol{n = 17,077,\ nnz(A) = 446,500}$]

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

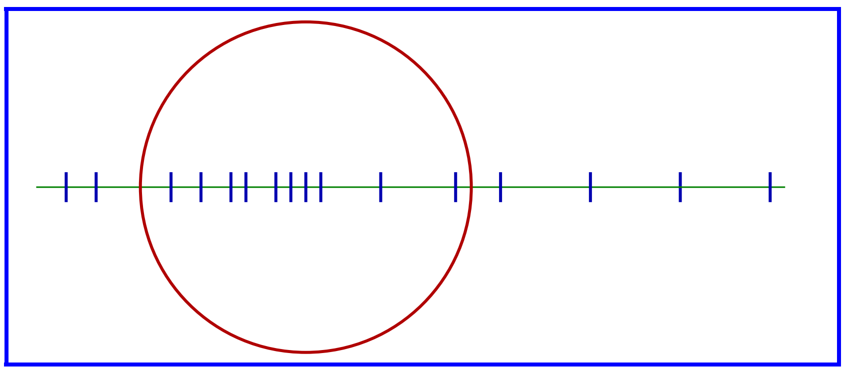
- Generalized eigenvalue problems.

- 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

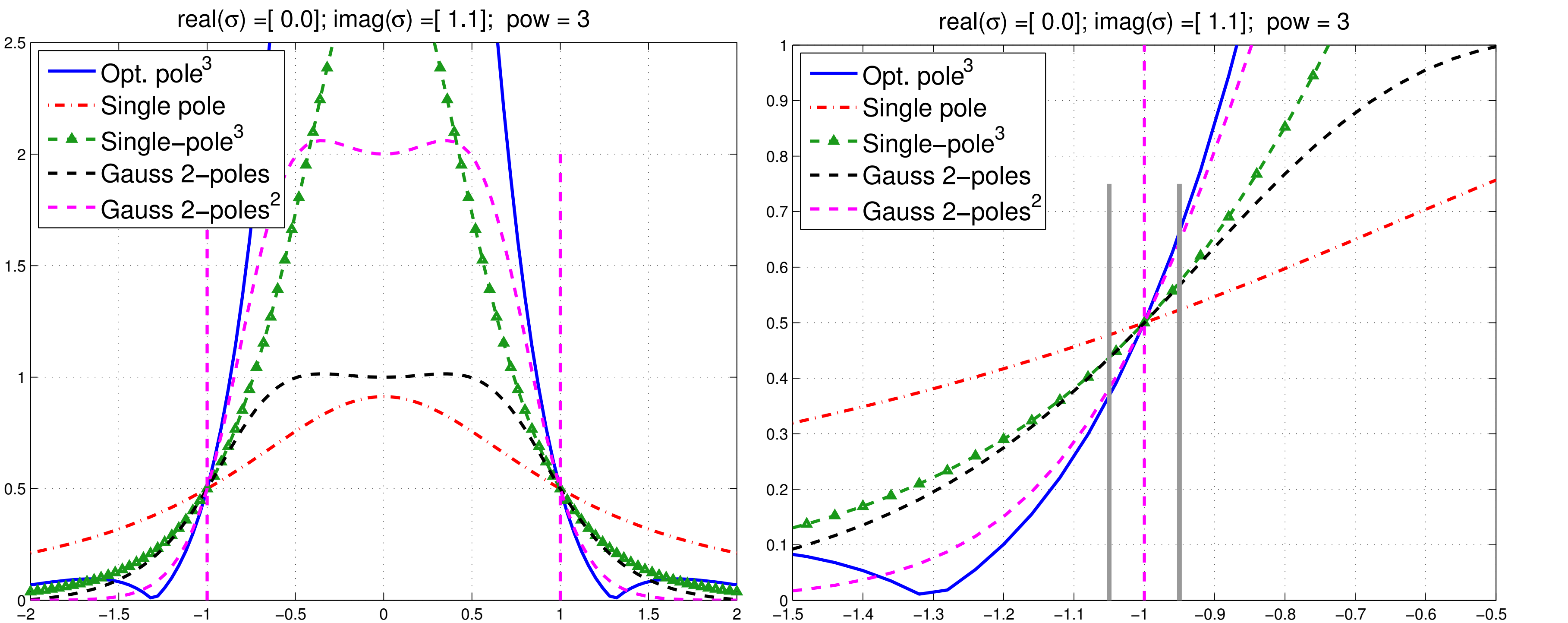


$$P = \frac{-1}{2i\pi} \int_\Gamma (A - sI)^{-1} ds$$

- 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 $a = 10$, $\beta = 0.1$

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

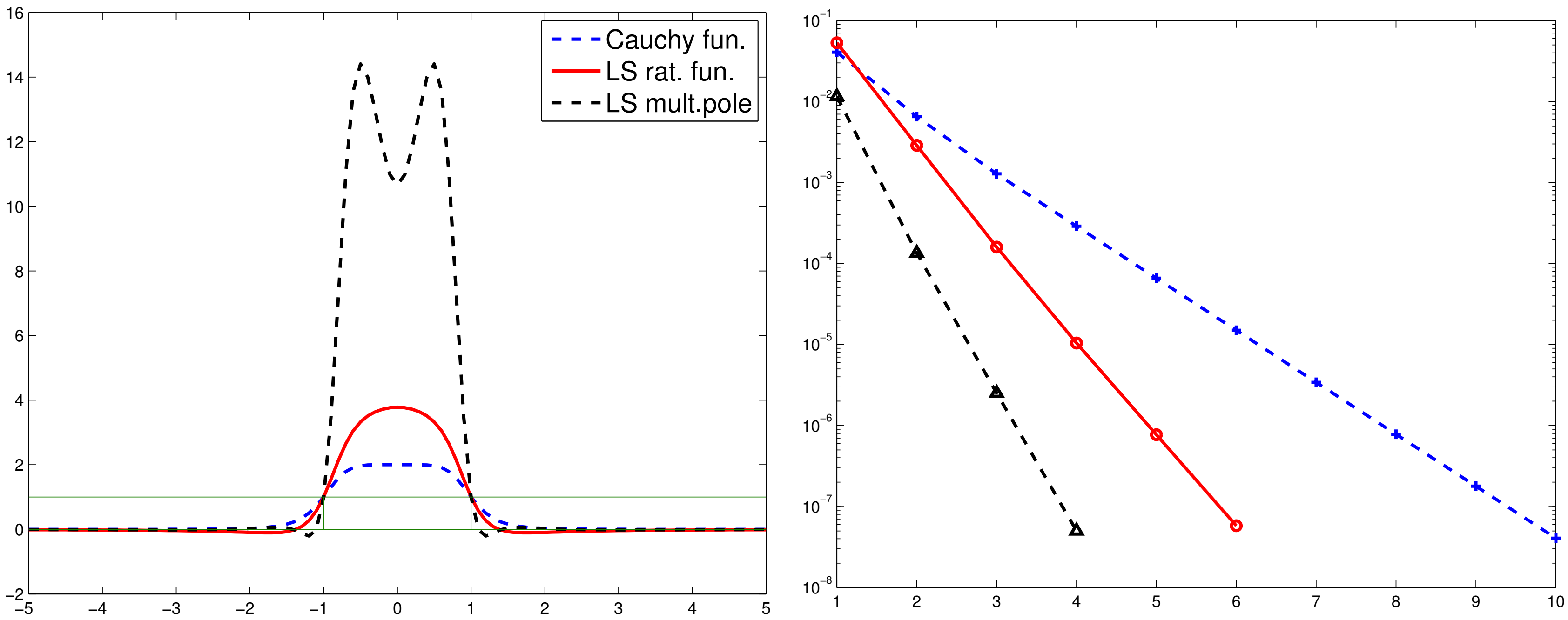
➤ Advantages:

- 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]

Better rational filters: Example

- Take same example as before 43×53 Laplacean
- Now take 6 poles [3×2 midpoint rule]
- Repeat each pole [double poles.]

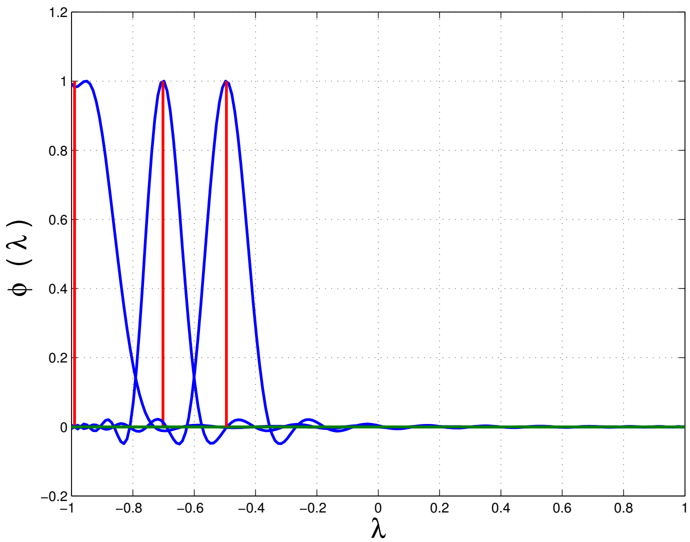


Spectrum Slicing and the EVSL project

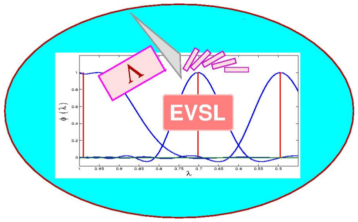
- Newly released EVSL uses polynomial and rational filters
- Each can be appealing in different situations.

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

For each subinterval: select a filter polynomial of a certain degree so its high part captures the wanted eigenvalues. In illustration, the polynomials are of degree 20 (left), 30 (middle), and 32 (right).



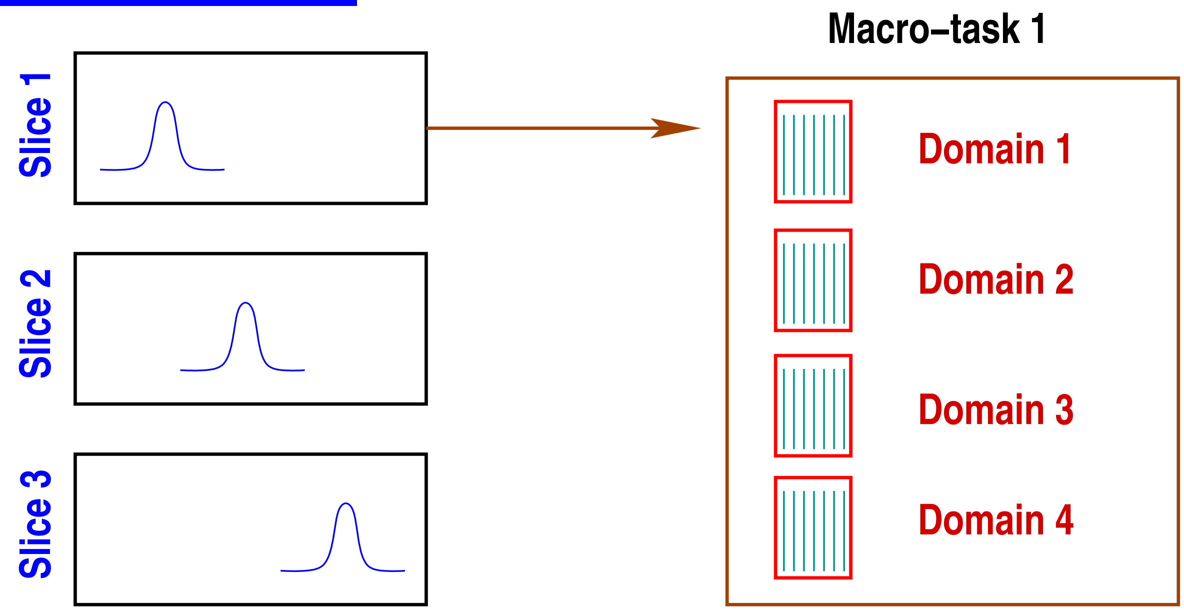
SOFTWARE



EVSL a library of (sequential) eigensolvers based on spectrum slicing. **Version 1.0** released on [09/11/2016]
EVSL provides routines for computing eigenvalues located in a given interval, and their associated eigenvectors, of real symmetric matrices. 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 algorithm (thick restart, no restart) a rational filtered Lanczos algorithm (thick restart, no restart), and a polynomial filtered subspace iteration.

ITSOL a library of (sequential) iterative solvers. **Version 2** released. [11/16/2010]
ITSOL can be viewed as an extension of the ITSOL module in the SPARSKIT package. It is written in C and aims at providing additional preconditioners for solving general sparse linear systems of equations. Preconditioners so far in this package include (1) ILUK (ILU preconditioner with level of fill) (2) ILUT (ILU preconditioner with threshold) (3) ILUC (Crout version of ILUT) (4) VBILUK (variable block preconditioner with level of fill - with automatic block detection) (5) VBILUT (variable block preconditioner with threshold - with automatic block detection) (6) ARMS (Algebraic Recursive Multilevel Solvers -- includes actually several methods - In particular the standard ARMS and the ddPQ version which uses nonsymmetric permutations).
ZITSOL a complex version of some of the methods in ITSOL is also available.

Levels of parallelism



The two main levels of parallelism in EVSL

EVSL: current status

- Released version _1.0 in Fall 2016
  - Matrices in CSR format
- (Very) soon - version _1.1 will add:
  - general `matvec`
  - $Ax = \lambda Bx$
  - Fortran (03) interface.
- Near future:
  - Fully parallel version [MPI + openMP]
  - Challenge application [in progress]

Conclusion

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

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