UNIVERSITY OF MINNESOTA TWIN CITIES

Divide and conquer algorithms for large eigenvalue problems

Yousef Saad

Department of Computer Science and Engineering

University of Minnesota

Applied Math. Seminar

Berkeley, Apr. 22, 2015

Collaborators:

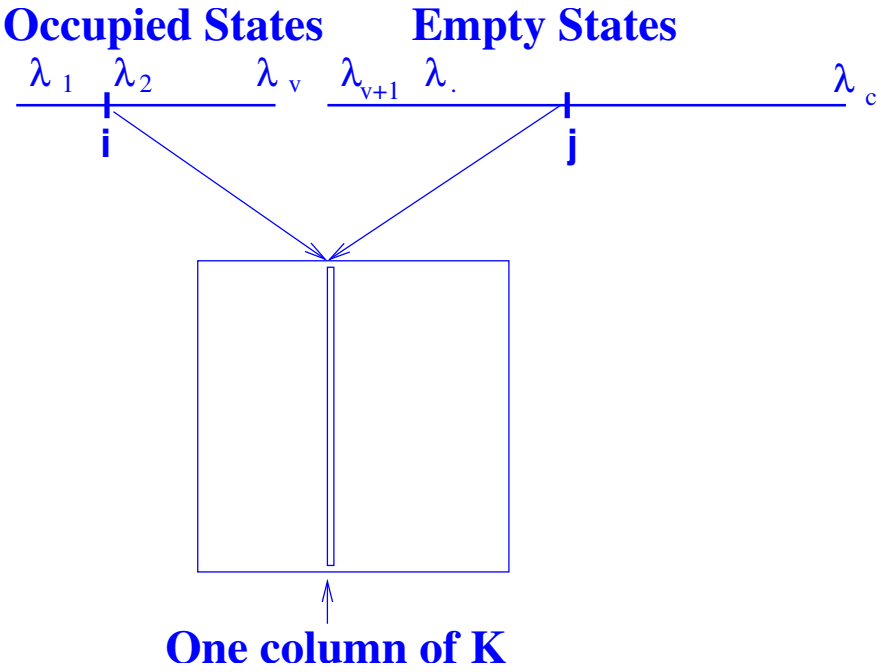
- Joint work with
  - Vassileos Kalantzis [grad student]
  - Ruipeng Li [grad student]
  - Haw-ren Fang [former post-doc]
  - Grady Schoefield and Jim Chelikowsky [UT Austin] [windowing into PARSEC]
- Work supported by DOE : Scalable Computational Tools for Discovery and Design: Excited State Phenomena in Energy Materials [Involves 3 institutions: UT Austin, UC Berkeley, U Minn]

Introduction & Motivation

- Density Functional Theory deals with ground states
- Excited states involve transitions and invariably lead to much more complex computations
- Problem: very large number of eigenpairs to compute

An illustration: Time-Dependent Density Functional Theory (TDDFT). In the so-called Cassida approach we need to compute the eigenvalues of a dense matrix $\boldsymbol{K}$ which is built using both occupied and unocuppied states

- Each pair $\longrightarrow$ one column of K
- To compute each column need to solve a Poisson eqn. on domain
- Intense calculation , lots of parallelism



An illustration: Si34 H36 [Done in 2004-2005]

- Number of Occupied States (Valence band) $n_v = 86$
- Number of Unoccupied States (Conduction band) $n_c = 154$
- Size of Coupling Matrix: $N = n_v * n_c = 13,244$
- Number of Poisson solves = 13,244

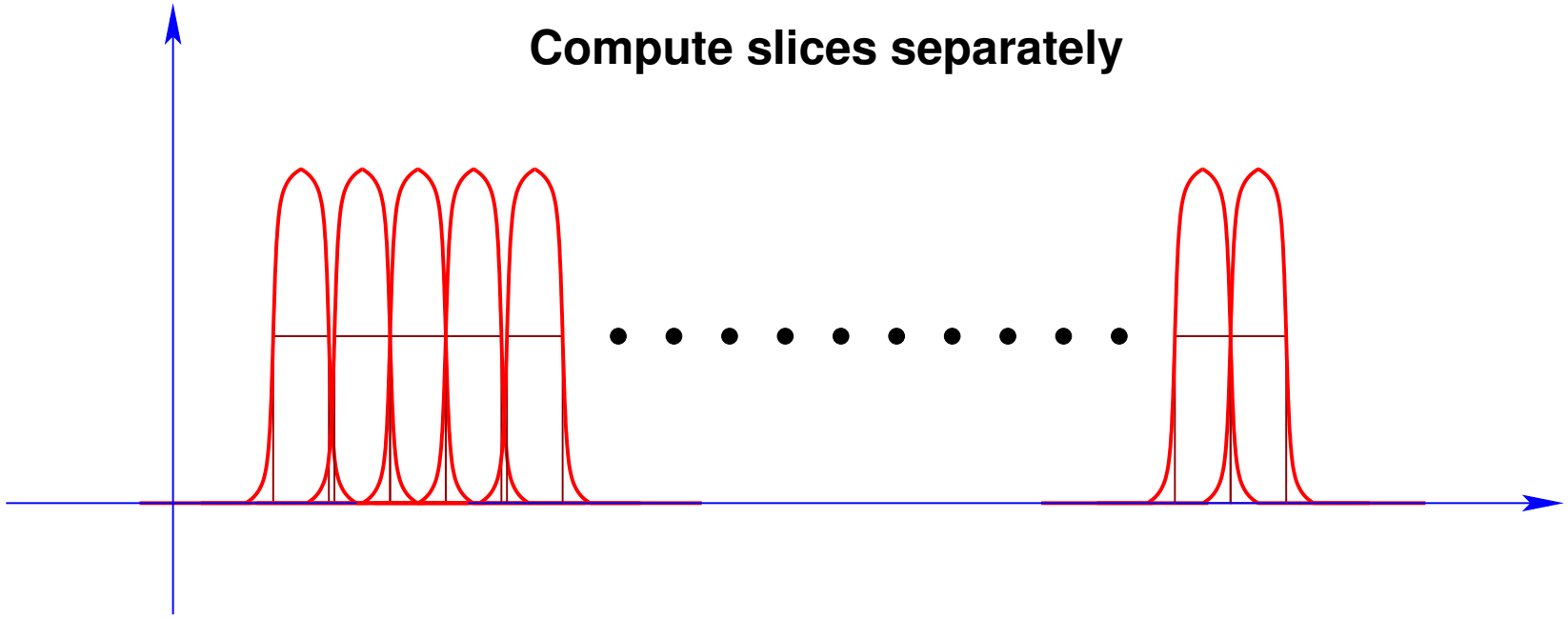
- Similar types of calculations in the GW approach [see, e.g., BerkeleyGW]
- But more complex
- Challenge:

'Hamiltonian of size $\sim$ 1 Million, get 10% of bands'

Solution: Spectrum Slicing

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



- Deceivingly simple looking idea.
- Issues:
  - Deal with interfaces : duplicate/missing eigenvalues
  - Window size [need estimate of eigenvalues]
  - How to compute each slice? [polynomial / rational filters?, ..]

Computing a slice of the spectrum

Q: How to compute eigenvalues in the middle of the spectrum of a large Hermitian matrix?

A: Common practice: Shift and invert + some projection process (Lanczos, subspace iteration..)

Main steps:

1) Select a shift (or sequence of shifts) σ;
2) Factor $A - \sigma I$: $A - \sigma I = LDL^T$
3) Apply Lanczos algorithm to $(A - \sigma I)^{-1}$

- Solves with $A - \sigma I$ carried out using factorization
- Limitation: factorization
- First Alternative: Polynomial filtering

POLYNOMIAL FILTERS

Polynomial filtering

- Apply Lanczos or Subspace iteration to: $M = \phi(A)$ where $\phi(t)$ is a polynomial
- Each matvec $y = Av$ is replaced by $y = \phi(A)v$
- Eigenvalues in high part of filter will be computed first
- Consider Subspace Iteration. In following script:
  - $B = (A - cI)/h$ so $\Lambda(B) \subset [-1,\ 1]$
  - Rayleigh-Ritz with B [shifted-scaled A]
  - Compute residuals w.r.t. B
  - Exit when enough eigenpairs have converged

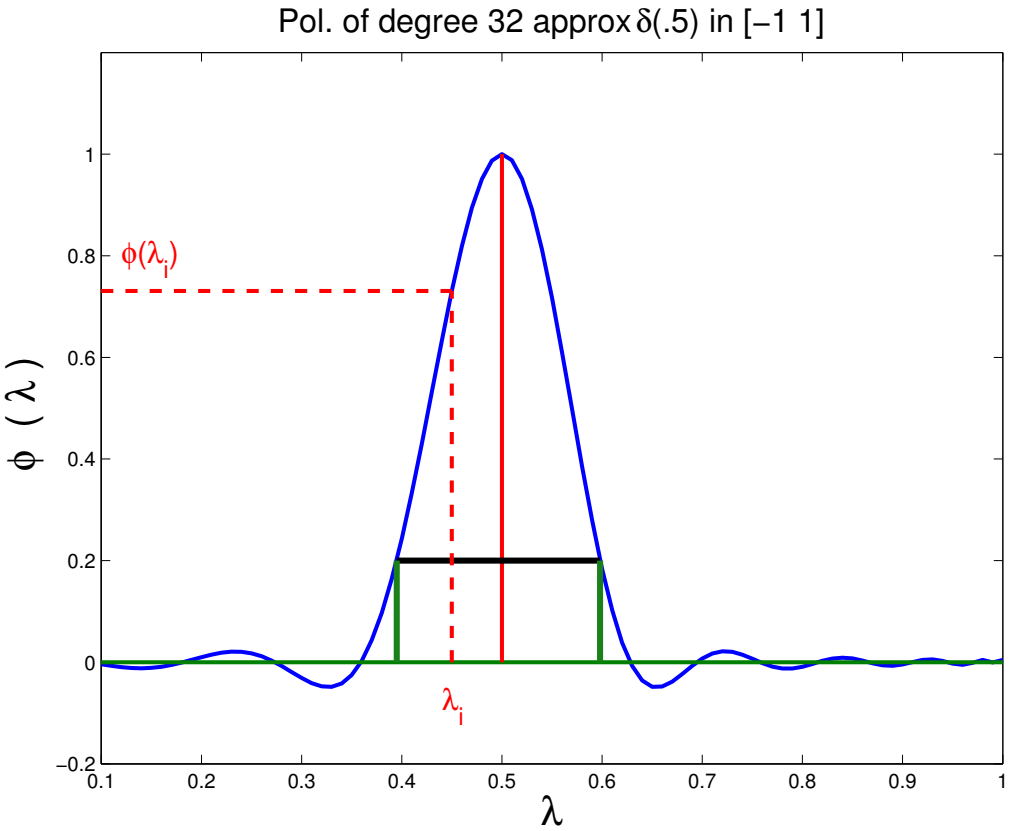
```
for iter=1:maxit
      Y = PolFilt(B, V, mu);
      [V, R] = qr(Y,0);
%— Rayleigh-Ritz with B
      C = V'*(B*V);
      [X, D] = eig(C);
      d = diag(D);
%— get e-values closest to center of interval
      [~, idx] = sort(abs(d-gam),'ascend');
%— get their residuals
      ...
%— if enough pairs have converged exit
      ...
end
```

What polynomials?

- For end-intervals can just use Chebyshev
- For inside intervals: several choices

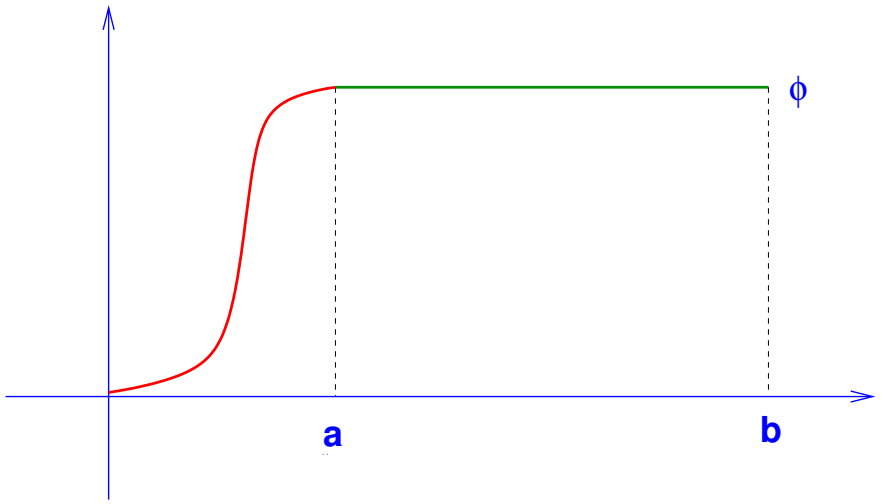
- Recall the main goal:

A polynomial that has large values for $\lambda \in [a, b]$ small values elsewhere



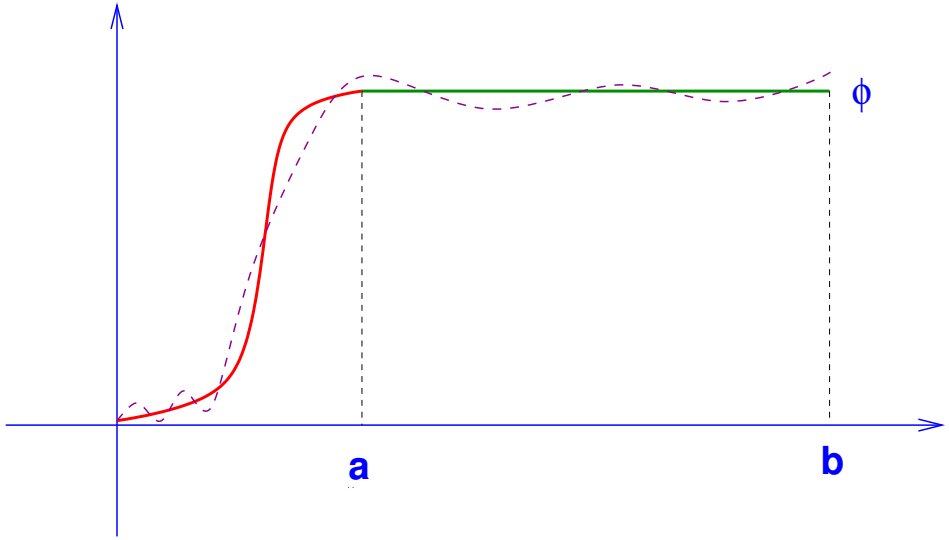
Least-squares approach

- Two stage approach used in filtlan [H-r Fang, YS 2011] -
- First select an "ideal filter"
- e.g., a piecewise polynomial function [a spline]



- For example $\phi =$ Hermite interpolating pol. in [0,a], and $\phi = 1$ in [a, b]
- Referred to as the 'Base filter'

- Then approximate base filter by degree k polynomial in a least-squares sense.
- Can do this without numerical integration

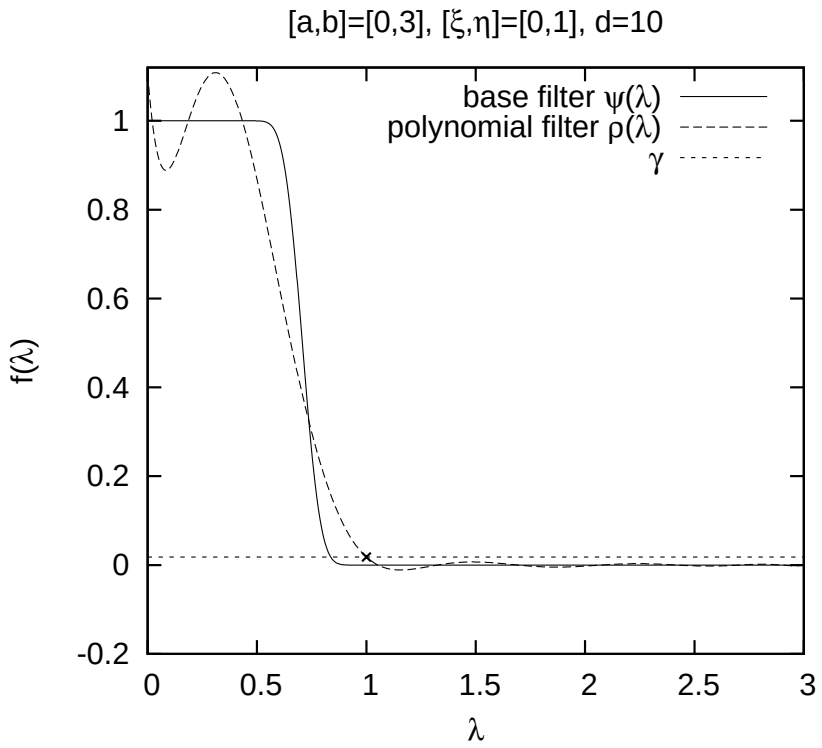


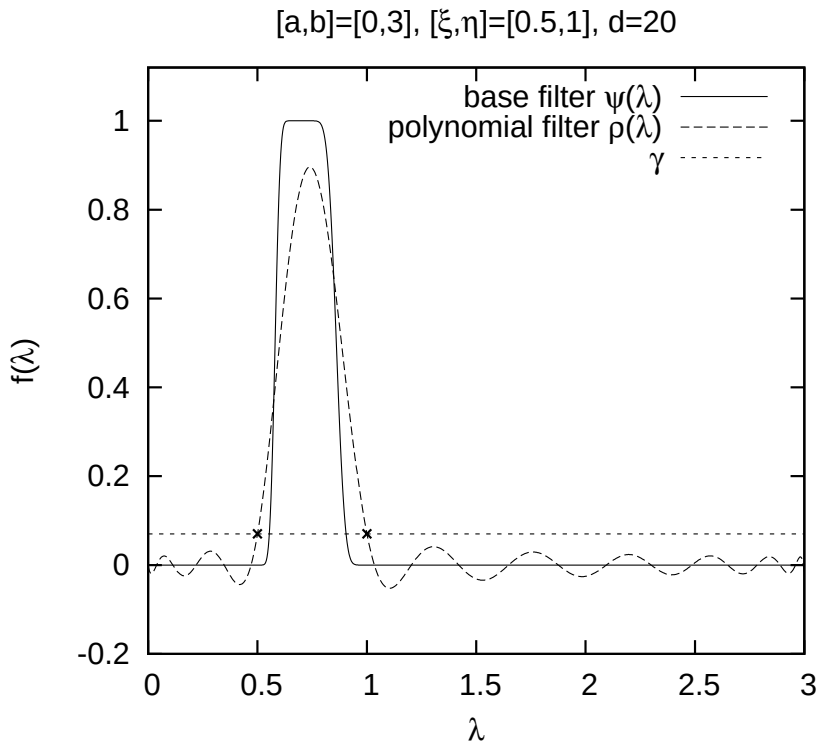
Main advantage: Extremely flexible.

Method: Build a sequence of polynomials ϕ_k which approximate the ideal PP filter ϕ, in the L_2 sense.

➤ In filtlan, we used a form of Conjugate Residual technique in polynomial space. Details skipped.

Low-pass, high-pass, & barrier (mid-pass) filters

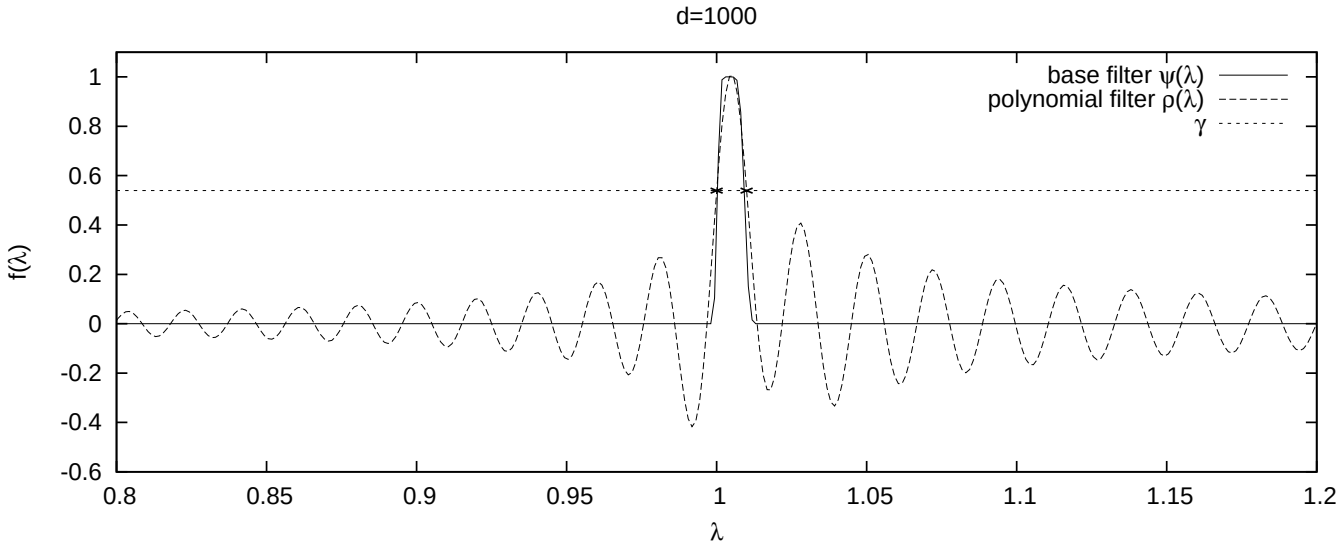




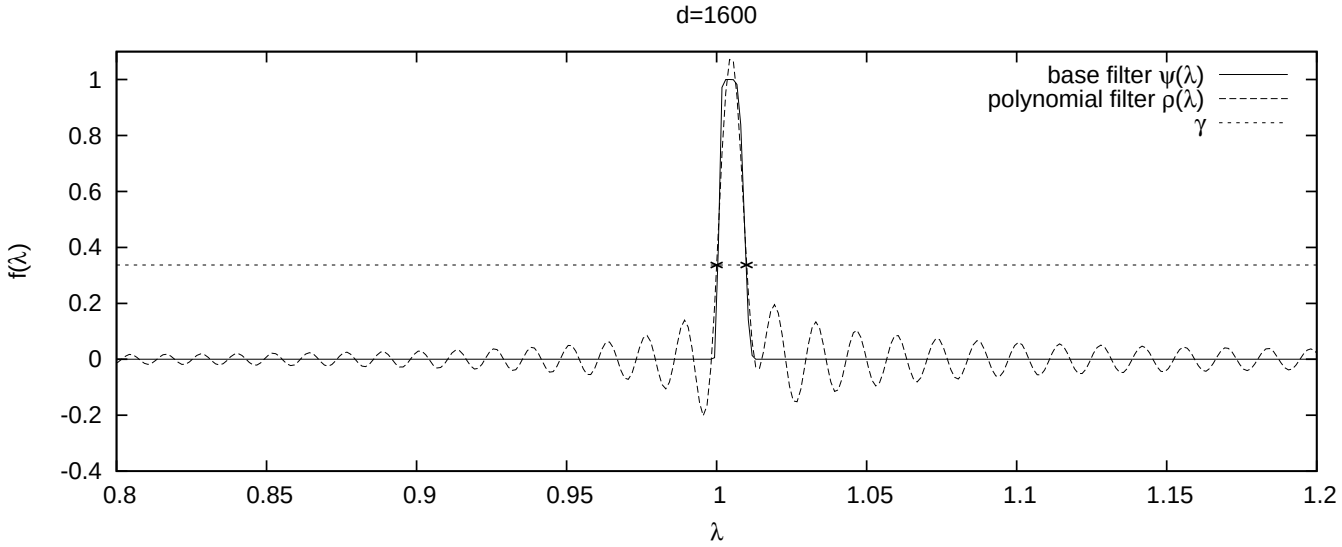
- See Reference on Lanczos + pol. filtering: Bekas, Kokiopoulou, YS (2008) for motivation, etc.
- H.-r Fang and YS "Filtlan" paper [SISC,2012] and code

Misconception: High degree polynomials are bad

Degree 1000 (zoom)



Degree 1600 (zoom)



A simpler approach: Chebyshev + Jackson damping

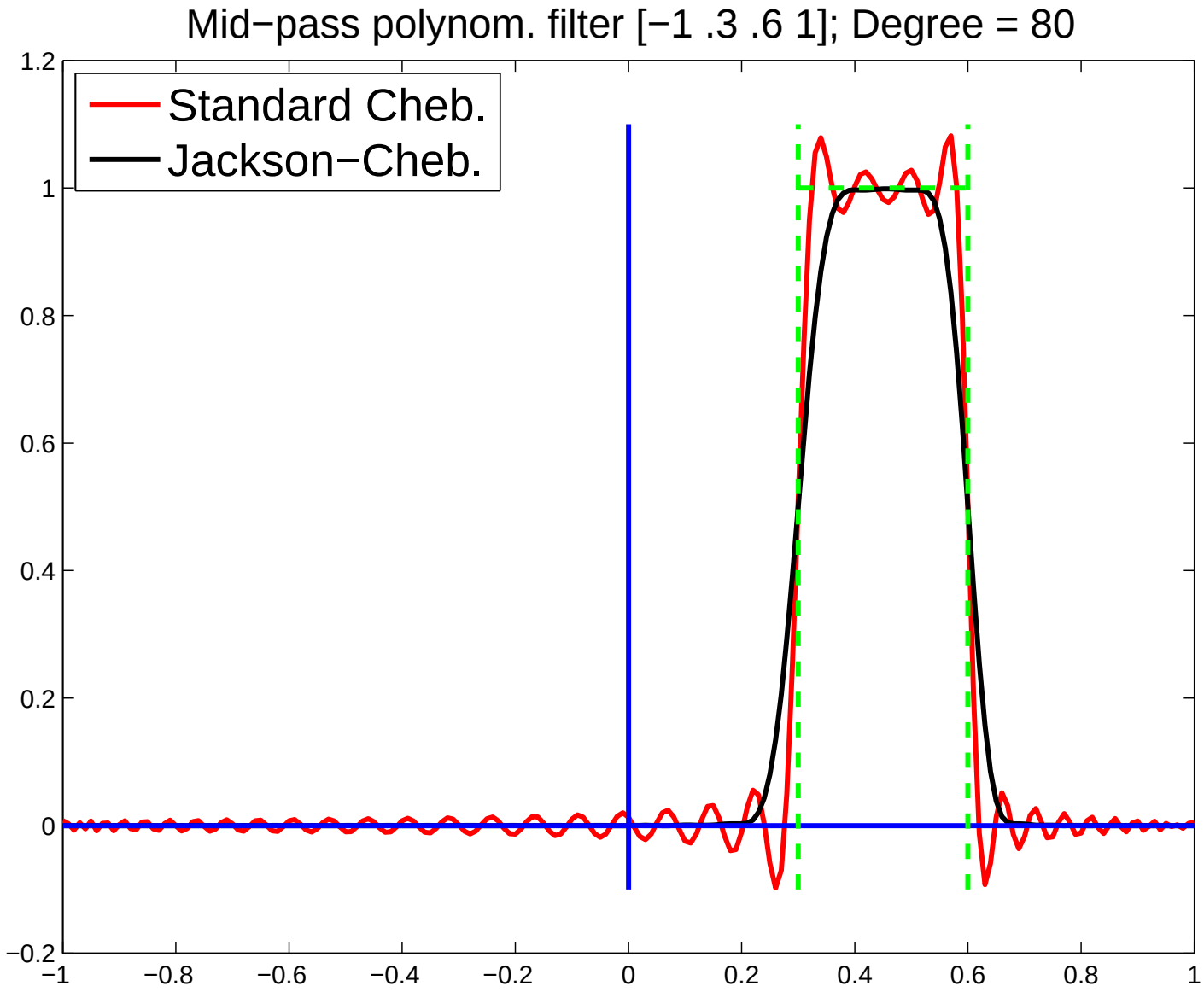
- Simply seek the best LS approximation to step function
- Add damping coefficients to reduce oscillations

Chebyshev-Jackson approximation of a function f:

$$f(x) \approx \sum_{i=0}^{k} g_i^k \gamma_i T_i(x)$$

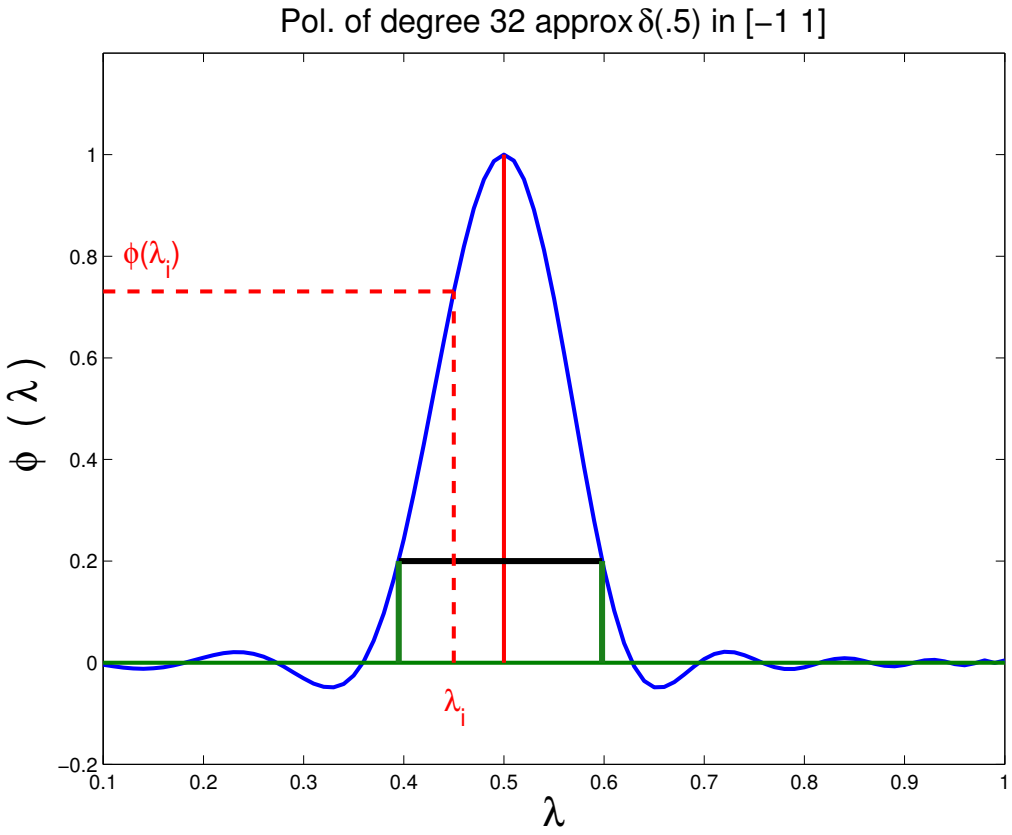
$$\gamma_i = \begin{cases} [\arccos(a) - \arccos(b)]/\pi : i = 0 \\ 2\,[\sin(i\arccos(a)) - \sin(i\arccos(b))]/(i\pi) : i > 0 \end{cases}$$

- Expression for g_i^k: see L. O. Jay, YS, J. R. Chelikowsky, '99

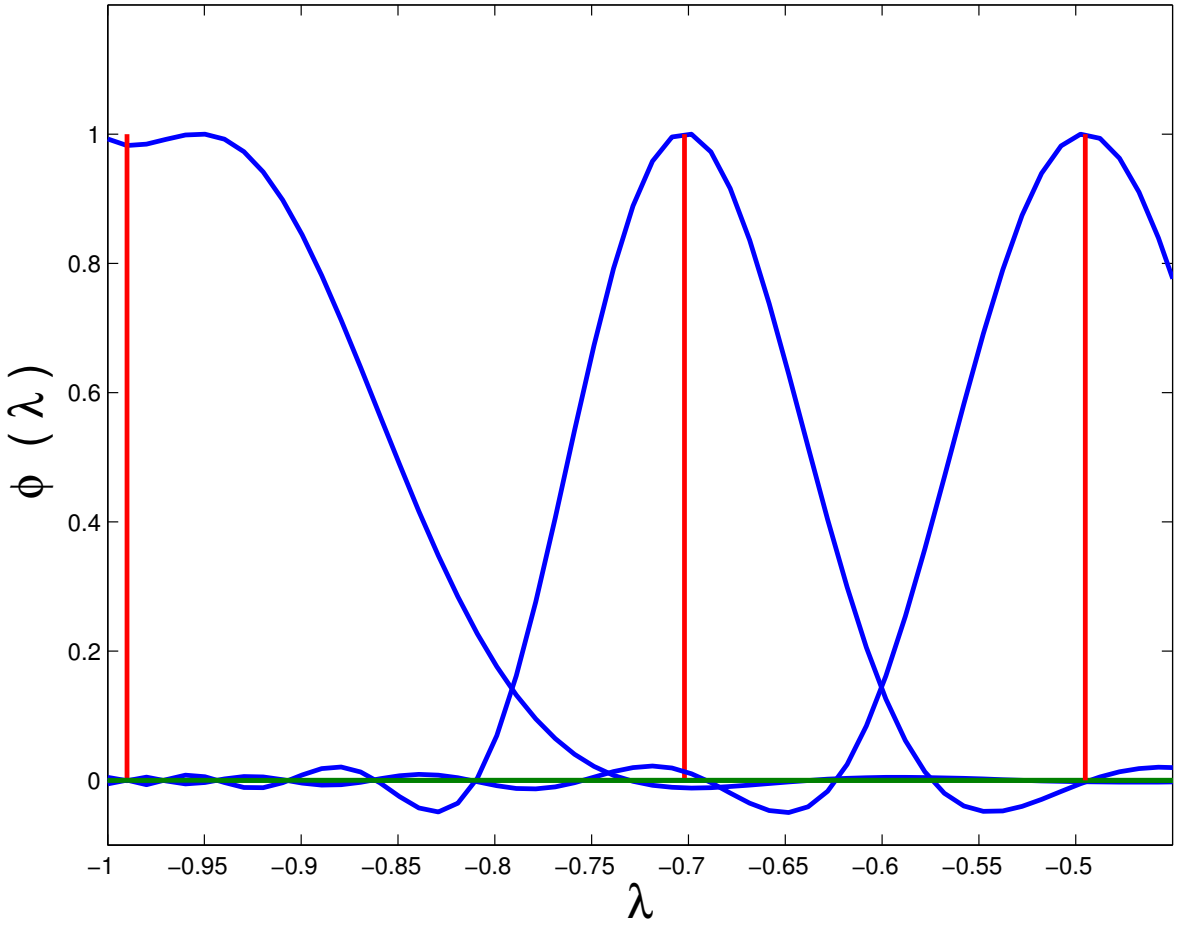

Mid−pass polynom. filter [−1 .3 .6 1]; Degree = 80
Standard Cheb.
Jackson−Cheb.
1.2
1
0.8
0.6
0.4
0.2
0
−0.2
−1
−0.8
−0.6
−0.4
−0.2
0
0.2
0.4
0.6
0.8
1

Polynomial filters: An even simpler approach

- Simply seek the LS approximation to the δ – Dirac function
- Centered at the middle of the interval.
- Can use same damping: Jackson, or Lanczos σ damping.



Filters for 3 slices



Tests – Test matrices

** From Filtlan article with H-R Fang

➤ Experiments on two dual-core AMD Opteron(tm) Processors 2214 @ 2.2GHz and 16GB memory.

Test matrices:

* Five Hamiltonians from electronic structure calculations,

* `Andrews` matrix $N = 60,000$, $nnz \approx 760K$, interval $[4, 5]$; nev=1,844 eigenvalues, (3,751 to the left of η)

* A discretized Laplacian (FD) $n = 10^6$, interval = $[1,\ 1.01]$, nev= 276, (>17,000 on the left of η)

➤ Here : report only on Andrews and Laplacean

Results for Andrews - set 1 of stats

method	degree	# iter	# matvecs	memory
filt. Lan. (mid-pass)	$d = 20$	9,440	188,800	4,829
	$d = 30$	6,040	180,120	2,799
	$d = 50$	3,800	190,000	1,947
	$d = 100$	2,360	236,000	1,131
filt. Lan. (high-pass)	$d = 10$	5,990	59,900	2,799
	$d = 20$	4,780	95,600	2,334
	$d = 30$	4,360	130,800	2,334
	$d = 50$	4,690	234,500	2,334
Part. $\perp$ Lanczos		22,345	22,345	10,312
ARPACK		30,716	30,716	6,129

Results for Andrews - CPU times (sec.)

method	degree	$\rho(A)v$	reorth	eigvec	total
filt. Lan. (mid-pass)	$d = 20$	2,797	192	4,834	9,840
	$d = 30$	2,429	115	2,151	5,279
	$d = 50$	3,040	65	521	3,810
	$d = 100$	3,757	93	220	4,147
filt. Lan. (high-pass)	$d = 10$	1,152	2,911	2,391	7,050
	$d = 20$	1,335	1,718	1,472	4,874
	$d = 30$	1,806	1,218	1,274	4,576
	$d = 50$	3,187	1,032	1,383	5,918
Part. $\perp$ Lanczos		217	30,455	64,223	112,664
ARPACK		345	†423,492	†18,094	441,934

Results for Laplacian – Matvecs and Memory

method	degree	# iter	# matvecs	memory
	600	1,400	840,000	10,913
mid-pass filter	$1,000$	950	950,000	7,640
	$1,600$	710	1,136,000	6,358

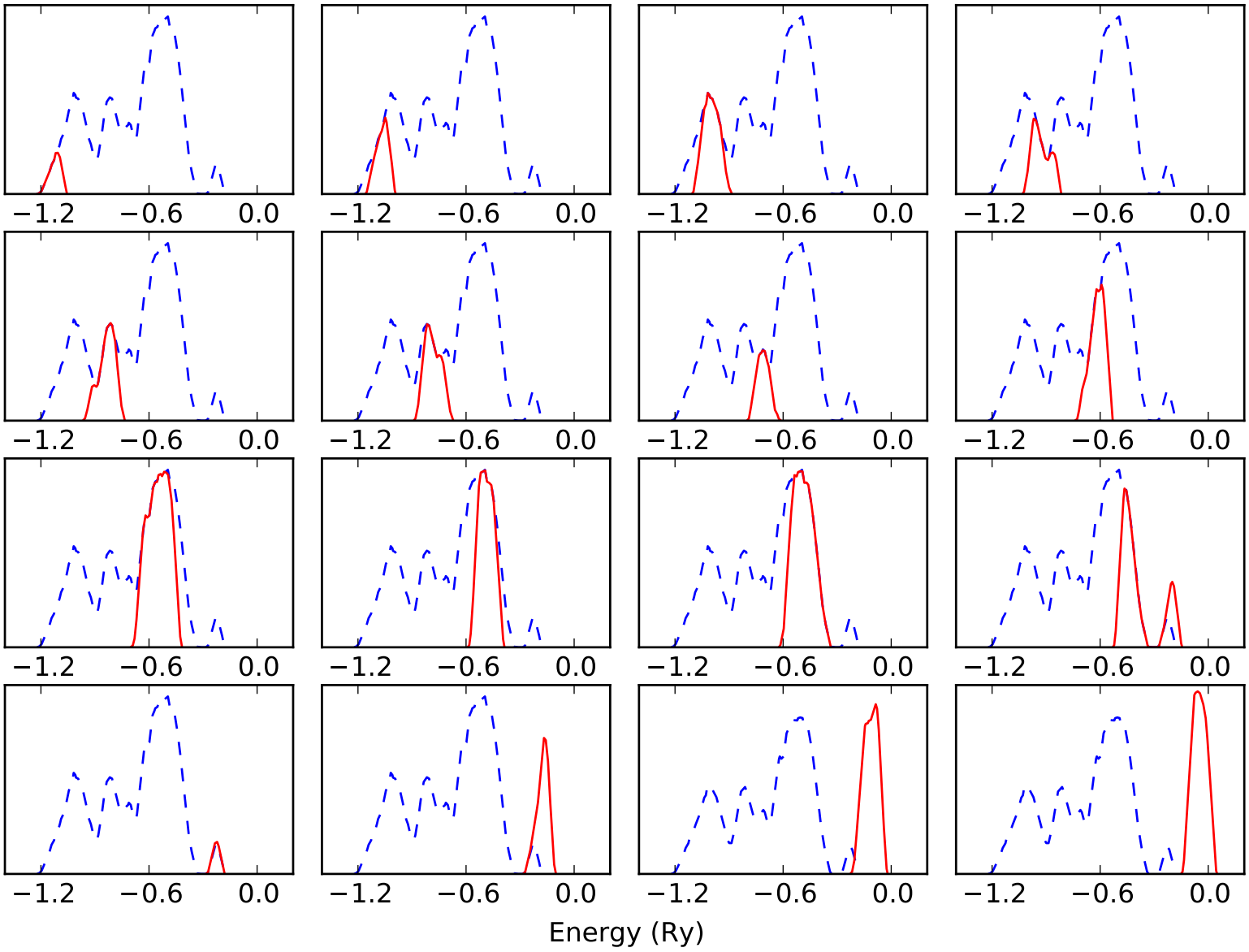
Results for Laplacian – CPU times

method	degree	$\rho(A)v$	reorth	eigvec	total
	600	97,817	927	241	99,279
mid-pass filter	$1,000$	119,242	773	162	120,384
	$1,600$	169,741	722	119	170,856

Spectrum slicing in PARSEC

** From: A Spectrum Slicing Method for the Kohn-Sham Problem, G. Schofield, J. R. Chelikowsky and YS, Computer Physics Comm., vol 183 (2011) pp. 487-505.

- Preliminary implementation in our code: *PARSEC*
- Uses the simpler Jackson-Chebyshev filters
- For details on windowing, etc., see paper
- Illustration shown next using 16 slices: *States for unrelaxed* $Si_{275}H_{172}$ *as computed using 16 slices. Each slice* $\rightarrow$ *solid line. Original full spectrum* $\rightarrow$ *dashed line.*

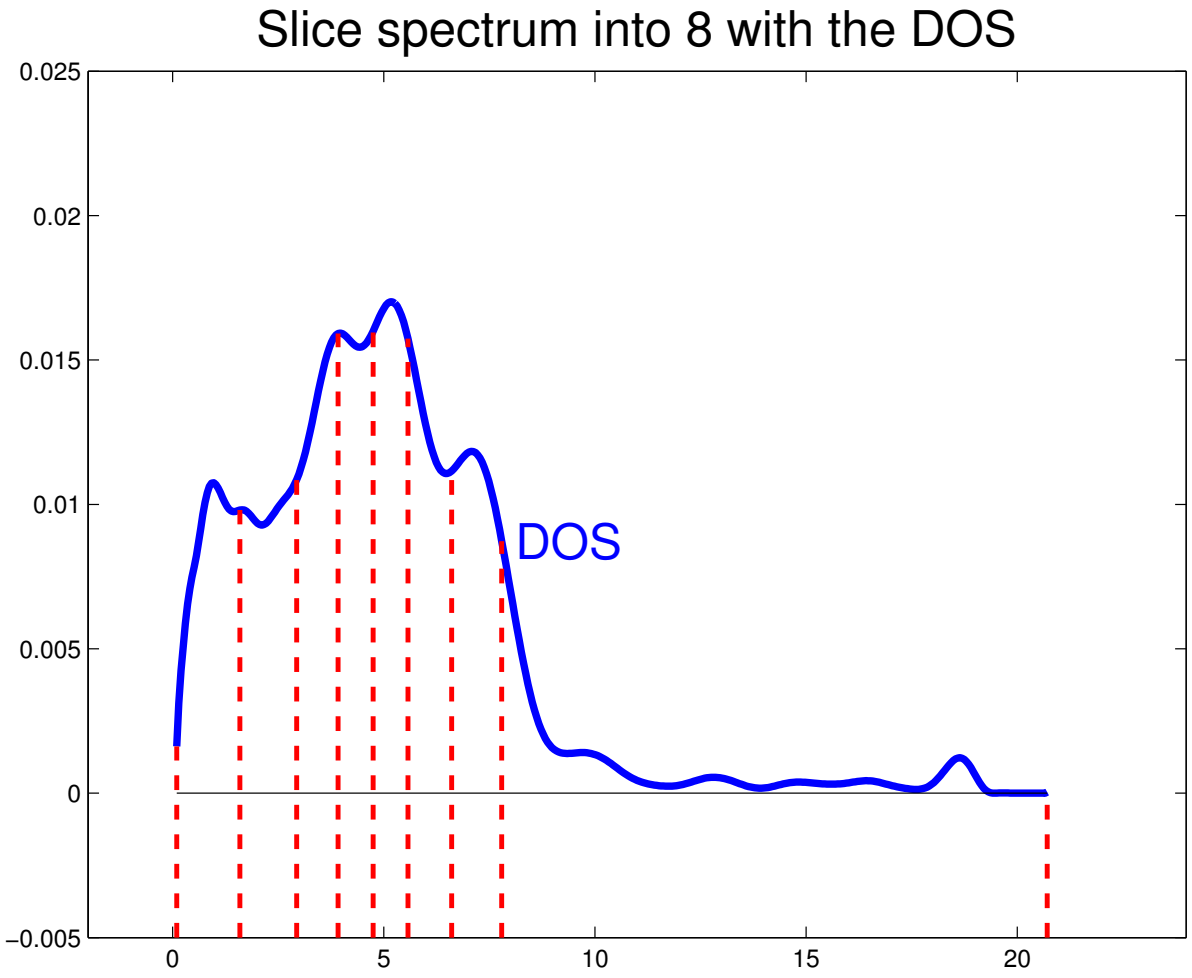

−1.2
−0.6
0.0
Energy (Ry)

How do I slice my spectrum?

Analogue question:

How would I slice an onion if I want each slice to have about the same mass?

- A good tool: Density of States – see:
  - L. Lin, YS, Chao Yang recent paper.
  - KPM method – see, e.g., : [Weisse, Wellein, Alvermann, Fehske, '06]
  - Interesting instance of a tool from physics used in linear algebra.
- Misconception: 'load balancing will be assured by just having slices with roughly equal numbers of eigenvalues'
- Situation is much more complex



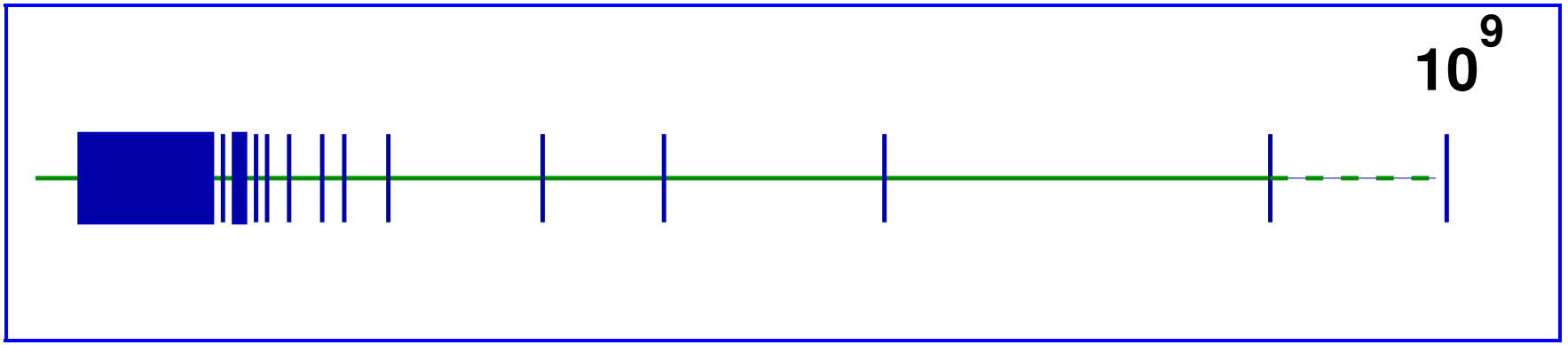
- We must have:

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

RATIONAL FILTERS

Why use rational filters?

- Consider a spectrum like this one:



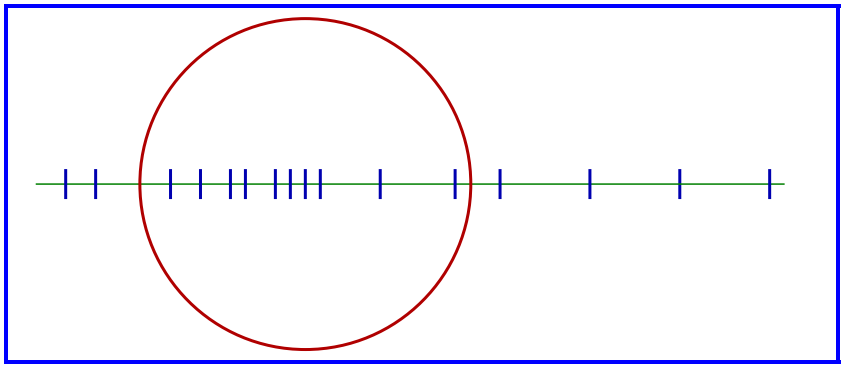
- Polynomial filtering approach would be utterly ineffective for this case

- Second issue: situation when Matrix-vector products are expensive

- Alternative is to use rational filters:

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

- 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]
- Polizzi [FEAST, Subsp. Iter.]

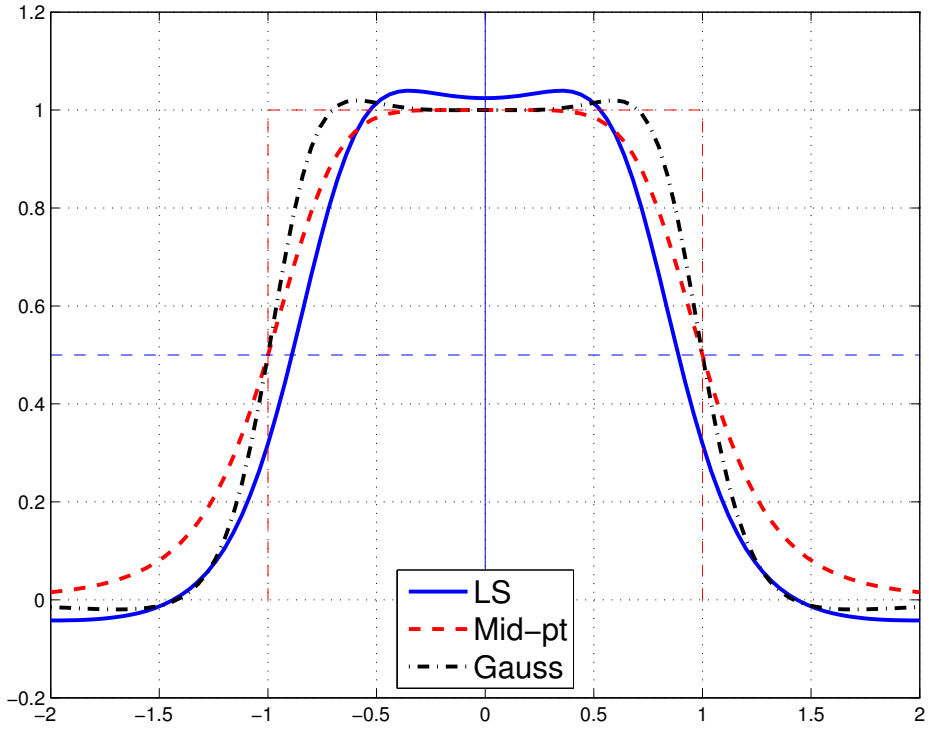
Better rational filters

➤ When using rational filters often one resorts to Cauchy integrals :

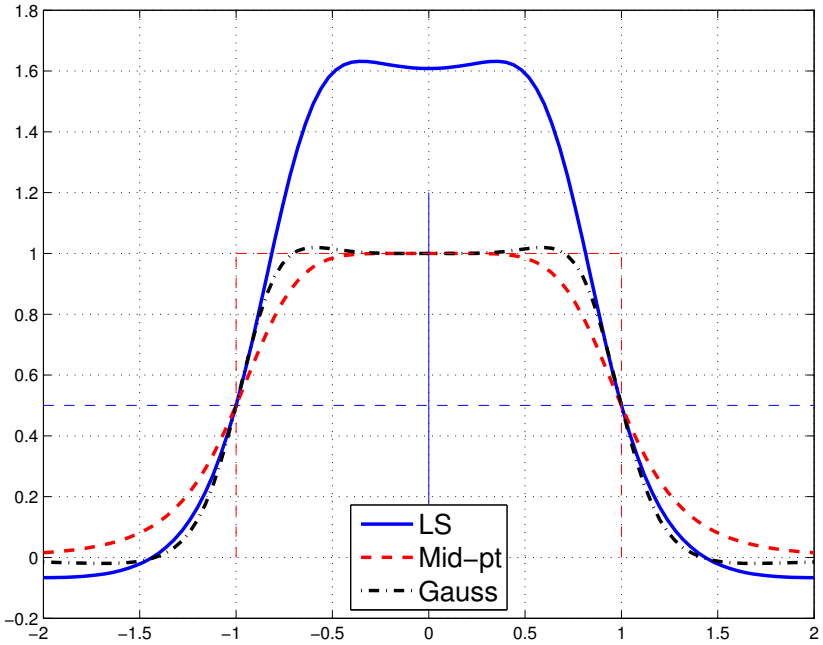
Spectral projector $\boldsymbol{P = \frac{-1}{2i\pi}\int_\Gamma (A - sI)^{-1} ds}$ + *Numer. Integ.*

➤ Approximation theory viewpoint:
Find rational function that approximates step function in $[-1, 1]$ *[after shifting + scaling]*

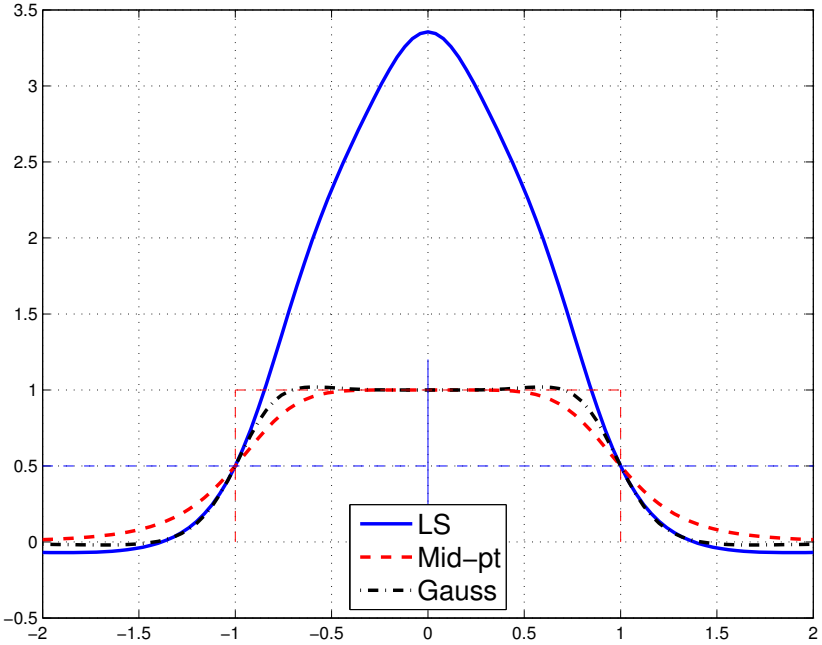
➤ e.g., work by S. Guettel et al. 2014. Uniform approx.



Different Look at the 3 curves

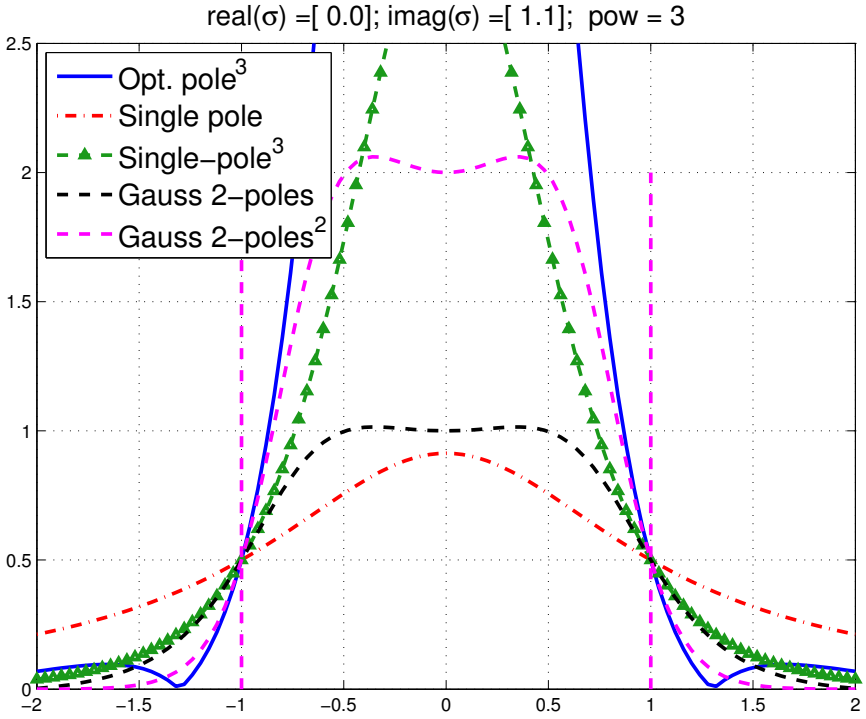


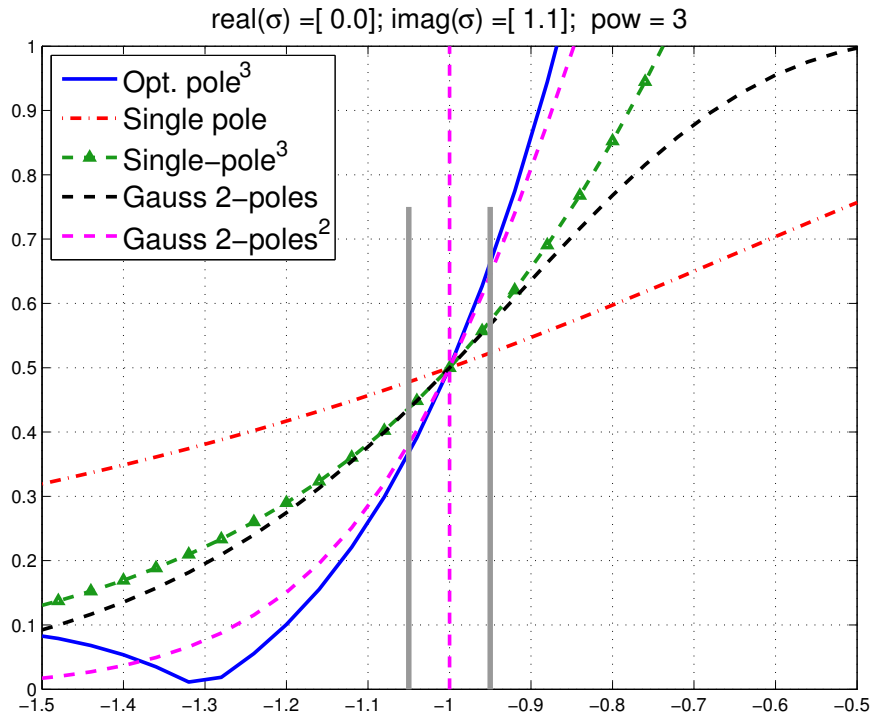
What about this case?



- The blue curve on the right is likely to give better results than the other 2 when subspace iteration is used.
- Want large values inside $[-1, 1]$ and small ones outside, but do not care much how well step function is approximated

What makes a good filter





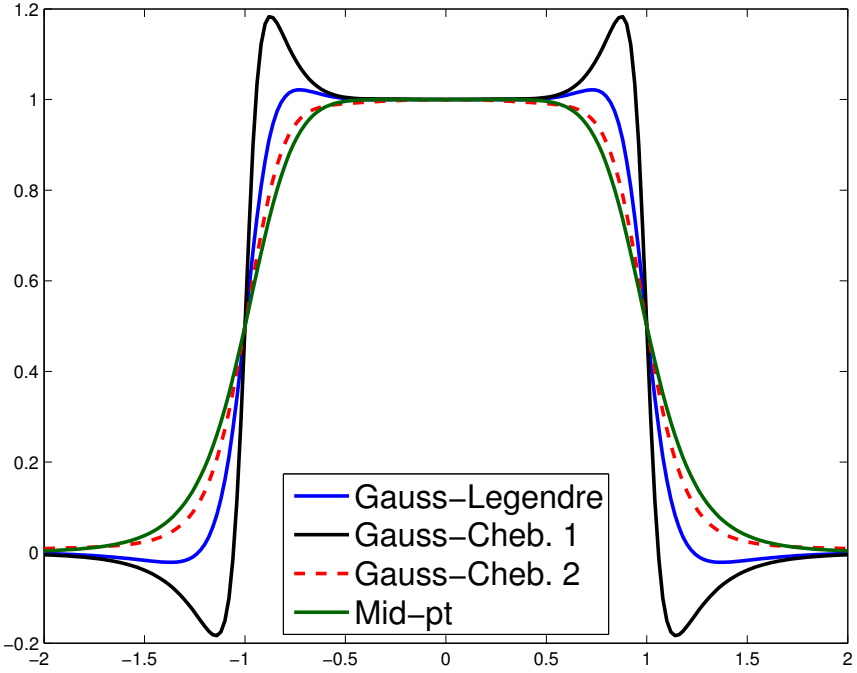
➤ Assume subspace iteration is used with above filters. Which filter will give better convergence?

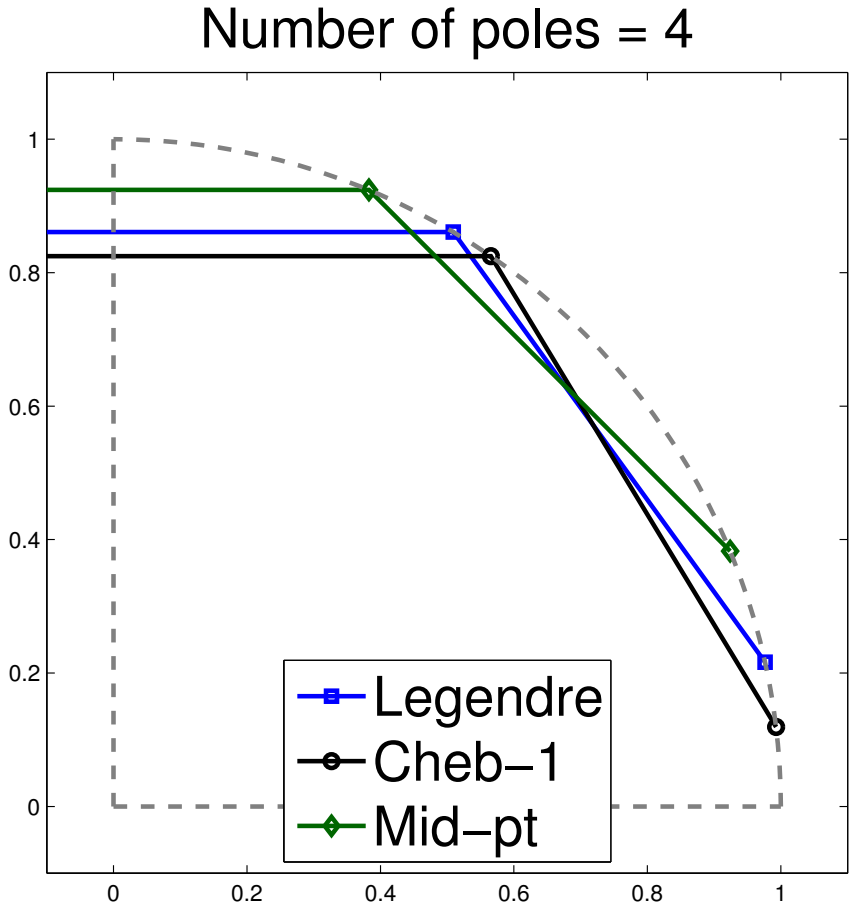
➤ Simplest and best indicator of performance of a filter is the magnitude of its derivative at -1 (or 1)

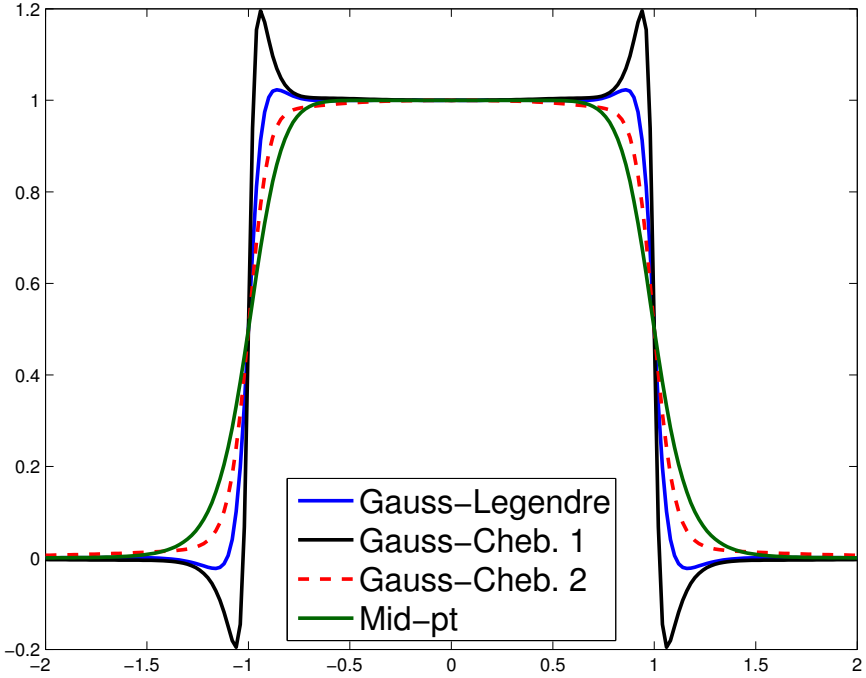
The Cauchy integral viewpoint

➤ Standard Mid-point, Gauss-Chebyshev (1st, 2nd) and Gauss-Legendre quadratures. Left: filters, right: poles

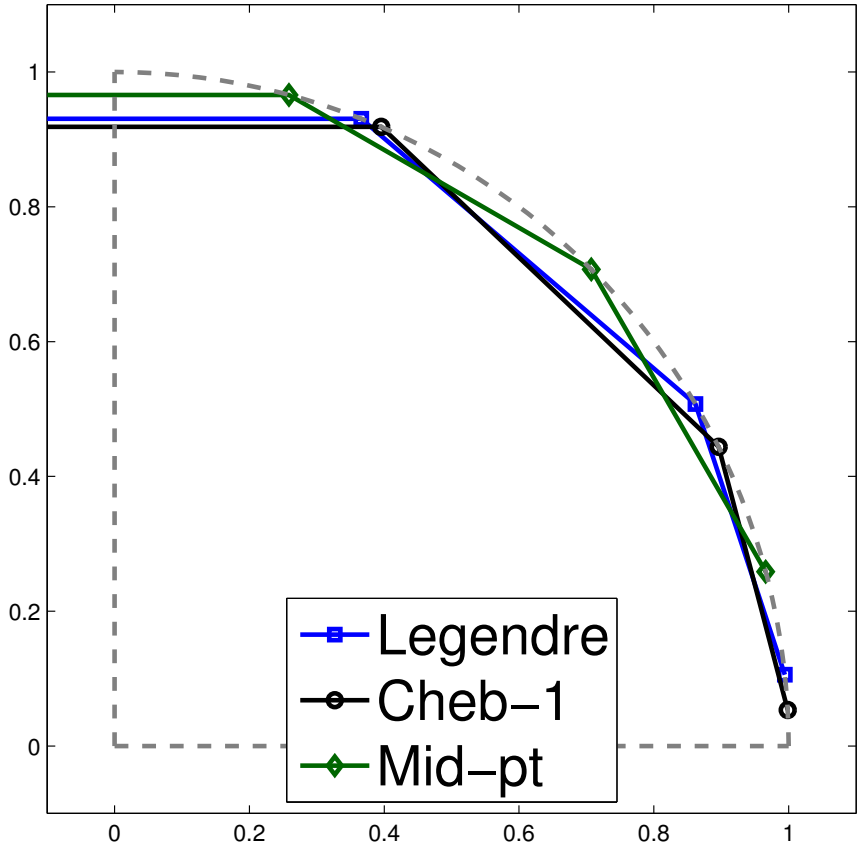
Scaling used so $\phi(-1) = \frac{1}{2}$







Number of poles = 6



- Notice how the sharper curves have poles close to real axis

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

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

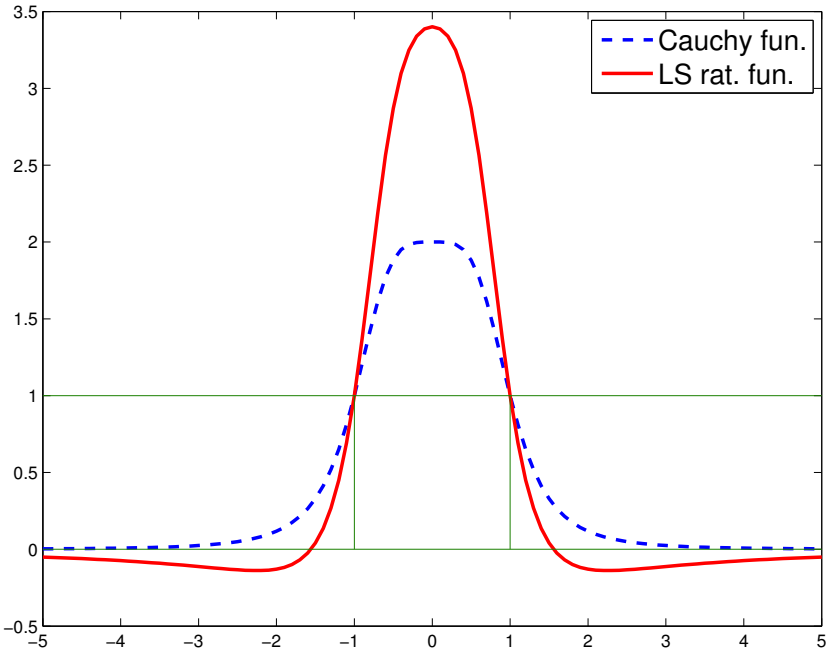
- $h(t)$ = step function $\chi_{[-1,1]}$.
- $w(t)$= weight function.

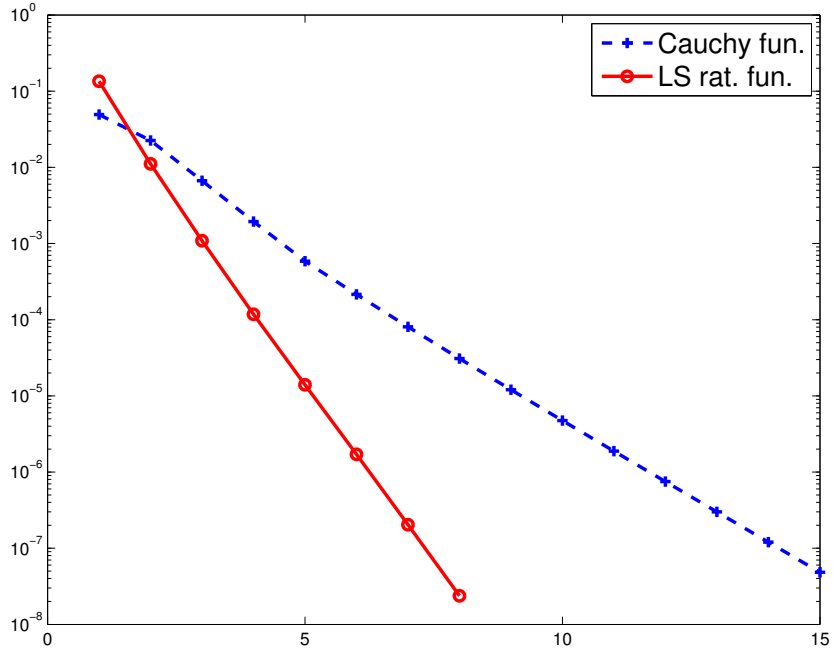
For example $a = 10$, $\beta = 0.1$

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

How does this work?

- A small example : Laplacean on a 43×53 grid. ($n = 2279$)
- Take 4 poles obtained from mid-point rule($N_c = 2$ on each 1/2 plane)
- Want: eigenvalues inside $[0,\ 0.2]$. There are $nev = 31$ of them.
- Use 1) standard subspace iteration + Cauchy (FEAST) then 2) subspace iteration + LS Rat. Appox.
- Use subspace of dim $nev + 6$
- $\beta = 0.2$



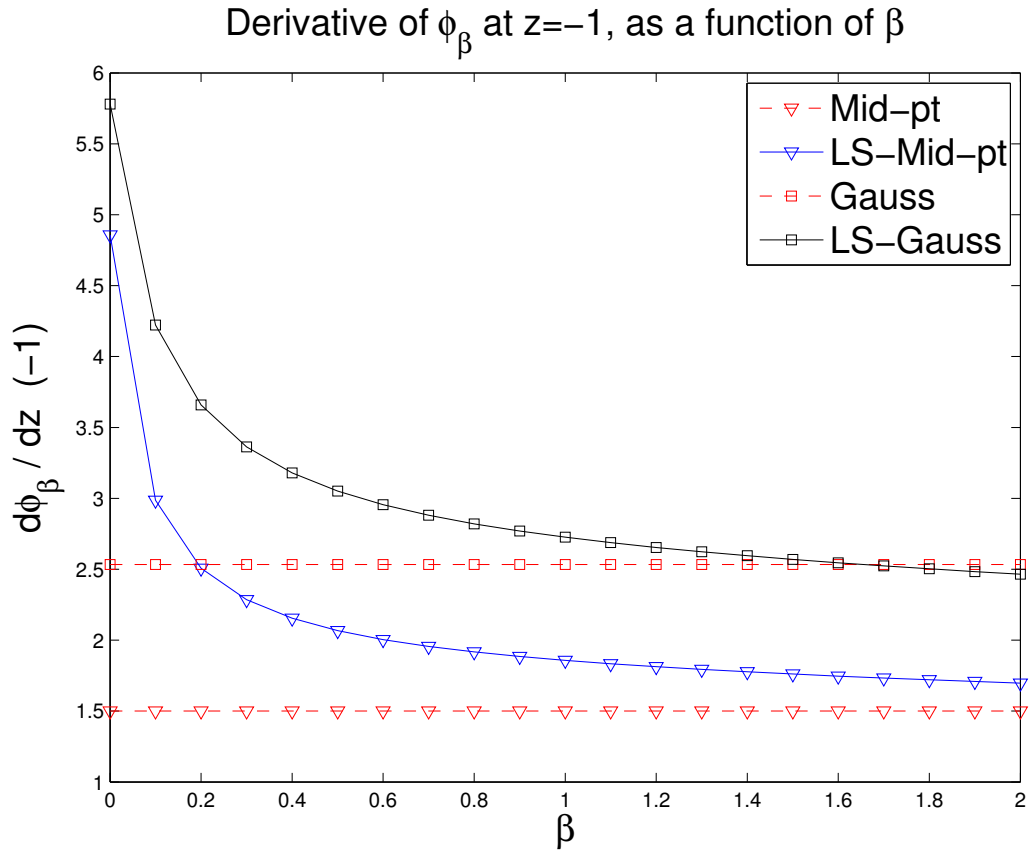


- LS Uses the same poles + same factorizations as Cauchy but
- ... much faster as expected from a look at the curves of the functions

- Other advantages:
  - Can select poles far away from real axis $\longrightarrow$ faster iterative solvers [E. Di Napoli, YS, et al-, work in progress]
  - Can use multiple poles (!)
  - Very flexible – can be adapted to many situations

Influence of parameter β

- Recall: large $\beta \rightarrow$ better approximation to h – not needed
- Smaller values are better



Better rational filters: Multiple poles

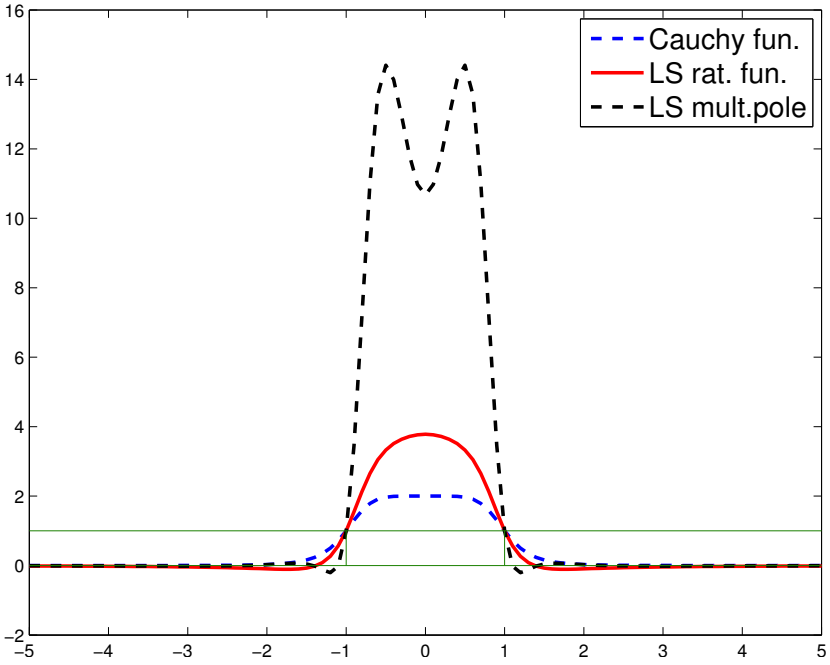
- Current choices: Mid-point/Trapezoidal rule, Gauss, Zolotarev,...
- None of these allows for repeated ('multiple') poles e.g., with one pole of multuplicity k:

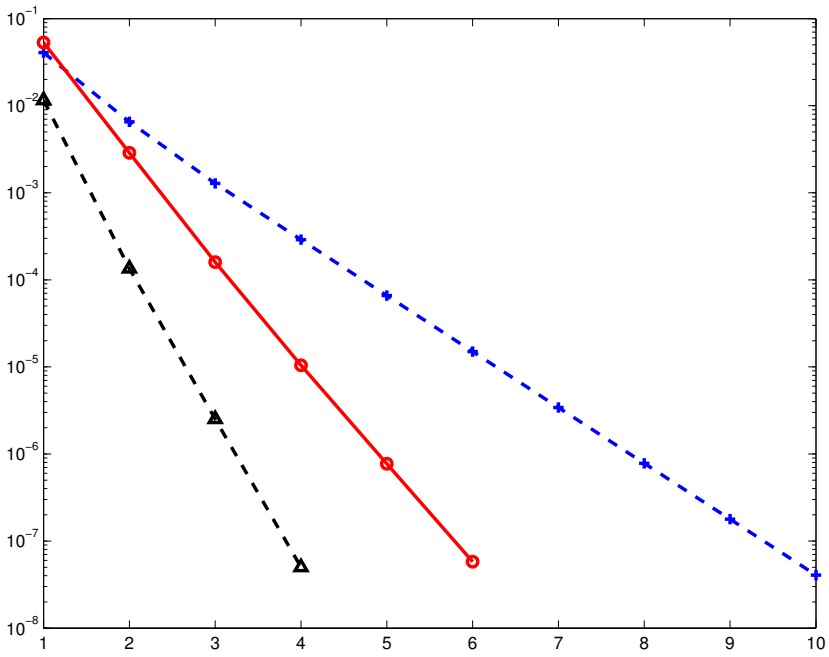
$$\phi(z) = \frac{\alpha_1}{z-\sigma} + \frac{\alpha_2}{(z-\sigma)^2} + \cdots + \frac{\alpha_k}{(z-\sigma)^k}$$

- Advantage: Fewer exact or incomplete factorizations needed for solves
- Next: Illustration with same example as before

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.]





- General form

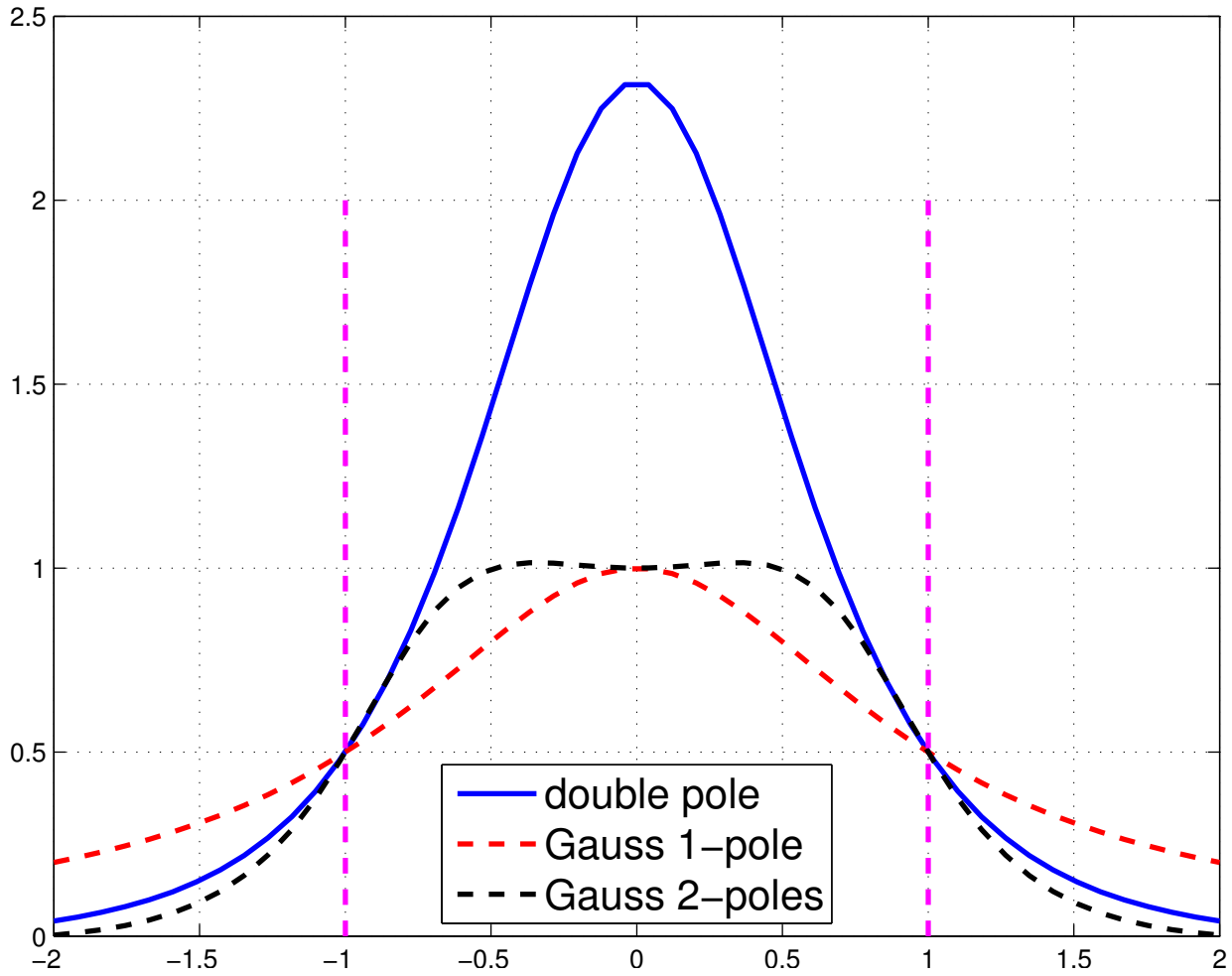
$$\phi(z) = \sum_{i=1}^{N_p} \sum_{j=1}^{k_i} \frac{\alpha_{ij}}{(z - z_i)^j}$$

- We then need to minimize (as before)

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

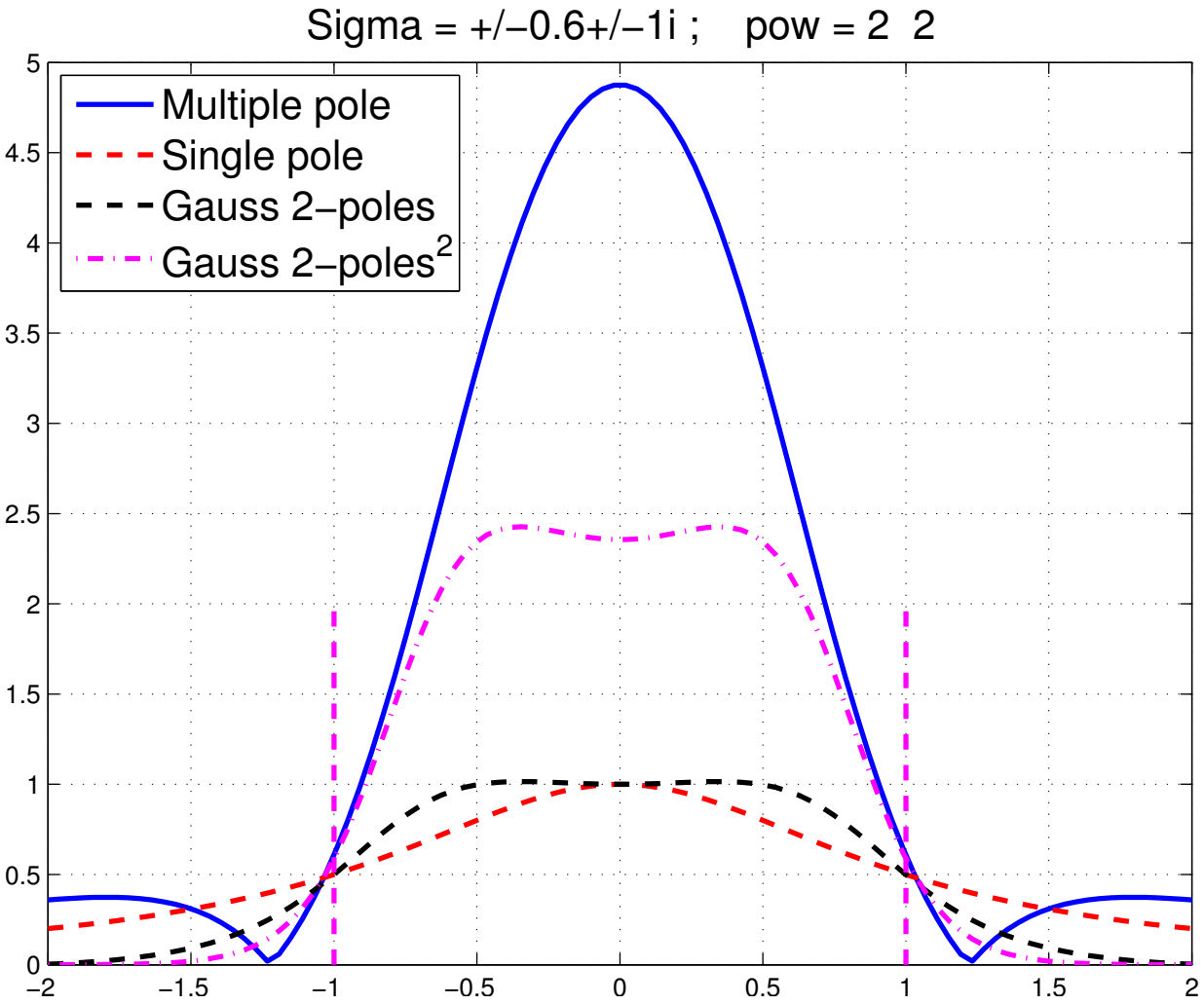
- Need to exploit partial fraction expansions
- Straightforward extension [but somewhat more cumbersome]

Next: See what we can do with *one* double pole



Who needs a circle?

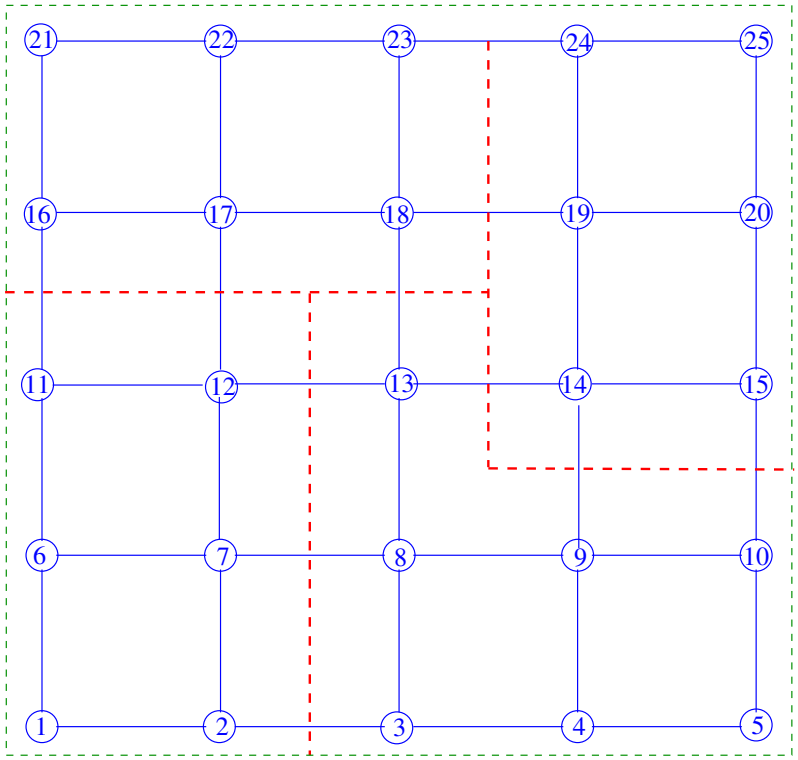
Two poles^2 + comparison with compounding



SPECTRAL SCHUR COMPLEMENT TECHNIQUES

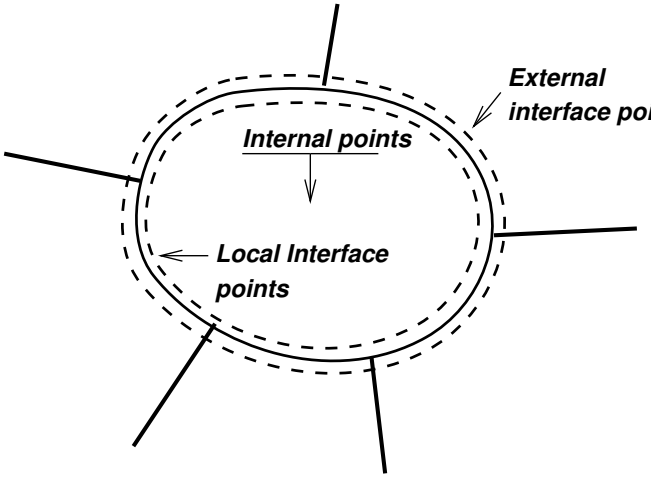
Introduction: Domain-Decomposition

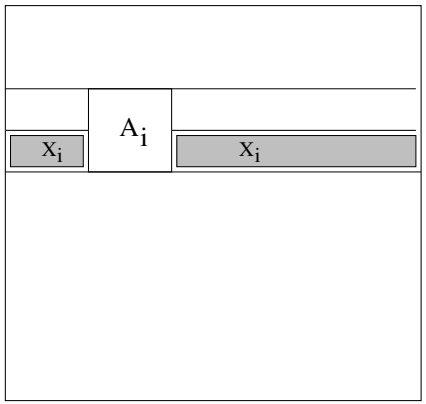
* Joint work with Vasilis Kalantzis and Ruipeng Li



➤ Partition graph using edge-separators ('vertex-based partitioning')

Distributed graph and its matrix representation





- Stack all interior variables $u_1, u_2, \cdots, u_p$ into a vector u, then interface variables y

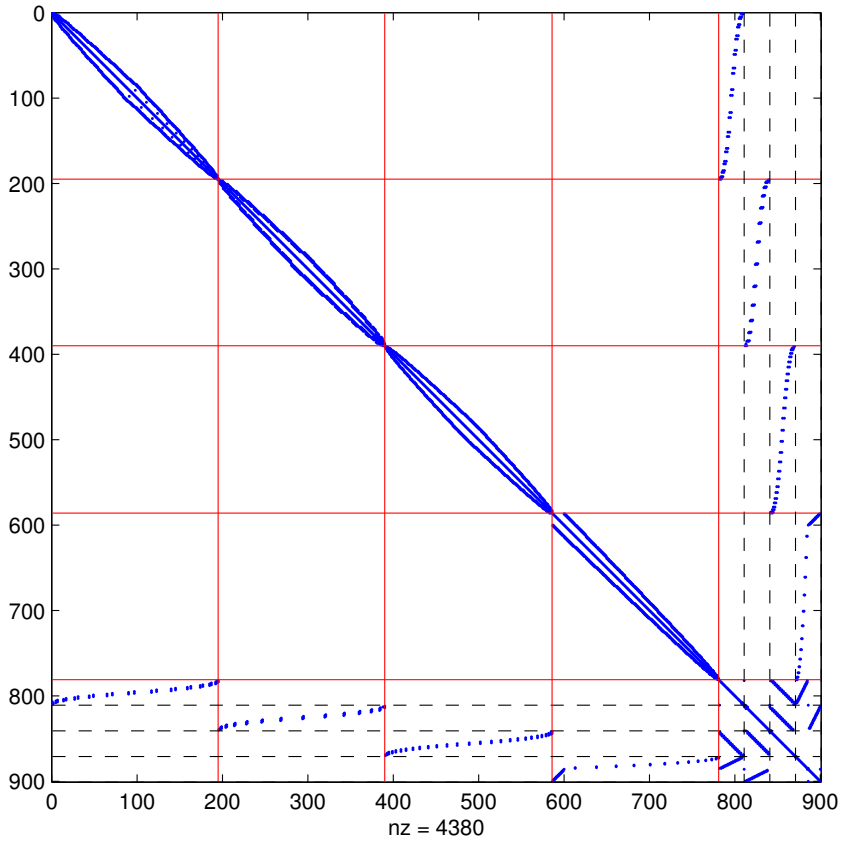
- Result:

$$\underbrace{\begin{pmatrix} B_1 & & & E_1 \\ & B_2 & & E_2 \\ \vdots & & \ddots & \vdots \\ & & B_p & E_p \\ E_1^T & E_2^T & \dots & E_p^T \quad C \end{pmatrix}}_{PAP^T} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_p \\ y \end{pmatrix} = \lambda \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_p \\ y \end{pmatrix}$$

Notation:

Write as:

$$A = \begin{pmatrix} B & E \\ E^T & C \end{pmatrix}$$



The spectral Schur complement

- Eliminating the u_i's we get

$$\begin{pmatrix} S_1(\lambda) & E_{12} & \cdots & E_{1p} \\ E_{21} & S_2(\lambda) & \cdots & E_{2p} \\ \vdots & & \ddots & \vdots \\ E_{p1}^\top & E_{p2}^\top & \cdots & S_p(\lambda) \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \end{pmatrix} = 0$$

- $S_i(\lambda) = C_i - \lambda I - E_i^\top (B_i - \lambda I)^{-1} E_i$
- Interface problem (non-linear): $S(\lambda) y(\lambda) = 0.$
- Top part can be recovered as $u_i = -(B - \lambda I)^{-1} E_i y(\lambda)$.
- See also AMLS [Bennighof, Lehoucq, 2003]

Spectral Schur complement (cont.)

State problem as:

- Find $\sigma \in \mathbb{R}$ such that

One eigenvalue of $S(\sigma) \equiv 0$, or,

➤ $\mu(\sigma) = 0$ where $\mu(\sigma)$ == smallest ($|.|$) eig of $S(\sigma)$.

➤ Can treat $\mu(\sigma)$ as a function $\rightarrow$ root-finding problem.

➤ The function $\mu(\sigma)$ is analytic for any $\sigma \notin \Lambda(B)$ with

$$\frac{d\mu(\sigma)}{d\sigma} = -1 - \frac{\|(B - \sigma I)^{-1} E y(\sigma)\|_2^2}{\|y(\sigma)\|_2^2}.$$

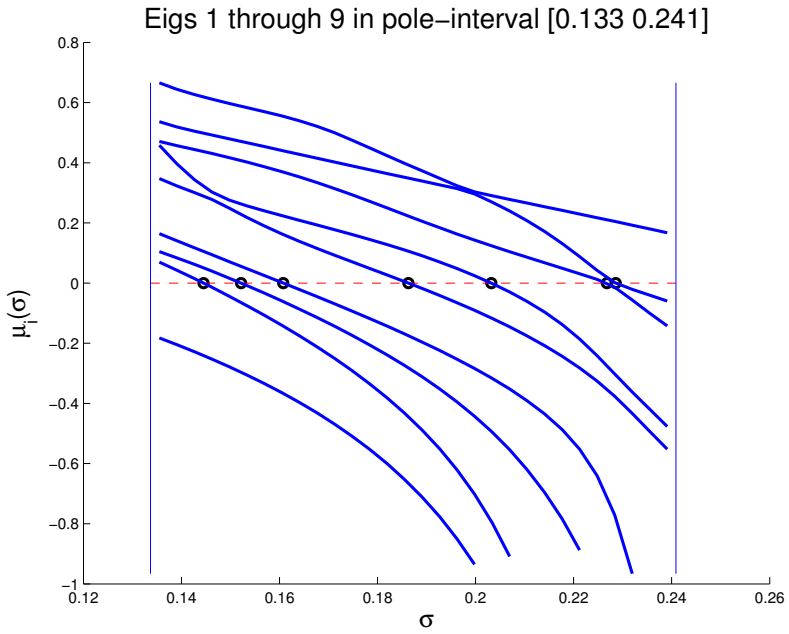
Basic algorithm - Newton's scheme

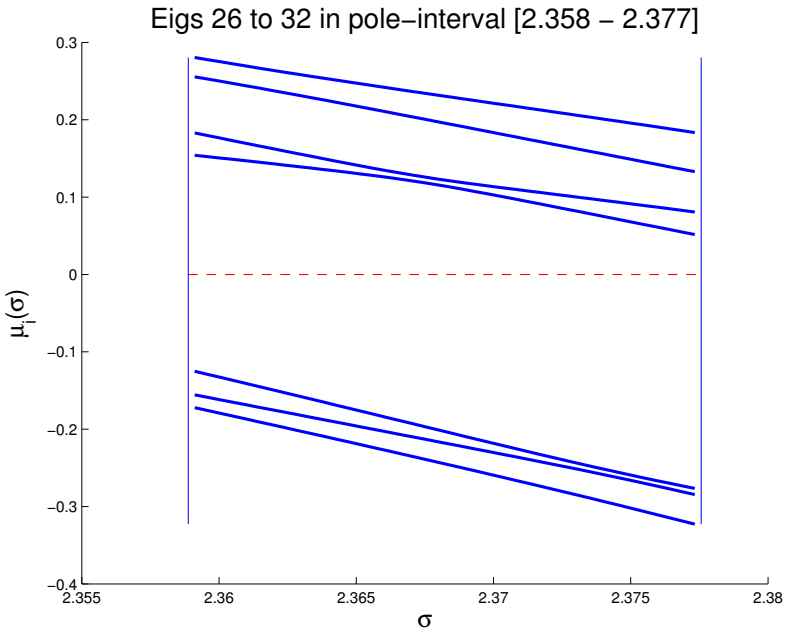
- We can formulate a Newton-based algorithm.

ALGORITHM : 1. *Newton Scheme*

1. *Select initial* σ
2. ***Repeat:***
3. *Compute* $\mu(\sigma) =$ *Smallest eigenvalue in modulus*
4. *of* $S(\sigma)$ *& associated eigenvector* $y(\sigma)$
5. *Set* $\eta := \|(B - \sigma I)^{-1} E y(\sigma)\|_2$
6. *Set* $\sigma := \sigma + \mu(\sigma)/(1 + \eta^2)$
7. ***Until:*** $|\mu(\sigma)| \leq \mathrm{tol}$

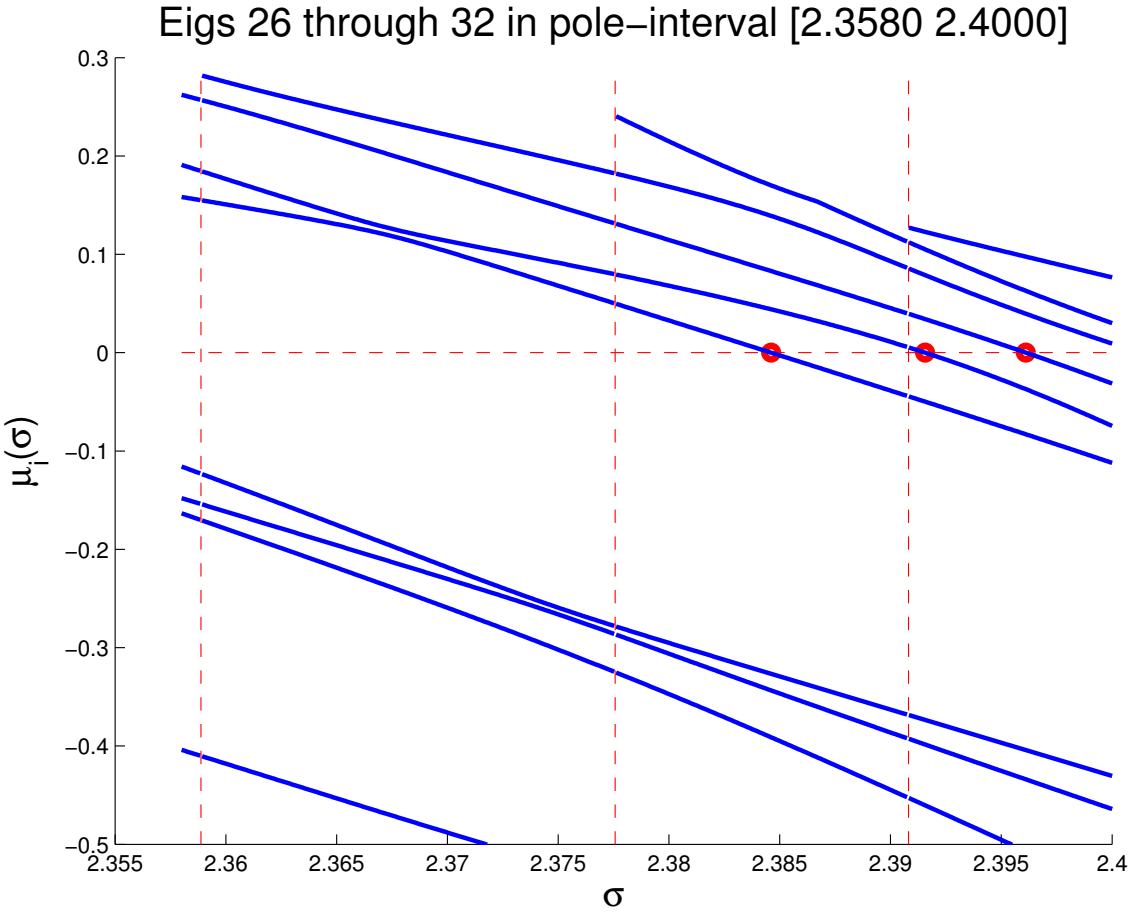
Short illustration - eigen-branches between two poles





➤ There may be a few, or no, eigenvalues λ between two poles.

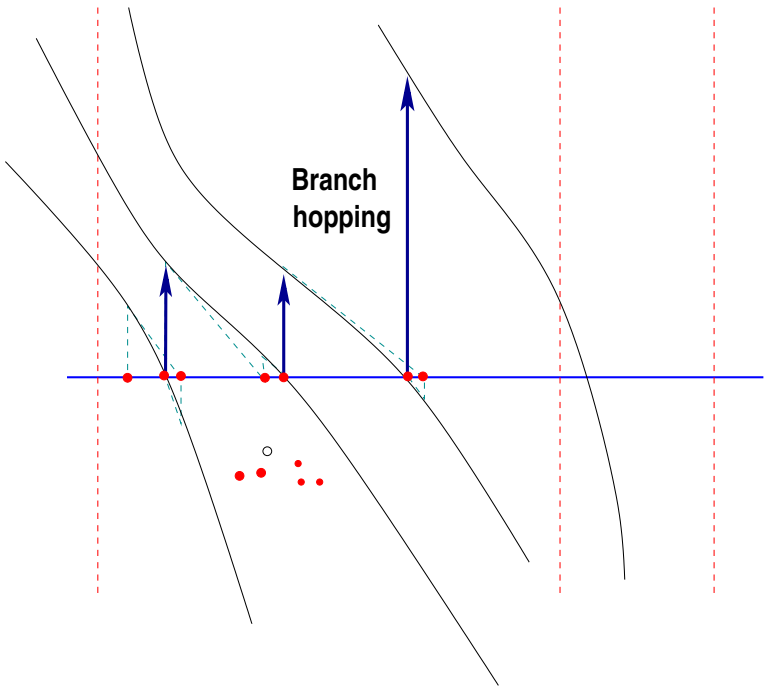
Eigen-branches across the poles

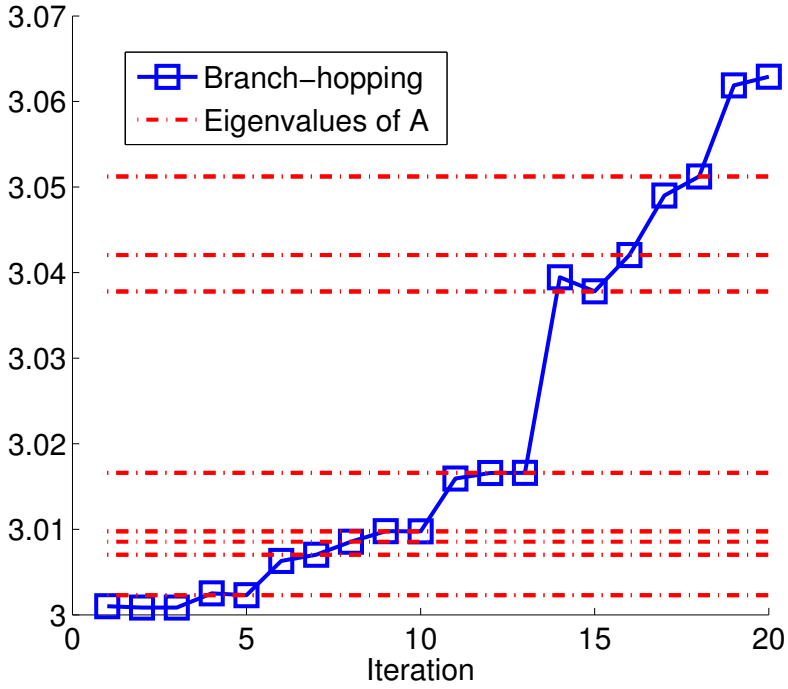


$$S(\sigma) = C - \sigma I - E^T(B - \sigma I)^{-1}E \equiv C - \sigma I - \sum_{i=1}^{m} \frac{w_i w_i^T}{\theta_i - \sigma}$$

Branch-hopping

- Once we converge ...
- ... start Newton from point on branch immediatly above current root.





Evaluating $\mu(\sigma)$

Inverse Iteration

- For any σ we just need one (two) eigenvalues of $S(\sigma)$.
- Good setting for "Inverse-Iteration" type approaches.

Lanczos algorithm

- Tends to be expensive for large p and eigenvalues deep inside spectrum

Numerical experiments

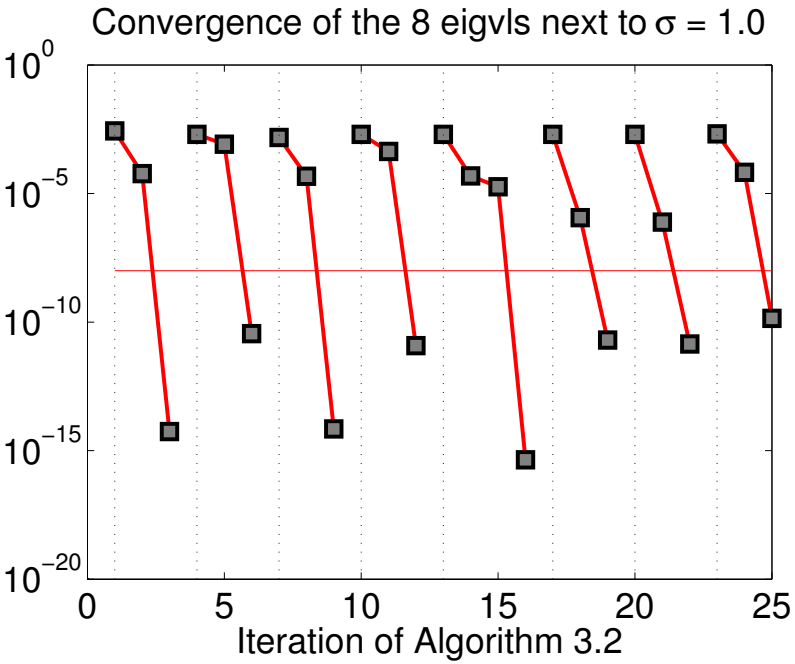
Some details

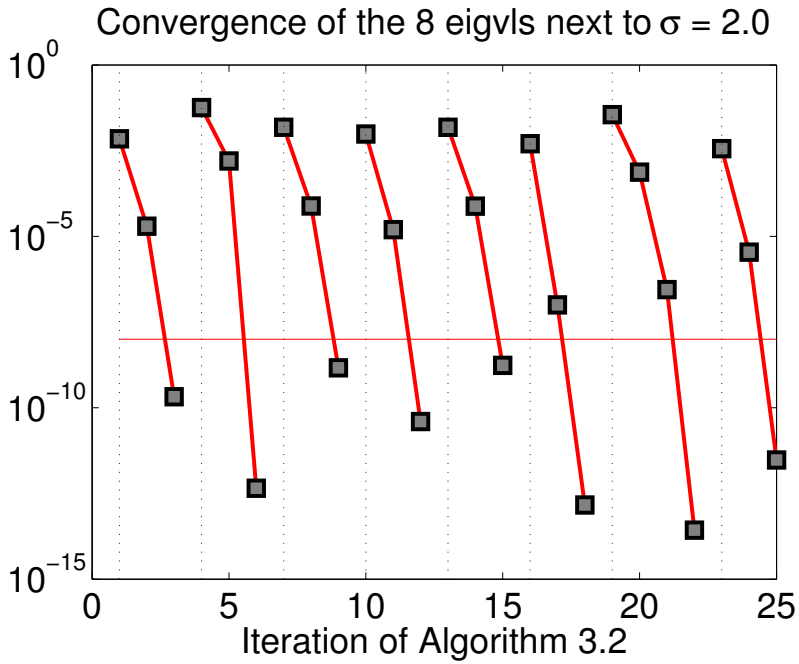
- Implementation in MPI-C++ using PETSc
- Tests performed on Itasca Linux cluster @ MSI.

The model problem

- Tests on 3-D dicretized Laplacians (7pt. st. – FD).
- We use n_x, n_y, n_z to denote the three dimensions.
- tol set to $1e-12$.
- Single-level partitioning – One node per sub-domain.

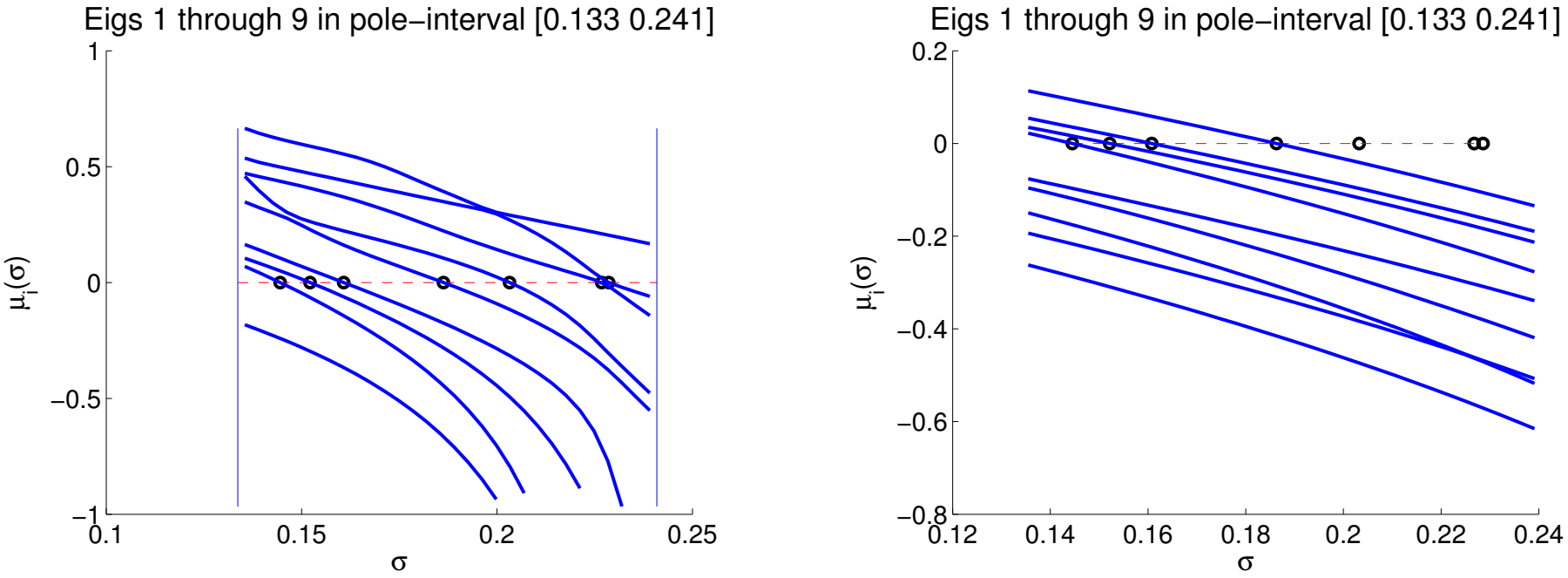
Example of convergence history





Rel. res. for a few consecutive eigenvalues.
Left $40 \times 40 \times 20$ grid. Right: $20 \times 20 \times 20$

Effect of number of domains p on convergence



Eigen-branches $\mu_1(\sigma), \ldots, \mu_9(\sigma)$ in $[0.133, 0.241]$ for a $33 \times 23 \times 1$ grid. Left: $p = 4$, right: $p = 16$.

Numerical experiments: Parallel tests

- Tests performed on Itasca Linux cluster @ MSI.
- Each node is a two-socket, quad-core 2.8 GHz Intel Xeon X5560 “Nehalem EP" with 24 GB of system memory.
- Interconnection : 40-gigabit QDR InfiniBand (IB).

The model problem

- Tests on 3-D dicretized Laplacians (7pt. st. – FD).
- We use n_x, n_y, n_z to denote the three dimensions.
- tol set to $1e-12$.

Wall-clock timings to compute the first $k = 1$ and $k = 5$ eigenpairs to the right of σ.

			$\sigma = 0.0$		$\sigma = 0.5$	
	(p,k)	s	Sec	It	Sec	It
$71 \times 70 \times 69$	(32,1)	65647	1.66	3	11.7	4
	(32,5)	— —	10.1	15	68.2	19
	(64,1)	83358	0.47	3	5.20	4
	(64,5)	— —	3.20	14	31.2	19
	(128,1)	108508	**0.16**	**3**	**1.90**	**4**
	(128,5)	— —	**1.10**	**14**	**11.7**	**18**
$101 \times 100 \times 99$	(64,1)	181901	5.90	3	41.1	3
	(64,5)	— —	28.3	15	221.5	15
	(128,1)	230849	1.18	3	7.80	3
	(128,5)	— —	6.24	15	41.5	14
	(256,1)	293626	**0.65**	**3**	**2.90**	**3**
	(256,5)	— —	**3.80**	**15**	**14.8**	**14**

	(p,k)	s	$\sigma = 0.0$ Sec	$\sigma = 0.0$ It	$\sigma = 0.5$ Sec	$\sigma = 0.5$ It
601×600	(16,1)	7951	0.65	3	3.10	3
	(16,5)	— —	4.42	15	15.9	15
	(32,1)	12377	0.26	3	2.90	3
	(32,5)	— —	1.60	14	16.4	15
	(64,1)	18495	**0.20**	**3**	**1.10**	**3**
	(64,5)	— —	**1.10**	**14**	**5.40**	**14**
801×800	(32,1)	16673	1.83	3	11.1	3
	(32,5)	— —	10.2	15	65.1	15
	(64,1)	24945	0.73	3	5.00	3
	(64,5)	— —	4.10	15	28.20	14
	(128,1)	36611	**0.31**	**3**	**2.10**	**3**
	(128,5)	— —	**1.80**	**15**	**9.70**	**14**

Conclusion

Part I: Polynomial filtering

- Polnym. Filter. appealing when # of eigenvectors to be computed is large and when Matvecs are inexpensive
- Will not work too well for generalized eigenvalue problem
- Will not work well for spectra with very large outliers.

Part II: Rational filtering

- We must rethink the way we view Rational filtering - away from Cauchy and into approximation of functions. LS approach is flexible, easy to implement, easy to understand.

Part III: Domain Decomposition

- We *must* combine DD with any filtering technique [rational or polynomial]
- Many ideas still to explore in Domain Decomposition for interior eigenvalue problems

- `Filtlan` code available here:

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