UNIVERSITY OF MINNESOTA TWIN CITIES

Filtered thick restart Lanczos algorithm and the EVSL package

Yousef Saad

Department of Computer Science and Engineering

University of Minnesota

PMAA-2016, Bordeaux (France)

Jul. 8th, 2016

Collaborators:

Past work:
- Haw-ren Fang [former post-doc]
- Grady Schoefield and Jim Chelikowsky [UT Austin] - Windowing into PARSEC

New group effort:
- Ruipeng Li [now at LLNL]
- Eugene Vecharynski
- Chao Yang
- Yuanzhe Xi

➤ 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. e.g.,:

Time-Dependent Density Functional Theory (TDDFT). The so-called Cassida approach computes eigenvalues of a matrix $\boldsymbol{K}$ built using both occupied and unocuppied states

- Similar types of calculations in the GW approach [see, e.g., BerkeleyGW] – But more complex
- Challenge: 'Hamiltonian of size $n \sim 10^6$ get 10% of bands'

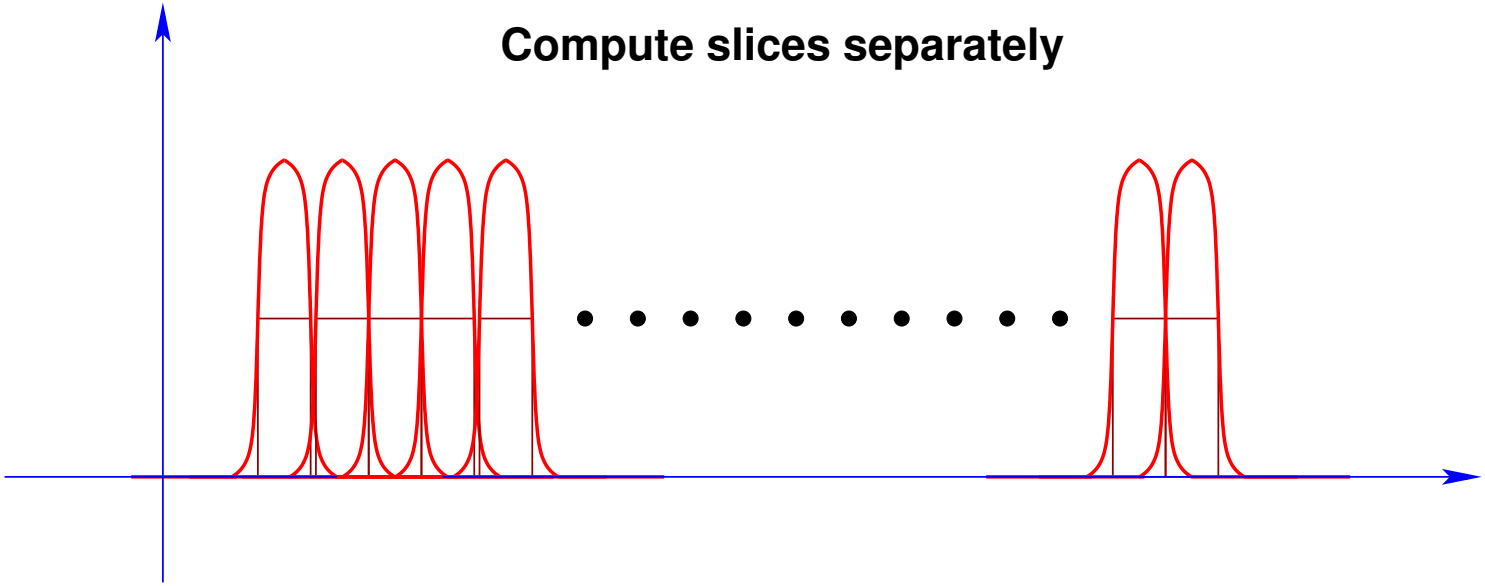
Solution: Spectrum Slicing

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

➤ In an approach of this type the filter is the key ingredient.

Goal: Compute each slice independently from the others.



- For each slice Do:
  [get *all* eigenpairs in a slice]
  EndDo

- Only need a good estimate of window size

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..)

- Requires factorizations of $A - \sigma I$ for a sequence of σ's
- Out of the question for realistic problems.
- First Alternative: Polynomial filtering

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

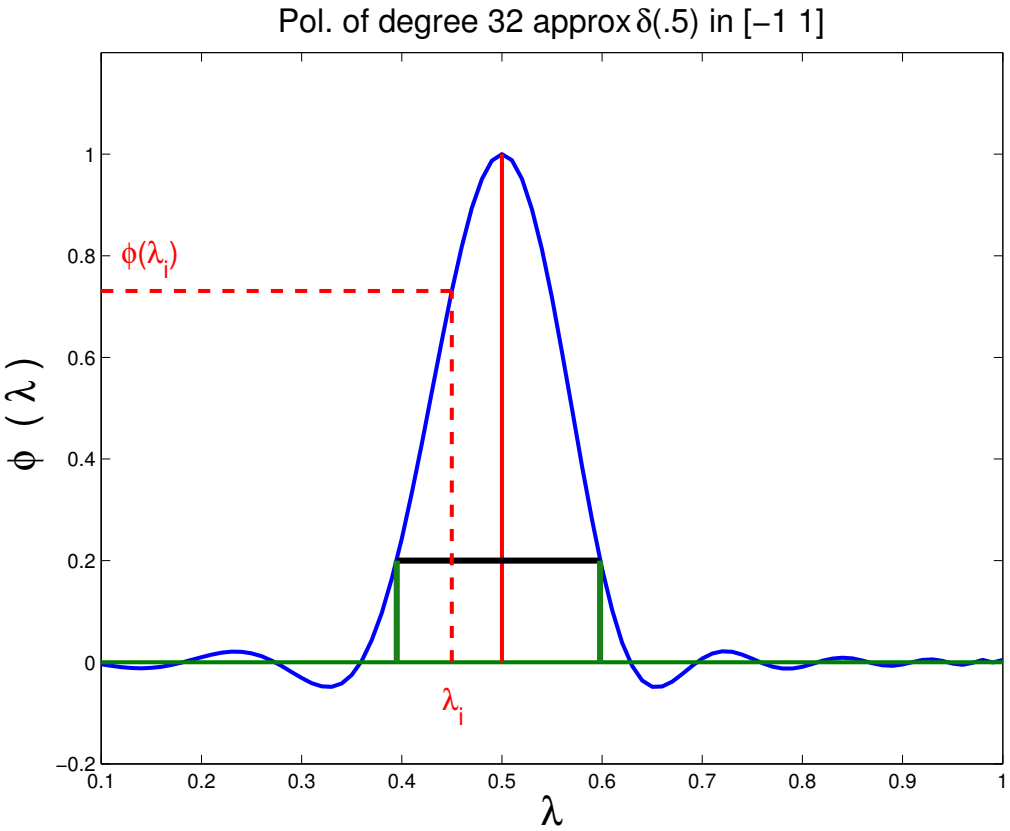
```
for iter=1:maxit
      Y = PolFilt(B, V, mu);
      [V, R] = qr(Y,0);
%— Rayleigh-Ritz with B
      C = V'*(B*V);
      [X, D] = eig(C);
      d = diag(D);
%— get e-values closest to center of interval
      [~, idx] = sort(abs(d-gam),'ascend');
%— get their residuals
      ...
%— if enough pairs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

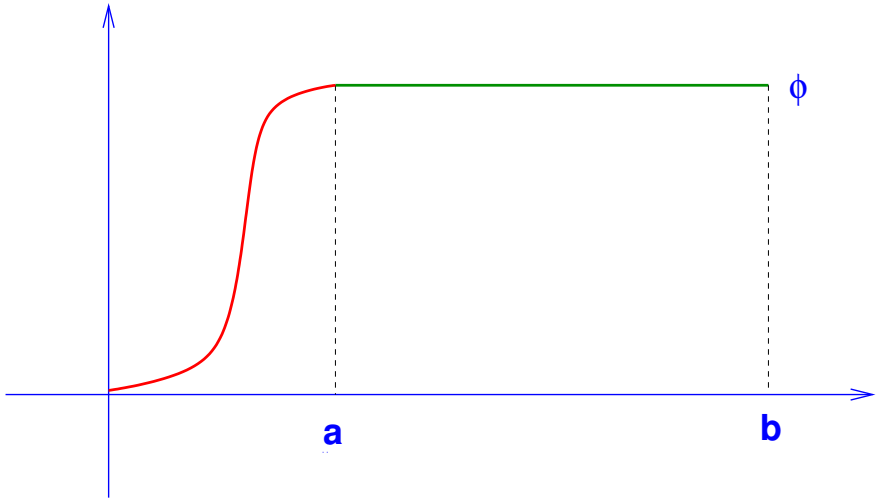


Past work: Two-stage approach

➤ Two stage approach used in filtlan [H-r Fang, YS 2011] -

➤ First select a "base filter"

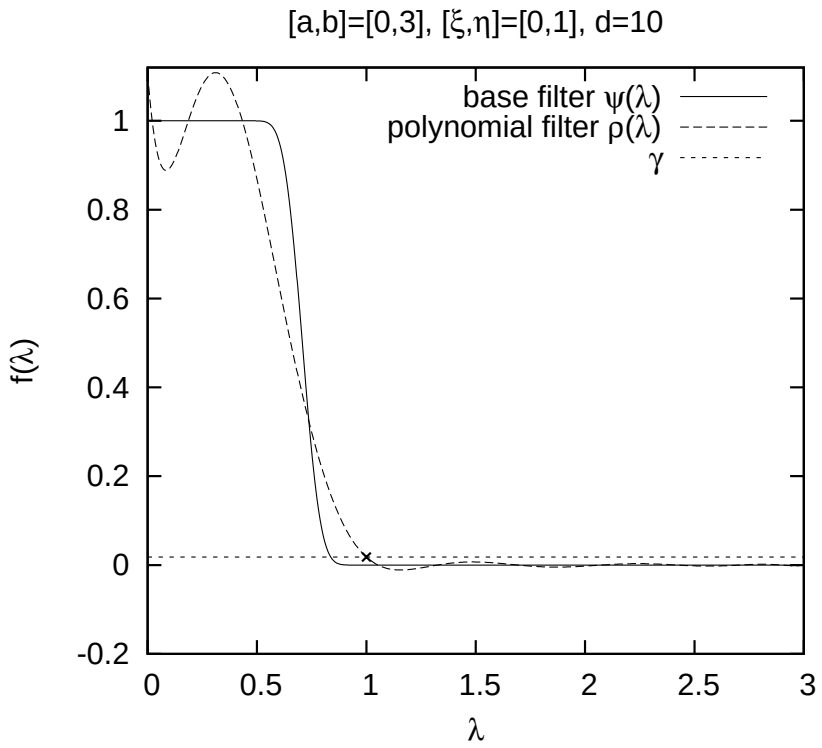
➤ e.g., a piecewise polynomial function [a spline]

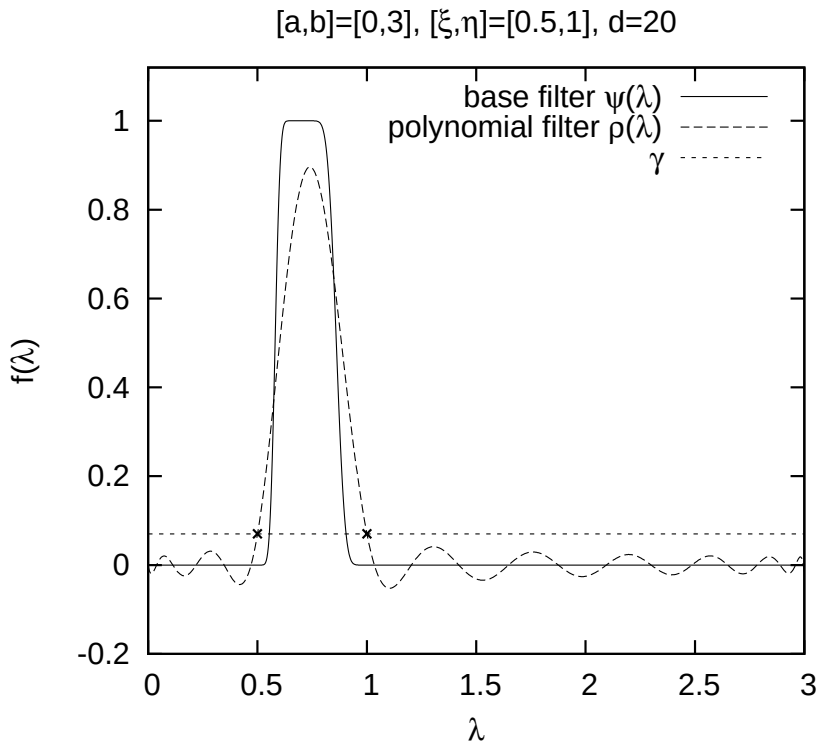


- Then approximate base filter by degree k polynomial in a least-squares sense.
- No numerical integration needed

Main advantage: Extremely flexible.

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



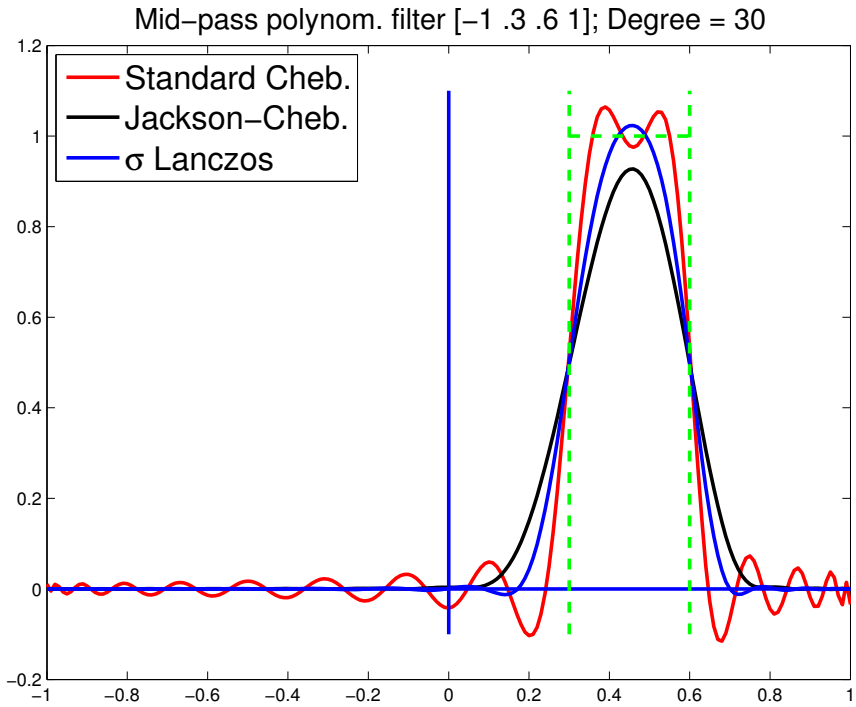


- See Reference on Lanczos + pol. filtering: Bekas, Kokiopoulou, YS (2008) for motivation, etc.
- H.-r Fang and YS "Filtlan" paper [SISC,2012] and code

Simpler: Step-function Chebyshev + Jackson damping

➤ Seek the best LS approximation to step function.

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



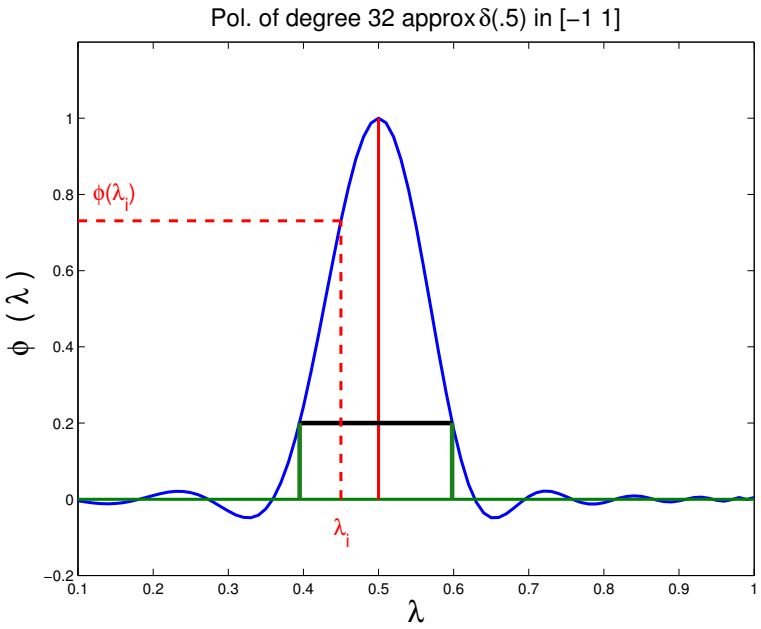
➤ Add 'Damping coefficients' to reduce/eliminate Gibbs oscillations

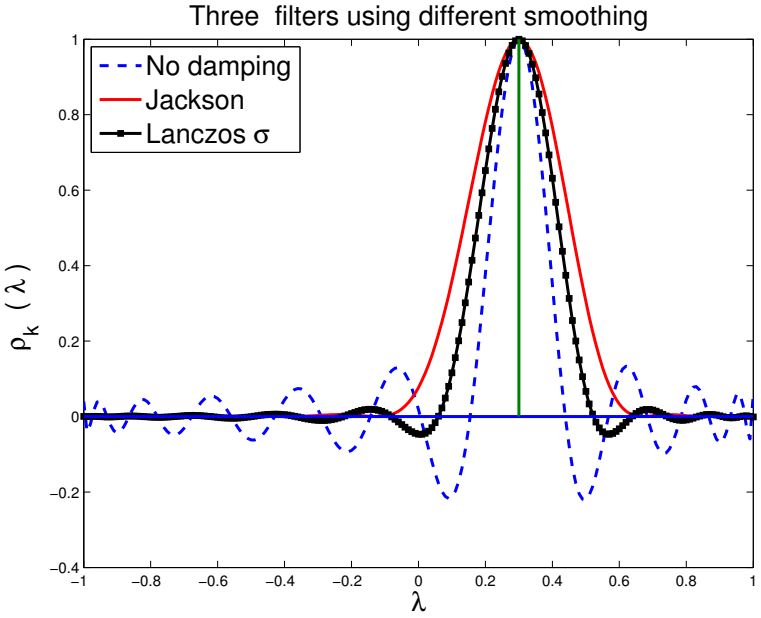
➤ G. Schofield, J. R. Chelikowsky and YS, CPC, 183, ('11)

Question: Why approximate the 'step function'?

Even Simpler: δ-Dirac function

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





⟵ Can use same damping: Jackson, Lanczos σ damping, or none.

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 $\leq k$, such that $r(\gamma) = 1$, where $\|.\|_w$ represents the Chebyshev L^2-norm. Then $\hat{\rho}_k(t) = \rho_k(t)/\rho_k(\gamma)$.

Theorem Assuming $k \geq 1$, the following equalities hold:

$$\int_{-1}^{1} \frac{[\hat{\rho}_k(s)]^2}{\sqrt{1-s^2}} ds = \frac{1}{\sum_{j=0}^{k} [\hat{T}_j(\gamma)]^2}$$

$$= \frac{2\pi}{(2k+1)} \times \frac{1}{1 + \frac{\sin(2k+1)\theta_\gamma}{(2k+1)\sin\theta_\gamma}},$$

where $\theta_\gamma = \cos^{-1}\gamma$.

'The soul of a new filter' – A few details

$$p_m(t) = \sum_{j=0}^{m} \gamma_j^{(m)} \mu_j T_j(t)$$

$$\mu_k = \begin{cases} 1/2 & \text{if } k == 0 \\ \cos(k \cos^{-1}(\gamma)) & \text{otherwise} \end{cases}$$

$$\gamma_j^{(m)} = \text{Damping coefficients.}$$

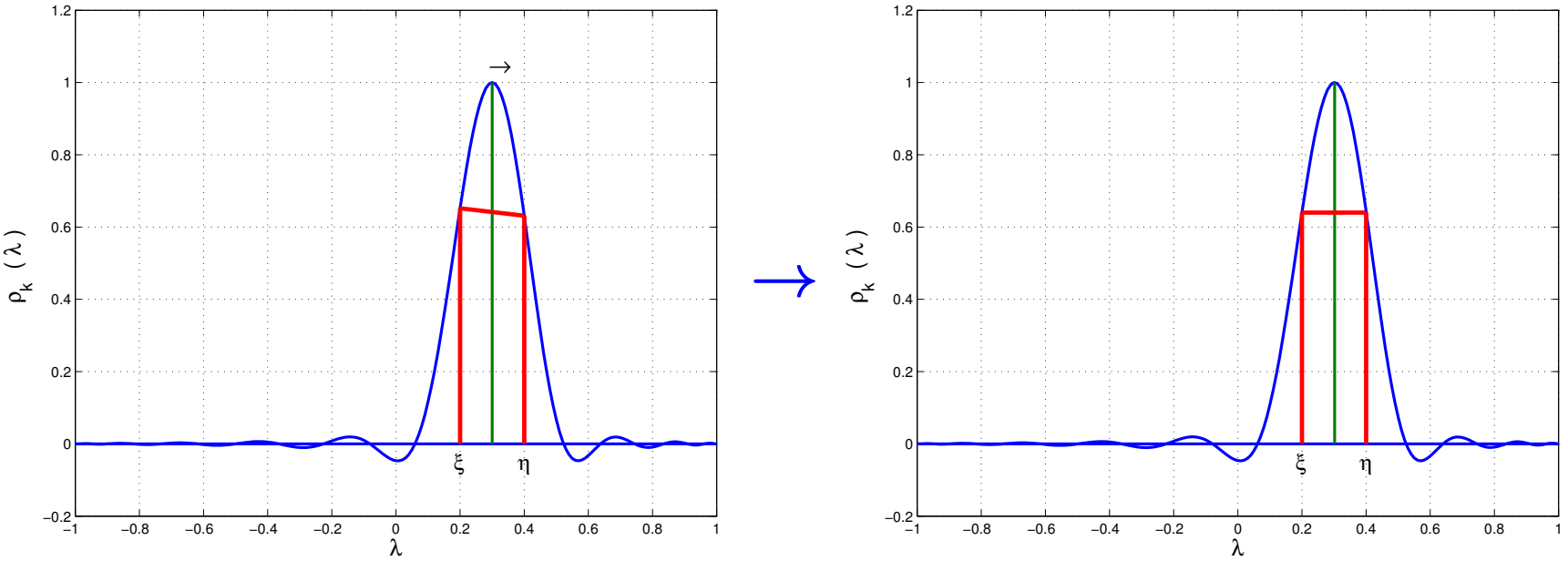
Problem:

Given interval $[\xi, \eta]$ find $p_m(t)$ of degree m, such that (1) convergence with e.g. subspace iteration or TR/IR-Lanczos is fast enough, (2) wanted eigenvalues are easy to identify and extract, and (3) unwanted ones are never an issue.

Issue # one: 'balance the filter'

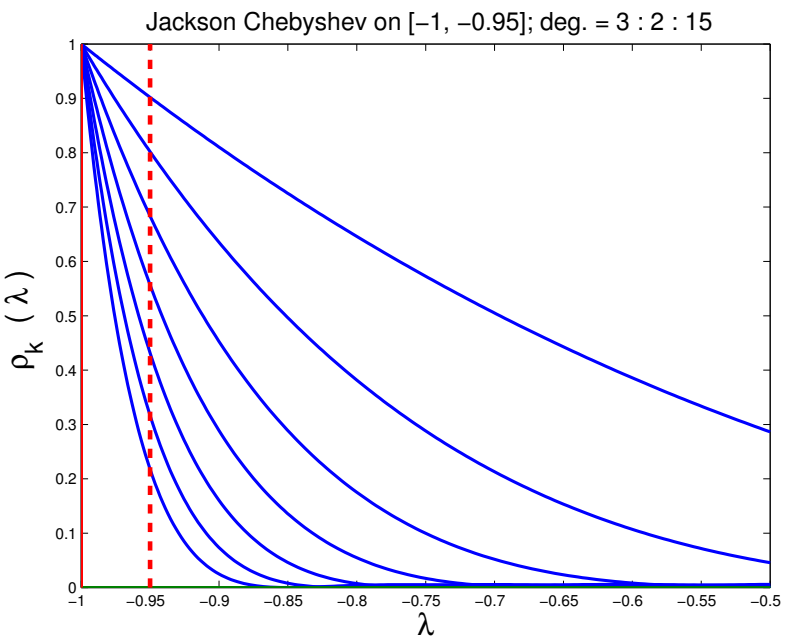
➤ To facilitate the selection of 'good' eigenvalues [Select λ's such that $\phi(\lambda) >$ bar] we need to ...

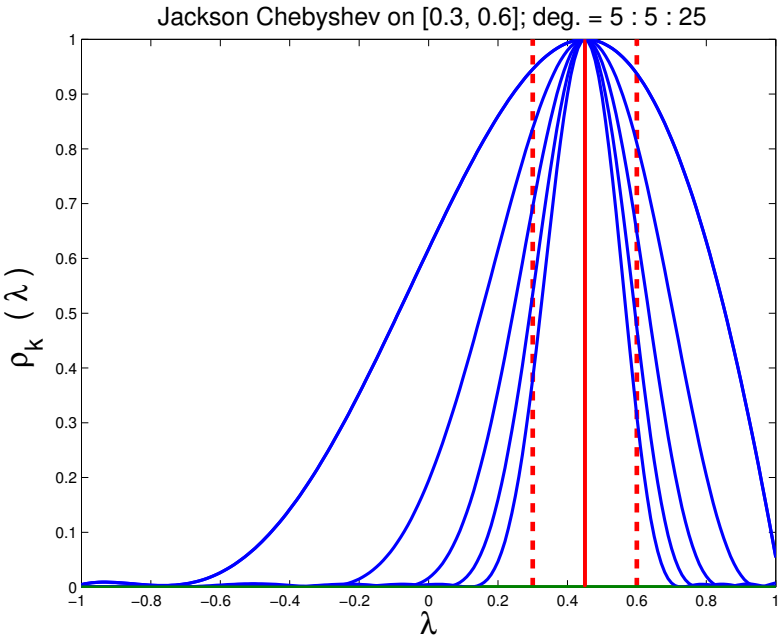
➤ ... find γ so that $\phi(\xi) == \phi(\eta)$



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

Issue # two: Determine degree (automatically)



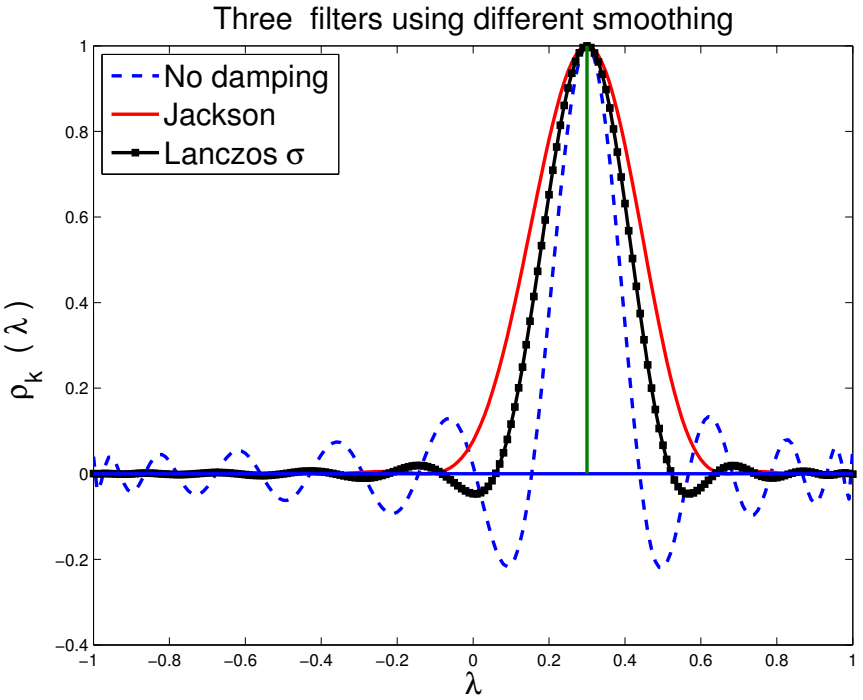


- 1) Start low (e.g. 2); 2) Increase degree until value (s) at the boundary (ies) become small enough –
- Eventually w'll use criterion based on derivatives at ξ & η

Issue # Three : Gibbs oscillations

- Discontinuous 'function' approximated → Gibbs oscillations

- Three options:
  - No damping
  - Jackson damping
  - Lanczos σ damping



- Good compromise: Lanczos σ damping

Extraction: Lanczos vs. Subspace iteration

➤ Subspace iteration is quite appealing in a electronic structure calculations – Can re-use previous subspace.

➤ Current emphasis: Lanczos with Thick-Restarting [TR Lanczos, Stathopoulos et al '98, Wu & Simon'00]

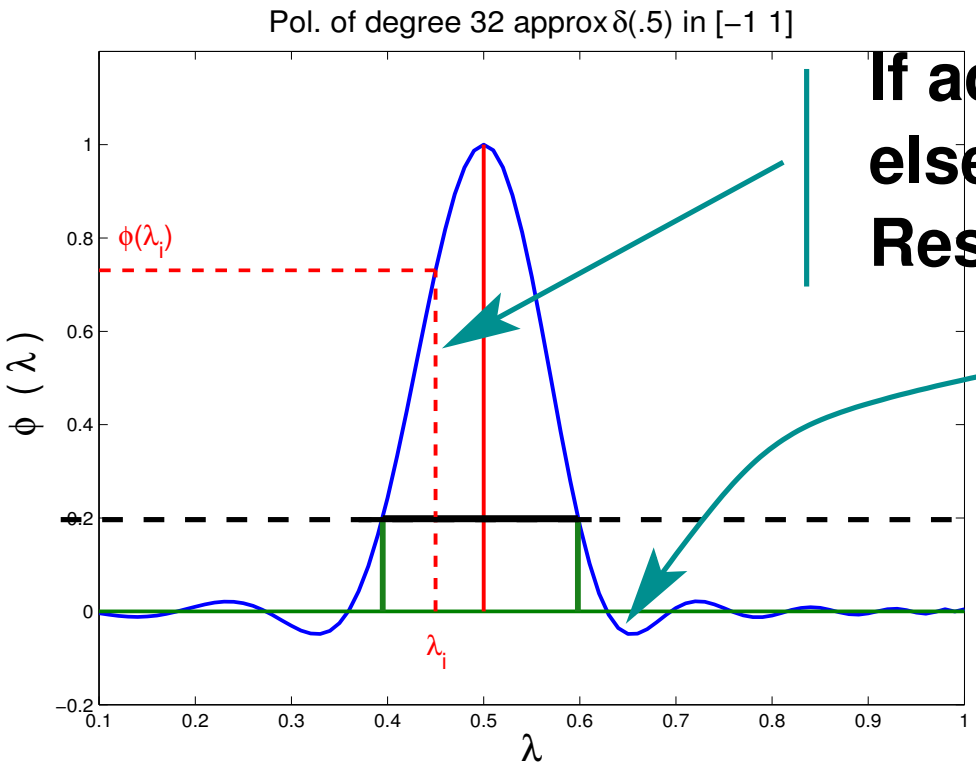
➤ EVSL will provide these two + other options.

➤ Crucial tool in TR Lanczos: deflation ('Locking')

Main idea: *Keep extracting eigenvalues in interval* $[\xi, \eta]$ *until none are left [remember: deflation]*

➤ If filter is good: Can catch all eigenvalues in interval thanks to deflation + Lanczos.

- PolFilt Thick-Restart Lanczos in a picture:



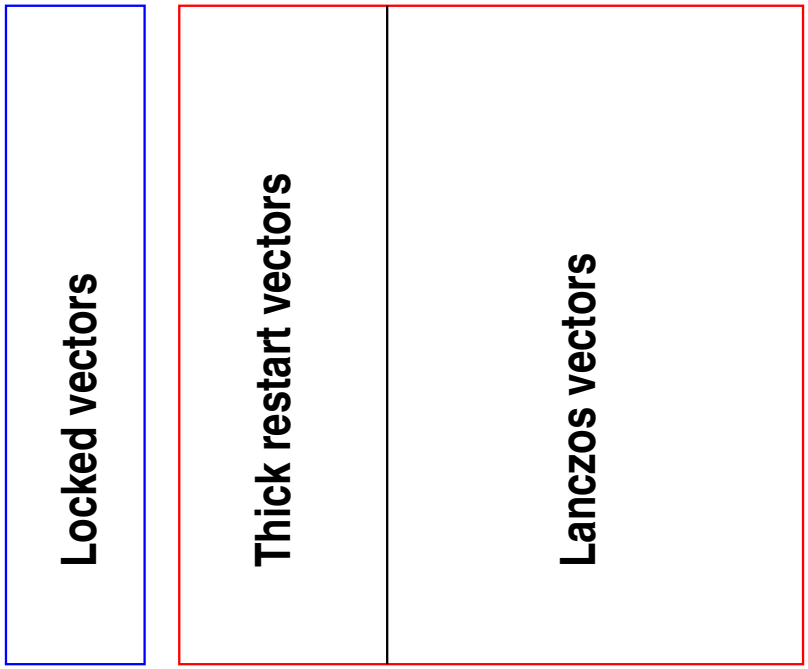
If accurate then lock else add to Thick Restart set.

Reject

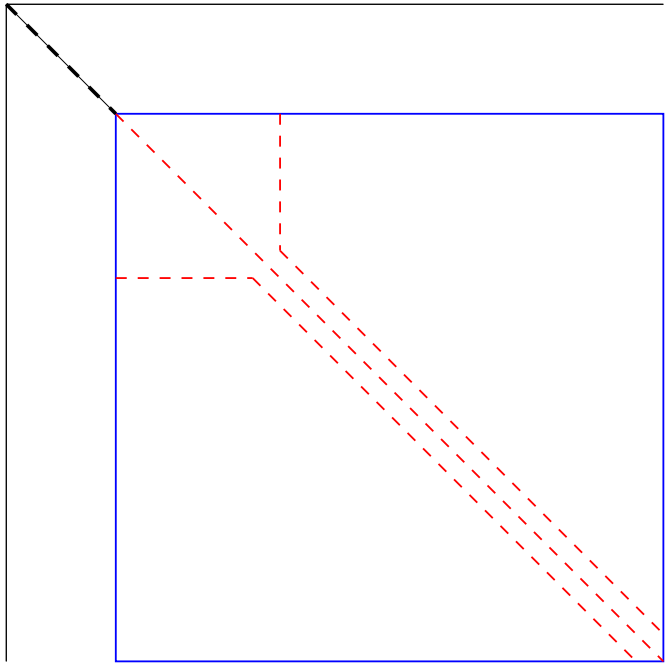
- Due to locking, no more candidates will show up in wanted area after some point → Stop.
- Similar procedure possible with subspace iteration.

The 3 types of basis vectors

Basis vectors



Matrix representation

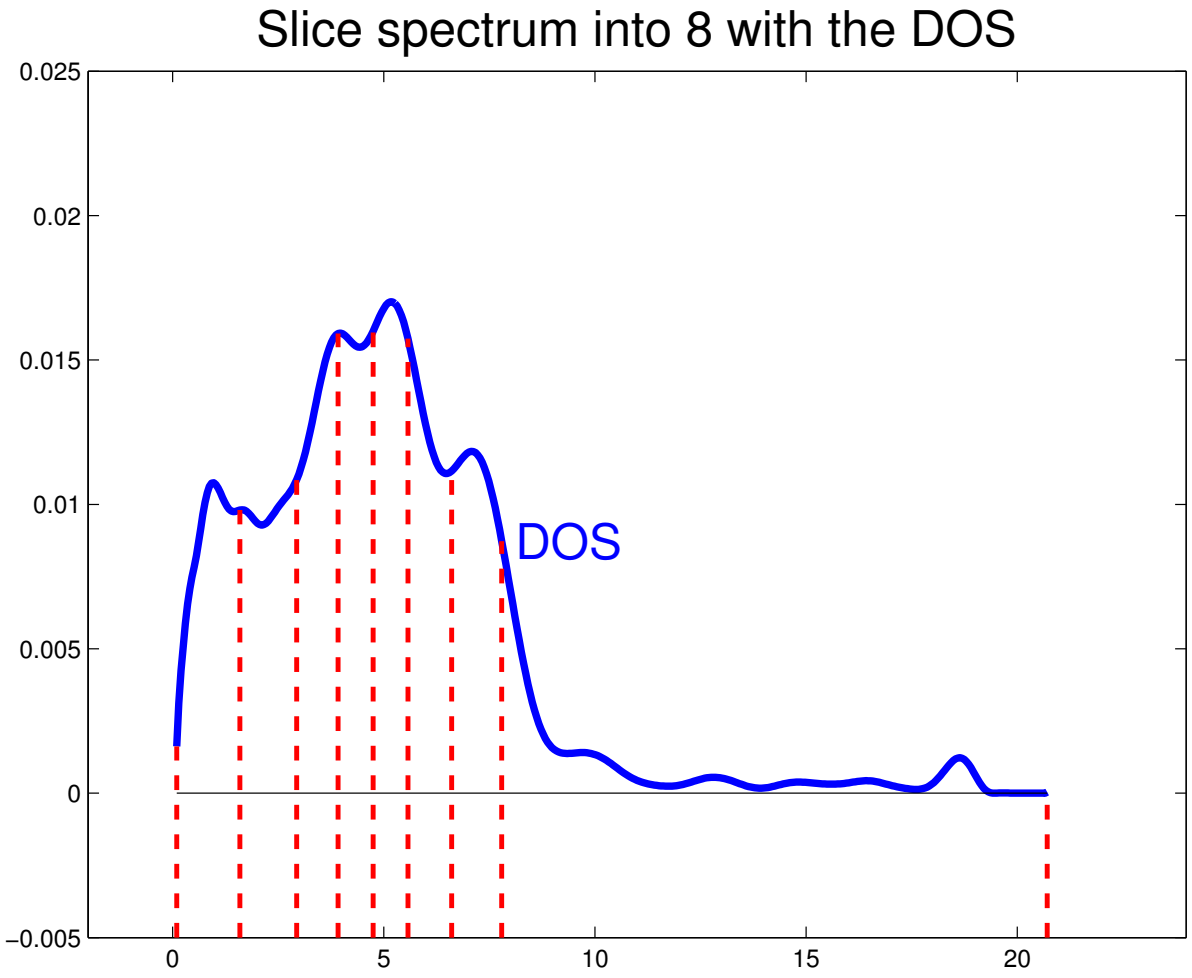


How do I slice a spectrum?

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'*
- In fact - helps mostly in equalizing memory usage..



- We must have:

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

An example

- Implemented in C: First version of (sequential). Does:
- Polynom. Filt. Thick-Restart Lanczos with deflation + spectrum slicing.
- A test example from the PARSEC collection.
- Matrix Ge99H100 $[n = 112,985, nnz = 7,892,195]$

Matrix	$[a, b]$	$[\eta, \xi]$	#eig
$\mathbf{Ge_{99}H_{100}}$	$[-1.2264, 32.7031]$	$[-0.65, -0.0096]$	250

- Asked to compute eigenvalues/vectors in 6 slices.

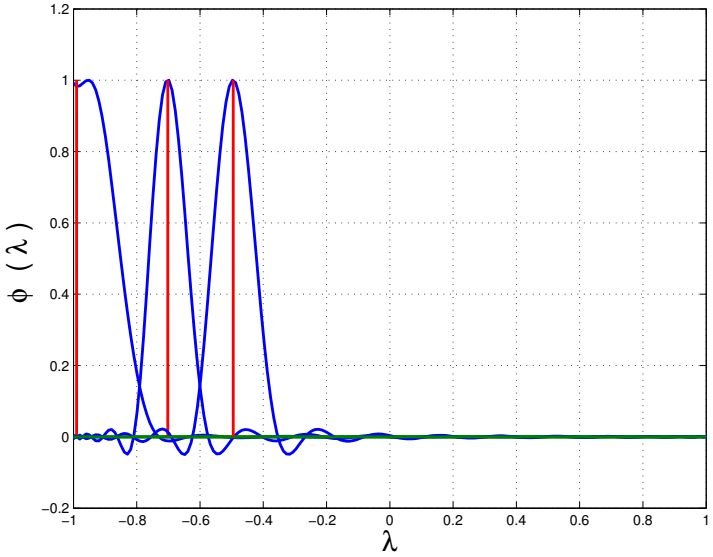
- DOS + integration $\longrightarrow$ (estimated) $\mathbf{242}$ eigenvalues in $[\xi,\ \eta]$ – roughly 40/interval [actual $\mathbf{250/6 \approx 41.66}$]

Slice #	Width	actual # e.v.	Pol. Deg.	# Matvecs
1	0.0869	38	169	50738
2	0.0708	46	220	33046
3	0.0997	42	165	49542
4	0.2542	42	71	21342
5	0.0740	38	264	79238
6	0.0547	44	301	60244

- Computed all 250 eigenvalues -
- Next step: fine tune the code and go parallel $\longrightarrow$ EVSL

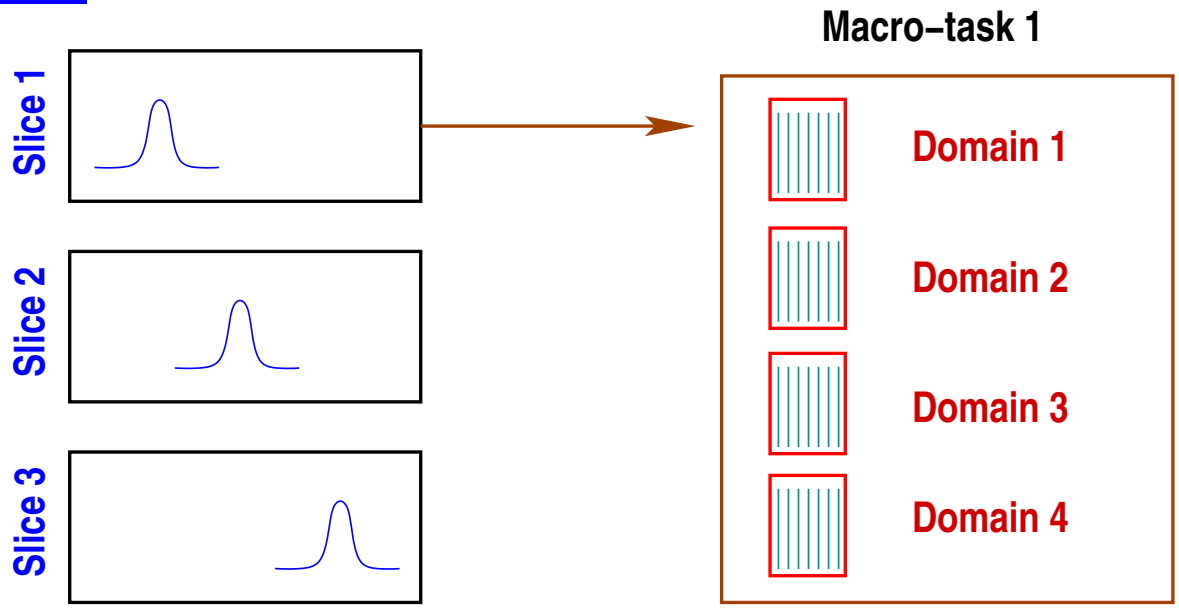
Spectrum Slicing and the EVSL project

- Part of our excited states project (DOE)
- First use polynomial filtering ...
- ... rational filters a possibility
- WIl have both TRlan and Subs. Iteration



- Several levels of parallelism:
  - Different slices
  - Domain decomposition
  - Finer grain: blocking, matvec's etc

Parallelism



The two main levels of parallelism in EVSL

Experiments

3D discrete Laplacian example ($60^3 \rightarrow n = 216,000$) *Used* $\phi = 0.8$. *Partitioning* $[0.6, 1.2]$ *into* 10 *sub-intervals.* ➤ *Goal: compute all 3,406 eigenvalues in interval [0.6, 1,2]*

i	$[\xi_i, \eta_i]$	$\eta_i - \xi_i$	$\nu_{[\xi_i,\eta_i]}$
1	$[0.60000, 0.67568]$	0.07568	337
2	$[0.67568, 0.74715]$	0.07147	351
3	$[0.74715, 0.81321]$	0.06606	355
4	$[0.81321, 0.87568]$	0.06247	321
5	$[0.87568, 0.93574]$	0.06006	333
6	$[0.93574, 0.99339]$	0.05765	340
7	$[0.99339, 1.04805]$	0.05466	348
8	$[1.04805, 1.10090]$	0.05285	339
9	$[1.10090, 1.15255]$	0.05165	334
10	$[1.15255, 1.20000]$	0.04745	348

Results

i	deg	iter	matvec	CPU time (sec)		residual	
				matvec	total	max	avg
1	116	1814	210892	430.11	759.24	6.90×10^{-09}	7.02×10^{-11}
2	129	2233	288681	587.14	986.67	5.30×10^{-09}	7.39×10^{-11}
3	145	2225	323293	658.44	1059.57	6.60×10^{-09}	5.25×10^{-11}
4	159	1785	284309	580.09	891.46	3.60×10^{-09}	4.72×10^{-11}
5	171	2239	383553	787.00	1180.67	6.80×10^{-09}	9.45×10^{-11}
6	183	2262	414668	848.71	1255.92	9.90×10^{-09}	1.13×10^{-11}
7	198	2277	451621	922.64	1338.47	2.30×10^{-09}	3.64×10^{-11}
8	209	1783	373211	762.39	1079.30	8.50×10^{-09}	1.34×10^{-10}
9	219	2283	500774	1023.24	1433.04	4.30×10^{-09}	4.41×10^{-11}
10	243	1753	426586	874.11	1184.76	5.70×10^{-09}	1.41×10^{-11}

Note: # of eigenvalues found inside each $[\xi_i, \eta_i]$ *is exact.*

Average statistics per slice for different numbers of slice (n_s), for the 3D discrete Laplacian example with $\phi = 0.8$.

n_s	deg	iter	matvec		CPU time	
			number	time	per slice	total
2	34.5	9284.5	328832.0	681.74	11817.35	23634.69
5	88.0	3891.8	347704.6	715.98	2126.97	10634.85
10	177.2	2065.4	365758.8	747.69	1116.91	11169.13
15	266.1	1351.9	361809.0	746.04	911.54	13673.12
20	356.8	1081.7	392083.3	807.46	909.62	18192.45

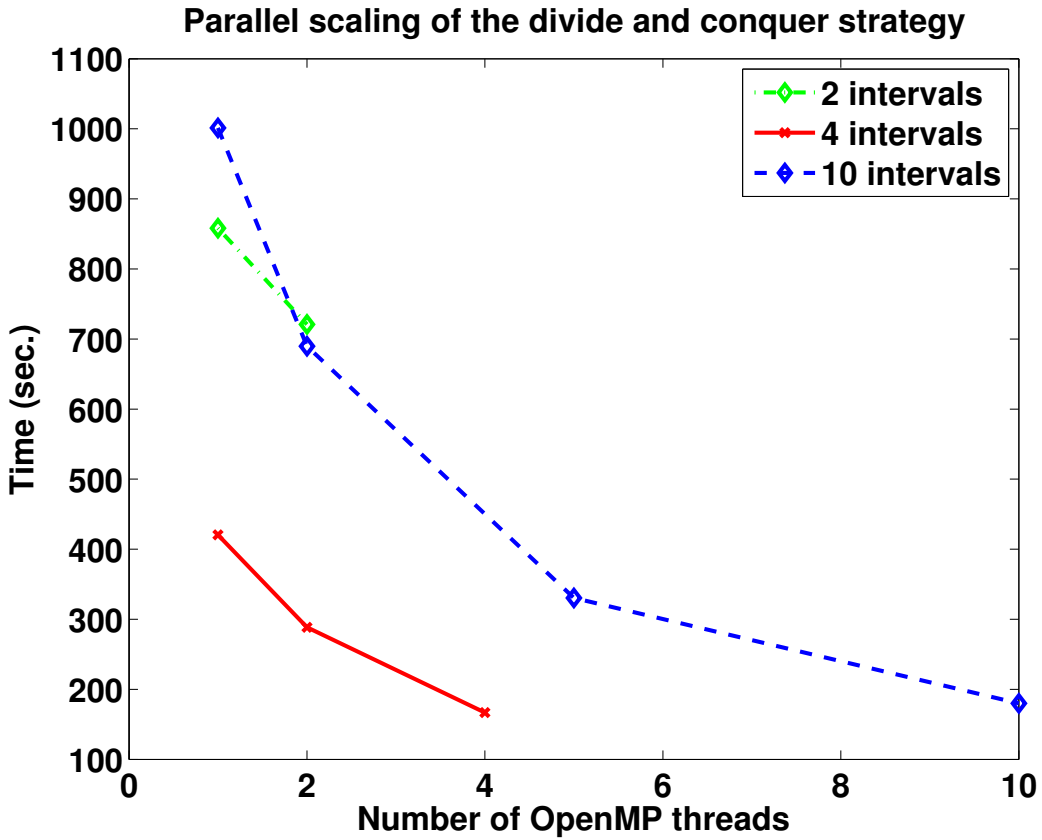
Hamiltonian matrices from the PARSEC set

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

Numerical results for PARSEC matrices

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	$\mathbf{4.20 \times 10^{-12}}$	$\mathbf{4.33 \times 10^{-13}}$

An OpenMP parallelization across 2, 4 and 10 spectral intervals of a divide and conquer approach for SiO matrix; $n = 33,401$ *and* $nnz = 1,317,655$.



- Eigenvalues computed: 1002 lowest eigenpairs.

Conclusion

- Polynom. Filter. appealing when # of eigenvectors to be computed is large and when Matvecs are inexpensive
- May not work too well for generalized eigenvalue problem...
- ... of for matrices with spectra that have large outliers.
- For these cases we plan to add rational filtering
- Issue there: *Iterative solvers!*
- Part of EVSL [no MPI, subspace iter + TRlan, spectrum slicing, ...] to be released within this summer.