UNIVERSITY OF MINNESOTA TWIN CITIES

# Divide and conquer algorithms for large eigenvalue problems

*Yousef Saad*

*Department of Computer Science and Engineering*

*University of Minnesota*

*PMAA 14*

*Lugano, July 4, 2014*

## *Collaborators:*

- Joint work with: Haw-ren Fang and Vassileos Kalantzis
- Grady Schoefield and Jim Chelikowsky [UT Austin] [windowing into PARSEC]
- Work supported in part by NSF (to 2012) and now by DOE

## *Introduction*

*Q:* How do you 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) $\sigma$;
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

Q: What if factoring $A$ is too expensive (e.g., Large 3-D similation)?

A: Obvious answer: Use iterative solvers ...

- But: systems highly indefinite $\rightarrow$ Won't work well.
- Other common issue: Need a very large number of eigenvalues and eigenvectors
- Applications: Excited states in quantum physics: TDDFT, GW, ... or just plain Density Functional Theory (DFT)
- Example: in real-space code (PARSEC), Hamiltonian can be of size a few Millions, and number of ev's in the tens of thousands

## *I. Polynomial filtered Lanczos*

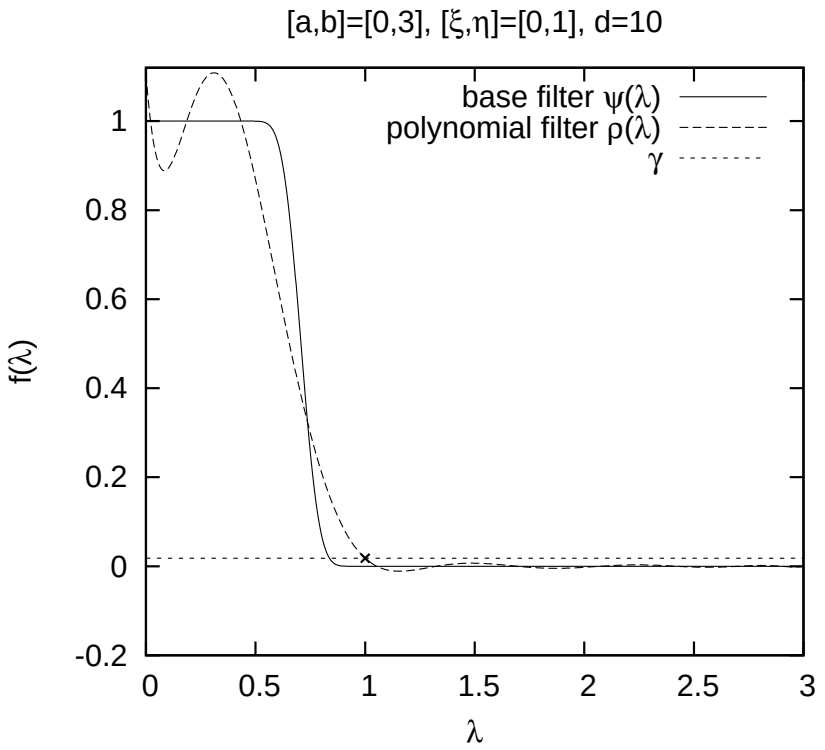
- Possible solution: Use Lanczos with polynomial filtering.
- In short: just replace $(A - \sigma I)^{-1}$ in S.I. Lanczos by $p_k(A)$ where $p_k(t) =$ polynomial of degree $k$
- Idea not new (and not too popular in the past)

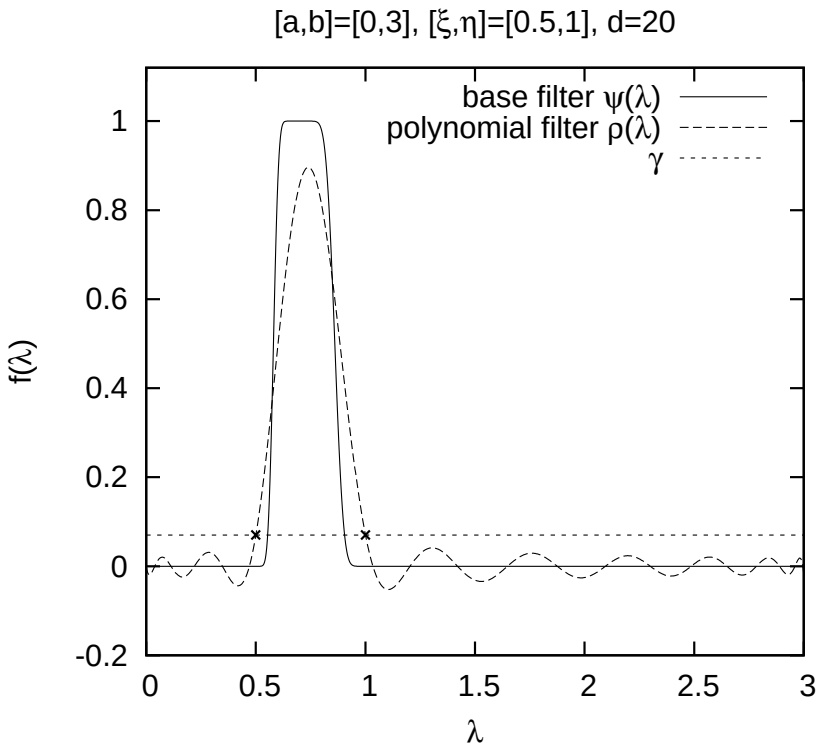
*What is new?*

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

- Important application: compute the spectrum by pieces ['spectrum slicing' a term coined by B. Parlett]
- Main attraction: reduce cost of orthogonalization

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

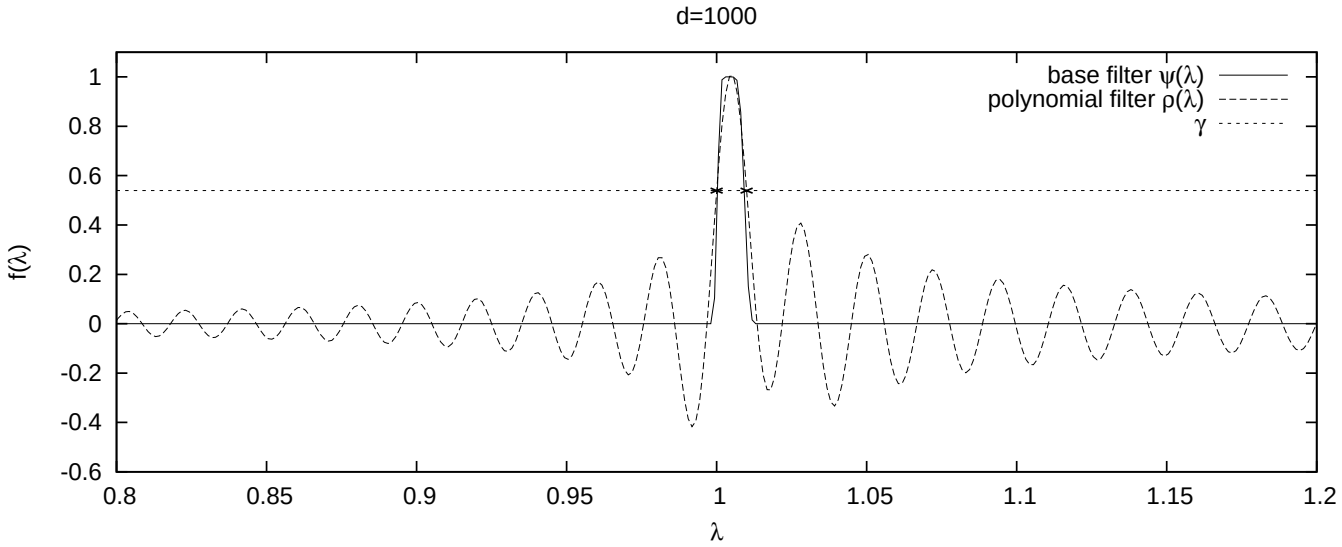


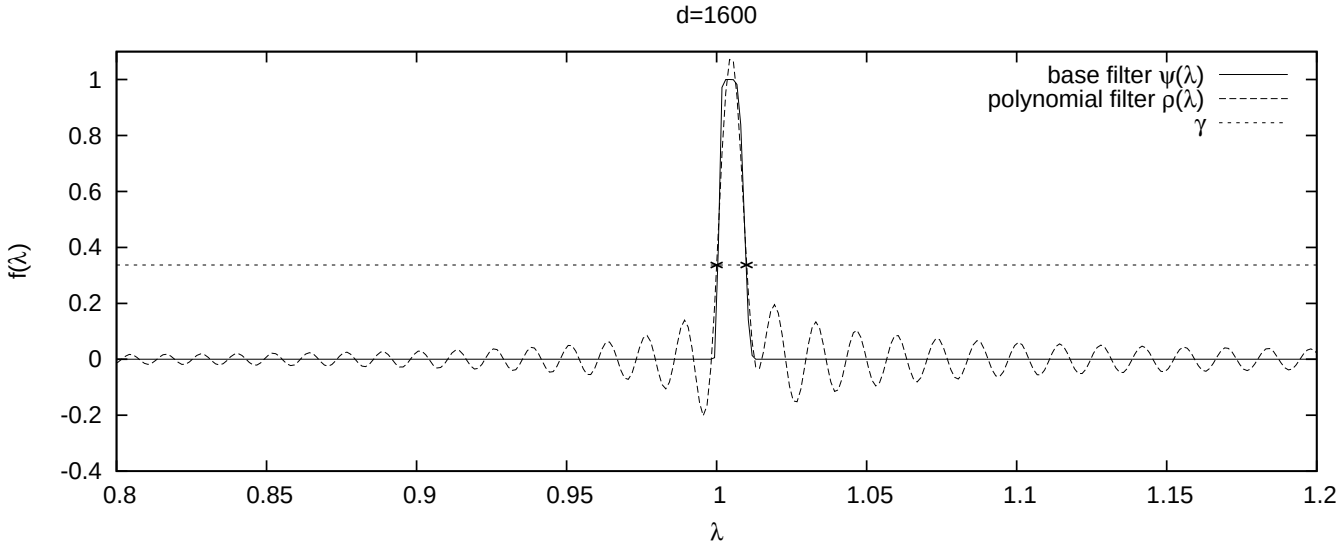

- See Reference on Lanczos + pol. filtering: Bekas, Kokiopoulou, YS (2008) for motivation, etc.
- H.-r Fang and YS "Filtlan" paper [SISC,2012] and code

# *Misconception: High degree polynomials are bad*

Degree 1000 (zoom)



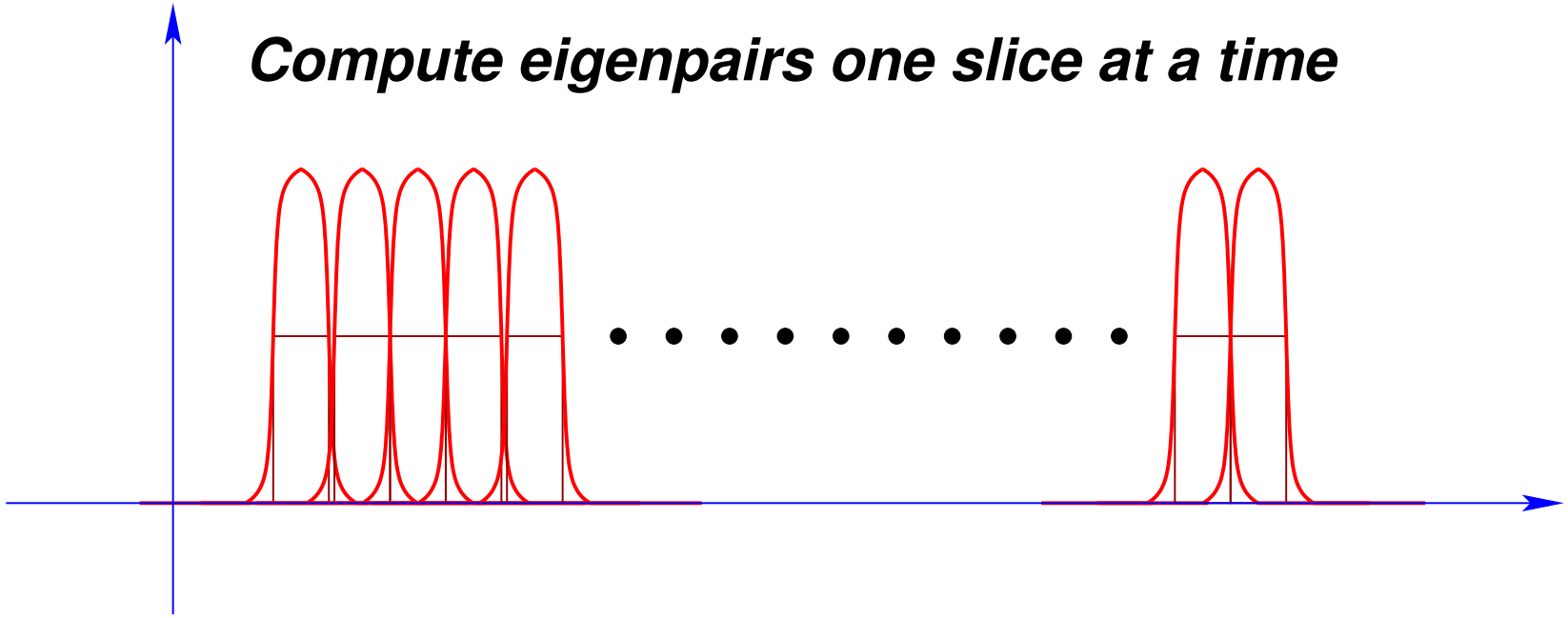

Degree 1600 (zoom)



## *'Spectrum slicing' or 'windowing'*

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

- Idea: Get the spectrum by 'slices' or 'windows'
- Can get a few hundreds or thousands of vectors at a time.




- Deceivingly simple looking idea.
- Issues:
  - Deal with interfaces : duplicate/missing eigenvalues
  - Window size [need estimate of eigenvalues]
  - polynomial degree

## *Spectrum slicing in PARSEC*

- Implemented in our code:

Pseudopotential Algorithm for Real-Space Electronic Calculations (PARSEC)

- See :

'*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.

- Refer to this paper for details on windowing and 'initial proof of concept'

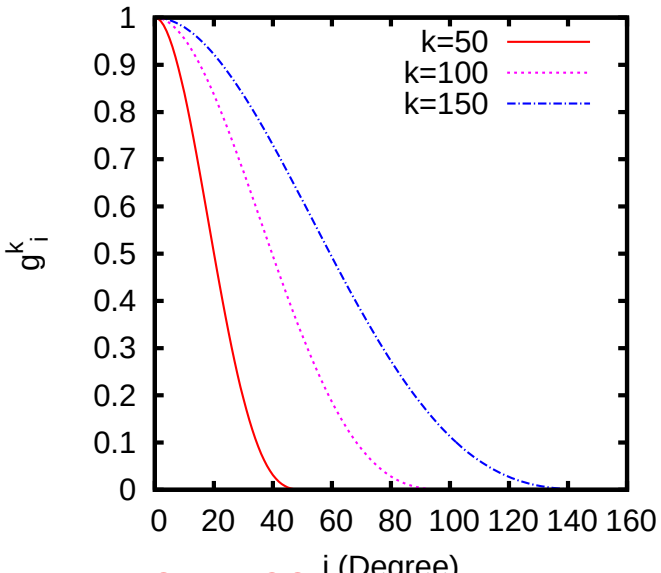
# *Computing the polynomials: Jackson-Chebyshev*

Chebyshev-Jackson approximation of a function $f$:

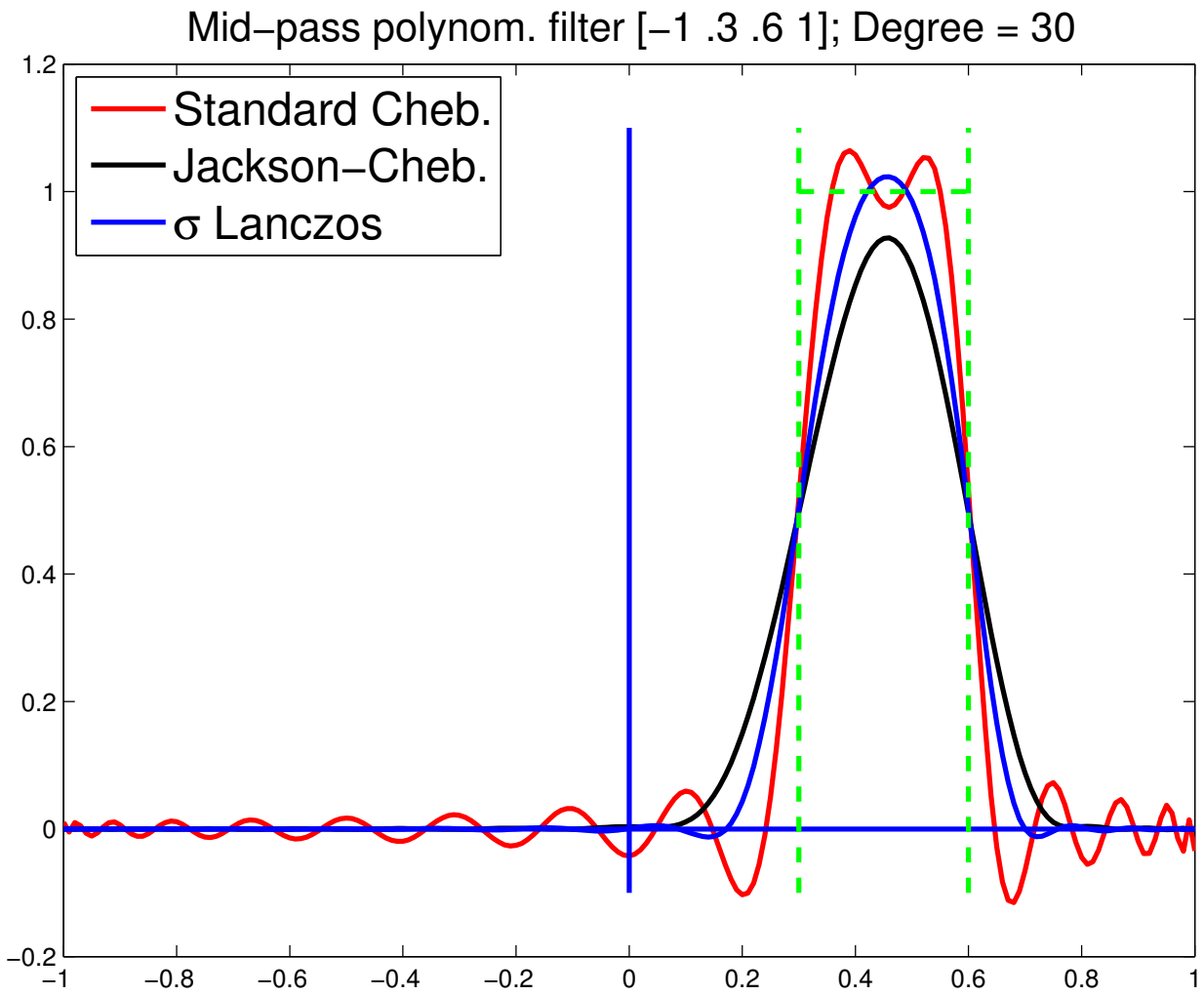
$$f(x) \approx \sum_{i=0}^{k} g_i^k \gamma_i T_i(x)$$

$$\gamma_i = \frac{2-\delta_{i0}}{\pi} \int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} f(x)dx \quad \delta_{i0} = \text{ Kronecker symbol}$$
$$= \text{ explicitly known}$$

- The $g_i^k$'s for $k$=50,100,150 $\rightarrow$
- The $g_i^k$'s dampen high order terms
- Alternative: Lanczos $\sigma$-damping



# *A mid-pass (barrier) filter - 3 damping methods*



## *Tests – Test matrices*

➤ 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 $\eta$)

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

➤ 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

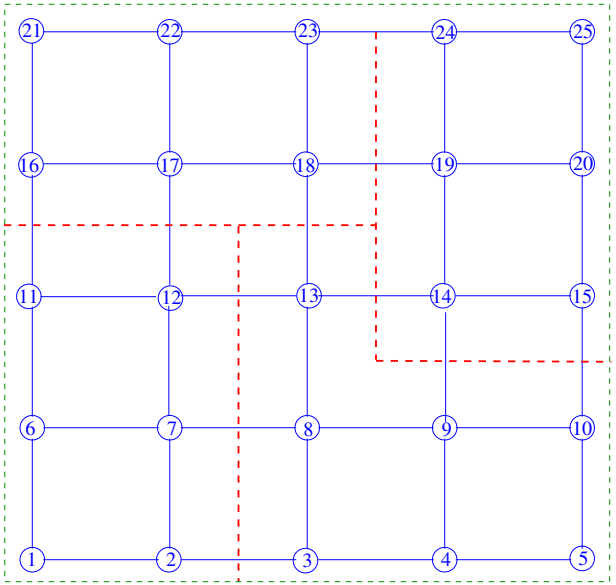
| method | degree | # iter | # matvecs | memory |
|---|---|---|---|---|
| mid-pass filter | $600$ | 1,400 | 840,000 | 10,913 |
| | $1,000$ | 950 | 950,000 | 7,640 |
| | $1,600$ | 710 | 1,136,000 | 6,358 |

## Results for Laplacian – CPU times

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

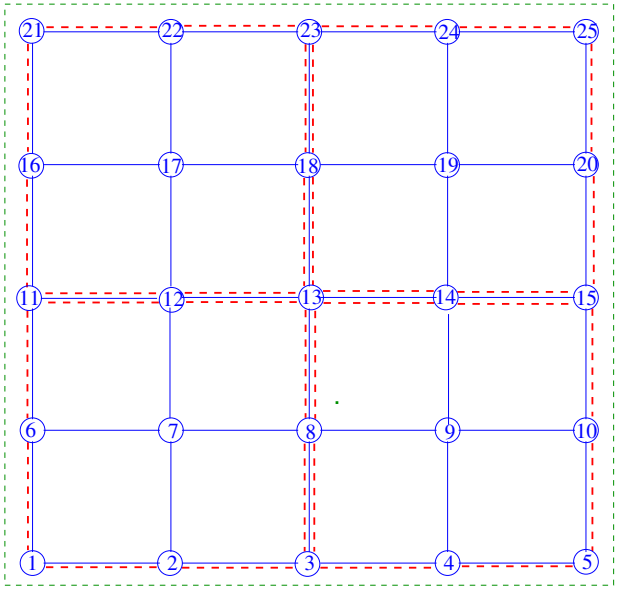
# *II. Domain decomposition ideas*

- Main idea: Cauchy integral-based method [e.g. FEAST]
- ... within a domain-decomposition framework:




← edge-separators

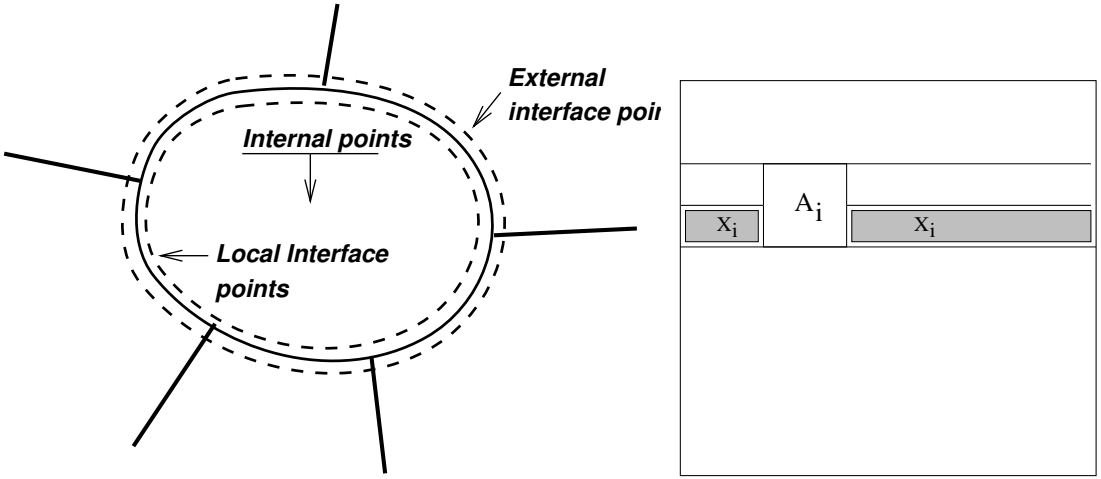
vertex-separators →




Two classical ways of partitioning a graph.

- We use edge-separators (vertex-based partitioning)

Distributed graph and its matrix representation

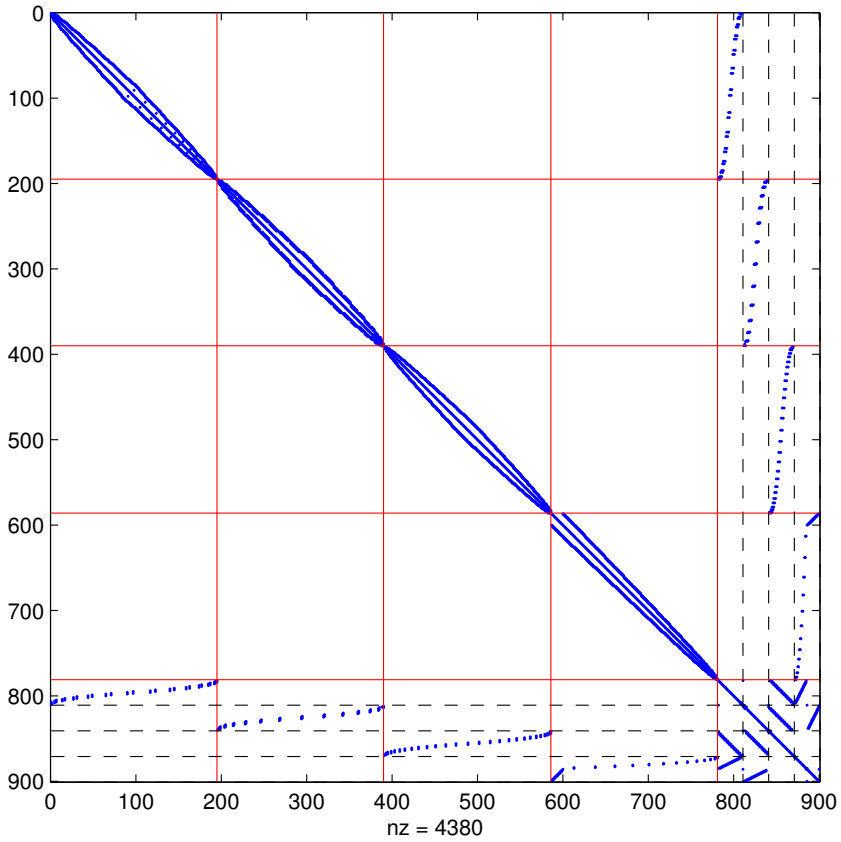



- Stack all interior variables $u_1, u_2, \cdots, u_p$ into a vector $u$, then interface variables $y$

- 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:

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

## *First idea: Schur complement techniques (On-going work)*

- Eliminate interior variables $u_i$ – Result:

$$\underbrace{\begin{pmatrix} S_1(\lambda) & E_{12} & \dots & E_{1p} \\ E_{21} & S_2(\lambda) & \dots & E_{2p} \\ \vdots & & \ddots & \vdots \\ E_{p1} & E_{p,2} & \dots & S_p(\lambda) \end{pmatrix}}_{S(\lambda)} \underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \end{pmatrix}}_{y} = 0$$

$S_i(\lambda) = C_i - E_i^T (B - \lambda I)^{-1} E_i \equiv$ Local Schur Complement

- Nonlinear eigenvalue problem.

$$S(\lambda) y = 0$$

- Involves only interface variables.

- Related to AMLS – see also Bekas and YS (2005)

## *Next: Schur complements + FEAST*

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

$$(A - sI)^{-1} = \left[\begin{array}{c|c} * & -(B - sI)^{-1} E S(s)^{-1} \\ \hline * & S(s)^{-1} \end{array}\right]$$

- Then, Cauchy integral formula for spectral projector yields:

$$P = \frac{-1}{2i\pi} \int_\Gamma R(s) ds \equiv \left[\begin{array}{c|c} * & -W \\ \hline * & G \end{array}\right] \quad \text{with}$$

$$G = \frac{-1}{2i\pi} \int_\Gamma S(s)^{-1} ds, \quad W = \frac{-1}{2i\pi} \int_\Gamma (B - sI)^{-1} E S(s)^{-1} ds$$

- Advantage: Does not involve inverse of whole matrix

- Let $$P = [P_1, P_2] \equiv \left[\begin{array}{c|c} * & -W \\ \hline * & G \end{array}\right]$$
- We know how to compute $P_2$ or $P_2 \times$ `randn` $(s, ns)$

*Q:* How can we recover eigenvectors of $A$ from $P_2$?

*A:* Write $P$ as $P = VV^T$, and $V = \begin{pmatrix} V_u \\ V_s \end{pmatrix}$ then note:

$$P_2 = VV_s^T$$

- Just capture the range of $P_2$
- Can use Lanczos on $P_2 P_2^T$ or just a random $X \in \mathbb{R}^{s \times ns}$
- Advantage of Lanczos: stops when dimension reached
- Drawbacks: 1) sequential; 2) $\approx$ Doubles the work
- So far: Both idea tested in matlab

## *Need better poles*

- Approach is a one-shot method [no easy way to iterate]

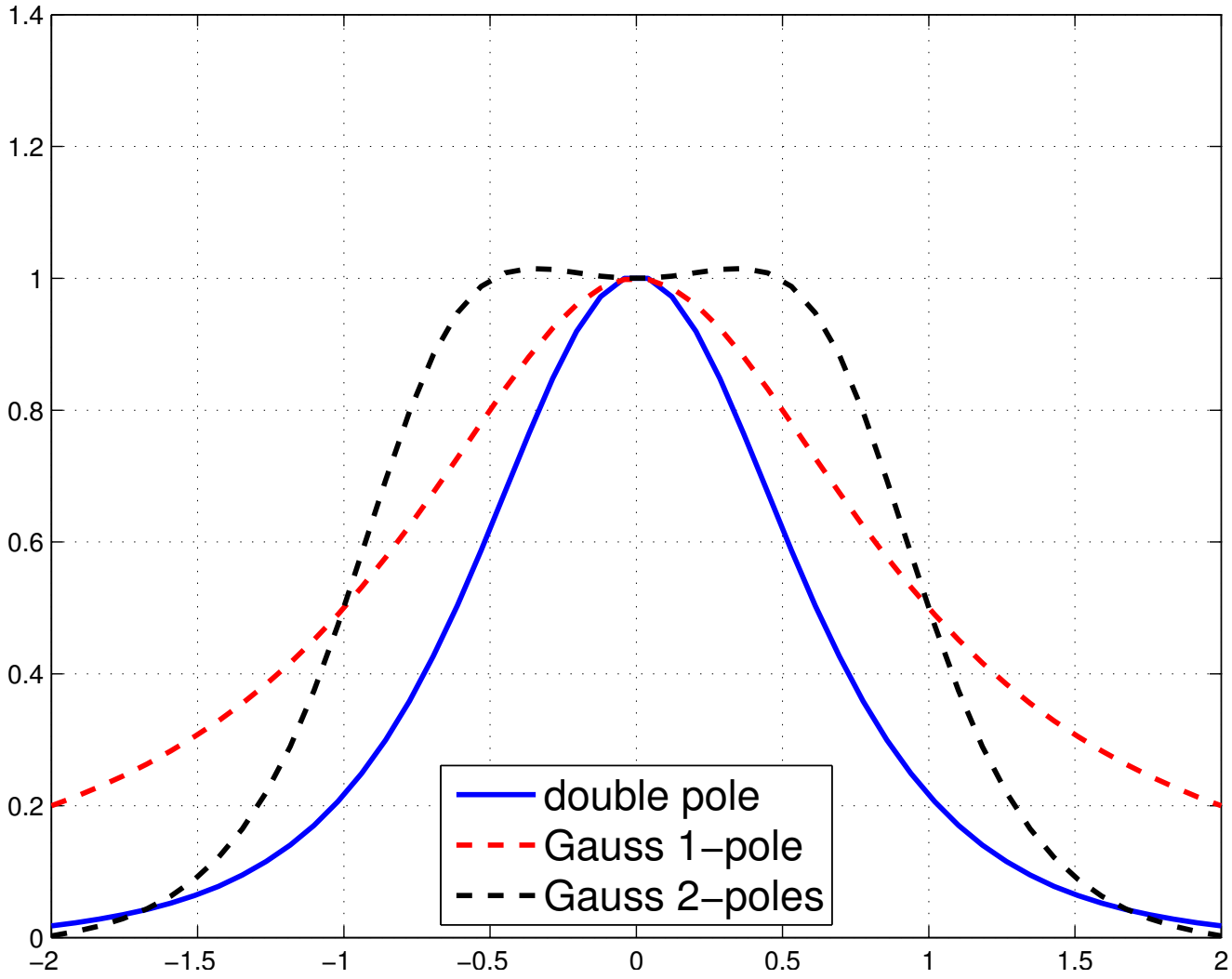
*Q:* How can we improve accuracy?

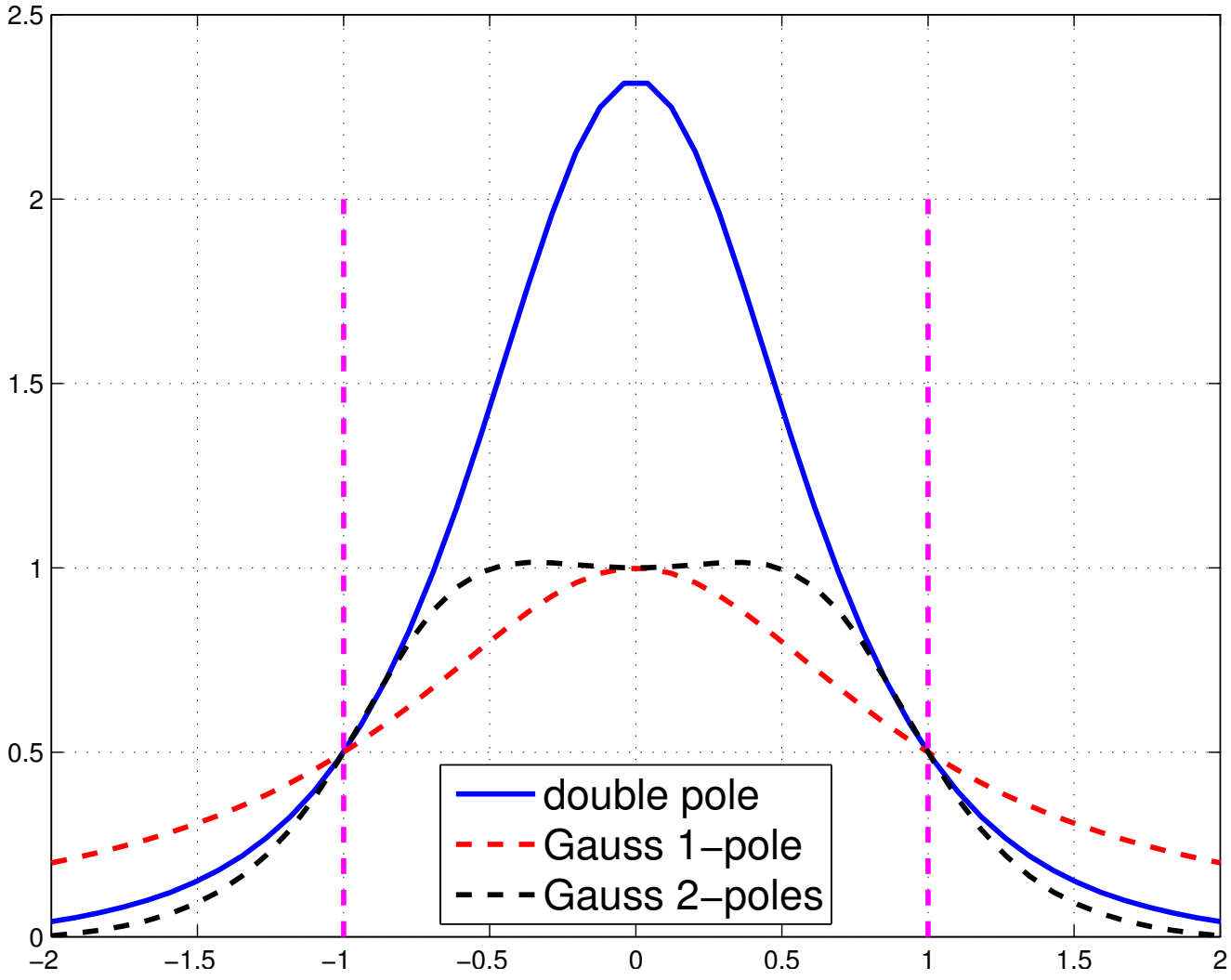
*A:* Select poles carefully.

- Current choices: trapezoidal rule, Gauss, Zolotarev,...
- None of these allows for repeated ('multiple') poles e.g.,

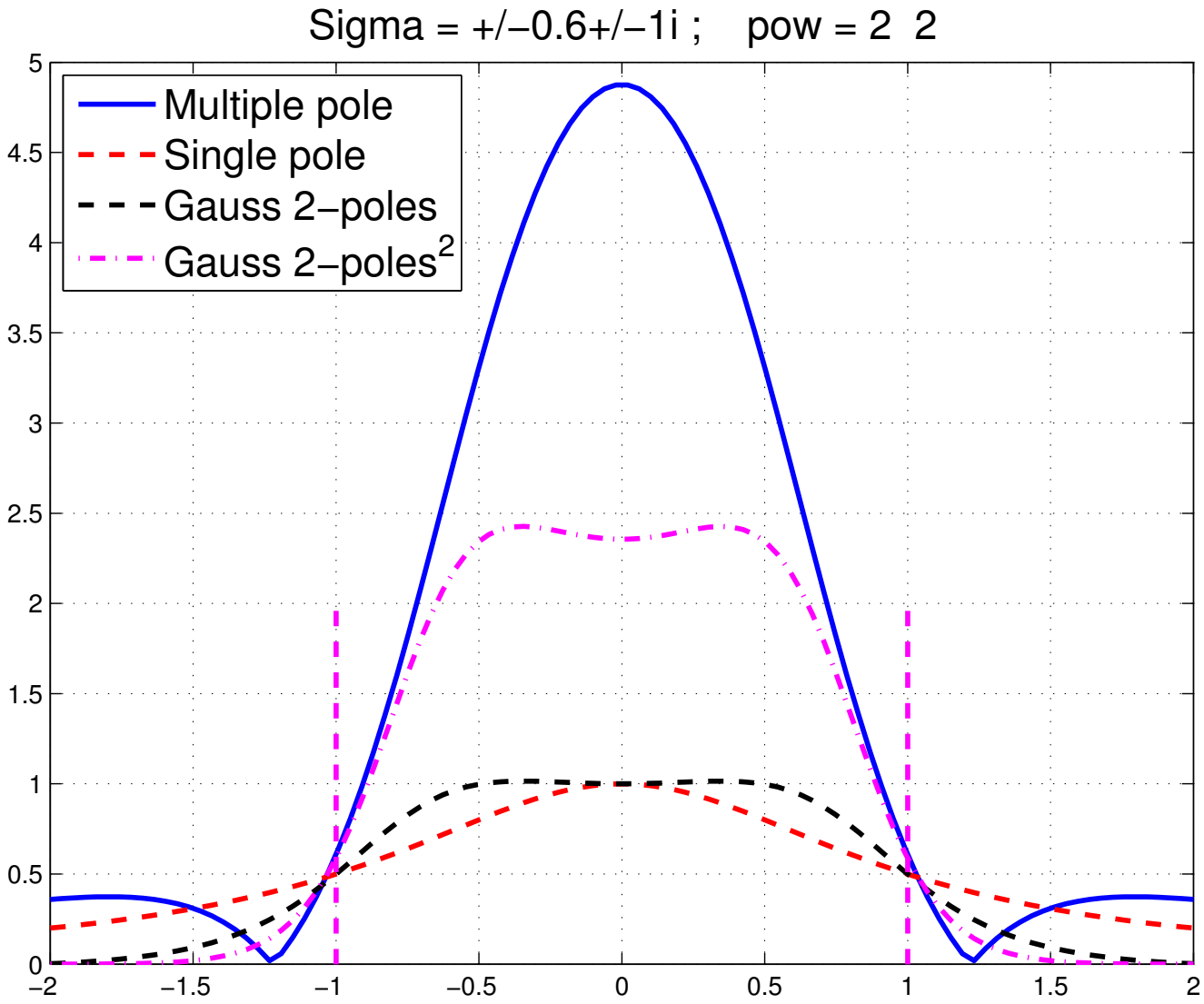
$$r(z) = \frac{\alpha_1}{z-\sigma} + \frac{\alpha_2}{(z-\sigma)^2} + \cdots + \frac{\alpha_k}{(z-\sigma)^k}$$

- This can be useful for any rational filtering approach
- Next: See what we can do with one double pole

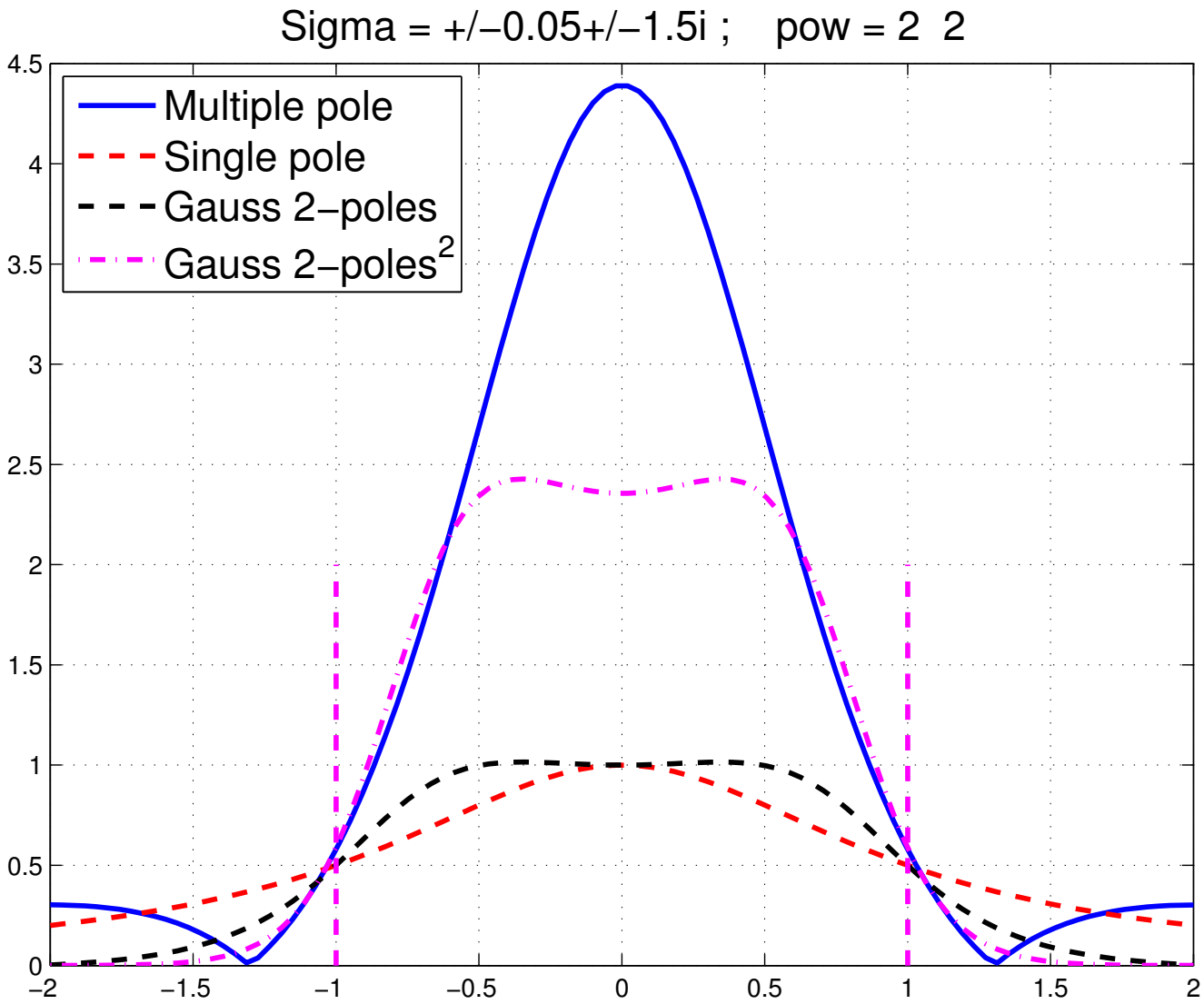

1.4
1.2
1
0.8
0.6
0.4
0.2
0
−2
−1.5
−1
−0.5
0
0.5
1
1.5
2
double pole
Gauss 1−pole
Gauss 2−poles


2.5
2
1.5
1
0.5
0
−2
−1.5
−1
−0.5
0
0.5
1
1.5
2
double pole
Gauss 1−pole
Gauss 2−poles

# Two double poles + comparison with compounding



## *Who needs a circle?* Two poles$^2$ far from the origin



## *Conclusion*

*Part I:*

- `FiltLan` is appealing when number of eigenvectors to be computed is large and when Matvecs are not too expensise
- Will not work too well for generalized eigenvalue problem
- Code available here `www.cs.umn.edu/~saad/software/filtlan`

*Part II:*

- Many ideas still to explore in Domain Decomposition for interior eigenvalue problems
- Viewpoint: look at rational filtering from angle of approx. theory