Efficient subspace iteration with Chebyshev-type filtering

Bruno Lang

Bergische Universität Wuppertal, Mathematik / Angewandte Informatik

PMAA2016, July 8, 2016

The ESSEX project: Equipping Sparse Solvers for Exascale

This work is supported by DFG Deutsche Forschungsgemeinschaft through the

Priority Programme 1648 "Software for Exascale Computing" SPPEXA

FAU Friedrich-Alexander Universität Erlangen-Nürnberg — G. Hager, M. Kreutzer, F. Shahzad, G. Wellein

Ernst Moritz Arndt Universität Greifswald — A. Alvermann, H. Fehske, A. Pieper

DLR — A. Basermann, M. Röhrig-Zöllner, J. Thies

Bergische Universität Wuppertal — M. Galgon, S. Huber, B. Lang

The University of Tokyo — K. Nakajima, A. Ida, M. Kawai

University of Tsukuba 筑波大学 — T. Sakurai, Y. Futamura, A. Imakura

Outline

Subspace iteration with Rayleigh–Ritz extraction

Improving the coefficients of the polynomials

Adaptive control of the degree

A priori information about the spectrum

High performance computational kernels

Subspace iteration with Rayleigh–Ritz extraction

Given: $\mathsf{A} \in \mathbb{C}^{n\times n}$, $I_\lambda = [\alpha, \beta] \subset \mathbb{R}$

Sought: Those eigenpairs (λ, v) of A such that $\lambda \in I_\lambda$

Start with a subspace $\mathsf{Y} \in \mathbb{C}^{n\times m}$ of suitable dimension m

While not yet converged

Compute $\mathsf{U} = f(\mathsf{A}) \cdot \mathsf{Y}$ for a suitable function f

Compute $\mathsf{A_U} = \mathsf{U^*AU}$ and $\mathsf{B_U} = \mathsf{U^*U}$

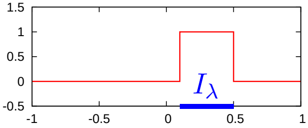
Solve the size-m generalized EVP $\mathsf{A_U W} = \mathsf{B_U W}\,\Lambda$

Replace Y with $\mathsf{U} \cdot \mathsf{W}$

Filter functions

Given an orthonormal set of eigenpairs $(\mathsf{x}_i, \lambda_i)$ of A and an arbitrary vector $\mathsf{y} = \sum \xi_i \mathsf{x}_i$, $f(\mathsf{A}) \cdot \mathsf{y}$ should

- retain the "wanted" components $\xi_i \mathsf{x}_i$, $\lambda_i \in I_\lambda$, and
- cancel the "unwanted" components $\xi_i \mathsf{x}_i$, $\lambda_i \notin I_\lambda$.



In practice, this function $f = \chi_{I_\lambda}$ must be approximated:

- Rational approximation, e.g., FEAST
- Polynomial approximation

Outline

Polynomial approximation and kernel smoothing I

Expansion w.r.t. Chebyshev polynomials of the first kind yields

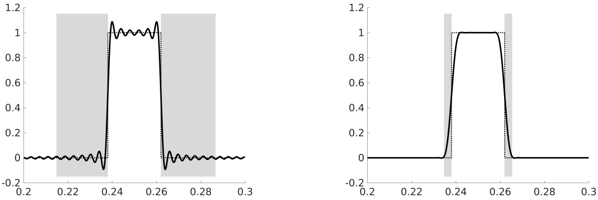
$$\chi_{[\alpha,\beta]}(x) \approx \sum_{k=0}^{d} c_k T_k(x) \ ,$$

where

$$\begin{aligned} T_0(x) &\equiv 1 \ , \\ T_1(x) &= x \ , \\ T_k(x) &= 2x \cdot T_{k-1}(x) - T_{k-2}(x) \ , \quad k \geq 2, \end{aligned}$$

and

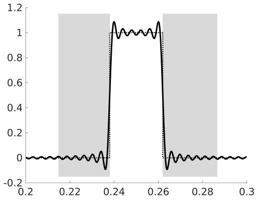
$$\begin{aligned} c_0 &= \frac{1}{\pi} \cdot (\arccos \alpha - \arccos \beta) \ , \\ c_k &= \frac{2}{k\pi} \cdot (\sin(k \cdot \arccos \alpha) - \sin(k \cdot \arccos \beta)) \ , \quad k \geq 1. \end{aligned}$$



- Left: Degree-1600 Chebyshev approximation $p(x)$ to $\chi_{[\alpha,\beta]}$ for $[\alpha, \beta] = [0.238, 0.262]$ ($\rightsquigarrow$ Gibbs oscillations)
- Right: With (Lanczos, $\mu = 2$) kernel smoothing: replace c_k with $g_k \cdot c_k$, where (Lanczos)

$$g_k = \left(\operatorname{sinc} \frac{k}{d+1}\right)^{\mu} , \quad k \geq 0 , \quad \text{with} \quad \operatorname{sinc} \xi = \frac{\sin(\pi \xi)}{\pi \xi} .$$

The target for improvement

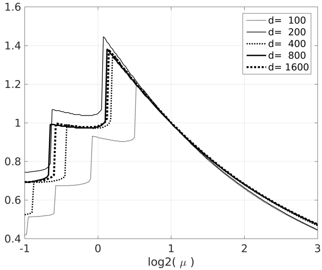


- Light grey areas: The "damping condition" $|p(x)| \leq \tau_{\text{outside}} = 0.01$ may be violated
- Try to reduce the margin (i.e., the width of the grey areas)
- For any filter, let

$$\text{gain} = \frac{\text{margin(Chebyshev approx with Lanczos kernel, } \mu = 2)}{\text{margin(filter under consideration)}}$$

Lanczos smoothing with optimized μ

- No need to have $\mu \in \mathbb{N}$:



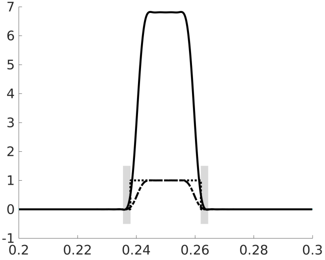
gain for $[\alpha, \beta] = [0.238, 0.262]$

- The optimum μ can be determined from α, β, and d by considering $p(x)$, without evaluating $p(A)$

Shrinking the interval I

- Determine a filter $\widehat{p}(x)$ for a smaller interval $[\alpha, \beta] \mapsto [\widetilde{\alpha}, \widetilde{\beta}] \subseteq [\alpha, \beta]$
- In general, $\widehat{p}(\alpha)$ and $\widehat{p}(\beta)$ will be smaller than 0.5 $\Rightarrow$ scale the polynomial,

$$\widetilde{p} = \varphi \cdot \widehat{p} \,, \quad \text{where} \quad \varphi = \frac{0.5}{\min\{\widehat{p}(\alpha), \widehat{p}(\beta)\}} \,.$$

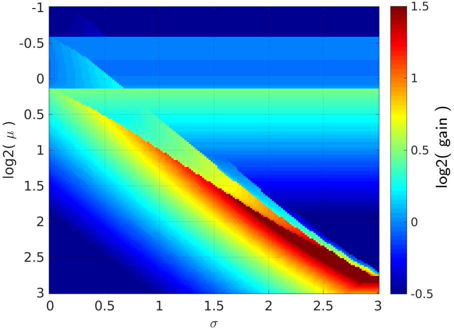


$[\alpha, \beta] = [0.238, 0.262]$,
$[\widetilde{\alpha}, \widetilde{\beta}] = [0.24032, 0.25969]$,
$d = 1600$

How to choose $\widetilde{\alpha}$ and $\widetilde{\beta}$?

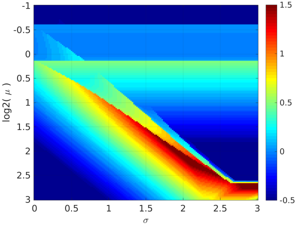
- Let $\sigma \geq 0$ such that

$$\widetilde{\alpha} := \alpha + \sigma \frac{p(\alpha)}{p'(\alpha)} \leq \frac{\alpha + \beta}{2} \leq \beta + \sigma \frac{p(\beta)}{p'(\beta)} =: \widetilde{\beta}$$

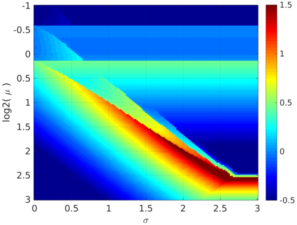


Shrinking the interval III

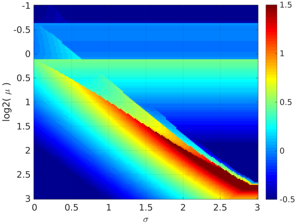
This pattern is rather generic:



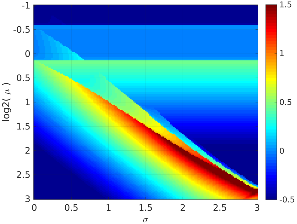
$[\alpha, \beta] = [-0.984, -0.960]$, $d = 400$



$[\alpha, \beta] = [0.560, 0.584]$, $d = 1131$



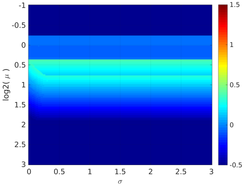
$[\alpha, \beta] = [-0.012, 0.012]$, $d = 1600$



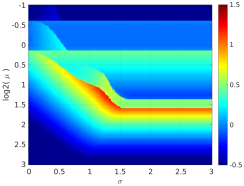
$[\alpha, \beta] = [0.150, 0.350]$, $d = 200$

Shrinking the interval IV

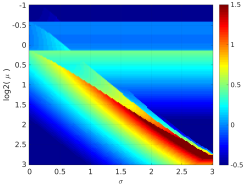
There are three qualitatively different patterns:



$[\alpha, \beta] = [0.238, 0.262]$
$d = 141$
"low degree"



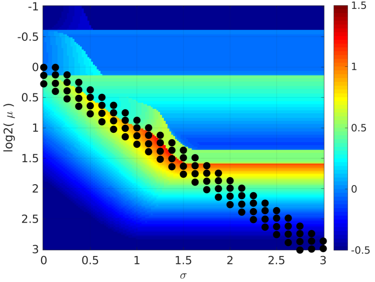
$[\alpha, \beta] = [0.238, 0.262]$
$d = 565$
"critical degree"



$[\alpha, \beta] = [0.238, 0.262]$
$d = 1600$
"high degree"

Shrinking the interval V

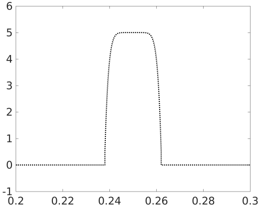
In all cases, the best gain(μ, δ) is found close to the diagonal $\log_2(\mu) = \sigma$:



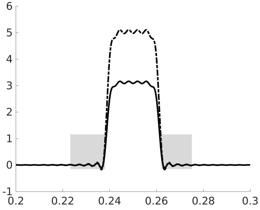
- Search on a grid along the BAND
- Optionally followed by refined seach (PATH)

Iteratively compensating filters I

Start with a suitable target function f_1 (e.g., remove the upper corners of the window)



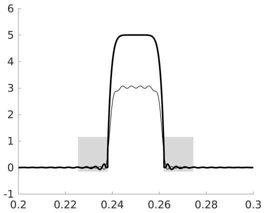
Determine approximation (dash-dotted) and scale to achieve $\min\{p_1(\alpha), p_1(\beta)\} = 0.5$ (solid)



Iteratively compensating filters II

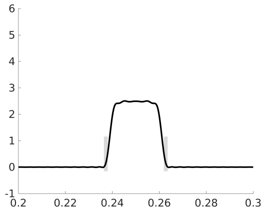
"Compensate" for the oscillations by prescribing $f_2(x) = -\rho \cdot p_1(x)$ outside $[\alpha, \beta]$ (thick line) and determine new approximation (thin line)

We used $\rho = 0.75$



Iterate until no more improvement

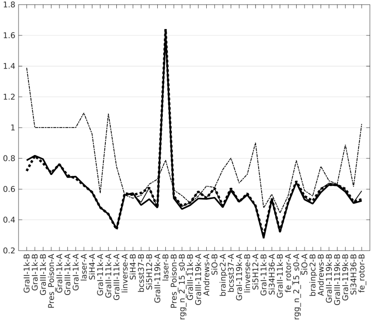
Resulting filter function $p = p_{34}$



- No closed formula for the c_k in the expansion ($\rightsquigarrow$ quadrature)

Numerical results I

Matlab: Number of overall MVMs vs. Lanczos ($\mu = 2$)



- Dotted: Shrunken Lanczos (BAND)
- Solid thin: Shrunken Lanczos (BAND and PATH)
- Dash-dotted: Iteratively compensating
- Solid thick: Combined

Numerical results II

Runs on Emmy (two 2.2GHz 10-core Xeon 2260v2 per node) at Erlangen Regional Computing Center

Filter	Final degree	Overall MVMs	Overall time	Time for coeffs
Topological insulator, $n = 268\,435\,456$, 148 evals, 128 nodes à 20 cores				
Lanczos ($\mu = 2$)	$4\,525$	$5\,598\,502$	7.11 h	0.00 h
Improved (combined)	$2\,255$	$2\,602\,360$	3.44 h	0.02 h
Topological insulator, $n = 67\,108\,864$, 148 evals, 64 nodes à 20 cores				
Lanczos ($\mu = 2$)	$2\,262$	$2\,726\,112$	1.97 h	0.00 h
Improved (combined)	$1\,127$	$1\,482\,035$	1.10 h	0.01 h

Outline

Subspace iteration with Rayleigh–Ritz extraction

Improving the coefficients of the polynomials

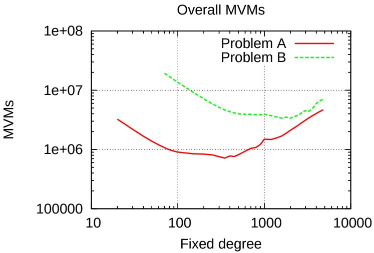
Adaptive control of the degree

A priori information about the spectrum

High performance computational kernels

How to choose the degree without prior knowledge ? I

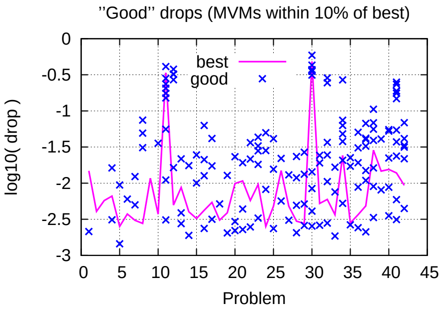
Overall MVM count for `linverse` ($n = 11,999$)



(Same matrix, both intervals contain roughly 300 eigenvalues)

How to choose the degree without prior knowledge ? II

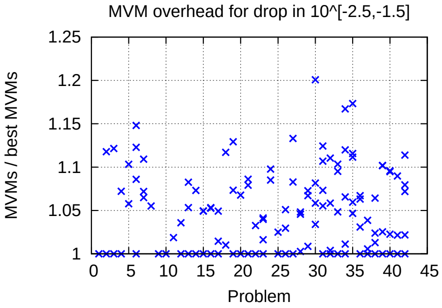
- Run the algorithm for different fixed degrees
- Count overall MVMs for each run
- Determine "drop" (of smallest residual) just before convergence sets in
- Determine the drops that lead to "close-to-best" MVM counts



$\Rightarrow$ most close-to-best runs achieved drop $\in [10^{-2.5}, 10^{-1.5}]$

How to choose the degree without prior knowledge ? III

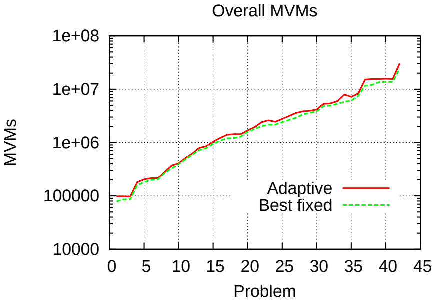
- Can going for a drop $\in [10^{-2.5}, 10^{-1.5}]$ be dangerous ?
- Determine "MVM overhead" for all runs that reached such drops



$\Rightarrow$ at most 20% more MVMs than the best fixed-degree run

How to choose the degree without prior knowledge ? IV

- Increase degree for the next iteration if drop $> 10^{-2}$

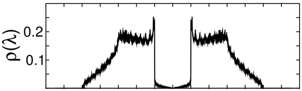


- On average 14% more MVMs than the best fixed-degree run
- In two cases $\geq 30\%$ more MVMs

Outline

Many applications require (approximate) *density of states* (DOS) of A,

$$\rho(\lambda) = \frac{1}{n} \sum_{k=1}^{n} \delta(\lambda - \lambda_k)$$



The Kernel Polynomial Method (KPM):

- Moment expansion of ρ:

$$\rho(x) \quad = \quad \mu_0 \, \phi_0(x) + 2 \sum_{m=1}^{\infty} \mu_m \, \phi_m(x) \ ,$$

where

$$\phi_m(\xi) = \frac{T_m(\xi)}{\pi \sqrt{1 - \xi^2}}$$

and

Estimating the number of eigenvalues in I_λ: The KPM II

$$\mu_m = \langle \rho, \phi_m \rangle = \int_{-1}^{+1} \rho(\xi)\, T_m(\xi) \mathrm{d}\xi = \frac{1}{n} \operatorname{trace}(\, T_m(\mathsf{A})\,) \ ,$$

with

$$\operatorname{trace}(\, T_m(\mathsf{A})\,) \approx \frac{1}{R} \sum_{r=1}^{R} \mathsf{r}_r^* \, T_m(\mathsf{A})\, \mathsf{r}_r$$

(r_r: suitable random vectors)

- Once you have the μ_m, evaluating ρ is easy (and cheap: FFT-type)
- The KPM uses the same Chebyshev kernel, with a few inner products after each MVM

Selecting a suitable subspace dimension and degree

- For certain distributions of eigenvalues ($\rightsquigarrow$ KPM), e.g.,
  - "flat"
  - "linearly increasing" from a (pseudo-)gap in I_λ

  "good" values for
  - m (size of the search space)
  - d (degree)

  can be derived
- "Over-populating" (selecting $m \gg \#_{\text{evals}}$) may reduce the overall MVM count

$\rightsquigarrow$ A. Pieper et al., `arXiv:1510.04895`

Outline

Subspace iteration with Rayleigh–Ritz extraction

Improving the coefficients of the polynomials

Adaptive control of the degree

A priori information about the spectrum

High performance computational kernels

High performance computational kernels

- Several kernels occur in different eigensolvers:
  - Sparse matrix times (block) vector
  - Apply $p(\mathsf{A})$ to a (block) vector
  - Orthogonalize columns of a block vector
  - ...
- Provide optimized versions for these

GHOST and PHIST

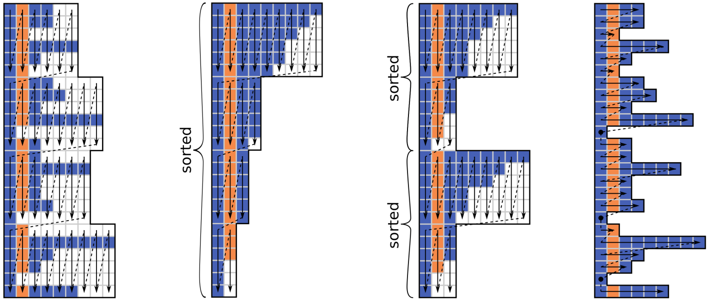
GHOST (General, Hybrid and Optimized Sparse Toolkit) provides

- shifted `sp(M)MVM`, augmented with dot products
- operations with dense block vectors (dense and scattered "views" to avoid copying)
- real and complex, single and double precision
- support for CPU, Phi, Nvidia (also in combination)
- possibility to specify "common" dimensions at compile time $\rightsquigarrow$ highly optimized kernels
- task management (e.g., for asynchronous checkpointing)

PHIST (Pipelined Hybrid-parallel Iterative Solver Toolkit) provides an abstraction layer and higher-level functionality (orthogonalization, etc.)

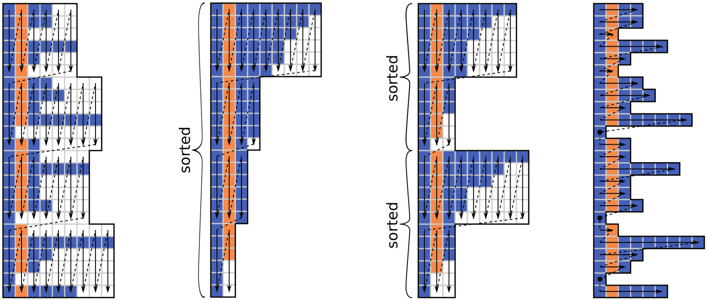
The SELL-C-σ format

Combine slicing and sorting:



SELL-6-1	SELL-6-24	SELL-6-12	SELL-1-1
aka SELL			aka CRS
$\beta = 0.51$	$\beta = 0.84$	$\beta = 0.66$	$\beta = 1.00$

Combine slicing and sorting:



SELL-6-1 aka SELL	SELL-6-24	SELL-6-12	SELL-1-1 aka CRS
$\beta = 0.51$	$\beta = 0.84$	$\beta = 0.66$	$\beta = 1.00$