ETH zürich

Master Thesis Presentation

Solving Large Scale Eigenvalue Problems in Amorphous Materials

Giuseppe Accaputo, 07.09.2017

Outline

- Introduction
- Goal
- Approaches
- Implementation and Parallelization
- Experimental Results
- Conclusion

Introduction: Amorphous Solids

Bulk metallic glasses (BMG) exhibit a variety of vibrational properties resulting from significant atomic scale disorder.

- Crystalline solids: vibrations are simply plane waves
- Amorphous solids: majority of vibrations are not plane waves
  - Propagative modes (e.g. plane waves) restricted to low frequency domain
  - Localized modes occupying high frequency tail
- Precise nature of these low frequency modes and how they are influenced by local atomic structure remains unclear

Introduction: Amorphous Solids

- Boson peak: existence of extra vibrational modes in amorphous solids
  - Considered to be one of the universal properties of glasses
  - Origin of the Boson peak is still debated
  - Boson peak is observed within a frequency interval
- Molecular dynamics simulations can produce atomistic BMG structures in which the equilibrium position of each atom is known
  - Generate and analyze Hessian matrix $\boldsymbol{H}$ containing second derivatives of the potential V at equilibrium geometry

Goal

- The following eigenproblem is given:

$$\boldsymbol{H}\boldsymbol{u}_j = \lambda_j \boldsymbol{u}_j \,, \quad j = 1, 2, \ldots, 3N \tag{1}$$

  - $\boldsymbol{u}_j$ are normal modes, i.e. displacements in case of Eq. (1)
  - $\lambda_j = \omega_j^2$, where ω_j are the fundamental frequencies

- Compute the eigenmodes in a low frequency region containing the Boson peak by diagonalizing $\boldsymbol{H}$
- **Challenge**: Region of interest is in the interior of the spectrum of $\boldsymbol{H}$

Previous Approach

- Sander Schaffner: "Using Trilinos to Solve Large Scale Eigenvalue Problems in Amorphous Materials" (2015)
- Advance into region of interest by computing a sufficiently large number of eigenvalues at the beginning of the spectrum
- **Problem**: for large matrices, too many eigenvalues and corresponding eigenvectors have to be calculated

New Approach: Interval-Specific Eigenvalue Computation

Goal: Efficient computation of eigenvalues in a specified interval of the spectrum of the matrix $\boldsymbol{H}$

- By specifying the interval of interest $[\xi, \eta] \subseteq [\lambda_{\min}, \lambda_{\max}]$ as the region containing the Boson peak, compute all the eigenvalues $\lambda \in [\xi, \eta]$ of $\boldsymbol{H}$.
  - $\lambda_{\min}, \lambda_{\max}$ are the extremal eigenvalues of $\boldsymbol{H}$
- Eigenvalues outside of the interval $[\xi, \eta]$ should be discarded to save computational time

Tools

- Polynomial filter, for filtering eigenvalues within a given interval of interest
- Iterative method for the computation of the eigenvalues of interest

Li, Ruipeng, et al. "A Thick-Restart Lanczos algorithm with polynomial filtering for Hermitian eigenvalue problems." *SIAM Journal on Scientific Computing* 38.4 (2016): A2512-A2534. [2]

Polynomial Filtering: Goal

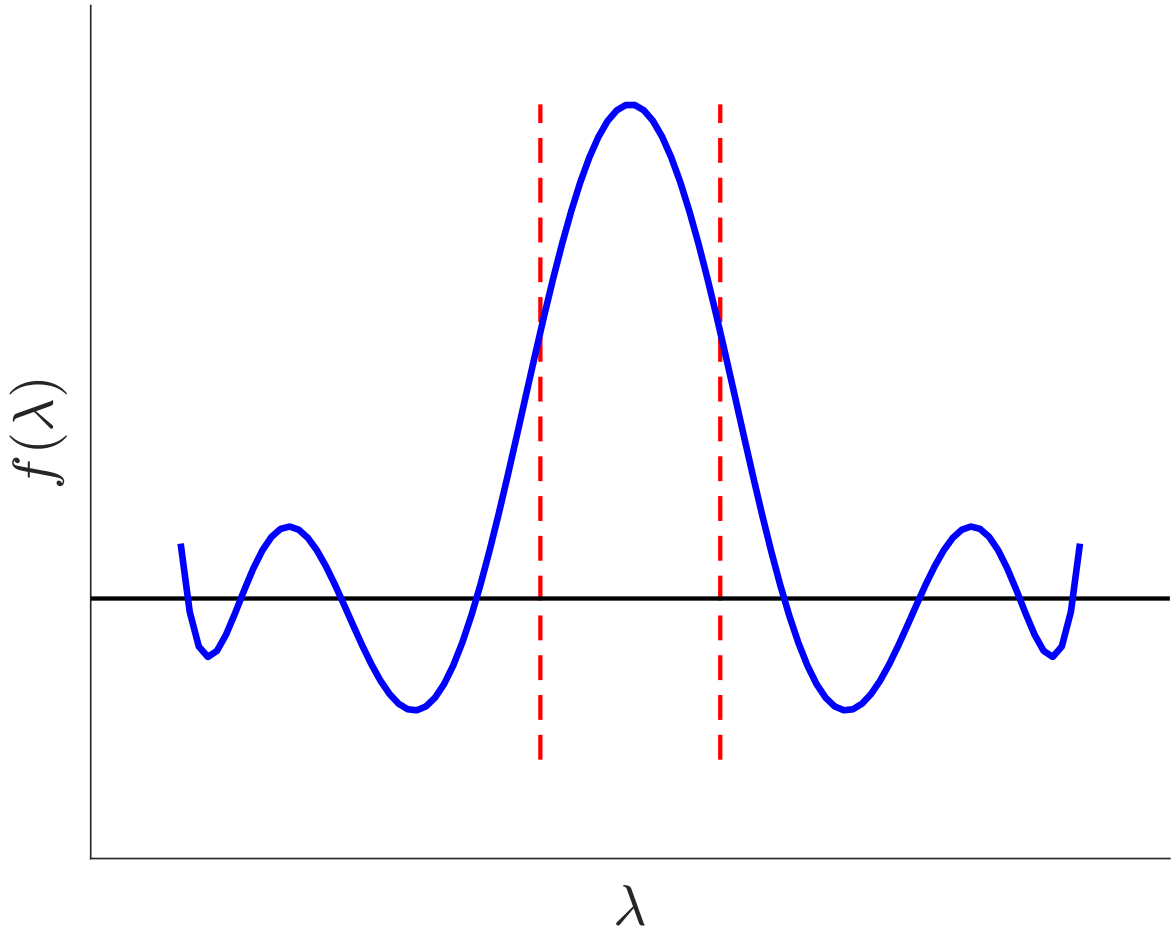


Figure 1: Example of a polynomial filter.

Polynomial Filtering: Least-Squares Polynomial Filter [2]

Approximate the Dirac delta function $\delta_\gamma = \delta(t-\gamma)$ centered at γ by

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

- $T_j(t)$ is the Chebyshev polynomial of the first kind of degree j

$$T_0(t) = 1 \tag{3}$$

$$T_1(t) = t \tag{4}$$

$$T_j(t) = 2\ t\ T_{j-1}(t) - T_{j-2}(t). \tag{5}$$

- $\hat{\rho}_k(t) = \rho_k(t)/\rho_k(\gamma)$ is an optimal filter [2]

Polynomial Filtering: Transformation

Chebyshev polynomials T_j are defined on the interval $[-1, 1]$. We want to apply the filter $\hat{\rho}_k$ on $\boldsymbol{H}$. Transform the eigenvalues of our matrix $\boldsymbol{H}$:

$$c = (\lambda_{\max} + \lambda_{\min})/2 \tag{6}$$

$$d = (\lambda_{\max} - \lambda_{\min})/2 \tag{7}$$

$$\hat{\boldsymbol{H}} = (\boldsymbol{H} - c\,\boldsymbol{I})/d \tag{8}$$

- All the eigenvalues $\hat{\lambda}$ of $\hat{\boldsymbol{H}}$ are in $[-1, 1]$; $\hat{\rho}_k(\hat{\boldsymbol{H}})$ can be evaluated
- Boundaries of interval of interest are also transformed:

$$\hat{\xi} = (\xi - c)/d \tag{9}$$

$$\hat{\eta} = (\eta - c)/d \tag{10}$$

Polynomial Filtering: Smoothing Approaches

- Expansions of discontinuous functions lead to oscillations near the discontinuities known as *Gibbs oscillations*
- Smoothing multipliers are added such that Eq. (2) actually is replaced by

$$\rho_k(t) = \sum_{j=0}^{k} g_j^k \mu_j T_j(t) \tag{11}$$

- Jackson and Lanczos smoothing available

Polynomial Filtering: Smoothing Approaches

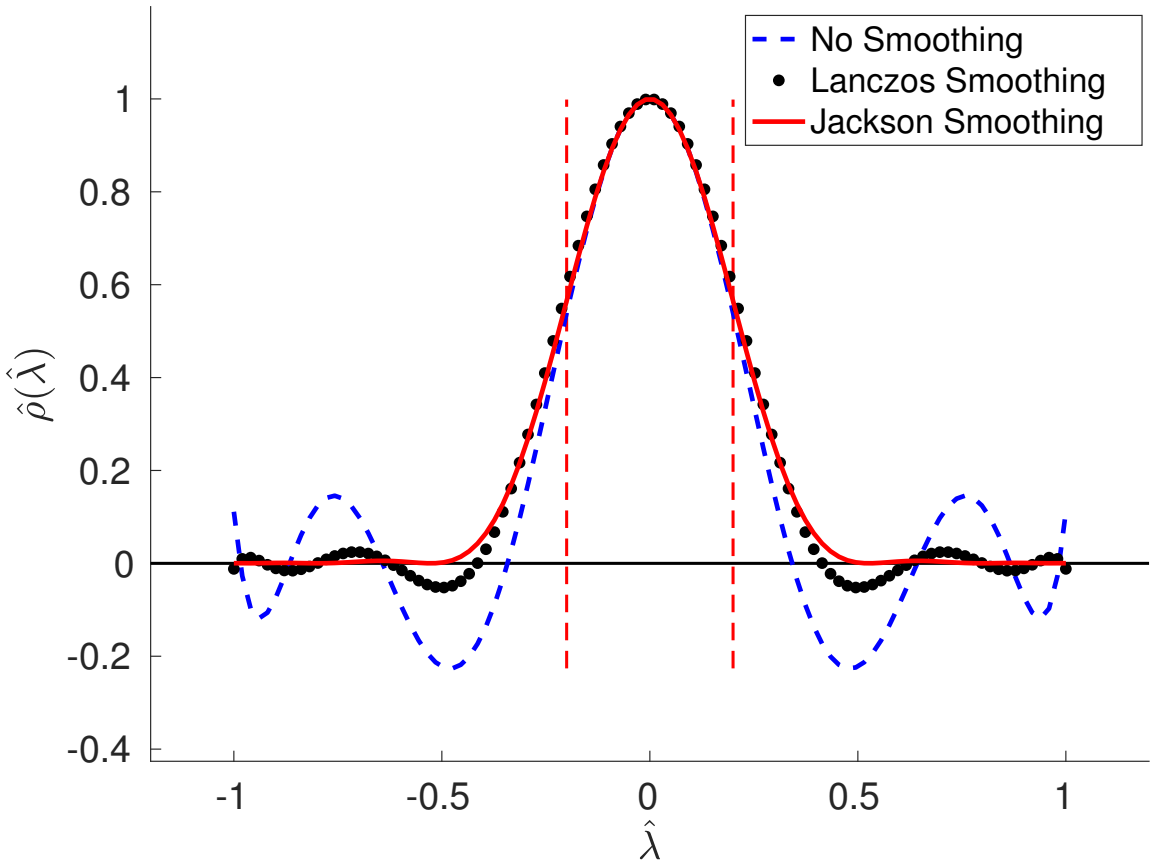


Figure 2: Filter polynomial $\hat{\rho}_k$ using different smoothing approaches.

Polynomial Filtering: Determining the Optimal Degree

1. Start from low degree (e.g. $k = 2$)
2. Increase degree until both boundary values become small enough, i.e.

$$\hat{\rho}_k(\hat{\xi}) < \phi \quad \text{and} \quad \hat{\rho}_k(\hat{\eta}) < \phi \quad , \tag{12}$$

   where ϕ is a specified threshold

- Most of the times: $\hat{\rho}_k(\hat{\xi}) \neq \hat{\rho}_k(\hat{\eta})$
  - Unfavorable for selection of eigenvalues

Polynomial Filtering: Determining the Optimal Degree

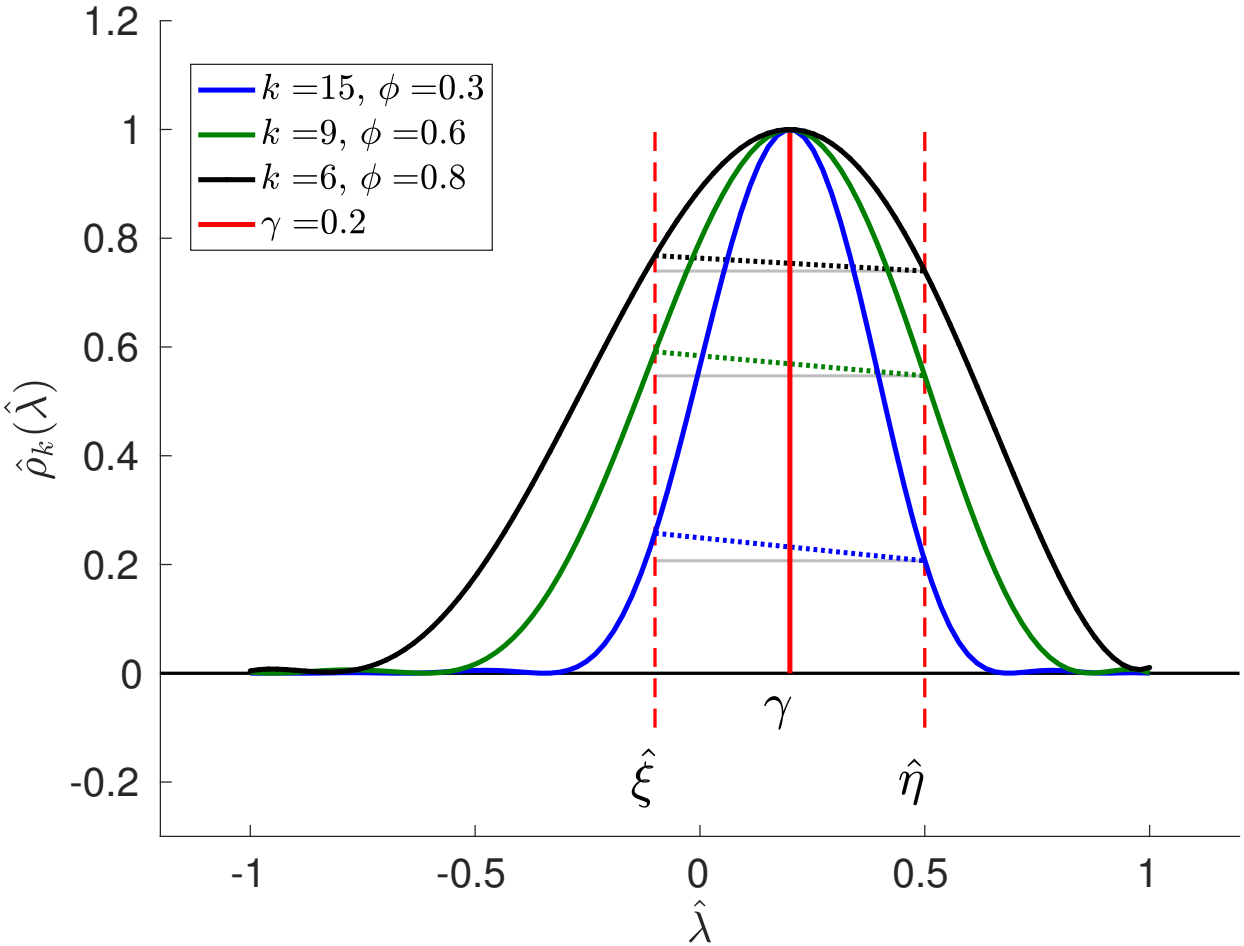


Figure 3: Jackson-smoothed filter polynomials $\hat{\rho}_k$.

Polynomial Filtering: Balancing the Filter

- To facilitate selection of eigenvalues it's preferable to have $\hat{\rho}_k(\hat{\xi}) = \hat{\rho}_k(\hat{\eta})$
- Move center γ such that $\hat{\rho}_k(\hat{\xi}) = \hat{\rho}_k(\hat{\eta})$
  and define a bar value τ with $\hat{\rho}_k(\hat{\xi}) = \hat{\rho}_k(\hat{\eta}) = \tau$
- Using τ determine if an eigenvalue λ is inside or outside of the interval of interest $[\xi, \eta]$:

$$\lambda \in [\xi, \eta] \iff \hat{\rho}_k(\hat{\lambda}) \geq \tau \quad \text{and} \quad \lambda \notin [\xi, \eta] \iff \hat{\rho}_k(\hat{\lambda}) < \tau \quad (13)$$

Polynomial Filtering: Balancing the Filter

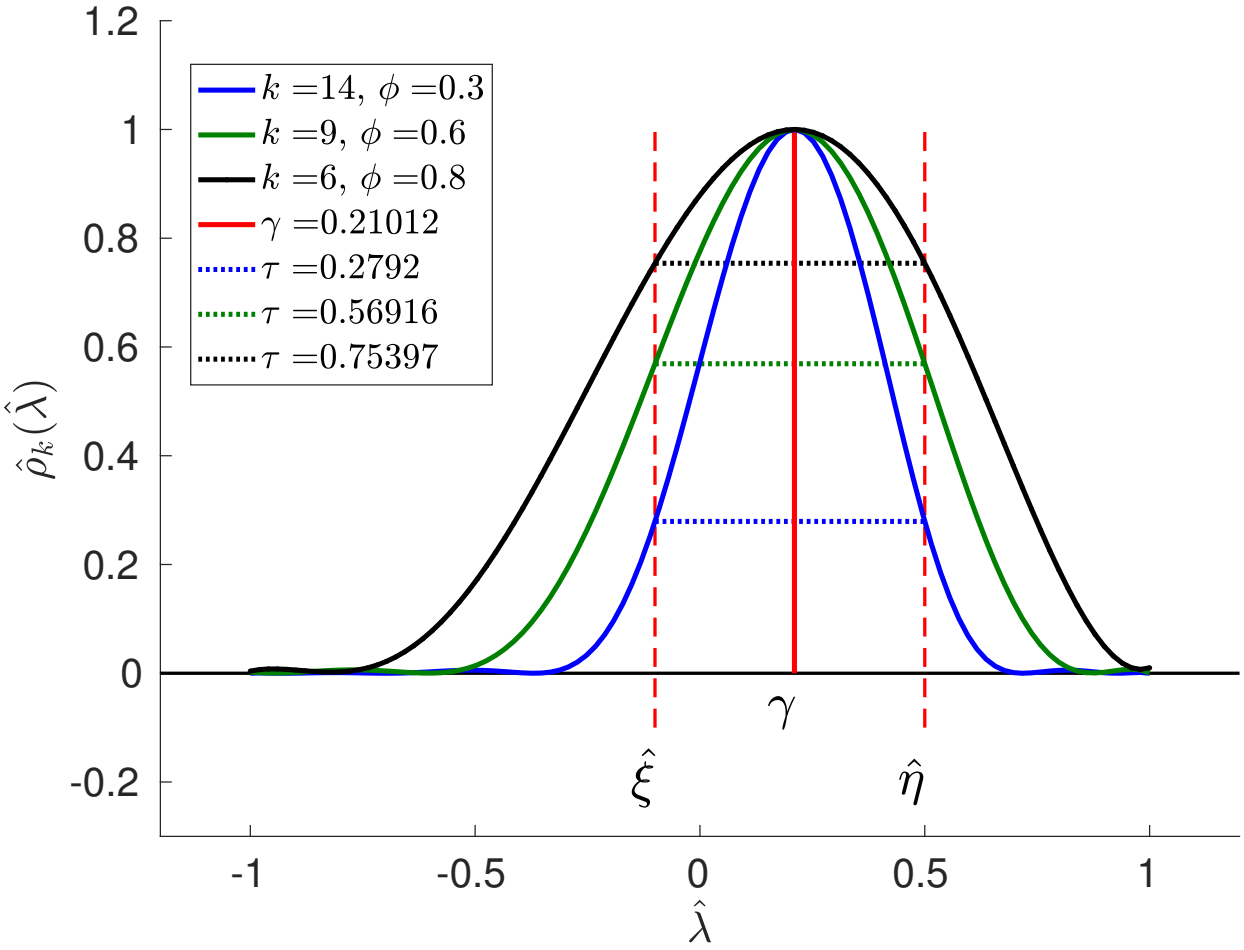


Figure 4: Balanced Jackson-smoothed filter polynomials $\hat{\rho}_k$.

Polynomial Filtering: Balancing the Filter

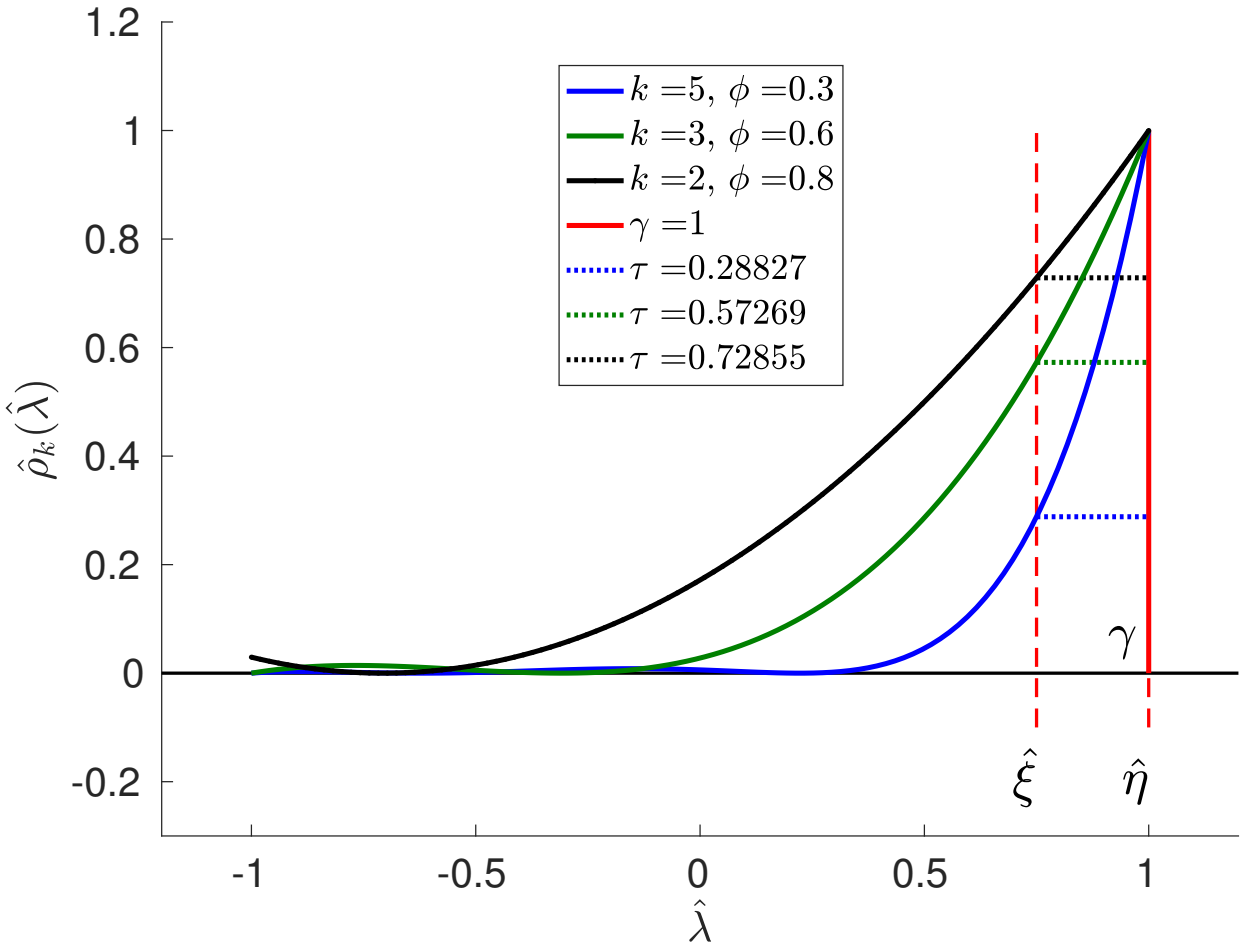


Figure 5: Jackson-smoothed Filter polynomial $\hat{\rho}_k$ for end intervals.

Polynomial Filtering: Visual Example

- $\lambda_{\min} = 0, \lambda_{\max} = 8$
- $\sigma = \{\lambda_{\min},\ 2,\ 2.3,\ 2.6,\ 3,\ 3.4,\ 6.1,\ 6.4,\ 7.1,\ 7.5,\ 7.7,\ \lambda_{\max}\}$
- Jackson smoothing is applied

Polynomial Filtering: Visual Example

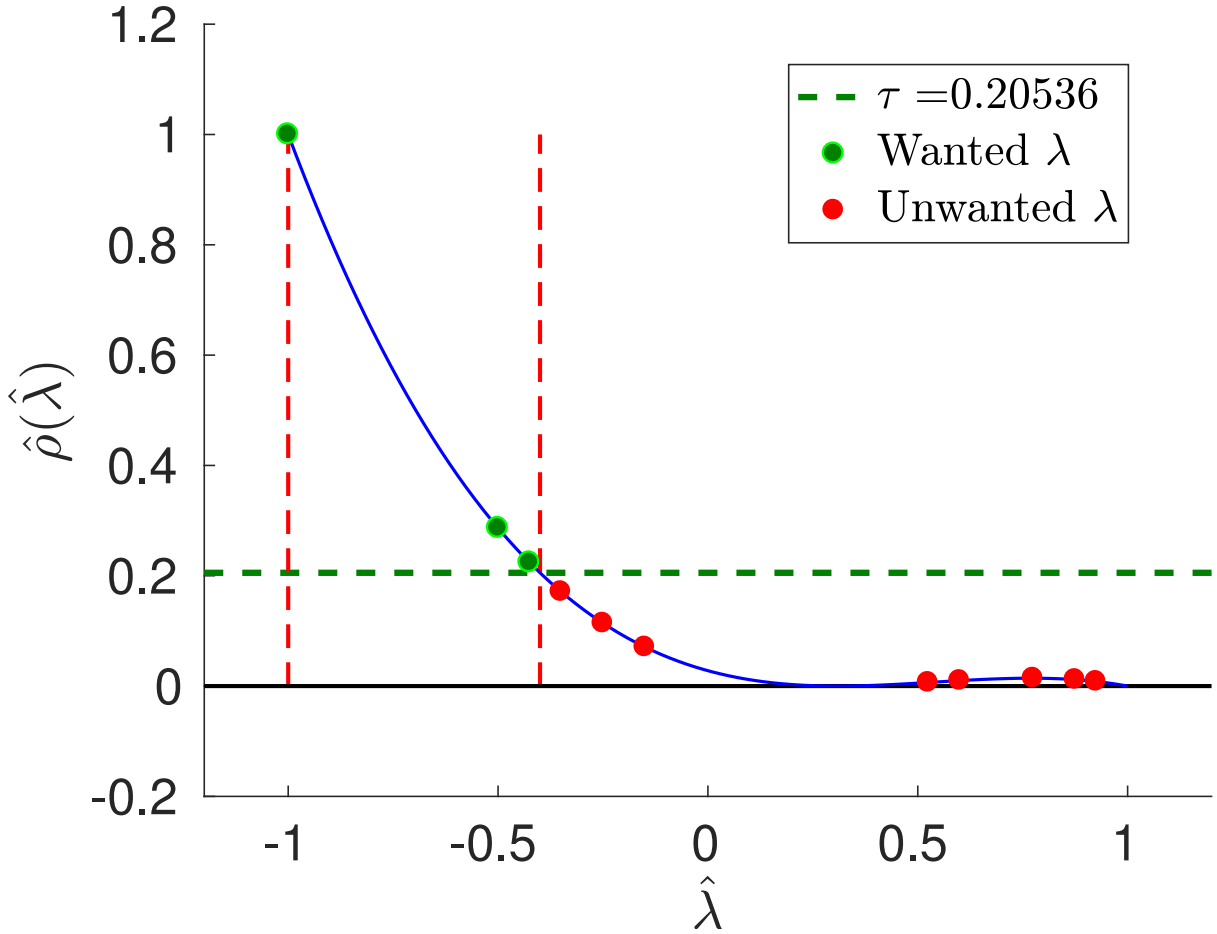


Figure 6: Filter polynomial $\hat{\rho}_k(\hat{\lambda})$ (k = 3) for the interval $[\lambda_{\min}, 2.4]$.

Polynomial Filtering: Visual Example

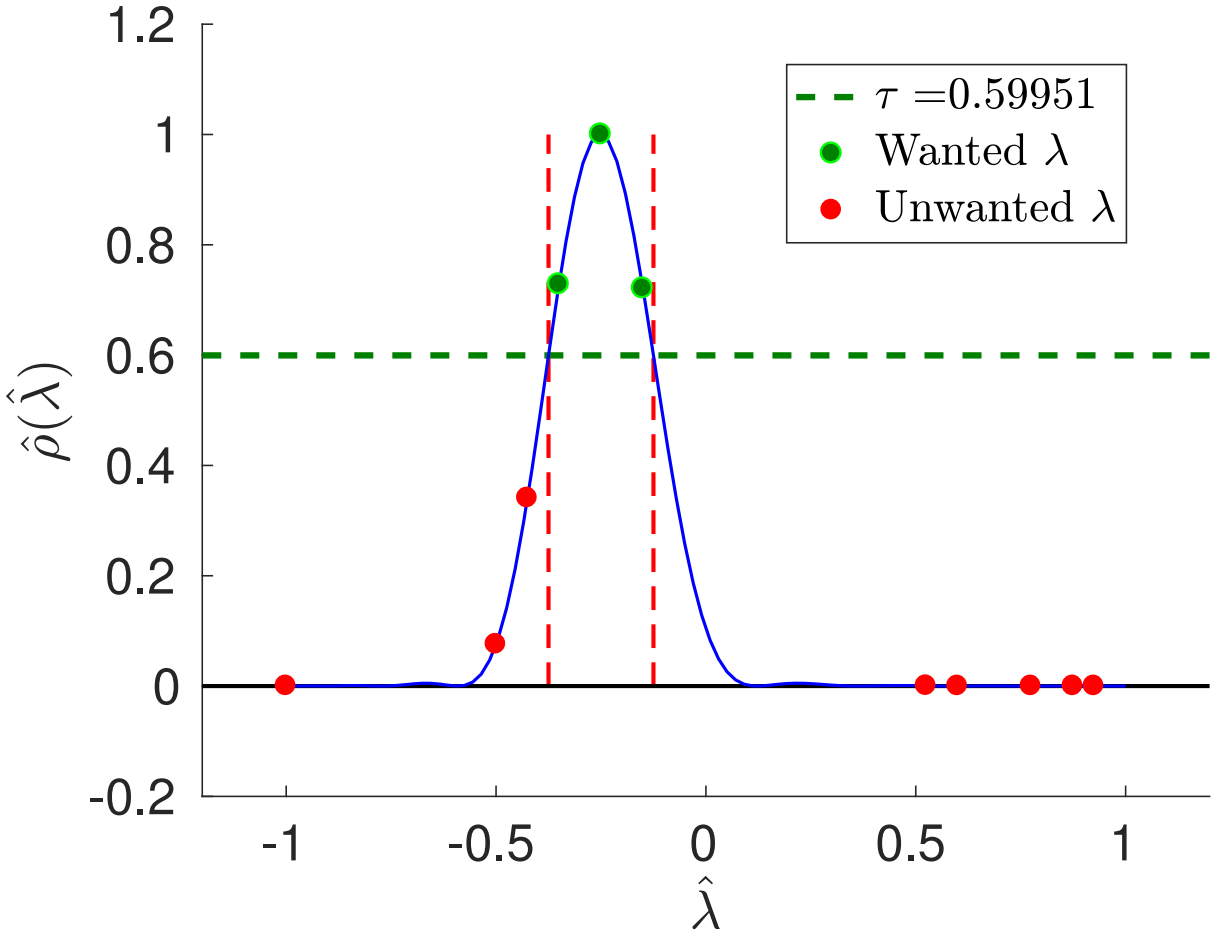


Figure 7: Filter polynomial $\hat{\rho}_k(\hat{\lambda})$ (k = 23) for the interval $[2.5, 3.5]$

Polynomial Filtering: Visual Example

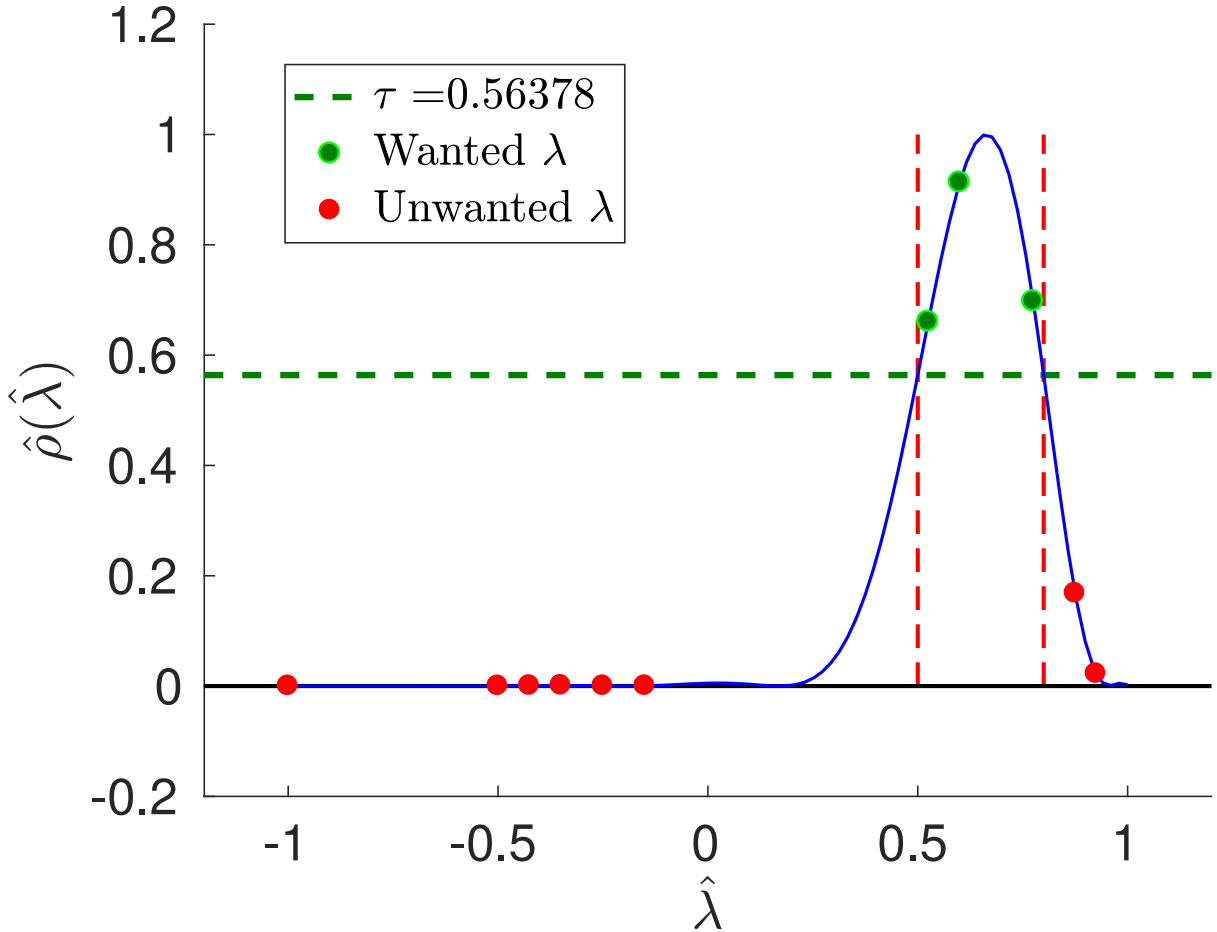


Figure 8: Filter polynomial $\hat{\rho}_k(\hat{\lambda})$ (k = 15) for the interval $[6, 7.2]$

Polynomial Filtering: Visual Example

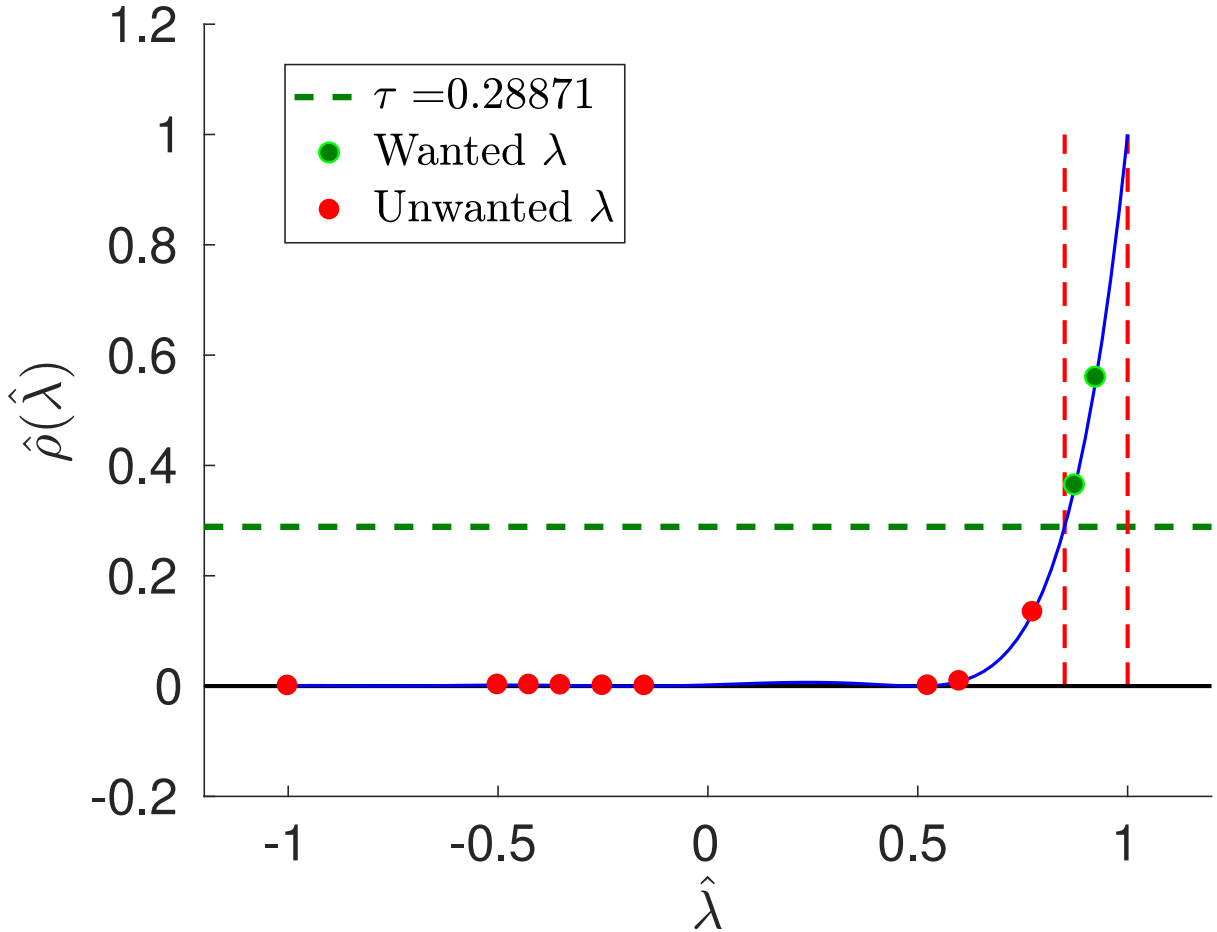


Figure 9: Filter polynomial $\hat{\rho}_k(\hat{\lambda})$ (k = 7) for the interval $[7.4, \lambda_{\max}]$

Eigenvalue Computation

Once we have defined the filter polynomial $\hat{\rho}_k$, we can apply it to our transformed matrix $\hat{\boldsymbol{H}}$ to get $\hat{\rho}_k(\hat{\boldsymbol{H}})$

- The eigenvalues of $\hat{\rho}_k(\hat{\boldsymbol{H}})$ are $\theta_1 = \hat{\rho}_k(\hat{\lambda}_1), \ldots, \theta_l = \hat{\rho}_k(\hat{\lambda}_l)$, with $\hat{\lambda}_i$ being the eigenvalues of $\hat{\boldsymbol{H}}$
- Apply the thick-restarted Lanczos algorithm to get the eigenpairs $(\theta_i, \boldsymbol{u}_i)$ for the few largest eigenvalues of $\hat{\rho}_k(\hat{\boldsymbol{H}})$
- If $\theta_i \geq \tau$, check if $\tilde{\lambda}_i = \boldsymbol{u}_i^T \boldsymbol{H} \boldsymbol{u}_i \in [\xi, \eta]$
  - $\theta_i \geq \tau$ is only used as a preselection tool
  - Eigenpairs $(\theta_i, \boldsymbol{u}_i)$ with $\theta_i < \tau$ are discarded

Eigenvalue Computation: Visual Example

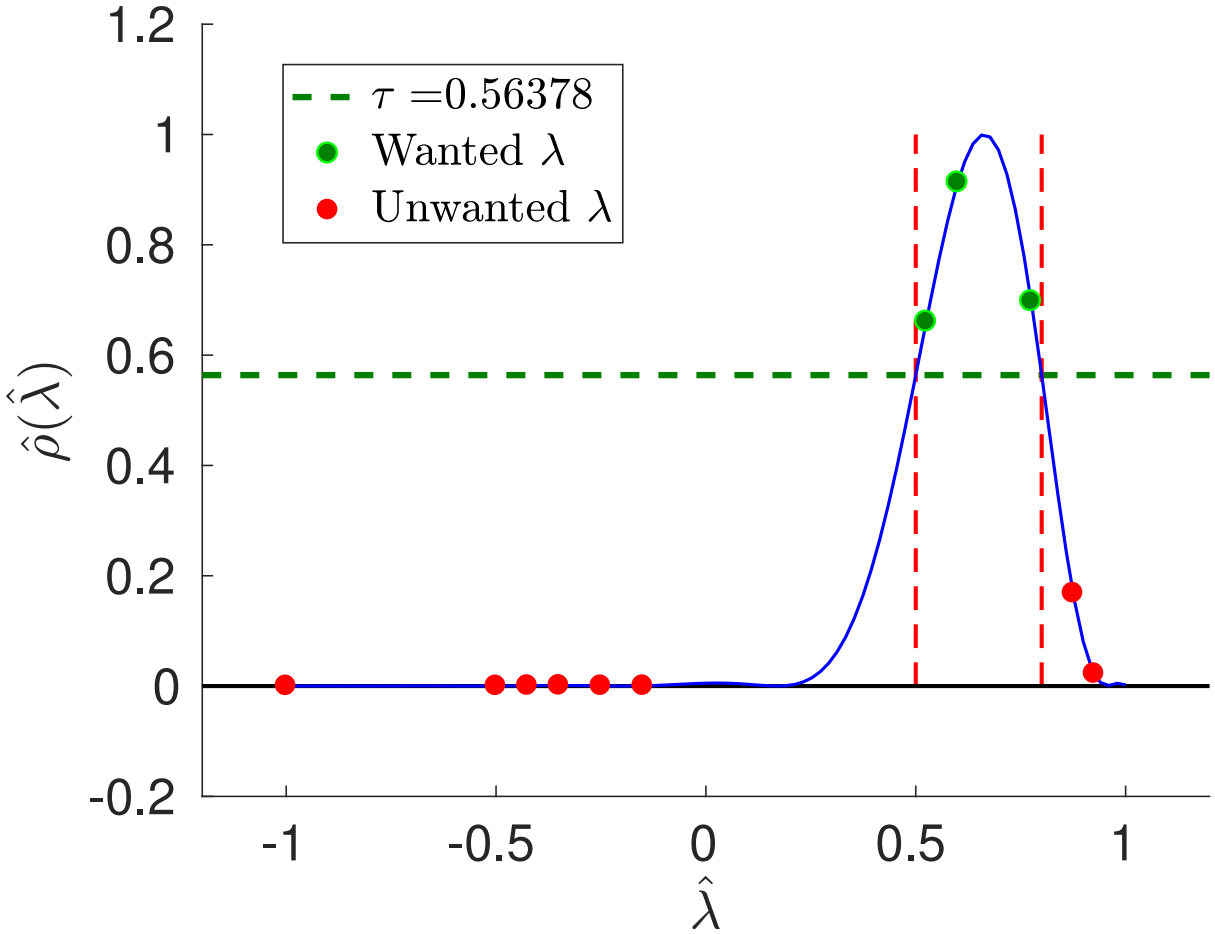


Figure 10: Filter polynomial $\hat{\rho}_k(\hat{\lambda})$ (k = 15) for the interval $[6, 7.2]$

Eigenvalue Count Estimation

- The size of the Krylov basis to be computed by the eigensolver depends on the number of eigenvalues we need
- The estimation of the number of eigenvalues $\mu_{[\xi,\eta]}$ in a given interval $[\xi, \eta]$ is based on an approximation of the trace of an eigenprojector [1]:

$$\mu_{[\xi,\eta]} = \mathrm{tr}(\boldsymbol{P}) \approx \frac{n}{n_v} \sum_{k=1}^{n_v} \left[\sum_{j=0}^{M} \nu_j \boldsymbol{v}_k^T \boldsymbol{T}_j(\boldsymbol{A}) \boldsymbol{v}_k \right] , \tag{14}$$

  - Works very well for well-separated eigenvalues in $[\xi, \eta]$
  - Runs into issues if spectrum contains clusters (also dependent on the separation of the eigenvalues)

Implementation and Parallelization

- Implementation based on C++11 and Trilinos
- Trilinos supports distributed-memory parallel computations through the Message Passing Interface (MPI)
- Row-wise distribution of the data: each MPI rank gets a contiguous and unique set of rows

Parallelization: Distribution Pattern

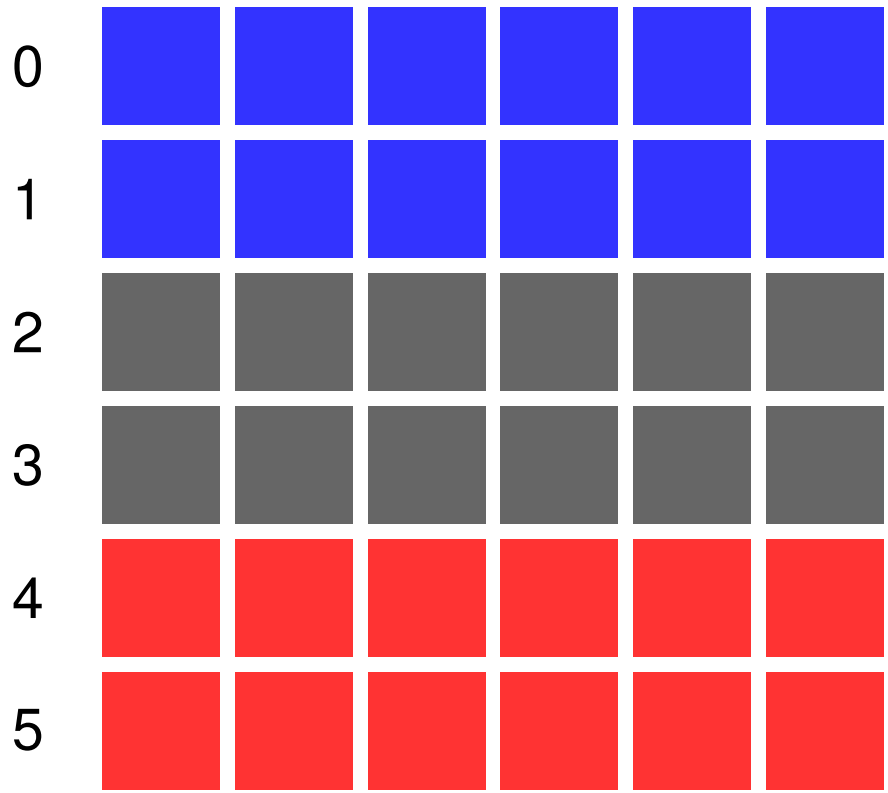


Figure 11: Distribution pattern of a matrix $\boldsymbol{H} \in R^{6\times 6}$ using 3 MPI ranks. Rank 1 gets rows $i = 0, 1$, rank 2 gets rows $i = 2, 3$, and rank 3 gets rows $i = 4, 5$

Polynomial Filter Operator $\hat{\rho}_k(\hat{\boldsymbol{H}})$

- During each Lanczos iteration the product $\hat{\rho}_k(\hat{\boldsymbol{H}})\boldsymbol{X}$ is computed, where $\boldsymbol{X} \in \mathbb{R}^{n \times b}$, with b being the block size:

$$\hat{\rho}_k(\hat{\boldsymbol{H}})\boldsymbol{X} = \sum_{j=0}^{k} \hat{\nu}_j^k \, T_j(\hat{\boldsymbol{H}}) \, \boldsymbol{X}, \quad \hat{\nu}_j^k = \mu_j \, g_j^k / \rho_k(\gamma) \tag{15}$$

- Rewrite Eq. (15) using $\boldsymbol{W}_j = T_j(\hat{\boldsymbol{H}}) \, \boldsymbol{X}$ with $\boldsymbol{W}_0 = \boldsymbol{X}$, $\boldsymbol{W}_1 = \hat{\boldsymbol{H}}\boldsymbol{X}$, and $\boldsymbol{W}_j = 2\,\hat{\boldsymbol{H}}\,\boldsymbol{W}_{j-1} - \boldsymbol{W}_{j-2}$ for $j > 1$

$$\hat{\rho}_k(\hat{\boldsymbol{H}})\boldsymbol{X} = \sum_{j=0}^{k} \hat{\nu}_j^k \boldsymbol{W}_j \tag{16}$$

Performance Measurements for $\hat{\rho}_k(\hat{\boldsymbol{H}}_{\boldsymbol{s}})\boldsymbol{X}$

- $\boldsymbol{H}_s \in \mathbb{R}^{96000\times 96000}, \quad nnz(\boldsymbol{H}_s) = 23985126$
- Distributed computation of $\hat{\rho}_k(\hat{\boldsymbol{H}}_{\boldsymbol{s}})\boldsymbol{X}$ with $\boldsymbol{X} \in \mathbb{R}^{n\times b}$
- Ra cluster @ PSI: 4 nodes, 2 Intel Ceon E5-2697Av4 (2.60 GHz) processors per node, 16 cores per processor. Total: 128 cores
- Measurement data: Mean time over 10 runs, with 2 warm-up computations before the actual measurements

Speedup and Parallel Efficiency

Speedup: $S_p = t_1/t_p$. Efficiency: $E_p = S_p/p$ (speedup per core)

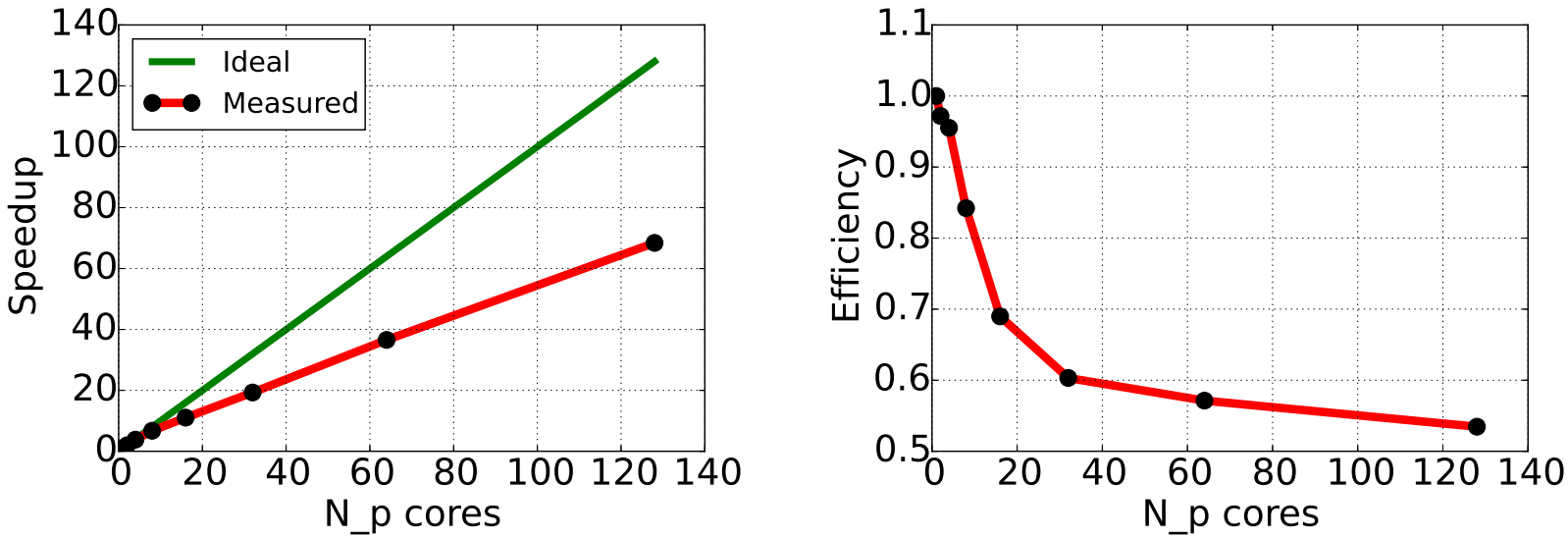


Figure 12: Speedup (left) and parallel efficiency (right) of the $\hat{\rho}_k(\hat{\boldsymbol{H}}_s)\boldsymbol{X}$ product with $k = 500$ and $b = 32$

Speedup and Parallel Efficiency

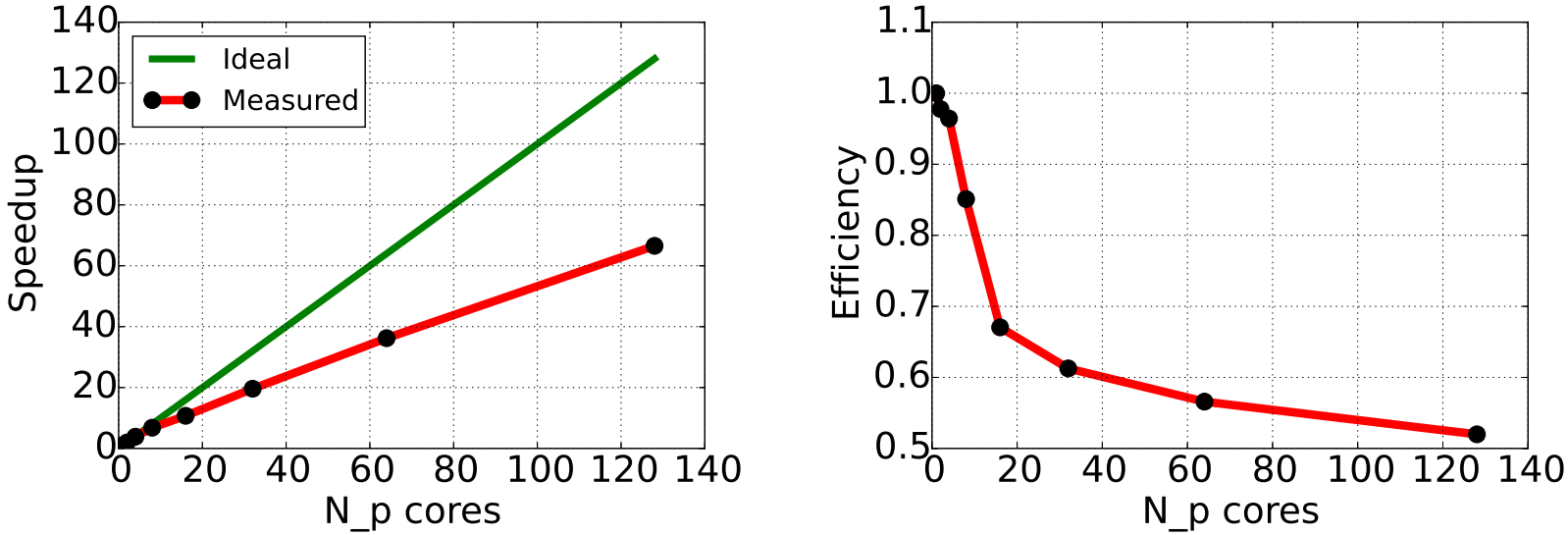
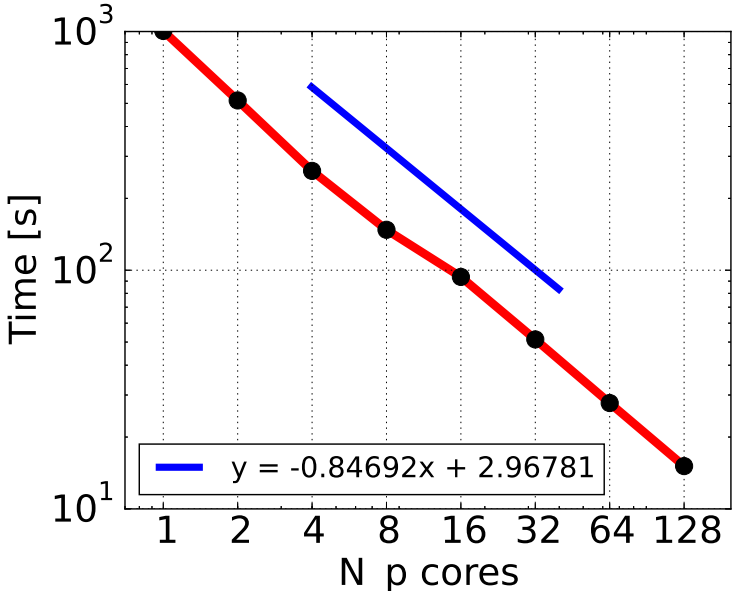


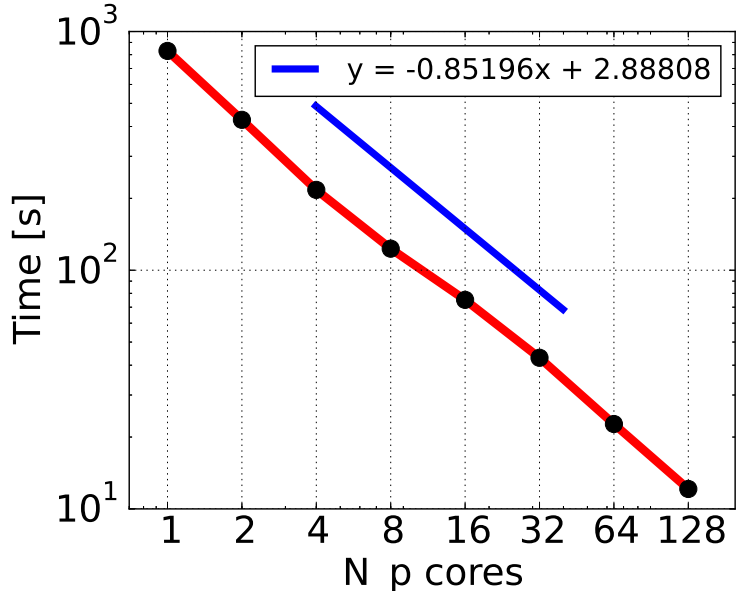
Figure 13: Speedup (left) and parallel efficiency (right) of the $\hat{\rho}_k(\hat{\boldsymbol{H}}_{\boldsymbol{s}})\boldsymbol{X}$ product with $k = 151$ and $b = 128$

Strong Scaling of $\hat{\rho}_k(\hat{\boldsymbol{H}}_{\boldsymbol{s}})\boldsymbol{X}$

Fix degree k or block size b and increase number of cores.



(a) $b = 32$ and $k = 500$.

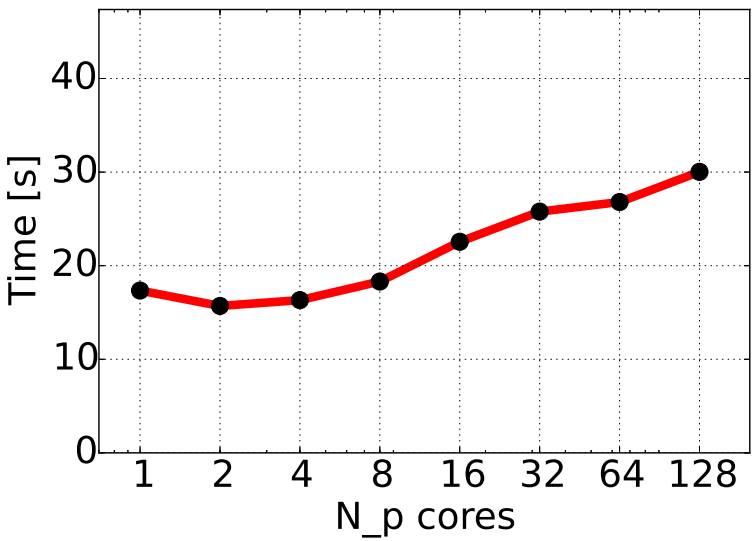


(b) $b = 128$ and $k = 151$.

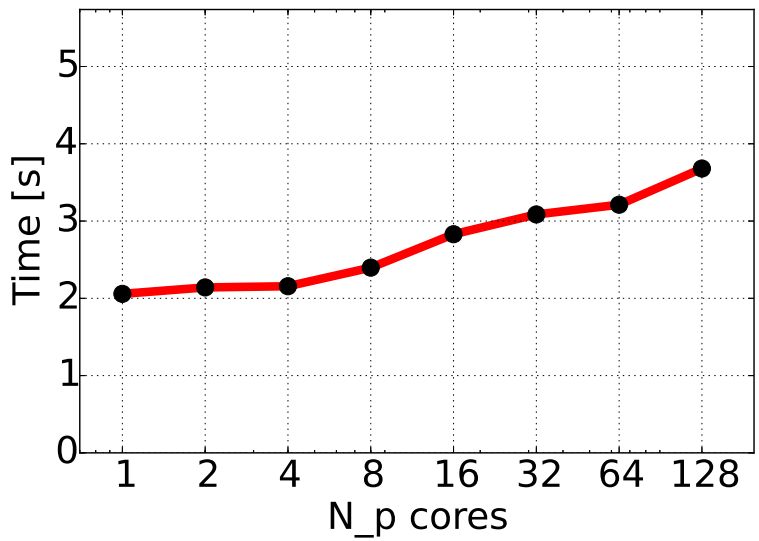
Figure 14: Strong scaling measurements of the $\hat{\rho}_k(\hat{\boldsymbol{H}}_{\boldsymbol{s}})\boldsymbol{X}$ product

Weak Scaling of $\hat{\rho}_k(\hat{\boldsymbol{H}}_{\boldsymbol{s}})\boldsymbol{X}$

Constant workload per core, add more cores to solve larger total problem.



(a) $b = 8$ per core and fixed $k = 151$.



(b) $k = 50$ per core and fixed $b = 32$.

Figure 15: Weak scaling measurements of the $\hat{\rho}_k(\hat{\boldsymbol{H}}_{\boldsymbol{s}})\boldsymbol{X}$ product

Diagonalization Results

- Diagonalized 8 Hessian matrices
  $\boldsymbol{H}_1, \ldots, \boldsymbol{H}_8 \in \mathbb{R}^{768000 \times 768000}$
- Residual vector: $\boldsymbol{r}_i = \boldsymbol{H}_k \boldsymbol{u}_i - \lambda_i \boldsymbol{u}_i, \quad k = 1, \ldots, 8$

Matrix	nnz	Num. $\lambda \in [0.1, 2]$	Max. $\|r_i\|$
$\boldsymbol{H}_1$	191893806	1068	1.1303×10^{-9}
$\boldsymbol{H}_2$	191883888	1030	7.2249×10^{-9}
$\boldsymbol{H}_3$	191903166	1050	3.5170×10^{-8}
$\boldsymbol{H}_4$	191851848	1075	7.0602×10^{-9}
$\boldsymbol{H}_5$	191832588	1077	9.1160×10^{-9}
$\boldsymbol{H}_6$	191859012	1079	4.0634×10^{-9}
$\boldsymbol{H}_7$	191887542	1058	1.9001×10^{-9}
$\boldsymbol{H}_8$	191853378	1051	8.0480×10^{-9}

Eigenmode Analysis: Participation Ratio

- Since sound has long wavelength, many particles should be affected by vibrational eigenmode
- Amount of particles moving together in the vibrational eigenmodes is usually quantified by the participation ratio (PR) defined for each eigenmode j as

$$PR(\omega_j) = \frac{1}{N} \frac{(\sum_i u_i^2(\omega_j))^2}{\sum_i u_i^4(\omega_j)} \quad , \tag{17}$$

  where $\boldsymbol{u}_i$ is the displacement of the ith atom

  - Isolated particle: $PR = 1/N$
  - Translational motion: $PR = 1$ (all particles are involved)

Eigenmode Analysis: Participation Ratio

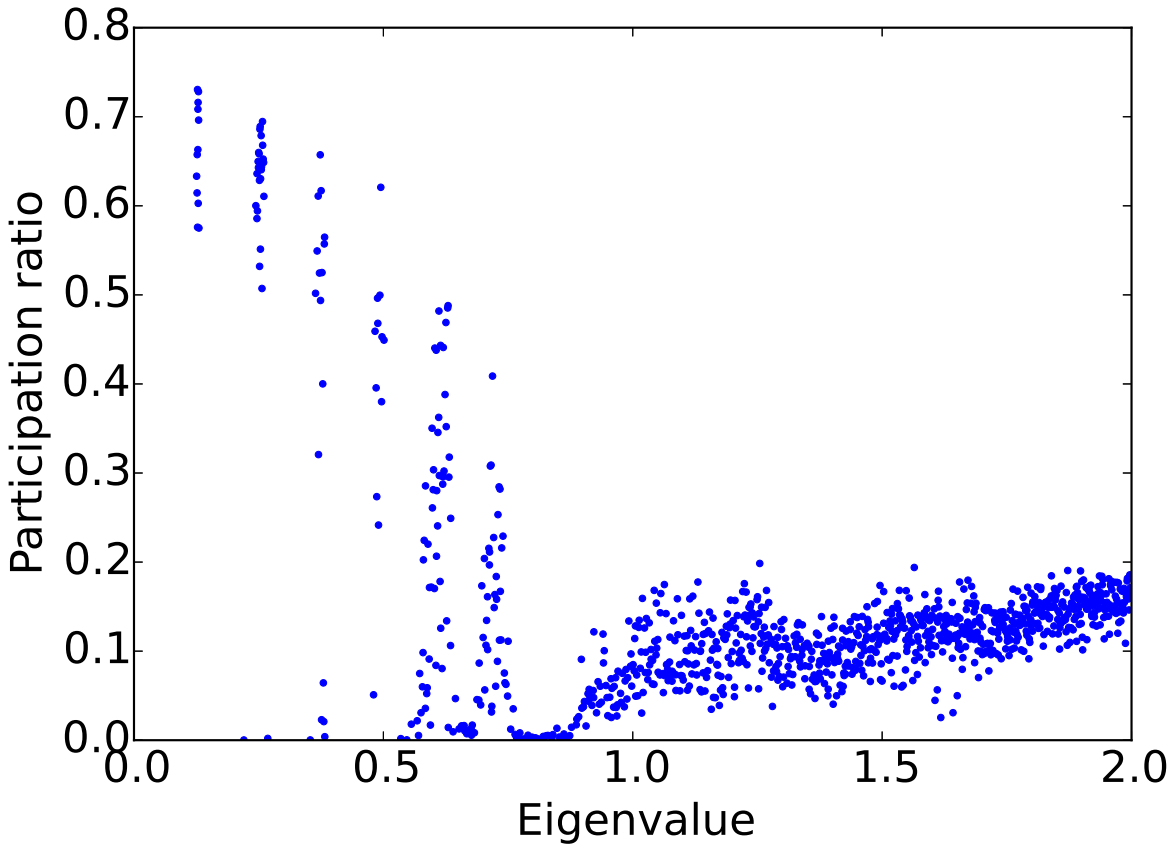


Figure 16: Participation ratio for $\boldsymbol{H}_1$

Eigenmode Analysis: Participation Ratio

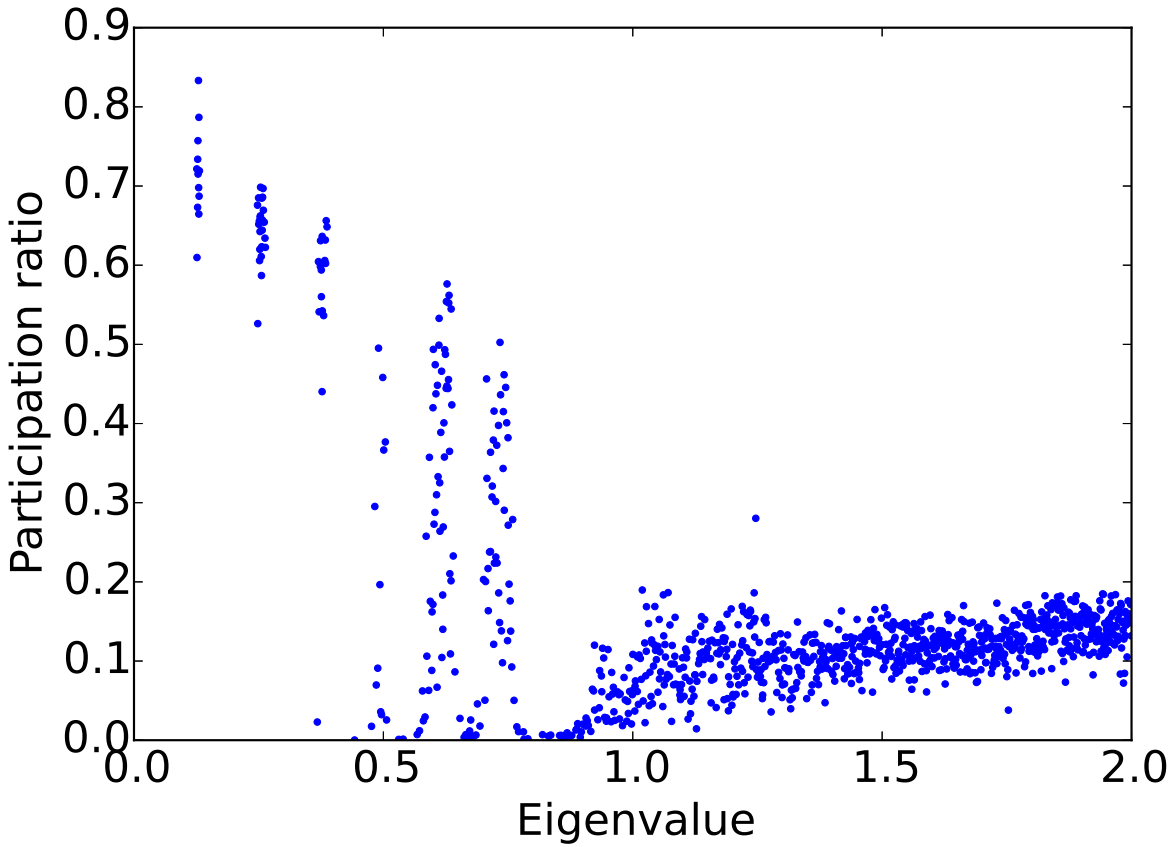


Figure 17: Participation ratio for $\boldsymbol{H}_2$

Eigenmode Analysis: Participation Ratio

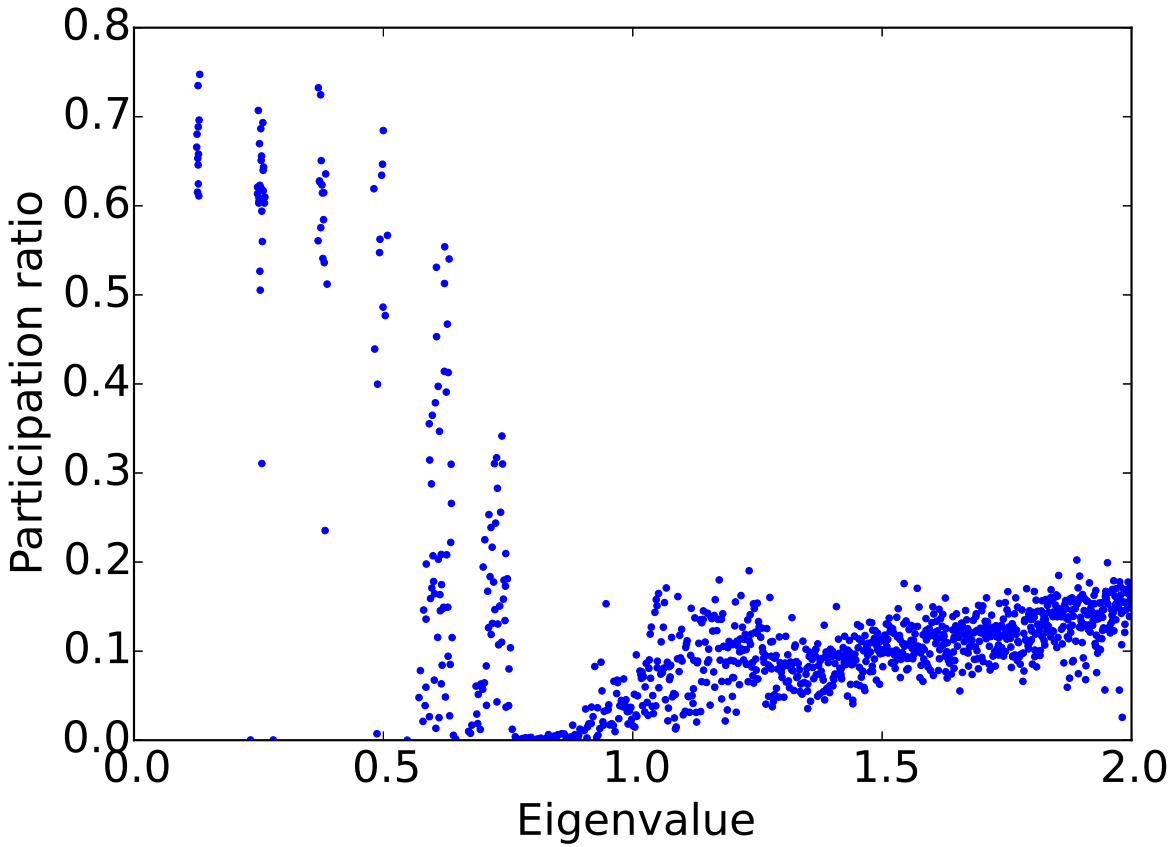


Figure 18: Participation ratio for H_3

Eigenmode Analysis: Displacement Fields

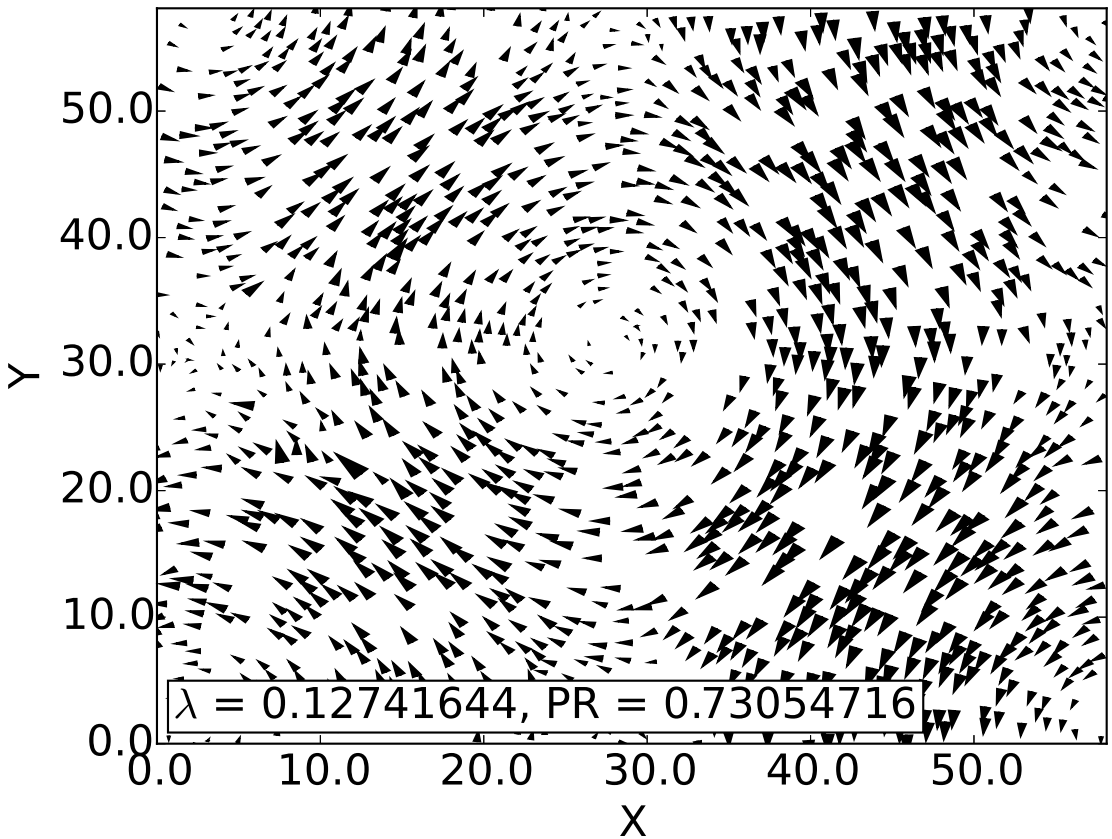


Figure 19: Displacements in $x - y$ plane ($\delta z = 0.15\text{Å}$) for eigenmode of $\boldsymbol{H}_1$. Arrow size $\propto$ displacement of particles ($\times$ 300)

Eigenmode Analysis: Displacement Fields

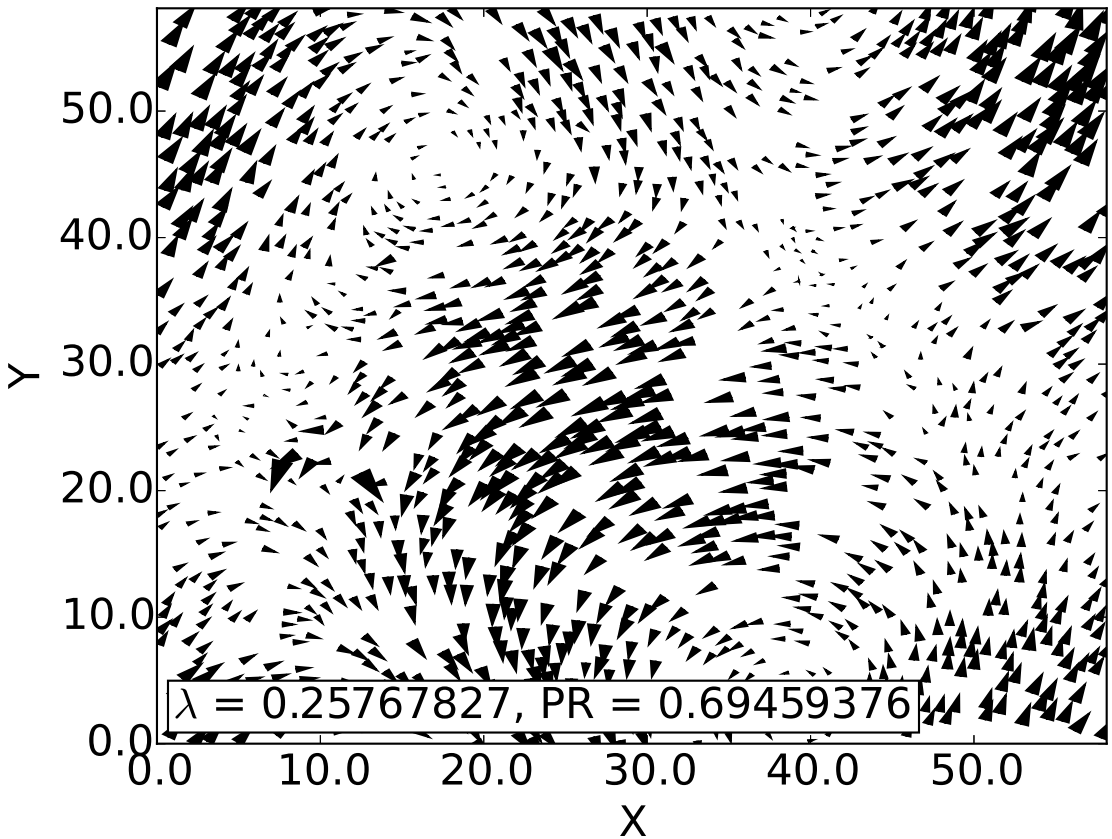


Figure 20: Displacements in $x - y$ plane ($\delta z = 0.15\text{Å}$) for eigenmode of $\boldsymbol{H}_1$. Arrow size $\propto$ displacement of particles ($\times$ 300)

Eigenmode Analysis: Displacement Fields

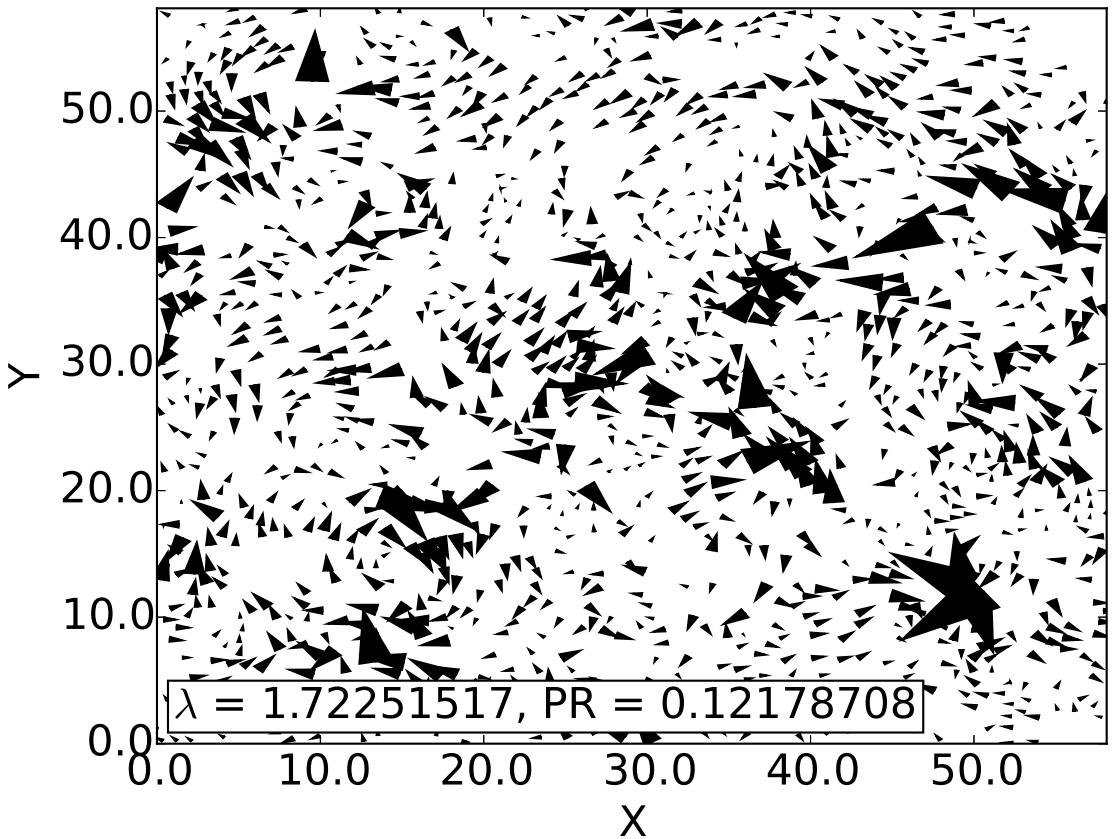


Figure 21: Displacements in $x - y$ plane ($\delta z = 0.15\text{Å}$) for eigenmode of $\boldsymbol{H}_1$. Arrow size $\propto$ displacement of particles ($\times$ 300)

Eigenmode Analysis: Displacement Fields

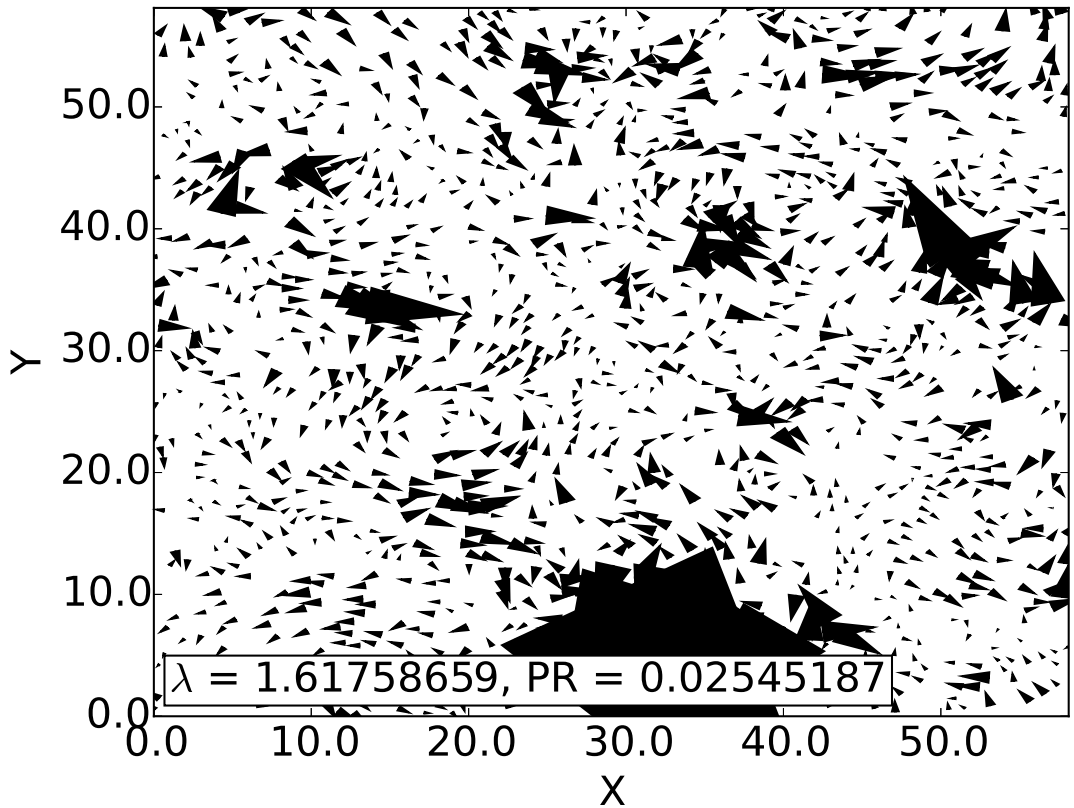


Figure 22: Displacements in $x - y$ plane ($\delta z = 0.15\text{Å}$) for eigenmode of $\boldsymbol{H}_1$. Arrow size $\propto$ displacement of particles ($\times$ 300)

Results: Conclusion

- Results for given samples look promising
  - Larger samples for further analysis
- Scaling results look good, too

Project: Conclusion

- Implementing numerical algorithms is fun (and hard)
  - Seeing results is even better
- Unit testing and debugging MPI code can be challenging
- Running code on different clusters can eventually be full of surprises

Future Work

- Run larger samples
- Extend testing framework
- Try a 2D parallel distribution pattern
- Implementation of slicing using different MPI world settings
- Implementation of different approaches for the approximation of the extremal eigenvalues
- Implementation of different approaches for the estimation of the eigenvalue count

Acknowledgements

- Peter Arbenz
- Peter Derlet
- The group

References

[1] Edoardo Di Napoli, Eric Polizzi, and Yousef Saad. Efficient estimation of eigenvalue counts in an interval. *Numerical Linear Algebra with Applications*, 23(4):674–692, 2016.

[2] Ruipeng Li, Yuanzhe Xi, Eugene Vecharynski, Chao Yang, and Yousef Saad. A thick-restart lanczos algorithm with polynomial filtering for hermitian eigenvalue problems. *SIAM Journal on Scientific Computing*, 38(4):A2512–A2534, 2016.