International Symposium on
Numerics and Scientific Computing
Schloß Bruchsal, 22–24 October 2013

Computing Eigenvalues of $\mathcal{H}$(ackbusch) Matrices

Peter Benner Thomas Mach

Max Planck Institute for Dynamics of Complex Technical Systems
Computational Methods in Systems and Control Theory

KU Leuven
Department of Computer Science

Max Planck Mathematicians...

61st Annual Meeting of the Max Planck Society, Hannover, 2010

Courtesy of Joachim Heinze.

Eigenvalue Problem

Definition

The pair $(\lambda, v) \in \mathbb{R} \times \mathbb{R}^n$ is called an *eigenpair* of the symmetric matrix $M = M^T \in \mathbb{R}^{n \times n}$, if

$$Mv = v\lambda.$$

The set $\Lambda(M) = \{\lambda | \exists v : (\lambda, v) \text{ eigenpair of } M\}$ is the spectrum of M.

Similarity Transformation

$$\Lambda(M) = \Lambda(P^{-1}MP) \qquad \forall P \text{ invertible}$$

Classification of Eigenvalue Problems

[Golub, Van der Vorst '00]

- Is M real or complex?

Classification of Eigenvalue Problems

[Golub, Van der Vorst '00]

- Is M real or complex?
  $M \in \mathbb{R}^{n \times n}$

Classification of Eigenvalue Problems

[Golub, Van der Vorst '00]

- Is M real or complex?
  $M \in \mathbb{R}^{n \times n}$

- Special properties (symmetric, Hermitian, skew-symmetric or unitary)?

Classification of Eigenvalue Problems

[Golub, Van der Vorst '00]

- Is M real or complex?
  $M \in \mathbb{R}^{n \times n}$
- Special properties (symmetric, Hermitian, skew-symmetric or unitary)?
  symmetric: $M = M^T$

Classification of Eigenvalue Problems

[Golub, Van der Vorst '00]

- Is M real or complex?
  $M \in \mathbb{R}^{n \times n}$

- Special properties (symmetric, Hermitian, skew-symmetric or unitary)?
  symmetric: $M = M^T$

- Further structure?

Classification of Eigenvalue Problems

[Golub, Van der Vorst '00]

- Is M real or complex?
  $M \in \mathbb{R}^{n\times n}$
- Special properties (symmetric, Hermitian, skew-symmetric or unitary)?
  symmetric: $M = M^T$
- Further structure? Yep.

Classification of Eigenvalue Problems

[Golub, Van der Vorst '00]

- Is M real or complex?
  $M \in \mathbb{R}^{n \times n}$
- Special properties (symmetric, Hermitian, skew-symmetric or unitary)?
  symmetric: $M = M^T$
- Further structure? Yep.
  $M \in \mathcal{H}(T_{\mathcal{I} \times \mathcal{I}}, k) \Rightarrow$ see next slide

Classification of Eigenvalue Problems

[Golub, Van der Vorst '00]

- Is M real or complex?
  $M \in \mathbb{R}^{n \times n}$
- Special properties (symmetric, Hermitian, skew-symmetric or unitary)?
  symmetric: $M = M^T$
- Further structure? Yep.
  $M \in \mathcal{H}(T_{\mathcal{I} \times \mathcal{I}}, k) \Rightarrow$ see next slide
- Which eigenvalues required?

Classification of Eigenvalue Problems

[Golub, Van der Vorst '00]

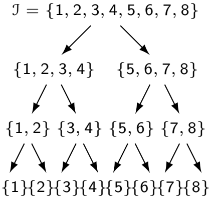
- Is M real or complex?
  $M \in \mathbb{R}^{n \times n}$
- Special properties (symmetric, Hermitian, skew-symmetric or unitary)?
  symmetric: $M = M^T$
- Further structure? Yep.
  $M \in \mathcal{H}(T_{\mathcal{I} \times \mathcal{I}}, k) \Rightarrow$ see next slide
- Which eigenvalues required?
  some (inner) or all eigenvalues

$\mathcal{H}$-Matrices

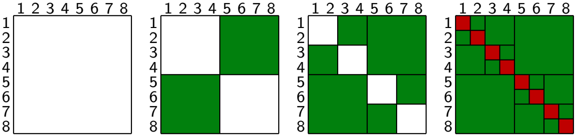
[Hackbusch 1998]

Some dense matrices, e.g. BEM or FEM, can be approximated by $\mathcal{H}$-matrices in a data-sparse manner.

hierarchical tree $T_{\mathcal{I}}$



block $\mathcal{H}$-tree $T_{\mathcal{I}\times\mathcal{I}}$



dense matrices, **rank-k-matrices**

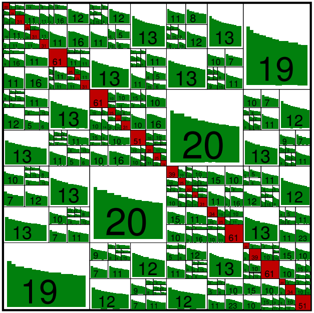
rank-k-matrix: $M_{a\times b} = AB^T$, $A \in \mathbb{R}^{n\times k}, B \in \mathbb{R}^{m\times k}$ $(k \ll n, m)$,

$\mathcal{H}$-Matrices

[Hackbusch 1998]

Hierarchical Matrices

$$\mathcal{H}(T_{\mathcal{I}\times\mathcal{I}}, k) = \left\{ M \in \mathbb{R}^{\mathcal{I}\times\mathcal{I}} \middle| \operatorname{rank}(M_{a\times b}) \leq k \ \forall a \times b \text{ admissible} \right\}$$



- adaptive rank $k(\varepsilon)$
- storage $N_{St,\mathcal{H}}(T, k) = \mathcal{O}(n \log n \; k(\varepsilon))$
- complexity of approximate arithmetic

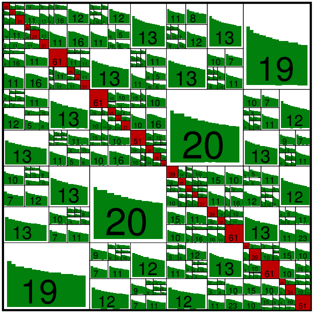
$M_{\mathcal{H}}v$	$\mathcal{O}(n \log n \; k(\varepsilon))$
$+_{\mathcal{H}}, -_{\mathcal{H}}$	$\mathcal{O}(n \log n \; k(\varepsilon)^2)$
$*_{\mathcal{H}}, \mathcal{H}LU(\cdot), (\cdot)^{-1}_{\mathcal{H}}$	$\mathcal{O}(n (\log n)^2 \; k(\varepsilon)^2)$

$\mathcal{H}$-Matrices

[Hackbusch 1998]

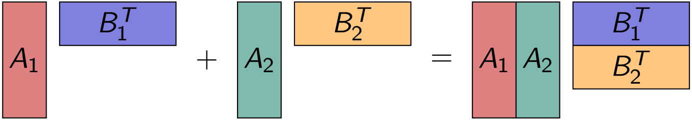
Hierarchical Matrices

$$\mathcal{H}(T_{\mathcal{I}\times\mathcal{I}}, k) = \left\{ M \in \mathbb{R}^{\mathcal{I}\times\mathcal{I}} \middle| \operatorname{rank}(M_{a\times b}) \leq k \;\; \forall a \times b \text{ admissible} \right\}$$



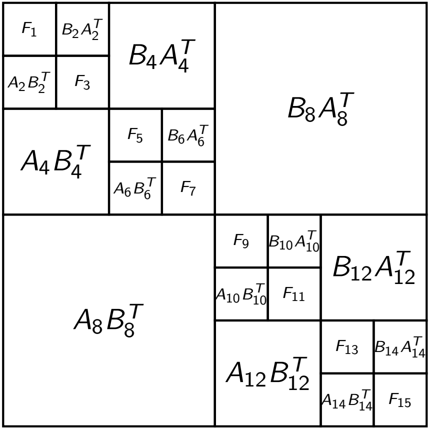
- adaptive rank $k(\varepsilon)$
- storage $N_{St,\mathcal{H}}(T, k) = \mathcal{O}(n \log n \; k(\varepsilon))$
- complexity of approximate arithmetic

$M_{\mathcal{H}} v$	$\mathcal{O}(n \log n \; k(\varepsilon))$
$+_{\mathcal{H}}, -_{\mathcal{H}}$	$\mathcal{O}(n \log n \; k(\varepsilon)^2)$
$*_{\mathcal{H}}, \mathcal{H}LU(\cdot), (\cdot)_{\mathcal{H}}^{-1}$	$\mathcal{O}(n (\log n)^2 \; k(\varepsilon)^2)$



Special Case: $\mathcal{H}_\ell$-Matrices

[Hackbusch 1998]



Structure of a symmetric $\mathcal{H}_3(k)$-matrix.

$\mathcal{H}$lib

$\mathcal{H}$lib [Börm, Grasedyck, et al.]

We use the $\mathcal{H}$lib (`www.hlib.org`) for the $\mathcal{H}$-arithmetic operations and some examples out of the library for testing the eigenvalue algorithm.

Eigenvalues of Symmetric $\mathcal{H}$-Matrices

$$M = M^T \in \mathcal{H}(T, k)$$
$$\Downarrow$$

$$\Lambda_{\mathcal{H}}(M) = \{\lambda_1, \ldots, \lambda_n\} \text{ in } \mathcal{O}(n^2 (\log n)^\alpha k^\beta)$$

$$\{\lambda_i\} \in \Lambda_{\mathcal{H}}(M) \text{ in } \mathcal{O}(n (\log n)^\alpha k^\beta)?$$

Eigenvalues of Symmetric $\mathcal{H}$-Matrices

$$M = M^T \in \mathcal{H}(T, k)$$
$$\Downarrow$$

$$\Lambda_{\mathcal{H}}(M) = \{\lambda_1, \ldots, \lambda_n\} \text{ in } \mathcal{O}(n^2 (\log n)^{\alpha} k^{\beta})$$

$$\{\lambda_i\} \in \Lambda_{\mathcal{H}}(M) \text{ in } \mathcal{O}(n (\log n)^{\alpha} k^{\beta})?$$

dense: $M + N$, Mv in $\mathcal{O}(n^2)$ and $\Lambda(M)$ in $\mathcal{O}(n^3)$

LR Cholesky Algorithm

QR-like Algorithm

LR Cholesky Algorithm

LR Cholesky Algorithm

LR Cholesky Algorithm

[RUTISHAUSER 1958]

LR-Cholesky Transformation

for $i = 1, \ldots$ **do**

$L_i L_i^T = M_i$

$M_{i+1} = L_i^T L_i$

end

LR Cholesky Algorithm

[Rutishauser 1958]

LR-Cholesky Transformation

for $i = 1, \ldots$ **do**

$$L_i L_i^T = M_i \Rightarrow L_i = M_i L_i^{-T}$$
$$M_{i+1} = L_i^T L_i = L_i^T M_i L_i^{-T}$$

end

$$\lim_{i\to\infty} M_i = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n) \in \mathcal{H}(T, 0)$$

LR Cholesky Algorithm

[Rutishauser 1958]

LR-Cholesky Transformation

for $i = 1, \ldots$ **do**

$$L_i L_i^T = M_i - \mu_i \mathcal{I}$$

$$M_{i+1} = L_i^T L_i + \mu_i \mathcal{I}$$

end

$$\lim_{i \to \infty} M_i = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n) \in \mathcal{H}(T, 0)$$

$\forall i$: $M_i - \mu_i \mathcal{I}$ symmetric positive definite

LR Cholesky Algorithm

[RUTISHAUSER 1958]

LR-Cholesky Transformation

for $i = 1, \ldots$ **do**

$L_i L_i^T = M_i - \mu_i \mathcal{I}$

$M_{i+1} = L_i^T L_i + \mu_i \mathcal{I}$

end

$$\lim_{i \to \infty} M_i = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n) \in \mathcal{H}(T, 0)$$

$\forall i$: $M_i - \mu_i \mathcal{I}$ symmetric positive definite

$\mathcal{H}$-LR-Cholesky Transformation

for $i = 1, \ldots$ **do**

$\tilde{L}_i = \mathcal{H}$-`Cholesky factorization`$(\tilde{M}_i - \mu_i \mathcal{I})$

$\tilde{M}_{i+1} = \tilde{L}_i^T *_{\mathcal{H}} \tilde{L}_i + \mu_i \mathcal{I}$

end

LR Cholesky Algorithm

[Rutishauser 1958]

LR-Cholesky Transformation

for $i = 1, \ldots$ **do**

$L_i L_i^T = M_i - \mu_i \mathcal{I}$

$M_{i+1} = L_i^T L_i + \mu_i \mathcal{I}$

end

$$\lim_{i\to\infty} M_i = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n) \in \mathcal{H}(T, 0)$$

$\forall i$: $M_i - \mu_i \mathcal{I}$ symmetric positive definite

$\mathcal{H}$-LR-Cholesky Transformation

for $i = 1, \ldots$ **do**

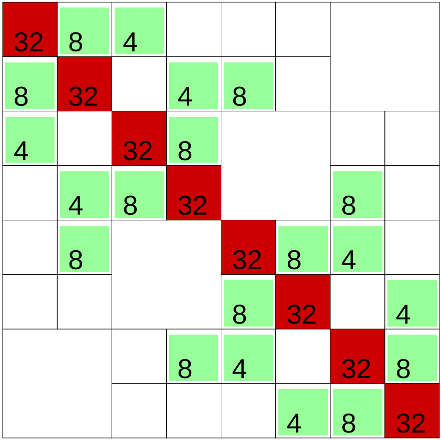
$\tilde{L}_i = \mathcal{H}\text{-}\texttt{Cholesky factorization}(\tilde{M}_i - \mu_i \mathcal{I})$

$\tilde{M}_{i+1} = \tilde{L}_i^T *_{\mathcal{H}} \tilde{L}_i + \mu_i \mathcal{I}$

end

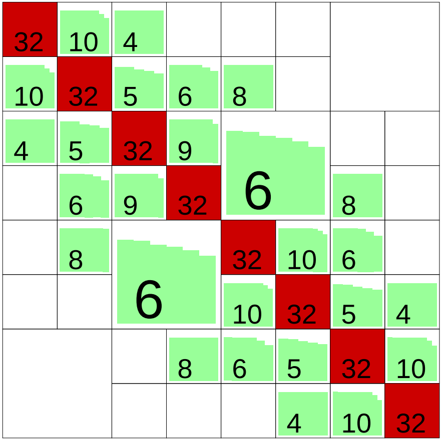
- shift strategy
- deflation

Example - $\mathcal{H}$-Fill-In



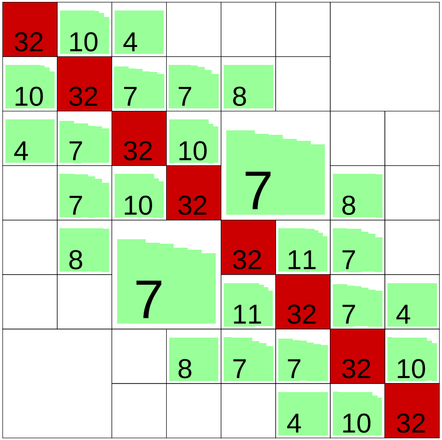
Matrix FEM16 ($\Delta_{2,h}$, 16 inner discr. points).

Example - $\mathcal{H}$-Fill-In



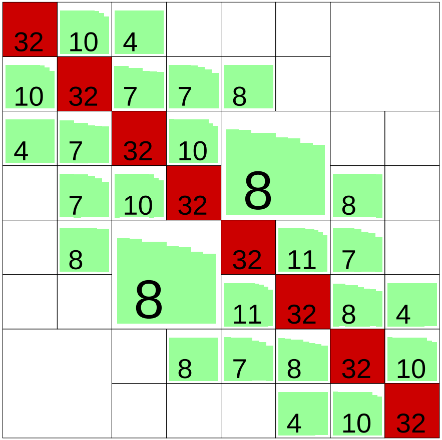
Matrix FEM16 ($\Delta_{2,h}$, 16 inner discr. points), after 1 step.

Example - $\mathcal{H}$-Fill-In



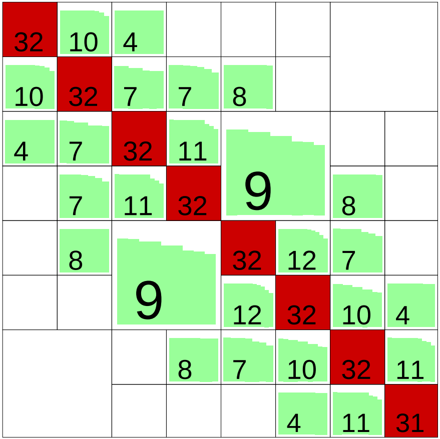
Matrix FEM16 ($\Delta_{2,h}$, 16 inner discr. points), after 2 steps.

Example - $\mathcal{H}$-Fill-In



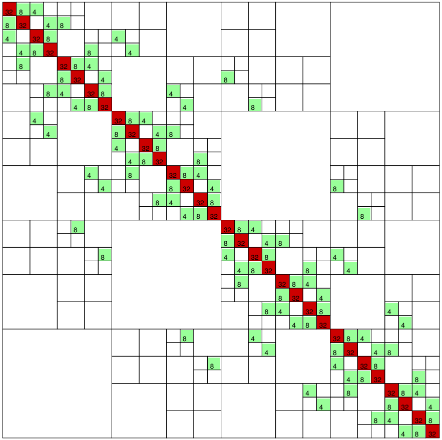
Matrix FEM16 ($\Delta_{2,h}$, 16 inner discr. points), after 3 steps.

Example - $\mathcal{H}$-Fill-In



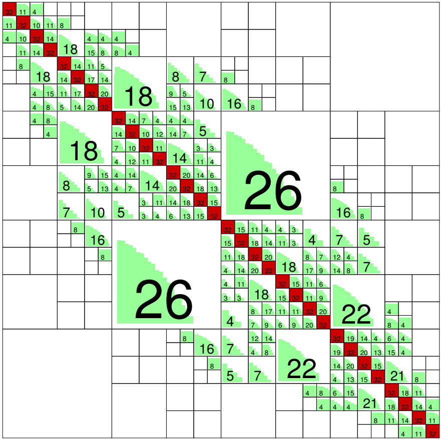
Matrix FEM16 ($\Delta_{2,h}$, 16 inner discr. points), after 4 steps.

Example - $\mathcal{H}$-Fill-In



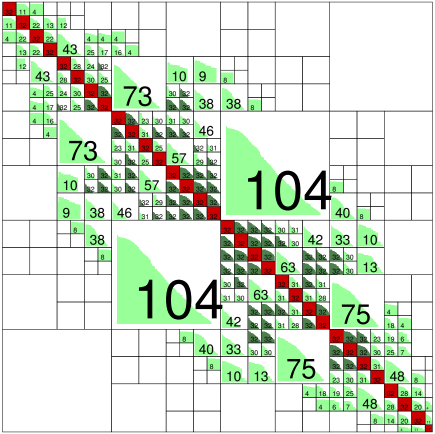
Matrix FEM32 ($\Delta_{2,h}$, 32 inner discr. points).

Example - $\mathcal{H}$-Fill-In



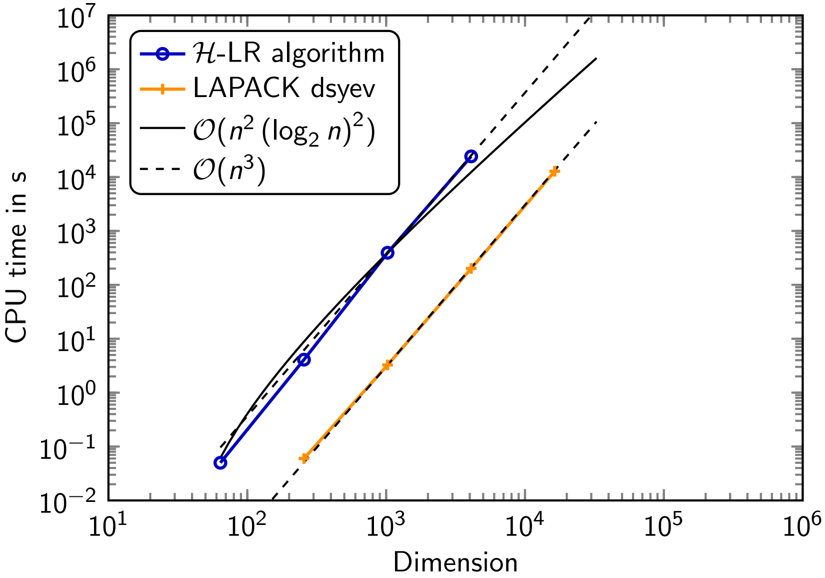
Matrix FEM32 ($\Delta_{2,h}$, 32 inner discr. points), after 1 step.

Example - $\mathcal{H}$-Fill-In

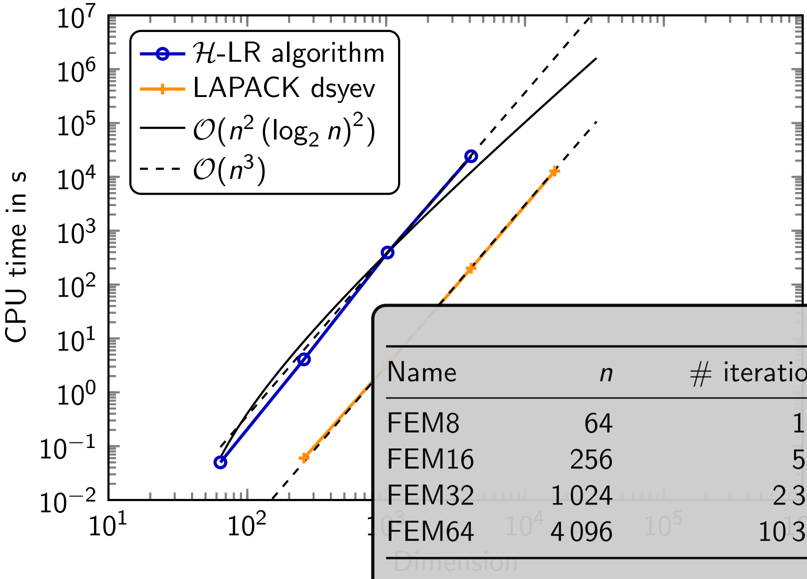


Matrix FEM32 ($\Delta_{2,h}$, 32 inner discr. points), after 50 steps.

Computation Time



Computation Time



Name	n	# iterations
FEM8	64	101
FEM16	256	556
FEM32	1 024	2 333
FEM64	4 096	10 320

Theorem

Adaption of [Fasino '05/Plestenjak, Van Barel, Van Camp '08]

$$M = \mathrm{diag}\left(d\right) + \sum_{i=1}^{r} \left(\mathrm{tril}\left(u_i v_i^T\right) + \mathrm{triu}\left(v_i u_i^T\right)\right)$$

Theorem

Adaption of [Fasino '05/Plestenjak, Van Barel, Van Camp '08]

$$M = \operatorname{diag}(d) + \sum_{i=1}^{r} \left(\operatorname{tril}\left(u_i v_i^T\right) + \operatorname{triu}\left(v_i u_i^T\right)\right)$$

Structure Preservation of dpss Matrices

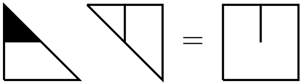
Let M be a symmetric positive definite diagonal plus semiseparable matrix, with a decomposition as in the definition. The Cholesky factor L of $M = LL^T$ can be written in the form

$$L = \operatorname{diag}\left(\tilde{d}\right) + \sum_{i=1}^{r} \operatorname{tril}\left(u_i \tilde{v}_i^T\right).$$

Multiplying the Cholesky factors in reverse order gives the next iterate $N = L^T L$ of the LR Cholesky algorithm. The matrix N has the same form as M,

$$N = \operatorname{diag}\left(\hat{d}\right) + \sum_{i=1}^{r} \left(\operatorname{tril}\left(\hat{u}_i \tilde{v}_i^T\right) + \operatorname{triu}\left(\tilde{v}_i \hat{u}_i^T\right)\right).$$

Proof Idea



$$L_{1:p-1,1:p-1} L_{p,1:p-1}^T = M_{1:p-1,p} = \sum_i v_i u_i^T$$

$$\Rightarrow L_{1:p-1,1:p-1}\ \tilde{v}_i\big|_{1:p-1} = v_i\big|_{1:p-1} \quad \text{and} \quad L_{p,1:p-1} = \sum_i u_i\big|_p\ \tilde{v}_i^T\Big|_{1:p-1}$$

$$\tilde{d}_p + \sum_i u_i\big|_p\ \tilde{v}_i\big|_p = L_{pp} = \sqrt{M_{pp} - L_{p,1:p-1} L_{p,1:p-1}^T}$$

L is a dpss matrix.

Proof Idea

$$N = L^T L = \left(\operatorname{diag}\left(\tilde{d}\right) + \sum_i \operatorname{tril}\left(u_i \tilde{v}_i^T\right)\right)^T \left(\operatorname{diag}\left(\tilde{d}\right) + \sum_i \operatorname{tril}\left(u_i \tilde{v}_i^T\right)\right)$$

$$\hat{u}^i = \left(Z + \operatorname{diag}\left(\tilde{d}\right)\right) u_i, \quad \text{with}$$

$$Z_{p,:} = \sum_j \tilde{v}_j\big|_p \begin{bmatrix} 0 & \cdots & 0 & u_j\big|_p & u_j\big|_{p+1} & \cdots & u_j\big|_n \end{bmatrix}$$

$$\operatorname{tril}(N, -1) = \sum_i \operatorname{tril}\left(\left(\operatorname{diag}(\tilde{d}) u_i + Z u_i\right) \tilde{v}_i^T, -1\right)$$

$$= \sum_i \operatorname{tril}\left(\hat{u}_i \tilde{v}_i^T, -1\right)$$

N is a dpss matrix.

Structure of $\hat{u}$ and $\tilde{v}$

$$M = \operatorname{diag}(d) + \sum_{i=1}^{r} \operatorname{tril}\left(u_i v_i^T\right) + \dots$$

$$N = \operatorname{diag}(d) + \sum_{i=1}^{r} \operatorname{tril}\left(\hat{u}_i \tilde{v}_i^T\right) + \dots$$

$$v_i = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ * \\ \vdots \\ * \\ 0 \\ \vdots \\ 0 \end{bmatrix} \rightsquigarrow \tilde{v}_i = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ * \\ \vdots \\ * \\ \color{blue}{*} \\ \color{blue}{\vdots} \\ \color{blue}{*} \end{bmatrix} \qquad u_i = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ * \\ \vdots \\ * \\ 0 \\ \vdots \\ 0 \end{bmatrix} \rightsquigarrow \hat{u}_i = \begin{bmatrix} \color{blue}{*} \\ \color{blue}{\vdots} \\ \color{blue}{*} \\ * \\ \vdots \\ * \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Hierarchical Matrices

$$\operatorname{rank}\left(M_{a:b,c:d}\right) = k \qquad M_{a:b,c:d} = AB^T$$

$$u_{i_r}^T = \begin{bmatrix} 0 & \cdots & 0 & A_{j,r}^T & 0 & \cdots & 0 \end{bmatrix}$$

$$v_{i_r}^T = \begin{bmatrix} 0 & \cdots & 0 & B_{j,r}^T & 0 & \cdots & 0 \end{bmatrix}, \quad r = 1, \ldots, k$$

$$\rightsquigarrow \hat{u}_{i_r}^T = \begin{bmatrix} * & \cdots & * & * & 0 & \cdots & 0 \end{bmatrix}$$

$$\tilde{v}_{i_r}^T = \begin{bmatrix} 0 & \cdots & 0 & * & * & \cdots & * \end{bmatrix}$$

$$\operatorname{tril}\left(u_i v_i^T\right) = \begin{bmatrix} 0 & & & & & & \\ 0 & 0 & & & & & \\ 0 & 0 & 0 & & & & \\ 0 & 0 & 0 & 0 & & & \\ 0 & * & * & 0 & 0 & & \\ 0 & * & * & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \rightsquigarrow \operatorname{tril}\left(\hat{u}_i \tilde{v}_i^T\right) = \begin{bmatrix} 0 & & & & & & \\ 0 & * & & & & & \\ 0 & * & * & & & & \\ 0 & * & * & * & & & \\ 0 & * & * & * & * & & \\ 0 & * & * & * & * & * & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Hierarchical Matrices

$$\operatorname{rank}(M_{a:b,c:d}) = k \qquad M_{a:b,c:d} = AB^T$$

$$u_{i_r}^T = \begin{bmatrix} 0 & \cdots & 0 & A_{j,r}^T & 0 & \cdots & 0 \end{bmatrix}$$

$$v_{i_r}^T = \begin{bmatrix} 0 & \cdots & 0 & B_{j,r}^T & 0 & \cdots & 0 \end{bmatrix}, \quad r = 1, \ldots, k$$

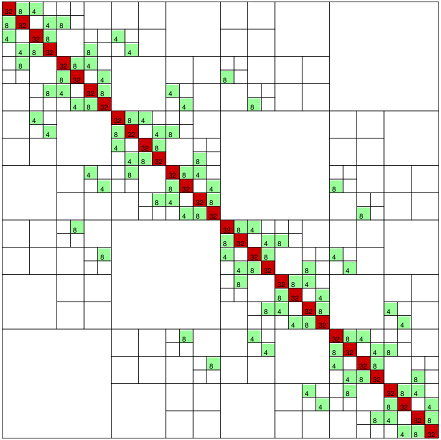
$$\rightsquigarrow \hat{u}_{i_r}^T = \begin{bmatrix} * & \cdots & * & * & 0 & \cdots & 0 \end{bmatrix}$$

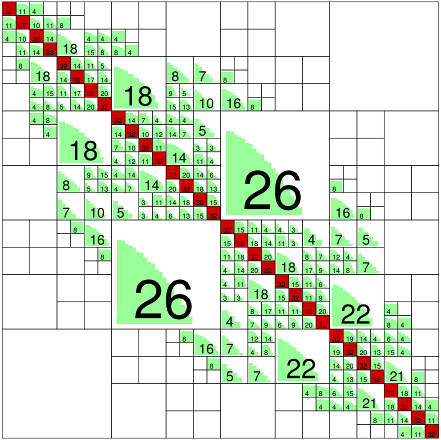
$$\tilde{v}_{i_r}^T = \begin{bmatrix} 0 & \cdots & 0 & * & * & \cdots & * \end{bmatrix}$$

$$\operatorname{tril}\left(u_i v_i^T\right) = \begin{bmatrix} 0 & & & & & & \\ 0 & 0 & & & & & \\ 0 & 0 & 0 & & & & \\ 0 & 0 & 0 & 0 & & & \\ 0 & * & * & 0 & 0 & & \\ 0 & * & * & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \rightsquigarrow \operatorname{tril}\left(\hat{u}_i \tilde{v}_i^T\right) = \begin{bmatrix} 0 & & & & & & \\ 0 & * & & & & & \\ 0 & * & * & & & & \\ 0 & * & * & * & & & \\ 0 & * & * & * & * & & \\ 0 & * & * & * & * & * & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

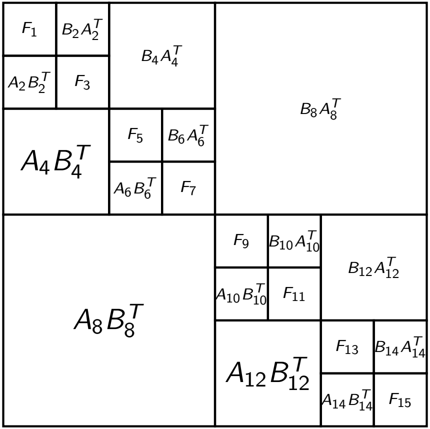
The structure of hierarchical matrices is not preserved under LR Cholesky transformations.

Example - $\mathcal{H}$-Fill-In

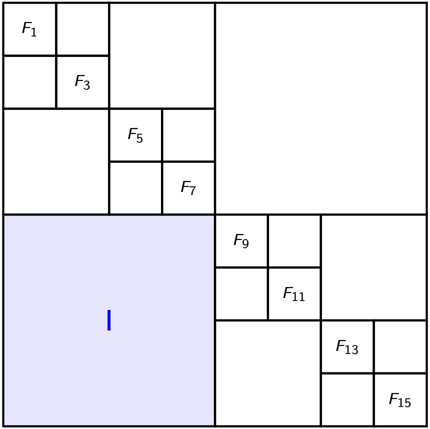




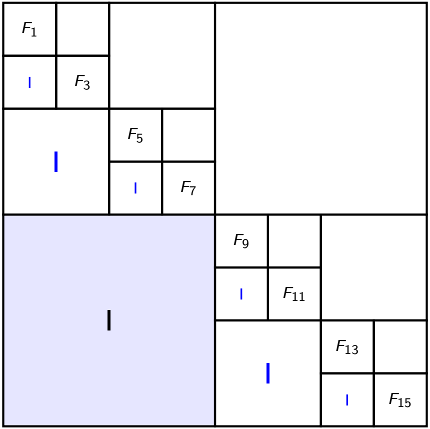
$\mathcal{H}_\ell$-Matrices



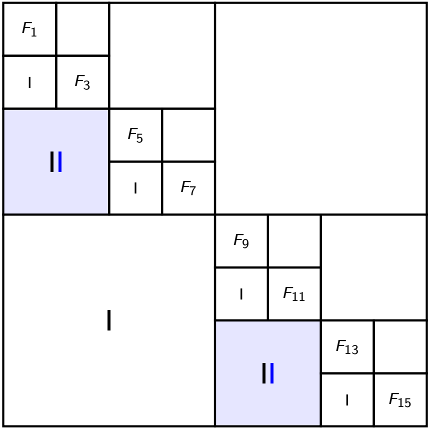
$\mathcal{H}_\ell$-Matrices



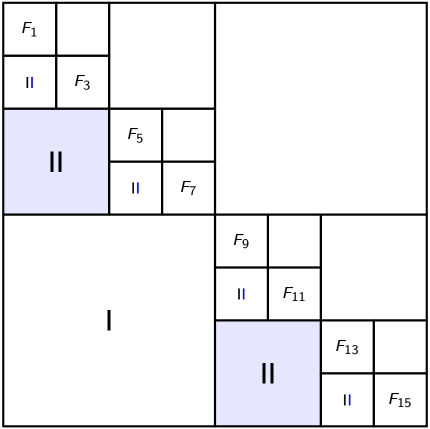
$\mathcal{H}_\ell$-Matrices



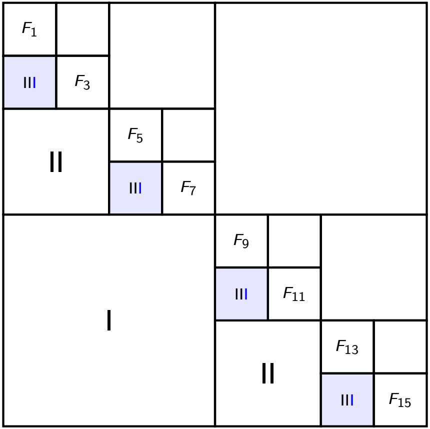
$\mathcal{H}_\ell$-Matrices



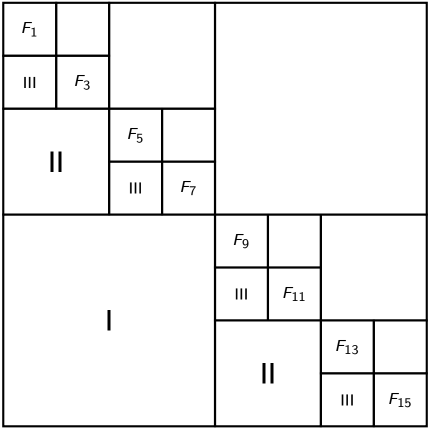
$\mathcal{H}_\ell$-Matrices



$\mathcal{H}_\ell$-Matrices



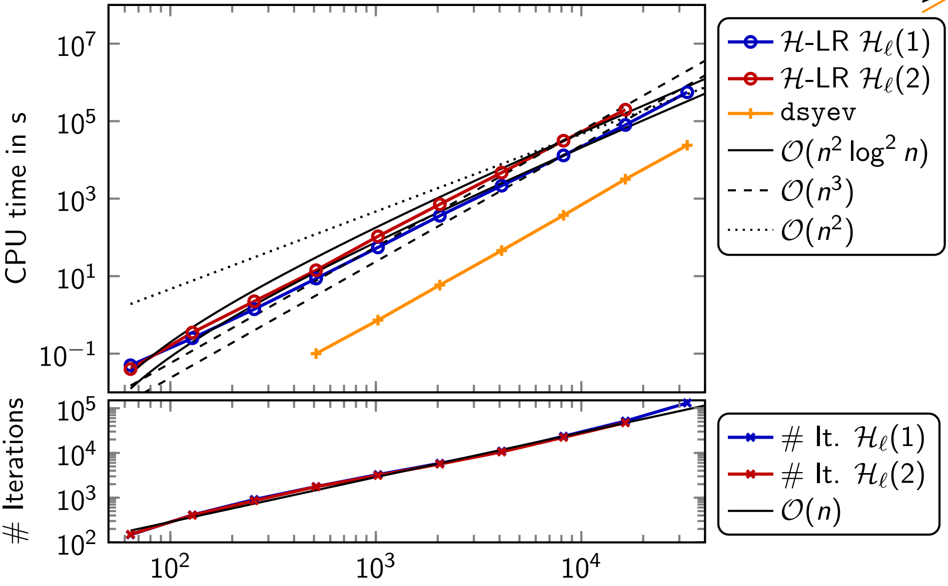
$\mathcal{H}_\ell$-Matrices



$\Rightarrow$ rank bounded by ℓk instead of k

$\Rightarrow$ total storage required by the low-rank parts of M is increased only from $2nk\ell$ to $2nk\frac{\ell(\ell-1)}{2}$

Computation Time $\mathcal{H}_\ell$-Matrices

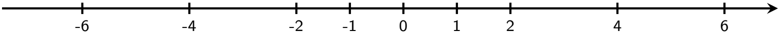


Slicing the Spectrum

Slicing the Spectrum

Bisectioning

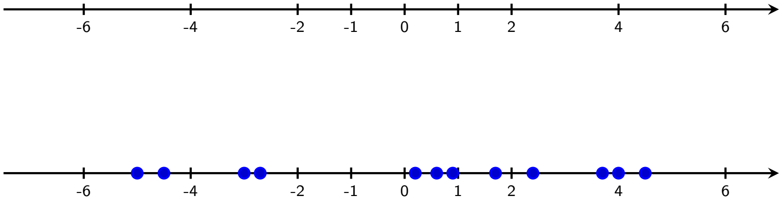
[Parlett '80]



Bisectioning

[Parlett '80]

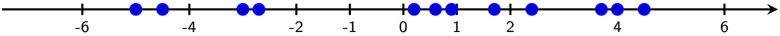
$\lambda_3 = ?$

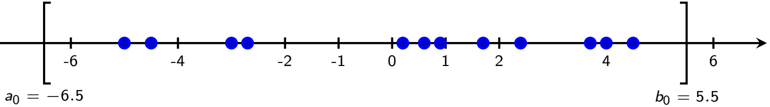


Bisectioning

[Parlett '80]

$\lambda_3 = ?$

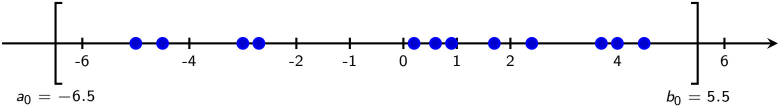


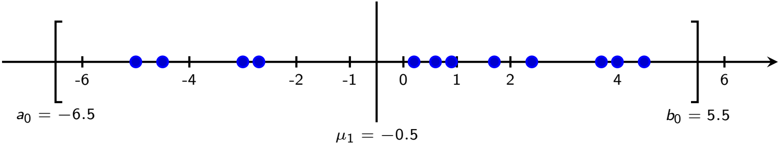


Bisectioning

[Parlett '80]

$\lambda_3 = ?$

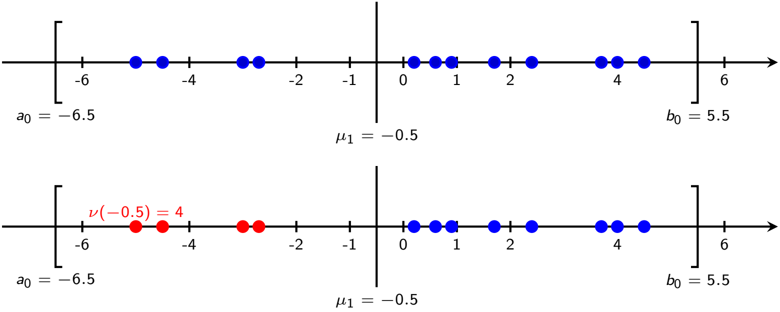




Bisectioning

[PARLETT '80]

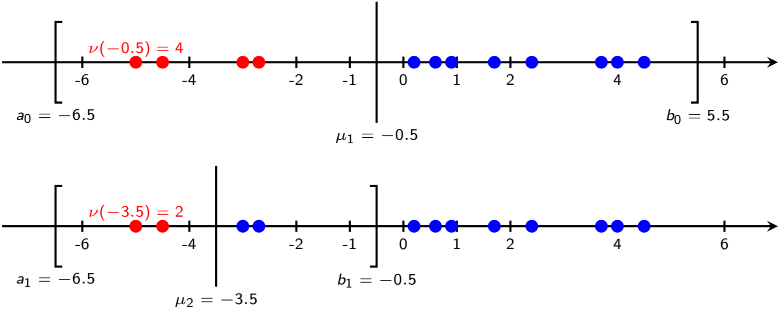
$\lambda_3 = ?$



Bisectioning

[Parlett '80]

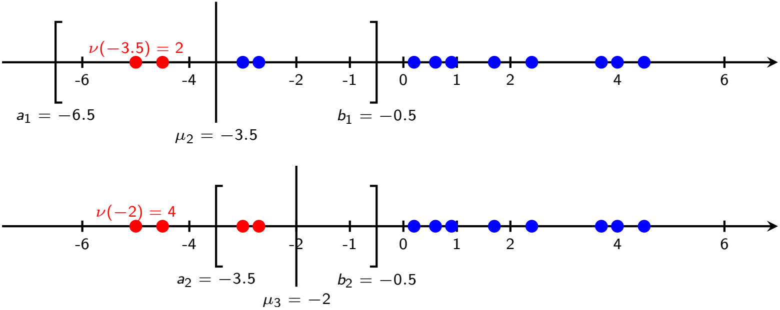
$\lambda_3 = ?$



Bisectioning

[PARLETT '80]

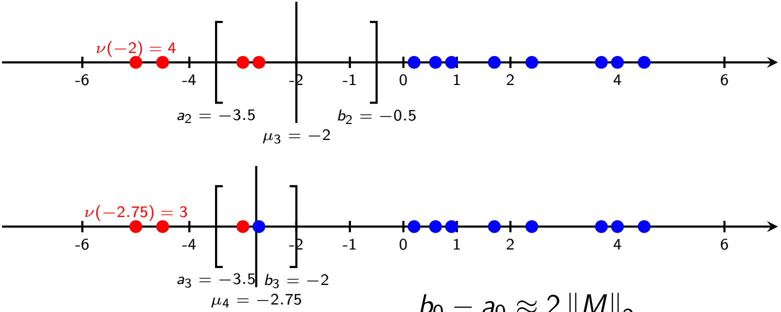
$\lambda_3 = ?$



Bisectioning

[PARLETT '80]

$\lambda_3 \in [-3.5, -2.75]$, $\hat{\lambda}_3 = -3.125$



$$b_0 - a_0 \approx 2\,\|M\|_2$$
$$\left|\lambda_i - \hat{\lambda}_i\right| < \epsilon \Leftrightarrow b_n - a_n < 2\epsilon$$
$$b_{i+1} - a_{i+1} = \tfrac{1}{2}\,(b_i - a_i)$$
$$\Rightarrow \mathcal{O}\left(\log_2(\|M\|_2\,/\epsilon)\right)$$

Evaluation of $\nu(\mu)$

Sylvester's Law of Inertia

Each matrix M is congruent to a matrix

$$\operatorname{diag}\left(-I_\nu, I_{\operatorname{rank}(M)-\nu}, 0_{n-\operatorname{rank}(M)}\right),$$

where ν is the number of negative eigenvalues. The triple

$$(\nu, \operatorname{rank}(M) - \nu, n - \operatorname{rank}(M))$$

is called the *inertia* of M.

$$\begin{aligned} M = LDL^T \quad &\Rightarrow \quad \nu(M) = \nu(D) \\ M - \mu I = L_\mu D_\mu L_\mu^T \quad &\Rightarrow \quad \nu(\mu) = \nu(M - \mu I) = \nu(D_\mu) \end{aligned}$$

Complexity

Flops per factorization (for $\mathcal{H}_\ell$-matrices):

$$\mathcal{O}\left(nk^2 (\log n)^4\right).$$

Factorization per eigenvalue:

$$\mathcal{O}\left(\log(\|M\|_2 / \epsilon)\right).$$

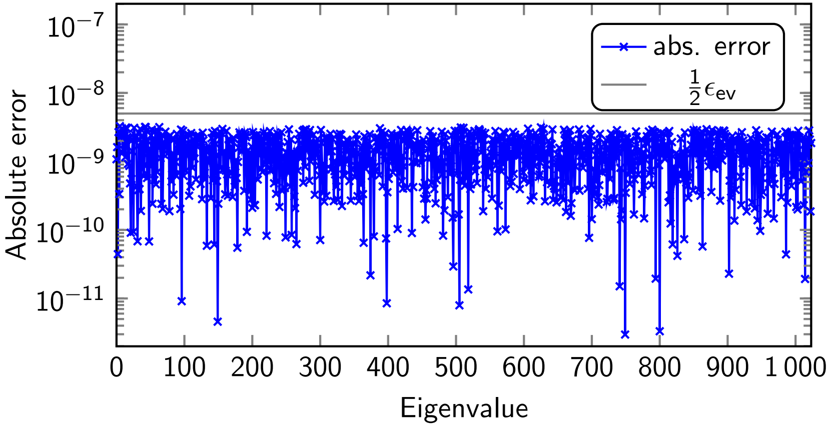
Flops per eigenvalue:

$$\mathcal{O}\left(nk^2 (\log n)^4 \log \|M\|_2 / \epsilon\right).$$

Slicing the whole spectrum:

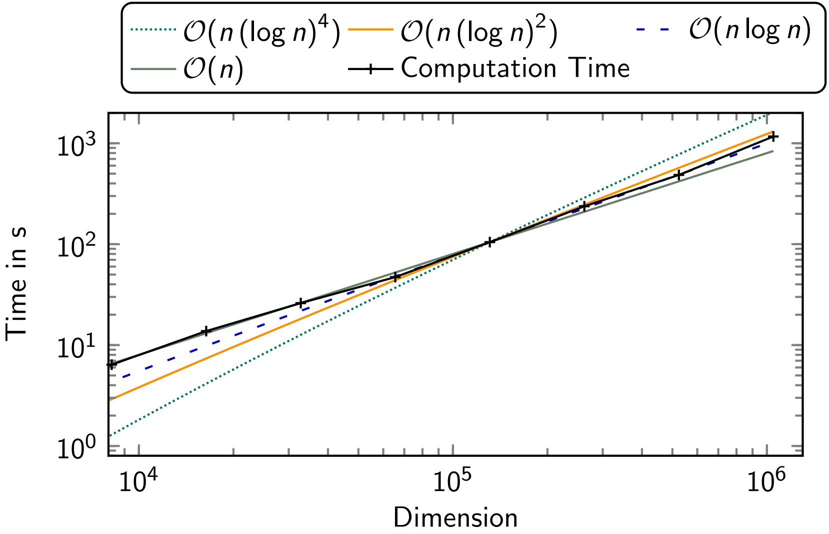
$$\mathcal{O}\left(n^2 k^2 (\log n)^4 \log(\|M\|_2 / \epsilon)\right).$$

Absolute Error for $\mathcal{H}_5(1)$-Matrix



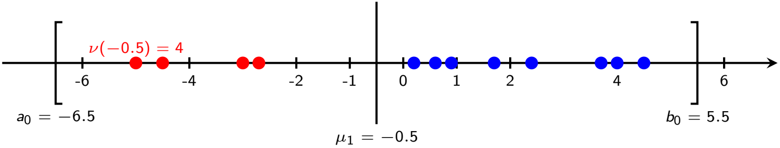
Absolute error $|\lambda_i - \hat{\lambda}_i|$ for the $1\,024 \times 1\,024$ $\mathcal{H}_5(1)$-matrix, $\epsilon_{\text{ev}} = 10^{-8}$.

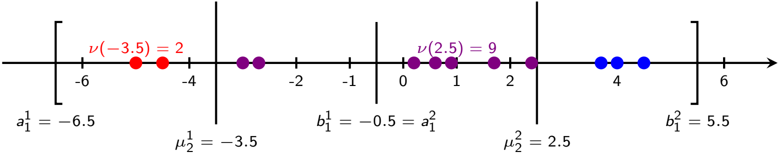
CPU Time for 10 Eigenvalues



Computation times for 10 eigenvalues of $\mathcal{H}_\ell(1)$-matrices ($\ell = 8, \ldots, 15$).

Parallelization





Parallelization Speedup

OpenMP:

Name	n	$t_{1\text{ core}}$ in s	t_{1c}/t_{2c}	t_{1c}/t_{4c}	$t_{8\text{ core}}$	t_{1c}/t_{8c}
H2 r1	128	0.33	1.83	3.30	0.06	5.50
H4 r1	512	9.44	1.94	3.67	1.43	6.60
H6 r1	2 048	219.28	1.91	3.64	33.88	6.47
H8 r1	8 192	4 022.80	1.87	3.44	676.57	5.95
H10 r1	32 768	49 012.24	1.93	3.18	10 006.60	4.90

Parallelization Speedup

Open MPI: $(\mathcal{H}_9(1) \in \mathbb{R}^{16\,384 \times 16\,384})$

No. of Processes	t in s	Speedup	Efficiency
1	16 564.58	1.00	1.00
2+1	8 340.22	1.99	0.66
4+1	4 044.13	4.10	0.82
6+1	2 678.95	6.18	0.88
11+1	1 494.33	11.08	0.92
23+1	713.80	23.21	0.96
35+1	476.44	34.77	0.96
47+1	364.30	45.47	0.95
95+1	188.92	87.68	0.91
191+1	100.61	164.64	0.86
287+1	71.86	230.50	0.80
383+1	61.91	267.56	0.70

$\mathcal{H}$-Matrices

- Shifting affects the structure.
- The LDL^T factorization is of almost linear complexity only for the original $\mathcal{H}$-matrix and not necessarily for the shifted ones.
- For $\mathcal{H}_\ell$-matrices we use the exact LDL^T factorization.
  For general $\mathcal{H}$-matrices: The truncation introduces errors in the LDL^T factorization — the computed D may have another inertia
  $\Rightarrow$ some eigenvalues may lie outside the computed interval
  $\Rightarrow$ larger errors

Preconditioned Inverse Iteration

Preconditioned Inverse Iteration for Hierarchical Matrices

Preconditioned Inverse Iteration

[Knyazev, Neymeyr, et al.]

Definition

The function

$$\mu(x) = \mu(x, M) = \frac{x^T M x}{x^T x}$$

is called the *Rayleigh quotient*.

Preconditioned Inverse Iteration

[Knyazev, Neymeyr, et al.]

Definition

The function

$$\mu(x) = \mu(x, M) = \frac{x^T M x}{x^T x}$$

is called the *Rayleigh quotient*.

Minimize the Rayleigh quotient by a gradient method:

$$x_{i+1} := x_i - \alpha \nabla \mu(x_i), \quad \nabla \mu(x) = \frac{2}{x^T x} \left(Mx - x\mu(x) \right),$$

Preconditioned Inverse Iteration

[Knyazev, Neymeyr, et al.]

Definition

The function

$$\mu(x) = \mu(x, M) = \frac{x^T M x}{x^T x}$$

is called the *Rayleigh quotient*.

Minimize the Rayleigh quotient by a gradient method:

$$x_{i+1} := x_i - \alpha \nabla \mu(x_i), \quad \nabla \mu(x) = \frac{2}{x^T x} \left(Mx - x\mu(x) \right),$$

$+$ preconditioning $\Rightarrow$ update equation:

$$x_{i+1} := x_i - B^{-1} \left(M x_i - x_i \mu(x_i) \right).$$

Preconditioned Inverse Iteration

[Knyazev, Neymeyr 2009]

$$x_{i+1} := x_i - B^{-1}\left(Mx_i - x_i\mu(x_i)\right)$$

If

- $M \in \mathbb{R}^{n\times n}$ symmetric positive definite and
- B^{-1} approximates the inverse of M, such that

$$\left\|\mathcal{I} - B^{-1}M\right\|_M \leq c < 1,$$

then Preconditioned INVerse ITeration (PINVIT) converges and the number of iterations is independent of n.

Algorithm and Complexity

The number of iterations is independent of matrix size n.

$\mathcal{H}$-PINVIT

Input: $M \in \mathbb{R}^{n \times n}$, $X_0 \in \mathbb{R}^{n \times d}$ ($X_0^T X_0 = I$, e.g. randomly chosen)
Output: $X_p \in \mathbb{R}^{n \times d}$, $\mu \in \mathbb{R}^{d \times d}$, with $\|MX_p - X_p\mu\| \leq \epsilon$

$B^{-1} = (M)_{\mathcal{H}}^{-1}$ or $B^{-1} = L_{\mathcal{H}}^{-T} L_{\mathcal{H}}^{-1}$
$R := MX_0 - X_0\mu, \quad \mu = X_0^T MX_0$
for $(i := 1;\ \|R\|_F > \epsilon;\ i++)$ **do**
  $X_i := \text{Orthogonalize}\left(X_{i-1} - B^{-1}R\right)$
  $R := MX_i - X_i\mu, \quad \mu = X_i^T MX_i$
end

Algorithm and Complexity

The number of iterations is independent of matrix size n.

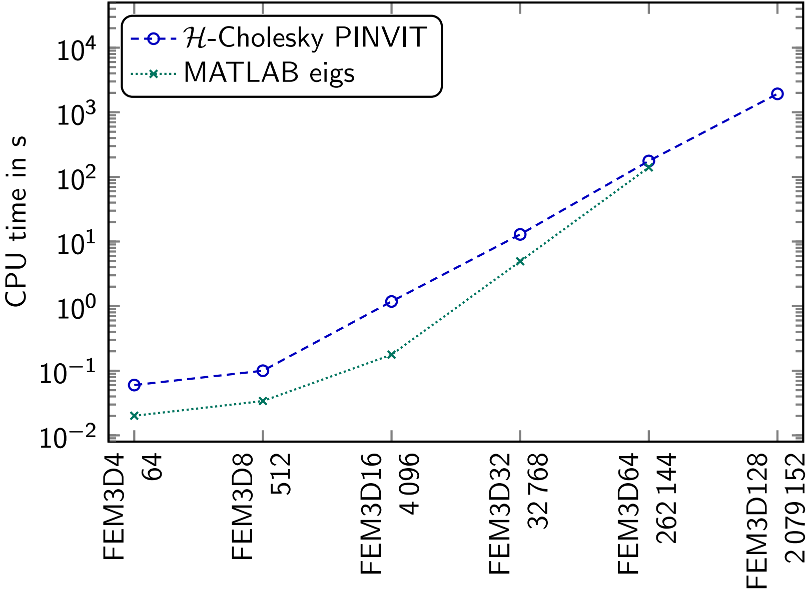
$\mathcal{H}$-PINVIT

Input: $M \in \mathbb{R}^{n\times n}$, $X_0 \in \mathbb{R}^{n\times d}$ ($X_0^T X_0 = I$, e.g. randomly chosen)
Output: $X_p \in \mathbb{R}^{n\times d}$, $\mu \in \mathbb{R}^{d\times d}$, with $\|MX_p - X_p\mu\| \leq \epsilon$

$B^{-1} = (M)_{\mathcal{H}}^{-1}$ or $B^{-1} = L_{\mathcal{H}}^{-T} L_{\mathcal{H}}^{-1}$ $\qquad \mathcal{O}(n\,(\log n)^2\,k\,(c)^2)$
$R := MX_0 - X_0\mu, \quad \mu = X_0^T M X_0$
for $(i := 1;\ \|R\|_F > \epsilon;\ i++)$ **do**
  $X_i := \text{Orthogonalize}\left(X_{i-1} - B^{-1}R\right)$ $\qquad \mathcal{O}(n\,(\log n)\,k\,(c)^2)$
  $R := MX_i - X_i\mu, \quad \mu = X_i^T MX_i$ $\qquad \mathcal{O}(n\,(\log n)\,k\,(c))$
end

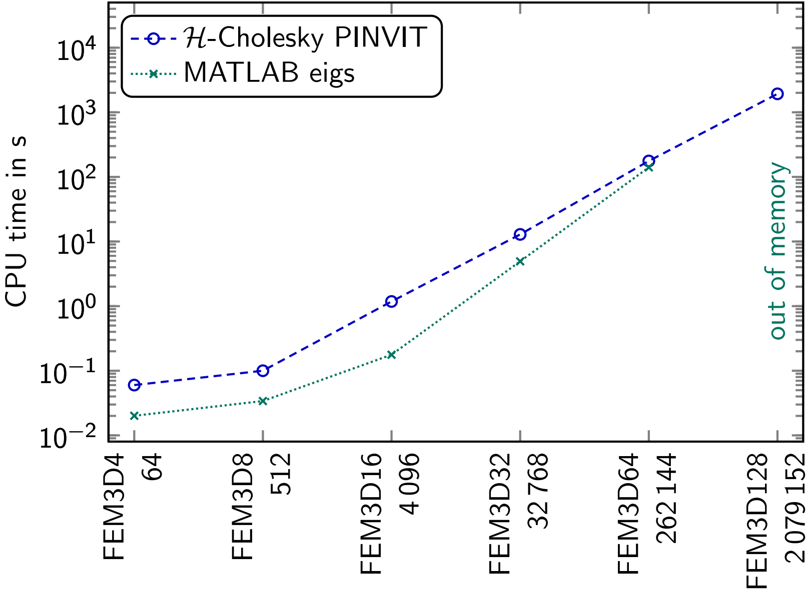
The complexity of the algorithm is determined by the $\mathcal{H}$-matrix inversion/Cholesky decomposition: $\Rightarrow \mathcal{O}(n\,(\log n)^2\,k\,(c)^2)$.

CPU Time

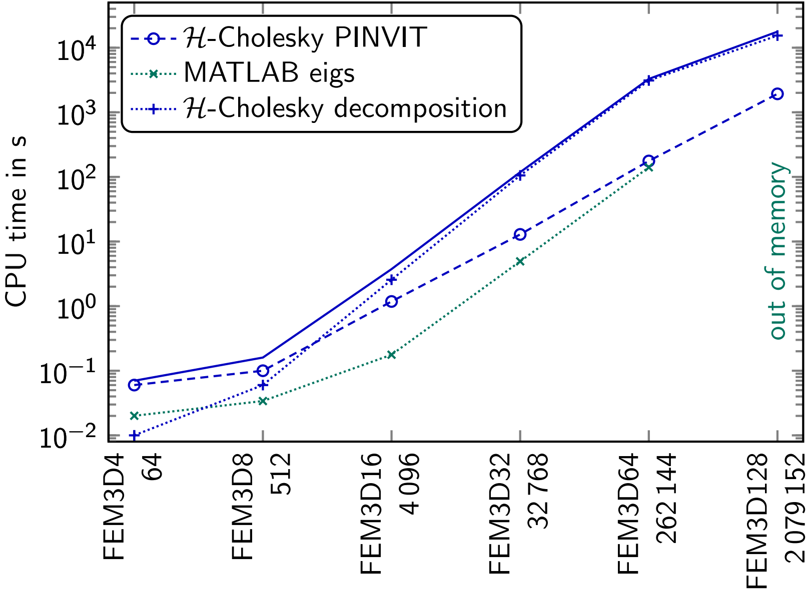


$\mathcal{H}$-Cholesky PINVIT
MATLAB eigs
CPU time in s
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
10^{3}
10^{4}
FEM3D4 64
FEM3D8 512
FEM3D16 4 096
FEM3D32 32 768
FEM3D64 262 144
FEM3D128 2 079 152

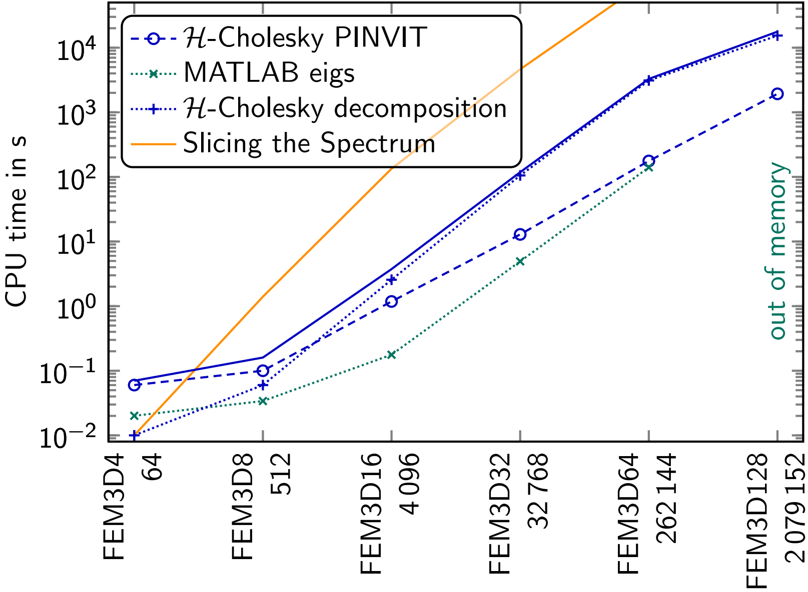
CPU Time


$\mathcal{H}$-Cholesky PINVIT
MATLAB eigs
CPU time in s
10^{4}
10^{3}
10^{2}
10^{1}
10^{0}
10^{-1}
10^{-2}
out of memory
FEM3D4 64
FEM3D8 512
FEM3D16 4 096
FEM3D32 32 768
FEM3D64 262 144
FEM3D128 2 079 152

CPU Time


$\mathcal{H}$-Cholesky PINVIT
MATLAB eigs
$\mathcal{H}$-Cholesky decomposition
CPU time in s
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
10^{3}
10^{4}
out of memory
FEM3D4 64
FEM3D8 512
FEM3D16 4 096
FEM3D32 32 768
FEM3D64 262 144
FEM3D128 2 079 152

CPU Time



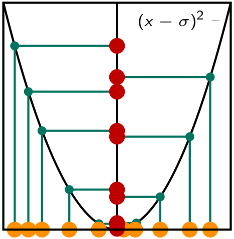
$\mathcal{H}$-Cholesky PINVIT
MATLAB eigs
$\mathcal{H}$-Cholesky decomposition
Slicing the Spectrum
CPU time in s
10^4
10^3
10^2
10^1
10^0
10^{-1}
10^{-2}
out of memory
FEM3D4 64
FEM3D8 512
FEM3D16 4 096
FEM3D32 32 768
FEM3D64 262 144
FEM3D128 2 079 152

Folded Spectrum Method

Folded Spectrum Method [Wang, Zunger 1994]

$$M_\sigma = (M - \sigma\mathcal{I})^2$$

M_σ is s.p.d., if M is s.p.d. and $\sigma \neq \lambda_i$.



Folded Spectrum Method

Folded Spectrum Method [Wang, Zunger 1994]

$$M_\sigma = (M - \sigma\mathcal{I})^2$$

M_σ is s.p.d., if M is s.p.d. and $\sigma \neq \lambda_i$.

- The condition number of $(M - \sigma I)^2$ is large.

$\Rightarrow$ The computation of M_σ^{-1} is more expensive.

$\Rightarrow$ M_σ^{-1} has larger local ranks.

$\Rightarrow$ $M_\sigma^{-1}v$ is more expensive.

Folded Spectrum Method

Folded Spectrum Method [Wang, Zunger 1994]

$$M_\sigma = (M - \sigma\mathcal{I})^2$$

M_σ is s.p.d., if M is s.p.d. and $\sigma \neq \lambda_i$.

- The condition number of $(M - \sigma I)^2$ is large.

$\Rightarrow$ The computation of M_σ^{-1} is more expensive.

$\Rightarrow$ M_σ^{-1} has larger local ranks.

$\Rightarrow$ $M_\sigma^{-1} v$ is more expensive.

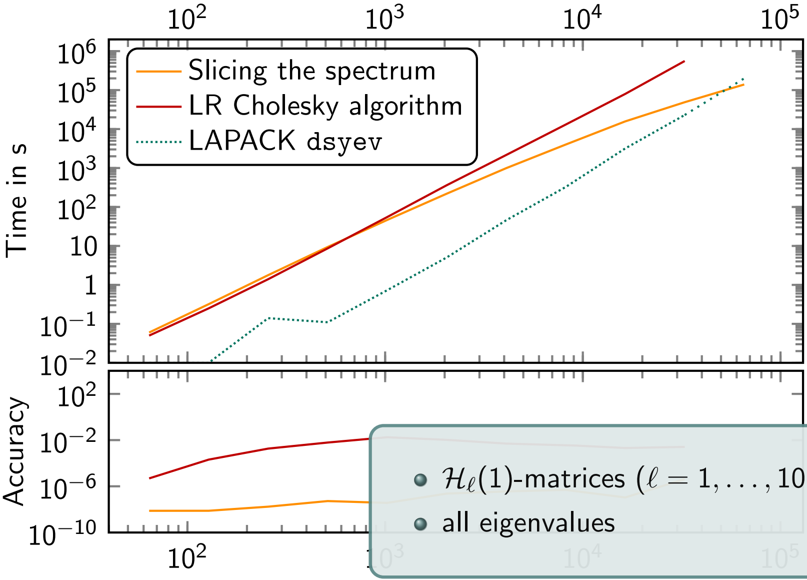
- Multiple eigenvalues of M_σ may lead to incomplete subspace information.

$\Rightarrow$ $v^T M v / v^T v$ does not approximate λ.

Conclusions

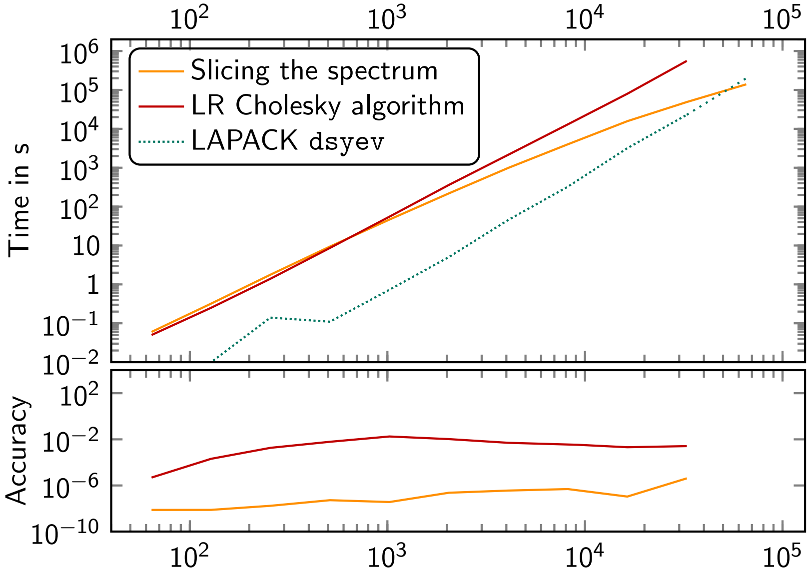
Conclusions

Slicing the Spectrum vs. LR Cholesky

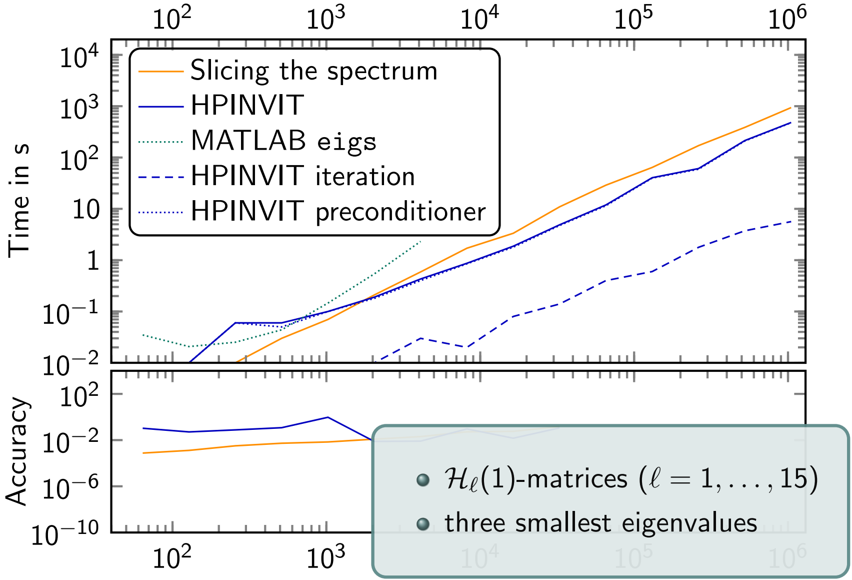


- $\mathcal{H}_\ell(1)$-matrices ($\ell = 1, \ldots, 10$)
- all eigenvalues

Slicing the Spectrum vs. LR Cholesky

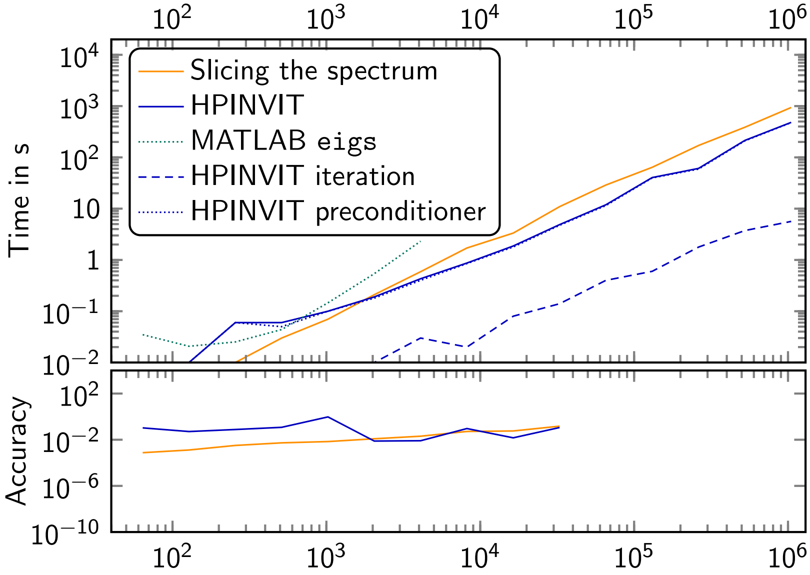


Slicing the Spectrum vs. $\mathcal{H}$-PINVIT



- $\mathcal{H}_\ell(1)$-matrices ($\ell = 1, \ldots, 15$)
- three smallest eigenvalues

Slicing the Spectrum vs. $\mathcal{H}$-PINVIT



Conclusions

- Three algorithms:
  - $\mathcal{H}$-**LR Cholesky algorithm:** efficient only for $\mathcal{H}_\ell$-matrices, but expensive otherwise

Conclusions

- Three algorithms:
  - $\mathcal{H}$-**LR Cholesky algorithm:** efficient only for $\mathcal{H}_\ell$-matrices, but expensive otherwise
  - **Slicing the spectrum:** efficient for $\mathcal{H}_\ell$-matrices, good parallel performance

Conclusions

- Three algorithms:
  - $\mathcal{H}$-**LR Cholesky algorithm:** efficient only for $\mathcal{H}_\ell$-matrices, but expensive otherwise
  - **Slicing the spectrum:** efficient for $\mathcal{H}_\ell$-matrices, good parallel performance
  - $\mathcal{H}$-**PINVIT:** efficient for the smallest eigenvalue(s) of positive definite $\mathcal{H}$-matrices, computation of inner eigenvalues is possible. Subspace-accelerated variants, LOBPCG also investigated.

Conclusions

- Three algorithms:
  - $\mathcal{H}$-**LR Cholesky algorithm:** efficient only for $\mathcal{H}_\ell$-matrices, but expensive otherwise
  - **Slicing the spectrum:** efficient for $\mathcal{H}_\ell$-matrices, good parallel performance
  - $\mathcal{H}$-**PINVIT:** efficient for the smallest eigenvalue(s) of positive definite $\mathcal{H}$-matrices, computation of inner eigenvalues is possible. Subspace-accelerated variants, LOBPCG also investigated.
- Other ideas ot compute eigenvalues of $\mathcal{H}$-matrices:

J. Gördes.
Eigenwertproblem von hierarchischen Matrizen mit lokalem Rang 1.
Diplomarbeit, Mathematisch-Naturwissenschaftlichen Fakultät, CAU Kiel, May 2009.

W. Hackbusch and W. Kress.
A projection method for the computation of inner eigenvalues using high degree rational operators.
Computing 81:259–268, 2007.

S. Delvaux, K. Frederix, and M. Van Barel.
Transforming a hierarchical into a unitary-weight representation.
Electr. Trans. Num. Anal. 33:163–188, 2009.

Conclusions

- Three algorithms:
  - $\mathcal{H}$-**LR Cholesky algorithm:** efficient only for $\mathcal{H}_\ell$-matrices, but expensive otherwise
  - **Slicing the spectrum:** efficient for $\mathcal{H}_\ell$-matrices, good parallel performance
  - $\mathcal{H}$-**PINVIT:** efficient for the smallest eigenvalue(s) of positive definite $\mathcal{H}$-matrices, computation of inner eigenvalues is possible. Subspace-accelerated variants, LOBPCG also investigated.
- Probably similar conclusions for unsymmetric case.

Conclusions

- Three algorithms:
  - $\mathcal{H}$-**LR Cholesky algorithm:** efficient only for $\mathcal{H}_\ell$-matrices, but expensive otherwise
  - **Slicing the spectrum:** efficient for $\mathcal{H}_\ell$-matrices, good parallel performance
  - $\mathcal{H}$-**PINVIT:** efficient for the smallest eigenvalue(s) of positive definite $\mathcal{H}$-matrices, computation of inner eigenvalues is possible. Subspace-accelerated variants, LOBPCG also investigated.
- Probably similar conclusions for unsymmetric case.
- Possible improvements employing PDE theory?

Conclusions

- Three algorithms:
  - $\mathcal{H}$-**LR Cholesky algorithm:** efficient only for $\mathcal{H}_\ell$-matrices, but expensive otherwise
  - **Slicing the spectrum:** efficient for $\mathcal{H}_\ell$-matrices, good parallel performance
  - $\mathcal{H}$-**PINVIT:** efficient for the smallest eigenvalue(s) of positive definite $\mathcal{H}$-matrices, computation of inner eigenvalues is possible. Subspace-accelerated variants, LOBPCG also investigated.
- Probably similar conclusions for unsymmetric case.
- Possible improvements employing PDE theory?
- Concepts carry over to TT format, but so far not competitive with tensor-specific methods.

Details to be found in...

Peter Benner and Thomas Mach.
The LR Cholesky algorithm for symmetric hierarchical matrices.
Linear Algebra and its Applications 439(4):1150–1166, 2013.

Peter Benner and Thomas Mach.
Computing all or some eigenvalues of symmetric $\mathcal{H}_\ell$-matrices.
SIAM Journal on Scientific Computing 34(1):A485–A496, 2012.

Peter Benner and Thomas Mach.
The preconditioned inverse iteration for hierarchical matrices.
Numerical Linear Algebra with Applications 20(1):150–166, 2013.

Peter Benner and Thomas Mach.
Locally optimal block preconditioned conjugate gradient method for hierarchical matrices.
Proceedings in Applied Mathematics and Mechanics 11:741-742, 2011.

Peter Benner and Thomas Mach.
On the QR decomposition of $\mathcal{H}$-matrices.
Computing 88(3–4):111–129, 2010.

All about $\mathcal{H}$ackbusch matrices to be found in. . .

All about $\mathcal{H}$ackbusch matrices to be found in...

HAPPY BIRTHDAYS!