Peter Benner Thomas Mach

The Preconditioned Inverse Iteration for Hierarchical Matrices

Max Planck Institute Magdeburg Preprints

MPIMD/11-01 **February 11, 2011**

Impressum:

Max Planck Institute for Dynamics of Complex Technical Systems, Magdeburg

Publisher:
Max Planck Institute for Dynamics of Complex Technical Systems

Address:
Max Planck Institute for Dynamics of Complex Technical Systems
Sandtorstr. 1
39106 Magdeburg

www.mpi-magdeburg.mpg.de/preprints

THE PRECONDITIONED INVERSE ITERATION FOR HIERARCHICAL MATRICES

PETER BENNER AND THOMAS MACH

Abstract. The preconditioned inverse iteration [Ney01a] is an efficient method to compute the smallest eigenpair of a symmetric positive definite matrix M. Here we use this method to find the smallest eigenvalues of a hierarchical matrix [Hac99]. The storage complexity of the data-sparse $\mathcal{H}$-matrices is almost linear. We use $\mathcal{H}$-arithmetic to precondition with an approximate inverse of M or an approximate Cholesky decomposition of M. In general $\mathcal{H}$-arithmetic is of linear-polylogarithmic complexity, so the computation of one eigenvalue is cheap.

We extend the ideas to the computation of inner eigenvalues by computing an invariant subspaces S of $(M-\mu I)^2$ by subspace preconditioned inverse iteration. The eigenvalues of the generalized matrix Rayleigh quotient $\mu_M(S)$ are the wanted inner eigenvalues of M. The idea of using $(M-\mu I)^2$ instead of M is known as folded spectrum method [WZ94].

Numerical results substantiate the convergence properties and show that the computation of the eigenvalues is superior to existing algorithms for non-sparse matrices.

Keywords: symmetric hierarchical matrices, smallest eigenvalues, preconditioned inverse iteration, folded spectrum method, inner eigenvalues, adaptive $\mathcal{H}$-Cholesky decomposition

Mathematics Subject Classification: 65F15, 65F50, 15A18

1. Introduction

The numerical solution of the eigenvalue problem for partial differential equations is of constant interest, since it is an important task in many applications, see [KMOX09]. For instance the computation of the electronic structure of molecules requires the solution of an eigenvalue problem [SRNB09].

Hierarchical matrices are a useful tool to handle discretizations of partial differential operators. The discretization of integral operators with non-local kernel function, like they occur in the boundary element method (BEM) [BK05] or in population dynamics [KHS07], leads to dense matrices. Hierarchical matrices permit the usage of the partial low-rank structure of these matrices to reduce the storage and computational complexity. So it is beneficial to be able to compute eigenvalues of hierarchical matrices.

Let M be a hierarchical matrix of dimension $n \times n$. Further let M be real and symmetric positive definite. The eigenvalue $\lambda_i \in \mathbb{R}$ and the corresponding eigenvector $v_i \in \mathbb{R}^n \setminus \{0\}$ fulfill

$$Mv_i = v_i\lambda_i.$$

We are interested in some eigenpairs (λ_i, v_i) of M. The special structure of hierarchical matrices permits the computation of matrix-vector products and approximate inverses in almost linear complexity. These arithmetic operations are used in the preconditioned inverse iteration, which is suitable for this problem, since it is a so called matrix-free method, not requiring to have M or M^{-1} available as matrices, but as functions $x \to Mx$ and $x \to M^{-1}x$. The preconditioned inverse iteration computes the smallest eigenvalue and the corresponding eigenvector by a minimization process. In the remainder of this introduction we will list some well known facts on eigenvalues and Rayleigh quotients and some notation required in the subsequent sections.

Date: February 11, 2011.

Since we assume M to be symmetric positive definite, the eigenvalues λ_i are real and positive. We sort the eigenvalues in descending order so that the spectrum of M is

$$\Lambda(M) = \{\lambda_1, \ldots, \lambda_n\} \text{ with } \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n > 0.$$

Further it is well known that each symmetric matrix M is similar to a diagonal matrix, and that there is an orthogonal similarity transformation, which diagonalizes M:

$$Q^T M Q = \operatorname{diag}(\lambda_1, \ldots, \lambda_n), \text{ with } Q^T Q = \mathcal{I}.$$

Hence the condition number of the symmetric eigenvalue problem is 1 [GV96]. Let $Q' \in \mathbb{R}^{n\times d}$ be an isometry ($Q'^T Q' = \mathcal{I}_d$) spanning an invariant subspace ($MQ' = Q'Y$). Then the eigenvalues corresponding to the eigenvectors in $\operatorname{span}(Q')$ are

$$\Lambda\left(Q'^T M Q'\right) = \Lambda(Y).$$

This leads to the definition of the Rayleigh quotient, which is minimized by the preconditioned inverse iteration.

Definition 1.1. *(Rayleigh quotient)*
The function

$$\mu(x, M) : \mathbb{R}^n \times \mathbb{R}^{n\times n} \to \mathbb{R} : (x, M) \to \mu(x) = \mu(x, M) = \frac{x^T M x}{x^T x} \tag{1}$$

is called the Rayleigh quotient.

The Rayleigh quotient of the eigenvector v_i is $\mu(v_i) = \lambda_i$. The definition of the Rayleigh quotient can be generalized to full rank rectangular matrices X. For this purpose we need the trace of the matrix M, which is defined by

$$\operatorname{tr}(M) := \sum_{i=1}^{n} m_{ii} = \sum_{i=1}^{n} \lambda_i.$$

Definition 1.2. *(matrix Rayleigh quotient, generalized scalar Rayleigh quotient)* [AMSV02]
Let X be an element of the Grassmann manifold $X \in Gr(d, n) = \left\{X \in \mathbb{R}^{n\times d}, \text{column rank } X = d\right\}$, $d \leq n$. Then the matrix Rayleigh quotient *is defined by*

$$R(X, M) = \left(X^T X\right)^{-1} X^T M X. \tag{2}$$

The function

$$\mu(X, M) : \mathbb{R}^{n\times d} \to \mathbb{R} : X \to \mu(X, M) = \operatorname{tr}(R(X, M)) \tag{3}$$

is called the generalized scalar Rayleigh quotient.

Let S span an invariant subspace with $MS = SY$. Then the matrix Rayleigh quotient of S is

$$R(S, M) = \left(S^T S\right)^{-1} S^T M S = \left(S^T S\right)^{-1} \left(S^T S\right) Y = Y.$$

In the next section we give a brief summary of the preconditioned inverse iteration. The third section contains a short review of hierarchical matrices and their approximate inverses and Cholesky decompositions. Both concepts are combined in the forth section to a preconditioned inverse iteration for hierarchical matrices. Afterwards, we use the ideas of the folded spectrum method [WZ94] to extend the preconditioned inverse iteration to interior eigenvalues. We finally present some numerical results and concluding remarks.

2. Preconditioned Inverse Iteration

The first iterative eigensolvers with preconditioner dates back to the late 1950s, see [Kny98] or [Ney01c] for references. In [Ney01c] Neymeyr classifies the different schemes of preconditioned inverse iteration and proves some statements on the convergence rates, that can be found in [Ney01a, Ney01b, KN03], too. Further he gives some motivation for using preconditioned inverse iteration, we pick the following one from there:

The preconditioned inverse iteration, short PINVIT, can be regarded as a gradient-based minimization method minimizing the Rayleigh quotient. We have

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

and

$$\nabla\mu(x) = \frac{2}{x^T x}\left(Mx - x\mu(x)\right).$$

This leads to the gradient method

$$x_i := x_{i-1} - \alpha\left(Mx_{i-1} - x_{i-1}\mu(x_{i-1})\right).$$

The convergence of this method is too slow, so one should use the acceleration by preconditioning the residuum $r = Mx - x\mu(x)$ with the preconditioner T^{-1}:

$$\nabla_T\mu(x) = \frac{2}{x^T x}T^{-1}\left(Mx - x\mu(x)\right).$$

Finally we get the update equation:

$$x_i := x_{i-1} - T^{-1}r_{i-1} = x_{i-1} - T^{-1}\left(Mx_{i-1} - x_{i-1}\mu(x_{i-1})\right). \tag{4}$$

A second really simple argument, why we use the PINVIT is, that there is a nice sharp bound on the convergence rate of PINVIT proven in [Ney01a, Ney01b, KN03] and recently in a shorter form in [KN09]:

Theorem 2.1. [KN09, Theorem 1.1]
Let $x \in \mathbb{R}^n$ and $\lambda_j \le \mu(x) < \lambda_{j+1}$. If the preconditioner T^{-1} satisfies

$$\left\|\mathcal{I} - T^{-1}M\right\|_M \le c < 1, \tag{5}$$

then it holds for the Rayleigh quotient of the next iterate $\mu(x')$, that either $\mu(x') < \lambda_j$ or $\lambda_j \le \mu(x') < \mu(x)$. In the latter case,

$$\frac{\mu(x') - \lambda_j}{\lambda_{j+1} - \mu(x')} \le \gamma^2 \frac{\mu(x) - \lambda_j}{\lambda_{j+1} - \mu(x)},$$

where

$$\gamma = 1 - (1-c)\left(1 - \frac{\lambda_j}{\lambda_{j+1}}\right) \tag{6}$$

is the convergence factor.

There are some variations of this method slightly improving the convergence. One can use the optimal vector in the subspace spanned by x_{i-1} and $T^{-1}r_{i-1}$ as next iterate x_i. Neymeyr calls this variation PINVIT(2), another name is preconditioned steepest descent. Further one can use the subspace spanned by x_{i-1}, x_{i-2} and $T^{-1}r_{i-1}$. This method is called linear optimal (block) preconditioned conjugate gradient method (LOBPCG) or PINVIT(3) in Neymeyr's classification. Finally one can replace x_i by a subspace X_i of dimension s, then the methods are called PINVIT(k, s). The convergence factor γ in the subspace case depends among others on the quotient

$$\frac{\lambda_s}{\lambda_{s+1}}. \tag{7}$$

If this quotient is one, the subspace is not unique.

We will compute the eigenvalues of a hierarchical matrix using preconditioned inverse iteration in the version PINVIT(k, s), $k \le 3$. To do this we need $\mathcal{H}$-arithmetic, which will be explained briefly in the next section. We will mainly focus on the approximate inverse and the approximate Cholesky decomposition within the $\mathcal{H}$-matrix format.

3. Hierarchical Matrices

In this section we give a short introduction to hierarchical matrices as far as it is necessary for the remainder of this paper, for more details see e. g. [GH03, BGH03, Hac09, Beb08]. Hierarchical ($\mathcal{H}$-) matrices were introduced by W. Hackbusch in 1998 [Hac99]. Some matrices like boundary element (BEM) or finite element (FEM) matrices have submatrices that admit low-rank approximations. The basic idea of the $\mathcal{H}$-matrix format is to use a hierarchical structure to find and access such submatrices and to use good low-rank approximations to reduce their storage amount and the computation time involving these submatrices. These low-rank approximations make the $\mathcal{H}$-matrix format data-sparse. The need for truncation in order to close the class of $\mathcal{H}$-matrices under addition, multiplication and inversion makes formal $\mathcal{H}$-arithmetic an approximative arithmetic.

We will need a few definitions first, for details see [GH03]. A hierarchical tree, short $\mathcal{H}$-tree, T_I of an index set I is a tree with the special conditions:

- the index set I is the root of T_I and
- a vertex $r \in T_I$ is either the disjoint union of its sons $S(r)$ or a leaf of T_I.

We denote the set of leaves of the $\mathcal{H}$-tree T_I with $\mathcal{L}(T_I)$. The maximum length of the paths from the root to each leave is called $\operatorname{depth}(T)$, which is in $\mathcal{O}(\log n)$, if cardinality or geometrically balanced clustering is used [GH03, p. 320ff].

A hierarchical product tree, short $\mathcal{H}_\times$-tree, $T_{I\times I}$ is a special $\mathcal{H}$-tree over the index set $I \times I$ and can be regarded as the product set $T_I \times T_I$. Every vertex of $T_{I\times I}$ is the product of two vertices of the same level of the $\mathcal{H}$-tree T_I. For simplification we assume that both $\mathcal{H}$-trees, which form the $\mathcal{H}_\times$-tree, are identical.

Now we are able to define the set of $\mathcal{H}$-matrices based on the $\mathcal{H}_\times$-tree $T_{I\times I}$ with maximum rank k and minimum block size $n_{\min}$ by

$$\mathcal{H}(T_{I\times I}, k) := \left\{ M \in \mathbb{R}^{I\times I} \,\middle|\, \begin{array}{c} \forall r \times s \in \mathcal{L}(T_{I\times I}) : \operatorname{rank}(M_{r\times s}) \le k \\ \text{or } \#r \le n_{\min} \text{ or } \#s \le n_{\min} \end{array} \right\}.$$

A hierarchical matrix $M \in \mathcal{H}(T_{I\times I}, k)$ requires a storage of

$$N_{\mathcal{H},st}(T_{I\times I}, k) = \mathcal{O}(kn \log n),$$

where n is the size of I. Also a lot of formatted arithmetic needs only linear-polylogarithmic complexity ($M_1, M_2 \in \mathcal{H}(T_{I\times I}, k)$, $v \in \mathbb{R}^n$):

$$\begin{aligned} &M_1 *_{\mathcal{H}} v : && N_{\mathcal{H}*v}(T_{I\times I}, k) \in \mathcal{O}(kn \log n), \\ &M_1 +_{\mathcal{H}} M_2, M_1 -_{\mathcal{H}} M_2 : && N_{\mathcal{H}+\mathcal{H}}(T_{I\times I}, k) \in \mathcal{O}(k^2 n \log n), \\ &M_1 *_{\mathcal{H}} M_2, (M_1)^{-1}_{\mathcal{H}}, \mathcal{H}\mathrm{LU}(M_1) : && N_{\mathcal{H}*\mathcal{H}}/N_{\mathcal{H}^{-1}}/N_{LU(\mathcal{H})} \in \mathcal{O}(k^2 n \log^2 n). \end{aligned}$$

These arithmetic operations (and a lot more) are implemented in the $\mathcal{H}$lib [HLi09], which we use for the numerical examples.

In the remainder we will use fixed accuracy $\mathcal{H}$-arithmetic, so we choose the blockwise rank in each truncation with respect to the given accuracy ϵ. The costs of the $\mathcal{H}$-arithmetic depends on the maximal blockwise rank k. Thus, in fixed accuracy $\mathcal{H}$-arithmetic the cost depends on $\log \epsilon$.

Further the cost depends on the constants $n_{\min}$, C_{id} and C_{sp}. The sparsity constant C_{sp} of an $\mathcal{H}$-matrix is defined as

$$C_{sp} := \max \left\{ \max_{r\in T_I} |\{ s \in T_I | \, r \times s \in T_{I\times I}\}| , \max_{s\in T_I} |\{ r \in T_I | \, r \times s \in T_{I\times I}\}| \right\}. \tag{8}$$

For many problems the sparsity constant C_{sp} is independent of the dimension n [Gra01]. The sparsity constant is necessary for the adaptive inverse we recall in the next subsection.

3.1. The Approximate Inverse of a Hierarchical Matrix.

In the next section we use the $\mathcal{H}$-inverse as preconditioner in the PINVIT for $\mathcal{H}$-matrices. Therefore we summarize some known facts about the $\mathcal{H}$-inverse here. The inversion of hierarchical matrices is treated already in the first $\mathcal{H}$-matrix paper [Hac99]. In many subsequent publications the $\mathcal{H}$-inversion is a subject of constant investigation, among others in [Gra01, Lin02, BGH03, GH03, HKK04, Beb08, Hac09].

Algorithm 1: Adaptive Inversion of an $\mathcal{H}$-Matrix [Gra01]

Input: $M \in \mathcal{H}(T_{I\times I})$, $\tilde{c} \in \mathbb{R}$

Output: $M^{-1}_{\text{adap.},\mathcal{H}}$, with $\left\|\mathcal{I} - M^{-1}_{\text{adap.},\mathcal{H}} M\right\|_2 < \tilde{c}$

```
Compute C_sp(M) and ‖M‖_2^H;
Compute M_H^{-1} with ε_local = c̃/(C_sp(M) ‖M‖_2^H);
δ_M^0 := ‖I − M_H^{-1} M‖_2^H / c̃;
while δ_M^i > 1 do
    Compute M^{-1}_{adap.,H} with ε_local := ε_local/δ_M^i or ε_local := ε_local/ max_i δ_M^i;
    δ_M^i := ‖I − M_H^{-1} M‖_2^H / c̃;
end
```

The preferred method for the inversion is the recursive block Gaussian elimination. The blockwise computations of the recursive block Gaussian elimination are done with the $\mathcal{H}$-accuracy ϵ. Unfortunately this does not guarantee a certain accuracy of the $\mathcal{H}$-inverse $M^{-1}_{\mathcal{H}}$. Grasedyck designed Algorithm 1 [Gra01], that solves this problem by estimating the required $\mathcal{H}$-accuracy ϵ to get an approximate inverse $M^{-1}_{\text{adap.},\mathcal{H}}$, that fulfills

$$\left\|\mathcal{I} - M^{-1}_{\text{adap.},\mathcal{H}} M\right\|_2^{\mathcal{H}} < \tilde{c}$$

for a prescribed tolerance $\tilde{c}$. If $LL^T = M$ is the Cholesky decomposition of M, then we have

$$\left\|I - T^{-1}M\right\|_M = \left\|L\left(I - T^{-1}M\right)\right\|_2 \leq \|L\|_2 \left\|I - T^{-1}M\right\|_2 = \sqrt{\|M\|_2}\left\|I - T^{-1}M\right\|_2.$$

Thus, we have to choose

$$\tilde{c} = \frac{c}{\sqrt{\|M\|_2}} \text{ and } c < 1,$$

to get an inverse fulfilling Condition (5).

The complexity of this process and the rank of the result is in general unknown. There are only some results for special matrices. In [BH03] it is shown, that the inverse of FEM matrices of uniformly elliptic operators with L^∞-coefficients can be approximated by an $\mathcal{H}$-matrix if the triangularization is regular. Bebendorf could improve the result in [Beb05, Beb08] to the following theorem.

Theorem 3.1. *Let M be a non-singular FEM matrix. Then under certain assumptions regarding the grid there is a matrix $C_{\mathcal{H}} \in \mathcal{H}(T_{I\times I}, k)$ with $k \sim (|\log\epsilon| / |\log\ell|)^m$ such that*

$$\left\|M^{-1} - C_{\mathcal{H}}\right\|_2 \leq \text{cond}_2(M)\,\epsilon\left\|M^{-1}\right\|_2,$$

where

$$\ell = 1 - \frac{1}{\text{cond}_2(M)^2}.$$

Further, if M is symmetric positive definite, then the bound is tightened to

$$\left\|M^{-1} - C_{\mathcal{H}}\right\|_2 \leq \epsilon\left\|M^{-1}\right\|_2,$$

and ℓ becomes smaller:

$$\ell = \frac{\sqrt{\text{cond}_2(M)} - 1}{\sqrt{\text{cond}_2(M)} + 1}.$$

Proof. See [Beb08, Theorem 4.15/4.16]. □

Algorithm 2: $\mathcal{H}$-Cholesky Decomposition [Hac09]

Input: $M \in \mathcal{H}(T_{I\times I})$, $r = I$
Output: $L \in \mathcal{H}(T_{I\times I})$, with $LL^T = M$

Function $\mathcal{H}$-`Cholesky-decomposition(`$M|_{r\times r}, r$`)`
if $r \times r \in \mathcal{L}(T_{I\times I})$ **then**
  $L|_{r\times r}$:=`Cholesky-decomposition`$\left(M|_{r\times r}\right)$; `/* `$M|_{r\times r}$ `is dense, use standard Cholesky decomposition */`
else
  Assume $\{s_1, s_2\} = S(r)$; `/* `T_I `is a binary tree */`
  $L|_{s_1\times s_1}$:=$\mathcal{H}$-`Cholesky-decomposition`$\left(M|_{s_1\times s_1}, s_1\right)$;
  Compute $L|_{s_2\times s_1}$, so that $L|_{s_2\times s_1}\, L|_{s_1\times s_1} = M|_{s_2\times s_1}$;
  $L|_{s_1\times s_1}$:=$\mathcal{H}$-`Cholesky-decomposition`$\left(M|_{s_2\times s_2} - L|_{s_2\times s_1}\, L|_{s_2\times s_1}^T, s_2\right)$;
end
return $L_{r\times r}$;

Now we know that there is under certain assumptions an $\mathcal{H}$-matrix $C_{\mathcal{H}}$ approximating the inverse of the matrix M. Numerical examples suggest, that the $\mathcal{H}$-inversion algorithm based on the recursive block Gaussian elimination is often good enough, that means we get after linear-polylogarithmic flops an $\mathcal{H}$-matrix of linear-polylogarithmic storage complexity near $C_{\mathcal{H}}$. But we can not expect to get the optimal matrix $C_{\mathcal{H}}$ from the previous theorem.

The evaluation costs for the preconditioner $M^{-1}_{\text{adap.},\mathcal{H}}$ we get from Algorithm 1 depends on the ranks of the admissible blocks. These ranks depend on the one hand on the used $\mathcal{H}$-accuracy ϵ and so on c. On the other hand, the ranks depend on the properties of M, especially on the maximal block rank k and the norms $\|M\|_2$ and $\left\|M^{-1}\right\|_2$. Both is quite natural: a better preconditioner is more expensive and a preconditioner for a matrix of better condition is cheaper.

How to find the optimal spectral equivalent preconditioner in the $\mathcal{H}$-format with the minimal blockwise rank to a given (symmetric positive definite) $\mathcal{H}$-matrix M is still an open problem [Hac09].

Experiments show that the $\mathcal{H}$-inversion is expensive, though it is of linear-polylogarithmic complexity. One alternative is the $\mathcal{H}$-Cholesky decomposition, which we investigate in the next subsection. In Section 6 we will see that the adaptive Cholesky decomposition is much cheaper than the adaptive $\mathcal{H}$-inversion.

3.2. **The Approximate Cholesky Decomposition of a Hierarchical Matrix.** For the preconditioned inverse iteration we need a preconditioner for the symmetric positive definite matrix M. The Cholesky decomposition together with a solver for upper and lower triangular systems of equations provides us such a preconditioner.

The Cholesky decomposition of hierarchical matrices is computed by a block recursive algorithm, see Algorithm 2 or [Lin02, Hac09].

The $\mathcal{H}$-Cholesky decomposition computes a preconditioner of a certain accuracy $\tilde{c}$ measured by

$$\left\|I - L^{-1}L^{-T}M\right\|_2 \leq \tilde{c},$$

where L^{-1} and L^{-T} are the operators for the forward resp. backward solution process of the matrix equation $LX = M$. But like for the $\mathcal{H}$-inversion this accuracy $\tilde{c}$ depends on C_{sp}, $\|M\|$, cond (M) and the $\mathcal{H}$-arithmetic accuracy ϵ. So a priori we do not know how small we must choose ϵ to get a preconditioner of the necessary accuracy for the preconditioned inverse iteration.

An adaptive Cholesky decomposition of an $\mathcal{H}$-matrix can be computed similar to the adaptive $\mathcal{H}$-inversion in Algorithm 1. We use the same estimate $\epsilon_{\text{local}} = \tilde{c}/(C_{sp}(M)\,\|M\|_2^{\mathcal{H}})$ for the $\mathcal{H}$-arithmetic accuracy. We compute the $\mathcal{H}$-Cholesky decomposition with ϵ_{local} and compute

$$\eta(\epsilon_{\text{local}}) = \left\|I - L^{-T}L^{-1}M\right\|_2.$$

Algorithm 3: Adaptive Cholesky Decomposition of an $\mathcal{H}$-Matrix [Gra01]

Input: $M \in \mathcal{H}(T_{I\times I})$, $\tilde{c} \in \mathbb{R}$

Output: $L_{\text{adap.},\mathcal{H}}$, with $\left\| \mathcal{I} - L^{-T}_{\text{adap.},\mathcal{H}} L^{-1}_{\text{adap.},\mathcal{H}} M \right\|_2 < \tilde{c}$

Compute $C_{sp}(M)$ and $\|M\|_2^{\mathcal{H}}$;
Compute $L_{\mathcal{H}} = \mathcal{H}\texttt{-Cholesky-decomposition}(M, I)$ with $\epsilon_{\text{local}} = \tilde{c}/(C_{sp}(M)\,\|M\|_2^{\mathcal{H}})$;
$\delta_0 := \left\|\mathcal{I} - L_{\mathcal{H}}^{-T} L_{\mathcal{H}}^{-1} M\right\|_2^{\mathcal{H}} / \tilde{c}$;
while $\delta_i > 1$ **do**
  $\epsilon_{\text{local}} := \epsilon_{\text{local}}/\delta_M^i$ or $\epsilon_{\text{local}} := \epsilon_{\text{local}}/\max_i \delta_M^i$;
  Compute $L_{\mathcal{H}} = \mathcal{H}\texttt{-Cholesky-decomposition}(M, I)$ with ϵ_{local};
  $\delta_i := \left\|\mathcal{I} - L_{\mathcal{H}}^{-T} L_{\mathcal{H}}^{-1} M\right\|_2^{\mathcal{H}} / \tilde{c}$;
end

Further we assume that the accuracy of the preconditioner η depends linearly on the $\mathcal{H}$-arithmetic accuracy ϵ:

$$\eta(\epsilon_{\text{local}}) = \gamma \epsilon_{\text{local}}.$$

With the pair ϵ_{local} and $\eta(\epsilon_{\text{local}})$ we make a guess for γ. The quotient $\tilde{c}/\gamma$ gives us a new estimate for the $\mathcal{H}$-arithmetic accuracy. Unfortunately the dependency between ϵ and η is not linear, so there is no guarantee, that the second $\mathcal{H}$-Cholesky decomposition is exact enough. Maybe we have to repeat the process until the preconditioner fulfills Equation (5). All these steps are summarized in Algorithm 3.

In the next section we apply PINVIT from the previous section to $\mathcal{H}$-matrices using the $\mathcal{H}$-inversion or the $\mathcal{H}$-Cholesky decomposition.

4. PINVIT for $\mathcal{H}$-Matrices

In this section we will investigate what happens if we use PINVIT to compute the eigenvalues of an $\mathcal{H}$-matrix. For generality we use the subspace version of PINVIT, see [Ney02] for details. We assume that the dimension d of the subspace is small compared to n and especially small enough to store rectangular matrices of dimension $n \times d$ in the dense format. We will use Algorithm 4, which is slightly different from the one in [Ney01a], where Neymeyr uses the Ritz vectors and Ritz values instead of X and $\mu(X)$, this would lead to the following steps replacing the computation of the residuum in the last line of the for-loop in Algorithm 4:

$$\begin{aligned} QDQ^T &:= \texttt{eigendecomposition}(\mu); \\ X_i &:= X_i Q; \\ R &:= MX_i - X_i D. \end{aligned}$$

These changes lead to the following update equation

$$\begin{aligned} X_i &= X_{i-1}Q - (M)_{PC}^{-1}\left(MX_{i-1}Q - X_{i-1}QD\right), \\ &= X_{i-1}Q - (M)_{PC}^{-1}\left(MX_{i-1} - X_{i-1}QQ^T\mu\right)Q, \end{aligned} \tag{9}$$

where $(M)_{PC}^{-1}$ has to be substituted by $L^{-T}L^{-1}$ in case of using the Cholesky decomposition as preconditioner. Since Q is orthogonal and square, it holds, that $QQ^T = \mathcal{I}$. The subspace version of Equation (4) and Equation (9) only differ in the factor Q, which is multiplied from the right hand side. This means, that the subspaces spanned by X_i are identical. The orthogonal Q only causes an orthogonal transformation of the spanning vectors within the spanned subspaces. Numerical tests do not show an advantage of the diagonalized version, since the extra solutions of the eigenvalue problems produce additional costs.

Algorithm 4 is equivalent to the algorithm SPINVIT in [Ney02] or [Ney01c], so the convergence analysis in the literature can be used.

Algorithm 4: Hierarchical Subspace Preconditioned Inverse Iteration

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

1 Orthogonalize X_0;
2 $\mu := X_0^T M X_0$;
3 $R := MX_0 - X_0\mu$;
4 $T^{-1} = (M)_{\mathcal{H}}^{-1}$; or $L =$`adaptive-`$\mathcal{H}$`-Cholesky-decomposition(`M`)`;
5 **for** ($i := 1$; $\|R\|_F > \epsilon$; $i++$) **do**
6 $\quad X_i := X_{i-1} - T^{-1}R$; or $X_i := X_{i-1} - L^{-T}L^{-1}R$;
7 $\quad$ Orthogonalize X_i;
8 $\quad \mu := X_i^T M X_i$;
9 $\quad R := MX_i - X_i\mu$;
10 **end**

Remark 4.1. *PINVIT can be applied to generalized eigenvalue problems*

$$Mx = \lambda Nx,$$

too. Then we have to orthogonalize in lines 1 and 7 with respect to N, so that $X^T N X = \mathcal{I}$. Furthermore, the computations of the residuum in line 3 and 9 have to be changed to

$$R := MX_i - NX_i\mu.$$

The main advantage of $\mathcal{H}$-arithmetic is, that most algorithms are of almost linear complexity. The PINVIT for $\mathcal{H}$-matrices is one of these algorithms. The computation of the preconditioner has to be done once and is of linear-polylogarithmic complexity. The non-adaptive $\mathcal{H}$-inversion as well as the $\mathcal{H}$-Cholesky decomposition have a complexity of $\mathcal{O}(k^2 n \log^2 n)$.

Further we need one matrix-vector product with M and one evaluation of the preconditioner, one matrix-vector product with $M_{\mathcal{H}}^{-1}$ or two solutions with the triangular L,L^T, per iteration step. Both products have a complexity equal to the storage complexity of M resp. $M_{\mathcal{H}}^{-1}$. The handling of X, R and μ require some dense arithmetic on $n \times d$ matrices with $\mathcal{O}(nd^2)$ flops. So one iteration has a lower complexity than the $\mathcal{H}$-inversion. Since the number of iterations is independent of the matrix dimension, the dominant computation in the whole process is the inversion of M.

Theorem 4.2. (Convergence Theorem)
Let $x \in \mathbb{R}^n$ and $\lambda_j < \mu(x) < \lambda_{j+1}$. If $M_{\mathcal{H}}^{-1}$ is computed by Algorithm 1 or $L_{\mathcal{H}}$ is computed by Algorithm 3 and $\tilde{c} = \frac{c}{\|M\|_2}$ with $c < 1$, then the Rayleigh quotient of the next iterate x' computed by Algorithm 4 is either $\mu(x') < \lambda_j$ or $\lambda_j \leq \mu(x') < \mu(x)$ with

$$\frac{\mu(x') - \lambda_j}{\lambda_{j+1} - \mu(x')} \leq \gamma^2 \frac{\mu(x) - \lambda_j}{\lambda_{j+1} - \mu(x)},$$

where

$$\gamma = 1 - (1 - c)\left(1 - \frac{\lambda_j}{\lambda_{j+1}}\right)$$

is the convergence factor.

Proof. In the description of Algorithm 1 we show that $M_{\mathcal{H}}^{-1}$ for $\tilde{c} = \frac{c}{\|M\|_2}$ fulfills

$$\left\|I - M_{\mathcal{H}}^{-1} M\right\|_M < c.$$

So we can apply Theorem 2.1. □

4.1. **Stopping Criterion.** If the subspace spanned by X converges to an invariant subspace, then the residuum $\|R\|$ converges to zero. Since we already compute R for the next iterate,

$$\|R\|_F < \epsilon \tag{10}$$

is a cheap stopping criterion. In [Wil65, p. 173f] the following fact is shown for vectors x.

Lemma 4.3. *Let x be a normalized approximation to the eigenvector v_j corresponding to the eigenvalue λ_j and let $\mu = x^T Mx$ be the Rayleigh quotient of x. Further we need a constant δ, with*

$$\delta < |\lambda_i - \mu|, \quad \forall i \in \{1, \dots, n\} \setminus \{j\}.$$

If $\|Mx - \mu x\|_2 = \epsilon < \delta$, then

$$|\mu - \lambda_j| < \frac{\frac{\epsilon^2}{\delta}}{\left(1 - \frac{\epsilon^2}{\delta^2}\right)}. \tag{11}$$

Proof. [Wil65, p. 173f] □

Relation (11) shows that the Rayleigh quotient has an error of order ϵ^2, if $\delta \gg \epsilon$.

In this section we have described the application of PINVIT to hierarchical matrices. This leads to an algorithm of almost linear complexity for the computation of the smallest eigenvalue. In the next section we will extend this procedure to compute interior eigenvalues.

5. The Interior of the Spectrum

Often we are interested not only in the smallest eigenvalues, but in eigenvalues in the interior of the spectrum. The $(n-j)$th eigenvalue, for j small, can be computed by the subspace preconditioned inverse iteration. But if j is large, say $j \gg \log n$, this is prohibitively expensive.

Instead, we will use the folded spectrum method [WZ94], that is also mentioned in [Mor91]. First we have to choose a shift σ. Then we compute the smallest eigenpair (λ_σ, v) of $M_\sigma = (M - \sigma I)^2$, here by PINVIT. The eigenvector v is the eigenvector of M to the eigenvalue next to σ. The Rayleigh quotient $\mu(v, M) = v^T Mv/(v^T v)$ is the searched eigenvalue λ.

The following Lemmata are well known:

Lemma 5.1. *The condition number of the eigenvalue problem for $M_\sigma = (M - \sigma I)^2$ is 1.*

Proof. M_σ is symmetric. Symmetric matrices can be diagonalized by orthogonal matrices [GV96, p. 393]. □

Lemma 5.2. *Let $M \in \mathbb{R}^{n\times n}$ be symmetric positive definite and let v be an eigenvector of M, with $Mv = v\lambda$. Then v is an eigenvector of M_σ, too. Further, the corresponding eigenvalue of M_σ is*

$$\lambda_\sigma := (\lambda - \sigma)^2.$$

If λ_σ is a simple eigenvector of M_σ, then $\sigma + \sqrt{\lambda_\sigma}$ or $\sigma - \sqrt{\lambda_\sigma}$ is an eigenvalue of M.

Proof.

$$M_\sigma v = (M - \sigma I)^2 v = (M - \sigma I)(Mv - \sigma v) = (M - \sigma I)v(\lambda - \sigma) = v(\lambda - \sigma)^2$$

Since M is symmetric positive definite, there are n disjoint eigenvectors v_i, for $i = 1, \dots, n$. Each eigenvector v_i is an eigenvector of the matrix M_σ. Let v_j be the eigenvector corresponding to λ_σ, then (λ, v_j) is an eigenpair of M and λ and λ_σ fulfill the relation:

$$(\lambda - \sigma)^2 = \lambda_\sigma.$$

□

Now we use again $\mathcal{H}$-arithmetic, since the shifted and squared matrix

$$\tilde{M}_\sigma = (M -_{\mathcal{H}} \sigma I) *_{\mathcal{H}} (M -_{\mathcal{H}} \sigma I) \approx M_\sigma$$

is an $\mathcal{H}$-matrix like M. The $\mathcal{H}$-inversion of $\tilde{M}_\sigma$ is an approximate inverse of M_σ, so that we can use $(\tilde{M}_\sigma)^{-1}_{\mathcal{H}}$ as preconditioner. Also the $\mathcal{H}$-Cholesky decomposition of $\tilde{M}_\sigma$ is an approximation to the Cholesky decomposition of M_σ and can be used as preconditioner, too. The additional truncation

Algorithm 5: Inner Eigenvalues by Folded Spectrum Method and Hierarchical Subspace Preconditioned Inverse Iteration

Input: $M \in \mathbb{R}^{n\times n}$, shift σ
Output: $X_p \in \mathbb{R}^{n\times d}$, $\mu \in \mathbb{R}^{r\times r}$, with $\|MX_p - X_p\mu\| \leq \epsilon$, $\Lambda(\mu)$ are approximations to the to σ nearest eigenvalues

Orthogonalize X_0;
$T^{-1} = ((M -_{\mathcal{H}} \sigma\mathcal{I}) *_{\mathcal{H}} (M -_{\mathcal{H}} \sigma\mathcal{I}))^{-1}_{\mathcal{H}}$;
$Y_0 := MX_0 - \sigma X_0$;
$\mu_0 := Y_0^T Y_0$;
$R := MY_0 - \sigma Y_0 - X_0\mu_\sigma$;
for $(i := 1;\ \|R\|_F > \epsilon;\ i++)$ **do**
  $X_i := X_{i-1} - T^{-1}R$;
  Orthogonalize X_i;
  $Y_i := MX_i - \sigma X_i$;
  $\mu_i := Y_i^T Y_i$;
  $R := MY_i - \sigma Y_i - X_i\mu_\sigma$;
end
$\mu := X_p^T M X_p$;

in the computation of $\tilde{M}_\sigma$ does not disturb the computed eigenvalues, since the preconditioner needs not to be an exact inverse. The product $(M - \sigma I)^2 x_i$ in the computation of the residuum need a more accurate evaluation by the formula $(M - \sigma I)\,((M - \sigma I)x_i)$ instead of $\tilde{M}_\sigma x$.

Here $\mathcal{H}$-arithmetic is particularly advantageous, since $\mathcal{H}$-arithmetic enables us to compute an approximation to M_σ^{-1} or an approximate Cholesky decomposition with reasonable costs.

The smallest eigenvalue of M_σ is the square of the absolute smallest eigenvalue of $M - \sigma\mathcal{I}$. So the smallest eigenvalue of M_σ is smaller than the smallest eigenvalue of $M - \sigma\mathcal{I}$. The squaring also increases the largest eigenvalue, and so the condition number of the linear system of equations with coefficients M_σ is increased by the shifting and squaring. A higher condition number leads to higher ranks in the admissible submatrices of the $\mathcal{H}$-inverse of M_σ, and this will increase the costs of the inversion and the application of the preconditioner. This is the main disadvantage of the folded spectrum method.

Further, if $\sigma + x$ and $\sigma - x$ are eigenvalues of M, then M_σ has a double eigenvalue and the folded spectrum method may fail to converge [Mor91]. Therefore, we should avoid shifts near the middle of two eigenvalues.

The combination of the folded spectrum method and the preconditioned inverse iteration forms Algorithm 5.

Theorem 5.3. *The for-loop of Algorithm 5 is the preconditioned inverse iteration applied to M_σ. If $M = M^T$ and $\sigma \notin \Lambda(M)$, then Theorem 2.1 holds for this algorithm.*

Proof. We have M symmetric and so $\lambda_i \in \mathbb{R}$. The eigenvalues λ_σ^i of $M_\sigma = (M - \sigma I)^2$ are $\lambda_\sigma^i = (\lambda_i - \sigma)^2$. Since $\sigma \neq \lambda_i$ we have $\lambda_\sigma^i > 0$ for all i. Hence, M_σ is symmetric positive definite.

The for-loop in Algorithm 5 is identical to the for-loop in Algorithm 4 except that M is replaced by M_σ. This means, that Algorithm 5 does a preconditioned inverse iteration. So Theorem 2.1 can be used here, too. □

In the middle of the next section numerical examples confirm, that this algorithm works well.

6. Numerical Results

In this section we will show some numerical results confirming that Algorithm 4 computes the smallest eigenvalues in almost linear complexity. We measure the required CPU time for the computation of the eigenvalues of matrices of different size. This time depends on the performance of the used computing platform, so all computations are performed on an Intel®Core™i7 CPU

Preconditioner: $\mathcal{H}$-inversion						
Name	n	abs. Err.	rel. Err.	t in s	t_i/t_{i-1}	$N(n_i)/N(n_{i-1})$
FEM8	64	5.6146E-010	7.4069E-012	0.01		
FEM16	256	4.5918E-010	5.8823E-012	0.02	2.00	106.67
FEM32	1 024	3.7550E-010	4.7702E-012	0.12	6.17	27.08
FEM64	4 096	3.8009E-010	4.8177E-012	0.82	6.68	8.06
FEM128	16 384	4.4099E-010	5.5867E-012	5.84	7.09	5.44
FEM256	65 536	3.9651E-010	5.0311E-012	34.47	5.91	5.22
FEM512	262 144	3.7877E-010	6.7432E-012	194.00	5.63	6.51
Preconditioner: $\mathcal{H}$-Cholesky decomposition						
Name	n	abs. Err.	rel. Err.	t in s	t_i/t_{i-1}	$N(n_i)/N(n_{i-1})$
FEM8	64	4.6920E-010	6.1898E-012	0.01		
FEM16	256	4.7963E-010	6.1442E-012	0.02	2.00	106.67
FEM32	1 024	3.4696E-010	4.4076E-012	0.08	4.00	27.08
FEM64	4 096	4.6414E-010	5.8832E-012	0.48	6.00	8.06
FEM128	16 384	3.3206E-010	4.2071E-012	3.20	6.67	5.44
FEM256	65 536	3.8468E-010	4.8815E-012	13.90	4.34	5.22
FEM512	262 144	3.1353E-010	6.1639E-012	62.40	4.49	6.51

TABLE 1. Numerical results FEM-series, PINVIT$(1, 4)$, $c = 0.2$, $\epsilon = 10^{-4}$.

920 with 12 GB RAM. The convergence rate of the algorithm is determined by the gap between the largest eigenvalue in the computed invariant subspace and the smallest not in the subspace, see Theorem 2.1, Equation (6). We choose the size of the subspace and the shift, for the last test, so that the gap, see Equation (7), converges for n to infinity to a fixed number smaller than one. Another factor of big influence is the complexity of the inversion. We will measure and investigate the time for the inversion separately.

We choose the start vectors/matrices randomly. The computation of the preconditioner is the only operation with truncation and since we use the adaptive inversion/adaptive Cholesky decomposition algorithm we do not need to fix an accuracy for the $\mathcal{H}$-arithmetic. We stop the iteration if the residuum is smaller than 10^{-4}, then the eigenvalues have under special conditions an accuracy of about 10^{-8}, see Lemma 4.3.

The influence of the sparsity and the idempotency constant, that are slightly increasing in our example series, will be accounted for. We expect, that the required CPU time grows asymptotically like

$$N(n_i) = n_i \left(\log_2 n_i\right)^2 C_{sp} C_{id}. \tag{12}$$

6.1. **FEM matrices.** Our first example series is the FEM discretization of the 2D Laplacian over the unit square. This example is part of the HLib [HLi09], that we use for $\mathcal{H}$-arithmetic operations in our computations. We vary the discretization from 8 inner discretization points in each direction to 512 points. The results are shown in Table 1. The absolute resp. relative error is the 2-norm of the vector with the absolute resp. relative error of each eigenvalue. Figure 1 compares the CPU time for PINVIT$(k, 4)$, $k = 1$ or 3 with $N(n_i)$. The results confirm our expectations. We should mention that the FEM matrices are sparse and sparse solvers like `eigs` in MATLAB® are faster. The preconditioned inverse iteration for $\mathcal{H}$-matrices is only competitive if we exclude the $\mathcal{H}$-inversion from the time measurements, that we have done in Table 1 and Figure 1. The inverse of a sparse matrices is in general not sparse. Here we know, that the inverses of the FEM matrices are still data-sparse in the $\mathcal{H}$-matrix sense.

Using the adaptive Cholesky decomposition as preconditioner reduces the costs of the algorithm, see Table 1. But even though the computation of the preconditioner is much cheaper, see Table 6, the whole algorithm is still about 20 times slower than MATLAB eigs. If we would use the adaptive inverse as preconditioner, then the whole algorithm is about 2000 times slower.

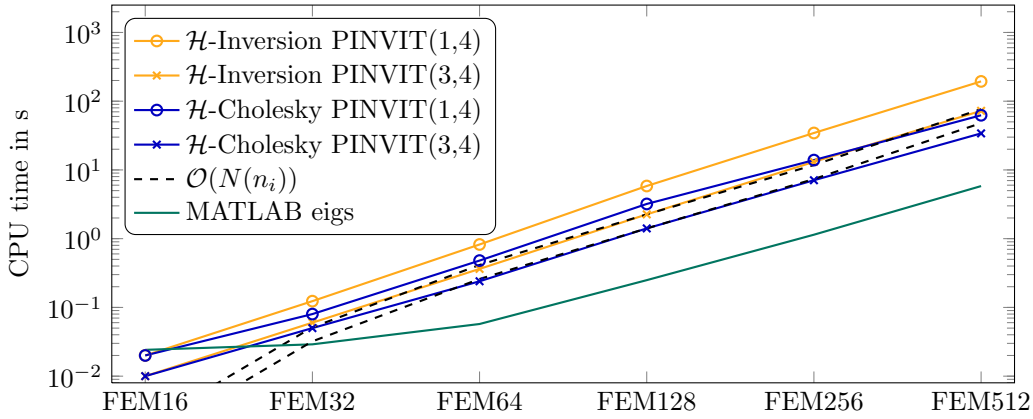

FIGURE 1. CPU time FEM-series, $c = 0.2$.

Preconditioner: $\mathcal{H}$-inversion						
Name	n	abs. Err.	rel. Err.	t in s	t_i/t_{i-1}	$N(n_i)/N(n_{i-1})$
BEM8	258	8.0358E-008	1.4669E-006	0.04		
BEM16	1 026	1.7211E-007	6.4798E-006	0.28	7.00	24.18
BEM32	4 098	8.6711E-007	6.3498E-005	1.95	6.96	24.22
BEM64	16 386	2.7006E-006	3.9930E-004	17.90	9.18	23.99
BEM128	65 538	–	–	186.00	10.39	14.66
Preconditioner: $\mathcal{H}$-Cholesky decomposition						
Name	n	abs. Err.	rel. Err.	t in s	t_i/t_{i-1}	$N(n_i)/N(n_{i-1})$
BEM8	258	7.9511E-008	1.4516E-006	0.01		
BEM16	1 026	2.1351E-007	8.0390E-006	0.14	14.00	24.18
BEM32	4 098	5.0143E-007	3.6719E-005	0.97	6.93	24.22
BEM64	16 386	2.1985E-006	3.2560E-004	7.04	7.26	23.99
BEM128	65 538	–	–	47.00	6.68	14.66

TABLE 2. Numerical results BEM-series, PINVIT$(1, 6)$, $c = 0.2$, BEM128 is too large for computing the exact eigenvalues using Lapack [ABB+99].

6.2. **BEM matrices.** As second example series we use a BEM discretization of the Poisson problem on the 3D unit sphere, again an example of the HLib. The results, see Table 2 and Figure 2, confirm again our expectation. Since the BEM matrices are dense, sparse solvers can not be used here. The usage of dense solvers is no alternative, since the matrix BEM128 would need about 32 GB storage. The MATLAB functions eigs is the fastest MATLAB build-in function for this problem, so that we compare here with eigs, too. We are not able to read in the matrix BEM64 into MATLAB due to the limitation of the size of m-files to 2 GB.

6.3. **Inner eigenvalues.** In this subsection we use the same matrices again. We shift them to demonstrate Algorithm 5. We choose the shift and the subspace dimension, so that we have a large gap between the computed and the other eigenvalues. The gap is almost constant for each of both example series. The results are shown in Tables 3 and 4. There was not enough memory for the $\mathcal{H}$-inversion of the shifted FEM512 matrix and not enough fast memory for the $\mathcal{H}$-Cholesky decomposition of the shifted FEM512 matrix.

6.4. **Adaptive $\mathcal{H}$-Inversion vs. Adaptive $\mathcal{H}$-Cholesky Decomposition.** Finally we have to investigate the computation of the preconditioner, which we excluded from the previous investigations. In Table 5 and 6 one can see, that for the FEM-matrices the storage and the computational complexity does not fulfill the expectations. Since the costs are still much lower than for dense

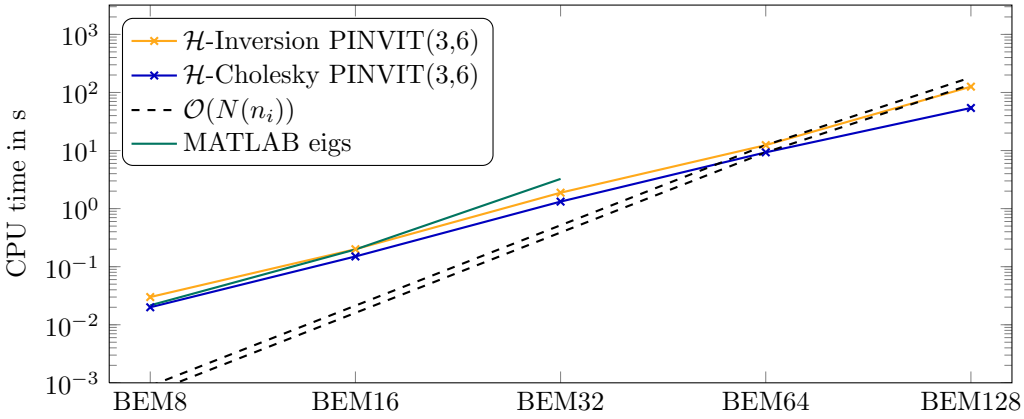

FIGURE 2. CPU time BEM-series, $c = 0.2$.

Preconditioner: $\mathcal{H}$-inversion						
Name	n	abs. Err.	rel. Err.	t in s	t_i/t_{i-1}	$N(n_i)/N(n_{i-1})$
FEM8	64	1.7219E-013	8.2738E-016	<0.010		
FEM16	256	7.2388E-012	4.1801E-014	0.035		106.67
FEM32	1 024	6.0045E-013	3.2162E-015	0.125	3.57	27.08
FEM64	4 096	1.0915E-012	5.8548E-015	0.935	7.48	8.06
FEM128	16 384	1.5655E-012	8.0577E-015	6.080	6.50	5.44
FEM256	65 536	1.1976E-011	6.3174E-014	52.400	8.62	5.22
Preconditioner: $\mathcal{H}$-Cholesky decomposition						
Name	n	abs. Err.	rel. Err.	t in s	t_i/t_{i-1}	$N(n_i)/N(n_{i-1})$
FEM8	64	1.5306E-013	7.5852E-016	<0.010		
FEM16	256	9.8071E-012	5.6638E-014	0.020		106.67
FEM32	1 024	1.0577E-012	5.8753E-015	0.050	2.50	27.08
FEM64	4 096	1.5776E-012	8.5785E-015	0.310	6.20	8.06
FEM128	16 384	2.1416E-012	1.0937E-014	1.640	5.29	5.44
FEM256	65 536	6.7400E-012	3.6758E-014	9.730	5.93	5.22
FEM512	262 144	1.5002E-010	7.9036E-013	545.000	56.01	6.51

TABLE 3. Numerical results for shifted FEM-series, PINVIT$(1, 3)$, $c = 0.2$, $\epsilon = 10^{-4}$.

Preconditioner: $\mathcal{H}$-inversion						
Name	n	abs. Err.	rel. Err.	t in s	t_i/t_{i-1}	$N(n_i)/N(n_{i-1})$
BEM8	258	9.8089E-010	3.0652E-010	0.01		
BEM16	1 026	3.0867E-005	8.4994E-006	0.31	31.00	24.18
BEM32	4 098	5.4469E-004	1.6450E-004	1.04	3.35	24.22
BEM64	16 386	5.0141E-005	1.5244E-005	11.50	11.06	23.99
BEM128	6 5538	–	–	15.90	1.38	14.66
Preconditioner: $\mathcal{H}$-Cholesky decomposition						
Name	n	abs. Err.	rel. Err.	t in s	t_i/t_{i-1}	$N(n_i)/N(n_{i-1})$
BEM8	258	1.6234E-009	5.0731E-010	0.01		
BEM16	1026	1.9052E-005	5.2460E-006	0.30	30.00	24.18
BEM32	4098	4.3829E-004	1.3237E-004	1.04	3.47	24.22
BEM64	16386	2.2523E-005	6.8473E-006	11.50	11.06	23.99

TABLE 4. Numerical results for shifted BEM-series, PINVIT$(1, 6)$, $c = 0.2$, $\epsilon = 10^{-4}$.

Name	n	$s(M^{-1})$ in kB	s_i/s_{i-1}	t in s	t_i/t_{i-1}	$N(n_i)/N(n_{i-1})$
FEM8	64	25		<0.01		
FEM16	256	246	10.02	0.03		106.67
FEM32	1 024	2 089	8.51	0.29	9.67	27.08
FEM64	4 096	16 539	7.92	5.49	18.93	8.06
FEM128	16 384	116 551	7.05	47.25	8.61	5.44
FEM256	65 536	755 422	6.48	366.14	7.75	5.22
FEM512	262 144	4 541 299	6.01	2 731.28	7.46	6.51

TABLE 5. Required storage and CPU-time for adaptive $\mathcal{H}$-inversion of different FEM-matrices, $c = 0.2$.

Name	n	$s(L)$ in kB	s_i/s_{i-1}	t in s	t_i/t_{i-1}	$N(n_i)/N(n_{i-1})$
FEM8	64	24		<0.01		
FEM16	256	175	7.13	<0.01		106.67
FEM32	1 024	1 137	6.51	0.02		27.08
FEM64	4 096	6 728	5.92	0.18	9.00	8.06
FEM128	16 384	37 177	5.53	1.28	7.11	5.44
FEM256	65 536	197 261	5.31	8.05	6.29	5.22
FEM512	262 144	1 033 787	5.24	48.98	6.08	6.51

TABLE 6. Required storage and CPU-time for adaptive $\mathcal{H}$-Cholesky decomposition of different FEM-matrices, $c = 0.2$.

Name	n	$s(M^{-1})$ in kB	s_i/s_{i-1}	t in s	t_i/t_{i-1}	$N(n_i)/N(n_{i-1})$
BEM8	258	542		0.31		
BEM16	1 026	5 439	10.04	2.03	6.55	24.18
BEM32	4 098	47 807	8.79	21.60	10.64	24.22
BEM64	16 386	470 197	9.84	529.98	24.54	23.99
BEM128	65 538	4 596 971	9.78	7 764.12	14.65	14.66

TABLE 7. Required storage and CPU-time for adaptive $\mathcal{H}$-inversion of different BEM-matrices, $c = 0.2$.

arithmetic, we will not further investigate this small deviation, that we found in the BEM-series only sporadically, see Tables 7 and 8.

The computation of the preconditioner is responsible for at least half of the cost of the whole algorithm. The only parameter directly effecting these costs is the c in Equation (5), that we choose relatively large with $c = 0.2$.

The adaptive $\mathcal{H}$-Cholesky decomposition is much cheaper than the adaptive $\mathcal{H}$-inversion, in some examples the Cholesky decomposition needs only 2% of the time of the inversion. The Cholesky factors require less storage than the $\mathcal{H}$-inverse. That is beneficial in two ways. On the one hand less storage means, that we can handle larger problems, e. g. shifted FEM512. And on the other hand the effort for the application of the preconditioner is reduced. This seem to be the main explanation for the lower CPU time we see in Table 1, 2, and 3, since the number of iterations are similar.

To summarize the recommendation is to use the adaptive $\mathcal{H}$-Cholesky decomposition instead of the adaptive $\mathcal{H}$-inversion.

7. Conclusions

We have seen, that the preconditioned inverse iteration can be used to compute the smallest eigenvalues of hierarchical matrices. Further one can use the ideas of the folded spectrum

Name	n	$s(L)$ in kB	s_i/s_{i-1}	t in s	t_i/t_{i-1}	$N(n_i)/N(n_{i-1})$
BEM8	258	452		0.03		
BEM16	1 026	3 111	6.88	0.38	12.67	24.18
BEM32	4 098	21 099	6.78	3.54	9.32	24.22
BEM64	16 386	127 066	6.02	27.94	7.89	23.99
BEM128	65 538	735 170	5.79	209.38	7.49	14.66

TABLE 8. Required storage and CPU-time for adaptive $\mathcal{H}$-Cholesky decomposition of different BEM-matrices, $c = 0.2$.

method together with the efficient $\mathcal{H}$-arithmetic to compute inner eigenvalues of an $\mathcal{H}$-matrix using PINVIT. This can be used to compute inner eigenvalues of sparse matrices as well as of data-sparse matrices like boundary element matrices.

For sparse matrices the handling as $\mathcal{H}$-matrix and the computation of the eigenvalues by preconditioned inverse iteration for $\mathcal{H}$-matrices is not competitive, since especially the $\mathcal{H}$-inversion is too expensive. If we have the $\mathcal{H}$-inverse or $\mathcal{H}$-Cholesky decomposition already available, for instance we have to solve some systems $Mx = b$ as well, then the approach presented here is competitive to sparse eigensolvers.

In the case of data-sparse matrices, that are not sparse matrices, the limitation of storage limits the use of dense eigensolvers. The data-sparse $\mathcal{H}$-arithmetic together with preconditioned inverse iteration enables us to solve larger eigenproblems, that do not fit in the storage otherwise.

The electronic structure computations in computational chemistry lead to dense but data-sparse matrices. In the future we will investigate the usage of preconditioned inverse iteration for hierarchical matrices in computational chemistry.

REFERENCES

[ABB+99] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. SIAM, Philadelphia, PA, third edition, 1999.

[AMSV02] P.-A. Absil, R. Mahony, R. Sepulchre, and P. Van Dooren. A Grassmann-Rayleigh quotient iteration for computing invariant subspaces. *SIAM Rev.*, 44(1):57–73, Mar. 2002.

[Beb05] M. Bebendorf. Efficient inversion of the Galerkin matrix of general second-order elliptic operators with nonsmooth coefficients. *Math. Comp.*, 74(251):1179–1200, 2005.

[Beb08] M. Bebendorf. *Hierarchical Matrices: A Means to Efficiently Solve Elliptic Boundary Value Problems*, volume 63 of *Lecture Notes in Computational Science and Engineering (LNCSE)*. Springer Verlag, Berlin Heidelberg, 2008.

[BGH03] St. Börm, L. Grasedyck, and W. Hackbusch. Introduction to hierarchical matrices with applications. *Eng. Anal. Boundary Elements*, 27:405–422, 2003.

[BH03] M. Bebendorf and W. Hackbusch. Existence of $\mathcal{H}$-Matrix Approximants to the Inverse FE-Matrix of elliptic Operators with L^∞-Coefficients. *Numer. Math.*, 95:1–28, 2003.

[BK05] M. Bebendorf and R. Kriemann. Fast parallel solution of boundary element systems. *Comput. Visual. Sci.*, 8:121–135, 2005.

[GH03] L. Grasedyck and W. Hackbusch. Construction and arithmetics of $\mathcal{H}$-matrices. *Computing*, 70(4):295–334, 2003.

[Gra01] L. Grasedyck. *Theorie und Anwendungen Hierarchischer Matrizen*. Dissertation, University of Kiel, Kiel, Germany, 2001. In German, available at `http://e-diss.uni-kiel.de/diss_454`.

[GV96] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, third edition, 1996.

[Hac99] W. Hackbusch. A Sparse Matrix Arithmetic Based on $\mathcal{H}$-Matrices. Part I: Introduction to $\mathcal{H}$-Matrices. *Computing*, 62(2):89–108, 1999.

[Hac09] W. Hackbusch. *Hierarchische Matrizen. Algorithmen und Analysis*. Springer-Verlag, Berlin, 2009.

[HKK04] W. Hackbusch, B.N. Khoromskij, and R. Kriemann. Hierarchical matrices based on a weak admissibility criterion. *Computing*, 73:207–243, 2004.

[HLi09] $\mathcal{H}$lib 1.3. `http://www.hlib.org`, 1999–2009.

[KHS07] J. Koch, W. Hackbusch, and K. Sundmacher. $\mathcal{H}$-matrix methods for linear and quasi-linear integral operators appearing in population balances. *Comp. & Chem. Eng.*, 31(7):745–759, 2007.

[KMOX09] A.V. Knyazev, V. Mehrmann, J. Osborn, and J. Xu, editors. *Linear and Nonlinear Eigenproblems for PDEs*, volume 2009/37 of *Oberwolfach Reports*. Mathematisches Forschungsinstitut Oberwolfach, 2009.

[KN03] A.V. Knyazev and K. Neymeyr. A geometric theory for preconditioned inverse iteration III: A short and sharp convergence estimate for generalized eigenvalue problems. *Linear Algebra Appl.*, 358(1-3):95–114, Jan. 2003.

[KN09] A.V. Knyazev and K. Neymeyr. Gradient flow approach to geometric convergence analysis of preconditioned eigensolvers. *SIAM J. Matrix Anal. Appl.*, 31(2):621–628, 2009.

[Kny98] A.V. Knyazev. Preconditioned eigensolvers — an oxymoron? *Electr. Trans. Num. Anal.*, 7:104–123, 1998.

[Lin02] M. Lintner. *Lösung der 2D Wellengleichung mittels hierarchischer Matrizen*. Dissertation, Fakultät für Mathematik, TU München, `http://tumb1.biblio.tu-muenchen.de/publ/diss/ma/2002/lintner.pdf`, Jun. 2002.

[Mor91] R.B. Morgan. Computing interior eigenvalues of large matrices. *Linear Algebra Appl.*, 154–156:289–309, 1991.

[Ney01a] K. Neymeyr. A geometric theory for preconditioned inverse iteration I: Extrema of the Rayleigh quotient. *Linear Algebra Appl.*, 322(1–3):61–85, Jan. 2001.

[Ney01b] K. Neymeyr. A geometric theory for preconditioned inverse iteration II: Convergence estimates. *Linear Algebra Appl.*, 322(1-3):87–104, Jan. 2001.

[Ney01c] K. Neymeyr. *A Hierarchy of Preconditioned Eigensolvers for Elliptic Differential Operators*. Habilitationsschrift, Mathematische Fakultät der Universität Tübingen, Sep. 2001.

[Ney02] K. Neymeyr. A geometric theory for preconditioned inverse iteration applied to a subspace. *Math. Comp.*, 71(237):197–216, 2002.

[SRNB09] R. Schneider, T. Rohwedder, A. Neelov, and J. Blauert. Direct minimization for calculating invariant subspaces in density functional computations of the electronic structure. *J. Comput. Math.*, 27:360–387, 2009.

[Wil65] J.H. Wilkinson. *The Algebraic Eigenvalue Problem*. Oxford University Press, Oxford, 1965.

[WZ94] L.-W. Wang and A. Zunger. Electronic structure pseudopotential calculations of large (∼1000 atoms) Si quantum dots. *J. Phys. Chem.*, 98(98):2158–2165, 1994.