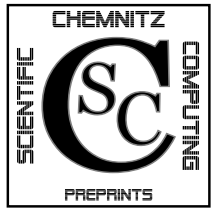
TECHNISCHE UNIVERSITÄT CHEMNITZ

Peter Benner Thomas Mach

On the QR Decomposition of $\mathcal{H}$-Matrices

CSC/09-04



Chemnitz Scientific Computing Preprints

Chemnitz Scientific Computing Preprints – ISSN 1864-0087

Impressum:

Chemnitz Scientific Computing Preprints — ISSN 1864-0087

(1995–2005: Preprintreihe des Chemnitzer SFB393)

Herausgeber:
Professuren für
Numerische und Angewandte Mathematik
an der Fakultät für Mathematik
der Technischen Universität Chemnitz

Postanschrift:
TU Chemnitz, Fakultät für Mathematik
09107 Chemnitz
Sitz:
Reichenhainer Str. 41, 09126 Chemnitz

http://www.tu-chemnitz.de/mathematik/csc/

07-02 A. Meyer. Grundgleichungen und adaptive Finite-Elemente-Simulation bei “Großen Deformationen”. Februar 2007.

07-03 P. Steinhorst. Rotationssymmetrie für piezoelektrische Probleme. Februar 2007.

07-04 S. Beuchler, T. Eibner, U. Langer. Primal and Dual Interface Concentrated Iterative Substructuring Methods. April 2007.

07-05 T. Hein, M. Meyer. Simultane Identifikation voneinander unabhängiger Materialparameter - numerische Studien. Juni 2007.

07-06 A. Bucher, U.-J. Görke, P. Steinhorst, R. Kreißig, A. Meyer. Ein Beitrag zur adaptiven gemischten Finite-Elemente-Formulierung der nahezu inkompressiblen Elastizität bei großen Verzerrungen. September 2007.

07-07 U.-J. Görke, A. Bucher, R. Kreißig Zur Numerik der inversen Aufgabe für gemischte (u/p) Formulierungen am Beispiel der nahezu inkompressiblen Elastizität bei großen Verzerrungen. October 2007.

07-08 A. Meyer, P. Steinhorst. Betrachtungen zur Spektraläquivalenz für das Schurkomplement im Bramble-Pasciak-CG bei piezoelektrischen Problemen. Oktober 2007.

07-09 T. Hein, M. Meyer. Identification of material parameters in linear elasticity - some numerical results. November 2007.

07-10 T. Hein. On solving implicitly defined inverse problems by SQP-approaches. December 2007.

08-01 P. Benner, M. Döhler, M. Pester, J. Saak. PLiCMR - Usage on CHiC. July 2008.

08-02 T. Eibner. A fast and efficient algorithm to compute BPX- and overlapping preconditioner for adaptive 3D-FEM. June 2008.

08-03 A. Meyer. Hierarchical Preconditioners and Adaptivity for Kirchhoff-Plates. September 2008.

08-04 U.-J. Görke, A. Bucher, R. Kreißig. Ein numerischer Vergleich alternativer Formulierungen des Materialmodells der anisotropen Elastoplastizität bei großen Verzerrungen. September 2008.

08-05 U.-J. Görke, R. Landgraf, R. Kreißig. Thermodynamisch konsistente Formulierung des gekoppelten Systems der Thermoelastoplastizität bei großen Verzerrungen auf der Basis eines Substrukturkonzepts. Oktober 2008.

08-06 M. Meyer, J. Müller. Identification of mechanical strains by measurements of a deformed electrical potential field. November 2008.

08-07 M. Striebel, J. Rommes. Model order reduction of nonlinear systems: status, open issues, and applications. November 2008.

08-08 P. Benner, C. Effenberger. A rational SHIRA method for the Hamiltonian eigenvalue problem. December 2008.

09-01 R. Unger. Obstacle Description with Radial Basis Functions for Contact Problems in Elasticity. January 2009.

09-02 U.-J. Görke, S. Kaiser, A. Bucher, R. Kreißig. A fast and efficient algorithm to compute BPX- and overlapping preconditioner for adaptive 3D-FEM. February 2009.

09-03 J. Glänzel. Kurzvorstellung der 3D-FEM Software SPC-PM3AdH-XX. January 2009.

The complete list of CSC and SFB393 preprints is available via
`http://www.tu-chemnitz.de/mathematik/csc/`

TECHNISCHE UNIVERSITÄT CHEMNITZ

Chemnitz Scientific Computing Preprints

Peter Benner **Thomas Mach**

On the QR Decomposition of $\mathcal{H}$-Matrices

CSC/09-04

Abstract

The hierarchical ($\mathcal{H}$-) matrix format allows storing a variety of dense matrices from certain applications in a special data-sparse way with linear-polylogarithmic complexity. Many operations from linear algebra like matrix-matrix and matrix-vector products, matrix inversion and LU decomposition can be implemented efficiently using the $\mathcal{H}$-matrix format. Due to its importance in solving many problems in numerical linear algebra like least-squares problems, it is also desirable to have an efficient QR decomposition of $\mathcal{H}$-matrices. In the past, two different approaches for this task have been suggested in [Lin02] and [Beb08]. We will review the resulting methods and suggest a new algorithm to compute the QR decomposition of an $\mathcal{H}$-matrix. Like other $\mathcal{H}$-arithmetic operations the $\mathcal{H}$QR decomposition is of linear-polylogarithmic complexity. We will compare our new algorithm with the older ones by using two series of test examples and discuss benefits and drawbacks of the new approach.

This work was supported by a grant of the Free State of Saxony (501-G-209).

Keywords: Hierarchical matrices, QR decomposition, Orthogonalisation, HQR decomposition

Mathematics Subject Classification: 65F25, 65F50, 15A23

CSC/09-04 **ISSN 1864-0087** **July 2009**

Contents

Author's addresses:

Peter Benner · Thomas Mach
TU Chemnitz
Fakultät für Mathematik
09107 Chemnitz

Some titles in this CSC preprint series and the former SFB393 preprint series:

05-01 C. Pester. A residual a posteriori error estimator for the eigenvalue problem for the Laplace-Beltrami operator. January 2005.

05-02 J. Badía, P. Benner, R. Mayo, E. Quintana-Ortí, G. Quintana-Ortí, J. Saak. Parallel Order Reduction via Balanced Truncation for Optimal Cooling of Steel Profiles. February 2005.

05-03 C. Pester. CoCoS – Computation of Corner Singularities. April 2005.

05-04 A. Meyer, P. Nestler. Mindlin-Reissner-Platte: Einige Elemente, Fehlerschätzer und Ergebnisse. April 2005.

05-05 P. Benner, J. Saak. Linear-Quadratic Regulator Design for Optimal Cooling of Steel Profiles. April 2005.

05-06 A. Meyer. A New Efficient Preconditioner for Crack Growth Problems. April 2005.

05-07 A. Meyer, P. Steinhorst. Überlegungen zur Parameterwahl im Bramble-Pasciak-CG für gemischte FEM. April 2005.

05-08 T. Eibner, J. M. Melenk. Fast algorithms for setting up the stiffness matrix in hp-FEM: a comparison. June 2005.

05-09 A. Meyer, P. Nestler. Mindlin-Reissner-Platte: Vergleich der Fehlerindikatoren in Bezug auf die Netzsteuerung Teil I. June 2005.

05-10 A. Meyer, P. Nestler. Mindlin-Reissner-Platte: Vergleich der Fehlerindikatoren in Bezug auf die Netzsteuerung Teil II. July 2005.

05-11 A. Meyer, R. Unger. Subspace-cg-techniques for clinch-problems. September 2005.

05-12 P. Ciarlet, Jr, B. Jung, S. Kaddouri, S. Labrunie, J. Zou. The Fourier Singular Complement Method for the Poisson Problem. Part III: Implementation Issues. October 2005.

05-13 T. Eibner, J. M. Melenk. Multilevel preconditioning for the boundary concentrated *hp*-FEM. December 2005.

05-14 M. Jung, A. M. Matsokin, S. V. Nepomnyaschikh, Yu. A. Tkachov. Multilevel preconditioning operators on locally modified grids. December 2005.

05-15 S. Barrachina, P. Benner, E. S. Quintana-Ortí. Solving Large-Scale Generalized Algebraic Bernoulli Equations via the Matrix Sign Function. December 2005.

05-16 B. Heinrich, B. Jung. Nitsche- and Fourier-finite-element method for the Poisson equation in axisymmetric domains with re-entrant edges. December 2005.

05-17 M. Randrianarivony, G. Brunnett. C^0-paving of closed meshes with quadrilateral patches. December 2005.

05-18 M. Randrianarivony, G. Brunnett. Quadrilateral removal and 2-ear theorems. December 2005.

05-19 P. Benner, E. S. Quintana-Ortí, G. Quintana-Ortí. Solving linear-quadratic optimal control problems on parallel computers. December 2005.

06-01 T. Eibner, J. M. Melenk. p-FEM quadrature error analysis on tetrahedra. October 2006.

06-02 P. Benner, H. Faßbender. On the solution of the rational matrix equation $X = Q + LX^{-1}L^\top$. September 2006.

06-03 P. Benner, H. Mena, J. Saak. On the Parameter Selection Problem in the Newton-ADI Iteration for Large Scale Riccati Equations. October 2006.

06-04 J. M. Badía, P. Benner, R. Mayo, E. S. Quintana-Ortí, G. Quintana-Ortí, A. Remón. Balanced Truncation Model Reduction of Large and Sparse Generalized Linear Systems. November 2006.

07-01 U. Baur, P. Benner. Gramian-Based Model Reduction for Data-Sparse Systems. February 2007.

[Gra01] L. Grasedyck. *Theorie und Anwendungen Hierarchischer Matrizen*. PhD thesis, Mathematisch-Naturwissenschaftlichen Fakultät der Christian-Albrechts-Universität zu Kiel, July 2001.

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

[Hig86] N.J. Higham. Computing the polar decomposition with applications. *SIAM J. Sci. Stat. Computing*, 7(4):1160–1174, 1986.

[Hig02] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, 2002.

[HK00] W. Hackbusch and B. N. Khoromskij. A sparse $\mathcal{H}$-matrix arithmetic. II. Application to multi-dimensional problems. *Computing*, 64(1):21–47, 2000.

[Hli09] $\mathcal{H}$lib 1.3. http://www.hlib.org, 1999-2009.

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

[Lin04] M. Lintner. The Eigenvalue Problem for the 2D Laplacian in $\mathcal{H}$-Matrix Arithmetic and Application to the Heat and Wave Equation. *Computing*, 72:293–323, 2004.

[Ste05] Olaf Steinbach. *Lösungsverfahren für lineare Gleichungssysteme. Algorithmen und Anwendungen*. Mathematik für Ingenieure und Naturwissenschaftler. Teubner, Wiesbaden, 2005.

1 Introduction

Given a real or complex $m \times n$-matrix M, the factorization

$$M = QR \tag{1}$$

is called the *QR decomposition* of M if Q is an $m \times m$ orthogonal (or unitary) matrix and R is an $m \times n$ upper triangular (or trapezoidal if $m < n$) matrix. The QR decomposition is used to solve a variety of matrix algebra problems, in particular least squares problems and linear systems of equations. It is also used in the archetype of the QR algorithm. See, e.g., [Dem97, GV96] for properties and applications of the QR decomposition, as well as algorithms for its computation. If M is a structured matrix and one is to define a set of functions for handling linear algebra problems involving such a structured matrix, it is thus desirable to have a QR decomposition in the portfolio due do its wide applicability,

The matrix structure we are interested in here is the hierarchical (short: $\mathcal{H}$-) matrix format. This allows a data-sparse representation for several types of matrices arising in various applications. Matrices that belong to this class result, for instance, from the discretization of partial differential or integral equations using the finite element or boundary element methods. Exploiting the special structure of these matrices in computational methods yields significantly reduced computing time and memory requirements. The $\mathcal{H}$-matrix format and the associated arithmetic are derived in [Gra01, GH03, Hac99, HK00] and are now also described in various textbooks and monographs, see [Beb08, Hac09, Ste05]. Regarding the above remarks on the importance of the QR decomposition, it will certainly be useful to have one for $\mathcal{H}$-matrices, too. We will call a QR decomposition an $\mathcal{H}$QR decomposition if the result is a pair of two hierarchical matrices.

In the last years many algorithms for $\mathcal{H}$-matrices were invented, e.g., the $\mathcal{H}$-LU factorisation [Beb05, BG06], the $\mathcal{H}$-Cholesky factorisation [Lin02, GKB08] and $\mathcal{H}$-inversion [Gra01]. Both factorisations are used to precondition or to solve linear systems of equations with a hierarchical coefficient matrix. Many arithmetic operations for $\mathcal{H}$-matrices have a linear-polylogarithmic complexity. An $\mathcal{H}$QR decomposition should also have such a complexity.

Beside having a complexity as expected for formatted arithmetic operations, the $\mathcal{H}$QR decomposition should be a good orthogonal decomposition, too. This means first that

$$r_{\mathrm{QR}} := \|QR - M\|_2 \tag{2}$$

should be small, and second the matrix Q should be orthogonal. We will call Q a *nearly orthogonal* matrix if

$$r_{\mathrm{orth}}(Q) := \left\|Q^T Q - I\right\|_2 \tag{3}$$

is small. Let

$$\left\|Q^T Q - I\right\|_2 \leq \epsilon \ll 1.$$

Then the norm of Q is

$$\|Q\|_2 \leq \sqrt{1+\epsilon}$$

and the condition number of Q is

$$\kappa_2(Q) = \|Q\|_2 \left\|Q^{-1}\right\|_2 \leq \sqrt{\frac{1+\epsilon}{1-\epsilon}} \approx 1 + \epsilon.$$

Two $\mathcal{H}$QR decompositions have so far been suggested in the literature, see [Beb08, Lin02]. Both have deficiencies in one of the above requirements as we will see later in Section 4. Therefore, we suggest an alternative approach that does not yet overcome all difficulties, but offers some advantages over the existing methods to compute the $\mathcal{H}$QR decomposition.

The outline of the paper is as follows. In the next section we will briefly explain the class of hierarchical matrices. The following section first provides a review of the two known methods to compute an $\mathcal{H}$QR decomposition and finally suggests an alternative approach. In Section 4 we compare the three methods using some numerical tests. Some concluding remarks will be given in the end.

2 Hierarchical Matrices

Hierarchical ($\mathcal{H}$-) matrices were introduced by W. Hackbusch in 1998 [Hac99]. Some matrices like BEM or FEM matrices have submatrices that admit low-rank approximations. The basic idea of the $\mathcal{H}$-matrix format is using a hierarchical structure to find and access such submatrices and to use good low rank approximations to reduce the storage amount and the computation time. This low rank approximations makes 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 [Gra01] or [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 $v \in T_I$ is either the disjoint union of its sons $w \in S(v)$ or a leaf of T_I.

The set of sons of a vertex $v \in T_I$ is called $S(v)$. We define the *descendants* $S^*(v)$ of a vertex $v \in T_I$ by

$$S^*(v) = \begin{cases} \{v\} & \text{if } S(v) = \emptyset, \\ \{v\} \cup \bigcup_{w \in S(v)} S^*(w) & \text{otherwise.} \end{cases}$$

We denote the set of leaves, vertices without sons $S(\cdot) = \emptyset$, of the $\mathcal{H}$-tree T_I with $\mathcal{L}(T_I)$. The $\mathcal{H}$-tree T has a depth $\operatorname{depth}(T)$, which is the maximum length of the paths from the root to each leave. If cardinality or geometrically balanced clustering is used the depth of the tree is in $\mathcal{O}(\log n)$ [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 of $T_I \times T_I$. 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 by

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

with the minimum block size $n_{\min}$.

We divide the set of leaves in admissible leaves $\mathcal{L}^+(T)$ and non-admissible leaves $\mathcal{L}^-(T)$. The submatrices corresponding to admissible leaves have at most rank k and will be stored as so called $\mathbf{R}$k-matrices AB^T, with $k = \operatorname{rank} AB^T$. The submatrices corresponding to non-admissible leaves will be stored in the standard way for dense matrices without any approximation.

If $s \times t$ is a vertex of the $\mathcal{H}_\times$-tree, then $M_{s\times t}$ is the corresponding submatrix of the $\mathcal{H}$-matrix. If necessary we renumber the indices in order to achieve continuous vertices. The vertex t is continuous if $t = \{t_{\min}, t_{\min}+1, \ldots, t_{\max}\}$. This allows us to order the vertices. We say the vertex t is smaller than s if and only if the largest index in t, $t_{\max}$, is lower than the smallest index in s, $s_{\min}$.

Our goal is to compute an $\mathcal{H}$QR decomposition of an $\mathcal{H}$-matrix. So we need to define: an upper triangular $\mathcal{H}$-matrix is an $\mathcal{H}$-matrix, where all blocks $t \times s$ with $s < t$ are zero and all diagonal blocks are upper triangular matrices.

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

4.3 The new $\mathcal{H}$QR decomposition

Algorithm 4 performs best for the FEM example series. Only Lintner's reorthogonalisation method has a higher orthogonality, but is not as accurate and about three times slower.

The results for the BEM matrices are not that good. For the large matrices the time consumption increases too fast. Using extrapolation tells us, that probably for such matrices with dimension 10^{5-6} or larger this method is the most expensive one. The two other indicators are good. The accuracy is as good as for the FEM matrices and the orthogonality is again the second best one.

Summarising the numerical tests there is no method dominating all others. Some methods are good for special matrices, like Algorithm 2 for block Hessenberg matrices or Algorithm 4 for matrices with many, large zero blocks. All but the Algorithm 1 are at least as expensive as the $\mathcal{H}$-inversion, so we must remark:

Remark 4.1. *The $\mathcal{H}$QR decomposition should not be used as a preconditioner or a solver for linear systems of equations, since either Q is not orthogonal or the $\mathcal{H}$LU / $\mathcal{H}$Cholesky factorisation or even the $\mathcal{H}$-inversion are faster.*

5 Conclusions

We have discussed three different methods to compute an approximate QR decomposition of $\mathcal{H}$-matrices. Such $\mathcal{H}$QR decompositions have been suggested previously in the literature. As the known approaches have some deficiencies either in efficiency or orthogonality of the Q-factor, we have derived a new method to compute an $\mathcal{H}$QR decomposition. The new approach is not superior in all aspects, but offers a good compromise of accuracy vs. efficiency. We have compared the three methods for two typical sets of $\mathcal{H}$-matrices and highlighted advantages and disadvantages of the three approaches using these examples. We believe the experiments show that our approach to compute $\mathcal{H}$QR decompositions presents a viable alternative to the existing ones. As none of the methods turns out to dominate the others w.r.t. overall performance, we hope the presented examples help to choose the suitable $\mathcal{H}$QR decomposition for concrete problems.

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.

[Beb05] M. Bebendorf. Hierarchical LU decomposition-based preconditioners for BEM. *Computing*, 74(3):225–247, 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.

[BG06] Sabine Le Borne and Lars Grasedyck. $\mathcal{H}$-matrix preconditioners in convection-dominated problems. *SIAM J. Matrix Anal. Appl.*, 27(4):1172–1189, 2006.

[Dem97] James W. Demmel. *Applied Numerical Linear Algebra.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

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

[GKB08] Lars Grasedyck, Ronald Kriemann, and Sabine Le Borne. Parallel black box $\mathcal{H}$-LU preconditioning for elliptic boundary value problems. *Comput. Vis. Sci.*, 11:273–291, 2008.

4 Numerical Results

We use two test example series. Both series are generated using the examples in the $\mathcal{H}$lib [Hli09]. The first example is the Finite-Element-Method discretisation of the 2D-Laplacian. These matrices have many and large rank-zero blocks. The second example is a Boundary-Element-Method discretisation for the 3D-unit-sphere. These matrices have no rank-zero blocks. We vary the dimension of the first example from 16 to 262,144 and of the second example from 66 to 65,538.

We will use the accuracy and the orthogonality of the $\mathcal{H}$QR decompositions and the needed CPU-time to compare the different algorithms. Some of our test matrices are too large to store them in a dense matrix-format, so we will use

$$r^{\mathcal{H}}_{\mathrm{QR}} := \|Q *_{\mathcal{H}} R -_{\mathcal{H}} M\|^{\mathcal{H}}_2 \qquad \text{and}$$
$$r^{\mathcal{H}}_{\mathrm{orth}}(Q) := \left\|Q^T *_{\mathcal{H}} Q -_{\mathcal{H}} I\right\|^{\mathcal{H}}_2$$

instead of the norms in (2) and (3) to measure orthogonality and accuracy of the $\mathcal{H}$QR decompositions. If we compute norms using $\mathcal{H}$-arithmetics the accuracy of the computation is determined by the approximation error of the $\mathcal{H}$-operations. If $r^{\mathcal{H}}_{\mathrm{QR}}$ or $r^{\mathcal{H}}_{\mathrm{orth}}$ is smaller than 10^{-5} the $\mathcal{H}$-arithmetics pretend us an accuracy which Q and R does not have. All we can say in this case is, that the accuracy or the orthogonality is in the range of the approximation error or lower. In Figures 2 and 4 we draw $\max\{r^{\mathcal{H}}_{\mathrm{QR}}, 10^{-5}\}$ and $\max\{r^{\mathcal{H}}_{\mathrm{orth}}, 10^{-5}\}$.

The computations were done on an Intel Xeon Dual Core CPU with 3.0 GHz. The RAM was large enough to store all matrices in the $\mathcal{H}$-matrix format. We used only one core.

4.1 Lintner's $\mathcal{H}$QR decomposition

The three different types of Lintner's $\mathcal{H}$QR decomposition show very different behaviour. The type of Algorithm 1 is the fastest $\mathcal{H}$QR decomposition in our test, but the matrices Q are far from being orthogonal, especially for large matrices. This is the only method, which is faster than the $\mathcal{H}$-inversion for the FEM matrices. This method includes an $\mathcal{H}$-Cholesky factorisation, so that it is quite natural, that this method is not faster than an $\mathcal{H}$-Cholesky factorisation. We should not use this method if we are interested in a factorisation of our matrix. The $\mathcal{H}$-LU or $\mathcal{H}$-Cholesky factorisation are better for this purposes. If we are interested in an orthogonal factorisation, this method is also not the best one, since the Q is often far from being orthogonal.

The second type, which uses the same method to reorthogonalise the result, is good in the two categories orthogonality and accuracy, as long as the reorthogonalisation process converges quick. For the shown numerical tests we stop the reorthogonalisation after six steps, if not before a Q with an orthogonality of at most 10^{-3} is computed. For both test matrices, with dimension 65,536 and 65,538, the reorthognalisation does not converge within this six steps. This is one of the most expensive $\mathcal{H}$QR decomposition, so increasing the number of reorthogonalisation steps will probably make the method the most expensive one.

The third type of Lintner's $\mathcal{H}$QR decomposition is the most expensive $\mathcal{H}$QR decomposition in this test. The orthogonality and the accuracy is in the middle of the other algorithms. To determine convergence in the polar decomposition we compare the norms of D_i and D_{i+1}. If the difference is smaller than the approximation accuracy of the $\mathcal{H}$-arithmetic, we stop the iteration. We stop after 11 iteration steps, too.

4.2 Bebendorf's $\mathcal{H}$QR decomposition

Algorithm 2 is cheaper then the last two algorithms, but the results are not so accurate. Also the orthogonality is not so good. We have not tested the algorithm with a block-Hessenberg matrix, but we expect better results for this type of matrices, see remark 3.2.

Algorithm 1: Lintner's $\mathcal{H}$QR Decomposition

Input: M
Output: Q, R, $M = QR$
1 $B := M^T *_{\mathcal{H}} M$;
2 $R^T :=$ $\mathcal{H}$Cholesky factorisation(B);
3 Solve $M = QR$; `/* Hlib function SolveLeftCholesky */`

complexity ($M_1, M_2 \in \mathcal{H}(T_{I\times I}, k)$, $v \in \mathbb{R}^n$):

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

We will use this well known arithmetic operations ([Gra01, GH03, Lin02, Beb08]) in the $\mathcal{H}$QR decomposition. The $\mathcal{H}$QR decomposition is another $\mathcal{H}$-arithmetic operation of linear-polylogarithmic complexity. We will proof this in Section 3, using the following constant: the *sparsity constant* of an $\mathcal{H}$-matrix is defined as

$$C_{sp} := \max\left\{\max_{r\in T_I}\left\{s \in T_j \,|\, r\times s \in T_{I\times I}\right\}, \max_{s\in T_I}\left\{r \in T_j \,|\, r\times s \in T_{I\times I}\right\}\right\}. \tag{4}$$

For many problems the sparsity constant C_{sp} is independent of the dimension n [Gra01]. We will assume that C_{sp} is constant.

In the next section we will discuss three different ways to compute a QR decomposition of an $\mathcal{H}$-matrix. In order to simplify this we will only investigate matrices on the product index set $I \times I$, which use the same $\mathcal{H}$-tree for the row and column index sets.

3 Three $\mathcal{H}$QR Decompositions

3.1 Lintner's $\mathcal{H}$QR Decomposition

Let $M = QR$ be the QR decomposition of M. Obviously

$$M^T M = R^T Q^T Q R \overset{Q^T Q = I}{=} R^T R \tag{5}$$

holds. Michael Lintner described in his dissertation thesis [Lin02] and later in [Lin04] an $\mathcal{H}$QR decomposition, which first computes R by the Cholesky factorisation of $M^T M$ and later the Q by solving an upper triangular system of equations. This leads to Algorithm 1, the first one for computing an $\mathcal{H}$QR decomposition.

This algorithm consists only of well known hierarchical operations of linear-polylogarithmic complexity. The solution of a linear upper triangular system is a part of the Cholesky decomposition. This $\mathcal{H}$QR decomposition can be implemented easily using the $\mathcal{H}$lib [Hli09], since the three needed functions are included.

As the matrix R is computed without any care to the orthogonality of Q, we can not expect a nearly orthogonal matrix Q. And indeed in many examples, see section 4, the computed Q is not orthogonal. M. Lintner suggests to compute the $\mathcal{H}$QR decomposition of Q

$$M = QR = Q'R'R = Q'(R'R)$$

and to use Q' times $R'R$ as QR decomposition. We can hope that $r_{\mathrm{orth}}(Q') < r_{\mathrm{orth}}(Q)$. If Q' is still not orthogonal enough we can repeat this reorthogonalisation process as often as necessary. But each reorthogonalisation reduces the accuracy of the decomposition and increases the costs.

The condition number of $M^T M$ is

$$\kappa(M^T M) \approx \kappa(M)^2.$$

For badly conditioned problems squaring the condition number increases the error-sensitivity dramatically and causes the low orthogonality.

M. Lintner uses the polar decomposition to reduce the condition number to the square root of $\kappa(M)$. The polar decomposition can be computed by the following iterative method [Hig86]:

$$\begin{aligned} D_0 &= M \\ D_{i+1} &= \frac{1}{2}\left(\gamma_i D_i +_{\mathcal{H}} \frac{1}{\gamma_i}(D_i)^{-T}_{\mathcal{H}}\right), i = 0, 1, 2, \ldots \\ \gamma_{i+1} &= \sqrt{\|D_i^{-1}\|_2} \Big/ \sqrt{\|D_{i+1}^{-1}\|_2} \\ D_i &\xrightarrow{i\to\infty} D^* \\ M &= D^* \underbrace{\left((D^*)^T M\right)}_{=M'}. \end{aligned}$$

D^* is orthogonal and M' is symmetric positive definite, for details see [Lin02]. This method of polar decomposition is similar to the sign-function iteration. Each iteration step has a complexity of $\mathcal{O}(k^2 n \log^2 n)$, since an $\mathcal{H}$-matrix inversion is involved. The $\mathcal{H}$-inversion has an extra large constant in front of the complexity estimate. We will see this in the numerical results in section 4.

Often five or six iteration steps are sufficient to achieve convergence. But one can show that the polar decomposition of the Finite-Element-Method discretisation of the Laplacian needs $\mathcal{O}(\log n)$ steps, so we have a total complexity of $\mathcal{O}(k^2 n \log^3 n)$ in this case.

M. Lintner suggested the following steps: First do the polar decomposition

$$M = QM',$$

then compute the Cholesky factorisation of M'

$$M' = R^T R$$

and finally use Algorithm 1 to decompose R^T

$$R^T = Q'R'.$$

If we insert these equations, we get

$$M = QM' = QR^T R = QQ'R'R.$$

Multiplying $Q *_{\mathcal{H}} Q'$ gives the orthogonal and $R' *_{\mathcal{H}} R$ the upper triangular factor of the $\mathcal{H}$QR decomposition of M. M. Lintner shows that the condition of the last step is

$$\kappa\left(R^T\right) = \sqrt{\kappa\left(M\right)}.$$

So the biggest disadvantage, the squaring of the condition, is more than made up. In the next subsection we will see a second possibility to avoid squaring the condition, but first a remark on the solution of the linear least squares problem:

Remark 3.1. *If we are interested in the solution of the linear least squares problem, then this method is not the best one. This method computes the Cholesky factorisation $R^T R$ of $M^T M$. The later computed Q is not necessary to solve the normal equation $R^T Rx = M^T Mx = Mb$. But solving the normal equations directly is dangerous for ill conditioned matrices M, see [Hig02, p. 386ff].*

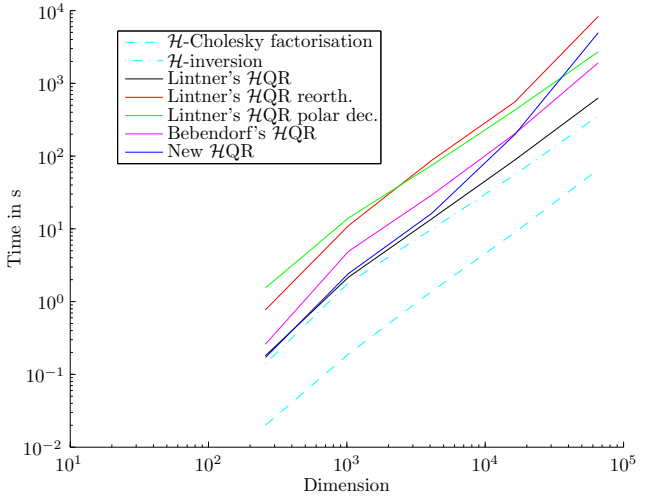


Figure 3: Computation time for the BEM example series

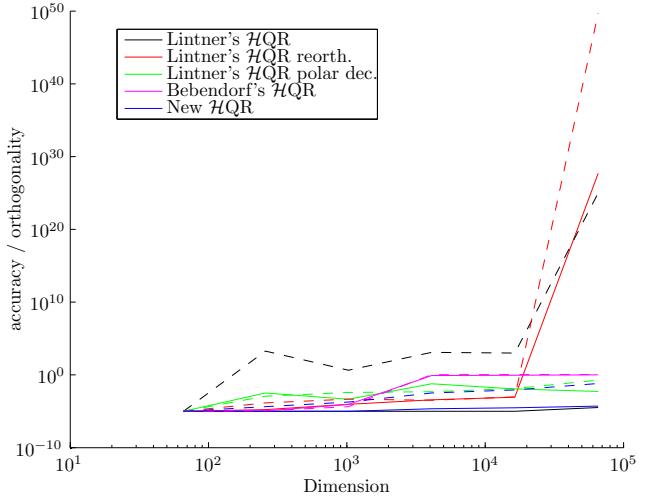


Figure 4: Accuracy and orthogonality (dotted) for the BEM example series

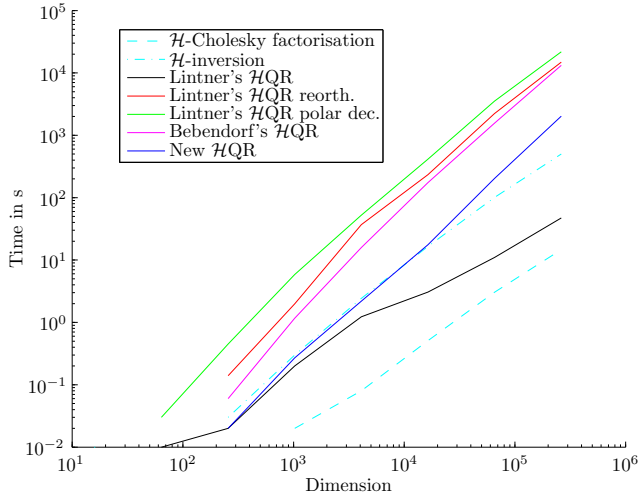


Figure 1: Computation time for the FEM example series

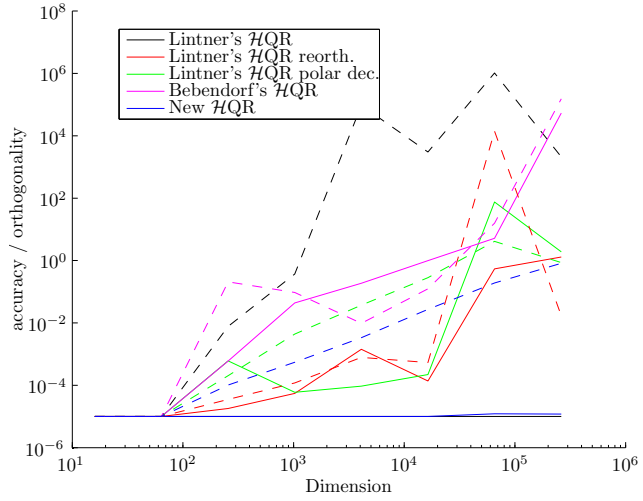


Figure 2: Accuracy and orthogonality (dotted) for the FEM example series

Algorithm 2: Bebendorf's $\mathcal{H}$QR Decomposition

Input: $M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}$
Output: Q, R, $M = QR$

1 **Function** $[Q, R] = \mathcal{H}\texttt{QR}(M)$ **begin**
2 **if** $M \notin L(T)$ **then**
3 $X := M_{21} *_{\mathcal{H}} (M_{11})^{-1}_{\mathcal{H}}$;
4 $L_1 L_1^T := \mathcal{H}$Cholesky factorisation$(I + X^T X)$;
5 $L_2 L_2^T := \mathcal{H}$Cholesky factorisation$(I + X X^T)$;
6 $R := \begin{bmatrix} L_1^T M_{11} & L_1^{-1}\left(M_{12} + X^T M_{22}\right) \\ 0 & L_2^{-1}\left(M_{22} - X M_{12}\right) \end{bmatrix}$;
7 $[Q_1, R_{11}] := \mathcal{H}\texttt{QR}(R_{11})$;
8 $R_{12} := Q_1^T R_{12}$;
9 $[Q_2, R_{22}] := \mathcal{H}\texttt{QR}(R_{22})$;
10 $Q := \begin{bmatrix} I & X^T \\ -X & I \end{bmatrix}^T *_{\mathcal{H}} \begin{bmatrix} L_1^{-1} & 0 \\ 0 & L_2^{-1} \end{bmatrix}^T *_{\mathcal{H}} \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix}$;
11 **else**
12 $[Q, R] = \texttt{QR}(A)$; `/* standard QR decomposition */`
13 **end**
14 **end**

3.2 Bebendorf's $\mathcal{H}$QR Decomposition

Algorithm 2 is described in [Beb08, p. 87ff]. In contrast to the one in the previous subsection this is a direct method. This algorithm computes a series of orthogonal transformations, which triangularise M. On the first recursion level the matrix is transformed to block upper triangular form. The two resulting diagonal blocks are triangularised by recursion. In line 10 of Algorithm 2 the orthogonal transformation of the current level is described by the product

$$\begin{bmatrix} L_1^{-1} & 0 \\ 0 & L_2^{-1} \end{bmatrix} \begin{bmatrix} I & X^T \\ -X & I \end{bmatrix} = \begin{bmatrix} L_1^{-1} & L_1^{-1} X^T \\ -L_2^{-1} X & L_2^{-1} \end{bmatrix}.$$

This is a kind of block Givens rotation, because

$$\det \begin{bmatrix} L_1^{-1} & L_1^{-1} X^T \\ -L_2^{-1} X & L_2^{-1} \end{bmatrix} = \det L_1^{-1} \ \det \left(L_2^{-1}\left(I + X X^T\right)\right) = 1.$$

On each recursion level the matrix M_{11} has to be inverted. Probably this makes the algorithm expensive. Further, not even all matrices have an invertible first diagonal block, i.e.

$$\begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix}$$

has a QR decomposition, but 0 is not invertible.

Bebendorf shows that the complexity of Algorithm 2 is determined by the complexity of the $\mathcal{H}$-matrix multiplication, so this is an algorithm of linear-polylogarithmic complexity, too. He further shows that increasing the $\mathcal{H}$-arithmetics precision from level to level as ϵ/l ensures an orthogonality of order $\epsilon \log d$, d being the depth of $T_{I\times I}$.

Remark 3.2. *If we assume, that the block A_{21} has the structure*

$$A_{21} = \begin{bmatrix} 0 & \cdots & 0 & * \\ 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 \end{bmatrix},$$

as may be the case in QR iterations, then inverting the last diagonal block of A_{11} is enough to zero A_{21}. The matrix Q then has the structure

$$Q = \begin{bmatrix} I & & & & & & & \\ & \ddots & & & & & & \\ & & I & & & & & \\ & & & L_1^{-1} & L_1^{-1}X^T & & & \\ & & & -L_2^{-1}X & L_2^{-1} & & & \\ & & & & & I & & \\ & & & & & & \ddots & \\ & & & & & & & I \end{bmatrix}.$$

This leads to a considerable reduction of cost. For this Q it is easy to see, that we use a block generalisation of Givens rotations.

3.3 A new Method for Computing the $\mathcal{H}$QR Decomposition

In this section we present a new $\mathcal{H}$QR decomposition. The main idea is to use the standard QR algorithm for dense matrices as often as possible. In the numerical comparison we will use the LAPACK [ABB+99] function dgeqrf for the QR decompositions of dense matrices. This new algorithm works recursively block-columnwise.

First we will investigate what to do on the lowest level of the hierarchical structure. Later we will describe the recursive computation for non-leave block-columns. On this higher levels we will use a block modified Gram-Schmidt orthogonalisation. For this block-orthogonalisation we use $\mathcal{H}$-matrix-matrix multiplications and additions. We do not use the expensive $\mathcal{H}$-inversion.

3.3.1 Leave Block-Column

Let M be an $\mathcal{H}$-matrix and $T_{I\times I}$ the corresponding $\mathcal{H}_\times$-tree based on $T_I \times T_I$. We will call a block-column $M_{I\times s}$ a leave block-column if s is a leave of T_I, $s \in \mathcal{L}(T_I)$. In this subsection we will compute the QR decomposition of such a leave block-column.

Our leave block-column $M_{I\times s}$ consists of blocks M_i. These blocks are elements of the block-hierarchical tree $\mathcal{H}_\times$ of our $\mathcal{H}$-matrix. In this subsection we will assume that the block $t_i \times s$ corresponding to M_i includes a leave t_i of our $\mathcal{H}$-tree. The other cases will be treated in the next subsection.

The block $t_i \times s$ is an admissible or a non-admissible block. If the block is non-admissible, we will treat this block like an admissible one by substituting M_i with $I\left(M_i^T\right)^T$ or M_iI. So all blocks of M have the structure $A_iB_i^T$,

$$M = \begin{bmatrix} M_1 \\ M_2 \\ \vdots \\ M_p \end{bmatrix} = \begin{bmatrix} A_1B_1^T \\ A_2B_2^T \\ \vdots \\ A_PB_P^T \end{bmatrix}.$$

We can now write M itself as a product of two matrices:

$$M = AB^T = \begin{bmatrix} A_1 & 0 & \cdots & 0 \\ 0 & A_2 & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & A_p \end{bmatrix} \begin{bmatrix} B_1 & B_2 & \cdots & B_p \end{bmatrix}^T.$$

We notice, that B is a dense matrix.

On the lowest level of the $\mathcal{H}$-tree we use only the standard QR decomposition without any truncation, so we can neglect the errors done on this level since the double-precision accuracy will in general be much higher than the precision of the $\mathcal{H}$-arithmetic approximations on the other levels.

For a binary tree the computation on level l simplifies to

$$\begin{aligned} M &= [M_1, M_2] \\ Q_1, R_{11} &= \mathcal{H}QR(M_1) \\ R_{12} &:= Q_1^T *_{\mathcal{H}} M_2 \\ \tilde{M}_2 &:= M_2 -_{\mathcal{H}} Q_1 *_{\mathcal{H}} R_{12} \\ Q_2, R_{22} &= \mathcal{H}QR(\tilde{M}_2) \end{aligned}$$

For $Q = [Q_1, Q_2]$ we want to estimate the norm of

$$Q^TQ - I = \begin{bmatrix} Q_1^TQ_1 - I & Q_1^TQ_2 \\ \left(Q_1^TQ_2\right)^T & Q_2^TQ_2 - I \end{bmatrix}.$$

From level $l+1$ we already know that $\left\|Q_i^TQ_i - I\right\|_2 \le \delta_{l+1},\ i = 1, 2$. To compute $\tilde{M}_2$ two $\mathcal{H}$-matrix-matrix-multiplications are necessary. From this it follows that there is a matrix F, with $\|F\|_2 \le 2\epsilon_l$, ϵ_l is the $\mathcal{H}$-arithmetic approximation error on level l , and

$$\tilde{M}_2 = M_2 - Q_1Q_1^TM_2 + F.$$

For $Q_1^TQ_2$ we get

$$Q_1^TQ_2 = Q_1^T\left(M_2 - Q_1Q_1^TM_2 + F\right)R_{22}^{-1} = Q_1^TFR_{22}^{-1},$$

and the norm is

$$\left\|Q_1^TQ_2\right\|_2 \le \|F\|_2 \left\|R_{22}^{-1}\right\|_2 \le 2\epsilon_l \left\|R_{22}^{-1}\right\|_2.$$

Now we can estimate, that

$$\delta_l = \left\|Q^TQ - I\right\|_2 = \left\|\begin{bmatrix} Q_1^TQ_1 - I & Q_1^TQ_2 \\ \left(Q_1^TQ_2\right)^T & Q_2^TQ_2 - I \end{bmatrix}\right\|_2 \le \sqrt{\delta_{l+1}^2 + 4\epsilon_l^2\left\|R_{22}^{-1}\right\|_2^2}.$$

It follows, that

$$\delta_l^2 \le \delta_{l+1}^2 + 2c\epsilon_l^2,$$

if $\left\|R_{22}^{-1}\right\|_2 \le c$. For constant ϵ_l we get

$$\delta_1^2 \le \sum_{i=1}^{L} 2c\epsilon_l^2 = 2cL\epsilon_l^2.$$

If we chose $\epsilon_l = \epsilon \Big/ \sqrt{L}$, $r_{\text{orth}}(Q)$ will be $\mathcal{O}(\epsilon)$.

Remark 3.3. *During the testing of this algorithm we observed that especially the last (or the last two) column(s) of Q are not orthogonal to the previous ones. A reorthogonalisation of the last column helps to get a nearly orthogonal Q.*

Algorithm 4: $\mathcal{H}$QR Decomposition of an $\mathcal{H}$-Matrix

Input: $M_{I\times t}$, $\{s_1,\dots,s_q\}\in S(t)$
Output: $Q\in\mathbb{R}^{I\times t}, R\in\mathbb{R}^{t\times t}$, $M=QR$

```
if t ∈ L(T) then
    Compute the QR decomposition using algorithm 3;
else
    for k = 1,...,p do
        If w_k × t is admissible, then compute the QR factorisation of A = QR, store Q and
        form RB^T;
        Split all dense matrix, which overlap more than one block-column;
    end
    for j = 1,...,q do
        for i = 1,...,j−1 do        /* modified Gram-Schmidt orthogonalisation */
            R_{s_i×s_j} := Q^T_{I×s_i} M_{I×s_j};
            M_{I×s_j} := M_{I×s_j} − Q_{I×s_i} R_{s_i×s_j};
        end
        Compute the QR decomposition of M_{I×s_j} recursively.
    end
    for k = 1,...,p do
        If w_k × t is admissible, use the stored Q and the computed rectangular matrix B'
        to form the Rk-matrix QB'^T;
        Recombine overlapping matrices;
    end
end
```

s_i is a father or an ancestor of s. There is only one s_i on each level fulfiling this condition. The definition of the sparsity constant C_{sp}, see equation (4), gives

$$\sum_{\substack{t_i\times s_i\in L^+(T_{I\times I})\\ s\in S^*(s_i)}} \operatorname{rank} M_{t_i\times s_i}\le C_{sp}\operatorname{depth}(T_{I\times I})k_{\max}.$$

So we can bound ρ by

$$\rho\le C_{sp}k_{\max}\,\mathcal{O}(\log n)+C_{sp}n_{\min}.$$

So this QR decompositions costs

$$N_{QR(B,one\ column)}\le C_{sp}\left(k_{\max}\,\mathcal{O}(\log n)+n_{\min}\right)\#sn_{\min}.$$

Summing over all columns gives

$$N_{QR(B,all)}\le C_{sp}\left(k_{\max}\,\mathcal{O}(\log n)+n_{\min}\right)n_{\min}n=\mathcal{O}(k_{\max}n\log n).$$

We can summerise that the used standard QR decompositions needs only a linear-polylogarithmic amount of work.

For the orthogonalisations of block-columns against the previous columns, we need in summation not more $\mathcal{H}$-operations than for two $\mathcal{H}$-matrix-matrix multiplications. Hence the total complexity is linear-polylogarithmic.

3.3.4 Orthogonality

In this subsection we investigate the dependency of the orthogonality on the $\mathcal{H}$-arithmetic approximation error. We will use a recursive approach starting on the lowest level to estimate $\left\|Q^TQ-I\right\|_2$. For simplification we will assume, that the $\mathcal{H}$-tree is a binary tree.

We will compute the QR decomposition of this block-column in two steps. First we transform A to an orthogonal matrix and after that we compute the QR decomposition of the resulting B'.

The matrix A can be orthogonalised block-by-block. Every A_i has to be factorised into $Q_{A_i}R_i$ using the standard QR decomposition, since A_i is dense. The matrices R_i will be multiplied from the right hand side to B_i, $B_i'=B_iR_i^T$.

The second step is as simple as the first one. The matrix B'^T is dense, so we can use the standard QR decomposition for the decomposition of $B'^T=Q_B^TR_B$. We get an orthogonal matrix Q_AQ_B and an upper triangular matrix R_B:

$$M=AB^T=\begin{bmatrix}Q_{A_1}&0&\cdots&0\\0&Q_{A_2}&\ddots&\vdots\\\vdots&\ddots&\ddots&0\\0&\cdots&0&Q_{A_p}\end{bmatrix}\begin{bmatrix}R_1B_1^T\\R_2B_2^T\\\vdots\\R_pB_p^T\end{bmatrix}=\begin{bmatrix}Q_{A_1}&0&\cdots&0\\0&Q_{A_2}&\ddots&\vdots\\\vdots&\ddots&\ddots&0\\0&\cdots&0&Q_{A_p}\end{bmatrix}Q_BR_B.$$

The resulting matrix Q_B^T can be subdivided like B^T into $Q_{B_i}^T$. We can combine the matrices $Q_{A_i}Q_{B_i}^T$ again and get a block-column Q with the same structure as M,

$$M=Q_AQ_BR_B=\begin{bmatrix}Q_{A_1}&0&\cdots&0\\0&Q_{A_2}&\ddots&0\\\vdots&\ddots&\ddots&\vdots\\0&0&\cdots&Q_{A_p}\end{bmatrix}\begin{bmatrix}Q_{B_1}^T\\Q_{B_2}^T\\\vdots\\Q_{B_p}^T\end{bmatrix}R_B.$$

We have

$$\begin{aligned}Q^TQ&=Q_B^T\begin{bmatrix}Q_{A_1}&0&\cdots&0\\0&Q_{A_2}&\ddots&\vdots\\\vdots&\ddots&\ddots&0\\0&\cdots&0&Q_{A_p}\end{bmatrix}^T\begin{bmatrix}Q_{A_1}&0&\cdots&0\\0&Q_{A_2}&\ddots&\vdots\\\vdots&\ddots&\ddots&0\\0&\cdots&0&Q_{A_p}\end{bmatrix}Q_B\\&=Q_B^T\begin{bmatrix}Q_{A_1}^TQ_{A_1}&0&\cdots&0\\0&Q_{A_2}^TQ_{A_2}&\ddots&\vdots\\\vdots&\ddots&\ddots&0\\0&\cdots&0&Q_{A_p}^TQ_{A_p}\end{bmatrix}Q_B=Q_B^TIQ_B=I,\end{aligned}$$

so that Q is orthogonal.

These steps are described in Algorithm 3. We only use matrix-matrix multiplications and QR decompositions for standard matrices, so the accuracy of the computation can be regarded as perfect compared with the approximation error of the $\mathcal{H}$-arithmetic.

3.3.2 Non-Leave Block Column

In the last section we have factorised a leave block-column $M_{I\times s}$, $s\in\mathcal{L}(T_I)$. In this section we will use this to compute recursively the $\mathcal{H}$QR decomposition of a hierarchical matrix. An $\mathcal{H}$-matrix $M\in\mathbb{R}^{I\times I}$ is in general a non-leave block-column.

Let $M_{I\times t}$ be a non-leave block column of M, that means $S(t)\neq\emptyset$. We have chosen the numbering of the indices, so that we can order the sons $s_i\in S(t)$:

$$\begin{aligned}&\forall s_i\in S(t)\setminus\{s_1\} && :s_1<s_i\\&\forall s_i\in S(t)\setminus\{s_1,s_2\} && :s_2<s_i\\&\qquad\vdots\\&\Rightarrow s_1<s_2<s_3<\cdots\end{aligned}$$

Algorithm 3: $\mathcal{H}$QR Decomposition of a Block-Column

Input: $M_{I\times s} = \left[A_i B_i^T\right]_{i=1}^p$
Output: Q, R, with $M_{I\times s} = QR$

```
1  for i = 1,...,p do
2      if M_i is admissible then
3          Compute the QR decomposition A_i = Q_{A_i} R_i;
4          A_i := Q_{A_i} and B_i^T := R_i B_i^T;
5      else
6          M_i ∈ R^{r_i × c_i};
7          if r_i ≤ c_i then
8              B_i^T = M_i (A_i := I);                  /* A_i is already orthogonal */
9          else
10             Compute the QR decomposition M_i = Q_{A_i} R_i;
11             A_i := Q_{A_i} and B_i^T := R_i;
12         end
13     end
14 end
15 Assemble B^T = [B_1 B_2 ··· B_p]^T;
16 Compute the QR decomposition B^T = Q_B R_B;
17 Partition B^T into B_i^T;
18 for i = 1,...,p do
19     if M_i is admissible then
20         A_i B_i^T is a block of Q;
21     else
22         if A_i = I then
23             B_i^T is a dense block of Q;
24         else
25             Compute A_i B_i^T to get a dense block of Q;
26         end
27     end
28 end
```


We get the QR decomposition of the first block-column $M_{I\times s_1}$ by using this recursion. The QR decomposition of the second block-column starts with the orthogonalisation w.r.t. the first block-column by using a block modified Gram-Schmidt orthogonalisation. We compute

$$R_{s_1\times s_2} = Q_{I\times s_1}^T *_{\mathcal{H}} M_{I\times s_2} \qquad \text{and}$$
$$M'_{I\times s_2} := M_{I\times s_2} -_{\mathcal{H}} Q_{I\times s_1} *_{\mathcal{H}} R_{s_1\times s_2}.$$

Now we apply again the recursion to compute the QR decomposition of $M'_{I\times s_2}$. If t has more than two sons, the other block-columns can be treated analogously. Algorithm 4 describes these steps in algorithmic form.

We have chosen the hierarchical block-tree in order to find large admissible blocks. Let $w \times t$ be an admissible block and $C \in \mathbb{R}^{s_i\times t} = AB^T$ the corresponding **R**k-matrix. We have to split C in two submatrices before we continue with the steps above. We will do something similar to Algorithm 3 to get a dense, easily partitionable matrix: computing the standard QR decomposition of $A = Q_A R_A$, multiplying $R_A B^T$ and storing Q_A. Using Q_A we can factorise our block-column in

$$M_{I\times t} = \begin{bmatrix} I & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & Q_A & 0 \\ 0 & \cdots & 0 & I \end{bmatrix} \begin{bmatrix} M_{w_1\times s_1} & M_{w_1\times s_2} \\ M_{w_2\times s_1} & M_{w_2\times s_2} \\ \vdots & \vdots \\ R_A B^T & \\ \vdots & \vdots \end{bmatrix}.$$

The matrix $R_A B^T \in \mathbb{R}^{k\times \#t}$ is a small rectangular dense matrix. Splitting a dense matrices into two block-columns in a columnwise organised storage simply means setting a second pointer on the first element of the second matrix. Note: on the next levels we have to split this dense matrix again.

The matrix Q has the same structure as M, apart from the adaptive chosen ranks of the admissible blocks. The matrix R is an $\mathcal{H}$-matrix, too. In the next section we will investigate the complexity of this algorithm.

3.3.3 Complexity

In order to simplify the complexity analysis, we assume the existence of a constant $k_{\max}$, which bounds the ranks k.

Each matrix A_i of every **R**k-matrix $A_i B_i^T$ is decomposed using the standard QR decomposition. The matrix A_i has the dimension $m \times k$, with k the rank of the **R**k-matrix. The standard QR decomposition of a matrix $F \in \mathbb{R}^{e\times f}$ needs $\mathcal{O}(f^2 e)$ flops, so the QR decomposition of A_i needs $\mathcal{O}(k^2 m)$ flops. The matrix A_i needs $N_{A_i,st} = mk$ storage. If we sum over all A_i, we get

$$\sum_i N_{QR(A_i)} \leq \mathcal{O}\left(k_{\max} N_{\mathcal{H},st}\right).$$

Analogously the number of flops for line 10 in Algorithm 3 is in $\mathcal{O}\left(n_{\min} N_{\mathcal{H},st}\right)$.

For each block-column $M_{I\times s}$ of the lowest level we compute another QR decomposition to treat the remaining factor of compounded B_i. The vertex s has at most $n_{\min}$ indices, otherwise the non-admissible diagonal block of M would be divided once more. It follows that such a block-column has at most $n_{\min}$ columns, $\#s \leq n_{\min}$. Further we need the number of rows ρ of B^T. The block-column is composed of matrix-blocks M_i (and parts of such blocks) of the form $t_i \times s_i$ with $t_i \times s_i \in L(T_{I\times I})$ and $s \in S^*(s_i)$. The number of rows ρ is the sum of

$$\rho = \sum_{\substack{t_i\times s_i \in L^+(T_{I\times I}) \\ s\in S^*(s_i)}} \operatorname{rank} M_{t_i\times s_i} + \sum_{\substack{t_i\times s_i \in L^-(T_{I\times I}) \\ s\in S^*(s_i)}} \min\left\{\#s_i, \#t_i\right\}.$$

In the non-admissible leaves we use the test in line 7 in Algorithm 3 to ensure that the number of rows of the corresponding B_i^T is $\min\{\#s_i, \#t_i\} \leq n_{\min}$. The condition $s \in S^*(s_i)$ means, that