A Block Householder Based Algorithm for the QR Decomposition of Hierarchical Matrices

Vom Promotionsausschuss der
Technischen Universität Hamburg

zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften (Dr. rer. nat.)

genehmigte Dissertation (Monographie)

von

Vincent Eric Griem

aus

Hamburg, Deutschland

2025

Gutachter:

1. Gutachter: Prof. Dr. Sabine Le Borne
2. Gutachter: Prof. Dr. Steffen Börm

Tag der mündlichen Prüfung: 09. Mai 2025

https://orcid.org/0009-0004-7508-5632
DOI: https://doi.org/10.15480/882.15934

Acknowledgements

Foremost, I would like to thank Professor Dr. Sabine Le Borne for her support and guidance throughout this work. Her thoughtful advice and steady encouragement were invaluable at many points along the way. I'm also grateful to Professor Dr. Steffen Börm for taking the time to review this thesis and for the interesting discussions.

I owe a special thanks to my family for their continuous support during this journey. Most of all, I want to thank my wife Sandra. Without her love and encouragement, this would not have been possible. I also want to thank my daughters Anna, Clara, and Sophie. All three were born while I was working on this thesis, and their presence brought both joy and perspective to this time in my life. I also owe gratitude to my parents, who always supported me in my love for mathematics.

Finally, I want to thank my colleagues at the Institute of Mathematics for the friendly and supportive atmosphere.

Summary

Hierarchical matrices are dense but data-sparse matrices that use low-rank factorizations of suitable submatrices to reduce storage and computational cost to linear-polylogarithmic complexity. In this thesis, we propose a new approach to efficiently compute QR factorizations in the hierarchical matrix format based on block Householder transformations. To avoid unnecessarily high ranks in the resulting factors and to increase speed and accuracy, the algorithm carefully tracks for which intermediate results low-rank factorizations are available.

This is done by introducing special admissibility functions that only allow admissible blocks if a low-rank factorization can be directly derived from the low-rank factorizations in the matrices involved. We also employ a special implicit storage scheme for the block Householder reflector to further reduce computational and storage cost.

We develop a way to directly describe these induced admissibility functions and, by finding an extensive set of upper bounds for them, we also find a bound on the computational cost. This is done without resorting to the often-used idempotency constant C_{id}. We also provide rather weak conditions under which these upper bounds are reached when no further data sparsity in the matrix has been taken into account.

Numerical tests on 2D Laplacian boundary element matrices, various RBF interpolation matrices, and square matrices with typical hierarchical matrix structures but filled with random entries, illustrate the performance of the new algorithm compared to some other QR algorithms for hierarchical matrices from the literature.

Unfortunately, the application of the algorithm to rectangular matrices was not as successful. However, this unsatisfactory result is supported by theoretical considerations that show why most QR factors of rectangular matrices are unlikely to allow an $\mathcal{H}$-matrix approximation in the absence of further data sparsity.

We also implement a version of the standard $\mathcal{H}$-LU decomposition that uses the induced admissibility function. Numerical results for the latter show that the application of this idea may also be beneficial for other computations of the $\mathcal{H}$-matrix.

Contents

List of Figures . . . 9
List of Algorithms . . . 11
List of Assumptions . . . 13
Nomenclature . . . 15

1 Introduction **19**
1.1 Contribution of this Thesis . . . 20
1.2 Notation . . . 22
1.3 Structure of this Thesis . . . 23

2 Preliminaries **25**
2.1 Linear Algebra . . . 25
2.2 Hierarchical Matrices . . . 34
2.2.1 Introduction . . . 34
2.2.2 Construction of a Hierarchical Matrix . . . 52
2.2.3 Hierarchical Matrix Arithmetics . . . 57
2.3 Further Matrix Formats . . . 70
2.3.1 Sparse Matrices . . . 70
2.3.2 $\mathcal{H}_p$-Matrices . . . 71
2.3.3 $\mathcal{H}^2$-Matrices . . . 73
2.4 Representation of Hierarchical Matrices in the Software Package H2Lib . . . 75
2.5 Radial Basis Functions . . . 77
2.6 Boundary Element Method . . . 80

3 QR Decomposition for Hierarchical Matrices **85**
3.1 Known QR Decompositions for Hierarchical Matrices . . . 85
3.1.1 Cholesky-based $\mathcal{H}$-QR Decomposition . . . 86
3.1.2 Gram-Schmidt $\mathcal{H}$-QR Decomposition . . . 88
3.1.3 Bebendorf's $\mathcal{H}$-QR Decomposition . . . 90
3.2 Block Householder Based Algorithm for the $\mathcal{H}$-QR Decomposition . . . 91
3.2.1 Highest Level . . . 92
3.2.2 Lower Levels . . . 98
3.2.3 Update Matrices . . . 104

3.3 Complexity Analysis . . . 114
3.3.1 Upper Bounds for the Induced Admissibility Functions . . . 119
3.3.2 Conditions for the Exactness of the Upper Bounds . . . 126
3.3.3 Induced Admissibility Function of Q . . . 137
3.3.4 Parameter Bounds . . . 142
3.3.5 Cost Estimate for the $\mathcal{H}$-QR Decomposition . . . 147
3.3.6 Reusing the Original Admissibility Function . . . 162
3.3.7 The Rectangular Case . . . 164
3.4 Implementation in the Software Package H2Lib . . . 167

4 Numerical Results **173**
4.1 Square Matrices . . . 176
4.1.1 2D Laplace Operator . . . 176
4.1.2 3D Laplace Operator . . . 180
4.1.3 1D RBF Interpolation . . . 183
4.1.4 2D RBF Interpolation . . . 189
4.1.5 Randomly Filled Hierarchical Matrix with X-like Structure . . . 191
4.2 Rectangular Matrices . . . 191
4.2.1 Random $\mathcal{H}_p$-Matrices . . . 192
4.2.2 1D RBF Approximation . . . 194
4.3 Summary . . . 198

5 Induced Admissibility for the $\mathcal{H}$-LU Decomposition **201**
5.1 Partially Induced Admissibility Function . . . 202
5.2 Numerical Results . . . 208

6 Conclusion and Outlook **213**

List of Figures

2. Preliminaries **25**

2.1 Illustration of the blocks in the recursive Householder block QR factorization. . . . 31
2.2 Illustration of an admissibility function. . . . 40
2.3 Computing the $\mathcal{H}$-matrix product on different levels. . . . 43
2.4 Induced matrix product admissibility function. . . . 45
2.5 Oriented $\mathcal{H}$-matrices. . . . 47
2.6 Two $\mathcal{H}$-matrices and their bandwidth C_{bw} and sparsity constant C_{sp}. . . . 49
2.7 Adaptive cross approximation. . . . 55
2.8 Visualization of Definition 2.45. . . . 65
2.9 Related sparse matrix formats. . . . 71
2.10 $\mathcal{H}_p$-matrix. . . . 72
2.11 Nested cluster basis for an $\mathcal{H}^2$-Marix. . . . 74
2.12 Illustration of Gaussian RBFs. . . . 77
2.13 llustration of the interior domain D and the boundary S. . . . 80

3. QR Decomposition for Hierarchical Matrices **85**

3.1 Vertical splitting during the block Householder QR decomposition. . . . 92
3.2 Application of Algorithm 3.7 to the matrix H from Example 3.1. . . . 93
3.3 Factors Y and R of a rectangular matrix. . . . 102
3.4 The set $P_{J\times J}^{upright}$ for a given $\mathcal{H}$-matrix. . . . 111
3.5 Partner index set. . . . 117
3.6 $\mathcal{H}$-Matrix examples to illustrate Assumption D. . . . 138
3.7 Comparison of the admissibility function adm_Y and the admissibility function of the bound for adm_{YT}. . . . 140
3.8 The factor Y for square and rectangular matrices. . . . 148
3.9 Sequence of all factors described in Theorem 3.24. . . . 151

4. Numerical Results **173**

4.1 (Square) $\mathcal{H}$-matrices used in our numerical tests. . . . 174
4.2 QR factors of a 2D Laplacian $\mathcal{H}$-matrix. . . . 177
4.3 Numerical results for 2D Laplacian matrices I. . . . 178
4.4 Numerical results for 2D Laplacian matrices II. . . . 179
4.5 Numerical results for 2D Laplacian matrices III. . . . 179

4.6 QR factors of a 3D Laplacian $\mathcal{H}$-matrix. 181
4.7 Numerical results for 3D Laplacian matrices I. 182
4.8 Numerical results for 3D Laplacian matrices II. 183
4.9 QR factors of a 1D RBF interpolation matrix. 184
4.10 Numerical results for 1D RBF interpolation matrices. 185
4.11 QR factors of a randomized 1D RBF interpolation matrix. 186
4.12 Numerical results for randomized 1D RBF interpolation matrices. 187
4.13 QR factors of a 2D RBF interpolation matrix. 188
4.14 Numerical results for 2D RBF interpolation matrices. 189
4.15 QR factors of a random X-like $\mathcal{H}$-matrix. 190
4.16 Numerical results for random X-like $\mathcal{H}$-matrices. 192
4.17 QR factors of a random rectangular $\mathcal{H}_p$-matrix. 193
4.18 Numerical results for random rectangular $\mathcal{H}_p$-matrices. 194
4.19 QR factors of a 1D RBF approximation matrix. 195
4.20 Numerical results for 1D RBF approximation matrices. 196
4.21 QR factors of a randomized 1D RBF approximation matrix. 197
4.22 Numerical results for randomized 1D RBF approximation matrices. 198

5. Induced Admissibility for the $\mathcal{H}$-LU Decomposition 201
5.1 LU factors of a randomized 2D RBF interpolation matrix. 208
5.2 Numerical results (LU) for randomized 2D RBF interpolation matrices I. 209
5.3 Numerical results (LU) for randomized 2D RBF interpolation matrices II. 210

List of Algorithms

2. Preliminaries **25**
2.1 Adaptive cross approximation with partial pivoting. 57
2.2 Truncation using the singular value decomposition. 59
2.3 Fast truncation of $\mathcal{H}$-matrices. 63

3. QR Decomposition for Hierarchical Matrices **85**
3.1 Cholesky decomposition of a symmetric positive definite $\mathcal{H}$-Matrix A. . . . 85
3.2 Forward substitution. 86
3.3 Backward substitution. 87
3.4 Cholesky-based QR decomposition. 88
3.5 Gram-Schmidt $\mathcal{H}$-QR decomposition - the recursion start. 89
3.6 Gram-Schmidt $\mathcal{H}$-QR decomposition. 90
3.7 Householder-based $\mathcal{H}$-QR decomposition - the recursion start. 97
3.8 Recursive Householder-based QR decomposition for $\mathcal{H}$-matrices. 99
3.9 Update algorithm computing $\widetilde{A}_2 = Q_1^T A_2$. 100
3.10 Combine results of the recursive calls for the column subblocks. 101
3.11 Multiplication of a low-rank $\mathcal{H}$-matrix with T given in its reduced form $\overline{T}$. 153
3.12 Multiplication of an $\mathcal{H}$-matrix with T given in its reduced $\mathcal{H}$-matrix $\overline{T}$. . . 155
3.13 Vertically split $\mathcal{H}$-matrices in H2Lib. 168
3.14 Induced structure for the $\mathcal{H}$-matrix sum in H2Lib. 169
3.15 Induced structure for the $\mathcal{H}$-matrix product in H2Lib. 170

5. Induced Admissibility for the $\mathcal{H}$-LU Decomposition **201**
5.1 LU decomposition for $\mathcal{H}$-matrices. 202

List of Assumptions

2. Preliminaries **25**

A All blocks in the hierarchy of block partitionings have zero, two or four non-trivial direct descendants. 38

3. QR Decomposition for Hierarchical Matrices **85**

B For every admissible block $A_{\sigma\times\tau} = UV^T$, the factor U has orthogonal columns. 94

C For every $\mathcal{H}$-matrix block column on the highest level, we can find a unique block row that contains a square inadmissible block. 101

D In every $\mathcal{H}$-matrix we can find a sequence of inadmissible blocks from the top left to the bottom right. 137

E The block column sizes of the near-field of every $\mathcal{H}$-matrix are balanced. . . 149

F The blocks on the highest level of the hierarchical partitioning of the rows and the columns are each roughly the same size. 165

Nomenclature

Acronyms

ACA	Adaptive cross approximation	Algorithm 2.1
BEM	Boundary element method	Section 2.6
FDM	Finite difference method	Section 2.6
FEM	Finite element method	Section 2.6
Flop	Floating point operation	
HODLR	Hierarchically off-diagonal low-rank (matrix)	Definition 2.51
RBF	Radial basis function	Section 2.5
SVD	Singular value decomposition	Theorem 2.10

Sets and Spaces

$\mathcal{H}(P^{\mathcal{H}}_{I\times J}, k, \text{adm})$	Set of $\mathcal{H}$-Matrices	Definition 2.23
$\mathcal{H}^p(P^{\mathcal{H}}_{I\times I}, k, \text{adm})$	Set of $\mathcal{H}_p$-Matrices	Definition 2.51
I_n	Identity matrix with n rows and columns	
$P^{streak}_{I\times J}$	Diagonal-like subset of $P^{L_{IJ}}_{I\times J}$	Assumption D
$P^{upright}_{J\times J}$	Blocks used in the superior structure of $\overline{T}$	Definition 3.4
$\mathcal{H}^2(P^{\mathcal{H}}_{I\times J}, k^V, k^W, \text{adm}, V, W)$	Set of $\mathcal{H}^2$-Matrices	Definition 2.54

Matrices

range(A)	Range of matrix A	Definition 2.1
rank(A)	Rank of matrix A	Definition 2.1

$A_{\sigma,\tau}$	Submatrix of A induced by the row index set σ and the column index set τ	

Hierarchical Structure

$\Sigma(\tau)$	Set of partners for lower diagonal blocks in Y	Definition 3.6
$\sigma(\tau)$	Union set of all elements from $\Sigma(\tau)$	Definition 3.6
adm^{cw}	Admissibility function that is forced to be column-wise oriented	Definition 3.7
adm^{ext}	Extended admissibility function	Definition 3.6
adm^{rw}	Admissibility function that is forced to be row-wise oriented	Definition 3.7
$\mathrm{anc}(I_{k'}^{\ell'})$	Direct ancestor of $I_{k'}^{\ell'}$	Definition 2.18
$\mathrm{anc}_{\ell}(I_{k'}^{\ell'})$	Ancestor of $I_{k'}^{\ell'}$ on level ℓ	Definition 2.18
$\mathrm{depth}(P_{I\times J}^{\mathcal{H}})$	Depth of $\mathcal{H}$-block partition $P_{I\times J}^{\mathcal{H}}$	Definition 2.20
$\mathrm{depth}_{col}(P_{I\times J}^{\mathcal{H}})$	Column-wise depth of $\mathcal{H}$-block partition $P_{I\times J}^{\mathcal{H}}$	Definition 2.20
$\mathrm{depth}_{row}(P_{I\times J}^{\mathcal{H}})$	Row-wise depth of $\mathcal{H}$-block partition $P_{I\times J}^{\mathcal{H}}$	Definition 2.20
$\mathrm{diam}(\sigma)$	Diameter of the cluster σ	Subsection 2.2.2
$\mathrm{dist}_A(\sigma,\sigma')$	Distance of two index sets in the $\mathcal{H}$-matrix A	Definition 2.26
$\mathrm{dist}_g(\sigma,\sigma')$	Distance between the clusters σ and σ'	Subsection 2.2.2
$\mathrm{dist}_{col;A}(\sigma,\sigma')$	Distance of two column index sets in the $\mathcal{H}$-matrix A	Definition 2.26
$\mathrm{dist}_{row;A}(\tau,\tau')$	Distance of two row index sets in the $\mathcal{H}$-matrix A	Definition 2.26
$\mathrm{level}(\sigma)$	Representative level of the partition in the hierarchy of partitionings that contains the index set σ	Definition 2.18
$L_{final}^{\ell}(\sigma,A)$	Index of the last non-zero block in the row induced by σ on level ℓ	Definition 3.17
$L^{col}(\sigma,J)$	All levels of $(P_J^{\ell})_{\ell=0}^{L_J}$ containing partners of σ in $P_{I\times J}^{hier}$	Definition 2.19
$L^{intersec}(\sigma,\gamma,J)$	Set of all intersecting levels for a block $\sigma\times\gamma\in P_{I\times K}^{\mathrm{hier}}$	Definition 2.25
$L^{max}(\sigma,\gamma,J)$	Maximum of $L^{intersec}(\sigma,\gamma,J)$	Definition 2.25
$L^{row}(\tau,I)$	All levels of $(P_I^{\ell})_{\ell=0}^{L_I}$ containing partners of τ in $P_{I\times J}^{hier}$	Definition 2.19

$L_{floor}(\sigma, \tau_i, A)$	Replaces the index i by $L^{\ell}_{final}(\sigma, A)$ if $\sigma \times \tau_i$ is a zero block	Definition 3.17
$P^{\mathcal{H},+}_{I\times J}$	Far-field of the $\mathcal{H}$-block partition $P^{\mathcal{H}}_{I\times J}$	Definition 2.22
$P^{\mathcal{H},-}_{I\times J}$	Near-field of the $\mathcal{H}$-block partition $P^{\mathcal{H}}_{I\times J}$	Definition 2.22
$P^{\mathcal{H}}_{I\times J}$	$\mathcal{H}$-block partition based on $(P^{\ell}_I)^{L_I}_{\ell=0}$	Definition 2.20
P^{hier}_I	Set of all subsets of the hierarchy of partitionings given by $\left(P^{\ell}_I\right)^{L}_{\ell=0}$	Definition 2.18
$U_\ell(\sigma \times \gamma)$	Partner clusters of σ and γ or their ancestors	Definition 2.45

Constants

$C_{bw,c}(\text{adm}, A)$	Column-wise bandwidth constant of A respective to adm	Definition 2.29
$C_{bw,r}(\text{adm}, A)$	Row-wise bandwidth constant of A respective to adm	Definition 2.29
$C_{bw}(\text{adm}, A)$	Bandwidth constant of A respective to adm	Definition 2.29
$C_{id}\left(\bar{P}^{\mathcal{H}}_{I\times K}, P^{\mathcal{H}}_{I\times K}\right)$	Idempotency constant of $\bar{P}^{\mathcal{H}}_{I\times K}$ relative to $P^{\mathcal{H}}_{I\times K}$	Definition 2.32
$C_{imb}(I, J)$	Imbalance constant of $\left(P^{\ell}_{I\times J}\right)^{L}_{\ell=0}$	Definition 2.33
$C_{sp,c}(P^{\mathcal{H}}_{I\times J})$	Column-wise sparsity constant of $P^{\mathcal{H}}_{I\times J}$	Definition 2.28
$C_{sp,r}(P^{\mathcal{H}}_{I\times J})$	Row-wise sparsity constant of $P^{\mathcal{H}}_{I\times J}$	Definition 2.28
$C_{sp}(P^{\mathcal{H}}_{I\times J})$	Sparsity constant of $P^{\mathcal{H}}_{I\times J}$	Definition 2.28

Computational and Storage Cost

$\mathcal{N}^{\mathcal{H},\text{fast}}_{k\leftarrow qk}$	Computational cost for the fast truncation of an $\mathcal{H}$-matrix from rank qk to k	Theorem 2.44
$\mathcal{N}_+(A, B, C)$	Computational cost for the additional truncation during the $\mathcal{H}$-matrix multiplication $A \cdot B = C$	Theorem 2.47
$\mathcal{N}^{\text{fast}}_+(A, B, C)$	Computational cost for the additional fast truncation during the $\mathcal{H}$-matrix multiplication $A \cdot B = C$	Theorem 2.47
$\mathcal{N}_{\mathcal{H},k'\leftarrow k}$	Computational cost for the truncation of an $\mathcal{H}$-matrix from rank k to k'	Theorem 2.42

$\mathcal{N}_{F,mv}(m,n)$	Computational cost for the matrix-vector multiplication with a m by n dense matrix	Theorem 2.35
$\mathcal{N}_{MM}^{fast}(A,B,C)$	Computational cost for the multiplication $A \cdot B = C$ with fast truncation	Theorem 2.47
$\mathcal{N}_{MV}(P_{I\times J}^{\mathcal{H}},k)$	Computational cost for the $\mathcal{H}$-matrix-vector product	Theorem 2.41
$\mathcal{N}_{R,\oplus}(n,m,k)$	Computational cost for the formatted addition of two rank n by m rank k matrices to a rank k matrix	Theorem 2.38
$\mathcal{N}_{R,k'\leftarrow k}(m,n)$	Computational cost for truncation by an SVD of an n by m rank k matrix to arbitrary rank $k' < k$	Theorem 2.37
$\mathcal{S}_F(m,n)$	Storage cost for a m by n dense matrix	Theorem 2.35
$\mathcal{S}_R(m,n,k)$	Storage cost for a m by n rank k matrix	Theorem 2.36
$N_{\mathcal{H},\oplus}(P_{I\times J}^{\mathcal{H}},k)$	Computational cost for the formatted addition of two rank k $\mathcal{H}$-matrices to a rank k $\mathcal{H}$-matrix	Theorem 2.43
$N_{lorth}(A)$	Computational cost for the left orthogonalization of an $\mathcal{H}$-matrix	Theorem 3.22
$N_{MM}(A,B,C)$	Computational cost for the exact $\mathcal{H}$-matrix product $A \cdot B = C$	Theorem 2.47
$N_{QR,comb}$	Computational cost for the recursive combination of the factors during the block Householder QR decomposition	Theorem 3.28
$N_{QR,high}(Y)$	Computational cost for the block Householder QR decomposition on the highest level	Theorem 3.23
$N_{QR,upd}$	Computational cost for all updates during the block Householder QR decomposition	Theorem 3.27
$N_{R,mv}(m,n,k)$	Computational cost for the matrix-vector multiplication with a n by m rank k matrix	Theorem 2.36
$S_{\mathcal{H}}(P_{I\times J}^{\mathcal{H}},k)$	Storage cost for an $\mathcal{H}$-matrix derived by $P_{I\times J}^{\mathcal{H}}$ with rank k	Theorem 2.40

Chapter 1

Introduction

$\mathcal{H}$-matrices, short for hierarchical matrices, are hierarchically structured matrices that use some underlying data sparsity to approximate certain suitable submatrices by a low-rank factorization. They were first introduced in 1998 by Hackbusch [42]. The observation that certain dense matrices allow for low-rank factorizations of suitable submatrices has already been used before, most notably in fast multipole methods [39, 5].

$\mathcal{H}$-matrices offer advantages in storage cost and allow efficient approximations of matrix operations, including matrix-matrix multiplications and the LU decomposition [36, 7, 37, 38]. In general, these operations have almost linear complexity

$$\mathcal{O}(n \log^{\alpha} n) \text{ for } 1 \leq \alpha \leq 2,$$

but aside from the matrix-vector multiplication, the results are approximations. This is because the result has to fit into a given hierarchical structure, which is done either by decreasing the rank of a submatrix that is already in a low-rank factorization or by approximating a dense submatrix by a low-rank factorization. To achieve the desired complexity, these approximations are addressed during the computation itself, often based on algorithms involving the singular value decomposition. To find low-rank approximations of matrices that are only implicitly known - a use case that is highly relevant for the creation of the $\mathcal{H}$-matrix in the first place - techniques such as the adaptive cross-approximation [34, 6, 9] are used.

The choice of an appropriate structure for the result of an $\mathcal{H}$-matrix operation has received the most attention in the context of (theoretical) cost analysis. It is necessary to know how often dense blocks interact to produce a dense block in the result in order to estimate the cost of the whole operation. However, the exact structural result is not explicitly used as the target structure. Furthermore, these considerations are mainly focused on $\mathcal{H}$-matrix addition and multiplication because the costs of other operations can often be traced back to these two cases.

In practical applications, such as the use of the $\mathcal{H}$-matrix LU decomposition, the reuse of the original structure of A for the factors L and U is the primary approach [7, 37, 4, 61]. This is very different from the situation for sparse matrices, which can be seen as a simple variant of $\mathcal{H}$-matrices, where zero entries are (1×1) matrices of rank 0 and non-zero entries are (1×1) dense matrices. For sparse matrices, a lot of research has been devoted to finding suitable sparsity patterns through reordering (e.g., bandwidth reduction or nested dissection) because often not the sparsity pattern of the original matrix is reused, but some or all emerging non-zero entries are kept. Therefore, it is essential to choose a sparsity pattern at the beginning that leads to the least fill-in. Note that, in the case of keeping all emerging non-zero entries, the sparse LU decomposition is exact.

Unfortunately, these techniques cannot be applied to $\mathcal{H}$-matrices in a similar manner due to their hierarchical structure. The $\mathcal{H}$-matrix structure of the original matrix depends on the problem it describes, and reorderings are generally not possible. Nevertheless, a decision has to be made as to which $\mathcal{H}$-matrix structure will be used for the result. Due to the possibility of using adaptive ranks in $\mathcal{H}$-matrix computations, the choice of an ill-fitting $\mathcal{H}$-matrix structure mainly leads to an increase in computational and storage costs and not necessarily to a loss of accuracy.

Algorithms for the QR decomposition of $\mathcal{H}$-matrices have also been developed [66, 65, 8, 10] but they all suffer numerical drawbacks that limit their use in many applications, especially if the matrix is ill-conditioned [60]. Furthermore, computing the (full) QR decomposition for rectangular $\mathcal{H}$-matrices was not explicitly done, although most of the suggested methods are generally applicable but may only compute reduced QR factorizations.

The QR factorization is useful for the solution of various linear algebra problems, such as solving systems of linear equations or least squares problems. Specifically, as shown in [53], square RBF kernel matrices from an interpolation scheme can be approximated as $\mathcal{H}$-matrices, and a transfer of these results to rectangular RBF kernel matrices and an approximation scheme seems natural. Access to an efficient QR decomposition would then allow the approximation problem to be solved efficiently. A full QR factorization can also be used to find an orthogonal basis of a nullspace, which is needed in so-called nullspace methods [64, 72].

1.1 Contribution of this Thesis

The main focus of this thesis is to develop a Householder-based alternative to the existing QR decomposition algorithms for $\mathcal{H}$-matrices that is stable and applicable to a wide range of $\mathcal{H}$-matrices.

Our approach was inspired by [60], which introduced a Householder-based QR decomposition for hierarchically off-diagonal low-rank (HODLR) matrices. HODLR matrices have a fixed hierarchical block structure in which all off-diagonal blocks are not further refined. A first goal was to generalize the Householder-based QR decomposition from [60] to more general (arbitrary) hierarchical matrix structures. This opens up flexibility not only for the input matrix A whose QR factors are to be determined, but also for the hierarchical matrix structures of the QR factors themselves. Therefore, in addition to the (primarily technical) generalization of the QR algorithm from HODLR to the general $\mathcal{H}$-matrix format, another novelty lies in the introduction of new induced admissibility functions and the resulting $\mathcal{H}$-block partitions for all occurring matrices.

The orthogonal matrix Q will not be computed explicitly but in the form $Q = I - YTY^T$ for trapezoidal matrices Y and T. Another notable contribution lies in the introduction of a reduced form to store and compute with the matrix T in this factored representation of Q.

Furthermore, we were also able to bound the cost of this algorithm with respect to the changing structures. This, in itself, is different compared to the cost estimates of other $\mathcal{H}$-matrix algorithms that, in most cases, use predefined target structures with their corresponding parameters. To do that, we bound the admissibility functions, meaning that we find admissibility functions that only induce more and not fewer inadmissible blocks, for all intermediate results until we have computed the complete QR decomposition. This allows us to express the admissibility functions of the resulting factors using the admissibility function of the initial matrix. Using this result, we can then relate further parameters of the QR decomposition to the corresponding parameters of the starting matrix and thus let the cost estimate depend only on the initial matrix.

Using the same technique, we are also able to show why existing methods for computing QR decomposition have problems for many $\mathcal{H}$-matrices. We also apply this method to the LU decomposition and are able to easily describe the change in the structure of the LU factors. The result itself is not surprising because it is a simple transfer of the well-known result regarding the sparsity of the LU factors of a sparse matrix. Still, it shows that this way of defining the admissibility functions of computational results can be applied elsewhere. Hence, this thesis may serve as a starting point to apply this technique in other $\mathcal{H}$-matrix operations, especially if several steps of computations are involved.

Finally, another contribution lies in numerical tests to illustrate the new $\mathcal{H}$-QR factorization compared to methods from the literature. Parts of this thesis - including most of the numerical analysis but excluding the cost estimate and further related results - were also published by us in [40].

1.2 Notation

Due to the hierarchical structure, especially when rectangular $\mathcal{H}$-matrices are involved, the notation can appear convoluted. An often used resort is only to use square or even symmetric $\mathcal{H}$-matrices. Unfortunately, this is not possible because even if we start with such an $\mathcal{H}$-matrix after our algorithm's first step, we get two non-symmetric and rectangular $\mathcal{H}$-matrices. Hence, we need to introduce and use rectangular $\mathcal{H}$-matrices throughout this thesis.

However, we try to simplify whenever possible without sacrificing general applicability. An essential part of this endeavour are assumptions that are scattered throughout this thesis. If not declared otherwise, we implicitly assume them to hold from the point of their introduction onwards. In most cases, they do not substantially change for which $\mathcal{H}$-matrices our statements are true; they only force a specific way in which their structure is described.

Another attempt to simplify the notation are abbreviations for often-used variables that depend on a variety of different arguments. They are all formally introduced in the respective chapters and summarised in the nomenclature at the beginning of this thesis.

Special care has been taken to keep the notation consistent. We follow the Householder notation for matrices and vectors, meaning matrices are denoted by capital letters and vectors by lower roman letters. Greek letters, however, are reserved for sets of indices.

For two index sets σ and τ, we denote by $A_{\sigma,\tau}$ the submatrix of A that is induced by the column index set σ and the row index set τ. Usually, σ and τ induce a connected set of rows and columns. If not more than one $\mathcal{H}$-matrix is involved and the $\mathcal{H}$-matrix itself is rectangular, then σ will generally denote a row index set and τ will denote a column index set.

We will define a special version of the matrix T involved in the decomposition of Q, denoted by $\overline{T}$. $\overline{T}$ will always refer to this reduced version, and T will always refer to the original.

The letters U and V are reserved for the low-rank factorization of admissible blocks or, in some rare cases, for the factors used in the singular value decomposition.

$\mathcal{H}$-matrices are often visualized as block matrices. We will generally show inadmissible blocks in red, admissible blocks in green, and if the admissibility of a certain block is not relevant, it is shown in grey. Blocks containing only zeros are transparent. Any other intricacies of the notation or visualization will be explained when they are first introduced.

1.3 Structure of this Thesis

The main focus of this thesis is the QR decomposition of $\mathcal{H}$-matrices, which is reflected by its structure. After the introductory remarks of Chapter 1, we follow up with Chapter 2, which lays the groundwork for our results. This includes a short overview of some, mostly basic, linear algebra facts and some results regarding the QR decomposition of dense matrices in Section 2.1, an introduction to $\mathcal{H}$-matrices, including cost and storage estimates, in Section 2.2, some related matrix formats in Section 2.3, information about the programming library we use in Section 2.4 and our two model problems, interpolation and approximation using radial basis functions in Section 2.5 and the boundary element method in Section 2.6.

Chapter 3 deals with the theoretical framework of computing QR decompositions for $\mathcal{H}$-matrices. We begin by introducing already known approaches to compute the QR decomposition of an $\mathcal{H}$-matrix in Section 3.1 and present our own algorithm based on a block Householder approach in Section 3.2. We continue with a complexity analysis of this approach in Section 3.3, where we also give a short overview of the changes in this result if we had reused the original $\mathcal{H}$-matrix structure and give some thoughts about the application to rectangular matrices. We finish this chapter by providing some details about the implementation in Section 3.4.

In Chapter 4, we then present our numerical results in which we test the new block Householder approach in various settings, including our two model problems from Sections 2.5 and 2.6. In Chapter 5, we transfer some of our ideas regarding the hierarchical structures that arise during the computation of the QR decomposition to the LU decomposition and provide some numerical tests. Chapter 6 summarises our results and offers some concluding remarks.

Chapter 2

Preliminaries

This chapter aims to provide the necessary background information for our further work. Most of it is well known, although some results are not widely used or are presented in a slightly modified form. If a statement is important to us or if there is no proof for its exact formulation in the literature, we provide one. Otherwise, we refer to an appropriate source.

We begin with important results, definitions, and theorems for standard dense matrices. The main focus is on the QR decomposition, low-rank factorizations, and related theorems.

After that, we introduce $\mathcal{H}$-matrices mainly following the notation and definitions from [53], which differ from the standard approach, e.g. given in [43]. This deviation will lead to better readability of the proofs in Section 3.3. We then give a short overview of the actual construction of an $\mathcal{H}$-matrix for a given problem and show some general complexity estimates. Since we slightly modify or extend the results compared to the primary sources, we provide all the proofs, although they are mostly very similar to those in the literature.

We then present some related matrix formats and introduce our two model problems, interpolation and approximation using radial basis functions and the boundary element discretization of the 2D and the 3D Laplacian.

2.1 Linear Algebra

We start by recalling the definition of the rank of a matrix, which is equal to its maximal number of linearly independent columns.

Definition 2.1 (Range and rank) *Let $A \in \mathbb{R}^{m\times n}$. Then the range of A is defined by*

$$range(A) := \{Ax \in \mathbb{R}^m \mid x \in \mathbb{R}^n\}$$

and the rank of A by

$$rank(A) = dim\,(range\,(A))\,.$$

We continue with this well-known result.

Theorem 2.2 *Let $A \in \mathbb{R}^{m\times k}$. If $A = B \cdot C$ for $B \in \mathbb{R}^{m\times n}$ and $C \in \mathbb{R}^{n\times k}$, then*

$$rank(A) \leq \min\{rank(B), rank(C)\}.$$

If $A = B + C$ for $B, C \in \mathbb{R}^{m\times k}$, then

$$rank(A) \leq rank(B) + rank(C).$$

We further define matrices with additional properties.

Definition 2.3 (Diagonal matrix) *Let $A \in \mathbb{R}^{n\times n}$ be a square matrix, and let a_{ij} denote the entry in the i-th row and j-th column. If A satisfies*

$$i \neq j \Rightarrow\ a_{ij} = 0$$

for all $0 < i, j \leq n$, then A is a diagonal matrix. If A is diagonal and

$$a_{ii} = 1$$

holds for all $0 < i \leq n$, then A is the identity matrix of size n and we denote it by $A = I_n$.

Definition 2.4 (Trapezoidal and triangular matrix) *Let $A \in \mathbb{R}^{m\times n}$. If*

$$i > j \Rightarrow\ a_{ij} = 0$$

holds for all $0 < i \leq m$ and $0 < j \leq n$, then A is an upper trapezoidal matrix. If $m = n$, then A is an upper triangular matrix. If A^T is upper trapezoidal, then we call A lower trapezoidal and, in case $m = n$, lower triangular.

Definition 2.5 (Orthogonal matrix) *Let $Q \in \mathbb{R}^{n\times n}$ be a square matrix. Q is called orthogonal if*

$$Q^TQ = QQ^T = I_n,$$

where $I_n \in \mathbb{R}^{n\times n}$ is the identity matrix.

Definition 2.6 (Positive definite matrix) *Let $A \in \mathbb{R}^{n\times n}$ be a square matrix. A is called positive definite if*

$$x^T Ax > 0$$

holds for all $x \in \mathbb{R}^n \setminus \{0\}$.

Many matrices allow for decompositions into matrices with special properties.

Theorem 2.7 (LU decomposition, [75]) *Let $A \in \mathbb{R}^{n\times n}$ be an invertible matrix, meaning there is an $A^{-1} \in \mathbb{R}^{n\times n}$ such that*

$$A^{-1} \cdot A = I_n.$$

Then it can be factored as

$$A = PLU,$$

where $P \in \mathbb{R}^{n\times n}$ is a permutation matrix, meaning it has exactly one entry of 1 in each row and column and zeros elsewhere, $L \in \mathbb{R}^{n\times n}$ is lower triangular with 1 as every diagonal element and $U \in \mathbb{R}^{n\times n}$ is upper triangular.

Theorem 2.8 (Cholesky decomposition, [75, 49]) *Let $A \in \mathbb{R}^{n\times n}$ be a symmetric positive definite matrix. Then it can be factored as*

$$A = U^T U$$

where $U \in \mathbb{R}^{n\times n}$ is upper triangular with positive diagonal elements.

The LU decomposition can be computed in $2n^3/3$ floating point operations (flops), and the Cholesky decomposition can be computed in $n^3/3$ flops using the inherent symmetry [50].

Theorem 2.9 (QR decomposition, [30]) *Let $A \in \mathbb{R}^{m\times n}$ be a matrix with linearly independent columns. Then it can be factored as*

$$A = \widehat{Q}\widehat{R},$$

where $\widehat{Q} \in \mathbb{R}^{m\times n}$ has orthonormal columns and $\widehat{R} \in \mathbb{R}^{n\times n}$ is upper triangular with non-zero diagonal elements. This decomposition is called the reduced [80] or thin [33] QR decomposition. It can be extended to the full QR decomposition by adding further orthogonal columns $\widetilde{Q}$ to $\widehat{Q}$ to obtain a square orthogonal matrix $Q \in \mathbb{R}^{m\times m}$ and adding zero rows in $\widehat{R}$ to obtain a rectangular $R \in \mathbb{R}^{m\times n}$. Then we have the full QR factorization

$$A = QR = \begin{pmatrix} \widehat{Q} & \widetilde{Q} \end{pmatrix} \cdot \begin{pmatrix} \widehat{R} \\ 0 \end{pmatrix} = \widehat{Q}\widehat{R}.$$

There are different approaches to computing this decomposition, such as variants of the Gram-Schmidt orthogonalization [30], and Givens rotations [31, 33]. Our work will mainly be based on the Householder reflectors first introduced in [52] and defined by us in the upcoming Definition 2.13.

We now introduce the singular value decomposition (SVD).

Theorem 2.10 (Singular value decomposition, [32]) *Let $A \in \mathbb{R}^{m\times n}$. It can be factored as*

$$A = \widehat{U}\widehat{\Sigma}\widehat{V}^T,$$

where $\widehat{U} \in \mathbb{R}^{m\times k}$, $\widehat{V} \in \mathbb{R}^{n\times k}$ have orthonormal columns, $k = \min\{m, n\}$ and $\widehat{\Sigma} \in \mathbb{R}^{k\times k}$ is a diagonal matrix with non-negative diagonal elements in decreasing order. The diagonal entries σ_i of $\widehat{\Sigma}$ with $1 \leq i \leq k$ are called singular values, and the decomposition itself is called the compact singular value decomposition (SVD). By adding orthonormal columns $\widetilde{U}$ to $\widehat{U}$ or $\widetilde{V}$ to $\widehat{V}$ to get orthogonal matrices $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ and adding zero rows or columns to $\widehat{\Sigma}$ to get the rectangular matrix $\Sigma \in \mathbb{R}^{m\times n}$, it can be extended to the full SVD

$$A = U\Sigma V^T.$$

If only $k' < k$ singular values are positive, we can also compute a thin SVD

$$A = U_1\Sigma_1 V_1^T,$$

where $U_1 \in \mathbb{R}^{m\times k'}$, $V_1 \in \mathbb{R}^{n\times k'}$ contain only the first k' columns of U respectively V and $\Sigma_1 \in \mathbb{R}^{k'\times k'}$ is a diagonal matrix containing only the positive singular values, again in decreasing order.

The SVD can be computed in $21mn^2 + 8n^3$ flops [50]. The SVD plays an essential role in the context of finding a low-rank approximation for a given matrix. If the SVD of a matrix is known, the best approximation of any rank k is available directly.

Theorem 2.11 (Best approximation by SVD, [23]) *Let $A \in \mathbb{R}^{m\times n}$ be a matrix with rank k and thin singular value decomposition given by*

$$A = U\Sigma V^T$$

with $U \in \mathbb{R}^{m\times k}$, $V \in \mathbb{R}^{n\times k}$ and $\Sigma \in \mathbb{R}^{k\times k}$. Let $k' < k$ and let $U_1 \in \mathbb{R}^{m\times k'}$, $V_1 \in \mathbb{R}^{n\times k'}$ contain only the first k' columns of U or, respectively, V and $\Sigma_1 \in \mathbb{R}^{k'\times k'}$ be a diagonal matrix containing only the first k' singular values of A, again in decreasing order. Set

$$A^* = U_1\Sigma_1 V_1^T.$$

Then A^* is a best approximation of A among all matrices of rank k' with respect to the Frobenius norm, meaning

$$A^* = \underset{\substack{\widehat{A}\in\mathbb{R}^{m\times n}\\ rank(\widehat{A})=k'}}{\arg\min} \|A-\widehat{A}\|_F$$

with

$$\|A-A^*\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}-a^*_{ij}|^2},$$

as well as a best approximation of A among all matrices of rank k' with respect to the 2-norm, i.e.

$$A^* = \underset{\substack{\widehat{A}\in\mathbb{R}^{m\times n}\\ rank(\widehat{A})=k'}}{\arg\min} \|A-\widehat{A}\|_2$$

with

$$\|A-A^*\|_2 = \sigma_{k'+1}.$$

An extensive discussion of this and other low-rank approximation techniques can be found in [58]. We will take a brief look at a heuristic approach to the (non-optimal) low-rank factorization of a matrix in the upcoming Subsection 2.2.2 by introducing the adaptive cross approximation. Another direct consequence of Theorem 2.10 is that there exists a factorization for any matrix where the sizes of the factors depend on the rank of the matrix.

Theorem 2.12 *Let $A \in \mathbb{R}^{m\times n}$ be a matrix with $rank(A) = k$. Then there exist matrices $U \in \mathbb{R}^{m\times k}$ and $V \in \mathbb{R}^{n\times k}$ with*

$$A = UV^T.$$

Proof By Theorem 2.10 there exists a thin SVD of A

$$A = U_1\Sigma_1V_1^T,$$

where $U_1 \in \mathbb{R}^{m\times k}$, $V_1 \in \mathbb{R}^{n\times k}$ and $\Sigma_1 \in \mathbb{R}^{k\times k}$. Let u_i denote the columns of U_1 and v_i the columns of V for $1 \le i \le k$. By rewriting this using

$$U = [\sigma_1u_1\ \sigma_2u_2\ \dots\ \sigma_ku_k],$$
$$V = [v_1\ v_2\ \dots\ v_k],$$

we have shown the statement. ∎

Theorem 2.12 shows the connection between the rank of a matrix and the amount of unique data saved in it. If the rank of a matrix is low, then the same information could be saved without any loss simply by saving the low-rank factorization shown in Theorem 2.12. From now on, whenever we use matrices with a low rank, we assume that they are given in this factorized form.

This idea also extends to full-rank matrices that are numerically low-rank, meaning most singular values are close to zero. As shown in Theorem 2.11, the loss of information in the 2-norm is bounded by the first dropped singular value, and thus the low-rank approximation contains nearly all of the information of the original matrix.

Before we can go over the definition of $\mathcal{H}$-matrices, we need to make some additional statements regarding the computation of the QR decomposition of a matrix. At first, we introduce Householder reflectors and then show a special factorization of Q based on them recursively.

Definition 2.13 (Householder reflector) *Let $v \in \mathbb{R}^n$. Then v defines a Householder reflector by*

$$P = I_n - \frac{2}{v^T v} v v^T.$$

It can be shown that Householder reflectors are orthogonal and symmetric [33]. Furthermore, they can be used to transform one vector into another, arbitrarily defined vector of the same length. This allows us, among other things, to introduce zeros into vectors, and hence we can use them to compute the QR decomposition. The way in which the appropriate vector v can be found is shown in the following theorem.

Theorem 2.14 ([33]) *Let $x \in \mathbb{R}^n$ be an arbitrary vector with first entry $x_1 \in \mathbb{R}$ and $e_1 \in \mathbb{R}^n$ the first unit vector, i.e.*

$$e_1 = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Set

$$v = x + \mathit{sign}(x_1)\|x\|_2 e_1,$$

where

$$\mathit{sign}(x_1) = \begin{cases} 1 & x_1 \geq 0, \\ -1 & x_1 < 0, \end{cases}$$

and let P be the Householder reflector corresponding to v. Then

$$Px = -sign(x_1)\|x\|_2 e_1.$$

In the computation of v we chose addition instead of subtraction ($x - \text{sign}(x_1)\|x\|e_1$ would also eliminate all except the first entry of x) to avoid any cancellation error in the case where x_1 and $\|x\|_2$ are almost equal. Using repeated applications of Householder reflectors that are chosen such that entries below the diagonal are eliminated, the QR decomposition of a given matrix $A \in \mathbb{R}^{m\times n}$ can be computed in $2n^2(m - n/3)$ flops [33]. Furthermore, the computation of the Householder vector and the multiplication of the corresponding Householder reflector with another matrix are well known to be stable [48].

The following Theorem 2.15 is based on results in [74, 12, 24, 62] and presents a recursive block version of the Householder QR factorization, which will be the basis for our proposed QR factorization using (efficient but only approximate) matrix arithmetic provided by hierarchical matrices.

If a matrix $A = (x) \in \mathbb{R}^{m\times 1}$ consists of only a single column, then its (full) QR factorization is by Definition 2.13 and Theorem 2.14 given by the Householder reflector

$$Q = I_m - \frac{2}{v^T v} vv^T = I - YTY^T \quad \text{with } Y = v \in \mathbb{R}^{m\times 1},\ T = \frac{2}{v^T v} \in \mathbb{R}^{1\times 1},$$

where v is defined as in Theorem 2.14 and $R = -\text{sign}(x_1)\|x\|_2 e_1$ is an upper trapezoidal matrix, where $e_1 \in \mathbb{R}^m$ denotes the first unit vector. If A consists of more than one column, its QR factorization can be computed recursively, as given in the following Theorem 2.15. The respective matrix subblocks occurring in this Theorem are illustrated in Fig. 2.1.

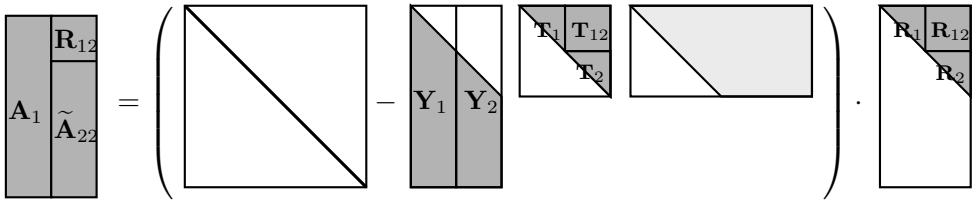


Figure 2.1. Illustration of the blocks in the recursive Householder block QR factorization.

Theorem 2.15 *For $m \geq n > 1$, let*

$$A = \begin{pmatrix} A_1 & A_2 \end{pmatrix} \in \mathbb{R}^{m\times n}$$

be split into two column blocks $A_1 \in \mathbb{R}^{m\times n_1}$, $A_2 \in \mathbb{R}^{m\times n_2}$ *with* $n_1 + n_2 = n$. *Let*

$$A_1 = Q_1 \begin{pmatrix} R_1 \\ 0 \end{pmatrix} \quad \textit{with} \quad Q_1 \in \mathbb{R}^{m\times m},\ R_1 \in \mathbb{R}^{n_1\times n_1},\ \begin{pmatrix} R_1 \\ 0 \end{pmatrix} \in \mathbb{R}^{m\times n_1}$$

be a QR decomposition of A_1, *and let*

$$\widetilde{A}_2 := \begin{pmatrix} \widetilde{A}_{12} \\ \widetilde{A}_{22} \end{pmatrix} := Q_1^T A_2 \quad \textit{with} \quad \widetilde{A}_{12} \in \mathbb{R}^{n_1\times n_2},\ \widetilde{A}_{22} \in \mathbb{R}^{\widetilde{m}\times n_2},$$

where $\widetilde{m} := m - n_1 \geq n_2$. *Then let*

$$\widetilde{A}_{22} = \widetilde{Q}_2 \begin{pmatrix} \widetilde{R}_2 \\ 0 \end{pmatrix} \quad \textit{with} \quad \widetilde{Q}_2 \in \mathbb{R}^{\widetilde{m}\times\widetilde{m}},\ \widetilde{R}_2 \in \mathbb{R}^{n_2\times n_2},\ \begin{pmatrix} \widetilde{R}_2 \\ 0 \end{pmatrix} \in \mathbb{R}^{\widetilde{m}\times n_2}$$

be a QR decomposition of $\widetilde{A}_{22}$. *Finally, we define*

$$Q = Q_1 \cdot Q_2 \in \mathbb{R}^{m\times m} \quad \textit{with} \quad Q_2 = \begin{pmatrix} I_{n_1} & 0 \\ 0 & \widetilde{Q}_2 \end{pmatrix} \in \mathbb{R}^{m\times m},$$
$$R = \begin{pmatrix} R_1 & \widetilde{A}_{12} \\ 0 & \widetilde{R}_2 \end{pmatrix} \in \mathbb{R}^{n\times n}.$$

Then the following holds:

1. $A = Q \begin{pmatrix} R \\ 0 \end{pmatrix}$ *is a QR decomposition of* A.

2. *Assuming that* Q_1, $\widetilde{Q}_2$ *have decompositions of the form*

$$Q_1 = I_m - Y_1 T_1 Y_1^T \ \textit{with}\ Y_1 \in \mathbb{R}^{m\times n_1},\ T_1 \in \mathbb{R}^{n_1\times n_1},$$
$$\widetilde{Q}_2 = I_{\widetilde{m}} - \widetilde{Y}_2 T_2 \widetilde{Y}_2^T \ \textit{with}\ \widetilde{Y}_2 \in \mathbb{R}^{\widetilde{m}\times n_2},\ T_2 \in \mathbb{R}^{n_2\times n_2}$$

for lower trapezoidal matrices Y_1, $\widetilde{Y}_2$ *and upper triangular matrices* T_1, T_2, *then it holds that*

$$Q = I_m - YTY^T$$

with lower trapezoidal $Y \in \mathbb{R}^{m\times n}$ *and upper triangular* $T \in \mathbb{R}^{n\times n}$ *given by*

$$Y = \begin{pmatrix} Y_1 & Y_2 \end{pmatrix} \quad \textit{with} \quad Y_2 = \begin{pmatrix} 0 \\ \widetilde{Y}_2 \end{pmatrix} \in \mathbb{R}^{m\times n_2},$$
$$T = \begin{pmatrix} T_1 & -T_1 Y_1^T Y_2 T_2 \\ 0 & T_2 \end{pmatrix}.$$

PROOF 1. The matrix Q is orthogonal since it is the product of two orthogonal matrices. The matrix R is upper triangular since $R_1, \widetilde{R}_{22}$ are upper triangular.

Furthermore, it holds that

$$\begin{aligned} QR &= Q_1 \cdot \begin{pmatrix} I_{n_1} & 0 \\ 0 & \widetilde{Q}_2 \end{pmatrix} \cdot \begin{pmatrix} R_1 & \widetilde{A}_{12} \\ 0 & \widetilde{R}_2 \\ 0 & 0 \end{pmatrix} = Q_1 \cdot \begin{pmatrix} R_1 & \widetilde{A}_{12} \\ 0 & \widetilde{A}_{22} \end{pmatrix} \\ &= \left(Q_1 \begin{pmatrix} R_1 \\ 0 \end{pmatrix} \quad Q_1 \widetilde{A}_2 \right) = \begin{pmatrix} A_1 & A_2 \end{pmatrix} = A. \end{aligned}$$

2. It holds that

$$\begin{aligned} I_m - YTY^T &= I_m - \begin{pmatrix} Y_1 & Y_2 \end{pmatrix} \cdot \begin{pmatrix} T_1 & -T_1 Y_1^T Y_2 T_2 \\ 0 & T_2 \end{pmatrix} \begin{pmatrix} Y_1^T \\ Y_2^T \end{pmatrix} \\ &= I_m - Y_1 T_1 Y_1^T - Y_2 T_2 Y_2^T + Y_1 T_1 Y_1^T Y_2 T_2 Y_2^T \\ &= (I_m - Y_1 T_1 Y_1^T)(I_m - Y_2 T_2 Y_2^T) \\ &= Q_1 \cdot \begin{pmatrix} I_{n_1} & 0 \\ 0 & \widetilde{Q}_2 \end{pmatrix} = Q_1 \cdot Q_2 = Q. \end{aligned}$$

■

Remark 2.16 Instead of computing the matrix product in the off-diagonal block T_{12} of

$$T = \begin{pmatrix} T_1 & T_{12} \\ 0 & T_2 \end{pmatrix} = \begin{pmatrix} T_1 & T_1 Y_1^T Y_2 T_2 \\ 0 & T_2 \end{pmatrix} \in \mathbb{R}^{(n_1+n_2)\times(n_1+n_2)},$$

we propose to only compute and store

$$\overline{T} := \begin{pmatrix} T_1 & \overline{T}_{12} \\ 0 & T_2 \end{pmatrix} := \begin{pmatrix} T_1 & Y_1^T Y_2 \\ 0 & T_2 \end{pmatrix} \in \mathbb{R}^{(n_1+n_2)\times(n_1+n_2)},$$

which we call the reduced form of T. $\overline{T}$ is less expensive to compute compared to T but allows to compute a matrix-matrix multiplication $T \cdot X$ (as needed in the computation of $\widetilde{A}_2$ in Theorem 2.15) at the same cost in view of

$$\begin{pmatrix} T_1 & T_1 Y_1^T Y_2 T_2 \\ 0 & T_2 \end{pmatrix} = \begin{pmatrix} T_1 & 0 \\ 0 & I \end{pmatrix} \cdot \begin{pmatrix} I & Y_1^T Y_2 \\ 0 & I \end{pmatrix} \cdot \begin{pmatrix} I & 0 \\ 0 & T_2 \end{pmatrix}$$

by storing and reusing the intermediate products $T_2 X_{21}$ and $T_2 X_{22}$ (imposing a 2×2 block structure on X that is compatible with the one of T).

In the case of dense matrices, the storage requirements of T and its reduced form $\overline{T}$ are the same. We will later store all matrices occurring in the QR factorization in a structured hierarchical matrix format. Then $\overline{T}$ is often not only advantageous concerning computational work but also storage cost since the additional matrix-matrix products to compute $T_{12} = T_1 Y_1^T Y_2 T_2$ compared to $\overline{T}_{12} = Y_1^T Y_2$ often increase the local ranks

of admissible matrix subblocks (or even require their refinement to smaller subblocks). To our knowledge, this reduced representation of T has not been used before in the literature. We will formalize this process as an algorithm for $\mathcal{H}$-matrices in Chapter 3.3. ♢

Note that, by construction, Y contains the vectors used to create all Householder reflectors used on the highest level of the recursion. Assuming one would have the result of the simple, non-block version of the QR decomposition by Householder reflectors, one could directly write down Y using the Householder vectors and the diagonal entries of T using the coefficients in the Householder reflectors. Furthermore, by this observation, the actual vertical splits used in the recursion are irrelevant for T as well. Independent of their choice, T is always the same.

The cost of computing T, as given in Theorem 2.15 when all Householder reflectors are known, can be bounded by $2mn^2$. The proof is very similar to the considerations for the block representation of Q, as given in [33], so we omit it here. Together with the cost of computing the Householder reflectors in the first place, the overall cost for the QR decomposition is bounded by $4mn^2$.

2.2 Hierarchical Matrices

2.2.1 Introduction

$\mathcal{H}$-matrices have been introduced about 20 years ago and their arithmetic has reached a somewhat mature state, see for example [36, 43] for detailed discussions on their construction and arithmetics.

Our goal of introducing a recursive Householder-based QR factorization in $\mathcal{H}$-matrix format requires a careful adaptation of the $\mathcal{H}$-matrix construction for two main reasons.

First, for the ($\mathcal{H}$-) matrix-matrix multiplication $A \cdot B$ of two $\mathcal{H}$-matrices A, B, their block structures must be compatible. Often, the $\mathcal{H}$-matrix structure for the product is fixed a priori, but here we want to be able to determine it in a suitable way "on-the-fly" based on the $\mathcal{H}$-matrix structures of the factors A, B. This is necessary since we do not know beforehand what structures may be suitable for the matrices Y, T, R representing the QR factors in Theorem 2.15.

Second, an $\mathcal{H}$-matrix is typically constructed by a (simultaneous) subdivision of the row and column index sets. However, the recursive block QR factorization is based on column-wise (only) subdivisions. Therefore, in the following description of a hierarchical

index set partitioning, we allow for "idle" steps, where an index set is not partitioned when going to the next level.

Our introduction of $\mathcal{H}$-matrices follows along the lines of [53]. For a (finite) index set I, $\#I$ denotes its number of elements. We use I for the row index set and J for the column index set of a matrix $A \in \mathbb{R}^{I \times J}$. When only the numbers n, m for rows/columns of a matrix are known, we write $A \in \mathbb{R}^{m \times n}$ or even use $A \in \mathbb{R}^{I \times n}$. For subsets $\sigma \subset I$ and $\tau \subset J$, the corresponding submatrix of A is denoted by $A_{\sigma,\tau}$. We call the set $\sigma \times \tau$ an (index) block and the submatrix $A_{\sigma,\tau}$ a matrix block. First, we recall the definition of a partition.

Definition 2.17 (Partition) *Let S be a set. Then a partition of S is a set of subsets $\mathcal{S}$ of S so that the following conditions are satisfied:*

1. $\forall\ S_1, S_2 \in \mathcal{S} :\ S_1 \cap S_2 = \emptyset \ \Leftrightarrow\ S_1 \neq S_2$,
2. $S = \bigcup_{S' \in \mathcal{S}} S'$,
3. $\forall\ S' \in \mathcal{S} :\ S' \neq \emptyset$.

Now we define hierarchical structures on the index sets I and J.

Definition 2.18 (Hierarchy of partitionings) *A sequence $(P_I^\ell)_{\ell=0}^L$ of partitions $P_I^\ell = \{I_0^\ell, \dots, I_{p^\ell}^\ell\}$, $\ell = 0, \dots, L$, for a finite set I is called a hierarchy of partitionings of I of depth L if*

$$P_I^0 = \{I\}$$

and for all $\ell \in \{0, \dots, L-1\}$ and $k \in \{0, \dots, p^\ell\}$

$$I_k^\ell = \bigcup_{j \in I_{sub}} I_j^{\ell+1} \text{ for a subset } I_{sub} \subset \{0, \dots, p^{\ell+1}\}$$

holds true. We further define the set of all subsets in a hierarchy of partitionings by

$$P_I^{hier} := P_I^{hier}\left(\left(P_I^\ell\right)_{\ell=0}^L\right) = \bigcup_{\ell \in \{0,\dots,L\}} P_I^\ell.$$

Let $\sigma, \sigma' \in P_I^{hier}$ with $\sigma' \subseteq \sigma$. Then we call σ' a descendant of σ and a non-trivial descendant if $\sigma' \subsetneq \sigma$. If $\sigma = I_k^\ell$ for some $\ell, k \in \mathbb{N}$, then σ has the level ℓ and is the uniquely defined ancestor of σ' on the ℓ-th level given by

$$anc_\ell(\sigma') = \sigma.$$

We call σ' a direct descendant of σ if $\sigma' = I_{k'}^{\ell+1}$ for some $k' \in \mathbb{N}$.

Although technically a set, as σ can (and will) have more than one level, we will nevertheless use the notation level(σ) to identify a suitable representative of this set of levels by choosing the highest level.

We give a simple example to illustrate the notation introduced in Definition 2.18.

Example 2.1 Let $I = \{0, 1, 2, 3\}$ and a sequence of partitions be given as specified in

level	partition	subsets
0	P_I^0	$I_0^0 = I$
1	P_I^1	$I_0^1 = \{0, 1\}, I_1^1 = \{2, 3\}$
2	P_I^2	$I_0^2 = \{0, 1\}, I_1^2 = \{2\}, I_2^2 = \{3\}$
3	P_I^3	$I_0^3 = \{0\}, I_1^3 = \{1\}, I_2^3 = \{2\}, I_3^3 = \{3\}$

Then,

$$P_I^{\text{hier}} = \{I, \{0, 1\}, \{2, 3\}, \{0\}, \{1\}, \{2\}, \{3\}\}\,.$$

The subset $\{0, 1\}$ has direct descendants $\{0, 1\}, \{0\}, \{1\}$, non-trivial direct descendants $\{0\}, \{1\}$ and (occurs on) levels 1 and 2, while the subset $\{2, 3\}$ has only direct descendants $\{2\}, \{3\}$ and (occurs on) a single level 1. ♢

We now construct a particular hierarchy of partitionings for the index set $I \times J$ based on hierarchies of partitionings of I and J. Our definition is more flexible than traditional definitions of block partitionings in which typically the direct descendants of blocks $\sigma \times \tau$ are given by the Cartesian product of the direct descendants of σ and τ, respectively. In our definition, other than direct descendants may be used as well.

Definition 2.19 (Hierarchy of block partitionings) *Let $(P_I^\ell)_{\ell=0}^{L_I}$ and $(P_J^\ell)_{\ell=0}^{L_J}$ be hierarchies of partitionings of the finite index sets I and J, respectively. The sequence of partitions $\left(P_{I\times J}^\ell\right)_{\ell=0}^{L}$ of $I \times J$ with $P_{I\times J}^\ell = \{b_1^\ell, \ldots, b_{q^\ell}^\ell\}, b_k^\ell \subset I \times J$, is called a hierarchy of block partitionings of $I \times J$ of depth L if the following two conditions are satisfied:*

1. *$\left(P_{I\times J}^\ell\right)_{\ell=0}^{L}$ is a hierarchy of partitionings of $I \times J$ according to Definition 2.18.*

2. *For every $\ell \in \{0, \ldots, L\}$, there exist $\ell' \in \{0, \ldots, L_I\}$ and $\ell'' \in \{0, \ldots, L_J\}$ such that*

$$P_{I\times J}^\ell = P_I^{\ell'} \times P_J^{\ell''}\,.$$

For a given $\sigma \in P_I^{hier}$, we define the set of levels of all $\tau \in P_J^{hier}$ with $\sigma \times \tau \in P_{I\times J}^{hier}$ by

$$\begin{aligned} L^{col}(\sigma, J) &:= L^{col}\left(\sigma, (P_J^\ell)_{\ell=0}^{L_J}, P_{I\times J}^{hier}\right) \\ &:= \left\{0 \le \ell \le L_J : \exists\, \tau \in P_J^\ell \text{ with } \sigma \times \tau \in P_{I\times J}^{hier}\right\}. \end{aligned}$$

Analogously, for a given $\tau \in P_J^{hier}$, we define

$$\begin{aligned} L^{row}(\tau, I) &:= L^{row}\left(\tau, (P_I^\ell)_{\ell=0}^{L_I}, P_{I\times J}^{hier}\right) \\ &:= \left\{0 \le \ell \le L_I : \exists\, \sigma \in P_I^\ell \text{ with } \sigma \times \tau \in P_{I\times J}^{hier}\right\}. \end{aligned}$$

We give an example of a hierarchy of block partitionings.

Example 2.2 Let I and its hierarchy of partitionings be given as in Example 2.1 and let $J = I$ with the same hierarchy of partitionings. We define the following hierarchy of block partitionings:

$$\begin{aligned} P_{I\times J}^0 &= P_I^0 \times P_J^0 = \{I \times J\}, \\ P_{I\times J}^1 &= P_I^0 \times P_J^1 = \{I \times \{0,1\}, I \times \{2,3\}\}, \\ P_{I\times J}^2 &= P_I^0 \times P_J^3 = \{I \times \{0\}, I \times \{1\}, I \times \{2\}, I \times \{3\}\}, \\ P_{I\times J}^3 &= P_I^1 \times P_J^3 = \{\{0,1\} \times \{0\}, \{0,1\} \times \{1\}, \ldots, \{2,3\} \times \{3\}\}. \end{aligned}$$

Then

$$\begin{array}{lll} L^{col}(\{0,1,2,3\}, J) = \{0,1,2,3\}, & L^{col}(\{0,1\}, J) = \{2,3\}, & L^{col}(\{0\}, J) = \emptyset, \\ L^{row}(\{0,1,2,3\}, I) = \{0\}, & L^{row}(\{0,1\}, I) = \{0\}, & L^{row}(\{0\}, I) = \{0,1,2\} \end{array}$$

holds true following Definition 2.19. ♢

We define a partition of $I \times J$ consisting of index blocks from different levels of a hierarchy of block partitionings. This partition will later determine the block structure of an $\mathcal{H}$-matrix.

Definition 2.20 ($\mathcal{H}$-block partition) *Let $\left(P_{I\times J}^\ell\right)_{\ell=0}^{L}$ be a hierarchy of block partitionings based on hierarchies of partitionings $(P_I^\ell)_{\ell=0}^{L_I}$, $(P_J^\ell)_{\ell=0}^{L_J}$. Then a partition $P_{I\times J}^{\mathcal{H}} = \{b_1, \ldots, b_q\}$ of $I \times J$ is called an $\mathcal{H}$-block partition if $b_i \in P_{I\times J}^{hier}$ for all $i = 1, \ldots, q$.*

We call $\tau \in P_J^{hier}$ a column cluster of $P_{I\times J}^{\mathcal{H}}$ if there exists a subset $\sigma \in P_I^{hier}$ with $\sigma \times \tau \in P_{I\times J}^{\mathcal{H}}$. A column cluster $\tau \in P_J^{hier}$ is called a column leaf cluster of $P_{I\times J}^{\mathcal{H}}$ if there exists no index block $\sigma' \times \tau' \in P_{I\times J}^{\mathcal{H}}$ with $\tau' \subsetneq \tau$. Analogously, we call $\sigma \in P_I^{hier}$ a row cluster of $P_{I\times J}^{\mathcal{H}}$ if there exists a subset $\tau \in P_J^{hier}$ with $\sigma \times \tau \in P_{I\times J}^{\mathcal{H}}$. A row cluster

$\tau \in P_J^{hier}$ is called a row leaf cluster of $P_{I\times J}^{\mathcal{H}}$ if there exists no index block $\sigma' \times \tau' \in P_{I\times J}^{\mathcal{H}}$ with $\sigma' \subsetneq \sigma$.

We further set

$$depth_{row}(P_{I\times J}^{\mathcal{H}}) = \max_{0\le k\le L_I} \left\{ k : \exists\ \sigma \times \tau \in P_{I\times J}^{\mathcal{H}} \text{ with } \sigma \in P_I^k, \tau \in P_J^{hier} \right\},$$
$$depth_{col}(P_{I\times J}^{\mathcal{H}}) = \max_{0\le k\le L_J} \left\{ k : \exists\ \sigma \times \tau \in P_{I\times J}^{\mathcal{H}} \text{ with } \sigma \in P_I^{hier}, \tau \in P_J^k \right\}$$

and define the depth of $P_{I\times J}^{\mathcal{H}}$ by

$$depth(P_{I\times J}^{\mathcal{H}}) = \max\left\{ depth_{row}(P_{I\times J}^{\mathcal{H}}), depth_{col}(P_{I\times J}^{\mathcal{H}}) \right\}.$$

Example 2.3 We continue our example and define the $\mathcal{H}$-block partition

$$P_{I\times J}^{\mathcal{H}} := \{I \times \{0,1\}, \{0,1\} \times \{2\}, \{0,1\} \times \{3\}, \{2,3\} \times \{2\}, \{2,3\} \times \{3\}\}.$$

$P_{I\times J}^{\mathcal{H}}$ has column leaf clusters $\{0,1\}, \{2\}$ and $\{3\}$, row leaf clusters $\{0,1\}$ and $\{2,3\}$ and furthermore

$$\mathrm{depth}_{\mathrm{row}}(P_{I\times J}^{\mathcal{H}}) = 2, \quad \mathrm{depth}_{\mathrm{col}}(P_{I\times J}^{\mathcal{H}}) = 3, \quad \mathrm{depth}(P_{I\times J}^{\mathcal{H}}) = 3. \qquad \diamondsuit$$

To ease the notational burden, we will make the following assumption:

Assumption A Let $P_{I\times J}^{\mathcal{H}} = \{b_1, \ldots, b_q\}$ be an $\mathcal{H}$-block partition of $I \times J$ based on the hierarchy of block partitionings $\left(P_{I\times J}^{\ell}\right)_{\ell=0}^{L}$ with hierarchy of partitionings $(P_I^{\ell})_{\ell=0}^{L_I}$ and $(P_J^{\ell})_{\ell=0}^{L_J}$. We assume that any given $\sigma \in P_I^{hier}$ and $\tau \in P_J^{hier}$ has at most two non-trivial direct descendants. Hence, a block $\sigma \times \tau \in P_{I\times J}^{hier}$ has zero, two or four non-trivial direct descendants.

This assumption does not limit the type of matrix we can use because we allow for just one direct descendant. For any given $\mathcal{H}$-block partition, the underlying hierarchy of (block) partitionings can be changed to satisfy this assumption at the cost of a higher depth.

We next introduce an admissibility function that maps each block in a hierarchy of block partitionings to $\{True, False\}$. The definition is motivated as follows: The admissibility function shall indicate whether the corresponding matrix block allows for a low-rank approximation. Since subblocks of a low-rank matrix are of low rank as well, any descendants of an admissible block shall be admissible as well.

Definition 2.21 (Admissibility function) *Let $P_{I\times J}^{hier}$ be all sets in a hierarchy of partitionings $\left(P_{I\times J}^{\ell}\right)_{\ell=0}^{L}$ of $I \times J$ for some index sets I, J. Then, a function*

$$adm : P_{I\times J}^{hier} \to \{True, False\}$$

is called an admissibility function if for all $b \in P_{I\times J}^{hier}$

$$adm(b) = True \quad \Rightarrow \quad adm(b_i) = True \; \forall \; b_i \in P_{I\times J}^{hier} \text{ with } b_i \subset b.$$

holds true. We will call a block $b \in P_{I\times J}^{hier}$ admissible if $adm(b) = True$ and inadmissible otherwise.

The following definition defines an $\mathcal{H}$-block partition to be consistent with the admissibility function if it is the coarsest possible partition made up of sets from $P_{I\times J}^{hier}$ in which all block clusters not on the highest level of the corresponding hierarchy of block partitionings are admissible.

Definition 2.22 (Consistent $\mathcal{H}$-block partition) *Let $P_{I\times J}^{\mathcal{H}}$ be an $\mathcal{H}$-block partition that is based on the hierarchy of block partitionings $\left(P_{I\times J}^{\ell}\right)_{\ell=0}^{L}$. We say that it is consistent with the admissibility function adm if all blocks $b \in P_{I\times J}^{\mathcal{H}}$ of the $\mathcal{H}$-block partition that have descendants $b^{des} \in P_{I\times J}^{hier}$ (in the hierarchy of block partitionings) with $b^{des} \subsetneq b$ are admissible and there is no admissible $b^{anc} \in P_{I\times J}^{hier}$ with $b \subsetneq b^{anc}$. We further define the near-field of $P_{I\times J}^{\mathcal{H}}$ by*

$$P_{I\times J}^{\mathcal{H},-} = P_{I\times J}^{\mathcal{H}} \cap P_{I\times J}^{L}$$

and the far-field by

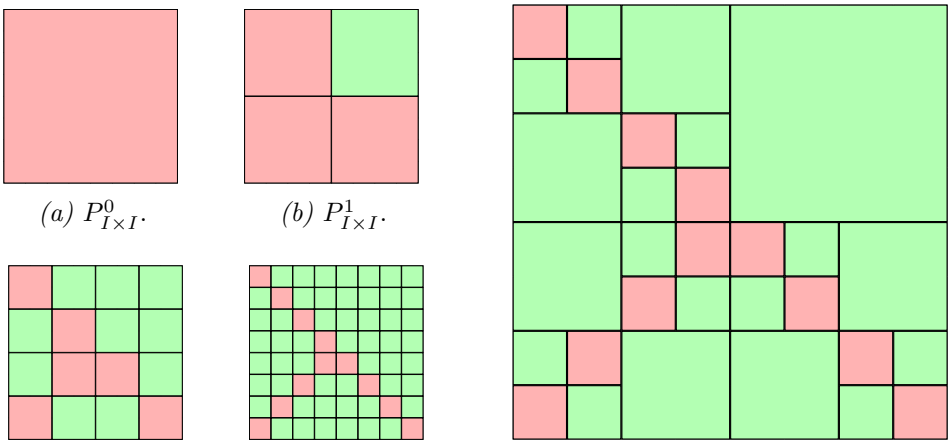
$$P_{I\times J}^{\mathcal{H},+} = P_{I\times J}^{\mathcal{H}} \setminus P_{I\times J}^{\mathcal{H},-}.$$

Blocks in $P_{I\times J}^{\mathcal{H},-}$ may be admissible or not, and we say that $P_{I\times J}^{\mathcal{H}}$ has the leaf size $n_{min} \in \mathbb{N}$ if

$$n_{min} \geq \max_{\sigma\times\tau \in P_{I\times J}^{\mathcal{H},-}} \{\min\{\#\sigma, \#\tau\}\}.$$

is satisfied.

Consistency of an $\mathcal{H}$-block partition $P_{I\times J}^{\mathcal{H}}$ with an admissibility function as defined in Definition 2.22 ensures that admissible blocks in $P_{I\times J}^{\mathcal{H}}$ are as large as possible. In particular, an admissibility function defines a unique consistent $\mathcal{H}$-block partition. We will call matrix blocks $A_{\sigma,\tau}$ admissible or inadmissible if the corresponding index block $\sigma \times \tau$ is admissible or inadmissible, respectively. The leaf size, as given in Definition 2.22, is



(a) $P^0_{I\times I}$. (b) $P^1_{I\times I}$. (c) $P^2_{I\times I}$. (d) $P^3_{I\times I}$. (e) $P^{\mathcal{H}}_{I\times I}$.

Figure 2.2. Illustration of an admissibility function *adm* on a hierarchy of block partitionings (a)-(d) and an $\mathcal{H}$-block partition (e) that is consistent with *adm*. Red index blocks are inadmissible, green index blocks are admissible.

the upper bound for the minimum number of rows or columns of each leaf. We now define the set of $\mathcal{H}$-matrices.

Definition 2.23 ($\mathcal{H}$-matrix) *Let $P^{\mathcal{H}}_{I\times J}$ be an $\mathcal{H}$-block partition of $I \times J$ that is consistent with the admissibility function adm, and let $k \in \mathbb{N}$. Then*

$$\mathcal{H}(P^{\mathcal{H}}_{I\times J}, k, adm) := \{B \in \mathbb{R}^{I\times J} \mid rank(B|_b) \leq k \text{ for all admissible blocks } b \in P^{\mathcal{H}}_{I\times J}\}$$

defines the set of $\mathcal{H}$-matrices with respect to $P^{\mathcal{H}}_{I\times J}$, adm and k.

Admissible matrix blocks $B|_b$ with $b = \sigma \times \tau$ can be represented in factored form

$$B|_b = UV^T \qquad \text{with } U \in \mathbb{R}^{\sigma\times k'},\ V \in \mathbb{R}^{\tau\times k'} \text{ for some } k' \leq k.$$

This factored representation is a key ingredient for the efficient arithmetic of $\mathcal{H}$-matrices, especially if k is small compared to $\#\sigma, \#\tau$. Matrix blocks $\sigma \times \tau$ in the $\mathcal{H}$-block partition $P^{\mathcal{H}}_{I\times J}$ that have no (non-trivial) descendants in $P^{\text{hier}}_{I\times J}$ may either be admissible or inadmissible. Depending on the rank k and their size $\#\sigma \cdot \#\tau$, such admissible blocks may be stored more efficiently in explicit (as opposed to factored) form.

In the literature, an admissibility function is sometimes only defined on the blocks of an $\mathcal{H}$-block partition and not on the entire hierarchy of block partitionings as we did in Definition 2.21. We will need this extended definition in order to define admissibility functions (and consistent $\mathcal{H}$-block partitions) for sums and products of $\mathcal{H}$-matrices in the upcoming Definitions 2.24 and 2.25.

We first recall an observation for the addition of low-rank matrices. Let $A, B \in \mathbb{R}^{n\times m}$ have respective ranks k_A, k_B and be given in factored forms

$$A = U_A \cdot V_A^T, \quad B = U_B \cdot V_B^T.$$

Then the sum $A+B$ has at most rank k_A+k_B and can be represented in factored form

$$A + B = [U_A\ U_B] \cdot [V_A\ V_B]^T.$$

Even though the ranks are added, the rank of the sum may still be small compared to the sizes of A and B, hence suggesting that the sum of two admissible blocks shall lead to an admissible block, which motivates the following definition. Note that for ease of notation we set

$$\text{True} > \text{False}.$$

Definition 2.24 (Matrix sum admissibility) *Let adm_A, adm_B be admissibility functions for the same hierarchy of block partitionings $\left(P_{I\times J}^{\ell}\right)_{\ell=0}^{L}$. Then $adm_C(b)$ is called the induced matrix sum admissibility function w.r.t. adm_A and adm_B if for all blocks $b = \sigma \times \tau \in P_{I\times J}^{hier}$*

$$adm_C(b) = \begin{cases} True & : \quad adm_A(b) = adm_B(b) = True, \\ False & : \quad else \end{cases}$$

holds true and a partially induced matrix sum admissibility function w.r.t. adm_A and adm_B if for all blocks $b = \sigma \times \tau \in P_{I\times J}^{hier}$

$$adm_C(b) \leq \begin{cases} True & : \quad adm_A(b) = adm_B(b) = True, \\ False & : \quad else \end{cases}$$

holds true.

The admissibility function from Definition 2.24 is actually an admissibility function as defined in Definition 2.21 because for every $b \in P_{I\times J}^{\text{hier}}$ with

$$\text{adm}_C(b) = \text{True}$$

we have by definition

$$\text{adm}_A(b) = \text{adm}_B(b) = \text{True}$$

and thus also

$$\text{adm}_A(b_i) = \text{adm}_B(b_i) = \text{True}\ \forall\ b_i \in P_{I\times J}^{\text{hier}} \text{ with } b_i \subset b.$$

The latter means, again by definition of adm_C, that

$$\mathrm{adm}(b_i) = \text{True } \forall\ b_i \in P^{\mathrm{hier}}_{I\times J} \text{ with } b_i \subset b.$$

According to Definition 2.22, an admissibility function yields a consistent $\mathcal{H}$-block partition. Hence, admissibility functions adm_A, adm_B and the resulting sum admissibility adm_C are associated with respective consistent $\mathcal{H}$-block partitions $P^{\mathcal{H}}_A$, $P^{\mathcal{H}}_B$ and $P^{\mathcal{H}}_C$, and we also say that the $\mathcal{H}$-block partition $P^{\mathcal{H}}_C$ is induced by summation of (consistent) matrices with $\mathcal{H}$-block partitions $P^{\mathcal{H}}_A$, $P^{\mathcal{H}}_B$.

To define an admissibility function that is induced by the product of two $\mathcal{H}$-matrices $A \in \mathbb{R}^{I\times J}, B \in \mathbb{R}^{J\times K}$, we first have to ensure that their $\mathcal{H}$-block partitions are compatible for multiplication, i. e. for each $\sigma \in P^{\mathrm{hier}}_I$ and $\tau \in P^{\mathrm{hier}}_K$, there must exist levels ℓ_I, ℓ_J, ℓ_K such that $\sigma \in P^{\ell_I}_I, \tau \in P^{\ell_K}_K$ as well as $P^{\ell_I}_I \times P^{\ell_J}_J \in P^{\mathrm{hier}}_{I\times J}$ and $P^{\ell_J}_J \times P^{\ell_K}_K \in P^{\mathrm{hier}}_{J\times K}$. Then the matrix product may be computed blockwise as a sum of matrix products

$$C_{\sigma,\gamma} = \sum_{\tau\in P^{\ell_J}_J} A_{\sigma,\tau} \cdot B_{\tau,\gamma}.$$

An intermediate product $A_{\sigma,\tau} \cdot B_{\tau,\gamma}$ has (at most) rank k if at least one of its factors has (at most) rank k, meaning the product is admissible if at least one of its factors is admissible. If we combine this with our motivation for the definition of an induced sum admissibility, then the block $C_{\sigma,\gamma}$ should be admissible if, for each of its intermediate products, at least one factor is admissible. However, the levels ℓ_I, ℓ_J, ℓ_K that were used in the above illustration may not be uniquely determined, and different levels $\ell'_I, \ell'_J, \ell'_K$ may yield a different result, as illustrated in Figure 2.3. In this example, we have $\ell'_J < \ell_J$. In the summation over $\tau \in^{\ell'_J}$, all intermediate products and hence the final product are inadmissible, whereas in the summation over $\tau \in P^{\ell_J}_J$, all intermediate products and hence the final product are admissible. We will hence base the definition of the product admissibility function on the highest possible level ℓ_J which is available to define the product in a given matrix block.

Definition 2.25 (Matrix product admissibility) *Let $P^{\mathcal{H}}_{I\times J}$ and $P^{\mathcal{H}}_{J\times K}$ be $\mathcal{H}$-block partitions that are consistent with admissibility functions adm_A and adm_B, respectively, and that are based on (block hierarchies of partitionings resulting from) the same hierarchies of partitionings $(P^\ell_I)^{L_I}_{\ell=0}$, $(P^\ell_J)^{L_J}_{\ell=0}$ and $(P^\ell_K)^{L_K}_{\ell=0}$. For $\sigma\times\gamma \in P^{hier}_{I\times K}$, let*

$$L^{intersec}(\sigma,\gamma,J) := L^{col}(\sigma,(P^\ell_J)^{L_J}_{\ell=0}, P^{hier}_{I\times J}) \cap L^{row}(\gamma,(P^\ell_J)^{L_J}_{\ell=0}, P^{hier}_{J\times K})$$

be the set of all intersecting levels for a block $\sigma\times\gamma \in P^{hier}_{I\times K}$. If $L^{intersec}(\sigma,\gamma,J) = \emptyset$, then the $\mathcal{H}$-block partitions $P^{\mathcal{H}}_{I\times J}$, $P^{\mathcal{H}}_{J\times K}$ are not compatible for multiplication. Otherwise, define the maximum

$$L^{max}(\sigma,\gamma,J) := \max\{\ell \mid \ell \in L^{intersec}(\sigma,\gamma,J)\}.$$

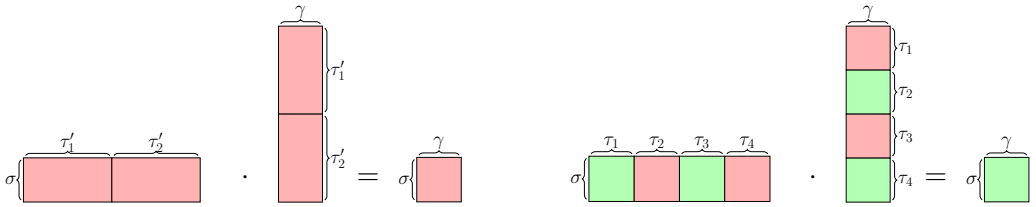


Figure 2.3. Computing the $\mathcal{H}$-matrix product $A_1 \cdot A_2$ for two $\mathcal{H}$-matrices A_1, A_2 on different levels of the hierarchy of block partitions. With $\tau_1' = \tau_1 \cup \tau_2$, $\tau_2' = \tau_3 \cup \tau_4$, we may assume that all τ_i' occur (only) on level ℓ, whereas the τ_j occur (only) on level $\ell + 1$ in the hierarchy of partitionings of J. Hence, $\{\ell, \ell+1\} \subset L^{col}(\sigma, J)$ as well as $\{\ell, \ell+1\} \subset L^{row}(\gamma, J)$. On the left, the summation is performed over the blocks in P_J^ℓ, whereas on the right, we sum over the blocks in $P_J^{\ell+1}$.

Then adm_C is called an induced matrix product admissibility function w.r.t. adm_A and adm_B if for all blocks $b = \sigma \times \gamma \in P_{I\times K}^{hier}$

$$adm_C(b) = \begin{cases} True & : \forall\, \tau \in P_J^{L^{max}(\sigma,\gamma,J)} : & adm_A(\sigma \times \tau) = True \\ & & or\ adm_B(\tau \times \gamma) = True, \\ False & : else & \end{cases}$$

holds true and a partially induced matrix product admissibility function w.r.t. adm_A and adm_B if for all blocks $b = \sigma \times \gamma \in P_{I\times K}^{hier}$

$$adm_C(b) \leq \begin{cases} True & : \forall\, \tau \in P_J^{L^{max}(\sigma,\gamma,J)} : & adm_A(\sigma \times \tau) = True \\ & & or\ adm_B(\tau \times \gamma) = True, \\ False & : else & \end{cases}$$

holds true.

Analogous to the induced matrix sum admissibility, the induced matrix product admissibility is also an admissibility condition as given by Definition 2.21. For every $\sigma \times \gamma \in P_{I\times K}^{\text{hier}}$ with

$$\text{adm}_C(\sigma \times \gamma) = \text{True}$$

either

$$\text{adm}_A(\sigma \times \tau) = \text{True or } \text{adm}_B(\tau \times \gamma) = \text{True}$$

holds $\forall\, \tau \in P_J^{L^{max}(\sigma,\gamma,J)}$. Hence, because adm_A and adm_B are admissibility functions as given in Definition 2.21, we can also find for $\sigma' \subseteq \sigma$, $\gamma' \subseteq \gamma$ with $\sigma' \times \gamma' \in P_{I\times K}^{hier}$ a

suitable τ' with

$$\mathrm{adm}_A(\sigma' \times \tau') = \text{True or } \mathrm{adm}_B(\tau' \times \gamma') = \text{True}.$$

By this we have

$$\mathrm{adm}_C(\sigma' \times \gamma') = \text{True } \forall\ \sigma' \times \gamma' \in P^{hier}_{I\times K} \text{ with } \sigma' \times \gamma' \subset \sigma \times \gamma.$$

Naturally, for any induced or partially induced matrix product admissibility function $\mathrm{adm}_C(\cdot)$ there is a consistent $\mathcal{H}$-block partition $P^{\mathcal{H}}_{I\times K}$ for the matrix product of matrices $A \in P^{\mathcal{H}}_{I\times J}$ and $B \in P^{\mathcal{H}}_{J\times K}$.

The induced admissibility function for the result of any operation that involves more than one product or sum can be found by consecutively applying Definitions 2.24 and 2.25. We will simplify the notation and call any admissibility function that was derived in such a way an induced admissibility function. Furthermore, we will sometimes call an induced admissibility function *fully* induced to differentiate it from partially induced admissibility functions.

The $\mathcal{H}$-matrix addition or multiplication may now be summarised in a greatly simplified manner as follows: Based on the $\mathcal{H}$-block partitions and admissibility functions of $\mathcal{H}$-matrices A and B, determine an $\mathcal{H}$-block partition that is consistent with an induced sum or product admissibility function. For an inadmissible block in this resulting $\mathcal{H}$-block partition, compute the exact sum or product. For an admissible block in this resulting $\mathcal{H}$-block partition, compute the exact sum or product in factored form and then truncate it to a given maximum rank (or relative accuracy). We refer to related work [36, 46], which considers suitable structures for $\mathcal{H}$-matrix products and derives complexity bounds based on notions of sparsity and idempotency.

Our definitions of induced sum and product admissibility functions are heuristic in the sense that a block deemed admissible (since it is computed as the sum of admissible blocks) may in fact have a rather large rank and incur truncation errors when truncated to a given maximum rank. On the other hand, a block that has been deemed inadmissible may have a low rank even if it is the sum of full-rank blocks. However, this is considered to be unlikely to happen in actual applications. Hence, the induced sum and product admissibility functions have been designed such that the resulting $\mathcal{H}$-block partition consists of (largest possible) blocks, which gives us reason to believe that they may be well approximated with low-rank data.

As illustrated in Figure 2.4, the number of inadmissible blocks in the induced matrix product admissibility function - and thus its likely data sparsity - depends mainly on the locations of the inadmissible blocks in the factors and not simply on their number. This is very similar to analogous considerations in the context of sparse matrices and leads to the same consequence. Only using data-sparse matrices in an arithmetic operation does

not mean that its result is data-sparse again. This has far-reaching consequences and is the main reason why the QR decomposition of some $\mathcal{H}$-matrices cannot be efficiently computed.

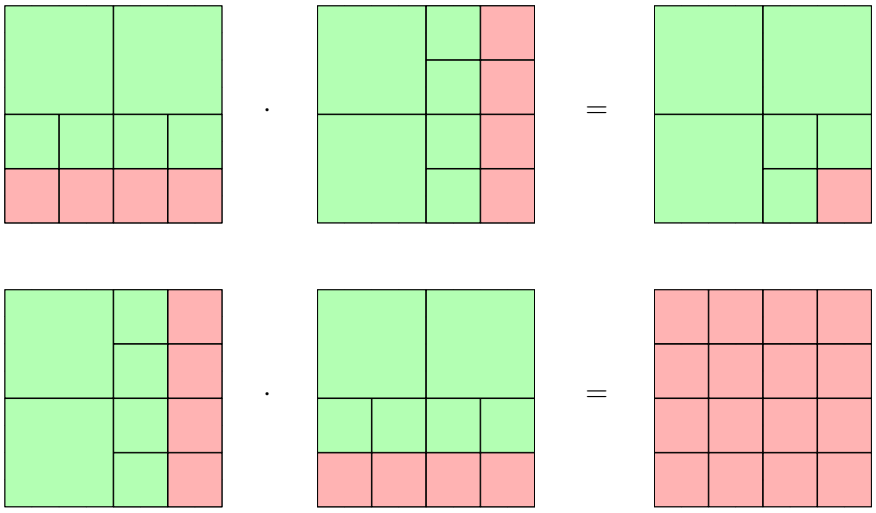


Figure 2.4. The induced matrix product admissibility function is highly dependent on the $\mathcal{H}$-matrix structure of its factors.

We have derived these induced sum and product admissibility functions to be used for all intermediate matrix products and sums in the $\mathcal{H}$-QR factorization based on Theorem 2.15. However, they may also be useful for other matrix operations, e. g. for computing suitable $\mathcal{H}$-block partitions for the LU factors in an $\mathcal{H}$-LU factorization, which so far in the literature mostly inherit the $\mathcal{H}$-block partition of the original matrix A. A notable example is [69], where a novel compressed matrix format is proposed that combines an adaptive hierarchical partitioning of the matrix with low-rank approximation. We will also briefly review the problem of finding a suitable $\mathcal{H}$-matrix structure for the LU factors in an $\mathcal{H}$-LU factorization in Chapter 5.

In both Definition 2.24 and 2.25, we have also defined the partially induced admissibility function. Using a partially induced admissibility function for the result of an operation ensures that no inadmissible block that gets calculated has to be turned into an admissible block so that it fits the hierarchical structure of the result. However, admissible blocks can be turned into inadmissible blocks. Hence, any partially induced admissibility function in a given context works as an upper bound on the number and position of inadmissible blocks when using the induced admissibility function. This will be essential for our cost analysis in Chapter 3.3.

We also need some further definitions for this cost analysis. Let $A \in \mathcal{H}(P^{\mathcal{H}}_{I\times J}, k, \mathrm{adm})$ be an $\mathcal{H}$-matrix based on the hierarchy of block partitionings $\left(P^{\ell}_{I\times J}\right)_{\ell=0}^{L}$ and let $\sigma \times \tau$ and $\sigma' \times \tau'$ be two blocks in $P^{hier}_{I\times J}$ but not necessarily in $P^{\mathcal{H}}_{I\times J}$. Then both blocks induce

submatrices in A, denoted by $A_{\sigma\times\tau}$ and $A_{\sigma'\times\tau'}$. Using the relative position of these submatrices, we can define an order on the sets in P_I^{hier} and P_J^{hier}.

Definition 2.26 (Order of blocks and distance) *Let $P_{I\times J}^{\mathcal{H}}$ be an $\mathcal{H}$-block partition that is based on the hierarchy of block partitionings $\left(P_{I\times J}^{\ell}\right)_{\ell=0}^{L}$, where $(P_I^{\ell})_{\ell=0}^{L_I}$ is the hierarchy of partitionings of the row index set, $(P_J^{k})_{\ell=0}^{L_J}$ is the hierarchy of partitionings of the column index set, and $A \in \mathcal{H}(P_{I\times J}^{\mathcal{H}}, k, adm_A)$ is an $\mathcal{H}$-matrix. Assume that the index sets I and J correspond to ordered rows and columns in A, meaning that the i-th row of A is induced by the index i and the j-th column of A is induced by j. For $\sigma, \sigma' \in P_I^{hier}$ with $\sigma \cap \sigma' = \emptyset$, we define*

$$\begin{aligned}\sigma <_{(A,row)} \sigma' &\;\Leftrightarrow\; \max\{i : i \in \sigma\} < \min\{j : j \in \sigma'\},\\ \sigma \leq_{(A,row)} \sigma' &\;\Leftrightarrow\; \max\{i : i \in \sigma\} \leq \max\{j : j \in \sigma'\},\end{aligned}$$

and for $\tau, \tau' \in P_J^{hier}$ with $\tau \cap \tau' = \emptyset$, we define

$$\begin{aligned}\tau <_{(A,col)} \tau' &\;\Leftrightarrow\; \max\{i : i \in \tau\} < \min\{j : j \in \tau'\},\\ \tau \leq_{(A,col)} \tau' &\;\Leftrightarrow\; \max\{i : i \in \tau\} \leq \max\{j : j \in \tau'\}.\end{aligned}$$

If the context is clear, we will write $<$ instead of $<_{(A,col)}$ and $<_{(A,row)}$ or $\leq$ instead of $\leq_{(A,col)}$ and $\leq_{(A,row)}$. This defines a distance between two column index sets $\tau, \tau' \in P_J^{\ell}$ of the same level ℓ with $\tau < \tau'$ by

$$dist_{row;A}(\tau, \tau') = \#\{\sigma \times \hat{\tau} \in P_{I\times J}^{hier} :\; level(\hat{\tau}) = \ell \;\wedge\; \tau < \hat{\tau} < \tau'\}$$

for some $\sigma \in P_I^{hier}$ with $\sigma \times \tau \in P_{I\times J}^{hier}$ and between two row index sets $\sigma, \sigma' \in P_I^{\ell}$ of the same level ℓ with $\sigma < \sigma'$ by

$$dist_{col;A}(\sigma, \sigma') = \#\{\hat{\sigma} \times \tau \in P_{I\times J}^{hier} :\; level(\hat{\sigma}) = \ell \;\wedge\; \sigma < \hat{\sigma} < \sigma'\}$$

for some $\tau \in P_J^{hier}$ with $\sigma \times \tau \in P_{I\times J}^{hier}$. Note that the actual choice of σ in the first case is irrelevant as by Definition 2.19 for any $\widetilde{\sigma}$ with $\widetilde{\sigma} \times \tau \in P_{I\times J}^{hier}$

$$\sigma \times \hat{\tau} \in P_{I\times J}^{hier} \Leftrightarrow \widetilde{\sigma} \times \hat{\tau} \in P_{I\times J}^{hier}$$

holds true. An equivalent statement is true for the second case. If again the context is clear, we will shorten our notation and write

$$dist_A(\tau, \tau') := dist_{row;A}(\tau, \tau')$$

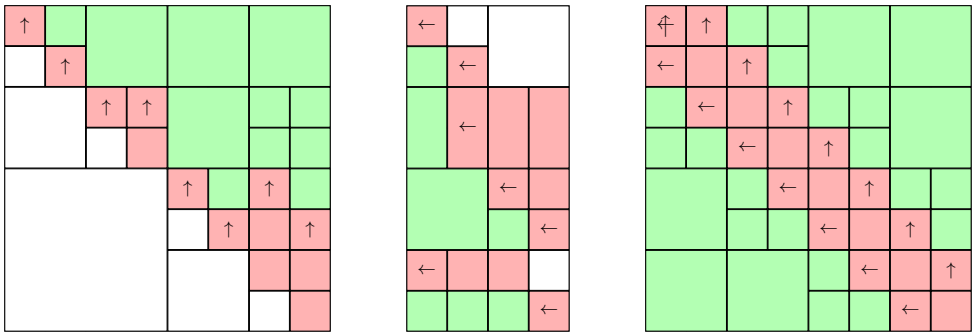
or

$$dist_A(\sigma, \sigma') := dist_{col;A}(\sigma, \sigma').$$

Before we continue, we introduce a third type of block. Until now, we have admissible

blocks given by their low-rank factorization and inadmissible blocks given by full-rank matrices. However, some blocks will consist only of zeros, especially if we work with upper or lower triangular $\mathcal{H}$-matrices. One could see them as admissible blocks with rank 0, but that would noticeably worsen our later cost estimates. Hence, we differentiate them further and any block consisting only of zeros will be called a zero block, and all other blocks will be called non-zero from now on.

In the upcoming Definition 2.27, we will use the order defined in Definition 2.26 to describe admissibility conditions and corresponding $\mathcal{H}$-matrices that exhibit a certain orientation, meaning inadmissible blocks are followed upon by other inadmissible blocks either in the same row or in the same column. This will be crucial in characterizing the structure of $\mathcal{H}$-matrices obtained from LU or QR decompositions. Examples for this can be seen in Figure 2.5.



(a) *(b)* *(c)*

Figure 2.5. The matrix in (a) is column-wise oriented, the matrix in (b) is row-wise oriented, the lower triangular part of the matrix in (c) is row-wise oriented, and the upper triangular part of the matrix in (c) is column-wise oriented. Outer blocks on the highest level that correspond to a row-wise orientation are marked with an arrow pointing left while those that correspond to a column-wise orientation are marked with an arrow pointing up. Note that for visualization purposes, the $\mathcal{H}$-matrices (and thus the corresponding $\mathcal{H}$-block partitions) are shown although the orientation is defined on the corresponding hierarchy of block partitionings.

Definition 2.27 (Oriented admissibility functions) *Let $P^{\mathcal{H}}_{I\times J}$ be an $\mathcal{H}$-block partition based on the hierarchy of block partitionings $\left(P^{\ell}_{I\times J}\right)_{\ell=0}^{L}$, where $(P^{\ell}_{I})_{\ell=0}^{L_I}$ is the hierarchy of partitionings of the row index set and $(P^{k}_{J})_{\ell=0}^{L_J}$ is the hierarchy of partitionings of the column index set. Let adm be an admissibility function (not necessarily with a consistent $\mathcal{H}$-block partition) of the hierarchy of block partitionings $\left(P^{\ell}_{I\times J}\right)_{\ell=0}^{L}$ and $A \in \mathcal{H}(P^{\mathcal{H}}_{I\times J}, k, adm)$ an $\mathcal{H}$-matrix. We will call adm row-wise oriented w.r.t. the*

$\mathcal{H}$-matrix A if for all $\sigma \times \tau \in P_{I\times J}^{hier}$

$$\begin{gathered} adm(\sigma \times \tau) = False \\ \Rightarrow \\ adm(\sigma \times \tau') = False \; \forall \; \tau' \in P_J^{level(\tau)} \text{ with } \tau' > \tau \wedge \text{block induced by } \sigma \times \tau' \text{ in } A \\ \text{is non-zero} \end{gathered}$$

holds true. It is column-wise oriented w.r.t. the $\mathcal{H}$-matrix A if for all $\sigma \times \tau \in P_{I\times J}^{hier}$

$$\begin{gathered} adm(\sigma \times \tau) = False \\ \Rightarrow \\ adm(\sigma' \times \tau) = False \; \forall \; \sigma' \in P_I^{level(\sigma)} \text{ with } \sigma' > \sigma \wedge \text{block induced by } \sigma' \times \tau \text{ in } A \\ \text{is non-zero} \end{gathered}$$

holds true. The $\mathcal{H}$-matrix A will be called row-wise oriented in the first and column-wise oriented in the second case. An inadmissible block $\sigma \times \tau \in P_{I\times J}^{hier}$ that satisfies

$$adm(\sigma \times \tau') = True \; \forall \; \tau' \in P_J^{level(\tau)} \text{ with } \tau' < \tau$$

in the first and

$$adm(\sigma' \times \tau) = True \; \forall \; \sigma' \in P_I^{level(\sigma)} \text{ with } \sigma' < \sigma$$

in the second case is then called an outer block (of level level(τ) or level(σ), respectively).

We will need additional parameters to calculate the storage and computational costs associated with an $\mathcal{H}$-matrix. We define the bandwidth and sparsity of an $\mathcal{H}$-matrix. The latter is a well-known constant in the context of $\mathcal{H}$-matrices, and the first extends the idea of a (sparse matrix) bandwidth to $\mathcal{H}$-matrices in a new meaning. It will be relevant for estimating how costly a QR decomposition for a given $\mathcal{H}$-matrix is. Both parameters are illustrated in Figure 2.6. We start with the sparsity constant.

Definition 2.28 ($\mathcal{H}$-matrix sparsity constant) *Let $P_{I\times J}^{\mathcal{H}}$ be an $\mathcal{H}$-block partition that is based on the hierarchy of block partitionings $\left(P_{I\times J}^{\ell}\right)_{\ell=0}^{L}$, where $(P_I^{\ell})_{\ell=0}^{L_I}$ is the hierarchy of partitionings of the row index set, $(P_J^{k})_{\ell=0}^{L_J}$ is the hierarchy of partitionings of the column index set and $A \in \mathcal{H}(P_{I\times J}^{\mathcal{H}}, k, adm_A)$ is an $\mathcal{H}$-matrix. By*

$$\begin{aligned} C_{sp,c}(\tau) &:= \#\{\sigma \in P_I^{hier} : \sigma \times \tau \in P_{I\times J}^{\mathcal{H}}\}, \\ C_{sp,r}(\sigma) &:= \#\{\tau \in P_J^{hier} : \sigma \times \tau \in P_{I\times J}^{\mathcal{H}}\} \end{aligned}$$

for $\sigma \in P_I^{hier}$ and $\tau \in P_J^{hier}$ we describe how often σ or τ appear as row or column index

set in the $\mathcal{H}$-block partition. By

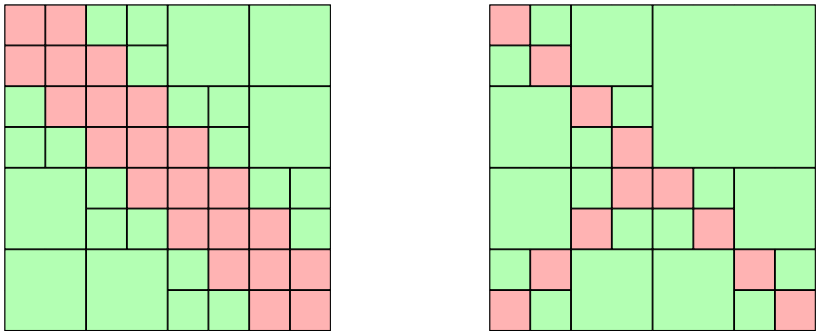
$$C_{sp}(P^{\mathcal{H}}_{I\times J}) := \max\left\{ \max_{\tau\in P_J^{hier}} C_{sp,c}(\tau), \max_{\sigma\in P_I^{hier}} C_{sp,r}(\sigma) \right\}$$

we define the sparsity constant of $P^{\mathcal{H}}_{I\times J}$.

Note here that two different definitions of the sparsity constant C_{sp} exist in the literature that differ slightly. Rewritten to our notation, the alternative is given by

$$C^{\ell,*}_{sp,c}(\tau) = \#\{\sigma \in P_I^\ell : \ \exists\ b \in P^{\mathcal{H}}_{I\times J} \text{ with } \sigma\times\tau \supseteq b\}$$

and accordingly for the rows. We will always use C_{sp} as an upper bound for the number of admissible blocks in a given block row or block column of the corresponding $\mathcal{H}$-block partition, hence we use the variant given in Definition 2.28. We continue with the bandwidth constant.



(a) $C_{bw} = 3$, $C_{sp} = 6$. *(b) $C_{bw} = 8$, $C_{sp} = 4$.*

Figure 2.6. Two $\mathcal{H}$-matrices and their bandwidth C_{bw} (assuming $\mathrm{adm}_b = \mathrm{adm}_A$ following the notation of Definition 2.29) and sparsity constant C_{sp}.

Definition 2.29 ($\mathcal{H}$-matrix bandwidth constant) *Let $P^{\mathcal{H}}_{I\times J}$ be an $\mathcal{H}$-block partition based on the hierarchy of block partitionings $\left(P^\ell_{I\times J}\right)_{\ell=0}^{L}$, where $(P_I^\ell)_{\ell=0}^{L_I}$ is the hierarchy of partitionings of the row index set, $(P_J^k)_{\ell=0}^{L_J}$ is the hierarchy of partitionings of the column index set, and $A \in \mathcal{H}(P^{\mathcal{H}}_{I\times J}, k, adm_A)$ is an $\mathcal{H}$-matrix. Let adm_b be an admissibility function corresponding to the same hierarchy of block partitionings as adm_A. We define the row-wise bandwidth of level k, $0 \le k \le L_J$, induced by adm_b and A by*

$$C^k_{bw,r}(adm_b, A) := \max\{dist(\tau,\tau') + 2 : \tau,\tau' \in P_J^k, \exists\ \sigma\times\tau, \sigma\times\tau' \in P^{hier}_{I\times J} \text{ with } adm_b(\sigma\times\tau) = False \text{ and } adm_b(\sigma\times\tau') = False\}$$

and the column-wise bandwidth of level ℓ, $0 \leq \ell \leq L_I$, *induced by* adm_b *and* A *by*

$$C^{\ell}_{bw,c}(adm_b, A) := \max\{\, dist(\sigma, \sigma') + 2 : \sigma, \sigma' \in P^{\ell}_I, \exists\ \sigma \times \tau, \sigma' \times \tau \in P^{hier}_{I\times J} \text{ with } adm_b(\sigma \times \tau) = False \text{ and } adm_b(\sigma' \times \tau) = False\}.$$

We then define the row-wise bandwidth $C_{bw,r}$, *column-wise bandwidth* $C_{bw,c}$ *and bandwidth* C_{bw} *by*

$$\begin{aligned} C_{bw,r}(adm_b, A) &:= \max_{0\leq k\leq L_J} \left(C^k_{bw,r}(adm_b, A) \right), \\ C_{bw,c}(adm_b, A) &:= \max_{0\leq \ell\leq L_I} \left(C^{\ell}_{bw,c}(adm_b, A) \right), \\ C_{bw}(adm_b, A) &:= \max\left(C_{bw,r}(adm_b, A), C_{bw,c}(adm_b, A) \right). \end{aligned}$$

Usually the admissibility functions adm_A and adm_b will be identical, but later on we will need this differentiation. The sparsity constant C_{sp} and the bandwidth C_{bw} are inherently linked. We will later see how we can use C_{bw} to bound C_{sp}.

In the cost analysis, we will retrace the change of the $\mathcal{H}$-matrix structure by directly working with the admissibility functions of all intermediate results. To do that, we rewrite the conditions in Definition 2.24 and Definition 2.25 by using boolean algebra and use $+$ and $\cdot$ instead of $\vee$ and $\wedge$. This allows us to use the sum and product symbols $\sum$ and $\prod$, which makes the notation easier to follow. As before, we set

$$\text{True} > \text{False}$$

and otherwise follow standard boolean algebra, meaning we have

$$\begin{aligned} \text{True} + \text{True} &= \text{True} & \qquad \text{True} + \text{False} &= \text{True} \\ \text{False} + \text{True} &= \text{True} & \qquad \text{False} + \text{False} &= \text{False} \end{aligned}$$

for the sum and

$$\begin{aligned} \text{True} \cdot \text{True} &= \text{True} & \qquad \text{True} \cdot \text{False} &= \text{False} \\ \text{False} \cdot \text{True} &= \text{False} & \qquad \text{False} \cdot \text{False} &= \text{False} \end{aligned}$$

for the product. We start with the induced sum admissibility function.

Theorem 2.30 (Matrix sum admissibility II) *Let us have the situation from Definition 2.24. If* adm_C *is an induced matrix sum admissibility function w.r.t.* adm_A *and* adm_B *then*

$$adm_C(\sigma \times \tau) = adm_A(\sigma \times \tau) \cdot adm_B(\sigma \times \tau)$$

holds true for all blocks $b = \sigma \times \tau \in P^{hier}_{I\times J}$. *If* adm_C *is a partially induced matrix sum*

admissibility function w.r.t. adm_A and adm_B then

$$adm_C(\sigma \times \tau) \leq adm_A(\sigma \times \tau) \cdot adm_B(\sigma \times \tau)$$

holds true for all blocks $\sigma \times \tau \in P^{hier}_{I \times J}$.

We continue with the induced product admissibility function.

Theorem 2.31 (Matrix product admissibility II) *Let us have the situation from Definition 2.25. If adm_C is an induced matrix product admissibility function w.r.t. adm_A and adm_B then*

$$adm_C(\sigma \times \gamma) = \prod_{\tau \in P_J^{L^{max}(\sigma,\gamma,J)}} (adm_A(\sigma \times \tau) + adm_B(\tau \times \gamma))$$

holds true for all blocks $b = \sigma \times \gamma \in P^{hier}_{I \times K}$. If adm_C is a partially induced matrix product admissibility function w.r.t. adm_A and adm_B then

$$adm_C(\sigma \times \gamma) \leq \prod_{\tau \in P_J^{L^{max}(\sigma,\gamma,J)}} (adm_A(\sigma \times \tau) + adm_B(\tau \times \gamma))$$

holds true for all blocks $b = \sigma \times \gamma \in P^{hier}_{I \times K}$.

Our next definition in this chapter deals with another constant that is usually central to the cost estimate of the $\mathcal{H}$-matrix product. It describes the difference between the structure that is actually used for the resulting $\mathcal{H}$-matrix and the structure predicted by the induced admissibility function from Theorem 2.31. We use the (partially) induced admissibility functions in our upcoming cost analysis, so we have no direct need for this. However, we will refer to this parameter later, so we introduce it formally.

Definition 2.32 (Idempotency constant) *Let us have the same situation as given in Definition 2.25 and let adm_C be an induced matrix product admissibility function w.r.t. to adm_A and adm_B, which yields the consistent H-block partition $P^{\mathcal{H}}_{I \times K}$. Let $\bar{P}^{\mathcal{H}}_{I \times K}$ be a second $\mathcal{H}$-block partition based on the same hierarchy of block partitionings as $P^{\mathcal{H}}_{I \times K}$ and let it be consistent with a second admissibility function adm'_C. Then we define for all blocks $\sigma \times \gamma \in \bar{P}^{\mathcal{H}}_{I \times K}$ the idempotency constant by*

$$C_{id}(\sigma \times \gamma, P^{\mathcal{H}}_{I \times K}) := \#\left\{\sigma' \times \gamma' \in P^{\mathcal{H}}_{I \times K} : \sigma' \times \gamma' \subseteq \sigma \times \gamma\right\}$$

and for $\bar{P}^{\mathcal{H}}_{I \times K}$ itself by

$$C_{id}\left(\bar{P}^{\mathcal{H}}_{I \times K}, P^{\mathcal{H}}_{I \times K}\right) = \max\left\{1, \max\left\{C_{id}(\sigma \times \gamma, P^{\mathcal{H}}_{I \times K}) : \sigma \times \gamma \in \bar{P}^{\mathcal{H}}_{I \times K}\right\}\right\}.$$

At last, we define the imbalance constant. It describes for an $\mathcal{H}$-matrix A with row index set I and column index set J the maximum amount of levels of the respective hierarchy of partitionings a row index cluster σ or column index cluster τ can have partners for in their collective hierarchy of block partitionings. For a square $\mathcal{H}$-matrix with $I = J$ this is usually 1, but for rectangular matrices it is often higher. Recall (the non-standard for square $\mathcal{H}$-matrices) Example 2.2 from this subsection, where the imbalance constant is 4 because the row column cluster $I = \{0, 1, 2, 3\}$ has partners in the hierarchy of partitionings of J on levels 0, 1, 2, and 3. This will be an important part of the upcoming Theorem 2.45 and by that of Theorem 2.47, which gives an upper bound to the cost of the $\mathcal{H}$-matrix product. This constant is newly introduced by us and we will further explain the necessity of its introduction after Theorem 2.46.

Definition 2.33 (Imbalance Constant) *Let $\left(P^{\ell}_{I\times J}\right)^{L}_{\ell=0}$ be a hierarchy of block partitionings based on hierarchies of partitionings $(P^{\ell}_{I})^{L_I}_{\ell=0}$, $(P^{\ell}_{J})^{L_J}_{\ell=0}$. Then we define the imbalance constant by*

$$C_{imb}(I,J) := \max\left\{ \max_{\sigma\in P_I^{hier}} \{L^{col}(\sigma, J)\}, \max_{\tau\in P_J^{hier}} \{L^{row}(\tau, I)\} \right\}.$$

2.2.2 Construction of a Hierarchical Matrix

Given a model problem, the creation of corresponding $\mathcal{H}$-matrices involves three main steps. At first, suitable hierarchies of partitionings for the row and column index sets have to be found, which then are used to construct the hierarchy of block partitionings. Second, using a suitable admissibility function, an $\mathcal{H}$-block partition is defined. Finally, a low-rank factorization technique is used to find suitable factorizations for all admissible blocks.

The first and second step are inherently linked. The partitions of the row and column sets on every level have to be chosen in a way that groups similar information together because otherwise we cannot expect the admissibility function to identify any admissible blocks. So the admissibility function has to be known first before the hierarchical structure can be developed based on that information.

When choosing the admissibility function, we need to pursue two opposing goals. On the one hand, we want small storage cost. Hence, we try to create an $\mathcal{H}$-matrix made up of large admissible blocks and only a few small inadmissible blocks. On the other hand, we want to approximate our original matrix as closely as possible. That is most easily achieved by many inadmissible blocks and - where possible - relatively small admissible blocks. Therefore, the primary task when deciding which admissibility function to choose is to balance these two goals.

Similar considerations must be made in the last step when finding the low-rank approximations of the admissible blocks. A higher rank leads to a better approximation of the original matrix but with higher storage and computational cost.

For many matrices arising in applications, the approach to solving the problem of finding an admissibility function is based on geometric information associated with its row and column indices. Given a matrix $A \in \mathbb{R}^{I \times J}$, we assume that every index $i \in I$ is associated with a subset X_i in $\mathbb{R}^d$, where X_i could be anything from a set containing only a point in $\mathbb{R}^d$ to a set with positive volume like an interval for $d = 1$. In the same way, any index $j \in J$ is associated with a subset $Y_j \subseteq \mathbb{R}^d$. We can easily extend this association to the subsets $\sigma \subset I$ and $\tau \subset J$ by setting

$$X_\sigma := \bigcup_{i \in \sigma} X_i, \quad Y_\tau := \bigcup_{j \in \tau} Y_j.$$

Using this association, we can now define the diameter of a cluster σ by

$$\operatorname{diam}(\sigma) := \max\{\|x' - x''\| : x', x'' \in X_\sigma\}$$

and the distance between two clusters by

$$\operatorname{dist}_g(\sigma, \sigma') = \min\{\|x - x'\| : x \in X_\sigma, x' \in X_{\sigma'}\}.$$

Note that this distance is different from the one we defined in Definition 2.26. We assume - and the appropriateness of this assumption relies on the data in the matrix and the underlying application - that diameter and distance of clusters give us information about the (numerical) rank (of an accurate approximation) of a given submatrix. If all associated clusters of the row index set are far enough from all associated clusters from the column index set, we assume the block to be admissible. We formalize this in the following definition.

Definition 2.34 (Standard admissibility function) *Let $A \in \mathcal{H}(P^{\mathcal{H}}_{I \times J}, k, adm_A)$ be an $\mathcal{H}$-matrix and $\eta \in \mathbb{R}$. Then adm_A is called the standard admissibility function with parameter $\eta > 0$ if it satisfies*

$$adm_A(\sigma \times \tau) = \begin{cases} True & \min\{diam(\sigma), diam(\tau)\} \leq \eta \cdot dist_g(\sigma, \tau) \\ False & else \end{cases}$$

for all $\sigma \times \tau \in P^{hier}_{I \times J}$.

Note that the standard admissibility function is an admissibility function as defined by us in Definition 2.21 because for all $b = \sigma \times \tau \in P^{hier}_{I \times J}$ with $\operatorname{adm}_A(b) = \text{True}$, all descendants $b' \subset b$ with $b' = \sigma' \times \tau' \in P^{hier}_{I \times J}$ satisfy by construction

$$\min\{\operatorname{diam}(\sigma'), \operatorname{diam}(\tau')\} \leq \min\{\operatorname{diam}(\sigma), \operatorname{diam}(\tau)\} \leq \eta \cdot \operatorname{dist}_g(\sigma, \tau) \leq \eta \cdot \operatorname{dist}_g(\sigma', \tau'),$$

and thus also

$$\mathrm{adm}_A(b') = \text{True}.$$

Whether this is a practical approach to distinguish between admissible and inadmissible blocks again depends on the data in the matrix, which stems from the underlying application. We will present an argument for our model problem in Subsection 2.5. A detailed justification for this approach can also be found in [43].

Unfortunately, the direct calculation of $\mathrm{diam}(\sigma)$, $\mathrm{diam}(\tau)$ and $\mathrm{dist}_g(\sigma, \tau)$ for index sets associated with arbitrary subsets in $\mathbb{R}^d$ may be costly. We simplify this by using more easily calculable bounding boxes that contain the associated subsets. To be more precise, for a given cluster σ we find a smallest possible cuboid

$$C_\sigma = \bigtimes_{j=1}^{d} [a_i, b_i]$$

that satisfies $X_\sigma \subset C_\sigma$ and then use C_σ instead of X_σ. It can be easily seen that

$$\mathrm{diam}(\sigma) \leq \mathrm{diam}(C_\sigma)$$

and

$$\mathrm{dist}_g(\sigma, \tau) \geq \mathrm{dist}_g(C_\sigma, C_\tau)$$

hold true. Therefore, this simplification could lead to the identification of admissible blocks as inadmissible, but not the other way around. In general, this is not a problem in the application. More information on this and other considerations can be found in [43].

The last step of the construction, finding low-rank factorizations for all admissible blocks, also benefits from a cost-conscious decision. Using the SVD would lead to the best accuracy, as we have seen in Theorem 2.11, but this could prove to be too costly. It undermines our goal of quasi-linear complexity if we need to use the dense version of a given submatrix before calculating its low-rank factorization.

An algorithm based on incomplete knowledge of the block in which we are interested would alleviate this problem. We want to calculate the low-rank factorization of the entire block while relying on having access only to specific entries of the matrix. This only works if the model problem allows for an independent calculation of specific entries of the matrix. This will be the case in our model problems.

One notable example of an algorithm that satisfies these conditions is the adaptive cross approximation [9], a variant of the cross approximation [6], first developed as skeleton approximation in [34].

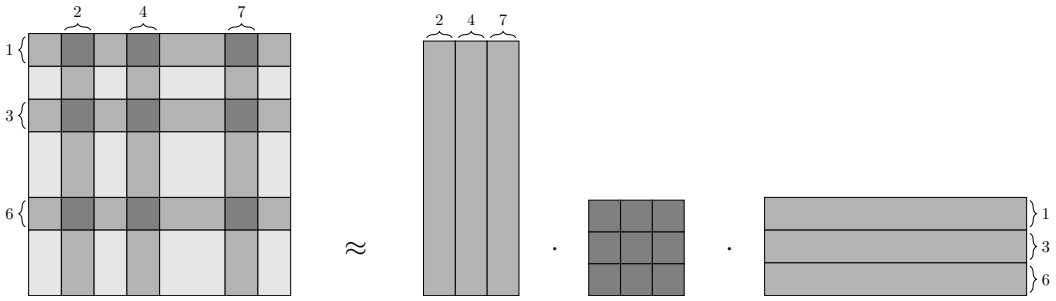


Figure 2.7. Visualization of the adaptive cross approximation [2]. Note that the middle block S on the right side of the equation is the inverse of all the dark grey blocks in the matrix A on the left.

The central idea is as follows. For a given dense matrix $A \in \mathbb{R}^{I \times J}$, we choose small sets $\sigma \subseteq I$ and $\tau \subseteq J$ so that for

$$\widetilde{A} = A_{I,\tau} \cdot S \cdot A_{\sigma,J}$$

with $S \in \mathbb{R}^{\sigma \times \tau}$ we have

$$\|A - \widetilde{A}\|_2 \leq \varepsilon$$

with $\varepsilon > 0$ small.

It can be shown that under the assumption of A having a good low-rank approximation, there is also an almost equally good cross approximation [34]. But existence alone does not solve our problem. We need an efficient way to compute such an approximation.

We want to construct the cross approximation as a sequence of rank 1 updates, meaning

$$\widetilde{A} = \sum_{\ell=1}^{k} u_\ell v_\ell^T .$$

The vectors $v_{\ell'}$ and $u_{\ell'}$ of the ℓ'-th step are calculated by the following scheme. First, we identify the index pair (i^*, j^*) for which the absolute value of the entry $A_{i,j}^{(\ell')}$ of the matrix

$$A^{(\ell')} = A - \sum_{\ell=1}^{\ell'-1} u_\ell v_\ell^T$$

is largest. We set $\delta = A^{(\ell')}_{i^*,j^*}$ and then

$$u_{\ell'} = \frac{1}{\delta} A^{(\ell')}_{I,j^*} \quad v_{\ell'} = A^{(\ell')}_{i^*,J}.$$

It can then be shown that these rank 1 updates can be rewritten as

$$\sum_{\ell=1}^{k} u_\ell v_\ell^T = A_{I,\tau} \cdot A^{-1}_{\sigma,\tau} \cdot A_{\sigma,J}.$$

Unfortunately, we still need access to the entire matrix because otherwise we cannot find the entry with the largest absolute value. A possible solution would be to change the identification of the pivot index pair (i^*, j^*) to a heuristic and not fully compute the matrices $A^{(\ell)}$.

In the ℓ-th step, we choose an unused row and calculate all previous updates for its entries. Then we find the entry with the largest absolute value in this row and calculate all previous updates for the corresponding column as well. The largest entry of this column is then the new pivot element. This method is often called adaptive cross approximation (with partial pivoting) and is formalized in Algorithm 2.1.

Finally, we need a useful stopping criterion. We again have to apply a heuristic because we do not have access to the entire matrix. A possible solution would be to use the norm of the rows and columns that were already calculated because we assume anyway that they are an indicator for how close our matrix is to a zero matrix after all previous updates. So we could use the relative criterion

$$\frac{\|u_\ell\|_2 \|v_\ell\|_2}{\|u_1\|_2 \|v_1\|_2} \leq \varepsilon.$$

Unfortunately, simple examples can be constructed in which this stopping criterion does not work, meaning we do not capture most of the information contained in the matrix, but we stop the algorithm anyway [15]. There are other options, such as using additional random, not yet computed entries of the matrices $A^{(\ell)}$ to check how good the low-rank approximation already is [29]. Still, that would not solve the fundamental problem because new counterexamples can be constructed, so we do not pursue this approach further.

Algorithm 2.1: Adaptive cross approximation with partial pivoting.

Data: Dense matrix $A \in \mathbb{R}^{I \times J}$ with $\#I = m$ and $\#J = n$, tolerance tol, stopping criterion *stopcrit*
Result: Approximation $A_k = U \cdot V^T$ of A with rank k
Function: $[U, V] = ACA(A, tol)$

Initialize $k = 1$;
Choose random index $1 \leq i \leq m$;
while *stopcrit not true* **do**
  Let v be the i-th row of A;
  Compute $v = v - \sum_{\ell=1}^{k-1} U_{i,\ell} \cdot V_{J,\ell}^T$;
  Let $j \in J$ be the index of the largest element $\delta := |v_j|$ of v;
  Let u be the j-th column of A;
  Compute $u = u - \sum_{\ell=1}^{k-1} U_{I,\ell} \cdot V_{j,\ell}$;
  Set $U = [U\ u]$ and $V = [V\ v^T/\delta]$;
  Let $i \in I$ be the index of the largest entry $|u_i|$ of u;
  Set $k = k + 1$;
end
return U, V;

2.2.3 Hierarchical Matrix Arithmetics

The following part is based on [36], [8] and [43]. We adapt the results to our notation, use the induced admissibility function instead of the idempotency constant C_{id} and generalize some results, mainly to include the rectangular case. Due to these changes, nearly all proofs are given here, but in most cases, only the notation and some details are different from the source. Under storage cost we understand the number of (real) matrix entries that have to be stored, and we use the number of floating point operations (flops) to measure the computational cost.

The cost generally does not depend on the actual $\mathcal{H}$-matrix - where the exact position of every admissible and inadmissible block is fixed - but on the $\mathcal{H}$-block partition - where at least on some levels some flexibility on the order of blocks is possible - combined with a corresponding admissibility function and rank. To reduce notational overhead, we often use the $\mathcal{H}$-matrix as an argument even if it originally was defined using the $\mathcal{H}$-block partition.

Theorem 2.35 (Dense matrix storage and matrix-vector product) *Let $A \in \mathbb{R}^{m \times n}$ be a matrix in dense matrix representation and $x \in \mathbb{R}^n$ a vector. Then the storage cost for A is*

$$\mathcal{S}_F(m, n) = m \cdot n,$$

and the computational cost of the matrix-vector product Ax is

$$\mathcal{N}_{F,mv}(m,n) = 2mn - m.$$

Theorem 2.36 (Low-rank matrix storage and matrix-vector product) *Assume a matrix $A \in \mathbb{R}^{m\times n}$ is given in low-rank representation $A = U\cdot V^T$ with $k \in \mathbb{N}$, $U \in \mathbb{R}^{m\times k}$, $V \in \mathbb{R}^{n\times k}$, and let $x \in \mathbb{R}^n$ be a vector. Then the storage cost for A is*

$$\mathcal{S}_R(m,n,k) = k(m+n),$$

and the computational cost of the matrix-vector product Ax is

$$\mathcal{N}_{R,mv}(m,n,k) = 2k(m+n) - m - k.$$

We will need to be able to reduce the rank of a low-rank matrix while losing as little information as possible. For a given matrix $A \in \mathbb{R}^{m\times n}$ with low-rank representation $A = U \cdot V^T$, $U \in \mathbb{R}^{m\times k}$, $V \in \mathbb{R}^{n\times k}$, we thus search for a matrix $A' \in \mathbb{R}^{m\times n}$ with low-rank representation $A = U \cdot V^T$, $U \in \mathbb{R}^{m\times k'}$, $V \in \mathbb{R}^{n\times k'}$ and given $k' < k$ so that

$$\|A - A'\|_2$$

is as small as possible.

Recalling Theorem 2.11, this can be done (optimally) by using the SVD, and a possible application of this idea is given in Algorithm 2.2. We define this *truncation operator for matrices with low-rank representation* as

$$\mathcal{T}^R_{k'\leftarrow k}(A) = A'.$$

The best approximation computed in Algorithm 2.2 is possibly not unique if A has repeated singular values. In that case, we arbitrarily choose one representative.

Theorem 2.37 (Truncation using SVD) *Let $R \in \mathbb{R}^{m\times n}$ be a low-rank matrix with factorization $R = UV^T$, $U \in \mathbb{R}^{m\times k}$, $V \in \mathbb{R}^{n\times k}$. Then the computational cost to find a best approximation of R with rank at most $k' \leq k$ using Algorithm 2.2 has a computational cost bounded by*

$$\mathcal{N}_{R,k'\leftarrow k}(m,n) \leq 6k^2(n+m) + 23k^3.$$

Proof We need to go through the computational cost of the steps in Algorithm 2.2. The QR decompositions of A and B cost at most $4nk^2$ and $4mk^2$, respectively. The multiplications cost $2k^3$ (for $R_U \cdot R_V^T$), $2nk^2$ (for $Q_U \cdot \widetilde{U} \cdot \Sigma$) and $2mk^2$ (for $Q_V \cdot \widetilde{V}$) and the SVD at most $21k^3$. Combined, the entire algorithm has a computational cost of at most

$$\mathcal{N}_{R,k'\leftarrow k}(m,n) \leq 6k^2(n+m) + 23k^3.$$ ∎

Algorithm 2.2: Truncation using the singular value decomposition.

Data: Low-rank matrix $R \in \mathbb{R}^{m\times n}$ given as $R = U \cdot V^T$ with $U \in \mathbb{R}^{m\times k}$, $V \in \mathbb{R}^{n\times k}$ and the desired rank $k' \leq k$
Result: Best approximation $R' \in \mathbb{R}^{m\times n}$ given as $R' = U' \cdot V'^T$ with $U' \in \mathbb{R}^{m\times k'}$ and $V' \in \mathbb{R}^{n\times k'}$
Function: $[U', V'] = \mathcal{T}^R_{k'\leftarrow k}(R)$

1 Compute the QR decomposition $U = Q_U \cdot R_U$ of U with $Q_U \in \mathbb{R}^{m\times k}$, $R_U \in \mathbb{R}^{k\times k}$;
2 Compute the QR decomposition $V = Q_V \cdot R_V$ of V with $Q_V \in \mathbb{R}^{n\times k}$, $R_V \in \mathbb{R}^{k\times k}$;
3 Compute the singular value decomposition $R_U R_V^T = \widetilde{U}\Sigma\widetilde{V}^T$ of $R_U R_V^T$ with $\widetilde{U}, \Sigma, \widetilde{V} \in \mathbb{R}^{k\times k}$;
4 Define U' as the first k' columns of $Q_U\widetilde{U}\Sigma$;
5 Define V' as the first k' columns of $Q_V\widetilde{V}$;
6 **return** U', V';

Theorem 2.38 (Formatted addition) *Let $R \in \mathbb{R}^{m\times n}$ and $S \in \mathbb{R}^{m\times n}$ be low-rank matrices with rank at most $k \in \mathbb{N}$ and factorizations $R = U_R \cdot V_R^T$ and $S = U_S \cdot V_S^T$ with $U_R, U_S \in \mathbb{R}^{m\times k}$ and $V_R, V_S \in \mathbb{R}^{n\times k}$. We define the formatted addition of two low-rank matrices of rank k to a target matrix $R \oplus S$ of rank k as*

$$R \oplus S := \mathcal{T}^R_{k\leftarrow 2k}(R + S).$$

The formatted addition has a computational cost of

$$\mathcal{N}_{R,\oplus}(n, m) \leq 24k^2(n + m) + 184k^3.$$

Proof Apply Theorem 2.37 to the low-rank matrix $[U_R\ U_S] \cdot [V_R\ V_S]^T$. ∎

Before we show similar statements for $\mathcal{H}$-matrices, we need to show the following theorem.

Theorem 2.39 *Let $P^{\mathcal{H}}_{I\times J}$ be an $\mathcal{H}$-block partition. Then it holds that*

$$\sum_{\sigma\times\tau\in P^{\mathcal{H}}_{I\times J}} \#\sigma \leq \big(depth_{row}(P^{\mathcal{H}}_{I\times J}) + 1\big) \cdot \#I \cdot C_{sp}(P^{\mathcal{H}}_{I\times J}),$$
$$\sum_{\sigma\times\tau\in P^{\mathcal{H}}_{I\times J}} \#\tau \leq \big(depth_{col}(P^{\mathcal{H}}_{I\times J}) + 1\big) \cdot \#J \cdot C_{sp}(P^{\mathcal{H}}_{I\times J}),$$

where $depth_{row}(\cdot)$, $depth_{row}(\cdot)$ have been defined in Definition 2.20 and $C_{sp}(\cdot)$ in Definition 2.28.

Proof We have

$$\begin{aligned}
\sum_{\sigma\times\tau\in P^{\mathcal{H}}_{I\times J}} \#\sigma &= \sum_{\sigma\in P^{hier}_I} \#\sigma \left(\sum_{\tau\in P^{hier}_J \text{ with } \sigma\times\tau\in P^{\mathcal{H}}_{I\times J}} 1 \right) \\
&\leq \sum_{\sigma\in P^{hier}_I} \#\sigma \cdot C_{sp}(P^{\mathcal{H}}_{I\times J}) \\
&= \left(\sum_{0\leq\ell\leq \operatorname{depth}_{\mathrm{row}}(P^{\ell}_{I\times J})} \sum_{\sigma\in P^{\ell}_I} \#\sigma \right) \cdot C_{sp}(P^{\mathcal{H}}_{I\times J}) \\
&\leq \left(\sum_{0\leq\ell\leq \operatorname{depth}_{\mathrm{row}}(P^{\ell}_{I\times J})} \#I \right) \cdot C_{sp}(P^{\mathcal{H}}_{I\times J}) \\
&= \Big(\operatorname{depth}_{\mathrm{row}}(P^{\ell}_{I\times J}) + 1 \Big) \cdot \#I \cdot C_{sp}(P^{\mathcal{H}}_{I\times J}).
\end{aligned}$$

The proof for the second part is analogous. ∎

Now we can bound the storage cost of an $\mathcal{H}$-matrix.

Theorem 2.40 ($\mathcal{H}$-matrix storage cost) *Let $P^{\mathcal{H}}_{I\times J}$ be an $\mathcal{H}$-block partition and adm a consistent (see Definition 2.22) admissibility function. Then the storage cost of a matrix $A \in \mathcal{H}(P^{\mathcal{H}}_{I\times J}, k, adm)$ denoted by $S_{\mathcal{H}}(P^{\mathcal{H}}_{I\times J}, k)$ is bounded by*

$$S_{\mathcal{H}}(P^{\mathcal{H}}_{I\times J}, k) \leq C_{sp}(P^{\mathcal{H}}_{I\times J}) \cdot \max\{n_{min}, k\} \cdot ((depth_{row}(A) + 1)\,\#I + (depth_{col}(A) + 1)\,\#J)$$

with n_{min} defined in Definition 2.22 as well.

Proof The storage cost is given by

$$S_{\mathcal{H}}(P^{\mathcal{H}}_{I\times J}, k) = k \cdot \sum_{\sigma\times\tau\in P^{+}_{I\times J}} (\#\sigma + \#\tau) + \sum_{\sigma\times\tau\in P^{-}_{I\times J}} (\#\sigma \cdot \#\tau)$$

where $P^{+}_{I\times J}$ and $P^{-}_{I\times J}$ have been defined in Definition 2.22. By construction, for every element of the near-field $P^{-}_{I\times J}(\cdot)$ $\min\{\#\sigma, \#\tau\}$ is at most of size n_{min}. We thus have for the elements in the second sum

$$\begin{aligned}
\#\sigma\#\tau &= \min\{\#\sigma, \#\tau\} \max\{\#\sigma, \#\tau\} \\
&\leq \min\{\#\sigma, \#\tau\}(\#\sigma + \#\tau) \leq n_{min}(\#\sigma + \#\tau),
\end{aligned}$$

and thus

$$S_{\mathcal{H}}(P^{\mathcal{H}}_{I\times J}, k) \leq \max\{n_{min}, k\} \sum_{\sigma\times\tau\in P^{\mathcal{H}}_{I\times J}} (\#\sigma + \#\tau).$$

Then we have the statement by applying Theorem 2.39. ∎

To simplify the notation, we will sometimes relate the storage cost directly to the $\mathcal{H}$-matrix A instead of its $\mathcal{H}$-block partition and rank k. So we will set

$$S_{\mathcal{H}}(A) := S_{\mathcal{H}}(P^{\mathcal{H}}_{I\times J}, k)$$

if the context is clear. Often, one can expect that the row and column depths of A satisfy

$$\begin{aligned}\operatorname{depth}_{row}(A) &= \mathcal{O}(\log(\#I)),\\ \operatorname{depth}_{col}(A) &= \mathcal{O}(\log(\#J)).\end{aligned}$$

Inserting this in our result from Theorem 2.40, we can see that the storage cost is of size

$$\mathcal{O}((\#I + \#J)\cdot\log(\#I + \#J)),$$

meaning it grows quasi-linear with the size of the $\mathcal{H}$-matrix. The storage cost of an $\mathcal{H}$-matrix can serve as a lower and upper bound for the matrix-vector product.

Theorem 2.41 (Hierarchical matrix-vector product) *Let $P^{\mathcal{H}}_{I\times J}$ be an $\mathcal{H}$-block partition, adm_A a consistent admissibility function and $A \in \mathcal{H}(P^{\mathcal{H}}_{I\times J}, k, adm_A)$ an $\mathcal{H}$-matrix. Let $x \in \mathbb{R}^J$. Then the computational cost of the matrix-vector product Ax can be bounded by the storage cost*

$$S_{\mathcal{H}}(P^{\mathcal{H}}_{I\times J}, k) \leq \mathcal{N}_{mv}(P^{\mathcal{H}}_{I\times J}, k) \leq 2S_{\mathcal{H}}(P^{\mathcal{H}}_{I\times J}, k).$$

Proof To compute the product Ax, we need to multiply every block $A_{\sigma,\tau}$ for $\sigma\times\tau \in P^{\mathcal{H}}_{I\times J}$ with the correct part x_τ of x and add the result to the part y_σ of the target vector y. If $A_{\sigma,\tau}$ is a full matrix block, then with Theorem 2.35 the multiplication costs

$$\mathcal{N}_{F,mv}(\sigma,\tau) = 2\cdot\#\sigma\cdot\#\tau - \#\sigma,$$

and the addition costs $\#\sigma$. If $A_{\sigma,\tau}$ is a low-rank block, we can apply Theorem 2.36, and the multiplication costs

$$\mathcal{N}_{R,mv}(\#\sigma,\#\tau,k) = 2k\cdot(\#\sigma + \#\tau) - \#\sigma - k.$$

The addition to the target vector costs again $\#\sigma$. Because the storage requirements are $\mathcal{S}_F(\#\sigma,\#\tau) = \#\tau\#\sigma$ for an inadmissible block and $\mathcal{S}_R(\#\sigma\#\tau) = k(\#\sigma + \tau\#)$

for an admissible block, the cost is in both cases bounded below by the respective storage requirements and bounded above by twice the respective storage requirements. Combining these blockwise results for the entire $\mathcal{H}$-matrix, we have

$$S_{\mathcal{H}}(P^{\mathcal{H}}_{I\times J}, k) \leq \mathcal{N}_{mv}(P^{\mathcal{H}}_{I\times J}, k) \leq 2S_{\mathcal{H}}(P^{\mathcal{H}}_{I\times J}, k).$$ ∎

Using Theorem 2.37 blockwise on all admissible blocks, we can also truncate an $\mathcal{H}$-matrix to a given rank. For that, we define the *truncation operator for $\mathcal{H}$-matrices* by

$$T^{\mathcal{H}}_{k'\leftarrow k}(A) = A',$$

where $A \in \mathcal{H}(P^{\mathcal{H}}_{I\times J}, k, \mathrm{adm})$ is the original $\mathcal{H}$-matrix with rank at most k and $A' \in \mathcal{H}(P^{\mathcal{H}}_{I\times J}, k', \mathrm{adm})$ the resulting $\mathcal{H}$-matrix with rank at most $k' < k$.

Theorem 2.42 (Truncation of $\mathcal{H}$-matrices) *Let $P^{\mathcal{H}}_{I\times J}$ be an $\mathcal{H}$-block partition, adm a consistent admissibility function, and $A \in \mathcal{H}(P^{\mathcal{H}}_{I\times J}, k, adm)$ an $\mathcal{H}$-matrix. The computational cost $\mathcal{N}_{\mathcal{H},k'\leftarrow k}(P^{\mathcal{H}}_{I\times J}, k)$ of finding an approximation of A in $\mathcal{H}(P^{\mathcal{H}}_{I\times J}, k', adm)$ with rank k' using the SVD as in Theorem 2.37 on all admissible blocks is given by*

$$\mathcal{N}_{\mathcal{H},k'\leftarrow k}(P^{\mathcal{H}}_{I\times J}, k) \leq 6kS_{\mathcal{H}}(P^{\mathcal{H}}_{I\times J}, k) + 23k^3 \# P^{\mathcal{H},+}_{I\times J}.$$

Proof We use Theorem 2.37:

$$\begin{aligned}
\mathcal{N}_{\mathcal{H},k'\leftarrow k}(P^{\mathcal{H}}_{I\times J}, k) &= \sum_{\sigma\times\tau\in P^{\mathcal{H},+}_{I\times J}} \mathcal{N}_{R,k'\leftarrow k}(\#\sigma, \#\tau) \\
&\leq \sum_{\sigma\times\tau\in P^{\mathcal{H},+}_{I\times J}} 6k^2(\#\tau + \#\sigma) + 23k^3 \\
&= 6k \left(\sum_{\sigma\times\tau\in P^{\mathcal{H},+}_{I\times J}} k(\#\sigma + \#\tau) \right) + 23k^3 \# P^{\mathcal{H},+}_{I\times J} \\
&\leq 6kS_{\mathcal{H}}(P^{\mathcal{H}}_{I\times J}) + 23k^3 \# P^{\mathcal{H},+}_{I\times J}.
\end{aligned}$$ ∎

The truncation can be used to reduce the ranks in admissible blocks of the sum of two $\mathcal{H}$-matrices in $\mathcal{H}(P^{\mathcal{H}}_{I\times J}, k, \mathrm{adm})$ to rank k again. We will call this operation the formatted $\mathcal{H}$-matrix addition and will define it in the upcoming theorem.

Theorem 2.43 (Formatted $\mathcal{H}$-matrix addition) *Let $P^{\mathcal{H}}_{I\times J}$ be an $\mathcal{H}$-block partition with a consistent admissibility function adm and $A, B \in \mathcal{H}(P^{\mathcal{H}}_{I\times J}, k, adm)$ two $\mathcal{H}$-matrices. We define the formatted addition of two $\mathcal{H}$-matrices to a target matrix*

$$A \oplus B \in \mathcal{H}(P^{\mathcal{H}}_{I\times J}, k, adm)$$

with admissible blocks of the same rank as

$$A \oplus B := T^{\mathcal{H}}_{k \leftarrow 2k}(A + B).$$

The formatted addition of two $\mathcal{H}$-matrices has a computational cost of

$$\mathcal{N}_{\mathcal{H},\oplus}(P^{\mathcal{H}}_{I\times J}, k) \leq 24kS_{\mathcal{H}}(P^{\mathcal{H}}_{I\times J}, k) + 184k^3 \# P^{\mathcal{H}}_{I\times J}.$$

Proof Replace k by $2k$ in Theorem 2.42 and use $S_{\mathcal{H}}(P^{\mathcal{H}}_{I\times J}, 2k) \leq 2 \cdot S_{\mathcal{H}}(P^{\mathcal{H}}_{I\times J}, k)$. ∎

If we want to reduce the rank of an $\mathcal{H}$-matrix with rank $q \cdot k$ to k, we can truncate pairwise by using the formatted addition $q - 1$ times. For that, we define the *fast truncation operator for $\mathcal{H}$-matrices* by

$$T^{\mathcal{H},\mathrm{fast}}_{k \leftarrow qk}(A, q, k) = A',$$

where $A \in \mathcal{H}(P^{\mathcal{H}}_{I\times J}, qk, \mathrm{adm})$ is the original $\mathcal{H}$-matrix with rank at most qk with $q > 1$ and $A' \in \mathcal{H}(P^{\mathcal{H}}_{I\times J}, k, \mathrm{adm})$ the resulting $\mathcal{H}$-matrix with rank at most k. This is implemented in Algorithm 2.3.

Algorithm 2.3: Fast truncation of $\mathcal{H}$-matrices.

Data: $A \in \mathcal{H}(P^{\mathcal{H},A}_{I\times J}, qk, \mathrm{adm}_A)$ with rank at most qk
Result: Approximation $A' \in \mathcal{H}(P^{\mathcal{H},A}_{I\times J}, k, \mathrm{adm}_A)$ of A with rank at most k
Function: $A' = T^{\mathcal{H},\mathrm{fast}}_{k \leftarrow qk}(A, q, k)$

```
Find matrices A_i ∈ H(P^{H,A}_{I×J}, k, adm_A) so that A = Σ_{i=1}^{q} A_i holds;
Set Ã_1 := A_1;
for 2 ≤ i ≤ q do
    Ã_i = T^{H}_{k←2k}(Ã_{i−1} + A_i);
end
return Ã_q;
```

This has a positive effect on the computation time, but the results can be arbitrarily poor, although that is not likely to occur in practice.

Theorem 2.44 (Fast truncation of $\mathcal{H}$-matrices) *Let $P^{\mathcal{H}}_{I\times J}$ be an $\mathcal{H}$-block partition with corresponding admissibility function adm and $A \in \mathcal{H}(P^{\mathcal{H}}_{I\times J}, qk, adm)$ an $\mathcal{H}$-matrix. The computational cost of finding an approximation of A in $\mathcal{H}(P^{\mathcal{H}}_{I\times J}, k, adm)$ using Algorithm 2.3 is given by*

$$\mathcal{N}^{\mathcal{H},fast}_{k\leftarrow qk}(P^{\mathcal{H}}_{I\times J}, k) \leq (q-1)\left(24kS_{\mathcal{H}}(P^{\mathcal{H}}_{I\times J}, k) + 184k^3 \# P^{\mathcal{H},+}_{I\times J}\right).$$

Proof We apply Theorem 2.38 $(q-1)$-times. ∎

In the product $A \cdot B = C$, a block in C is computed by multiplying the corresponding block row of A and block column of B. Because of the different sizes of the low-rank blocks, this means that, in some cases, we need to extract only a part of a larger low-rank matrix. In the upcoming definition, we will define $U_\ell(\sigma \times \gamma)$, which is the set of the row or column index clusters (of level ℓ in the corresponding hierarchy of partitionings) of those larger low-rank matrices, depending on whether the block in A or B is larger, for a block $C_{\sigma,\gamma}$. See Figure 2.8 for a visualization.

For the upcoming arguments, recall the following definitions. In Definition 2.18 we set $\mathrm{anc}_i(\sigma)$ as the (uniquely defined) ancestor of σ with $\mathrm{level}(\sigma) \geq i$ on the i-th level, and in Definition 2.19 we set $L^{col}(\sigma, J)$ as the set of all levels of $(P_J^\ell)_{\ell=0}^{L_J}$ containing partners of σ in $P_{I\times J}^{hier}$ and $L^{row}(\tau, I)$ as the set of all levels of $(P_I^\ell)_{\ell=0}^{L_I}$ containing partners of τ in $P_{I\times J}^{hier}$. Furthermore, in Definition 2.25 we defined

$$L^{intersec}(\sigma, \gamma, J) := L^{col}(\sigma, (P_J^\ell)_{\ell=0}^{L_J}, P_{I\times J}^{\mathrm{hier}}) \cap L^{row}(\gamma, (P_J^\ell)_{\ell=0}^{L_J}, P_{J\times K}^{\mathrm{hier}})$$

as the set of all intersecting levels for a block $\sigma \times \gamma \in P_{I\times K}^{\mathrm{hier}}$ and

$$L^{max}(\sigma, \gamma, J) := \max\{\ell \mid \ell \in L^{intersec}(\sigma, \gamma, J)\}$$

as the maximum in each of those sets. Also note that the prerequisites of the following definition mirror those in Definition 2.25.

Definition 2.45 *Let $P_{I\times J}^{\mathcal{H}}$, $P_{J\times K}^{\mathcal{H}}$ and $P_{I\times K}^{\mathcal{H}}$ be $\mathcal{H}$-block partitions that are consistent with admissibility functions adm_A, adm_B and adm_C, respectively, and that are based on (block hierarchies of partitionings resulting from) the same hierarchies of partitionings $(P_I^\ell)_{\ell=0}^{L_I}$, $(P_J^\ell)_{\ell=0}^{L_J}$ and $(P_K^\ell)_{\ell=0}^{L_K}$.*

Assume $L^{intersec}(\sigma, \gamma, J) \neq \emptyset$ for all blocks $\sigma \times \gamma \in P_{I\times K}^{\mathcal{H}}$, meaning $L^{max}(\sigma, \gamma, J)$ always exists. Let adm_C be partially induced as given in Definition 2.25. We define for a block cluster $\sigma \times \gamma \in P_{I\times K}^{\mathcal{H}}$ the sets $U_\ell(\sigma \times \gamma) \subseteq P_J^{hier}$ for $\ell = 0$ by

$$\begin{aligned} U_0(\sigma \times \gamma) := \big\{\tau \in P_J^0 : \exists\ & i \in L^{row}(\tau, I) \text{ and } j \in L^{col}(\tau, K)\big\} \text{ with} \\ & anc_i(\sigma) \times \tau \in P_{I\times J}^{hier} \text{ and } \tau \times anc_j(\gamma) \in P_{J\times K}^{hier} \text{ and} \\ & \big(anc_i(\sigma) \times \tau \in P_{I\times J}^{\mathcal{H}} \text{ or } \tau \times anc_j(\gamma) \in P_{J\times K}^{\mathcal{H}}\big)\big\}. \end{aligned}$$

and recursively for $1 \leq \ell \leq L_J$ by

$$\begin{aligned} U_\ell(\sigma \times \gamma) := \big\{\tau \in P_J^\ell : \nexists\ & \hat{\tau} \in \bigcup_{0\leq k<\ell} U_k(\sigma \times \gamma) \text{ with } \tau \cap \hat{\tau} \neq \emptyset \text{ and} \\ & \exists\ i \in L^{row}(\tau, I) \text{ and } j \in L^{col}(\tau, K)\big\} \text{ with} \\ & anc_i(\sigma) \times \tau \in P_{I\times J}^{hier} \text{ and } \tau \times anc_j(\gamma) \in P_{J\times K}^{hier} \text{ and} \\ & \big(anc_i(\sigma) \times \tau \in P_{I\times J}^{\mathcal{H}} \text{ or } \tau \times anc_j(\gamma) \in P_{J\times K}^{\mathcal{H}}\big)\big\}. \end{aligned}$$

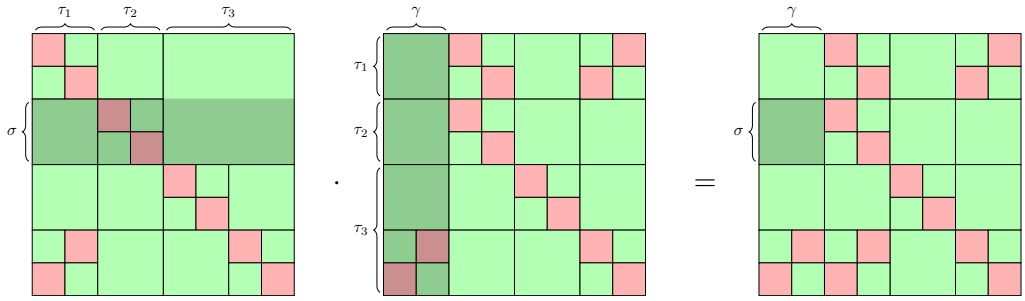


Figure 2.8. Visualization of Definition 2.45. Note that only the $\mathcal{H}$-block partitions are shown, and we assume that the hierarchies of block partitionings were created by a standard bisection process. We have $U_0(\sigma \times \gamma) = \emptyset$, $U_1(\sigma \times \gamma) = \{\tau_3\}$, $U_2(\sigma \times \gamma) = \{\tau_1, \tau_2\}$, and $U_3(\sigma \times \gamma) = \emptyset$. Note that, by the exclusion of elements τ in $U_\ell(\sigma \times \gamma)$ that are part of some $\hat{\tau}$ included in $U_k(\sigma \times \gamma)$ for some $k < \ell$, we ensure that the ancestors of τ_3 on the second level are not part of $U_2(\sigma \times \gamma)$. Hence, $U_\ell(\sigma \times \gamma) \cap U_k(\sigma \times \gamma) = \emptyset$ is always true for $\ell \neq k$.

We need one final statement before we can analyze the cost of the $\mathcal{H}$-matrix product. We use the imbalance constant that, as defined in Definition 2.33, describes for an $\mathcal{H}$-matrix A with row index set I and column index set J the maximum amount of levels of the respective hierarchy of partitionings a row index cluster σ or column index cluster τ can have partners for in their collective hierarchy of block partitionings.

Theorem 2.46 *Let us have the situation of Definition 2.45. Then it holds that*

$$\bigcup_{\ell=0}^{L_J} \bigcup_{\tau \in U_\ell(\sigma \times \gamma)} \tau = J.$$

Furthermore, we have

$$\#U_\ell(\sigma \times \gamma) \leq C_{imb}(I, J) \cdot C_{sp}(P^{\mathcal{H}}_{I \times J}) + C_{imb}(J, K) \cdot C_{sp}(P^{\mathcal{H}}_{J \times K}).$$

Proof We start by proving the first part of the statement. Let $v \in J$ be an index in J and $\sigma \times \gamma \in P^{\mathcal{H}}_{I \times K}$. We set $b_0 = \mathrm{anc}_0(\sigma) \times J = I \times J \in P^{hier}_{I \times J}$ and $b_0' = J \times \mathrm{anc}_0(\gamma) = J \times K \in P^{hier}_{J \times K}$. If b_0 or b_0' is a leaf, then $J \in U_0(\sigma \times \gamma)$, and hence our statement holds. If neither b_0 nor b_0' is a leaf, we need to check two cases.

First, we check whether $L^{row}(J, I)$ and $L^{col}(J, K)$ contain any levels other than 0. If so, denote them by i and j, respectively, and check for all of them, except for those where $\mathrm{anc}_i(\sigma)$ or $\mathrm{anc}_j(\gamma)$ does not exist, whether $\mathrm{anc}_i(\sigma) \times J$ or $J \times \mathrm{anc}_j(\gamma)$ is a leaf. If that is the case, we have $J \in U_0(\sigma \times \gamma)$, and hence our statement is true.

If not, we start over. Let $\widetilde{\tau} \in P^{hier}_J$ be the first non-trivial descendant of J containing v and i, j are of lowest possible level in $L^{col}(\widetilde{\tau}, I)$ and $L^{row}(\widetilde{\tau}, K)$, respectively. Let

$b_1 = \operatorname{anc}_i(\sigma) \times \widetilde{\tau} \in (P^{hier}_{I\times J})$ and $b'_1 = \widetilde{\tau} \times \operatorname{anc}_j(\gamma) \in P^{hier}_{J\times K})$, and let us repeat our steps from before. Then either $\widetilde{\tau} \in U_1(\sigma \times \gamma)$ or we continue, if necessary, with the first non-trivial descendant of $\widetilde{\tau}$ that contains v.

Let $\hat{\tau}$ be the cluster on level $L^{max}(\sigma, \gamma, J)$ that contains the index v. If $\sigma \times \gamma$ is admissible, then at least

$$\operatorname{adm}_A(\sigma \times \hat{\tau}) = \text{True}$$

or

$$\operatorname{adm}_B(\hat{\tau} \times \gamma) = \text{True}$$

hold because adm_C is partially induced. If $\sigma \times \gamma$ is inadmissible, then it is one the highest level of $P^{\mathcal{H}}_{I\times K}$, and $\operatorname{adm}_A(\sigma \times \tau)$ and $\operatorname{adm}_B(\tau \times \gamma)$ are, by definition of the partially induced admissibility function, inadmissible leaves.

Hence, the process described above will terminate in both cases either here or before because at least $\hat{\tau}$ satisfies the conditions (possibly except the disjunctness with elements from $U_{\bar{\ell}}(\sigma \times \gamma)$ with $\ell < L^{max}(\sigma, \gamma, J)$) for being part of $U_{L^{max}(\sigma,\gamma,J)}(\sigma \times \gamma)$.

Let $\tau_\ell \in P^\ell_J$. Then

$$\begin{aligned}
\#U_\ell(\sigma \times \gamma) &\leq \sum_{i \in L^{row}(\tau_\ell, I)} \#\{\tau \in P^{hier}_J : \operatorname{anc}_i(\sigma) \times \tau \in P^{\mathcal{H}}_{J\times K}\} \\
&\quad + \sum_{j \in L^{col}(\tau_\ell, K)} \#\{\tau \in P^{hier}_J : \tau \times \operatorname{anc}_j(\gamma) \in P^{\mathcal{H}}_{I\times J}\} \\
&\leq C_{imb}(I, J) \cdot C_{sp}(P^{\mathcal{H}}_{I\times J}) + C_{imb}(J, K) \cdot C_{sp}(P^{\mathcal{H}}_{J\times K})
\end{aligned}$$

holds, and we have also proven the second part of our statement. ∎

Note that Theorem 2.46 and the previous Definition 2.45 differ from their usual versions as seen, for example, in [36] and [8]. We had to adapt it to rectangular $\mathcal{H}$-matrices and thus are no longer able to assume the same hierarchy of partitionings for I, J and K, which, crucially, means that

$$\#L^{intersec}(\sigma, \gamma, J) = 1$$

does not necessarily hold anymore. This is the main reason for the introduction of the imbalance constant and a different upper bound for the number of elements in $U_\ell(\sigma \times \gamma)$.

We can now use these results in the following theorem, which combines the complexity of the exact $\mathcal{H}$-matrix product with additional truncation.

Theorem 2.47 ($\mathcal{H}$-matrix product) *Let $P^{\mathcal{H}}_{I\times J}$, $P^{\mathcal{H}}_{J\times K}$ and $P^{\mathcal{H}}_{I\times K}$ be $\mathcal{H}$-block partitions that are consistent with admissibility functions adm_A, adm_B and adm_C that are based on hierarchies of block partitionings $\left(P^{\ell}_{I\times J}\right)_{\ell=0}^{L_{I\times J}}$, $\left(P^{\ell}_{J\times K}\right)_{\ell=0}^{L_{J\times K}}$ and $\left(P^{\ell}_{I\times K}\right)_{\ell=0}^{L_{I\times K}}$ resulting from the same hierarchies of partitionings $(P^{\ell}_{I})_{\ell=0}^{L_I}$, $(P^{\ell}_{J})_{\ell=0}^{L_J}$ and $(P^{\ell}_{K})_{\ell=0}^{L_K}$ that are compatible for multiplication. Furthermore, let adm_C be the induced matrix product admissibility function w.r.t. adm_A and adm_B, and let us have $\mathcal{H}$-matrices*

$$A \in \mathcal{H}(P^{\mathcal{H}}_{I\times J}, k_A, adm_A),\ B \in \mathcal{H}(P^{\mathcal{H}}_{J\times K}, k_B, adm_B),\ C \in \mathcal{H}(P^{\mathcal{H}}_{I\times K}, k_C, adm_C).$$

The exact product $C = A\cdot B$ - where C does not have the fixed rank k_C, and no truncation occurs at all - can be computed with a cost of at most

$$\mathcal{N}_{MM}(A,B,C) \leq C_{sp}(C)\cdot\left(k'_B\cdot(L_I+1)\cdot\mathcal{N}_{MV}(A) + k'_A\cdot(L_K+1)\cdot\mathcal{N}_{MV}(B)\right)$$

flops, where $k'_A = \max\{k_A, n_{min}\}$ and $k'_B = \max\{k_B, n_{min}\}$. The maximal rank $\widetilde{k}$ of an admissible block $C_{\sigma,\gamma}$ is then bounded by

$$(L_J+1)\cdot\left(C_{imb}(I,J)\cdot C_{sp}(P^{\mathcal{H}}_{I\times J}) + C_{imb}(J,K)\cdot C_{sp}(P^{\mathcal{H}}_{J\times K})\right)\cdot\max\{k_A,k_B\}.$$

Assume that $k_C < \widetilde{k}$. If we use the truncation from Theorem 2.42 to truncate the blocks to an arbitrary rank smaller than $\widetilde{k}$, we get an additional cost - independent from the choice of k_C - of at most

$$\mathcal{N}_{+}(A,B,C) \leq 6\widetilde{k}S_{\mathcal{H}}(P^{\mathcal{H}}_{I\times K},\widetilde{k}) + 23\widetilde{k}^3\#P^{\mathcal{H}}_{I\times K}.$$

If we use the fast truncation from Theorem 2.44 for truncation to rank k_C, we have an additional cost of at most

$$\mathcal{N}^{fast}_{+}(A,B,C) \leq (L_J+1)\cdot\min\{C_{sp}(A),C_{sp}(B)\}(24\max\{k_A,k_B\}S_{\mathcal{H}}(P^{\mathcal{H}}_{I\times K},k_C) + 184\max\{k_A,k_B\}^3\#P^{\mathcal{H},+}_{I\times K}).$$

The overall cost with fast truncation is given by

$$\mathcal{N}^{fast}_{MM}(A,B,C) = \mathcal{N}_{MM}(A,B,C) + \mathcal{N}^{fast}_{+}(A,B,C).$$

Proof Let $C_{\sigma,\gamma}$ be a leaf of C induced by $\sigma\times\gamma \in P^{\mathcal{H}}_{I\times K}$. Then we have

$$C_{\sigma,\gamma} = \sum_{\hat{\tau}\in P_J^{L^{max}(\sigma,\gamma,J)}} A_{\sigma,\hat{\tau}}B_{\hat{\tau},\gamma},$$

and by construction

$$\bigcup_{\hat{\tau}\in P_J^{L^{max}(\sigma,\gamma,J)}} \hat{\tau} = J.$$

Note that $A_{\sigma,\hat{\tau}}$ and $B_{\hat{\tau},\gamma}$ are not necessarily induced by a block in $P^{\mathcal{H}}_{I\times J}$ or $P^{\mathcal{H}}_{J\times K}$, respectively. We rewrite this sum with the first part of Theorem 2.46 to

$$C_{\sigma,\gamma} = \sum_{\ell=0}^{L_J} \sum_{\tau\in U_\ell(\sigma\times\gamma)} A_{\sigma,\tau} B_{\tau,\gamma}.$$

At least one of the two factors in every summand is now a block induced by an element of $P^{\mathcal{H}}_{I\times J}$ or $P^{\mathcal{H}}_{J\times K}$, respectively, because adm_C is induced by the $\mathcal{H}$-matrix multiplication. Using the second part of Theorem 2.46, this also means that the resulting rank of $C_{\sigma,\gamma}$ is bounded by

$$\begin{aligned}\widetilde{k} &\leq \sum_{0\leq\ell\leq L_J} \#U_\ell(\sigma\times\gamma)\max\{k_A,k_B\}\\ &\leq (L_J+1)\cdot\left(C_{imb}(I,J)\cdot C_{sp}(P^{\mathcal{H}}_{I\times J}) + C_{imb}(J,K)\cdot C_{sp}(P^{\mathcal{H}}_{J\times K})\right)\cdot\max\{k_A,k_B\}.\end{aligned}$$

The products $A_{\sigma,\tau}B_{\tau,\gamma}$ are computed as either $\max\{k_A,n_{min}\} = k'_A$ or $\max\{k_B,n_{min}\} = k'_B$ matrix vector products, depending on whether $A_{\sigma,\tau}$ or $B_{\tau,\gamma}$ is admissible. Hence, the overall cost is given by

$$\begin{aligned}\mathcal{N}_{MM}(A,B,C) &\leq \sum_{\sigma\times\gamma\in P^{\mathcal{H}}_{I\times K}} \sum_{\ell=0}^{L_J} \sum_{\tau\in U_\ell(\sigma\times\gamma)} \max\left\{k'_B\mathcal{N}_{MV}\left(A_{\sigma,\tau}\right), k'_A\mathcal{N}_{MV}\left(B_{\tau,\gamma}\right)\right\}\\ &\leq \sum_{\sigma\times\gamma\in P^{\mathcal{H}}_{I\times K}} k'_B\mathcal{N}_{MV}\left(A_{\sigma,J}\right) + k'_A\mathcal{N}_{MV}\left(B_{J,\gamma}\right),\end{aligned}$$

where we applied the first part of Theorem 2.46 in the second step. We review both summands independently. First, we have

$$\begin{aligned}\sum_{\sigma\times\gamma\in P^{\mathcal{H}}_{I\times K}} k'_B\mathcal{N}_{MV}\left(A_{\sigma,J}\right) &\leq C_{sp}(C)\cdot k'_B\cdot\sum_{\sigma\in P^{hier}_I}\mathcal{N}_{MV}\left(A_{\sigma,J}\right)\\ &= C_{sp}(C)\cdot k'_B\cdot\sum_{\ell=0}^{L_I}\sum_{\sigma\in P^\ell_I}\mathcal{N}_{MV}\left(A_{\sigma,J}\right)\\ &\leq C_{sp}(C)\cdot k'_B\cdot\sum_{\ell=0}^{L_I}\mathcal{N}_{MV}\left(A_{I,J}\right)\\ &= C_{sp}(C)\cdot k'_B\cdot(L_I+1)\cdot\mathcal{N}_{MV}\left(A_{I,J}\right).\end{aligned}$$

With the same argument for the other summand, we have

$$\sum_{\sigma\times\gamma\in P^{\mathcal{H}}_{I\times K}} k'_A\mathcal{N}_{MV}\left(B_{J,\gamma}\right) \leq C_{sp}(C)\cdot k'_A\cdot(L_K+1)\cdot\mathcal{N}_{MV}\left(B_{J,K}\right).$$

Combining both results leads to an overall cost of at most

$$C_{sp}(C) \cdot \left(k'_B \cdot (L_I + 1) \cdot \mathcal{N}_{MV}\left(A_{I,J}\right) + k'_A \cdot (L_K + 1) \cdot \mathcal{N}_{MV}\left(B_{J,K}\right)\right).$$

Only the truncation costs remain. We apply Theorem 2.42 for the first case and get

$$\mathcal{N}_+ \leq 6\widetilde{k} S_{\mathcal{H}}(P^{\mathcal{H}}_{I\times K}, \widetilde{k}) + 23\widetilde{k}^3 \# P^{\mathcal{H}}_{I\times K}.$$

For the second case, we use Theorem 2.44 and get

$$\begin{aligned}\mathcal{N}^{\text{fast}}_+ \leq (L_J + 1) \cdot \min\{C_{sp}(P^{\mathcal{H}}_{I\times J}), C_{sp}(P^{\mathcal{H}}_{J\times K}\}(&24 \max\{k_A, k_B\} S_{\mathcal{H}}(P^{\mathcal{H}}_{I\times K}, k)\\ &+184 \max\{k_A, k_B\}^3 \# P^{\mathcal{H}}_{I\times K})\end{aligned}$$

because there at most $(L_J + 1) \cdot \min\{C_{sp}(P^{\mathcal{H}}_{I\times J}), C_{sp}(P^{\mathcal{H}}_{J\times K}\}$ summands in the computation of an arbitrary leaf $A_{\sigma,\gamma}$. ∎

By applying Theorem 2.41, we can estimate the cost of the matrix-matrix multiplication of $\mathcal{H}$-matrices by considering the storage cost of both factors and the result. If the admissibility function of C is induced, then there is no hidden computational cost in the multiplication. This means that if all the matrices involved, including the resulting product, can be represented cost-efficiently as $\mathcal{H}$-matrices, then the product can be computed efficiently. In the upcoming chapters we will extend this result to our approach to the block Householder-based QR decomposition of $\mathcal{H}$-matrices.

We will also use the notation

$$\mathcal{N}_{MM}(A, B, C)$$

in a modified way. Assuming that all parameters of the $\mathcal{H}$-matrices A and B that are relevant for the cost estimate (e.g. sparsity constant, depth of the $\mathcal{H}$-matrix C etc.) are smaller than the corresponding ones of C, we will use

$$\mathcal{N}_{MM}(A, B, C) \leq \mathcal{N}_{MM}(C, C, C)$$

although the product $C \times C = C$ may not be defined or at least does not satisfy the conditions of Theorem 2.47. However, this will only be used to considerably shorten the notation in an effort to get to an overall bound by using the upper bounds from Theorem 2.47.

Beyond these basic $\mathcal{H}$-matrix arithmetics, further operations are available that efficiently use the $\mathcal{H}$-matrix representation. Aside from the LU decomposition, which we will consider in Chapter 5, the inversion of an $\mathcal{H}$-matrix and the Cholesky decomposition, among other things, are available. We refer once more to [43] for a detailed overview.

2.3 Further Matrix Formats

2.3.1 Sparse Matrices

Sparse matrices are matrices that are populated mainly by zero entries, but there is no generally accepted definition of how many non-zero entries a sparse matrix can have, and this is highly dependent on the intended use case [21]. The term *sparse matrix* itself was initially coined by Markowitz but further developed by others [68]. Often, a matrix is defined as sparse simply if it is advantageous to make use of its zero pattern [83, 76], and we will follow this convention here as well.

Definition 2.48 (Sparse matrix) *Let $A \in \mathbb{R}^{m\times n}$ be a matrix and N_{nz} the number of its non-zero entries. If N_{nz} is small enough so that it is beneficial to exploit the zero entries of A in numerical computations, then A is a sparse matrix.*

A matrix that is not sparse will be called *dense*. Any sparse matrix can be seen as a special case of an $\mathcal{H}$-matrix, where the leaf size is 1, subblocks containing only zero entries are associated with rank 0 admissible blocks of varying sizes and non-zero entries are associated with inadmissible (1 by 1) blocks. The highest level of the hierarchy of block partitionings contains every index pair as a block cluster, and the $\mathcal{H}$-block partition is, as usual, chosen so that the admissible blocks are of the largest possible size.

Similar to $\mathcal{H}$-matrices, the sparsity of a matrix can be used to significantly reduce storage and computational cost. And again, similar to $\mathcal{H}$-matrices, the pattern of the non-zero entries significantly influences the cost and fill-in that occurs for various operations. A common attempt to improve performance is by reordering the rows and columns. Unfortunately, these ideas cannot be easily transferred to $\mathcal{H}$-matrices. Reordering for $\mathcal{H}$-matrices with the same goal would not involve individual matrix columns but block columns. Due to the hierarchical structure, this is often not possible without splitting larger admissible blocks, which would create and not solve problems.

Aside from the general sparse matrix in Definition 2.48, we will also define two special types of sparse matrices with additional properties. All three are illustrated in Figure 2.9.

Definition 2.49 (Band matrix) *Let $A \in \mathbb{R}^{n\times n}$ be a matrix. If*

$$|i-j| \geq \ell \Rightarrow \; A_{ij} = 0$$

holds for all $0 < i, j \leq n$ with some $\ell \in \mathbb{N}$, then A is a band matrix with bandwidth

$$C_{bw}^{sparse} = \ell.$$

Following this definition, a band matrix with bandwidth ℓ has at most $2\ell - 1$ non-zero entries per row. If we see a band matrix as an $\mathcal{H}$-matrix with leaf size 1 and subblocks containing only zero entries as admissible blocks, then the bandwidth defined here would lead to a bandwidth constant C_{bw} from Theorem 2.29 of $C_{bw} = 2 \cdot C_{bw}^{sparse} - 1$.

Definition 2.50 (Skyline matrix) *Let $A \in \mathbb{R}^{n \times n}$ be a sparse matrix and $0 < i, j \leq n$. If for $j < i$*

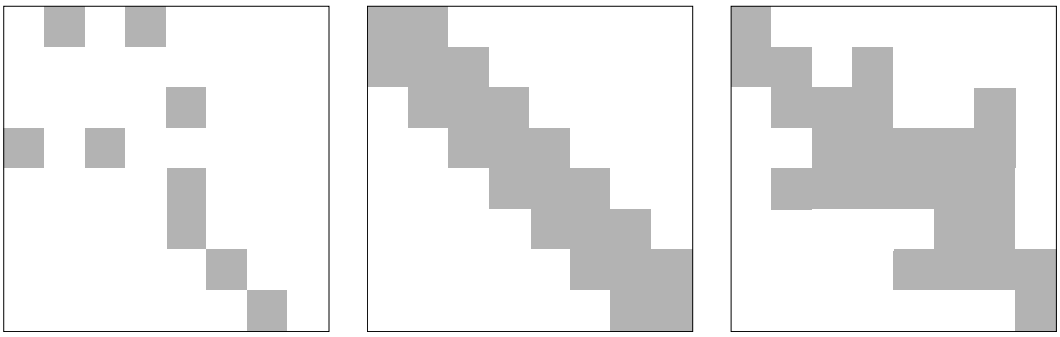
$$A_{ij} \neq 0 \;\Rightarrow\; A_{ik} \neq 0 \;\forall\; j \leq k \leq i$$

and for $i < j$

$$A_{ij} \neq 0 \;\Rightarrow\; A_{kj} \neq 0 \;\forall\; i \leq k \leq j$$

hold, then A is a skyline matrix.

The concept of skyline matrices was first introduced in [55], where a corresponding scheme to efficiently store such matrices was proposed. It can be seen that the sparsity pattern of a skyline matrix is inherited by its LU decomposition. We will prove a similar statement for $\mathcal{H}$-matrices in Chapter 5. The counterpart for $\mathcal{H}$-matrices has been formalized in Definition 2.27. For a skyline matrix interpreted as an $\mathcal{H}$-matrix, its lower triangular part is row-wise oriented, and its upper triangular part is column-wise oriented.



(a) Sparse Matrix. *(b)* Band Matrix with bandwidth 2. *(c)* Skyline Matrix.

Figure 2.9. Sparsity patterns for related matrix formats. White parts are zero, grey parts are non-zero.

2.3.2 $\mathcal{H}_p$-Matrices

An $\mathcal{H}_p$-matrix is a special type of $\mathcal{H}$-matrix often used as an introductory example to $\mathcal{H}$-matrices [43]. They were already mentioned in the first paper about $\mathcal{H}$-matrices

[42] and can be used in a range of applications such as in large-scale Gaussian process modeling [1], the stationary distribution of quasi-birth-death Markov chains [11], or matrix equations [59].

The main inspiration to work on a block Householder approach for $\mathcal{H}$-matrices also arose from a similar approach to the simpler $\mathcal{H}_p$-matrix format in [60], where these matrices are denoted as HODLR (hierarchically off-diagonal low-rank) matrices.

Definition 2.51 ($\mathcal{H}_p$-matrix) *Let $P^{\mathcal{H}}_{I\times I}$ be an $\mathcal{H}$-block partition of $I\times I$ of depth $p\in\mathbb{N}$, adm a corresponding admissibility function and $k\in\mathbb{N}$. If the admissibility function adm satisfies*

$$adm(\sigma_1\times\sigma_2)=\begin{cases}True & \sigma_1\neq\sigma_2\\ False & \sigma_1=\sigma_2\end{cases}$$

then

$$\mathcal{H}_p(P^{\mathcal{H}}_{I\times I},k,adm):=\{B\in\mathbb{R}^{I\times I}\mid rank(B|_b)\leq k \text{ for all admissible blocks } b\in P^{\mathcal{H}}_{I\times I}\}$$

is the set of $\mathcal{H}_p$-matrices with respect to $P^{\mathcal{H}}_{I\times I}$, adm and k.

The admissibility function used in Definition 2.51 is often called the weak admissibility function [47] in contrast to the standard admissibility function given in Definition 2.34. Due to their structure, $\mathcal{H}_p$-matrices usually arise from discretizations or approximations in one spatial dimension.

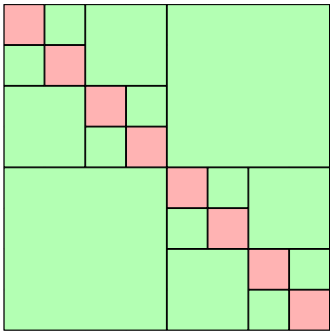


Figure 2.10. $\mathcal{H}_p$-matrix.

When analyzing the effect of addition and multiplication with $\mathcal{H}_p$-matrices, it can be quickly seen that they are data-sparse enough not to increase the number of inadmissible blocks in the result. In the literature, this is often proven by showing that the idempotency constant C_{id} is 1. Related statements will be of importance for us too, e.g. Theorem 3.8, but we will use a different approach using the boolean admissibility functions to prove them.

The definition of an $\mathcal{H}_p$-matrix can be extended to include general rectangular matrices. To do that, we cannot directly use the admissibility function from Definition 2.51 because the row and column index sets are no longer the same. But depending on the setting, for example by using the associated subsets X_σ and Y_τ from Subsection 2.2.2, we can achieve the desired structure. This will be relevant in our numerical examples in Subsection 4.2.1.

2.3.3 $\mathcal{H}^2$-Matrices

$\mathcal{H}^2$-matrices were first introduced in [45] for discrete elliptic problems and discrete integral operators from the boundary element method. They are very similar to $\mathcal{H}$-matrices but use a second hierarchical structure (hence the superscript ^{2}) to further reduce storage and computational cost.

An extensive introduction to $\mathcal{H}^2$-matrices can be found in [43, 13]. This short overview is inspired by [14] but builds upon the definition of $\mathcal{H}$-matrices seen in Subsection 2.2.1.

Definition 2.52 (Cluster basis) *Let $(P_I^\ell)_{\ell=0}^L$ be a hierarchy of partitionings and for every $\sigma \in P_I^{hier}$ there is a $k_\sigma \in \mathbb{N}$. Then the set*

$$K = \{k_\sigma \in \mathbb{N} : \sigma \in P_I^{hier}\}$$

is called a rank distribution of $(P_I^\ell)_{\ell=0}^L$. Furthermore, let there be a matrix $V^\sigma \in \mathbb{R}^{\#\sigma \times k_\sigma}$ for every $\sigma \in P_I^{hier}$. Then the set

$$V = \{V^\sigma \ : \sigma \in P_I^{hier}\}$$

is a cluster basis of $(P_I^\ell)_{\ell=0}^L$ with rank distribution K.

Similarly to the admissible blocks in $\mathcal{H}$-matrices, it is generally preferred that the ranks in the rank distribution are as small as possible.

Definition 2.53 (Nested cluster basis) *Let $(P_I^\ell)_{\ell=0}^L$ be a hierarchy of partitionings and V a corresponding cluster basis with rank distribution K. For $\sigma \subseteq \sigma'$ with $\sigma, \sigma' \in P_I^{hier}$, let $V_\sigma^{\sigma'}$ be the submatrix of $V^{\sigma'}$ that is induced by the row indices σ and the same column indices as $V^{\sigma'}$. Let there be matrices $E^\sigma \in \mathbb{R}^{k_\sigma \times k_{anc(\sigma)}}$ for every $\sigma \in P_I^{hier}$ with a direct ancestor of σ in P_I^{hier} denoted by $anc(\sigma)$ that satisfy*

$$V_\sigma^{anc(\sigma)} = V^\sigma E_\sigma.$$

Let the set E be given by

$$E = \{E_\sigma : \sigma \in P_I^{hier} \text{ with } anc(\sigma) \in P_I^{hier}\}.$$

Then V is a nested cluster basis with transfer matrices given by E.

See Figure 2.11 for a visualization of this definition.

Definition 2.54 ($\mathcal{H}^2$-matrix) *Let $P_{I\times J}^{\mathcal{H}}$ be an $\mathcal{H}$-block partition of $I \times J$ and adm a corresponding admissibility function. Let $P_{I\times J}^{\mathcal{H}}$ be based on the hierarchy of block partitionings $\left(P_{I\times J}^\ell\right)_{\ell=0}^L$ that itself is based in the hierarchy of partitionings $(P_I^\ell)_{\ell=0}^{L_I}$ and*

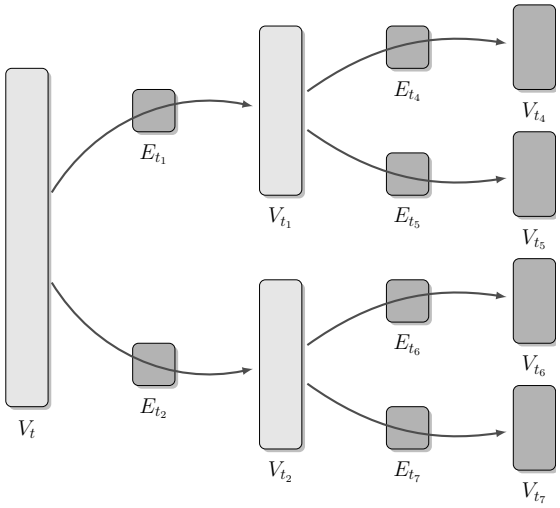


Figure 2.11. Nested cluster basis for an $\mathcal{H}^2$-matrix. Only the basis on the highest level and the transfer matrices need to be stored.

$(P_J^\ell)_{\ell=0}^{L_J}$. *Let V and W be the corresponding nested cluster bases for the rows and columns, respectively, with rank distribution k^V for V and k^W for W. Then*

$$\mathcal{H}^2(P^{\mathcal{H}}_{I\times J}, k^V, k^W, adm, V, W) := \big\{B \in \mathbb{R}^{I\times J} : \exists\ S_b \in \mathbb{R}^{k^V_\sigma \times k^W_\tau} \text{ with } B_{\sigma,\tau} = V^\sigma S_b W^{\tau,T}\ \forall\ b = \sigma\times\tau \in P^{\mathcal{H},+}_{I\times J}\big\}$$

is the set of $\mathcal{H}^2$-matrices with respect to $P^{\mathcal{H}}_{I\times J}$, adm, k^V, k^W, V and W. The matrices S_b are called coefficient matrices.

Compared to $\mathcal{H}$-matrices, we have a further reduction in storage cost because for each admissible block we only have to store the even smaller coefficient matrix as opposed to the larger low-rank factorization. We additionally have to save the row and column basis on the finest level and the corresponding transfer matrices, but this is considerably less if the $\mathcal{H}^2$-matrix is large enough. We have to do it only once for the rows and once for the columns and not independently of each other for every block.

It can be shown that, in contrast to $\mathcal{H}$-matrices, only $\mathcal{O}(n)$ data is needed to represent an $\mathcal{H}^2$-matrix for certain matrices from applications. This further extends to linear complexity in matrix-vector multiplication and some other operations [44].

2.4 Representation of Hierarchical Matrices in the Software Package H2Lib

The H2Lib [41] is a sequential library written in C that provides a wide variety of $\mathcal{H}$-matrix techniques, ranging from standard $\mathcal{H}$-matrix operations such as addition and multiplication to more sophisticated applications like a module for boundary element methods in 2D and 3D using $\mathcal{H}$-matrices.

The implementation for our numerical tests in Chapter 4 builds on these existing routines, adds new ones, and extends some others. This introduction is partly inspired by the longer one offered in [18], which, regardless of its primary focus, provides a good starting point for working with H2Lib. A detailed overview of all the features and intricacies of the H2Lib library can be found in the official documentation [41] and we give a short overview of our additions for the QR decomposition in Section 3.4. Note that most of the changes to the existing code were directly or indirectly necessary because certain operations, such as the $\mathcal{H}$-matrix multiplication, were not adapted to the specially structured rectangular $\mathcal{H}$-matrices we needed to use in the QR factorization of Theorem 2.15.

Central to any implementation involving $\mathcal{H}$-matrices is the way in which the $\mathcal{H}$-matrix itself is stored. This is done using the structure `hmatrix`, which is defined as follows:

```
struct hmatrix {
    pcluster     rc;        // Row cluster.
    pcluster     cc;        // Column cluster.
    prkmatrix    r;         // Low-rank matrix (admissible leaf).
    pamatrix     f;         // Standard matrix (inadmissible leaf).
    phmatrix     *son;      // Submatrices.
    uint         rsons;     // Number of block rows.
    uint         csons;     // Number of block columns.
    uint         refs;      // Number of references to this hmatrix.
    uint         desc;      // Number of descendants in matrix tree.
}
```

The `p` in front of the first 5 elements appears because all of them are included as pointers and not directly.

Using this structure, we can recursively define an $\mathcal{H}$-matrix. If the $\mathcal{H}$-matrix is a true $\mathcal{H}$-matrix, meaning it is not a leaf and does not consist of only one block, then `r` and `f` both point to a null pointer and `*son` points to an array of pointers to the sons. Suppose the $\mathcal{H}$-matrix only consists of one admissible block. In that case, `r` points to the corresponding `rkmatrix` object that includes, among other things, the matrices U and V that together serve as the low-rank approximation and both `f` and `*son` point to null pointers. If the $\mathcal{H}$-matrix consists only of one inadmissible block, then, by the same

logic, `f` points to the corresponding `amatrix` object containing the full matrix, whereas both `r` and `*son` point to null pointers.

At last, `rc` and `cc` point to the `cluster` objects that contain the information about the hierarchy of partitionings for the row index set (in `rc`) and the hierarchy of partitionings for the column index set (in `cc`) and are defined as follows:

```
struct cluster {
    uint        size;       // Number of indices.
    uint        *idx;       // Index set.
    uint        sons;       // Number of sons.
    pcluster    *son;       // Son clusters.
    uint        dim;        // Spatial dimension of bounding box.
    real        *bmin;      // Minimal coordinates of bounding box.
    real        *bmax;      // Maximal coordinates of bounding box.
    uint        desc;       // Number of descendants.
    uint        type;       // Type of cluster (not relevant for us).
}
```

Note again the `p` in front of `pcluster` because the array of pointers to the son clusters is included as a pointer. The index sets are saved in `*idx`, although in Chapter 3.4 we will denote the entire `cluster` object with σ, τ and γ to reduce notational clutter.

The `cluster` objects can be created using structures called `clustergeometry`, which include the index sets I and J for the rows and columns and additional data that defines how the partitions on every level of the hierarchy of partitionings should be created. Depending on this data, the i-th index in I does not necessarily induce the i-th row in the final $\mathcal{H}$-matrix, and the entries in `*idx` in the respective `cluster` object are reordered accordingly.

To get from the row and column `cluster` to the $\mathcal{H}$-matrix, an auxiliary structure called `block` can be used. This structure is similar to the final $\mathcal{H}$-matrix, except that it does not contain any pointers to the actual entries of the admissible or inadmissible blocks, only the information on which blocks are admissible and which are not. Unlike the `cluster`, the `block` does not contain the entire hierarchy of block partitionings, further splitting is stopped whenever an admissible block is reached. Hence, it is similar to the $\mathcal{H}$-block partition but still includes the hierarchy from the hierarchy of block partitionings. Thus, an appropriate admissibility function has to be chosen before creating the `block` object. To identify admissible and inadmissible blocks, the data from the entries `*bmin`, `*bmax` and `dim` in the row and column `cluster` objects, which describe the bounding box introduced in Subsection 2.2.2, can be used.

The last step is the calculation of the entries of the matrix when creating the `hmatrix` object. A suitable approximation technique, such as ACA, has to be chosen for the low-rank approximation of admissible blocks. The entries of all inadmissible blocks are

calculated directly.

One consequence of this hierarchical approach is that most algorithms processing $\mathcal{H}$-matrices have to be recursive and include calls to themselves. Otherwise, they cannot access all blocks of a given $\mathcal{H}$-matrix.

2.5 Radial Basis Functions

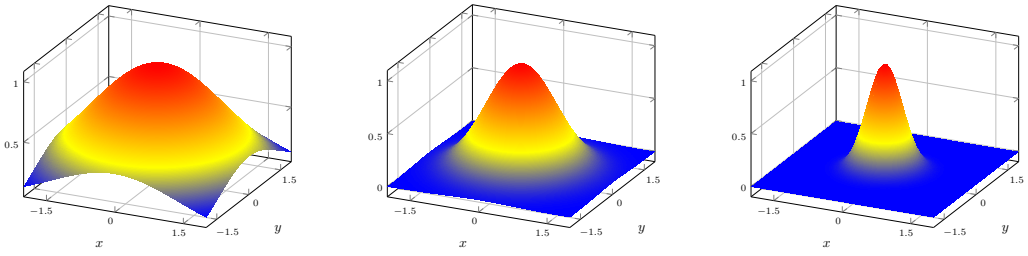
A radial basis function (RBF) is a bivariate function for which the function value only depends on the distance between both arguments, meaning ϕ is an RBF if we can find another function $\hat{\phi}$ so that

$$\phi(x, y) = \hat{\phi}(\|x - y\|)$$

holds for all $x, y \in \mathbb{R}^d$. The norm $\|\cdot\|$ is usually the Euclidean distance given by $\|\cdot\|_2$, although any other norm is possible as well. The most widely used example of an RBF is the Gaussian given by

$$k(x, y) = e^{-\varepsilon^2 \|x-y\|_2^2},$$

but there is a variety of other functions, including some with compact support [81, 73]. The parameter ε is called the shape parameter and serves an important function because it influences the shape of the Gaussian, as can be seen in Figure 2.12.



(a) $k(x, y) = e^{-\frac{1}{4}(x^2+y^2)}$. *(b)* $k(x, y) = e^{-(x^2+y^2)}$. *(c)* $k(x, y) = e^{-4(x^2+y^2)}$.

Figure 2.12. Illustration of Gaussian RBFs with center $(0, 0)$ and different shape parameters.

RBFs are used in a wide range of applications like the numerical solution of partial differential equations [71], Gaussian processes [77], and multivariate integration [79]. RBFs are also used for support vector classification [51], and the approximation by RBFs can be seen as a simple kind of neural network [17].

The underlying problem in many applications is the reconstruction of a multivariate

function from unstructured data. A self-contained introduction to this, including the analytical background, can be found in [82], but we will also offer a short introduction here. Given data

$$X = \{x_1, x_2, \ldots, x_N\} \subset \mathbb{R}^d,$$
$$Y = \{y_1, y_2, \ldots, y_N\} \subset \mathbb{R},$$

we want to find a (continuous) function f so that

$$f(x_i) = y_i \; \forall \; i = 1, \ldots, N$$

holds. Note that the elements of x_i in X are pairwise distinct. To be more precise, we want f to be a linear combination of, in our case, Gaussian kernels, meaning

$$f(x) = \sum_{j=1}^{N} c_j e^{-\varepsilon^2 \|x - x_j\|_2^2},$$

where c_j for $1 \leq j \leq N$ are the sought-after coefficients. We define $f_X, c_X \in \mathbb{R}^N$ as

$$f_X = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \quad c_X = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{pmatrix}.$$

This leads to the linear system

$$f_Y = A_{X,X} \cdot c_X,$$

where

$$A_{X,X} = (\phi(x_j, x_k))_{1 \leq j,k \leq N} \in \mathbb{R}^{N \times N}$$

is called the kernel matrix. In our case, we set $\phi(x_j, x_k) = k(x_j, x_k)$ to use the Gaussian RBFs. This also means that the matrix is positive definite independent of the point set X, and thus there is always a unique solution. Functions with this property are called positive definite. There is an overlap between RBFs and positive definite functions, but no inclusion of one in the other. There are positive definite functions that are not RBFs and vice versa [25]. However, we will only work with Gaussians, which are both positive definite and RBFs.

Reducing the number of centres but keeping the same number of data points turns this interpolation problem into an approximation problem. In that case, we set

$$X' = \{x'_1, x'_2, \ldots, x'_M\}$$

and define

$$c_{X'} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_M \end{pmatrix}.$$

Hence, we need to solve the least squares problem

$$\min_{c_{X'} \in \mathbb{R}^m} \|A_{X,X'} \cdot c_{X'} - f_X\|_2$$

with kernel matrix

$$A_{X,X'} = (\phi(x_j, x'_k))_{\substack{1 \le j \le N \\ 1 \le k \le M}} \in \mathbb{R}^{N \times M},$$

where again we set $\phi(x_j, x'_k) = k(x_j, x'_k)$. As mentioned above, the choice of the parameter ε defines the shape of our kernel functions and thus indirectly influences the solution of this data approximation problem. We can observe one crucial effect. If ε decreases, the condition number of the kernel matrix increases but the solution to the approximation problem improves [70].

At some point in a numerical setting, the increasing condition number offsets any possible gain that could be made by further decreasing the shape parameter due to the associated numerical instability. There is some discussion about the choice of the "optimal" shape parameter [27, 78] and efforts are made to mitigate these problems by choosing a stable basis [28, 63], but this is not our main focus here.

However, it is important to realize that these limitations, especially the fact that we cannot expect a low condition number of our kernel matrix, are important factors in deciding how to solve the approximation problem numerically. To increase the accuracy as much as possible, we need to accept a condition number for the kernel matrix that is relatively high. That means that in the case of the approximation problem, we likely should avoid forming the normal equations

$$A_{X,X'}^T A_{X,X'} c_{X'} = A_{X,X'}^T f_X$$

due to the squaring of the condition number in $A_{X,X'}^T A_{X,X'}$ compared to $A_{X,X'}$.

The Gaussian kernel matrices presented here are inherently suited to be approximated by $\mathcal{H}$-matrices using the standard admissibility function, where the associated subset X_σ of an index set σ consists of the corresponding data points, meaning we set

$$X_\sigma := \{x_i \mid i \subset \sigma\}.$$

This was already done in [53] for the kernel matrix of the interpolation problem, and we can easily extend it to the rectangular matrices of the approximation problems. We will restrict ourselves to using ACA as described in Subsection 2.2.2 to derive an $\mathcal{H}$-matrix approximation of the kernel matrix.

2.6 Boundary Element Method

The boundary element method (BEM) is an important computational tool to solve partial differential equations numerically. Compared to other methods like the finite element method (FEM) or the Finite Difference Method (FDM) , where the numerical discretization is performed on the domain, for BEM it is performed only on the boundary and hence at a lower spatial dimension. This often leads to an advantage in computation time because the resulting linear systems are smaller. Unfortunately, BEM is not as generally applicable because it requires explicit knowledge of a fundamental solution of the differential equation, which is not always available. Hence, we are restricted to linear partial differential equations with constant or specifically variable coefficients.

We base this short introduction on [20, 57] and will only introduce the BEM as a solver for the Laplace equation because that will be our model problem later. For a more detailed overview, see the above sources or the books [54, 16, 3]. An overview of the history of BEM can be found in [19].

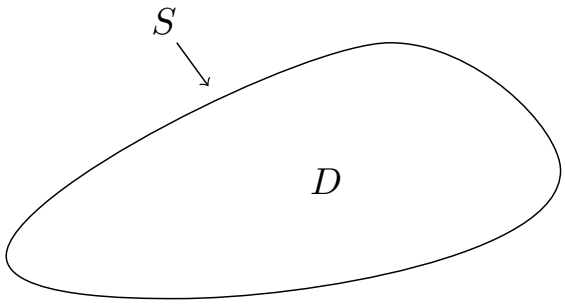


Figure 2.13. llustration of the interior domain D and the boundary S.

Our model problem is the two-dimensional Laplace equation

$$\nabla^2 \phi(p) = 0 \quad (p \in D)$$

in an interior domain D with a smooth boundary S and a Robin boundary condition, i.e. the function ϕ thus has to satisfy

$$\alpha(p)\phi(p) + \beta(p)\frac{\partial \phi}{\partial n_p}(p) = f(p) \qquad (p \in S),$$

where n_p is the unit outward normal to the boundary at p while α, β, and f are real-valued functions. For $\alpha(p) = 1, \beta(p) = 0$ we hence have the Dirichlet form, and for $\alpha(p) = 0, \beta(p) = 1$ we have the Neumann form.

The first step in applying the boundary element method is to reformulate this partial differential equation into an integral equation defined on the domain's boundary and an integral that relates the boundary solution to the solution at points in the domain.

Historically, there were two main approaches to finding this equivalent equation, the direct and the indirect method. Assume that ϕ and ψ are twice-differentiable scalar functions in the domain D bounded by the closed surface S and ϕ solves the two-dimensional Laplace equation. Then the direct method uses Green's second identity

$$\int_D (\phi(q)\nabla^2\psi(q) - \psi(q)\nabla^2\phi(q))dV_q = \int_S \phi(q)\frac{\partial\psi(q)}{\partial n_q} - \psi(q)\frac{\partial\phi(q)}{\partial n_q}dS_q$$

and the fundamental solution of the corresponding PDE - in our case, the two-dimensional Laplace equation - given by

$$G(p,q) = -\frac{1}{2\pi}\ln(\|p-q\|),$$

to get the boundary integral equation

$$\int_S \phi(q)\frac{\partial G(p,q)}{\partial n_q} - G(p,q)\frac{\partial\phi(q)}{\partial n_q}dS_q = -\frac{1}{2}\phi(p).$$

For the indirect method, we assume that

$$\phi(p) = \int_S G(p,q)\omega(q)dS_q \qquad (p \in D \cup S)$$

holds, meaning ϕ is related by a layer potential on the boundary. This then leads to the boundary integral equation

$$\frac{\phi(p)}{\partial n_p} = \int_S \frac{\partial G(p,q)}{\partial n_p}\sigma(q)dS_q + \frac{1}{2}\sigma(p) \qquad (p \in S).$$

We can modify both approaches in several ways, e.g. by further differentiating in the direct method or using the double layer potential in the indirect method. Those changes are not necessarily needed in our application to the Laplace problem but could help in a different situation. More details about the intricacies of the different approaches can be found in the mentioned sources, especially in [56].

Because the data on the boundary is known, we can use these equations to find the unknown function ϕ. To do that numerically, we need to discretise the integral equation

using a numerical integration technique and approximate the boundary functions. One widely used approach is the method of collocation, in which we have to choose a suitable set of points and force the equation to be satisfied at these points. A different option would be the Galerkin approach, in which we multiply our boundary integral equation with test functions from a finite-dimensional functional space.

We start the discretization process and approximate the boundary in our two-dimensional case simply as a set of m line segments. This is often called a set of panels. Hence, we have

$$S \approx \widetilde{S} = \sum_{j=1}^{n} \Delta S_j,$$

where each S_j is a line segment. We approximate the boundary functions on each panel by a constant and thus assume that for all $x \in S_j$ we have

$$\begin{aligned} \phi(x) &= \phi_j, \\ \frac{\partial \phi}{\partial n_q} &= t_j, \\ x &= X_i, \end{aligned}$$

where X_j is the midpoint of S_j. Using the direct approach from above, we get

$$\sum_{j=1}^{m} \frac{1}{2\pi} \int_{S_j} t_j \ln\left(X_i - y\right) - \phi_j \frac{\partial}{\partial n_y} \left(\ln(X_i - y)\right) dS_q = \frac{1}{2}\phi_i$$

for every $1 \leq j \leq n$. The integral on the left side can be computed analytically or numerically, after which we can write the linear system of equations in matrix form, which then has to be solved to find our approximation for ϕ. At this point, the $\mathcal{H}$-matrix approximation can be used. Usually, the arising matrix is, different from matrices that appear in the finite element method, not sparse. However, it can be shown that in certain applications it can be approximated well as an $\mathcal{H}$-matrix [42]. We only want to sketch the idea here.

Assume we are interested in approximating the matrix $A \in \mathbb{R}^{n \times n}$ with entries a_{ij} given by

$$a_{ij} = \int_{x_{j-1}}^{x_j} \ln(X_i - y),$$

where X_i is again the midpoint of the line between x_{i-1} and x_i. Matrices of this type appear in the linear system for the boundary element method we developed above. We

can then replace the kernel function $K(x, y) = \ln(x - y)$ with an approximation

$$\widetilde{K}(x, y) = \sum_{\ell=1}^{k} g_\ell(x) \cdot h_\ell(y),$$

by using Taylor's formula applied with respect to y. Using this approximation for all entries in admissible blocks, we can easily find local low-rank approximations for these submatrices of A with rank k.

Analyzing the corresponding error estimates, we can see that, with the standard admissibility function, this approach leads to a useful $\mathcal{H}$-matrix approximation.

After solving this system efficiently by exploiting the $\mathcal{H}$-matrix structure of A, we have found the missing boundary values. If we are only interested in those, we may stop here. Otherwise, we can apply these to the results initially used to find the boundary integral equation and obtain the solution for any point in the interior domain D.

Chapter 3

QR Decomposition for Hierarchical Matrices

3.1 Known QR Decompositions for Hierarchical Matrices

Several different ways to compute the QR decomposition of $\mathcal{H}$ matrices have been proposed. We will present them here and in the later Chapter 4 compare some of them numerically with our newly developed method. These methods have already been discussed in the literature, e.g. in [67, 60], so we will keep this review brief.

Algorithm 3.1: Cholesky decomposition of a symmetric positive definite $\mathcal{H}$-Matrix A.

Data: Symmetrix positive definite $\mathcal{H}$-matrix $A \in \mathcal{H}(P_{I\times I}^{\mathcal{H},A}, k_A, \mathrm{adm}_A)$
Result: Lower triangular $\mathcal{H}$-matrix $L \in \mathcal{H}(P_{I\times I}^{\mathcal{H},L}, k_L, \mathrm{adm}_L)$ with $A \approx L \cdot L^T$
Function: $[L] = \mathcal{H}Cholesky(A)$

if *there are no* $\sigma_1, \sigma_2 \in P_I^{hier}$ *with* $\sigma_1 \cap \sigma_2 = \emptyset$ *and* $\sigma_1 \cup \sigma_2 = I$ **then**
  Compute the dense Cholesky decomposition $L \cdot L^T = A$ with $L \in \mathbb{R}^{n,n}$;
else
  Find $\sigma_1, \sigma_2 \in P_I^{hier}$ with $\sigma_1 \cap \sigma_2 = \emptyset$ and $\sigma_1 \cup \sigma_2 = \sigma$;
  $L_{\sigma_1,\sigma_1} = \mathcal{H}Cholesky(A_{\sigma_1,\sigma_1})$;
  $L_{\sigma_2,\sigma_1}^T = \mathcal{H}FwSub(L_{\sigma_1,\sigma_1}, A_{\sigma_1,\sigma_2})$;
  $L_{\sigma_2,\sigma_2} = \mathcal{H}Cholesky(A_{\sigma_2,\sigma_2} - L_{\sigma_2,\sigma_1} \cdot L_{\sigma_2,\sigma_1}^T)$;
  Set $L = \begin{bmatrix} L_{\sigma_1,\sigma_1} & \\ L_{\sigma_2,\sigma_1} & L_{\sigma_2,\sigma_2} \end{bmatrix}$;
end
return L;

3.1.1 Cholesky-based $\mathcal{H}$-QR Decomposition

Algorithm 3.2: Forward substitution.

Data: Lower triangular $\mathcal{H}$-matrix $L \in \mathcal{H}(P_{I\times I}^{\mathcal{H},L}, k_L, \mathrm{adm}_L)$ and $\mathcal{H}$-matrix $A \in \mathcal{H}(P_{I\times J}^{\mathcal{H},A}, k_A, \mathrm{adm}_A)$
Result: $\mathcal{H}$-matrix $B \in \mathcal{H}(P_{I\times J}^{\mathcal{H},B}, k_B, \mathrm{adm}_B)$ with $L \cdot B = A$
Function: $[B] = \mathcal{H}FwSub(L, A)$

if *there are no* $\sigma_1, \sigma_2 \in P_I^{hier}$ *with* $\sigma_1 \cap \sigma_2 = \emptyset$ *and* $\sigma_1 \cup \sigma_2 = I$ **then**
  Solve the dense equation $L \cdot B = A$ for B with forward substitution;
else
  Find $\sigma_1, \sigma_2 \in P_I^{hier}$ with $\sigma_1 \cap \sigma_2 = \emptyset$ and $\sigma_1 \cup \sigma_2 = I$;
  if *there are* $\tau_1, \tau_2 \in P_J^{hier}$ *with* $\tau_1 \cap \tau_2 = \emptyset$, $\tau_1 \cup \tau_2 = J$ *and* $\tau_1 \times \sigma_1 \in P_{I\times J}^{hier}$ **then**
    $B_{\sigma_1,\tau_1} = \mathcal{H}FwSub(L_{\sigma_1,\sigma_1}, A_{\sigma_1,\tau_1})$;
    $B_{\sigma_1,\tau_2} = \mathcal{H}FwSub(L_{\sigma_1,\sigma_1}, A_{\sigma_1,\tau_2})$;
    $B_{\sigma_2,\tau_1} = \mathcal{H}FwSub(L_{\sigma_2,\sigma_2}, A_{\sigma_2,\tau_1} - L_{\sigma_2,\sigma_1} \cdot B_{\sigma_1,\tau_1})$;
    $B_{\sigma_2,\tau_2} = \mathcal{H}FwSub(L_{\sigma_2,\sigma_2}, A_{\sigma_2,\tau_2} - L_{\sigma_2,\sigma_1} \cdot B_{\sigma_1,\tau_2})$;
    Set $B = \begin{bmatrix} B_{\sigma_1,\tau_1} & B_{\sigma_1,\tau_2} \\ B_{\sigma_2,\tau_1} & B_{\sigma_2,\tau_2} \end{bmatrix}$;
  else
    $B_{\sigma_1,\tau} = \mathcal{H}FwSub(L_{\sigma_1,\sigma_1}, A_{\sigma_1,\tau})$;
    $B_{\sigma_2,\tau} = \mathcal{H}FwSub(L_{\sigma_2,\sigma_2}, A_{\sigma_2,\tau} - L_{\sigma_2,\sigma_1} \cdot B_{\sigma_1,\tau})$;
    Set $B = \begin{bmatrix} B_{\sigma_1,\tau} \\ B_{\sigma_2,\tau} \end{bmatrix}$;
  end
end
return B;

In [66], Lintner developed an approach based on the Cholesky decomposition, which is for dense matrices often called Cholesky QR. Let $A = QR$ be the QR decomposition of A with upper triangular R with positive diagonal entries, and let LL^T be the Cholesky decomposition of A^TA. Then using

$$A^TA = R^TQ^TQR = R^TR = LL^T,$$

we can see that the upper triangular matrix R in the QR decomposition of A is the transpose of the lower triangular matrix L in the Cholesky decomposition of A^TA. Because the computation of the Cholesky decomposition of an $\mathcal{H}$-matrix as well as the multiplication A^TA is available in $\mathcal{H}$-matrix arithmetic, we can apply it to A^TA to compute the upper triangular part of the QR decomposition. Then Q is calculated by an upper triangular solve from $A = QR$. An algorithmic representation of this approach can be found in Algorithm 3.4 for a bisection-based $\mathcal{H}$-block partition. As explained, it

uses the (efficient) Cholesky decomposition of $\mathcal{H}$-matrices, which is given in algorithmic form in Algorithm 3.1, the forward substitution as given in Algorithm 3.2 and the backward substitution as given in Algorithm 3.3, all three also for a bisection-based $\mathcal{H}$-block partition.

Algorithm 3.3: Backward substitution.

Data: Upper triangular $\mathcal{H}$-matrix $U \in \mathcal{H}(P_{I\times I}^{\mathcal{H},U}, k_U, \mathrm{adm}_U)$ and $\mathcal{H}$-matrix $A \in \mathcal{H}(P_{J\times I}^{\mathcal{H},A}, k_A, \mathrm{adm}_A)$
Result: $\mathcal{H}$-matrix $B \in \mathcal{H}(P_{J\times I}^{\mathcal{H},B}, k_B, \mathrm{adm}_B)$ with $B \cdot U = A$
Function: $[B] = \mathcal{H}BwSub(U, A)$

if *there are no* $\sigma_1, \sigma_2 \in P_I^{hier}$ *with* $\sigma_1 \cap \sigma_2 = \emptyset$ *and* $\sigma_1 \cup \sigma_2 = I$ **then**
  Solve the dense equation $B \cdot U = A$ for B with backward substitution;
else
  Find $\sigma_1, \sigma_2 \in P_I^{hier}$ with $\sigma_1 \cap \sigma_2 = \emptyset$ and $\sigma_1 \cup \sigma_2 = I$;
  if *there are* $\tau_1, \tau_2 \in P_J^{hier}$ *with* $\tau_1 \cap \tau_2 = \emptyset$, $\tau_1 \cup \tau_2 = J$ *and* $\tau_1 \times \sigma_1 \in P_{J\times I}^{hier}$ **then**
    $B_{\tau_1,\sigma_1} = \mathcal{H}BwSub(U_{\sigma_1,\sigma_1}, A_{\tau_1,\sigma_1})$;
    $B_{\tau_2,\sigma_1} = \mathcal{H}BwSub(U_{\sigma_1,\sigma_1}, A_{\tau_2,\sigma_1})$;
    $B_{\tau_1,\sigma_2} = \mathcal{H}BwSub(U_{\sigma_2,\sigma_2}, A_{\tau_1,\sigma_2} - B_{\tau_1,\sigma_1}U_{\sigma_1,\sigma_2})$;
    $B_{\tau_2,\sigma_2} = \mathcal{H}BwSub(U_{\sigma_2,\sigma_2}, A_{\tau_2,\sigma_2} - B_{\tau_2,\sigma_1}U_{\sigma_1,\sigma_2})$;
    Set $B = \begin{bmatrix} B_{\tau_1,\sigma_1} & B_{\tau_1,\sigma_2} \\ B_{\tau_2,\sigma_1} & B_{\tau_2,\sigma_2} \end{bmatrix}$;
  else
    $B_{\tau,\sigma_1} = \mathcal{H}BwSub(U_{\sigma_1,\sigma_1}, A_{\tau,\sigma_1})$;
    $B_{\tau,\sigma_2} = \mathcal{H}BwSub(U_{\sigma_2,\sigma_2}, A_{\tau,\sigma_2} - B_{\tau,\sigma_1}U_{\sigma_1,\sigma_2})$;
    Set $B = \begin{bmatrix} B_{\tau,\sigma_1} & B_{\tau,\sigma_2} \end{bmatrix}$;
  end
end
return B;

The first and most obvious problem using this method is the squaring of the condition number because

$$\kappa(A^T A) = \kappa(A)^2.$$

Lintner proposed to use the polar decomposition to reduce the condition of the problem.

Another question that arises from the ill-conditioning is whether the resulting Q is orthogonal. The inherent inaccuracy of the $\mathcal{H}$-matrix arithmetics and the arising error propagation during the Cholesky decomposition could lead to poor orthogonality in Q. We will later observe this in our numerical results. Similar to the idea for the dense case

Algorithm 3.4: Cholesky-based QR decomposition.

Data: $A \in \mathcal{H}(P_{I\times J}^{\mathcal{H},A}, k_A, \mathrm{adm}_A)$
Result: $Q \in \mathcal{H}(P_{I\times J}^{\mathcal{H},Q}, k_Q, \mathrm{adm}_Q)$ and $R \in \mathcal{H}(P_{J\times J}^{\mathcal{H},R}, k_R, \mathrm{adm}_R)$ defining an approximate QR decomposition of A by $A \approx Q \cdot R$
Function: $[Q, R] = \mathcal{H}QRCholesky(A)$

1 Set $B \in \mathcal{H}(P_{J\times J}^{\mathcal{H},B}, k_B, \mathrm{adm}_B)$ with $\mathcal{H}(P_{J\times J}^{\mathcal{H},B})$ induced by $A^T \cdot A$;
2 $B = A^T \cdot A$;
3 Set $L \in \mathcal{H}(P_{J\times J}^{\mathcal{H},L}, k_L, \mathrm{adm}_L)$ with $\mathcal{H}(P_{J\times J}^{\mathcal{H},L})$ induced by the Cholesky decomposition of B;
4 $L = \mathcal{H}Cholesky(B)$;
5 Set $Q \in \mathcal{H}(P_{I\times J}^{\mathcal{H},Q}, k_Q, \mathrm{adm}_Q)$ with $\mathcal{H}(P_{I\times J}^{\mathcal{H},Q})$ induced by $Q = A \cdot \left(L^T\right)^{-1}$;
6 $Q = \mathcal{H}BwSub(L^T, A)$;
7 **return** Q, R;

in [84], Lintner proposed applying the algorithm more than once to alleviate this problem. We will refrain from using this approach later because the increase in computation time would make this variant of the algorithm not competitive. It is also important to mention that the Cholesky-based $\mathcal{H}$-QR approach only computes the reduced QR factorization for rectangular matrices.

3.1.2 Gram-Schmidt $\mathcal{H}$-QR Decomposition

In [10], another approach based on a block Gram Schmidt procedure was presented. The main idea was already shown in Theorem 2.15, which will also be the basis for our new algorithm. A given $\mathcal{H}$-matrix A is partitioned into two block columns A_1 and A_2 of roughly the same size, and the QR decomposition

$$\begin{pmatrix} A_1 & A_2 \end{pmatrix} = \begin{pmatrix} Q_1 & Q_2 \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$$

is to be computed. As shown in the aforementioned theorem, this can be done in three steps [60]:

1. Compute the QR decomposition $A_1 = Q_1 R_{11}$.
2. Compute $R_{12} = Q_1^T A_2$ and $\widetilde{A}_2 = A_2 - Q_1 R_{12}$.
3. Compute the QR decomposition $\widetilde{A}_2 = Q_2 R_{22}$.

Assuming the QR decompositions of A_1 and $\widetilde{A}_2$ are available as $\mathcal{H}$-matrices, a QR

decomposition of A can also be expressed as an $\mathcal{H}$-matrix because then R_{12} and $\widetilde{A}_2$ themselves can be computed by $\mathcal{H}$-matrix arithmetics. For the QR decompositions of A_1 and $\widetilde{A}_2$, we recursively apply the same idea until we have reached the highest level of the hierarchical block partitioning. The QR decomposition of these block columns can be calculated cost-efficiently while staying in the $\mathcal{H}$-matrix format if the factor U of all admissible blocks is orthogonal. This process closely resembles our approach, which we will present in the upcoming chapter. This includes a detailed description of the necessary vertical splits for larger, admissible blocks in Subsection 3.2.2. One major difference between both approaches is that Q is explicitly calculated in the Gram-Schmidt $\mathcal{H}$-QR decomposition. We will only indirectly compute Q by applying the recursion on the decomposition $Q = I - YTY^T$ whose existence was shown in Theorem 2.15.

Algorithm 3.5: Gram-Schmidt $\mathcal{H}$-QR decomposition - the recursion start.

Data: $A \in \mathcal{H}(P_{I\times J}^{\mathcal{H},A}, k_A, \mathrm{adm}_A)$ with $\mathrm{depth}_{col}(P_{I\times J}^{\mathcal{H},A}) = 0$
Result: $Q \in \mathcal{H}(P_{I\times J}^{\mathcal{H},Q}, k_Q, \mathrm{adm}_Q)$ and $R \in \mathcal{H}(P_{J\times J}^{\mathcal{H},R}, k_R, \mathrm{adm}_R)$ defining a QR decomposition of $A = Q \cdot R$
Function: $[Q, R] = \mathcal{H}GramSchmidt_{high}(A)$

Create the dense matrix $\widetilde{A} \in \mathbb{R}^{\widetilde{m}\times \#J}$ consisting of blocks $\widetilde{A}_i$ as in Theorem 3.1;
for *all subblocks* A_i *of* A **do**
  if A_i *is a low-rank matrix given by* $A_i = U_i \cdot V_i^T$ *with* $U_i^T U_i = I_{k_i}$ **then**
    Set $\widetilde{A}_i = V_i^T$;
  else
    Set $\widetilde{A}_i = A_i$;
  end
end
Compute the dense QR decomposition $\widetilde{Q} \cdot \widetilde{R} = \widetilde{A}$ with $\widetilde{Q} \in \mathbb{R}^{\widetilde{m},\#J}$ and $\widetilde{R} \in \mathbb{R}^{\#J,\#J}$;
for *all subblocks* A_i *of* A **do**
  if A_i *is a low-rank matrix given by* $A_i = U_i \cdot V_i^T$ **then**
    Set $Q_{\sigma_i,J} = U_i \cdot \widetilde{Q}_i$; `//` σ_i `is the row index set of` A_i`.`
  else
    Set $Q_{\sigma_i,J} = \widetilde{Q}_i$; `//` σ_i `is the row index set of` A_i`.`
  end
end
Set $P_{J\times J}^{\mathcal{H},R} = \{J \times J\}$, $R = \widetilde{R}$;
return Q, R;

For a more detailed explanation, we also refer to the original source [10]. An algorithmic representation of this approach is shown as Algorithm 3.5 and Algorithm 3.6. Again we suffer from numerical instability because the inherent problems of the Gram-Schmidt orthogonalization transfer to the $\mathcal{H}$-matrix case as well.

Algorithm 3.6: Gram-Schmidt $\mathcal{H}$-QR decomposition.

Data: $A \in \mathcal{H}(P_{I\times J}^{\mathcal{H},A}, k_A, \mathrm{adm}_A)$
Result: $Q \in \mathcal{H}(P_{I\times J}^{\mathcal{H},Q}, k_Q, \mathrm{adm}_Q)$ and $R \in \mathcal{H}(P_{J\times J}^{\mathcal{H},R}, k_R, \mathrm{adm}_R)$ defining a QR decomposition of A by $A = Q \cdot R$
Function: $[Q, R] = \mathcal{H}GramSchmidt(A)$

```
if there are no τ1, τ2 ∈ P_J^hier with τ1 ∪ τ2 = J and τ1 ∩ τ2 = ∅ then
    [Q, R] = HGramSchmidt_high(A);
else
    Find τ1, τ2 ∈ P_J^hier with τ1 ∪ τ2 = J and τ1 ∩ τ2 = ∅;
    Partition A=[A_{I,τ1} A_{I,τ2}];
    [Q_{I,τ1}, R_{τ1,τ1}] = HGramSchmidt(A_{I,τ1});
    R_{τ1,τ2} = Q_{I,τ1}^T · A_{I,τ2};
    (Q_{I,τ2}, R_{τ2,τ2}) = HGramSchmidt(A_{I,τ2} − Q_{I,τ1} · R_{τ1,τ2});
end
return Q, R;
```

Furthermore, the version presented in [10] uses the admissibility function of A for Q. As we will see in Subsection 3.3.3, this is often very different from the induced admissibility function of Q, which is unfortunately comprised of predominantly inadmissible blocks in many cases. Hence, using this version of the algorithm can lead to high ranks in admissible blocks or low accuracy and orthogonality, depending on the truncation approach.

It is important to note that the numerical results in [10] do not reflect this problem because they test only the 2D Laplacian matrix for which the conditions of the upcoming Theorem 3.3.3 are not satisfied.

3.1.3 Bebendorf's $\mathcal{H}$-QR Decomposition

Bebendorf suggested another approach in [8]. The QR decomposition is computed by orthogonalizing the results of a blockwise LU decomposition. For an $\mathcal{H}$-matrix A that can be partitioned as

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix},$$

we first consider the block LU decomposition

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} I & 0 \\ X & I \end{pmatrix} \cdot \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} - XA_{12} \end{pmatrix},$$

where $X = A_{21}A_{11}^{-1}$. Scaling with the inverted Cholesky factors of $I + X^TX = R_1^TR_1$ and $I + XX^T = R_2^TR_2$, we get

$$A = \underbrace{\begin{pmatrix} I & -X^T \\ X & I \end{pmatrix}\begin{pmatrix} R_1^{-1} & 0 \\ 0 & R_2^{-1} \end{pmatrix}}_{=:\widetilde{Q}} \underbrace{\begin{pmatrix} R_1A_{11} & R_1^{-T}(A_{12} + X^TA_{22}) \\ 0 & R_2^{-T}(A_{22} - XA_{12}) \end{pmatrix}}_{=:\widetilde{R}}.$$

$\widetilde{Q}$ is orthogonal by construction, and we can recursively apply the procedure to the diagonal blocks of $\widetilde{R}$. We only need standard $\mathcal{H}$-matrix arithmetic, $\mathcal{H}$-matrix inversion, and the Cholesky decomposition, which are all available so that the result will be an $\mathcal{H}$-matrix again.

Relying on the inversion of the leading subblock A_{11} at every recursion step is, however, a problem. We cannot assume the existence of this inverse for every matrix that has a QR decomposition, and, even if there is one, the calculation is not necessarily numerically stable. Due to these problems, we refrain from using this approach in our numerical tests in Chapter 4.

3.2 Block Householder Based Algorithm for the $\mathcal{H}$-QR Decomposition

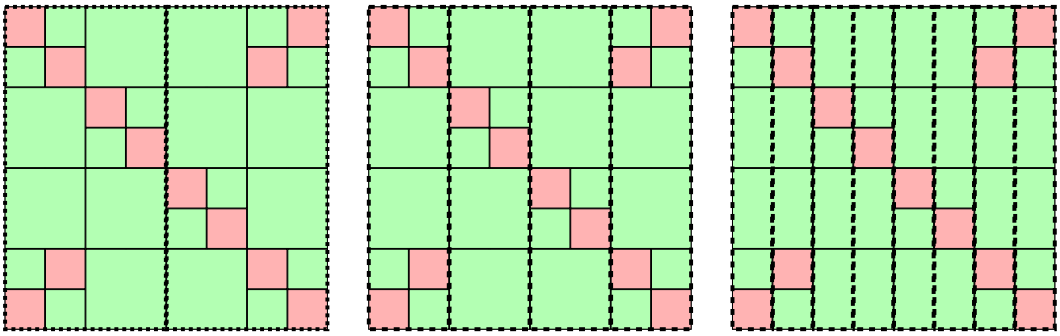
We aim to combine the recursive computation of the QR decomposition of Theorem 2.15 with $\mathcal{H}$-matrix arithmetic to compute the QR decomposition of an $\mathcal{H}$-matrix. For each addition and multiplication of matrix subblocks, we use the induced admissibility functions of Definitions 2.24 and 2.25 to determine the $\mathcal{H}$-block partition of the resulting $\mathcal{H}$-matrix. Furthermore, for the orthogonal matrix $Q = I - YTY^T$, we compute and store T in its reduced form $\overline{T}$ and then perform all subsequent matrix-matrix multiplications with T as explained in Remark 2.16. As noted before, this approach, but not the upcoming theoretical results from Section 3.3, was already presented by us in [40].

Let $A \in \mathbb{R}^{I\times\tau}$ be an $\mathcal{H}$-matrix based on hierarchical (block) partitionings of the index sets I, J and $I \times J$, consisting only of the block column associated with the column cluster $\tau \in P_J^{\text{hier}}$ of the $\mathcal{H}$-block partition $P_{I\times J}^{\mathcal{H}}$. If the column index set $\tau \in P_J^{\text{hier}}$ is a column leaf of $P_{I\times J}^{\mathcal{H}}$, then all blocks in the $\mathcal{H}$-block partition of $A \in \mathbb{R}^{I\times\tau}$ have column index set τ. This will be the start of the recursion, and we will compute the full QR decomposition of A in implicit representation. To do so, we can exploit the $\mathcal{H}$-matrix structure (especially its admissible blocks) of A, which is explained in the upcoming Subsection 3.2.1.

If τ is not a column leaf, then we need to vertically split the $\mathcal{H}$-matrix further. We do this by using the first two non-trivial descendants $\tau_1, \tau_2 \in P_J^{hier}$ of τ so that $\tau_1 \cup \tau_2 = \tau$

holds. The intricacies of this step are explained in the upcoming Subsection 3.2.2, and the splitting process is shown in Figure 3.1.

The underlying idea has already been used for the block Gram-Schmidt algorithm for $\mathcal{H}$-matrices [10] as shown in Subsection 3.1.2 and in [60] for a QR factorization for hierarchically off-diagonal low-rank (HODLR) matrices. We defined HODLR matrices as $\mathcal{H}_p$-matrices in Subsection 2.3.2. They form a subset of the more general class of $\mathcal{H}$-matrices.



(a) Level 1. *(b)* Level 2. *(c)* Level 3.

Figure 3.1. Vertical Splitting into block columns used during the recursion of the block Householder QR decomposition for $\mathcal{H}$-matrices.

3.2.1 Highest Level

We start with an introductory example to show how a QR decomposition for an $\mathcal{H}$-matrix on the highest level can be computed. The process described here is also shown in Figure 3.2.

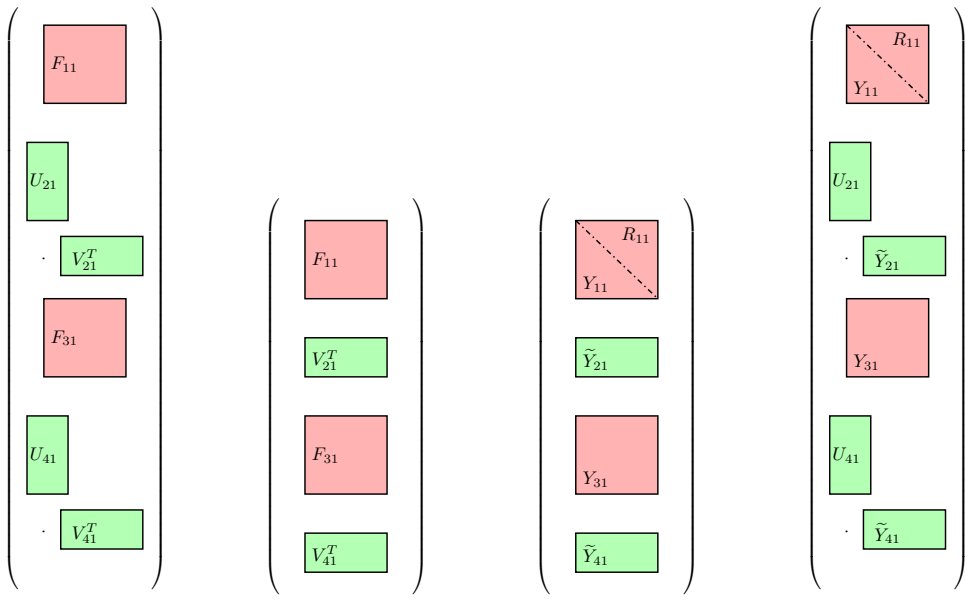
Example 3.1 Let us consider an $\mathcal{H}$-matrix H consisting of dense and low-rank blocks

$$H = \begin{bmatrix} F_{11} \\ R_{21} \\ F_{31} \\ R_{41} \end{bmatrix}, \quad \begin{matrix} F_{11} \in \mathbb{R}^{m_1 \times n}, \ R_{21} \in \mathbb{R}^{m_2 \times n}, \\ F_{31} \in \mathbb{R}^{m_3 \times n}, \ R_{41} \in \mathbb{R}^{m_4 \times n}, \end{matrix}$$

where $m_1 = n$. Hence, H is tall and skinny, i.e. $m_1 + m_2 + m_3 + m_4 > n$. Assume the blocks F_{11} and F_{31} to be given as dense matrices. R_{21}, R_{41} are given in the form of low-rank factorizations

$$R_{21} = U_{21} \cdot V_{21}^T, \quad R_{41} = U_{41} \cdot V_{41}^T$$

with $U_{21} \in \mathbb{R}^{m_2 \times k_2}$, $V_{21} \in \mathbb{R}^{n \times k_2}$, $U_{41} \in \mathbb{R}^{m_4 \times k_4}$ and $V_{41} \in \mathbb{R}^{n \times k_4}$. We define the dense



(a) H. $\rightarrow$ *(b)* H_{cpr}. $\rightarrow$ *(c)* R and Y_{cpr}. $\rightarrow$ *(d)* R and Y.

Figure 3.2. Application of Algorithm 3.7 to the matrix H from Example 3.1.

matrix H_{cpr}, where the subscript *cpr* abbreviates *compressed* as

$$H_{cpr} = \begin{bmatrix} F_{11} \\ V_{21}^T \\ F_{31} \\ V_{41}^T \end{bmatrix} \in \mathbb{R}^{(m_1+k_2+m_3+k_4)\times n}.$$

Let

$$Y_{cpr} = \begin{bmatrix} Y_{11} \\ \widetilde{Y}_{21} \\ Y_{31} \\ \widetilde{Y}_{41} \end{bmatrix} \in \mathbb{R}^{(m_1+k_2+m_3+k_4)\times n}, T \in \mathbb{R}^{n\times n}, R_{cpr} = \begin{bmatrix} R_{11} \\ 0 \\ 0 \\ 0 \end{bmatrix} \in \mathbb{R}^{(m_1+k_2+m_3+k_4)\times n}$$

be the matrices defining the QR decomposition of H_{cpr} as described in Theorem 2.15. Then, assuming U_{21} and U_{41} have orthonormal columns, we can obtain the QR decomposition of H without any additional cost by setting

$$Y = \begin{bmatrix} Y_{11} \\ U_{21} \cdot \widetilde{Y}_{21} \\ Y_{31} \\ U_{41} \cdot \widetilde{Y}_{41} \end{bmatrix} \in \mathbb{R}^{(m_1+m_2+m_3+m_4)\times n}, R = \begin{bmatrix} R_{cpr} \\ 0 \end{bmatrix} \in \mathbb{R}^{(m_1+m_2+m_3+m_4)\times n}$$

and reusing T. This will be verified in the upcoming Theorem 3.1. ♢

Before we show this in the general case, we want to formalize the assumption that from now on, we will always have the first factor in the factorization of low-rank blocks consisting of orthogonal columns.

Assumption B Let $A \in \mathcal{H}(P^{\mathcal{H}}_{I\times J}, k, \mathrm{adm}_A)$ be an $\mathcal{H}$-matrix with $\mathcal{H}$-block partition $P^{\mathcal{H}}_{I\times J}$ based on $\left(P^{\ell}_{I\times J}\right)_{\ell=0}^{L}$. Then for every admissible block $A_{\sigma\times\tau} = UV^T$, the factor U has orthonormal columns.

We will later see in Theorem 3.22 that this is not a restriction from the cost perspective because any given $\mathcal{H}$-matrix can be transformed into one satisfying Assumption B efficiently.

Theorem 3.1 (Block Householder $\mathcal{H}$-QR decomposition - highest level) *Let*

$$A = \begin{pmatrix} A_1 \\ \vdots \\ A_z \end{pmatrix} \in \mathbb{R}^{m\times n}$$

be a hierarchical matrix with a full (inadmissible) square matrix block $A_1 \in \mathbb{R}^{m_1\times n}$, $m_1 = n$, and matrix blocks $A_i \in \mathbb{R}^{m_i\times n}$, $i = 2,\dots,z$ that are either full (inadmissible) or low-rank (admissible) matrix blocks. Let the admissible blocks be represented by $A_i = U_i \cdot V_i^T$, $U_i \in \mathbb{R}^{m_i\times k_i}$, $V_i \in \mathbb{R}^{n\times k_i}$, where U_i has orthonormal columns. We define matrices that consist only of all the inadmissible (full) blocks and the second factors V_i^T of admissible blocks by

$$\widetilde{A}_i := \begin{cases} A_i & A_i \text{ is a full matrix,} \\ V_i^T & A_i \text{ is a low-rank matrix,} \end{cases} \qquad \widetilde{m}_i := \begin{cases} m_i & A_i \text{ is a full matrix,} \\ k_i & A_i \text{ is a low-rank matrix,} \end{cases}$$

and set

$$\widetilde{A} = \begin{pmatrix} \widetilde{A}_1 \\ \vdots \\ \widetilde{A}_z \end{pmatrix} \in \mathbb{R}^{\widetilde{m}\times n} \quad \text{with } \widetilde{m} := \sum_{i=1}^{z} \widetilde{m}_i.$$

Let

$$\widetilde{A} = (I_{\widetilde{m}} - \widetilde{Y}T\widetilde{Y}^T)\begin{pmatrix} R \\ \widetilde{0} \end{pmatrix} \quad \textit{with}$$

$$\widetilde{Y} = \begin{pmatrix} \widetilde{Y}_1 \\ \vdots \\ \widetilde{Y}_z \end{pmatrix} \in \mathbb{R}^{\widetilde{m}\times n},\ T \in \mathbb{R}^{n\times n},\ R \in \mathbb{R}^{n\times n},\ \begin{pmatrix} R \\ \widetilde{0} \end{pmatrix} \in \mathbb{R}^{\widetilde{m}\times n}$$

be a QR decomposition of $\widetilde{A}$, where $\widetilde{Y}$ inherits the block structure of $\widetilde{A}$. Then the QR factorization of A is given by

$$A = (I_m - YTY^T)\cdot\begin{pmatrix} R \\ 0 \end{pmatrix} = Q \cdot \begin{pmatrix} R \\ 0 \end{pmatrix}, \begin{pmatrix} R \\ 0 \end{pmatrix} \in \mathbb{R}^{m\times n},$$

where

$$Q := I_m - YTY^T, \quad Y = \begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_z \end{pmatrix}, \quad Y_i = \begin{cases} \widetilde{Y}_i & A_i \textit{ is a full matrix,} \\ U_i \cdot \widetilde{Y}_i & A_i \textit{ is a low-rank matrix.} \end{cases}$$

Proof We define $Q_* \in \mathbb{R}^{m\times\widetilde{m}}$ as a block diagonal matrix with (possibly non-square) blocks $Q_{i,*} \in \mathbb{R}^{m_i\times\widetilde{m}_i}$ for $1 \le i \le z$ along the diagonal given by

$$Q_{i,*} = \begin{cases} I_{m_i} & A_i \text{ is a full matrix,} \\ U_i & A_i \text{ is a low-rank matrix.} \end{cases}.$$

Since U_i has orthonormal columns, it holds that $Q_*^T Q_* = I_{\widetilde{m}}$. Furthermore, we have

$$Q_* \cdot \widetilde{A} = A, \quad Q_* \cdot \widetilde{Y} = Y$$

since for all $i = 1, \ldots, z$ we have

$$Q_{i,*}\widetilde{A}_i = \begin{cases} I_{m_i}A_i & A_i \text{ is a full matrix,} \\ U_i V_i^T & A_i \text{ is a low-rank matrix} \end{cases} = A_i,$$

$$Q_{i,*}\widetilde{Y}_i = \begin{cases} I_{m_i}Y_i & A_i \text{ is a full matrix,} \\ U_i\widetilde{Y}_i & A_i \text{ is a low-rank matrix} \end{cases} = Y_i.$$

Then it follows that

$$\begin{aligned} Q^T \cdot A &= (I - YT^TY^T) \cdot Q_*\widetilde{A} = Q_*\widetilde{A} - YT^TY^TQ_*\widetilde{A} \\ &\overset{Q_*\widetilde{Y}=Y}{=} Q_*\widetilde{A} - Q_*\widetilde{Y}T^T\widetilde{Y}^T \underbrace{Q_*^TQ_*}_{=I_{\widetilde{m}}}\widetilde{A} \\ &= Q_*\left(I - \widetilde{Y}T^T\widetilde{Y}^T\right)\widetilde{A} = Q_*\begin{pmatrix} R \\ \widetilde{0} \end{pmatrix} = \begin{pmatrix} R \\ 0 \end{pmatrix}. \end{aligned}$$

The last equation follows from $R \in \mathbb{R}^{n\times n}$ and the fact that $Q_{1,*} = I_n$ since A_1 had been assumed to be a full $n \times n$ matrix. ■

Theorem 3.1 is relevant for the start of the recursion in Theorem 2.15 and requires the leading block A_1 to be inadmissible and square. While this is often the case when the recursive QR algorithm is applied to square $\mathcal{H}$-matrices, this may no longer be the case for more general $\mathcal{H}$-block partitions. However, it is sufficient if any of the blocks A_i is inadmissible and square since we can compute the QR factorization for the permuted matrix PA that has an inadmissible square leading block. We obtain

$$PA = (I - YTY^T)R \quad \Longleftrightarrow \quad A = \left(I - P^TYT(P^TY)^T\right)P^TR,$$

i. e., A is factored into an orthogonal matrix $I - P^TYT(P^TY)^T$ with (row-) permuted lower trapezoidal matrix P^TY and a (row-)permuted upper trapezoidal matrix P^TR. Figure 3.3 shows an example for a rectangular $\mathcal{H}$-matrix A. In each block column τ_i'' corresponding to a column leaf, an inadmissible rectangular block is subdivided into two blocks, the first being square and used as leading block A_1 in Theorem 3.1. The resulting $\mathcal{H}$-matrices Y and R are row-permuted lower/upper trapezoidal matrices, respectively. In the subsequent algorithms, we abuse notation and use P to denote both the permutation on the index set as well as the corresponding (row) permutation matrix.

If A has no square subblock in $P^{\mathcal{H}}_{I\times\tau}$ but a rectangular inadmissible block with more rows than columns, we can subdivide this block into a square one and its remainder. If A has only admissible blocks, we can convert an admissible block (or even several) into an inadmissible one with equal or more rows than columns and then proceed as described above for inadmissible blocks.

Theorem 3.1 corresponds to the start of the recursion and is formalized in Algorithm 3.7. Note that at the start of the recursion, there is no difference between T and its reduced representation $\overline{T}$ (see Remark 2.16).

Algorithm 3.7: Householder-based $\mathcal{H}$-QR decomposition - the recursion start.

Data: $A \in \mathcal{H}(P^{\mathcal{H},A}_{I\times J}, k, \mathrm{adm}_A)$ with $\mathrm{depth}_{col}(P^{\mathcal{H},A}_{I\times J}) = 0$
Result: $Y \in \mathcal{H}(P^{\mathcal{H},Y}_{I\times J}, k_Y, \mathrm{adm}_Y)$, $T \in \mathcal{H}(P^{\mathcal{H},T}_{J\times J}, k_T, \mathrm{adm}_T)$, $R \in \mathcal{H}(P^{\mathcal{H},R}_{I\times J}, k_R, \mathrm{adm}_R)$ and permutation matrix $P \in \mathbb{R}^{I\times I}$ defining a QR decomposition of A given by $A = (I - YTY^T)R$ with row-permuted trapezoidal matrices Y, R and upper triangular T (PY, PR are trape- zoidal as well)
Function: $[Y, T, R, P] = \mathcal{H}\mathrm{QR}_{\mathrm{start}}(A)$

Create the dense matrix $\widetilde{A} \in \mathbb{R}^{\widetilde{m}\times \#J}$ consisting of blocks $\widetilde{A}_i$ as in Theorem 3.1;
for *all subblocks* A_i *of* A **do**
  if A_i *is a low-rank matrix given by* $A_i = U_i \cdot V_i^T$ *with* $U_i^T U_i = I_{k_i}$ **then**
    Set $\widetilde{A}_i = V_i^T$;
  else
    Set $\widetilde{A}_i = A_i$;
  end
end
Let σ denote the row index set of an inadmissible square subblock A_i of A;
Reorder the blocks of $\widetilde{A}$ such that the corresponding $\widetilde{A_i}$ comes first;
Let P denote the permutation (matrix) satisfying $P(\sigma) = \{1, \ldots, \#J\}$;
Compute the dense QR decomposition $\widetilde{A} = \widetilde{Q} \cdot \begin{pmatrix} \widetilde{R} \\ \widetilde{0} \end{pmatrix}$ with $\widetilde{Q} = I_{\widetilde{m},\widetilde{m}} - \widetilde{Y}\widetilde{T}\widetilde{Y}^T \in \mathbb{R}^{\widetilde{m}\times\widetilde{m}}$, $\widetilde{R} \in \mathbb{R}^{J\times J}$, $\widetilde{Y} \in \mathbb{R}^{\widetilde{m}\times J}$ and $\widetilde{T} \in \mathbb{R}^{J\times J}$;
Reverse the reordering of $\widetilde{A}$ in $\widetilde{Y}$: $\widetilde{Y} \leftarrow P^{-1}\widetilde{Y}$;
for *all subblocks* A_i *of* A **do**
  if A_i *is a low-rank matrix given by* $A_i = U_i \cdot V_i^T$ **then**
    Set $Y_i = U_i \cdot \widetilde{Y}_i$;
  else
    Set $Y_i = \widetilde{Y}_i$;
  end
end
Set $P^{\mathcal{H},T}_{J\times J} = \{J \times J\}$ and $T = \widetilde{T}$;
Set $P^{\mathcal{H},R}_{I\times J} = \{\sigma \times J, (I \setminus \sigma) \times J\}$, $R_{\sigma,J} = \widetilde{R}$ and $R_{(I\setminus\sigma),J}$ as admissible (with rank 0);
return Y, T, R, P;

3.2.2 Lower Levels

We will use the recursion shown in Theorem 2.15 to compute a QR decomposition. This method applied to $\mathcal{H}$-matrices is summarised in the Algorithms 3.8, 3.9 and 3.10. The recursion for non-leaf block columns of A according to Theorem 2.15 is implemented in Algorithm 3.8 (main recursive QR algorithm) with the auxiliary Algorithm 3.9 for the update $\widetilde{A}_2 = Q_1^T A_2$ and Algorithm 3.10 to merge the results obtained from the recursive calls for the two column block submatrices of A. All operations are done using $\mathcal{H}$-matrix arithmetics and the induced admissibility function given by Definitions 2.24 and 2.25. Furthermore, we also use the more efficient approach of saving $\overline{T}$ instead of T as explained in Remark 2.16.

Two main factors determine whether this is a helpful approach. First, using the induced admissibility function could lead to so many inadmissible blocks in Y, R or T that we lose any computational advantage. Second, we have to vertically split admissible blocks given by a low-rank factorization. It is initially unclear what the structure of the corresponding factors in Y is.

We will show that such splits in A can later be reversed in Y. For that, let us review an example.

Example 3.2 Let the matrix H be given by

$$H = \begin{bmatrix} F_{11} & R_{12} \\ R_{21} & F_{22} \\ F_{31} & R_{32} \\ & R_4 \end{bmatrix} \in \mathbb{R}^{(m_1+m_2+m_3+m_4)\times(n_1+n_2)}, \quad \begin{array}{l} F_{11} \in \mathbb{R}^{m_1\times n_1},\ R_{12} \in \mathbb{R}^{m_1\times n_2}, \\ F_{21} \in \mathbb{R}^{m_2\times n_1},\ F_{22} \in \mathbb{R}^{m_2\times n_2}, \\ F_{31} \in \mathbb{R}^{m_3\times n_1},\ R_{32} \in \mathbb{R}^{m_3\times n_2}, \\ R_4 \in \mathbb{R}^{m_4\times(n_1+n_2)} \end{array}$$

where R_{12}, R_{21}, R_{32}, and R_4 allow for a low-rank factorization with ranks k_{12}, k_{21}, k_{32}, and k_4. More precisely, $R_4 = U_4 \cdot V_4^T$ with $U_4 \in \mathbb{R}^{m_4\times k_4}$ and $V_4 \in \mathbb{R}^{(n_1+n_2)\times k_4}$. Since

$$R_4 = U_4 \cdot V_4^T = U_4 \cdot \begin{bmatrix} V_{41}^T & V_{42}^T \end{bmatrix} = \begin{bmatrix} U_4 \cdot V_{41}^T & U_4 \cdot V_{42}^T \end{bmatrix}$$

with $V_{41} \in \mathbb{R}^{n_1\times k_4}$ and $V_{42} \in \mathbb{R}^{n_2\times k_4}$, we can rewrite H as

$$H = \begin{bmatrix} F_{11} & R_{12} \\ R_{21} & F_{22} \\ F_{31} & R_{32} \\ R_{41} & R_{42} \end{bmatrix},$$

where $R_{41} = U_4 \cdot V_{41}^T$ and $R_{42} = U_4 \cdot V_{42}^T$.

By Theorem 3.1, there is a QR decomposition of the first block column of H so that for

Algorithm 3.8: Recursive Householder-based QR decomposition for $\mathcal{H}$-matrices.

Data: $A \in \mathcal{H}(P_{I\times J}^{\mathcal{H},A}, k_A, \mathrm{adm}_A)$

Result: $Y \in \mathcal{H}(P_{I\times J}^{\mathcal{H},Y}, k_Y, \mathrm{adm}_Y)$, $\overline{T} \in \mathcal{H}(P_{J\times J}^{\mathcal{H},\overline{T}}, k_{\overline{T}}, \mathrm{adm}_{\overline{T}})$, $R \in \mathcal{H}(P_{I\times J}^{\mathcal{H},R}, k_R, \mathrm{adm}_R)$ defining a QR decomposition of A by $A = (I - YTY^T)\cdot R$ where $\overline{T}$ is the reduced form of T (see Remark 2.16). $P \in \mathbb{R}^{I\times I}$ is a permutation (matrix) such that PY, PR are trapezoidal

Function: $[Y, \overline{T}, R, P] = \mathcal{H}\mathrm{QR}(A)$

1 **if** *J is a column leaf cluster of $P_{I\times J}^{\mathcal{H},A}$* **then**
2 $\quad [Y, \overline{T}, R, P] = \mathcal{H}\mathrm{QR}_{\mathrm{start}}(A)$; `// See Algorithm 3.7.`
3 **else**
4 $\quad$ Find non-trivial direct descendants $\tau_1, \tau_2 \in P_J^{\mathrm{hier}}$ with $\tau_1 \cup \tau_2 = J$ and $\tau_1 \cap \tau_2 = \emptyset$;
5 $\quad [Y_{I,\tau_1}, \overline{T}_{\tau_1,\tau_1}, R_{I,\tau_1}, P_{I,I}^{\tau_1}] = \mathcal{H}QR(A_{I,\tau_1})$;
6 $\quad A_{I,\tau_2} = \mathcal{H}\mathrm{QR}_{\mathrm{update}}(A_{I,\tau_2}, Y_{I,\tau_1}, \overline{T}_{\tau_1,\tau_1})$; `// see Algorithm 3.9`
7 $\quad$ Set $I_1 = \left(P_{I,I}^{\tau_1}\right)^{-1}(\{1, \ldots, \#\tau_1\})$ and $I_2 = I \setminus I_1$;
8 $\quad [Y_{I_2,\tau_2}, \overline{T}_{\tau_2,\tau_2}, R_{I_2,\tau_2}, P_{I_2,I_2}^{\tau_2}] = \mathcal{H}\mathrm{QR}(A_{I_2,\tau_2})$;
9 $\quad P = P_{I_2,I_2}^{\tau_2} \circ P_{I,I}^{\tau_1}$ (with $P_{I_2,I_2}^{\tau_2}$ extended to I by the identity on τ);
10 $\quad [Y, \overline{T}, R, P] = \mathcal{H}\mathrm{QR}_{cb}(Y_{I,\tau_1}, R_{I,\tau_1}, \overline{T}_{\tau_1,\tau_1},\ Y_{I_2,\tau_1}, R_{I_2,\tau_2}, \overline{T}_{\tau_2,\tau_2},\ A_{I_1,\tau_2})$; `// See Algorithm 3.10.`
11 **end**
12 **return** $Y, \overline{T}, R, P$;

the block Y_{41} corresponding to R_{41}

$$Y_{41} = U_4 \cdot V_{41,Y}^T$$

holds true with some $V_{41,Y} \in \mathbb{R}^{n_1\times k_4}$. After the update by Algorithm 3.9, the block $\widetilde{R}_{42} \in \mathbb{R}^{m_4\times n_4}$ in the second block column corresponding to R_{42} is given by

$$\widetilde{R}_{42} = R_{42} - Y_{41}\cdot S_2 = U_4\cdot V_{42}^T - U_4\cdot V_{41,Y}^T\cdot S_2 = U_4\cdot(V_{42}^T - V_{41,Y}^T\cdot S_2) = U_4\cdot\widetilde{V}_{42},$$

where $S_2 \in \mathbb{R}^{n_1\times n_2}$ has been defined in line 2 of Algorithm 3.9 and $\widetilde{V}_{42} \in \mathbb{R}^{n_2\times k_4}$ by $\widetilde{V}_{42} = V_{42}^T - V_{41,Y}^T\cdot S_2$. Again with Theorem 3.1, the corresponding factor $Y_{42} \in \mathbb{R}^{m_4\times n_2}$ in Y is given by

$$Y_{42} = U_4\cdot V_{42,Y}^T$$

with $V_{42,Y} \in \mathbb{R}^{n_2\times k_4}$. We can thus reverse the split of R_4 and get the corresponding

Algorithm 3.9: Update algorithm computing $\widetilde{A}_2 = Q_1^T A_2$.

Data: $\mathcal{H}$-matrix $A_{I,\tau_2} \in \mathcal{H}(P^{\mathcal{H},A}_{I\times\tau_2}, k_A, \mathrm{adm}_A)$, orthogonal factor $Q = YTY^T$ of QR decomposition of $A_{I,\tau_1} \in \mathcal{H}(P^{\mathcal{H},A}_{I\times\tau_1}, k_A, \mathrm{adm}_A)$ given by $Y_{I,\tau_1} \in \mathcal{H}(P^{\mathcal{H},Y}_{I\times\tau_1}, k_Y, \mathrm{adm}_Y)$ and $\overline{T}_{\tau_1,\tau_1} \in \mathcal{H}(P^{\mathcal{H},\overline{T}}_{\tau_1\times\tau_1}, k_{\overline{T}}, \mathrm{adm}_{\overline{T}})$

Result: Updated $\mathcal{H}$-matrix $\hat{A}_{I,\tau_2} = (I - Y_{I,\tau_1} T^T_{\tau_1,\tau_1} Y^T_{I,\tau_1}) A_{I,\tau_2} \in \mathcal{H}(P^{\mathcal{H},\hat{A}}_{I\times\tau_2}, k_{\hat{A}}, \mathrm{adm}_{\hat{A}})$

Function: $\hat{A}_{I\times\tau_2} = \mathcal{H}\mathrm{QR}_{\mathrm{update}}(A_{I,\tau_2}, Y_{I,\tau_1}, T_{\tau_1,\tau_1})$

1 Compute $S_1 = Y^T_{I,\tau_1} \cdot A_{I,\tau_2} \in \mathcal{H}(P^{\mathcal{H},S_1}_{\tau_1\times\tau_2}, k_{S_1}, \mathrm{adm}_{S_1})$ where adm_{S_1} is induced w. r. t. adm_Y, adm_A and $P^{\mathcal{H},S_1}_{\tau_1\times\tau_2}$ consistent with adm_{S_1};

2 Compute $S_2 = T^T_{\tau_1,\tau_1} \cdot S_1 \in \mathcal{H}(P^{\mathcal{H},S_2}_{\tau_1\times\tau_2}, k_{S_2}, \mathrm{adm}_{S_2})$ where adm_{S_2} is induced w. r. t. adm_T, adm_{S_1} and $P^{\mathcal{H},S_2}_{\tau_1\times\tau_2}$ consistent with adm_{S_2} using the reduced representation $\overline{T}_{\tau_1,\tau_1}$ of T_{τ_1,τ_1};

3 Compute $\hat{A}_{I\times\tau_2} = A_{I\times\tau_2} - Y_{I\times\tau_1} \cdot S_2 \in \mathcal{H}(P^{\mathcal{H},\hat{A}}_{I\times\tau_2}, k_{\hat{A}}, \mathrm{adm}_{\hat{A}})$ where $\mathrm{adm}_{\hat{A}}$ is induced w. r. t. adm_Y, adm_{S_2}, adm_A and $P^{\mathcal{H},\hat{A}}_{I\times\tau_2}$ consistent with $\mathrm{adm}_{\hat{A}}$;

4 **return** $\hat{A}_{I\times\tau_2}$;

block $Y_4 \in \mathbb{R}^{m_4\times(n_1+n_2)}$ in Y as

$$Y_4 = \begin{bmatrix} U_4 \cdot V^T_{41,Y} & U_4 \cdot V^T_{42,Y} \end{bmatrix} = U_4 \cdot \begin{bmatrix} V^T_{41,Y} & V^T_{42,Y} \end{bmatrix}.$$

The entire Y is then given by

$$Y = \begin{bmatrix} Y_{11} & 0 \\ Y_{21} & Y_{22} \\ Y_{31} & Y_{32} \\ & Y_4 \end{bmatrix} \in \mathbb{R}^{(m_1+m_2+m_3+m_4)\times n}, \quad \begin{matrix} Y_{11} \in \mathbb{R}^{m_1\times n_1}, \\ Y_{21} \in \mathbb{R}^{m_2\times n_1},\ Y_{22} \in \mathbb{R}^{m_2\times n_2}, \\ Y_{31} \in \mathbb{R}^{m_3\times n_1},\ Y_{32} \in \mathbb{R}^{m_3\times n_2}, \\ Y_4 \in \mathbb{R}^{m_4\times(n_1+n_2)}, \end{matrix}$$

where Y_4 allows for a low rank factorization of rank k_4. ♢

The general case can be shown using the same ideas presented here, but we can also prove it as a direct consequence of Theorem 3.1, which is notably faster. However, the constructive approach from Example 3.2 is the basis for the implementation of this part of the algorithm.

As already mentioned at the end of Subsection 3.2.1, we need to ensure that we will always find an inadmissible (square) block in every $\mathcal{H}$-matrix block column on the highest level. Henceforth, we make the following assumption.

Algorithm 3.10: Combine results of the recursive calls for the column sub-blocks.

Data: QR decomposition of A_{I,τ_1} given by Y_{I,τ_1}, R_{I,τ_1}, $\overline{T}_{\tau_1,\tau_1}$, QR decomposition of (updated) A_{I_2,τ_2} given by Y_{I_2,τ_2}, R_{I_2,τ_2} and $\overline{T}_{\tau_2,\tau_2}$ and (updated) matrix block A_{I_1,τ_2} (including the index sets I_1, I_2 such that permutation matrices do not have to be passed)

Result: QR decomposition of A given by Y, R and $\overline{T}$ where P is a permutation matrix such that PY, PR are trapezoidal

Function: $[Y, T, R, P] = \mathcal{H}\mathrm{QR}_{cb}(Y_{I,\tau_1}, R_{I,\tau_1}, \overline{T}_{\tau_1,\tau_1}, Y_{I_2,\tau_1}, R_{I_2,\tau_1}, \overline{T}_{\tau_2,\tau_2}, A_{I_1,\tau_2})$

1 Set $P^{\mathcal{H},Y}_{I\times J} = P^{\mathcal{H},Y}_{I\times\tau_1} \cup P^{\mathcal{H},Y}_{I_2\times\tau_2} \cup \{I_1 \times \tau_2\}$ and combine Y_{I,τ_1} and $Y_{I_2\times\tau_2}$ to an $\mathcal{H}$-matrix $Y = \begin{pmatrix} Y_{I,\tau_1} & 0 \\ & Y_{I_2,\tau_2} \end{pmatrix} \in \mathcal{H}(P^{\mathcal{H},Y}_{I\times J}, k_Y, \mathrm{adm}_Y)$ with admissible (rank 0) block Y_{I_1,τ_2};

2 Apply Theorem 3.2 to blocks in Y corresponding to admissible blocks that have been split during the recursion, i. e. merge them into a single admissible block and adjust $P^{\mathcal{H},Y}_{I\times J}$ accordingly;

3 Set $P^{\mathcal{H},R}_{I\times J} = P^{\mathcal{H},R}_{I\times\tau_1} \cup P^{\mathcal{H},R}_{I_2\times\tau_2} \cup P^{\mathcal{H},A}_{I_1\times\tau_2}$ and combine R_{I,τ_1}, $R_{I_2\times\tau_2}$ and $A_{I_1\times\tau_2}$ into $R = \begin{pmatrix} R_{I,\tau_1} & A_{I_1\times\tau_2} \\ & R_{I_2,\tau_2} \end{pmatrix} \in \mathcal{H}(P^{\mathcal{H},R}_{I\times J}, k_R, \mathrm{adm}_R)$;

4 Compute $\overline{T}_{\tau_1,\tau_2} = -Y^T_{I,\tau_1} Y_{I,\tau_2} \in \mathcal{H}(P^{\mathcal{H},\overline{T}}_{\tau_1\times\tau_2}, k_{\overline{T}}, \mathrm{adm}_{\overline{T}})$ where $\mathrm{adm}_{\overline{T}}$ is induced w. r. t. $\mathrm{adm}_{Y_{I,\tau_1}}, \mathrm{adm}_{Y_{I,\tau_2}}$ and $P^{\mathcal{H},\overline{T}}_{\tau_1\times\tau_2}$ consistent with $\mathrm{adm}_{\overline{T}}$;

5 Set $P^{\mathcal{H},\overline{T}}_{J\times J} = P^{\mathcal{H},\overline{T}}_{\tau_1\times\tau_1} \cup P^{\mathcal{H},\overline{T}}_{\tau_2\times\tau_2} \cup P^{\mathcal{H},\overline{T}}_{\tau_1\times\tau_2} \cup \{\tau_2 \times \tau_1\}$ and combine $\overline{T}_{\tau_1,\tau_1}$, $T_{\tau_2\times\tau_2}$ and $\overline{T}_{\tau_1\times\tau_2}$ into $\overline{T} = \begin{pmatrix} \overline{T}_{\tau_1,\tau_1} & \overline{T}_{\tau_1,\tau_2} \\ 0 & \overline{T}_{\tau_2,\tau_2} \end{pmatrix} \in \mathcal{H}(P^{\mathcal{H},\overline{T}}_{J\times J}, k_T, \mathrm{adm}_T)$ with admissible (rank 0) block $\overline{T}_{\tau_2,\tau_1}$;

6 **return** Y, $\overline{T}$, R;

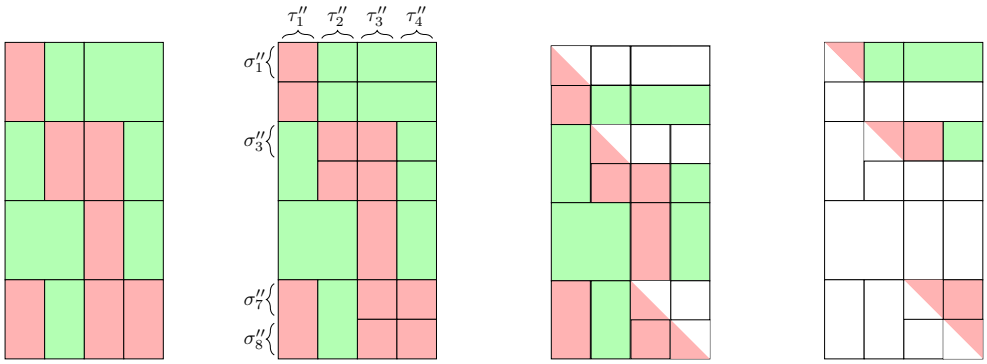
Assumption C Let $A \in \mathcal{H}(P^{\mathcal{H},A}_{I\times J}, k_A, \mathrm{adm}_A)$ be an $\mathcal{H}$-matrix. For every column index cluster $\tau \in P^{L_J}_J$ on the highest level L_J, there exists $\sigma_\tau \in P^{hier}_I$ so that the following conditions hold:

1. $\mathrm{adm}_A(\sigma_\tau \times \tau) = \mathrm{False}$,
2. $\sigma_\tau \cap \sigma_{\tau'} = \emptyset \ \forall\ \tau' \in P^{L_J}_J$ with $\tau \neq \tau'$,
3. $\#\sigma_\tau = \#\tau$,
4. $\sigma_\tau \times \tau \in P^{\mathcal{H},-}_{I\times J}$.

We define $P^{split}_{I\times J} \subset P^{L_{IJ}}_{I\times J}$ as the set that contains all blocks $\sigma_\tau \times \tau$, meaning we have

$$P^{split}_{I\times J} = \{\sigma_\tau \times \sigma \ : \ \tau \in P^{L_J}_J\}.$$

Additionally, for every block $\sigma \times \tau \in P^{hier}_{I\times J}$ we have $\#\sigma \geq \#\tau$.



(a) Original A. *(b)* Split A. *(c)* Y. *(d)* R.

Figure 3.3. To compute the QR decomposition of A (a), we need to change the $\mathcal{H}$-block partition to create the necessary square inadmissible blocks in every column. One possibility is shown in (b). The resulting factors Y and R of the permuted QR decomposition can be seen in (c) and (d). The diagonal blocks are given by $\sigma''_1 \times \tau''_1$, $\sigma''_3 \times \tau''_2$, $\sigma''_7 \times \tau''_3$ and $\sigma''_8 \times \tau''_4$.

The assumption that our topmost block after permutation is a square matrix does not restrict us to only a special type of rectangular matrices. For any matrix with rectangular blocks, we can add one level to the hierarchy of partitionings so that every rectangular block $\sigma \times \tau \in P^{hier}_{I\times J}$ with $\sigma \in P^{hier}_I$ and $\tau \in P^{hier}_J$ has two descendants given by $\sigma' \times \tau$ and $\sigma'' \times \tau$ so that $\#\sigma' = \#\tau$ and $\sigma' \cup \sigma'' = \sigma$ hold and change the $\mathcal{H}$-block partition accordingly. Note that although the hierarchy of block partitionings gains an entirely new level, in the $\mathcal{H}$-block partition the rectangular blocks on the formerly lowest level get replaced by their two new descendants only where it is necessary to apply Algorithm 3.7.

Now we can finally generalize Example 3.2.

Theorem 3.2 (Block Householder $\mathcal{H}$-QR decomposition - low-rank splits) *Let $A \in \mathcal{H}(P^{\mathcal{H},A}_{I\times J}, k_A, adm_A)$ be an $\mathcal{H}$-matrix with $\mathcal{H}$-block partition $P^{\mathcal{H},A}_{I\times J}$ based on the hierarchy of partitionings $\left(P^\ell_{I\times J}\right)^L_{\ell=0}$. Let $\sigma \times J \in P^{\mathcal{H},A}_{I\times J}$, and let*

$$Y \in \mathcal{H}(P^{\mathcal{H},Y}_{I\times J}, k_Y, adm_Y), \quad \overline{T} \in \mathcal{H}(P^{\mathcal{H},\overline{T}}_{I\times I}, k_{\overline{T}}, adm_{\overline{T}}), \quad R \in \mathcal{H}(P^{\mathcal{H}}_{I\times J}, k_R, adm_R)$$

form a QR decomposition of A computed by Algorithm 3.8 using Algorithms 3.9 and 3.10. Let $A_{\sigma,J}$ denote the block in A and $Y_{\sigma,J}$ denote the block in Y that are both induced by $\sigma \times J$. Let

$$A_{\sigma,J} = U_\sigma \cdot V_J^T, \quad U_\sigma \in \mathbb{R}^{\#\sigma \times k_A}, V_J \in \mathbb{R}^{\#J \times k_A}$$

be a low-rank factorization of $A_{\sigma,J}$. Then we can find a low-rank factorization of $Y_{\sigma,J}$ given by

$$Y_{\sigma,J} = U_\sigma \cdot V_{J,Y}^T, \quad V_{J,Y} \in \mathbb{R}^{\#J \times k_A}.$$

Proof Let A be an $\mathcal{H}$-matrix as described in the statement. If the hierarchy of partitionings of the column index set has depth 0, then the statement is true with Theorem 3.1. Now assume that the statement is true for all $\mathcal{H}$-matrices where the depth of the hierarchy of partitionings of the column index set is ℓ' and $0 \leq \ell' < \ell$. We show that it is also true for $\mathcal{H}$-matrices where the depth of the hierarchy of block partitionings of the column index set is ℓ. Let A be such an $\mathcal{H}$-matrix and A_{I,τ_1} and A_{I,τ_2} be the two block columns used during the algorithm induced by τ_1, $\tau_2 \in P_J^{hier}$ with $\tau_1 \cup \tau_2 = J$, $\tau_1 \cap \tau_2 = \emptyset$ and

$$A_{I,J} = [A_{I,\tau_1} \; A_{I,\tau_2}].$$

Let $\sigma \in P_I^{hier}$ so that $A_{\sigma,J} = U_\sigma \cdot V_J^T$ with $U_\sigma \in \mathbb{R}^{\#\sigma \times k_A}$ and $V_J \in \mathbb{R}^{\#J \times k_A}$ is an admissible block of $A_{I,J}$. Then the corresponding blocks in A_{I,τ_1} and A_{I,τ_2} are given by

$$\begin{aligned} A_{\sigma,\tau_1} &= U_\sigma \cdot V_{\tau_1}^T, \quad V_{\tau_1} \in \mathbb{R}^{\#\tau_1 \times k_A}, \\ A_{\sigma,\tau_2} &= U_\sigma \cdot V_{\tau_2}^T, \quad V_{\tau_2} \in \mathbb{R}^{\#\tau_2 \times k_A}, \end{aligned}$$

where V_{τ_1} consists of the first $\#\tau_1$ rows of V_J and V_{τ_2} of the remaining $(\#\tau_2)$ rows of V_J. By induction, the block in Y corresponding to A_{σ,τ_1} is then given by

$$Y_{\sigma,\tau_1} = U_\sigma \cdot V_{\tau_1,Y}^T, \quad V_{\tau_1,Y} \in \mathbb{R}^{\#\tau_1 \times k_A}.$$

Let $S_2 \in \mathbb{R}^{\tau_1 \times \tau_2}$ be defined as in Algorithm 3.9, and let $\hat{A}_{\sigma,\tau_2}$ denote the block A_{σ,τ_2} after applying Algorithm 3.9. Then $\hat{A}_{\sigma,\tau_2}$ can be computed by

$$\begin{aligned} \hat{A}_{\sigma,\tau_2} &= A_{\sigma,\tau_2} - Y_{\sigma,\tau_1} \cdot S_2 \\ &= U_\sigma \cdot V_{\tau_2}^T - U_\sigma \cdot V_{\tau_1,Y}^T \cdot S_2 \\ &= U_\sigma \cdot \left(V_{\tau_2}^T - V_{\tau_1,Y}^T \cdot S_2 \right), \end{aligned}$$

and thus we have a low-rank factorization $\hat{A}_{\sigma,\tau_2} = U_\sigma \cdot \hat{V}_{\tau_2}^T$ with

$$\hat{V}_{\tau_2} := V_{\tau_2} - S_2^T \cdot V_{\tau_1,Y}.$$

We apply the induction again, and the block in Y corresponding to $\hat{A}_{\sigma,\tau_2}$ is then given

by

$$Y_{\sigma,\tau_2} = U_\sigma \cdot V_{\tau_2,Y}^T, \quad V_{\tau_2,Y} \in \mathbb{R}^{\#\tau_2 \times k_A}.$$

Thus, the block in Y corresponding to $A_{\sigma,J}$ is given by

$$\begin{aligned} Y_{\sigma,J} &= [Y_{\sigma,\tau_1}\ Y_{\sigma,\tau_2}] \\ &= \left[U_\sigma \cdot V_{\tau_1,Y}^T\ U_\sigma \cdot V_{\tau_2,Y}^T\right] \\ &= U_\sigma \cdot [V_{\tau_1,Y}\ V_{\tau_2,Y}]^T, \end{aligned}$$

and we have found a low-rank factorization for $Y_{\sigma,J}$. This proves our statement. ∎

3.2.3 Update Matrices

Theorem 3.2 unfortunately does not imply that the matrix Y in the representation $Q = I - YTY^T$ of the orthogonal factor Q inherits the $\mathcal{H}$-block structure of A. All but the first leaf block columns of A are updated in the course of the recursion described in Theorem 2.15 (by the multiplication $\widetilde{A}_2 = Q_1^T A_2$). Low rank blocks that contain part of this first leaf column (as R_4 in Example 3.2) are preserved as low rank blocks, all other blocks may (and typically do) change their $\mathcal{H}$-block structures.

To address this issue and maintain reasonable bounds on computational and storage costs, we must estimate the change in admissibility that occurs during the algorithm. This section focuses on the (many) intermediate results S_1 and S_2 computed in Algorithm 3.9. As shown there, the updated $\mathcal{H}$-matrix $\hat{A}_{I,\tau_2}$ is given by

$$\hat{A}_{I,\tau_2} = Q_1^T A_{I,\tau_2} = \left(I - Y_{I,\tau_1} T_{\tau_1,\tau_1}^T Y_{I,\tau_1}^T\right) A_{I,\tau_2} = A_{I,\tau_2} - Y_{I,\tau_1} \overbrace{T_{\tau_1,\tau_1}^T \underbrace{Y_{I,\tau_1}^T A_{I,\tau_2}}_{=S_1}}^{=S_2}.$$

Our goal is to construct larger $\mathcal{H}$-matrices that encompass all these intermediate results S_1 and S_2 and rephrase certain parts of the algorithm in a theoretical (albeit impractical) manner that improves the understanding of the resulting structures.

Before doing so, we must first prove one preliminary statement.

Theorem 3.3 *Let us have the situation of Theorem 3.2. Let $\tau, \tau' \in P_J^{hier}$ with level(τ) = level(τ') and $\tau < \tau'$ (see Definition 2.26) be given. Then there exists a uniquely defined update $\widehat{A}_{I,\tau_2} = A_{I,\tau_2} - Y_{I,\tau_1} \cdot S_2$ computed in line 3 of Algorithm 3.9 during the computation of the QR decomposition of A with Algorithm 3.8 so that $\tau \subseteq \tau_1$ and $\tau' \subseteq \tau_2$ hold.*

Proof Let $\tau, \tau' \in P_J^{hier}$ with level(τ) = level(τ') and $\tau < \tau'$ be given. Then there is a first common ancestor $\widehat{\tau} \in P_J^{hier}$ of τ and τ' which has two direct and non-trivial

descendants $\tau_1, \tau_2 \in P_J^{hier}$ with $\tau_1 < \tau_2$, $\tau \subseteq \tau_1$ and $\tau' \subseteq \tau_2$. This ancestor is uniquely defined because all different (and thus larger) ancestors of both τ and τ' contain $\widehat{\tau}$. During the recursion, we thus calculate

$$A_{I,\tau_2} - Y_{I,\tau_1} \cdot S_2 \qquad \blacksquare$$

in line 3 of Algorithm 3.9 by construction of the vertical splitting process.

With Theorem 3.3 we have shown that every update matrix S_1 and S_2 is uniquely defined by the column clusters τ_1 and τ_2 that define the columns of the $\mathcal{H}$-matrix A_{I,τ_2} that is to be updated and the $\mathcal{H}$-matrix A_{I,τ_1} that is used during the update. Hence, we can add the superscript (τ_1, τ_2) to identify the exact update in which these intermediate results are used. Formally, we set

$$\begin{aligned} S_1^{(\tau_1,\tau_2)} &= Y_{I,\tau_1}^T A_{I,\tau_2}, \\ S_2^{(\tau_1,\tau_2)} &= T_{\tau_1,\tau_1}^T S_1^{(\tau_1,\tau_2)} = T_{\tau_1,\tau_1}^T Y_{I,\tau_1}^T A_{I,\tau_2}. \end{aligned}$$

Following Algorithm 3.9, the row cluster I used in this definition is not the row cluster of the original matrix but only a subset that depends on the horizontal splitting. However, this subtlety is of no consequence for the definition of $S_1^{(\tau_1,\tau_2)}$ and $S_2^{(\tau_1,\tau_2)}$ because, if we used the row index set of the original matrix A, any row indices that are not part of this smaller I would lead by construction to zero rows in the corresponding Y_{I,τ_1}^T and hence would have no influence on the result.

Furthermore, this result also shows that we can define larger matrices $S_1^{comb} \in \mathbb{R}^{J \times J}$ and $S_2^{comb} \in \mathbb{R}^{J \times J}$ that contain all update matrices S_1 and S_2 and some additional zero blocks. Before we formally define these matrices, we illustrate them in the next example.

Example 3.3 Let

$$H = \begin{bmatrix} H_1 & H_2 & H_3 & H_4 \end{bmatrix} \in \mathbb{R}^{m \times (n_1+n_2+n_3+n_4)}, \quad \begin{matrix} H_1 \in \mathbb{R}^{m \times n_1}, & H_2 \in \mathbb{R}^{m \times n_2}, \\ H_3 \in \mathbb{R}^{m \times n_3}, & H_4 \in \mathbb{R}^{m \times n_4}, \end{matrix}$$

where H_1, H_2, H_3 and H_4 are $\mathcal{H}$-matrix block columns, the matrix Y in the QR decomposition $H = I - YTY^T$ is given by

$$Y = \begin{bmatrix} Y_1 & Y_2 & Y_3 & Y_4 \end{bmatrix} \in \mathbb{R}^{m \times (n_1+n_2+n_3+n_4)}, \quad \begin{matrix} Y_1 \in \mathbb{R}^{m \times n_1}, & Y_2 \in \mathbb{R}^{m \times n_2}, \\ Y_3 \in \mathbb{R}^{m \times n_3}, & Y_4 \in \mathbb{R}^{m \times n_4}, \end{matrix}$$

and the matrix $T \in \mathbb{R}^{n\times n}$ for $n = n_1 + n_2 + n_3 + n_4$ of the QR decomposition is given by

$$T = \left[\begin{array}{cc|cc} T^{(1)} & T^{(1,2)} & \multicolumn{2}{c}{T^{(12,34)}} \\ & T^{(2)} & & \\ \hline & & T^{(3)} & T^{(3,4)} \\ & & & T^{(4)} \end{array}\right], \quad \begin{array}{l} T^{(1)} \in \mathbb{R}^{n_1\times n_1},\ T^{(2)} \in \mathbb{R}^{n_2\times n_2},\ T^{(3)} \in \mathbb{R}^{n_3\times n_3}, \\ T^{(4)} \in \mathbb{R}^{n_4\times n_4},\ T^{(1,2)} \in \mathbb{R}^{n_1\times n_2}, \\ T^{(3,4)} \in \mathbb{R}^{n_3\times n_4},\ T^{(12,34)} \in \mathbb{R}^{(n_1+n_2)\times(n_3+n_4)}. \end{array}$$

For simplification of this example we do not use the reduced form of T as seen in Remark 2.16. Hence, the off-diagonal blocks include the additional multiplication with the appropiate blocks of T and are given by

$$\begin{aligned} T^{(1,2)} &= T^{(1)} Y_1^T Y_2 T^{(2)}, \\ T^{(12,34)} &= \begin{bmatrix} T^{(1)} & T^{(1,2)} \\ & T^{(2)} \end{bmatrix} \begin{bmatrix} Y_1 & Y_2 \end{bmatrix}^T \begin{bmatrix} Y_3 & Y_4 \end{bmatrix} \begin{bmatrix} T^{(3)} & T^{(3,4)} \\ & T^{(4)} \end{bmatrix}, \\ T^{(3,4)} &= T^{(3)} Y_3^T Y_4 T^{(4)}. \end{aligned}$$

We define agglomeration matrices $S_1^{comb}, S_2^{comb} \in \mathbb{R}^{n\times n}$ of the updates for H by

$$S_1^{comb} = \left[\begin{array}{cc|cc} 0 & S_1^{(1,2)} & \multicolumn{2}{c}{S_1^{(12,34)}} \\ & 0 & & \\ \hline & & 0 & S_1^{(3,4)} \\ & & & 0 \end{array}\right], \quad S_2^{comb} = \left[\begin{array}{cc|cc} 0 & S_2^{(1,2)} & \multicolumn{2}{c}{S_2^{(12,34)}} \\ & 0 & & \\ \hline & & 0 & S_2^{(3,4)} \\ & & & 0 \end{array}\right],$$

where

$$S_1^{(1,2)},\ S_2^{(1,2)} \in \mathbb{R}^{n_1\times n_2}$$

refer to the update of H_2 with Y_1,

$$S_1^{(12,34)},\ S_2^{(12,34)} \in \mathbb{R}^{(n_1+n_2)\times(n_3+n_4)}$$

to the update of $[H_3\ H_4]$ with $[Y_1\ Y_2]$ and

$$S_1^{(3,4)},\ S_2^{(3,4)} \in \mathbb{R}^{n_3\times n_4}$$

to the update of $\widehat{H}_4$ with Y_3. $\widehat{H}_4$ denotes the block column in place of H_4 after the update by $[Y_1\ Y_2]$, and $\widehat{\widehat{H}}_4$ is the block column after the subsequent update by Y_3.

So $\widehat{\widehat{H}}_4$ after both updates is given by

$$\widehat{\widehat{H}}_4 = \widehat{H}_4 - Y_3 \cdot S_2^{(3,4)},$$

where $\widehat{H}_4$ is computed as part of

$$[\widehat{H}_3\ \widehat{H}_4] = [H_3\ H_4] - [Y_1\ Y_2] \cdot S_2^{(12,34)}.$$

Combining both, the computation of $\widehat{\widehat{H}}_4$ can be rewritten using the agglomeration matrix S_2^{comb} containing $S_2^{(3,4)}$ and $S_2^{(12,34)}$ to

$$\widehat{\widehat{H}}_4 = H_4 - [Y_1\ Y_2\ Y_3\ 0] \cdot S_2^{4,comb} = H_4 - [Y_1\ Y_2\ Y_3\ Y_4] \cdot S_2^{4,comb},$$

where $S_2^{4,comb} \in \mathbb{R}^{m \times n_4}$ denotes the last n_4 columns of S_2^{comb}. The last equation holds because the bottom right matrix block of $S_2^{4,comb}$ containing the last n_4 rows of S_2^{comb} is zero.

For the other block columns, we define $S_2^{1,comb} \in \mathbb{R}^{m \times n_1}$, $S_2^{2,comb} \in \mathbb{R}^{m \times n_2}$ and $S_2^{3,comb} \in \mathbb{R}^{m \times n_3}$ as the first, second and third block column of S_2^{comb} and receive the analogous results

$$\begin{aligned}
\widehat{H}_3 &= H_3 - [Y_1\ Y_2\ 0\ 0] \cdot S_2^{3,comb} = H_3 - [Y_1\ Y_2\ Y_3\ Y_4] \cdot S_2^{3,comb}, \\
\widehat{H}_2 &= H_2 - [Y_1\ 0\ 0\ 0] \cdot S_2^{2,comb} = H_2 - [Y_1\ Y_2\ Y_3\ Y_4] \cdot S_2^{2,comb}, \\
H_1 &= H_1 - [0\ 0\ 0\ 0] \cdot S_2^{1,comb} = H_1 - [Y_1\ Y_2\ Y_3\ Y_4] \cdot S_2^{1,comb}.
\end{aligned}$$

Let

$$\widehat{H} = [H_1\ \widehat{H}_2\ \widehat{H}_3\ \widehat{\widehat{H}}_4]$$

denote the matrix H after all updates. Then we can write

$$\widehat{H} = H - Y \cdot S_2^{comb}.$$

Additionally, the matrix computation of S_2^{comb} can be expressed using only T and S_1^{comb}. We have

$$\begin{aligned}
S_2^{(1,2)} &= T^{(1),T} \cdot S_1^{(1,2)}, \\
S_2^{(3,4)} &= T^{(3),T} \cdot S_1^{(3,4)}, \\
S_2^{(12,34)} &= \begin{bmatrix} T^{(1),T} & 0 \\ T^{(1,2),T} & T_2^T \end{bmatrix} \cdot S_1^{(12,34)}.
\end{aligned}$$

All these computations are part of the larger product

$$T^T \cdot S_1^{comb} = \left[\begin{array}{cc|cc} T^{(1),T} & & & \\ T^{(1,2),T} & T^{(2)^T} & & \\ \hline & & T^{(3),T} & \\ T^{(12,34),T} & & T^{(3,4),T} & T^{(4),T} \end{array}\right] \cdot \left[\begin{array}{cc|cc} 0 & S_1^{(1,2)} & & S_1^{(12,34)} \\ & 0 & & \\ \hline & & 0 & S_1^{(3,4)} \\ & & & 0 \end{array}\right]$$
$$= \left[\begin{array}{cc|cc} 0 & S_2^{(1,2)} & & S_2^{(12,34)} \\ & 0 & & \\ \hline & & 0 & S_2^{(3,4)} \\ & & & 0 \end{array}\right] + X,$$

where X contains the the remaining blocks, i.e. $X = T^T S_1^{comb} - S_2^{comb}$. Finally, we have

$$\begin{aligned} S_1^{(1,2)} &= Y_1^T \cdot H_2, \\ S_1^{(12,34)} &= [Y_1\ Y_2]^T \cdot [H_3\ H_4], \\ S_1^{(3,4)} &= Y_3^T \cdot \widehat{H}_4. \end{aligned}$$

They are computed as part of the operation

$$Y^T \cdot \widehat{H} = [Y_1\ Y_2\ Y_3\ Y_4]^T \cdot [H_1\ \widehat{H}_2\ \widehat{H}_3\ \widehat{\widehat{H}}_4]$$

because

$$Y_1^T \cdot \widehat{H}_2 = Y_1^T \cdot H_2 - Y_1^T Y_1 \cdot S_2^{(1,2)} = S_1^{(1,2)} - Y_1^T Y_1 \cdot S_2^{(1,2)}$$

includes the computation of $S_1^{(1,2)}$,

$$\begin{aligned} [Y_1\ Y_2]^T \cdot [\widehat{H}_3\ \widehat{\widehat{H}}_4] =& [Y_1\ Y_2]^T \cdot [\widehat{H}_3\ \widehat{H}_4] - [Y_1\ Y_2]^T \cdot [0\ Y_3 \cdot S_2^{(3,4)}] \\ =& [Y_1\ Y_2]^T \cdot [H_3\ H_4] - [Y_1\ Y_2]^T \cdot [Y_1\ Y_2] \cdot S_2^{(12,34)} \\ & - [Y_1\ Y_2]^T \cdot [0\ Y_3 \cdot S_2^{(3,4)}] \\ =& S_1^{(12,34)} - [Y_1\ Y_2]^T \cdot [Y_1\ Y_2] \cdot S_2^{(12,34)} - [Y_1\ Y_2]^T \cdot [0\ Y_3 \cdot S_2^{(3,4)}] \end{aligned}$$

includes the computation of $S_1^{(12,34)}$ and

$$\begin{aligned} Y_3^T \cdot \widehat{\widehat{H}}_4 &= Y_3^T \cdot \widehat{H}_4 - Y_3^T \cdot Y_3 \cdot S_2^{(3,4)} \\ &= S_1^{(3,4)} - Y_3^T \cdot Y_3 \cdot S_2^{(3,4)} \end{aligned}$$

includes the computation of $S_1^{(3,4)}$. Thus, we have

$$Y^T \cdot \widehat{H} = S_1^{comb} + W + Z,$$

with

$$Z = -\left[\begin{array}{c|c} \begin{matrix} 0 & Y_1^T Y_1 \cdot S_2^{(1,2)} \\ & 0 \end{matrix} & [Y_1\ Y_2]^T \cdot \left([Y_1\ Y_2] \cdot S_2^{(12,34)} - [0\ Y_3 \cdot S_2^{(3,4)}]\right) \\ \hline & \begin{matrix} 0 & Y_3^T \cdot Y_3 \cdot S_2^{(3,4)} \\ & 0 \end{matrix} \end{array}\right],$$

$$W = \left[\begin{array}{c|c} \begin{matrix} Y_1^T H_1 & \\ Y_2^T H_1 & Y_2^T \widehat{H}_2 \end{matrix} & \\ \hline [Y_3\ Y_4]^T [H_1\ \widehat{H}_2] & \begin{matrix} Y_3^T \widehat{H}_3 & \\ Y_4^T \widehat{H}_3 & Y_4^T \widehat{\widehat{H}}_4 \end{matrix} \end{array}\right]. \qquad \diamondsuit$$

Note that this result is of theoretical interest only and does not show a way of rewriting the Algorithm 3.8 to operations involving only the matrix as a whole because S_1^{comb} and S_2^{comb} are not available at the start of the algorithm.

We later want to use this representation to derive the (partially) induced admissibility function of the resulting factors of the QR decomposition and estimate the cost of the entire algorithm. Aside from defining the (agglomeration) matrices from Example 3.3 in the general case, in the following definition we also describe the block structure in S_1^{comb} and S_2^{comb} that is induced by the smaller update matrices that are agglomerated in them. The resulting partition called $P_{J\times J}^{upright}$ is illustrated in Figure 3.4 and is the coarsest possible partition of the upper triangular part of an $\mathcal{H}$-matrix. This makes it identical to an $\mathcal{H}$-block partition of the upper triangular part of an $\mathcal{H}^P$-matrix as defined in Definition 2.51.

Definition 3.4 *Let us have an $\mathcal{H}$-block partition $P_{J\times J}^{\mathcal{H}}$ based on a hierarchy of block partitionings $\left(P_{J\times J}^{\ell}\right)_{\ell=0}^{L}$ with $L > 0$, some $\tau \in P_J^{\ell}$ for $\ell > 0$ and $\tau_1, \tau_2 \in P_J^{\ell+1}$ with $\tau_1 \cup \tau_2 = \tau$. Then we call τ_1 the left and τ_2 the right descendant of τ corresponding to some $A \in \mathcal{H}(P_{J\times J}^{\mathcal{H}}, k, adm)$ if $\tau_1 < \tau_2$ and denote it by $\tau_1 = Des_l(\tau)$ and $\tau_2 = Des_r(\tau)$. We define subsets of $P_{J\times J}^{hier}$ by*

$$P_{J\times J}^{0,upright} = \emptyset$$

and then recursively by

$$P_{J\times J}^{\ell,upright} = \Bigg\{ \tau_1 \times \tau_2 \in P_{J\times J}^{\ell} : \exists\ \tau \in P_J^{hier} \text{ with } \tau_1 = Des_l(\tau), \tau_2 = Des_r(\tau) \\ \nexists\ b \in \bigcup_{1\le k<\ell} P_{J\times J}^{k,upright} \text{ with } \tau_1 \times \tau_2 \subseteq b \Bigg\}$$

for $1 \leq \ell \leq L$. *Finally, we set*

$$P_{J\times J}^{upright} = \bigcup_{1\leq \ell \leq L} P_{J\times J}^{\ell,upright},$$

which is again a subset of $P_{J\times J}^{hier}$. *Note that by definition all block clusters contained in it are disjoint.*

Let us have the situation from Theorem 3.2. Let $\tau, \tau' \in P_J^{hier}$ *with* $\tau < \tau'$ *so that* $S_1^{\tau,\tau'}$ *and* $S_2^{\tau,\tau'}$ *denote the update matrices used during the update of the block column* $A_{I,\tau'}$ *by the block column* $Y_{I,\tau}$ *computed in lines 1 and 2 of Algorithm 3.9 and let* $P_{\tau\times\tau'}^{\mathcal{H},S_1^{\tau,\tau'}}$, $P_{\tau\times\tau'}^{\mathcal{H},S_2^{\tau,\tau'}}$ *be their corresponding* $\mathcal{H}$*-block partitions and* $adm_{S_1^{\tau,\tau'}}$, $adm_{S_2^{\tau,\tau'}}$ *their corresponding (consistent) admissibility functions. We define admissibility functions blockwise for* $\tau\times\tau' \in P_{J\times J}^{hier}$ *by*

$$adm_{S_1}(\tau\times\tau') = \begin{cases} adm_{S_1^{\widehat{\tau},\widehat{\tau}'}}(\tau\times\tau') & \exists\ \widehat{\tau}\times\widehat{\tau}' \in P_{J\times J}^{upright} \text{ with } \tau\subseteq\widehat{\tau}, \tau'\subseteq\widehat{\tau}', \\ False & \exists\ \widehat{\tau}\times\widehat{\tau}' \in P_{J\times J}^{upright} \text{ with } \widehat{\tau}\subsetneq\tau \text{ or } \widehat{\tau}'\subsetneq\tau', \\ True & else, \end{cases}$$

$$adm_{S_2}(\tau\times\tau') = \begin{cases} adm_{S_2^{\widehat{\tau},\widehat{\tau}'}}(\tau\times\tau') & \exists\ \widehat{\tau}\times\widehat{\tau}' \in P_{J\times J}^{upright} \text{ with } \tau\subseteq\widehat{\tau}, \tau'\subseteq\widehat{\tau}', \\ False & \exists\ \widehat{\tau}\times\widehat{\tau}' \in P_{J\times J}^{upright} \text{ with } \widehat{\tau}\subsetneq\tau \text{ or } \widehat{\tau}'\subsetneq\tau', \\ True & else. \end{cases}$$

with corresponding consistent $\mathcal{H}$*-block partitions* $P_{J\times J}^{\mathcal{H},S_1}$ *and* $P_{J\times J}^{\mathcal{H},S_2}$, *respectively. Then we define* $S_1^{comb} \in \mathcal{H}(P_{J\times J}^{\mathcal{H},S_1}, k_{S_1}, adm_{S_1})$ *and* $S_2^{comb} \in \mathcal{H}(P_{J\times J}^{\mathcal{H},S_2}, k_{S_2}, adm_{S_2})$ *to be the agglomeration of all corresponding update matrices so that*

$$S_{1,\tau,\tau'}^{comb} = S_1^{\tau,\tau'} \quad and \quad S_{2,\tau,\tau'}^{comb} = S_2^{\tau,\tau'}$$

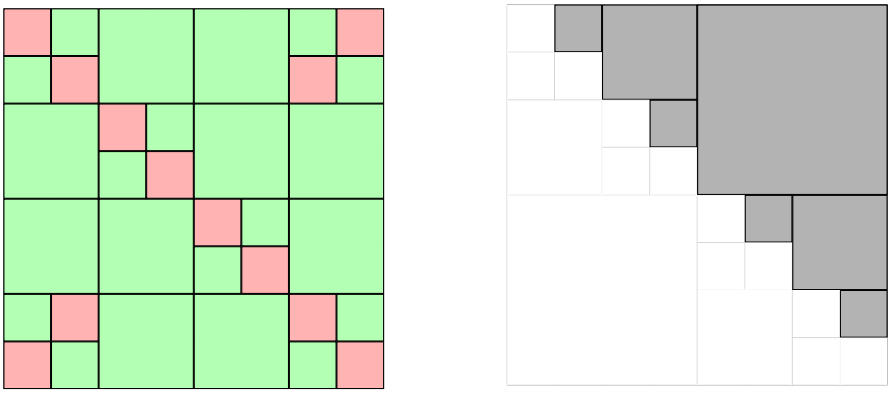
is satisfied for all $\tau\times\tau' \in P_{J\times J}^{upright}$ *and* S_1^{comb}, S_2^{comb} *are zero everywhere else.*

By definition the $\mathcal{H}$-block partitions $P_{J\times J}^{\mathcal{H},S_1}$ and $P_{J\times J}^{\mathcal{H},S_2}$ satisfy

$$P_{J\times J}^{\mathcal{H},S_1} = \bigcup_{\tau\times\tau'\in P_{J\times J}^{upright}} P_{\tau\times\tau'}^{\mathcal{H},S_1^{\tau,\tau'}} + (*),$$

$$P_{J\times J}^{\mathcal{H},S_2} = \bigcup_{\tau\times\tau'\in P_{J\times J}^{upright}} P_{\tau\times\tau'}^{\mathcal{H},S_2^{\tau,\tau'}} + (*),$$

where $(*)$ describes the remaining blocks not covered by block clusters contained in $P_{J\times J}^{upright}$.



(a) Blocks in $P^{\mathcal{H}}_{J\times J}$ based on some $P^{hier}_{J\times J}$.

(b) Blocks in $P^{upright}_{J\times J}$ based on the same $P^{hier}_{J\times J}$ as in (a).

Figure 3.4. The set $P^{upright}_{J\times J}$ for a given $\mathcal{H}$-matrix.

S^{comb}_1 and S^{comb}_2 are (strictly) upper triangular matrices for which any block on the diagonal with no descendants is admissible and contains only zeros. Note that the blocks induced by $P^{upright}_{J\times J}$ in S^{comb}_1 and S^{comb}_2 are not necessarily admissible. They are far more likely to be inadmissible and contain further subdivided blocks. $P^{upright}_{J\times J}$ only creates the overall structure in which the individual update matrices are embedded.

The following theorem is the generalization of Example 3.3. Note that there is a small but important difference in the upcoming definitions of $A^{upd}_{\sigma,\tau}$ and A^{fin} as the first can - depending on the choice of σ and τ - include only some of the updates of A during the algorithm and the second always includes all updates.

Theorem 3.5 *Let us have the situation of Theorem 3.2. Let $A_{\sigma,\tau}$ be a block in A induced by $\sigma\times\tau\in P^{hier}_{I\times J}$ with level$(\tau)=\ell$. Then $A^{upd}_{\sigma,\tau}$ after all updates made by Algorithm 3 before the potential vertical split of $A_{\sigma,\tau}$, is given by*

$$A^{upd}_{\sigma,\tau} = A_{\sigma,\tau} - \sum_{\widehat{\tau}\in P_J^{level(\tau)}:\ \widehat{\tau}<\tau} Y_{\sigma,\widehat{\tau}}\cdot S^{comb}_{2;\widehat{\tau},\tau}. \tag{3.1}$$

The matrix after all updates - including after all vertical splits - made by Algorithm 3 is given by

$$A^{fin} = A - Y\cdot S^{comb}_2. \tag{3.2}$$

Let $\tau\times\tau'\in P^{upright}_{J\times J}$, and let $A^{\tau'}_{I,\tau'}$ denote the partially updated block column before the update involving $S^{comb}_{1;\tau,\tau'}$ and $S^{comb}_{2;\tau,\tau'}$ is applied - meaning it is the input argument A_{I,τ_2} in Algorithm 3.9. Let $A^{inner,\tau'}_{I,\tau'}$ be the sum of all subsequent updates of $A^{\tau'}_{I,\tau'}$ including the

update involving $S^{comb}_{1;\tau,\tau'}$ *and* $S^{comb}_{2;\tau,\tau'}$*. Then*

$$Y^T \cdot A^{fin} = S^{comb}_1 + W + Z \tag{3.3}$$

holds true, where Z *has the same zero blocks as* S^{comb}_1 *and is otherwise blockwise given by*

$$Z_{\tau,\tau'} = Y^T_{I,\tau} \cdot A^{inner,\tau'}_{I,\tau'}$$

and W *is - using* L_{JJ} *as the number of levels in the hierarchy of block partitionings* $(P^\ell_{J\times J})_{\ell=1}^{L_{JJ}}$ *- for* $\tau,\tau' \in P^{hier}_J$ *blockwise given by*

$$W_{\tau,\tau'} = \begin{cases} Y^T_{I,\tau} \cdot A^{fin}_{I,\tau'} & \tau\times\tau' \in P^{upright}_{J\times J} \vee (\tau = \tau' \wedge level(\tau) = L_{JJ}) \\ 0 & else \end{cases}.$$

Furthermore, we have

$$T^T \cdot S^{comb}_1 = S^{comb}_2 + X,$$

where X *is blockwise given by*

$$X_{\tau,\tau'} = T^T_{J\setminus\tau,\tau} \cdot S^{comb}_{1;J\setminus\tau,\tau'}$$

for all $\tau\times\tau' \in P^{upright}_{J\times J}$ *and*

$$X_{\tau,\tau'} = T^T_{J,\tau} \cdot S^{comb}_{1;J,\tau'}$$

for the remaining blocks (given by an arbitrary partition of the set consisting of all $\tau\times\tau' \in P^{hier}_{J\times J}$ *with* $\tau\times\tau' \cap \bigcup_{b\in P^{upright}_{J\times J}} b = \emptyset$*).*

Proof First we check the updates of the block $A_{\sigma,\tau}$ induced by $\sigma\times\tau \in P^{hier}_{I\times J}$ with $\tau \in P^\ell_J$ that happen before it is split further. Let $\widehat{\tau} \in P^\ell_J$ with $\widehat{\tau} < \tau$. Then by Theorem 3.3 there are uniquely defined $\tau_1,\tau_2 \in P^{hier}_J$ with $\text{level}(\tau_1) = \text{level}(\tau_2) < \text{level}(\tau)$, $\widehat{\tau} \subset \tau_1$ and $\tau \subset \tau_2$ so that the update

$$\breve{A}_{I,\tau_2} - Y_{I,\tau_1} \cdot S^{comb}_{2;\tau_1,\tau_2}$$

is computed during the algorithm. Note that $\breve{A}_{I,\tau_2}$ is the matrix after all updates up to but not including to the one induced by the split into τ_1 and τ_2. The (sub-)block $\breve{A}_{\sigma,\tau}$ is hence updated by

$$\breve{A}_{\sigma,\tau} - Y_{\sigma,\tau_1} \cdot S^{comb}_{2;\tau_1,\tau},$$

which includes the update by

$$-Y_{\sigma,\widehat{\tau}} \cdot S^{comb}_{2;\widehat{\tau},\tau}$$

because $\hat{\tau} \subset \tau_1$. Using this argument for all blocks $\widehat{\tau} \in P_J^{\ell}$ with $\widehat{\tau} < \tau$ shows equation 3.1.

The second equation 3.2 follows from the first by applying it to blocks in A of the highest level and the fact that by construction of S_1^{comb}

$$S^{comb}_{2;\widehat{\tau},\tau} = 0$$

holds with $\tau \in P_J^{hier}$ for all $\widehat{\tau} \in P_J^{\text{level}(\tau)}$ with $\widehat{\tau} > \tau$ and we have

$$S^{comb}_{2;\tau,\tau} = 0$$

if $\tau \in P_J^{hier}$ is on the highest level. For the third equation 3.3, let $\tau, \tau' \in P_J^{hier}$ with $\tau \times \tau' \in P_{J\times J}^{hier}$. Then

$$\begin{aligned} Y^T_{I,\tau} \cdot A^{fin}_{I,\tau'} &= Y^T_{I,\tau} \cdot \left(A^{\tau'}_{I,\tau'} + A^{inner,\tau'}_{I,\tau'} \right) \\ &= Y^T_{I,\tau} \cdot A^{\tau'}_{I,\tau'} + Y^T_{I,\tau} \cdot A^{inner,\tau'}_{I,\tau'} \\ &= S^{comb}_{1,\tau,\tau'} + W_{\tau,\tau'} + Z_{\tau,\tau'}, \end{aligned}$$

where $A^{\tau'}_{I,\tau'}$ denotes the partially updated block column used during the update involving $S^{comb}_{1;\tau,\tau'}$ and $A^{inner,\tau'}_{I,\tau'}$ all subsequent updates. Note that $A^{\tau'}_{I,\tau'}$ is not the actual argument in Algorithm 3 because it is an entire block column of (the partially updated) A. However, all additional blocks correspond by construction to zero blocks in $Y^T_{I,\tau}$ and thus we actually compute $S^{comb}_{1,\tau,\tau'}$.

The fourth and last statement is true because for any $\tau, \tau' \in P_J^{hier}$ with $\tau \times \tau' \in P_{J\times J}^{upright}$ it holds that

$$T^T_{J,\tau} \cdot S^{comb}_{1;J,\tau'} = T^T_{\tau,\tau} \cdot S^{comb}_{1;\tau,\tau'} + T^T_{J\setminus\tau,\tau} \cdot S^{comb}_{1;J\setminus\tau,\tau'} = S^{comb}_{2;\tau,\tau'} + X_{\tau,\tau'}$$

and

$$S^{comb}_{2;\tau,\tau'} = 0 \;\forall\; \tau \times \tau' \in P^{hier}_{J\times J} : \tau \times \tau' \cap \bigcup_{b \in P^{upright}_{J\times J}} b = \emptyset. \qquad \blacksquare$$

Hence, we have shown that the calculations in all updates that are performed throughout the algorithm can be seen as part of three $\mathcal{H}$-matrix-matrix multiplications. The

calculation of all intermediate results S_1 is part of

$$Y^T \cdot A^{fin},$$

the calculation of all intermediate results S_2 is part of

$$T^T \cdot S_1^{comb}$$

and the updates themselves are given by

$$A - Y \cdot S_2^{comb}.$$

The only calculation - aside from the (dense) QR decompositions of the highest level - we have not yet paid attention to is the computation of $\overline{T}$. However, all those calculations are by definition simply a part of

$$Y^T \cdot Y.$$

This result will be essential for our cost analysis in Subsection 3.3.5 because it allows us to relate the overall computational cost of our algorithm to the computational cost of a small number of $\mathcal{H}$-matrix multiplications.

3.3 Complexity Analysis

We want to bound the computational cost of the block Householder approach for the QR decomposition of a given $\mathcal{H}$-matrix that we have just developed relative to its storage in $\mathcal{H}$-matrix representation. We do this in several steps. First, we derive partially induced admissibility functions (see Definitions 2.24 and 2.25) for all factors in the QR decomposition, including the agglomerated update matrices S_1^{comb} and S_2^{comb} (see Definition 3.4), based only on the admissibility function of the starting matrix (see Theorem 3.9 and 3.11). Furthermore, we can show that the admissibility functions of S_1^{comb} and S_2^{comb} can never be worse than the admissibility function of R, meaning admissible blocks of R are also admissible in S_1^{comb} and S_2^{comb} (see Theorem 3.11 again). Then we develop bounds on the cost of the whole algorithm, depending only on the parameters like the sparsity constant, depth, maximum rank of the factors Y, R and $\overline{T}$ that would also occur if we wanted to bound an $\mathcal{H}$-matrix multiplication involving these factors (see Theorem 3.29). Thus, if there is a useful QR decomposition of a given $\mathcal{H}$-matrix into the factors Y, R and the reduced factor $\overline{T}$, we can compute it efficiently with the proposed algorithm.

In addition, we can bound the $\mathcal{H}$-matrix bandwidth constants of R and $\overline{T}$ by the $\mathcal{H}$-

matrix bandwidth constant of A and Y (see Definition 2.29 and Theorem 3.20 for the result). Since we also show that the bandwidth of Y is smaller than or equal to that of A for square matrices A (see Theorem 3.21), we can then base the entire cost estimation in this case only on the bandwidth of A. Thus, the QR decomposition of any low-bandwidth square $\mathcal{H}$-matrix can be efficiently computed using our proposed algorithm.

Furthermore, under relatively weak additional assumptions, the factor T - in its unreduced form - consists entirely of inadmissible blocks (see Theorem 3.12), and the partially induced admissibility functions for Y and $\overline{T}$ that we develop are actually fully induced (see Theorems 3.14 and 3.15), meaning they are not only an upper bound but exact.

Finally, we derive a lower bound for the admissibility function of the explicitly computed Q - meaning the true (fully) induced admissibility function may have more, not fewer, inadmissible blocks - and thus show that Q, if the induced admissibility condition is a good measure for its data sparsity, does not allow for a good representation in the $\mathcal{H}$-matrix format, even if one exists for A (see Theorem 3.18).

We will extensively use the addition and multiplication that was defined directly after Definition 2.29 and used in Theorems 2.30 and 2.31. Note that for these products, it always holds that

$$a \cdot b \leq a \text{ and } a \cdot b \leq b$$

for $a, b \in \{\text{False}, \text{True}\}$. In the following, we will first define all $\mathcal{H}$-matrices and corresponding structures because many of the upcoming theorems will rely on mostly the same set of $\mathcal{H}$-matrices. This will considerably shorten the prerequisites of theorems and enhance the readability. Note that a theorem may be based on Prerequisites A without requiring all matrices defined in it. In many cases the unreduced matrix T and the factor Q in its explicit form are not directly needed.

Prerequisites A Let $A \in \mathcal{H}(P^{\mathcal{H},A}_{I\times J}, k, \text{adm}_A)$ be an $\mathcal{H}$-matrix with QR decomposition

$$A = Q \cdot R_0 = (I - YTY^T) \cdot R_0$$

given by

$$\begin{aligned} &Y \in \mathcal{H}(P^{\mathcal{H},Y}_{I\times J}, k, \text{adm}_Y), \quad T \in \mathcal{H}(P^{\mathcal{H},T}_{J\times J}, k, \text{adm}_T), \\ &R_0 \in \mathcal{H}(P^{\mathcal{H},R_0}_{I\times J}, k, \text{adm}_{R_0}), \quad R \in \mathcal{H}(P^{\mathcal{H},R}_{J\times J}, k, \text{adm}_R), \\ &Q \in \mathcal{H}(P^{\mathcal{H},Q}_{I\times I}, k, \text{adm}_Q), \end{aligned}$$

where R is the upper triangular square part of R_0 after permutation to an upper

trapezoidal matrix. Let

$$\overline{T} \in \mathcal{H}(P^{\mathcal{H},\overline{T}}_{J\times J}, k, \mathrm{adm}_{\overline{T}})$$

be the reduced form of T (see Remark 2.16) and

$$S_1^{comb} \in \mathcal{H}(P^{\mathcal{H},S_1}_{J\times J}, k, \mathrm{adm}_{S_1}), \quad S_2^{comb} \in \mathcal{H}(P^{\mathcal{H},S_2}_{J\times J}, k, \mathrm{adm}_{S_2})$$

be the agglomerated update matrices described in Definition 3.4. Furthermore, let

$$A^{fin} \in \mathcal{H}(P^{\mathcal{H},Y}_{I\times J}, k, \mathrm{adm}_{A^{fin}})$$

denote the matrix that results after all updates have been applied to A (see Theorem 3.5). The corresponding hierarchies of block partitionings are denoted by

$$\begin{aligned}
&(P^{\ell}_{I\times I})_{\ell=0}^{L_{II}} \text{ for } Q,\\
&(P^{\ell}_{I\times J})_{\ell=0}^{L_{IJ}} \text{ for } A, A^{fin}, Y, R_0\\
&(P^{\ell}_{J\times J})_{\ell=0}^{L_{JJ}} \text{ for } T, \overline{T}, R, S_1^{comb}, S_2^{comb},
\end{aligned}$$

which are themselves built upon the hierarchies of partitionings

$$\left(P_I^{\ell}\right)_{\ell=0}^{L_I} \text{ for } I, \quad \left(P_J^{\ell}\right)_{\ell=0}^{L_J} \text{ for } J.$$

Finally, let

$$\mathrm{depth}\left(P^{\mathcal{H},A}_{I\times J}\right) = L_{IJ}$$

hold, and let the admissibility functions adm_Y of Y, $\mathrm{adm}_{\overline{T}}$ of $\overline{T}$, adm_R of R, adm_{S_1} of S_1^{comb}, adm_{S_2} of S_2^{comb} and adm_Q of Q be induced by the $\mathcal{H}$-matrix arithmetics used in their computation.

From now on, the $\mathcal{H}$-matrix R_0 will only be of subordinate importance; we will usually refer to R instead because it makes certain statements easier to write down.

Note that by using J as the set of row indices for R, we need to identify relevant row index sets $\sigma \in P_I^{hier}$ - meaning with potential non-zero entries in the corresponding row of R_0 - by their "natural" partner index set $\tau \in P_J^{hier}$, meaning τ is the column index set so that $\sigma \times \tau$ induces a lower triangular block in Y (and an upper triangular block in R_0). This will be formalized in the upcoming Definition 3.6, visualized in Figure 3.5 and is especially useful for the statements in Subsection 3.3.4.

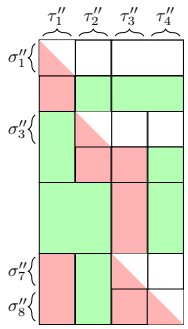


Figure 3.5. We recall the $\mathcal{H}$-matrix Y from Figure 3.3c and set $\tau_1' = \tau_1'' \cup \tau_2''$, $\tau_2' = \tau_3'' \cup \tau_4''$ and $\tau_1 = \tau_1' \cup \tau_2'$. Then, following the notation from Definition 3.6, the partner index sets involving τ_1'' are given by $\sigma(\tau_1'') = \sigma_1''$, $\sigma(\tau_1') = \sigma_1'' \cup \sigma_3''$ and $\sigma(\tau_1) = \sigma_1'' \cup \sigma_3'' \cup \sigma_7'' \cup \sigma_8''$. Note that both $\sigma(\tau_1')$ and $\sigma(\tau_1)$ are not part of the hierarchy of block partitionings of Y, which shows why defining the extended admissibility condition is necessary.

Furthermore, to apply our method of calculating the induced admissibility function, we need to transform the admissibility function that refers to the rectangular matrix R_0 into an admissibility function that refers to the square matrix R and vice versa. For that, we need to define the so-called extended admissibility function that can also process blocks that are not in $P_{I\times J}^{hier}$ but in the power set of $I \times J$ given by $\mathcal{P}(I \times J)$. This will also be defined in the following and is illustrated in Figure 3.5.

Definition 3.6 (Partner index set and extended admissibility function) *Let Prerequisites A be satisfied and let $P_{I\times J}^{split}$ be defined as in Assumption C. Define for $\tau \in P_J^{hier}$ the set of all $\sigma \in P_I^{hier}$ so that $\sigma \times \tau'$ with $\tau' \in P_J^{hier}, \tau' \subseteq \tau$ is an element in $P_{I\times J}^{split}$, meaning it induces a lower triangular block in Y and an upper triangular block in R, by*

$$\Sigma(\tau) = \{\sigma \in P_I^{hier} : \exists\, \tau' \in P_J^{hier} : \tau' \subseteq \tau \wedge \sigma \times \tau' \in P_{I\times J}^{split}\}$$

and its union by

$$\sigma(\tau) = \bigcup_{\sigma \in \Sigma(\tau)} \sigma.$$

Then $\sigma(\tau) \subset I$ is called the partner index set of τ in Y. Let adm be an admissibility function on the hierarchy of block partitionings $(P_{I\times J}^{\ell})_{\ell=0}^{L_{IJ}}$ that is not necessarily consistent with any of the named $\mathcal{H}$-block partitions from Prerequisites A. Then we define the function

$$adm^{ext} : \mathcal{P}(I \times J) \to \{True, False\},$$

where $\mathcal{P}$ denotes the power set, by

$$adm^{ext}(\sigma \times \tau) = \begin{cases} adm(\sigma \times \tau) & \sigma \times \tau \in P^{hier}_{I\times J}, \\ True & \exists\ \widetilde{\sigma} \times \tau \in P^{hier}_{I\times J} : \sigma \subset \widetilde{\sigma}, adm_A(\widetilde{\sigma} \times \tau) = True, \\ False & else. \end{cases}$$

and call it the extended admissibility function adm^{ext}.

We continue with some preliminary work and define admissibility functions that are forced to be row-wise or column-wise oriented (see Definition 2.27) while keeping all inadmissible blocks inadmissible and as many admissible blocks admissible as possible. Note that - as already explained after Definition 2.18 - if a set σ (technically) has more than one level we choose level(σ) to identify the highest level as a representative.

Definition 3.7 *Let $A \in \mathcal{H}(P^{\mathcal{H},A}_{I\times J}, k_A, adm_A)$ be an $\mathcal{H}$-matrix with admissibility function adm_A and $\hat{\sigma} \times \hat{\tau} \in P^{hier}_{I\times J}$. We define admissibility functions*

$$\begin{aligned} adm_A^{rw(\hat{\sigma},\hat{\tau})} &: P^{hier}_{I\times J} \to \{True, False\}, \\ adm_A^{cw(\hat{\sigma},\hat{\tau})} &: P^{hier}_{I\times J} \to \{True, False\} \end{aligned}$$

blockwise for $\sigma \times \tau \in P^{hier}_{I\times J}$ by

$$\begin{aligned} adm_A^{rw(\hat{\sigma},\hat{\tau})}(\sigma \times \tau) =& adm_A(\sigma \times \tau) \cdot \prod_{\substack{\tau' \in P_J^{level(\tau)}:\ \sigma\times\tau' \subseteq \hat{\sigma}\times\hat{\tau}, \\ \tau' < \tau}} adm_A(\sigma \times \tau'), \\ adm_A^{cw(\hat{\sigma},\hat{\tau})}(\sigma \times \tau) =& adm_A(\sigma \times \tau) \cdot \prod_{\substack{\sigma' \in P_I^{level(\sigma)}:\ \sigma'\times\tau \subseteq \hat{\sigma}\times\hat{\tau}, \\ \sigma' < \sigma}} adm_A(\sigma' \times \tau). \end{aligned}$$

We further set

$$\begin{aligned} adm_A^{rw}(\sigma \times \tau) :=& adm_A^{rw(I,J)}(\sigma \times \tau), \\ adm_A^{cw}(\sigma \times \tau) :=& adm_A^{cw(I,J)}(\sigma \times \tau). \end{aligned}$$

An $\mathcal{H}$-matrix $A^{rw} \in \mathcal{H}(P^{\mathcal{H},A^{rw}}_{I\times J}, k_A, \mathrm{adm}_A^{rw})$ is by definition row-wise oriented and furthermore, any inadmissible matrix block from $A \in \mathcal{H}(P^{\mathcal{H},A}_{I\times J}, k_A, \mathrm{adm}_A)$ is also inadmissible in A^{rw}. A^{rw} is the $\mathcal{H}$-matrix with the most admissible blocks that satisfies these two conditions. A similar statement can be made for $A^{cw} \in \mathcal{H}(P^{\mathcal{H},A^{cw}}_{I\times J}, k_A, \mathrm{adm}_A^{cw})$. See also Figure 2.5 for examples of row-wise or column-wise orientation.

3.3.1 Upper Bounds for the Induced Admissibility Functions

We begin by proving that multiplication with a square $\mathcal{H}$-matrix that has at least an $\mathcal{H}_p$-matrix structure, meaning all blocks on the diagonal are inadmissible (see Definition 2.51), can only increase and not decrease the number of inadmissible blocks in the result.

Theorem 3.8 *Let $A \in \mathcal{H}(P_{I\times I}^{\mathcal{H},A}, k_A, adm_A)$ be an $\mathcal{H}$-matrix with admissibility function adm_A that satisfies*

$$adm_A(\sigma \times \sigma) = \mathit{False}\ \forall\ \sigma \in P_I^{hier}.$$

Let $B \in \mathcal{H}(P_{I\times J}^{\mathcal{H},B}, k_B, adm_B)$ and $C \in \mathcal{H}(P_{I\times J}^{\mathcal{H},C}, k_C, adm_C)$ be $\mathcal{H}$-matrices with structures compatible with the multiplication $C = A \cdot B$ (see Definition 2.25). Let adm_C be partially induced by the multiplication w.r.t. adm_A and adm_B. Then it holds that

$$adm_B(\sigma \times \tau) \geq adm_C(\sigma \times \tau)\ \forall\ \sigma \times \tau \in P_{I\times J}^{hier}.$$

Proof Using the definition of the induced admissibility function for the product $A \cdot B$ in the first equation, we have

$$\begin{aligned}\mathrm{adm}_C(\sigma \times \tau) &= \prod_{\hat{\sigma} \in P_I^{Lmax(\sigma,\tau,I)}} (\mathrm{adm}_A(\sigma \times \hat{\sigma}) + \mathrm{adm}_B(\hat{\sigma} \times \tau)) \\ &\leq \mathrm{adm}_A(\sigma \times \sigma) + \mathrm{adm}_B(\sigma \times \tau) = \mathrm{adm}_B(\sigma \times \tau)\end{aligned}$$

and thus the desired result. ∎

We continue by describing the admissibility functions of the factors Y and $\overline{T}$.

Theorem 3.9 *Let Prerequisites A be satisfied. Then we have*

$$adm_Y(\sigma \times \tau) \geq \prod_{\hat{\tau} \in P_J^{level(\tau)}:\ \hat{\tau} \leq \tau} adm_A(\sigma \times \hat{\tau})$$

for all $\sigma \times \tau \in P_{I\times J}^{hier}$ and

$$adm_{\overline{T}}(\tau \times \tau') = \prod_{\hat{\sigma} \in P_I^{Lmax(\tau,\tau',I)}} adm_Y(\hat{\sigma} \times \tau) + adm_Y(\hat{\sigma} \times \tau')$$

for all $\tau \times \tau' \in P_{J\times J}^{hier}$.

Proof By construction, every block of Y has the same induced admissibility function as the corresponding block in the updated A after all recursive updates of the columns are done because with Theorems 3.1 and 3.2 there are no changes to the $\mathcal{H}$-matrix structure

during the actual computation of Y. This matrix that occurs after all updates have been applied was already relevant in Theorem 3.5 and was denoted by A^{fin}. Hence, we only need to check for the change of admissibility in the updated A induced by the operation in line 3 in Algorithm 3.9. The statement is true for the first block in every block row of Y on every level (of the corresponding hierarchy of block partitionings) because no update of a lower or the same level is applied and updates of higher levels cannot - by construction of the algorithm and especially Theorem 3.2 - have influence on the admissibility.

For all other blocks, we prove this by induction. Assume that the statement is true for the first $i-1$ blocks in every block row on every level of the hierarchy of block partitionings and let $\sigma \times \tau_i \in P_{I\times J}^{hier,Y}$ be the i-th block in the block row corresponding to $\sigma \in P_I^{hier}$. With Theorem 3.5, the admissibility of this block can be computed by

$$\begin{aligned}
\mathrm{adm}_Y(\sigma \times \tau_i) &= \mathrm{adm}_A(\sigma \times \tau_i) \cdot \prod_{\hat{\tau} \in P_J^{\mathrm{level}(\tau_i)}:\ \hat{\tau} < \tau_i} (\mathrm{adm}_Y(\sigma \times \hat{\tau}) + \mathrm{adm}_{S_2}(\hat{\tau} \times \tau_i)) \\
&\geq \mathrm{adm}_A(\sigma \times \tau_i) \cdot \prod_{\hat{\tau} \in P_J^{\mathrm{level}(\tau_i)}:\ \hat{\tau} < \tau_i} \mathrm{adm}_Y(\sigma \times \hat{\tau}) \\
&\geq \mathrm{adm}_A(\sigma \times \tau_i) \cdot \prod_{\hat{\tau} \in P_J^{\mathrm{level}(\tau_i)}:\ \hat{\tau} < \tau_i} \left(\prod_{\widetilde{\tau} \in P_J^{\mathrm{level}(\tau)}:\ \widetilde{\tau} \leq \hat{\tau}} \mathrm{adm}_A(\sigma \times \widetilde{\tau}) \right) \\
&= \prod_{\hat{\tau} \in P_J^{\mathrm{level}(\tau_i)}:\ \hat{\tau} \leq \tau_i} \mathrm{adm}_A(\sigma \times \widetilde{\tau}).
\end{aligned}$$

In the penultimate step, we used our induction assumption. Thus, the statement is true for the i-th block in every block row on every level if it is true for all up to the $(i-1)th$ block in every block row on every level and by that for all blocks.

Now we prove the statement about $\mathrm{adm}_{\overline{T}}$. For off-diagonal blocks, the statement is just the definition of the induced product admissibility function for the product $Y^T \cdot Y$ and hence true by definition of $\overline{T}$. For all diagonal blocks $\tau \times \tau \in P_{J\times J}^{hier}$ of $\overline{T}$, we have

$$\mathrm{adm}_{\overline{T}}(\tau \times \tau) = \mathrm{False}$$

because all diagonal blocks on the highest level are part of $P_{I\times J}^{split}$ and thus inadmissible. For the statement to be correct, this must also hold for

$$\prod_{\sigma \in P_I^{L^{max}(\tau,\tau,I)}} (\mathrm{adm}_Y(\sigma \times \tau) + \mathrm{adm}_Y(\sigma \times \tau)) = \prod_{\sigma \in P_I^{L^{max}(\tau,\tau,I)}} \mathrm{adm}_Y(\tau \times \sigma).$$

This term is False if there exists an inadmissible block in the block column of Y induced by τ. Since Y is a permuted lower trapezoidal matrix, this is guaranteed to be the case

because there is always one block that contains a lower triangular block, which must be inadmissible. ∎

Theorem 3.10 *Let $A \in \mathcal{H}(P_{I\times I}^{\mathcal{H},A}, k_A, adm_A)$ be an $\mathcal{H}$-matrix, where all blocks above the block diagonal are admissible, meaning its (consistent) admissibility function adm_A satisfies*

$$adm_A(\sigma \times \sigma') = \mathit{True}\ \forall\ \sigma, \sigma' \in P_I^{hier} :\ \sigma < \sigma', \mathit{level}(\sigma') = \mathit{level}(\sigma).$$

Let $B \in \mathcal{H}(P_{I\times J}^{\mathcal{H},B}, k_B, adm_B)$ and $C \in \mathcal{H}(P_{I\times J}^{\mathcal{H},C}, k_C, adm_C)$ be $\mathcal{H}$-matrices with structures so that A and B are compatible for multiplication and result in C. Let adm_C be induced by the multiplication w.r.t. adm_A and adm_B. Then it holds that

$$adm_C(\sigma \times \tau) \geq adm_B^{cw}(\sigma \times \tau)\ \forall\ \sigma \times \tau \in P_{I\times J}^{hier},$$

meaning adm_C can only have inadmissible blocks in places, where adm_B^{cw} (see Definition 3.7) has inadmissible blocks.

Proof Let $\sigma\times\tau \in P_{I\times J}^{hier}$. Then using Theorem 2.31 for the admissibility of the $\mathcal{H}$-matrix product and the properties of A, we have

$$\begin{aligned}
\mathrm{adm}_C(\sigma \times \tau) &= \prod_{\hat{\sigma}\in P_I^{L^{max}(\sigma,\tau,I)}} (\mathrm{adm}_A(\sigma \times \hat{\sigma}) + \mathrm{adm}_B(\hat{\sigma} \times \tau)) \\
&= \prod_{\substack{\hat{\sigma}\in P_I^{L^{max}(\sigma,\tau,I)},\\ \hat{\sigma}\leq\sigma}} (\mathrm{adm}_A(\sigma \times \hat{\sigma}) + \mathrm{adm}_B(\hat{\sigma} \times \tau)) \\
&\quad \cdot \prod_{\substack{\hat{\sigma}\in P_I^{L^{max}(\sigma,\tau,I)},\\ \hat{\sigma}>\sigma}} (\mathrm{adm}_A(\sigma \times \hat{\sigma}) + \mathrm{adm}_B(\hat{\sigma} \times \tau)) \\
&\geq \prod_{\substack{\hat{\sigma}\in P_I^{L^{max}(\sigma,\tau,I)},\\ \hat{\sigma}\leq\sigma}} (\mathrm{False} + \mathrm{adm}_B(\hat{\sigma} \times \tau)) \cdot \prod_{\substack{\hat{\sigma}\in P_I^{L^{max}(\sigma,\tau,I)}\\ \hat{\sigma}>\sigma}} (\mathrm{True} + \mathrm{adm}_B(\hat{\sigma} \times \tau)) \\
&= \prod_{\substack{\hat{\sigma}\in P_I^{L^{max}(\sigma,\tau,I)},\\ \hat{\sigma}\leq\sigma}} \mathrm{adm}_B(\hat{\sigma} \times \tau) \\
&= \mathrm{adm}_B^{cw}(\sigma \times \tau)
\end{aligned}$$

and thus the statement. ∎

We can now describe the induced admissibility function of R using the admissibility functions of Y and A.

Theorem 3.11 *Let Prerequisites A be satisfied. Define a function on* $P^{hier}_{J\times J}$ *by*

$$a(\tau\times\tau') = \prod_{\sigma\in P_I^{L^{max}(\tau,\tau',I)}} adm_Y(\sigma\times\tau) + adm_A(\sigma\times\tau')$$

for all $\tau\times\tau' \in P^{hier}_{J\times J}$*. Then* a *is an admissibility function and furthermore using Definitions 3.6 (for the partner index set and the extended admissibility function) and 3.7 (for the forced column-wise admissibility condition), we have*

$$adm_R(\tau\times\tau') \geq adm_A^{ext}(\sigma(\tau)\times\tau')\cdot a^{cw}(\tau\times\tau')$$

and

$$adm_{S_1}(\tau\times\tau') \geq a^{cw}(\tau\times\tau'),$$
$$adm_{S_2}(\tau\times\tau') \geq a^{cw}(\tau\times\tau'),$$

as well as

$$adm_{S_1}(\tau\times\tau') \geq adm_R(\tau\times\tau'),$$
$$adm_{S_2}(\tau\times\tau') \geq adm_R(\tau\times\tau')$$

for all $\tau\times\tau' \in P^{hier}_{J\times J}$ *with* adm_{S_1} *and* adm_{S_2} *defined in Definition 3.4.*

Proof We first show that a is an admissibility function as given in Definition 2.21. If

$$a(\tau\times\tau') = \text{True}$$

holds, then there are no block pairs $\sigma\times\tau$, $\sigma\times\tau'$ for which

$$\text{adm}_Y(\sigma\times\tau) = \text{adm}_A(\sigma\times\tau') = \text{False}$$

is true. Hence, for every descendant $\hat{\tau}\subset\tau$ and $\hat{\tau}'\subset\tau'$ with $\hat{\tau}\times\hat{\tau}'\in P^{hier}_{J\times J}$ we also cannot find $\hat{\sigma}\in P_I^{hier}$ so that

$$\text{adm}_Y(\hat{\sigma}\times\hat{\tau}) = \text{adm}_A(\hat{\sigma}\times\hat{\tau}') = \text{False}$$

holds because adm_Y and adm_A are admissibility functions, and thus admissible blocks can only contain admissible blocks. By this we have

$$a(\tau\times\tau') = \text{True} \;\Rightarrow\; a(\hat{\tau}\times\hat{\tau}')\;\forall\;\hat{\tau}\times\hat{\tau}'\in P^{hier}_{J\times J} \text{ with } \hat{\tau}\times\hat{\tau}'\subset\tau\times\tau'$$

and a is an admissibility function. We now develop a bound for adm_R.

1. With line 2 of Algorithm 3.9, it holds that

$$S^{comb}_{2,\tau_p\times\tau_{p+1}} = T^T_{\tau_p,\tau_p}\cdot S^{comb}_{1;\tau_p\times\tau_{p+1}}$$

for $\tau_p, \tau_{p+1} \in P_J^{hier}$ with $\tau_p \times \tau_{p+1} \in P_{J\times J}^{upright}$ and $p, p+1 \in \mathbb{N}$ chosen according to the order defined in Definition 2.26 for some level $0 \leq \ell \leq L_J$. Because T^T is a block lower triangular matrix and thus its admissibility function satisfies the condition of Theorem 3.10, it holds that for any $\tau \times \tau' \subseteq \tau_p \times \tau_{p+1}$ with $\tau_p \times \tau_{p+1} \in P_{J\times J}^{upright}$

$$\mathrm{adm}_{S_2}(\tau \times \tau') \geq \mathrm{adm}_{S_1}^{cw}(\tau \times \tau').$$

2. Let $\tau_1, \tau_2 \in P_J^\ell$ be given so that τ_1 is the first and τ_2 is the second block according to the order defined in Definition 2.26 for some level $0 \leq \ell \leq L_J$. Then it holds that $\tau_1 \times \tau_2 \in P_{J\times J}^{upright}$ as given by Definition 3.4 and

$$S_{1;\tau_1,\tau_2}^{comb} = Y_{I,\tau_1}^T \cdot A_{I,\tau_2}.$$

Note here that A_{I,τ_2} is the original block column of A without any updates because there is only one block column of the same level left to it, and hence there has been no prior update.

Let $\tau, \tau' \in P_J^{hier}$ with $\tau \subset \tau_1$ and $\tau' \subset \tau_2$. Then with line 1 of Algorithm 3.9 we know that

$$S_{1;\tau,\tau'}^{comb} = Y_{I,\tau}^T \cdot A_{I,\tau'}$$

holds and thus, using Theorem 2.31 for the induced admissibility function of the $\mathcal{H}$-matrix product, we have

$$\begin{aligned}
\mathrm{adm}_{S_1}(\tau \times \tau') &= \prod_{\sigma \in P_I^{L^{max}(\tau,\tau',I)}} \mathrm{adm}_Y(\sigma \times \tau) + \mathrm{adm}_A(\sigma \times \tau') \\
&= a(\tau \times \tau') \\
&\geq a^{cw(\tau_1,\tau_2)}(\tau \times \tau') \\
&= a^{cw}(\tau \times \tau').
\end{aligned}$$

Let $\tau \subseteq \tau_j$ and $\tau' \subseteq \tau_{j+1}$ with $\tau_j \times \tau_{j+1} \in P_{J\times J}^{upright}$ and $j, j+1$ chosen according to the order defined in Definition 2.26 for some level $0 \leq \ell \leq L_J$. We continue with an induction proof. Assume that

$$\mathrm{adm}_{S_1}(\tau \times \tau') \geq a^{cw}(\tau \times \tau')$$

is true for all $\tau \subseteq \tau_j$, $\tau' \subseteq \tau_{j+1}$, $\tau_j \times \tau_{j+1} \in P_{J\times J}^{upright}$ with $1 \leq j < p$ on every level of $P_{J\times J}^{hier}$, where j and p describe the position of the block according to the order of Definition 2.26, meaning τ_j is the j-th and τ_p is the p-th block in the corresponding block row. We then show that this statement is also true for $j = p$. We use Theorem 3.5 to collect all updates applied to A at this point in the algorithm and - note

that some of the inequalities are explained in more detail below - get

$$\begin{aligned}
\mathrm{adm}_{S_1}(\tau \times \tau') = & \prod_{\sigma \in P_I^{L^{max}(\tau,\tau',I)}} \Bigg(\mathrm{adm}_Y(\sigma \times \tau) + \mathrm{adm}_A(\sigma \times \tau') \\
& \cdot \prod_{\hat{\tau} \in P_J^{\mathrm{level}(\tau')} :\, \hat{\tau} < \tau_p} \big(\mathrm{adm}_Y(\sigma \times \hat{\tau}) + \mathrm{adm}_{S_2}(\hat{\tau} \times \tau') \big) \Bigg) \\
\geq & \prod_{\sigma \in P_I^{L^{max}(\tau,\tau',I)}} \Bigg(\mathrm{adm}_Y(\sigma \times \tau) \\
& + \mathrm{adm}_A(\sigma \times \tau') \cdot \prod_{\hat{\tau} \in P_J^{\mathrm{level}(\tau')} :\, \hat{\tau} < \tau_p} \mathrm{adm}_{S_2}(\hat{\tau} \times \tau') \Bigg) \\
\geq & \Bigg(\prod_{\sigma \in P_I^{L^{max}(\tau,\tau',I)}} \big(\mathrm{adm}_Y(\sigma \times \tau) + \mathrm{adm}_A(\sigma \times \tau') \big) \Bigg) \\
& \cdot \prod_{\hat{\tau} \in P_J^{\mathrm{level}(\tau')} :\, \hat{\tau} < \tau_p} \mathrm{adm}_{S_2}(\hat{\tau} \times \tau') \\
= & \, a(\tau \times \tau') \cdot \prod_{\hat{\tau} \in P_J^{\mathrm{level}(\tau')} :\, \hat{\tau} < \tau_p} \big(\mathrm{adm}_{S_2}(\hat{\tau} \times \tau') \big) \\
\geq & \, a(\tau \times \tau') \cdot \prod_{\hat{\tau} \in P_J^{\mathrm{level}(\tau')} :\, \hat{\tau} < \tau_p} \big(a^{cw}(\hat{\tau} \times \tau') \big) \\
= & \, a^{cw}(\tau \times \tau').
\end{aligned}$$

For the second inequality we used

$$\max(a, \min(b, c)) \geq \min(\max(a, b), c)$$

for $a, b, c \in \{\mathrm{True}, \mathrm{False}\}$ and True $>$ False as defined before. The last inequality is derived as follows: Using the result of the first step, we have

$$\mathrm{adm}_{S_2}(\tau \times \tau') \geq \mathrm{adm}_{S_1}^{cw}(\tau \times \tau').$$

Furthermore, we assumed the statement is true for all $\tau \subseteq \tau_j$, $\tau' \subseteq \tau_{j+1}$, $\tau_j \times \tau_{j+1} \in P_{J \times J}^{upright}$ with $1 \leq j < p$ on every level of $P_{J \times J}^{hier}$. Using that, we also have

$$\mathrm{adm}_{S_1}^{cw}(\tau \times \tau') \geq a^{cw}(\tau \times \tau')$$

because any outer block (see Definition 2.21) in adm_{S_1} is also inadmissible in a^{cw} and both $\mathrm{adm}_{S_1}^{cw}$ and a^{cw} are column-wise oriented in S_1^{comb}.

3. Any block or set of blocks $\sigma(\tau) \times \tau'$ (see Definition 3.6) in A with $\tau, \tau' \in P_J^{hier}$ and $\tau \times \tau' \in P_{J\times J}^{hier}$ is updated only by blocks in the block row induced by $\sigma(\tau)$ up until the block $\sigma(\tau) \times \tau$. During all other updates, the corresponding block in Y is a zero block, and thus the block induced by $\sigma(\tau) \times \tau'$ does not change. Hence, we have

$$\begin{aligned}
\mathrm{adm}_R(\tau \times \tau') &= \mathrm{adm}_A^{ext}(\sigma(\tau) \times \tau') \cdot \prod_{\substack{\hat{\tau} \in P_J^{\mathrm{level}(\tau)} \\ \hat{\tau} \leq \tau}} \big(\mathrm{adm}_Y(\sigma(\tau) \times \hat{\tau}) + \mathrm{adm}_{S_2}(\hat{\tau} \times \tau')\big) \\
&\geq \mathrm{adm}_A^{ext}(\sigma(\tau) \times \tau') \cdot \prod_{\substack{\hat{\tau} \in P_J^{\mathrm{level}(\tau)} \\ \hat{\tau} \leq \tau}} \big(\mathrm{adm}_Y(\sigma(\tau) \times \hat{\tau}) + a^{cw}(\hat{\tau} \times \tau')\big) \\
&\geq \mathrm{adm}_A^{ext}(\sigma(\tau) \times \tau') \cdot \prod_{\substack{\hat{\tau} \in P_J^{\mathrm{level}(\tau)} \\ \hat{\tau} \leq \tau}} \big(a^{cw}(\hat{\tau} \times \tau')\big) \\
&= \mathrm{adm}_A^{ext}(\sigma(\tau) \times \tau') \cdot a^{cw}(\tau \times \tau').
\end{aligned}$$

Note that the last equation used the column-wise orientation of a^{cw}, and we assumed at the beginning that $\tau \times \tau' \subseteq \tau_p \times \tau_{p+1} \in P_{J\times J}^{upright}$ holds. For any block $\tau \times \tau' \supseteq \tau_p \times \tau_{p+1}$ we have by construction of R, S_1^{comb} and S_2^{comb}

$$\mathrm{adm}_R(\tau \times \tau') = \mathrm{adm}_{S_1}(\tau \times \tau') = \mathrm{adm}_{S_2}(\tau \times \tau') = \mathrm{False},$$

and thus the statement is true as well.

Now we show the connection between the admissibility functions of R, S_1^{comb} and S_2^{comb}. Let $\tau, \tau' \in P_J^{hier}$ with $\tau \in \tau_p$ and $\tau' \in \tau_{p+1}$ for $\tau_p, \tau_{p+1} \in P_J^{hier}$ with $\tau_p \times \tau_{p+1} \in P_{J\times J}^{upright}$, $\tau \times \tau' \in P_{J\times J}^{hier}$ and $p, p+1$ are chosen according to the order defined in Definition 2.26 for some level $0 \leq \ell \leq L_J$. Since $\tau \times \tau$ is a diagonal block, we have $\mathrm{adm}_{\overline{T}}(\tau \times \tau) = \mathrm{False}$ and by that

$$\begin{aligned}
\mathrm{adm}_{S_2}(\tau \times \tau') &= \prod_{\substack{\hat{\tau} \in P_J^{L^{max}(\tau,\tau',J)} \\ \hat{\tau} \subset \tau_p}} \big(\mathrm{adm}_{\overline{T}}(\hat{\tau} \times \tau) + \mathrm{adm}_{S_1}(\hat{\tau} \times \tau')\big) \\
&\leq \mathrm{adm}_{\overline{T}}(\tau \times \tau) + \mathrm{adm}_{S_1}(\tau \times \tau') \\
&= \mathrm{adm}_{S_1}(\tau \times \tau').
\end{aligned}$$

The last statement then holds with

$$\begin{aligned}\operatorname{adm}_R(\tau \times \tau') &= \operatorname{adm}_A^{ext}(\sigma(\tau) \times \tau') \cdot \prod_{\substack{\hat{\tau} \in P_J^{\operatorname{level}(\tau)} \\ \hat{\tau} \leq \tau}} \left(\operatorname{adm}_Y(\sigma(\tau) \times \hat{\tau}) + \operatorname{adm}_{S_2}(\hat{\tau} \times \tau')\right) \\ &\leq \prod_{\substack{\hat{\tau} \in P_J^{\operatorname{level}(\tau)} \\ \hat{\tau} \leq \tau}} \left(\operatorname{adm}_Y(\sigma(\tau) \times \hat{\tau}) + \operatorname{adm}_{S_2}(\hat{\tau} \times \tau')\right) \\ &\leq \operatorname{adm}_Y(\sigma(\tau) \times \tau) + \operatorname{adm}_{S_2}(\tau \times \tau') \\ &= \operatorname{adm}_{S_2}(\tau \times \tau').\end{aligned}$$

The last equation holds because $\operatorname{adm}_Y(\sigma(\tau) \times \tau)$ is inadmissible due to the construction of $\sigma(\tau)$. ∎

We have found a way to bound the admissibility functions of all factors involved in our factorization of the QR decomposition by finding suitable partially induced admissibility functions. Two main questions to look into remain. First: How good are our bounds? We will answer this in the next section. Second: What does that mean for the cost of our algorithm? This will be the topic in Subsections 3.3.4 and 3.3.5.

3.3.2 Conditions for the Exactness of the Upper Bounds

Theorems 3.9 and 3.11 only give a lower bound on the number and position of the admissible blocks on every level of the hierarchy of block partitions. The question remains whether Y, $\overline{T}$ and R actually have only that many admissible blocks or if it is only a pessimistic estimate, meaning that there are more and larger admissible blocks. We want to answer that in this section. We will find additional conditions so that the partially induced admissibility functions given in Subsection 3.3.1 are actually fully induced. We will later see that these additional conditions are often satisfied if we deviate from the simple $\mathcal{H}_p$-matrix structure.

We begin by analyzing the induced admissibility function of the unreduced factor T. This will show us whether saving $\overline{T}$ instead of T leads to lower storage cost and is also relevant for later theorems.

Theorem 3.12 *Let Prerequisites A be true. Assume that for all neighboring block columns $Y_{I,\tau}$ and $Y_{I,\tau'}$ with $\tau, \tau' \in P_J^{L_J}$ there exists $\sigma \in P_I^{L^{max}(\tau,\tau',I)}$ so that*

$$adm_Y(\sigma \times \tau) = adm_Y(\sigma \times \tau') = False$$

holds true. Then all blocks in the upper triangular part of T are inadmissible, meaning

it holds that

$$adm_T(\tau \times \tau') = \mathit{False} \;\forall\; \tau \times \tau' \in P^{hier}_{J\times J} : \tau \leq \tau'.$$

Proof Let $\tau, \tau' \in P_J^{L_J}$ induce neighboring block columns $Y_{I,\tau}$ and $Y_{I,\tau'}$ in Y, and let $\hat{\sigma} \in P_I^{L^{max}(\tau,\tau',I)}$ be chosen so that

$$\mathrm{adm}_Y(\hat{\sigma} \times \tau) = \mathrm{adm}_Y(\hat{\sigma} \times \tau') = \mathrm{False}$$

holds. Let $\widetilde{\tau}$ be the first common ancestor of τ and τ'. Then $\widetilde{\tau}$ has two direct descendants $\hat{\tau}$ and $\hat{\tau}'$ with $\tau \subset \hat{\tau}$ and $\tau' \subset \hat{\tau}'$ so that (the reduced) $\overline{T}_{\hat{\tau},\hat{\tau}'}$ is computed by

$$\overline{T}_{\hat{\tau},\hat{\tau}'} = Y_{I,\hat{\tau}}^T Y_{I,\hat{\tau}'}.$$

The block $\overline{T}_{\tau,\tau'}$ is thus simply computed by

$$\overline{T}_{\tau,\tau'} = Y_{I,\tau}^T Y_{I,\tau'},$$

and we have

$$\begin{aligned}\mathrm{adm}_{\overline{T}}(\tau \times \tau') &= \prod_{\sigma \in P_J^{L^{max}(\tau,\tau',J)}} \mathrm{adm}_Y(\sigma \times \tau) + \mathrm{adm}_Y(\sigma \times \tau') \\ &\leq \mathrm{adm}_Y(\hat{\sigma} \times \tau) + \mathrm{adm}_Y(\hat{\sigma} \times \tau') \\ &= \mathrm{False} + \mathrm{False} = \mathrm{False}.\end{aligned}$$

To compute (the unreduced) $T_{\hat{\tau},\hat{\tau}'}$ we use

$$T_{\hat{\tau},\hat{\tau}'} = T_{\hat{\tau},\hat{\tau}} \cdot \overline{T}_{\hat{\tau},\hat{\tau}'} \cdot T_{\hat{\tau}',\hat{\tau}'}.$$

Both $T_{\hat{\tau},\hat{\tau}}$ and $T_{\hat{\tau}',\hat{\tau}'}$ are square matrices with inadmissible blocks on the block diagonal. We apply Theorem 3.8 twice (first to $T_{\hat{\tau},\hat{\tau}} \cdot \overline{T}_{\hat{\tau},\hat{\tau}'}$ and then to $T^T_{\hat{\tau}',\hat{\tau}'} \cdot \left(T_{\hat{\tau},\hat{\tau}} \cdot \overline{T}_{\hat{\tau},\hat{\tau}'}\right)^T$) and get

$$\mathrm{adm}_T(\hat{\tau} \times \hat{\tau}') \leq \mathrm{adm}_{\overline{T}}(\hat{\tau} \times \hat{\tau}')$$

and especially

$$\mathrm{adm}_T(\tau \times \tau') \leq \mathrm{adm}_{\overline{T}}(\tau \times \tau') = \mathrm{False}.$$

Hence

$$\mathrm{adm}_T(\tau \times \tau') = \mathrm{False}.$$

We continue with a proof by induction over the levels of the hierarchy of partitionings $\left(P_J^\ell\right)_{\ell=0}^{L_J}$ and thus have just proven the start of the induction for the highest level L_J.

Let $0 \leq \ell < L_J$ be a level and assume that

$$\mathrm{adm}_T(\tau \times \tau') = \text{False } \forall\ \tau \times \tau' \in P^{hier}_{J\times J} : \tau \subseteq \tau_k\ \wedge \tau' \subseteq \tau'_k$$

holds for all $\tau_k, \tau'_k \in P^k_J$ and k with $0 \leq \ell < k \leq L_J$ that induce neighboring block columns Y_{I,τ_k} and Y_{I,τ'_k} for some level $0 \leq \ell < L_J$. We show that the statement also holds for the ℓ-th level, meaning that

$$\mathrm{adm}_T(\tau \times \tau') = \text{False } \forall\ \tau \times \tau' \in P^{hier}_{J\times J} : \tau \subseteq \tau_\ell\ \wedge \tau' \subseteq \tau'_\ell$$

holds for all $\tau_\ell, \tau'_\ell \in P^\ell_J$ that induce neighboring block columns in Y.

Assume we have $\tau_\ell, \tau'_\ell \in P^\ell_J$ that induce neighboring block columns in Y. Then the blocks $\tau_\ell \times \tau_\ell$ and $\tau'_\ell \times \tau'_\ell$ themselves are inadmissible because they are diagonal blocks of T. Furthermore, they contain only inadmissible blocks on or above the diagonal because the, at most, 4 direct descendants of a diagonal block are either again a diagonal block or the row and column cluster induce neighboring block columns. In the latter case for the one above the diagonal, one can thus apply our assumption because they are at level $\ell + 1$.

Furthermore because τ_ℓ and τ'_ℓ induce neighboring block columns in Y there are descendants $\check{\tau} \in P^{L_J}_J$ of τ_ℓ and $\check{\tau}' \in P^{L_J}_J$ of τ'_ℓ that also induce neighboring block columns in Y. Thus, we have by our prerequisites

$$\mathrm{adm}_{\overline{T}}(\check{\tau} \times \check{\tau}') = \text{False}.$$

Now we have everything that we need. Let $\tau \times \tau' \in P^{L_J J}_{J\times J}$ with $\tau, \tau' \in P^{L_J}_J$ denote a block of highest level with $\tau \subset \tau_\ell$ and $\tau' \subset \tau'_\ell$. Then we compute the induced admissibility function of the triple product $T_{\tau_\ell,\tau'_\ell} = \left(T_{\tau_\ell,\tau_\ell} \cdot \overline{T}_{\tau_\ell,\tau'_\ell}\right) \cdot T_{\tau'_\ell,\tau'_\ell}$ and get

$$\begin{aligned}
\mathrm{adm}_T(\tau \times \tau') &= \prod_{\bar{\tau} \in P^{L_J}_J : \bar{\tau} \subseteq \tau'_\ell} \left(\prod_{\check{\tau} \in P^{L_J}_J : \check{\tau} \subseteq \tau_\ell} \left(\mathrm{adm}_T(\tau \times \check{\tau}) + \mathrm{adm}_{\overline{T}}(\check{\tau} \times \bar{\tau})\right) + \mathrm{adm}_T(\bar{\tau} \times \tau') \right) \\
&\leq \prod_{\check{\tau} \in P^{L_J}_J : \check{\tau} \subseteq \tau_\ell} \left(\mathrm{adm}_T(\tau \times \check{\tau}) + \mathrm{adm}_{\overline{T}}(\check{\tau} \times \check{\tau}')\right) + \mathrm{adm}_T(\check{\tau}' \times \tau') \\
&\leq \left(\mathrm{adm}_T(\tau \times \check{\tau}) + \mathrm{adm}_{\overline{T}}(\check{\tau} \times \check{\tau}')\right) + \mathrm{adm}_T(\check{\tau}' \times \tau') \\
&= \text{False} + \text{False} + \text{False} = \text{False}.
\end{aligned}$$

The last equation holds because $\tau \times \check{\tau} \subset \tau_\ell \times \tau_\ell$ and $\check{\tau}' \times \tau' \subset \tau'_\ell \times \tau'_\ell$. Hence, all blocks on the highest level contained in $\tau_\ell \times \tau'_\ell$ are inadmissible. Thus, all blocks contained in $\tau_\ell \times \tau'_\ell$ including $\tau_\ell \times \tau'_\ell$ are inadmissible. ∎

This statement, together with Theorem 3.9, shows that saving $\overline{T}$ instead of T has the potential to improve the storage cost because the condition of Theorem 3.12 is satisfied in most cases aside from $\mathcal{H}_p$-matrices, whereas $\overline{T}$ may only have a few inadmissible blocks in those cases.

In Subsection 3.3.1, we found a way to bound the admissibility of all factors relevant to the computation of our algorithm from below, meaning the true induced admissibility functions can have more but never fewer admissible blocks. However, we also observed considerable fill-in, especially in the factor Y (see Theorem 3.9). We are now interested in finding conditions under which the partially induced admissibility function for Y is actually fully induced, and hence this fill-in materializes. Before we do this, we need to state one simple preliminary finding first.

Theorem 3.13 *Let $A \in \mathcal{H}(P^{\mathcal{H},A}_{I\times J}, k_A, adm_A)$ be an $\mathcal{H}$-matrix, where for all neighboring block columns $A_{I,\tau}$ and $A_{I,\tau'}$ with $\tau, \tau' \in P^{L_J}_J$ there exists $\sigma \in P_I^{L^{max}(\tau,\tau',I)}$ so that*

$$adm_A(\sigma \times \tau) = adm_A(\sigma \times \tau') = False$$

holds. Then the same is true - independent of their level - for all neighboring block columns $A_{I,\tau}$ and $A_{I,\tau'}$ of A with $\tau, \tau' \in P^{hier}_J$, meaning we can find for each neighboring pair τ, τ' a row cluster set $\sigma \in P^{hier}_I$ so that

$$adm_A(\sigma \times \tau) = adm_A(\sigma \times \tau') = False$$

holds.

Proof Let $A_{I,\tau}$ and $A_{I,\tau'}$ be neighboring block columns of A with $\tau, \tau' \in P^{hier}_J$. If $\tau, \tau' \in P^{L_J}_J$ then the statement is true because it is identical to the prerequisites. Hence, assume that $\tau, \tau' \notin P^{L_J}_J$. Because $A_{I,\tau}$ and $A_{I,\tau'}$ are neighboring block columns, we can find $\hat{\tau} \subset \tau$ and $\hat{\tau}' \subset \tau'$ with $\hat{\tau}, \hat{\tau}' \in P^{L_J}_J$ that also induce neighboring block columns. With the prerequisites, we can thus find $\hat{\sigma} \in P_I^{L^{max}(\tau,\tau',I)}$ so that

$$\mathrm{adm}_A(\hat{\sigma} \times \hat{\tau}) = \mathrm{adm}_A(\hat{\sigma} \times \hat{\tau}') = \mathrm{False}$$

holds. Furthermore, by construction of the hierarchy of block partitionings, there is also $\sigma \in P^{hier}_I$ with $\hat{\sigma} \subset \sigma$ and $\sigma \times \tau, \sigma \times \tau' \in P^{hier}_{I\times J}$, and thus we have

$$\mathrm{adm}_A(\sigma \times \tau) = \mathrm{adm}_A(\sigma \times \tau') = \mathrm{False}$$

because by definition of an admissibility function, admissible blocks can only contain admissible blocks. This shows the overall statement. ∎

Now we can continue.

Theorem 3.14 *Let Prerequisites A be true. Assume that for all neighboring block columns $A_{I,\tau}$ and $A_{I,\tau'}$ of A with $\tau, \tau' \in P_J^{L_J}$ there exists $\sigma \in P_I^{L^{max}(\tau,\tau',I)}$ so that*

$$adm_A(\sigma \times \tau) = adm_A(\sigma \times \tau') = False.$$

holds true. Then the admissibility function for Y from Theorem 3.9 is (fully) induced, meaning it holds that

$$adm_Y(\sigma \times \tau) = \prod_{\hat{\tau} \in P_J^{level(\tau)} \colon \hat{\tau} \leq \tau} adm_A(\sigma \times \hat{\tau})$$

for all $\sigma \times \tau \in P_{I\times J}^{hier}$.

Proof We prove the statement in three steps. First, let $\sigma \times \tau \in P_{I\times J}^{hier}$ be inadmissible in A and non-zero in Y. Then by construction of our algorithm - inadmissible blocks in A induce inadmissible blocks in Y (see Theorem 3.1) - $\sigma \times \tau$ also induces an inadmissible block in Y, and hence in that case we have

$$\mathrm{adm}_Y(\sigma \times \tau) = \mathrm{False} = \mathrm{adm}_A(\sigma \times \tau) = \prod_{\hat{\tau} \in P_J^{\mathrm{level}(\tau)} \colon \hat{\tau} \leq \tau} \mathrm{adm}_A(\sigma \times \hat{\tau}).$$

Second, let $\sigma \times \tau$ induce an admissible block in A, and let there be no $\widetilde{\tau} \leq \tau$ so that

$$\mathrm{adm}_A(\sigma \times \widetilde{\tau}) = \mathrm{False}$$

holds. Then, applying Theorem 3.9, we have

$$\mathrm{adm}_Y(\sigma \times \tau) \geq \prod_{\hat{\tau} \in P_J^{\mathrm{level}(\tau)} \colon \hat{\tau} \leq \tau} \mathrm{adm}_A(\sigma \times \hat{\tau}) = \mathrm{True},$$

and thus

$$\mathrm{adm}_Y(\sigma \times \tau) = \mathrm{True} = \prod_{\hat{\tau} \in P_J^{\mathrm{level}(\tau)} \colon \hat{\tau} \leq \tau} \mathrm{adm}_A(\sigma \times \hat{\tau}).$$

Third, let $\sigma \times \check{\tau}$ induce an admissible block in A, and let there be a $\tau \leq \check{\tau}$ so that

$$\mathrm{adm}_A(\sigma \times \tau) = \mathrm{False}$$

holds. We want to show that all blocks to the right of $Y_{\sigma,\tau}$, including $Y_{\sigma,\check{\tau}}$, are also inadmissible.

Let τ' be the first block to the right of τ of the same level, meaning $Y_{I,\tau}$ and $Y_{I,\tau'}$ are neighboring block columns in Y. Apply Theorem 3.13, and let $\hat{\sigma} \in P_I^{L^{max}(\tau,\tau',I)}$ be

chosen so that

$$\mathrm{adm}_A(\hat{\sigma} \times \tau) = \mathrm{adm}_A(\hat{\sigma} \times \tau') = \mathrm{False}$$

holds. Let $\widetilde{\tau} \in P_J^{hier}$ be the first common ancestor of both. Then $\widetilde{\tau}$ has two direct descendants $\hat{\tau} \in P_J^{hier}$ and $\hat{\tau}' \in P_J^{hier}$ with $\tau \subset \hat{\tau}$ and $\tau' \subset \hat{\tau}'$ so that

$$S_{1;\hat{\tau},\hat{\tau}'} = Y_{I,\hat{\tau}}^T \cdot A_{I,\hat{\tau}'}^{\hat{\tau}'},$$

where $A_{I,\hat{\tau}'}^{\hat{\tau}'}$ denotes the correct partially updated block column of A directly before the update the with $Y_{I,\hat{\tau}}$ is applied. This is similar to the notation in Theorem 3.5. We thus have

$$S_{1;\tau,\tau'} = Y_{I,\tau}^T \cdot A_{I,\tau'}^{\hat{\tau}'}.$$

Because updates can only turn admissible blocks inadmissible and not the other way around, we replace the admissibility of $A_{I,\tau'}^{\hat{\tau}'}$ with the admissibility of $A_{I,\tau'}$ and get

$$\begin{aligned}
\mathrm{adm}_{S_1}(\tau \times \tau') &\leq \prod_{\sigma \in P_I^{L^{max}(\tau,\tau',I)}} \mathrm{adm}_Y(\sigma \times \tau) + \mathrm{adm}_A(\sigma \times \tau') \\
&\leq \mathrm{adm}_Y(\hat{\sigma} \times \tau) + \mathrm{adm}_A(\hat{\sigma} \times \tau') \\
&\leq \mathrm{adm}_A(\hat{\sigma} \times \tau) + \mathrm{adm}_A(\hat{\sigma} \times \tau') \\
&= \mathrm{False}.
\end{aligned}$$

Furthermore,

$$S_{2,\hat{\tau},\hat{\tau}'}^{comb} = T_{\hat{\tau},\hat{\tau}}^T \cdot S_{1;\hat{\tau},\hat{\tau}'}$$

holds true, and thus

$$S_{2,\tau,\tau'}^{comb} = T_{\hat{\tau},\tau}^T \cdot S_{1;\hat{\tau},\tau'}$$

leading to

$$\begin{aligned}
\mathrm{adm}_{S_2}(\tau \times \tau') &= \prod_{\overline{\tau} \in P_J^{L^{max}(\tau,\tau',J)} : \overline{\tau} \subseteq \hat{\tau}} \mathrm{adm}_T(\overline{\tau} \times \tau) + \mathrm{adm}_{S_1}(\overline{\tau} \times \tau') \\
&\leq \mathrm{adm}_T(\tau \times \tau) + \mathrm{adm}_{S_1}(\tau \times \tau') \\
&= \mathrm{False} + \mathrm{False} = \mathrm{False},
\end{aligned}$$

where we use the fact that all diagonal blocks of T must be inadmissible. Finally, we look at the update step

$$A_{\sigma,\tau'}^{upd} = A_{\sigma,\tau'} - Y_{\sigma,\tau} \cdot S_{2;\tau,\tau'}^{comb} - \sum_{\overline{\tau} < \tau} Y_{\sigma,\overline{\tau}} \cdot S_{2;\overline{\tau},\tau'}^{comb}.$$

We know that $Y_{\sigma,\tau}$ is inadmissible because $A_{\sigma,\tau}$ is inadmissible. Furthermore $S^{comb}_{2;\tau,\tau'}$ is inadmissible, then so is $A^{upd}_{\sigma,\tau'}$ because the sum of an inadmissible block with any other block is inadmissible and we have

$$\begin{aligned}\mathrm{adm}_Y(\sigma \times \tau') &\leq \mathrm{adm}_Y(\sigma \times \tau) + \mathrm{adm}_{S_2}(\tau \times \tau') \\ &= \mathrm{False} + \mathrm{False} = \mathrm{False}.\end{aligned}$$

Repeating this argument consecutively for all non-zero blocks to the right of $Y_{I,\tau'}$ proves that they are all, including $\sigma \times \check{\tau}$, inadmissible. Hence, we have

$$\mathrm{adm}_Y(\sigma \times \check{\tau}) = \mathrm{False} = \mathrm{adm}_A(\sigma \times \tau) = \prod_{\hat{\tau} \in P_J^{\mathrm{level}(\check{\tau})} :\ \hat{\tau} \leq \check{\tau}} \mathrm{adm}_A(\sigma \times \hat{\tau}),$$

and the entire statement is true. ∎

Now we check the factor R using the same ideas.

Theorem 3.15 *Let Prerequisites A be true, and let (as in Theorem 3.11) for all* $\tau \times \tau' \in P^{hier}_{J \times J}$

$$a(\tau \times \tau') = \prod_{\sigma \in P_I^{L^{max}(\tau,\tau',I)}} adm_Y(\sigma \times \tau) + adm_A(\sigma \times \tau')$$

define an admissibility function on $P^{hier}_{J \times J}$*. Assume that for all neighboring block columns* $A_{I,\tau}$ *and* $A_{I,\tau'}$ *of* A *and* $Y_{I,\tau}$ *and* $Y_{I,\tau'}$ *of* Y *with* $\tau, \tau' \in P_J^{L_J}$ *there exist* $\sigma_A, \sigma_Y \in P_I^{L^{max}(\tau,\tau',I)}$ *so that*

$$adm_A(\sigma_A \times \tau) = adm_A(\sigma_A \times \tau') = False$$

and

$$adm_Y(\sigma_Y \times \tau) = adm_Y(\sigma_Y \times \tau') = False$$

hold true. Furthermore, assume that the admissibility function a is column-wise oriented, meaning

$$a(\tau \times \tau') = a^{cw}(\tau \times \tau')$$

holds for all $\tau \times \tau' \in P^{hier}_{J \times J}$ *(see Theorem 3.16 for a case in which this is true). Then the admissibility function for R from Theorem 3.11 is (fully) induced, meaning*

$$adm_R(\tau \times \tau') = adm^{ext}_A(\sigma(\tau) \times \tau') \cdot a^{cw}(\tau \times \tau')$$

holds true. Furthermore, for S_1^{comb}, S_2^{comb} *it holds that*

$$adm_{S_1}(\tau \times \tau') = a(\tau \times \tau'),$$
$$adm_{S_2}(\tau \times \tau') = a(\tau \times \tau').$$

PROOF Let $\tau, \tau' \in P_J^{hier}$ with $\tau \in \tau_p$ and $\tau' \in \tau_{p+1}$ so that $\tau \times \tau' \in P_{I\times J}^{hier}$ and $\tau_p \times \tau_{p+1} \in P_{J\times J}^{upright}$. We do not have to prove our statement for blocks in $P_{J\times J}^{hier}$ because it is already true with Theorem 3.11. As in the proof for Theorem 3.11, we collect all updates applied to A at this point in the algorithm and get

$$\begin{aligned}
\mathrm{adm}_{S_1}(\tau \times \tau') = & \prod_{\sigma \in P_I^{L^{max}(\tau,\tau',I)}} \Bigg(\mathrm{adm}_Y(\sigma \times \tau) + \Bigg(\mathrm{adm}_A(\sigma \times \tau') \\
& \cdot \prod_{\hat{\tau} \in P_J^{level(\tau)}: \hat{\tau} < \tau} \left(\mathrm{adm}_Y(\sigma \times \hat{\tau}) + \mathrm{adm}_{S_2}(\hat{\tau} \times \tau') \right) \Bigg) \Bigg) \\
\leq & \prod_{\sigma \in P_I^{L^{max}(\tau,\tau',I)}} \mathrm{adm}_Y(\sigma \times \tau) + \mathrm{adm}_A(\sigma \times \tau') \\
= & \, a(\tau \times \tau') = a^{cw}(\tau \times \tau')
\end{aligned}$$

and thus with Theorem 3.11

$$\mathrm{adm}_{S_1}(\tau \times \tau') = a^{cw}(\tau \times \tau').$$

With Theorem 3.12 we know that under the given prerequisites the upper triangular part of T consists entirely of inadmissible blocks, hence using the calculation of S_2 given in Theorem 3.5 for the same $\tau, \tau' \in P_J^{hier}$ as before we have

$$\begin{aligned}
\mathrm{adm}_{S_2}(\tau \times \tau') = & \prod_{\substack{\hat{\tau} \in P_J^{L^{max}(\tau,\tau',J)} \\ \hat{\tau} \subset \tau_p}} \mathrm{adm}_T(\tau \times \hat{\tau}) + \mathrm{adm}_{S_1}(\hat{\tau} \times \tau') \\
= & \prod_{\substack{\hat{\tau} \in P_J^{L^{max}(\tau,\tau',J)} \\ \hat{\tau} \subset \tau_p}} \mathrm{adm}_{S_1}(\hat{\tau} \times \tau') \\
\leq & \prod_{\hat{\tau} \in P_J^{L^{max}(\tau,\tau',J)}} \mathrm{adm}_{S_1}(\hat{\tau} \times \tau') \\
= & \prod_{\hat{\tau} \in P_J^{L^{max}(\tau,\tau',J)}} a^{cw}(\hat{\tau} \times \tau') \\
= & \, a^{cw}(\hat{\tau} \times \tau')
\end{aligned}$$

and then with the result from Theorem 3.11

$$a^{cw}(\tau \times \tau') \geq \mathrm{adm}_{S_2}(\tau \times \tau') \geq a^{cw}(\tau \times \tau').$$

Hence

$$\mathrm{adm}_{S_2}(\tau \times \tau') = \mathrm{adm}_{S_1}(\tau \times \tau') = a^{cw}(\tau \times \tau') = a(\tau \times \tau').$$

Using $\mathrm{adm}_Y(\sigma(\tau) \times \tau) = \mathrm{False}$ by definition of $\sigma(\tau)$ and our results as well as $\tau, \tau' \in P_J^{hier}$ from before we have at last

$$\begin{aligned}
\mathrm{adm}_R(\tau \times \tau') &= \mathrm{adm}_A^{ext}(\sigma(\tau) \times \tau') \cdot \prod_{\substack{\hat{\tau} \in P^{\mathrm{level}(\tau)} \\ \hat{\tau} \leq \tau}} \big(\mathrm{adm}_Y(\sigma(\tau) \times \hat{\tau}) + \mathrm{adm}_{S_2}(\hat{\tau} \times \tau')\big) \\
&\leq \mathrm{adm}_A^{ext}(\sigma(\tau) \times \tau') \cdot \big(\mathrm{adm}_Y(\sigma(\tau) \times \tau) + \mathrm{adm}_{S_2}(\tau \times \tau')\big) \\
&= \mathrm{adm}_A^{ext}(\sigma(\tau) \times \tau') \cdot \mathrm{adm}_{S_2}(\tau \times \tau') \\
&= \mathrm{adm}_A^{ext}(\sigma(\tau) \times \tau') \cdot a(\tau \times \tau').
\end{aligned}$$

Combined with Theorem 3.11, we have

$$\mathrm{adm}_R(\tau \times \tau') = \mathrm{adm}_A^{ext}(\sigma(\tau) \times \tau') \cdot a(\tau \times \tau')$$

and thus our entire statement. ∎

The condition that the admissibility function a is column-wise oriented is somewhat technical, and we cannot easily see in which cases it holds. But we can show that for a square $\mathcal{H}$-matrix with a symmetric block structure, a is column-wise oriented if the condition from Theorem 3.14 is true.

Theorem 3.16 *Let Prerequisites A be true, and let (as in Theorem 3.11) for all* $\tau \times \tau' \in P_{J \times J}^{hier}$

$$a(\tau \times \tau') = \prod_{\sigma \in P_I^{L^{max}(\tau, \tau', I)}} adm_Y(\sigma \times \tau) + adm_A(\sigma \times \tau')$$

define an admissibility function on $P_{J \times J}^{hier}$*. Assume that for all neighboring block columns* $A_{I,\tau}$ *and* $A_{I,\tau'}$ *of* A *with* $\tau, \tau' \in P_J^{L_J}$ *there exists* $\sigma \in P_I^{L^{max}(\tau,\tau',I)}$ *so that*

$$adm_A(\sigma \times \tau) = adm_A(\sigma \times \tau') = False$$

holds true. Furthermore, let A *be a square* $\mathcal{H}$*-matrix and with symmetric structure, meaning* $I = J$ *and*

$$adm_A(\sigma \times \sigma') = adm_A(\sigma' \times \sigma)$$

for all $\sigma, \sigma' \in P_I^{hier}$ with $\sigma \times \sigma' \in P_{I\times I}^{hier}$. Then the admissibility function a as defined above is column-wise oriented.

Proof Let $\sigma_1, \sigma_2, \sigma' \in P_J^{hier}$ with $\sigma_1 \times \sigma', \sigma_2 \times \sigma' \in P_{I\times J}^{hier}$ and $\sigma_1 < \sigma_2$. We can also assume $\sigma_2 < \sigma'$ because, for now, only the upper triangular part is relevant. Let the block columns induced by σ_1 and σ_2 be neighbors in A and

$$a(\sigma_1 \times \sigma') = \prod_{\hat{\sigma} \in P_I^{Lmax(\sigma_1, \sigma', I)}} \mathrm{adm}_Y(\hat{\sigma} \times \sigma_1) + \mathrm{adm}_A(\hat{\sigma} \times \sigma') = \mathrm{False}.$$

For the last equation to hold, there has to exist $\widetilde{\sigma} \in P_I^{hier}$ so that

$$\mathrm{adm}_Y(\widetilde{\sigma} \times \sigma_1) = \mathrm{False}$$

and

$$\mathrm{adm}_A(\widetilde{\sigma} \times \sigma') = \mathrm{False}.$$

We know that $\widetilde{\sigma} \geq \sigma_1$ because otherwise $\widetilde{\sigma} \times \sigma_1$ is in the upper triangular part of Y, which consists only of zeros and thus

$$\mathrm{adm}_Y(\widetilde{\sigma} \times \sigma') = \mathrm{True}.$$

We want to show that

$$a(\sigma_2 \times \sigma') = \mathrm{False}$$

holds because that would imply the column-wise orientation of a. If $\widetilde{\sigma} = \sigma_1$, then due to the structural symmetry of A we have

$$\mathrm{adm}_A(\sigma_1 \times \sigma') = \mathrm{False} = \mathrm{adm}_A(\sigma' \times \sigma_1),$$

which also means that

$$\mathrm{adm}_Y(\sigma' \times \sigma_1) = \mathrm{False}.$$

We know with Theorem 3.14 that Y is row-wise oriented, and thus with $\sigma_2 > \sigma_1$ we have

$$\mathrm{adm}_Y(\sigma' \times \sigma_2) = \mathrm{False}$$

as well. Combined, we have

$$\begin{aligned} a(\sigma_2 \times \sigma') &= \prod_{\hat{\sigma} \in P_I^{Lmax(\sigma_2, \sigma', I)}} \mathrm{adm}_Y(\hat{\sigma} \times \sigma_2) + (\mathrm{adm}_A(\hat{\sigma} \times \sigma') \\ &\leq \mathrm{adm}_Y(\sigma' \times \sigma_2) + \mathrm{adm}_A(\sigma' \times \sigma') \\ &= \mathrm{False} + \mathrm{False} = \mathrm{False}. \end{aligned}$$

If $\widetilde{\sigma} > \sigma_1$, then we have $\widetilde{\sigma} \geq \sigma_2$ and $\widetilde{\sigma} \times \sigma_2$ is a non-zero block in Y because it is in the lower triangular part. Because the admissibility function of Y is row-wise oriented with Theorem 3.14 $\mathrm{adm}_Y(\widetilde{\sigma} \times \sigma_1) = \mathrm{False}$ also implies

$$\mathrm{adm}_Y(\widetilde{\sigma} \times \sigma_2) = \mathrm{False},$$

and thus

$$\begin{aligned} a(\sigma_2 \times \sigma') &= \prod_{\hat{\sigma} \in P^{L^{max}(\sigma_2, \sigma', I)}} \mathrm{adm}_Y(\hat{\sigma} \times \sigma_2) + \mathrm{adm}_A(\hat{\sigma} \times \sigma') \\ &\leq \mathrm{adm}_Y(\widetilde{\sigma} \times \sigma_2) + \mathrm{adm}_A(\widetilde{\sigma} \times \sigma') = \mathrm{False} + \mathrm{False} = \mathrm{False}. \end{aligned}$$

Assume now that $\widetilde{\sigma} < \sigma_1$. Then by the symmetry of A, we have

$$\mathrm{adm}_A(\sigma' \times \widetilde{\sigma}) = \mathrm{False}.$$

Hence

$$\mathrm{adm}_Y(\sigma' \times \widetilde{\sigma}) = \mathrm{False}$$

as well and by the row-wise orientation of Y

$$\begin{aligned} \mathrm{adm}_Y(\sigma' \times \sigma_1) &= \mathrm{False}, \\ \mathrm{adm}_Y(\sigma' \times \sigma_2) &= \mathrm{False} \end{aligned}$$

because, due to $\sigma' > \sigma_2 > \sigma_1$, both $\sigma' \times \sigma_1$ and $\sigma' \times \sigma_2$ are non-zero blocks in Y. Because diagonal blocks are inadmissible, we have

$$\begin{aligned} a(\tau_2 \times \tau') &= \prod_{\hat{\sigma} \in P^{L(\sigma_2, \sigma')}} \mathrm{adm}_Y(\hat{\sigma} \times \sigma_2) + \mathrm{adm}_A(\hat{\sigma} \times \sigma') \\ &\leq \mathrm{adm}_Y(\sigma' \times \sigma_2) + \mathrm{adm}_A(\sigma' \times \sigma') = \mathrm{False} + \mathrm{False} = \mathrm{False}. \end{aligned}$$

By using this argument repeatedly, we can show that

$$a(\tau \times \tau') = a^{cw}(\tau \times \tau')$$

holds. ∎

To summarise, we have found conditions under which the partially induced admissibility functions for Y, R and the agglomeration matrices S_1^{comb}, S_2^{comb} are fully induced. The admissibility function for $\overline{T}$ from Subsection 3.3.1 was already fully induced. We will make one short excursus about the admissibility function of the explicit orthogonal factor Q, and then we will see how we can use our results from this section to bound the cost of the block Householder approach to the QR decomposition of $\mathcal{H}$-matrices.

3.3.3 Induced Admissibility Function of Q

We will now make a statement regarding the structure of Q computed by the operation $Q = Y - YTY^T$. Although not directly computed in our algorithm, it serves as a benchmark for the efficiency of our storage scheme. We need one additional assumption that is generally satisfied for $\mathcal{H}$-matrices due to their construction to make the notation easier to follow. We assume that we can find a sequence of inadmissible blocks in the $\mathcal{H}$-matrix A from the top left to the bottom right, which we will call a streak. For a square matrix A, this could simply correspond to the blocks along the diagonal. We formalize this in the following assumption.

Assumption D Let $A \in \mathcal{H}(P_{I\times J}^{\mathcal{H},A}, k_A, \mathrm{adm}_A)$ be an $\mathcal{H}$-matrix. Furthermore, let the indices of the index sets $\sigma \in P_I^\ell$ and $\tau \in P_J^k$ for $0 \leq \ell \leq L_I$ and $0 \leq k \leq L_J$ reflect the order from Definition 2.26, meaning that for $\sigma_m, \sigma_n \in P_I^\ell$, we have

$$\sigma_m <_{(A,row)} \sigma_n \;\Leftrightarrow\; m < n,$$

and for $\tau_m, \tau_n \in P_J^k$, we have

$$\tau_m <_{(A,col)} \tau_n \;\Leftrightarrow\; m < n.$$

Then there is a set $P_{I\times J}^{streak} \subset P_{I\times J}^{L_{IJ}}$ that satisfies the following conditions:

1. Every block in $P_{I\times J}^{streak}$ is inadmissible.

2. Let τ, τ' denote the first and last cluster of $P_J^{L_J}$, and let σ, σ' denote the first and last cluster of $P_I^{L_I}$ both according to the order defined in Definition 2.26. Then we have $\sigma \times \tau, \sigma' \times \tau' \in P_{I\times J}^{streak}$.

3. Let $\sigma_i \times \tau_j \in P_{I\times J}^{streak}$. Then one of the following conditions is satisfied:

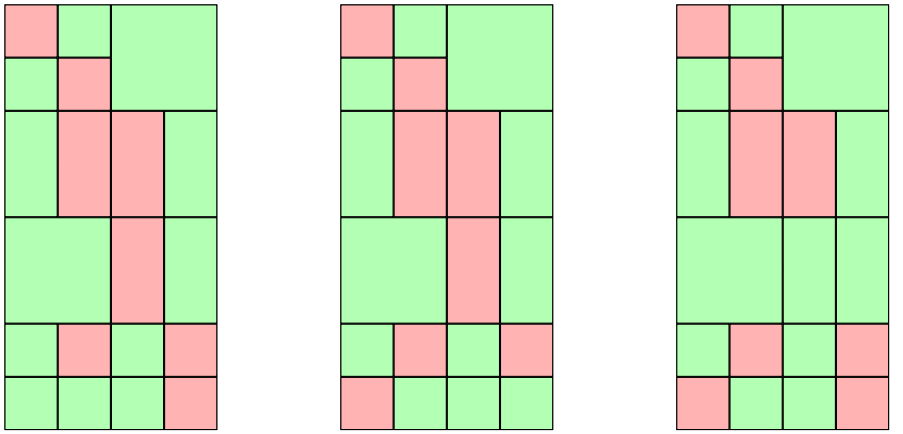
   a) τ_j is the first cluster of all clusters in $P_J^{L_J}$, and either $\sigma_{i+1} \times \tau_j \in P_{I\times J}^{streak}$ or $\sigma_{i+1} \times \tau_{j+1} \in P_{I\times J}^{streak}$.

   b) τ_j is the last cluster of all clusters in $P_J^{L_J}$, and either $\sigma_{i-1} \times \tau_j \in P_{I\times J}^{streak}$ or $\sigma_{i-1} \times \tau_{j-1} \in P_{I\times J}^{streak}$.

   c) Both of the following conditions are satisfied:

      i. $\sigma_{i+1} \times \tau_j \in P_{I\times J}^{streak}$ or $\sigma_{i+1} \times \tau_{j+1} \in P_{I\times J}^{streak}$,

ii. $\sigma_{i-1} \times \tau_j \in P_{I\times J}^{streak}$ or $\sigma_{i-1} \times \tau_{j-1} \in P_{I\times J}^{streak}$.



(a) Assumption D satisfied.

(b) Assumption D not satisfied.

(c) Assumption D not satisfied.

Figure 3.6. $\mathcal{H}$-Matrix examples to illustrate Assumption D.

We need one last definition.

Definition 3.17 *Let us have the situation of Definition 2.26, meaning $P_{I\times J}^{\mathcal{H}}$ is an $\mathcal{H}$-block partition based on the hierarchy of block partitionings $\left(P_{I\times J}^{\ell}\right)_{\ell=0}^{L}$, where $(P_I^{\ell})_{\ell=0}^{L_I}$ is the hierarchy of partitionings of the row index set, $(P_J^{k})_{\ell=0}^{L_J}$ is the hierarchy of partitionings of the column index set, and $A \in \mathcal{H}(P_{I\times J}^{\mathcal{H}}, k, adm_A)$ is an $\mathcal{H}$-matrix. Furthermore, let the index sets I and J correspond to ordered rows and columns in A, meaning that the i-th row of A is induced by the index i, and the j-th column of A is induced by j.*

Let $\sigma \in P_I^{hier}$, and let $\tau_i \in P_J^{\ell}$ be the last cluster on level ℓ according to the (index-based) order from Definition 2.26 inducing a non-zero block $\sigma \times \tau_i$. Then we set

$$L_{final}^{\ell}(\sigma, A) = i.$$

We use this to define a function that associates a cluster τ_k with the index of the last non-zero block $L_{final}^{\ell}(\sigma, A)$ if $k > L_{final}^{\ell}(\sigma, A)$ (τ_k is a zero block) and with k if $k \leq L_{final}^{\ell}(\sigma, A)$ (τ_k is a non-zero block). For that, we set

$$L_{floor}(\sigma, \tau_k, A) = \begin{cases} k & \sigma \times \tau_k \text{ is non-zero block in } A, \\ L_{final}^{\ell}(\sigma, A) & \sigma \times \tau_k \text{ is zero block in } A. \end{cases}$$

See Figure 3.7 for a visualization of this definition. We are now able to derive the (fully) induced admissibility function of Q.

Theorem 3.18 *Let Prerequisites A be true. Assume that all blocks (on every level) in the upper triangular part of T are inadmissible, Y is row-wise oriented, and Assumption D is satisfied for A. Then the admissibility function of Q, induced by the multiplication $Q = I - YTY^T$, is given by*

$$adm_Q(\sigma_1 \times \sigma_2) = \begin{cases} adm_Y(\sigma_1 \times \tau_{L^{\ell}_{final}(\sigma_2,Y)}) & L^{\ell}_{final}(\sigma_1, Y) > L^{\ell}_{final}(\sigma_2, Y), \\ False & L^{\ell}_{final}(\sigma_1, Y) \leq L^{\ell}_{final}(\sigma_2, Y) \end{cases}$$

for all $\sigma_1 \times \sigma_2 \in P^{hier}_{I\times I}$, where $\ell = L^{intersec}(\sigma_1, \sigma_2, J)$.

Proof Let adm_{YT} be the induced admissibility function of the product $Y \cdot T$ on $(P^{\ell}_{I\times J})_{\ell=0}^{L_{IJ}}$. Let $\tau_i, \tau_j \in P^{hier}_J$, and let the indices i, j be denoted according to the order given in Definition 2.26. Because T is upper triangular, it holds that

$$i > j \Rightarrow \ \mathrm{adm}_T(\tau_i \times \tau_j) = \mathrm{True},$$

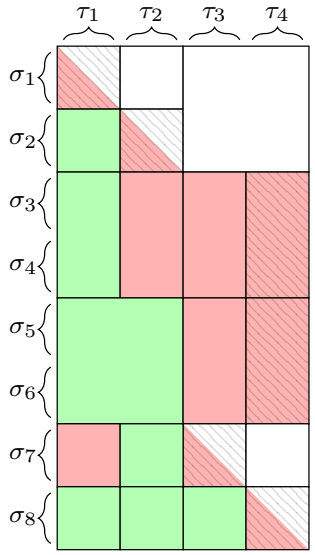
and, because all blocks (on every level) in the upper triangular part of T are inadmissible, $\mathrm{adm}_T(\tau_i, \tau_i) = \mathrm{False}$. Furthermore by Assumption D, we know that every block row of A includes at least one inadmissible block in its lower trapezoidal part. Thus, every block row of Y also contains an inadmissible block, and the last block of every row in Y must be inadmissible because Y is row-wise oriented. That means that for any $\sigma \times \tau \in P^{hier}_{I\times J}$

$$\mathrm{adm}_Y(\sigma \times \tau_{L^{\mathrm{level}(\tau)}_{\mathrm{final}}(\sigma,Y)}) = \mathrm{False}.$$

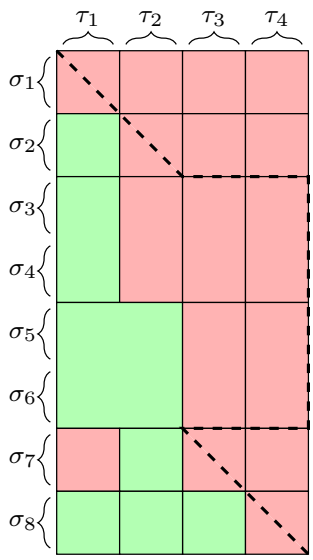
holds true. Hence, for any $\sigma \times \tau_i \in P^{hier}_{I\times J}$ with $L_{floor}(\sigma, \tau_i, Y) < i$ we have

$$\mathrm{adm}_Y\left(\sigma \times \tau_{L_{floor}(\sigma,\tau_i,Y)}\right) = \mathrm{adm}_Y\left(\sigma \times \tau_{L^{\mathrm{level}(\tau_i)}_{final}(\sigma,Y)}\right) = \mathrm{False}.$$

Applying these results to the induced admissibility function of the product $Y \cdot T$ (as given by Theorem 2.31) and using the fact that all blocks in the upper triangular part



(a) The factor Y of our example matrix with adm_Y. The block $\sigma_i \times \tau_{L_{final}(\sigma_i)}$ is marked for every row cluster $\sigma_i \in P_I^{L_I}$.



(b) The $\mathcal{H}$-matrix Y after the admissibility of $\sigma_i \times \tau_j$ was replaced by $\sigma_i \times \tau_{L_{floor}(\sigma_i,\tau_j,Y)}$.

Figure 3.7. Comparison of the admissibility function adm_Y and the admissibility function of the bound for adm_{YT}.

of T are inadmissible as well as the row-wise orientation of Y, we get for a block $\sigma \times \tau_j$

$$
\begin{aligned}
\mathrm{adm}_{YT}(\sigma \times \tau_j) &= \prod_{0 \le i \le L_{final}^{\mathrm{level}(\tau_j)}(\sigma,Y)} \left(\mathrm{adm}_Y(\sigma \times \tau_i) + \mathrm{adm}_T(\tau_i \times \tau_j)\right) \\
&= \prod_{0 \le i \le j} \left(\mathrm{adm}_Y(\sigma \times \tau_i) + \mathrm{adm}_T(\tau_i \times \tau_j)\right) \\
&= \begin{cases} \mathrm{adm}_Y(\sigma \times \tau_j) + \mathrm{adm}_T(\tau_j \times \tau_j) & L_{final}^{\mathrm{level}(\tau_j)}(\sigma,Y) \ge j \\ \mathrm{adm}_Y\left(\sigma \times \tau_{L_{final}^{\mathrm{level}(\tau_j)}(\sigma,Y)}\right) & \\ \quad + \mathrm{adm}_T\left(\tau_{L_{final}^{\mathrm{level}(\tau_j)}(\sigma,Y)} \times \tau_j\right) & L_{final}^{\mathrm{level}(\tau_j)}(\sigma,Y) < j \end{cases} \\
&= \begin{cases} \mathrm{adm}_Y(\sigma \times \tau_j) + \mathrm{False} & L_{floor}(\sigma,\tau_j,Y) = j \\ \mathrm{False} + \mathrm{False} & L_{floor}(\sigma,\tau_j,Y) < j \end{cases} \\
&= \begin{cases} \mathrm{adm}_Y(\sigma \times \tau_j) & L_{floor}(\sigma,\tau_j,Y) = j \\ \mathrm{adm}_Y(\sigma \times \tau_{L_{floor}(\sigma,\tau_j,Y)}) & L_{floor}(\sigma,\tau_j,Y) < j \end{cases} \\
&= \mathrm{adm}_Y\left(\sigma \times \tau_{L_{floor}(\sigma,\tau_j,Y)}\right).
\end{aligned}
$$

We now want to find the admissibility function of $Q = I - YTY^T$. We ignore the identity matrix at first, and only check YTY^T. Let adm_{YTY^T} denote the corresponding induced admissibility function on $(P^\ell_{I\times I})_{\ell=0}^{L_{II}}$. Note that the sequence in which we multiply the three factors Y, T and Y^T does not influence the induced admissibility function of the result. Set

$$L_1 := L_{\text{final}}^{L^{max}(\sigma_1,\sigma_2,J)}(\sigma_1, Y),$$
$$L_2 := L_{\text{final}}^{L^{max}(\sigma_1,\sigma_2,J)}(\sigma_2, Y).$$

Let $\sigma_1 \times \sigma_2 \in P^{hier}_{I\times I}$ induce a block in YTY^T. In the following, we use the product $(Y \cdot T) \cdot Y^T$, and again apply Theorem 2.31 to describe the product admissibility. Note that L_2 denotes the index of the last non-zero block in the block row of Y that is induced by σ_2 on the level that is used for the multiplication of $Y \cdot T$ with Y^T. The induced admissibility function of YTY^T is hence given by

$$\begin{aligned}
\mathrm{adm}_{YTY^T}(\sigma_1 \times \sigma_2) &= \prod_{0\le i\le L_2} \left(\mathrm{adm}_{YT}(\sigma_1 \times \tau_i) + \mathrm{adm}_Y(\sigma_2 \times \tau_i)\right) \\
&= \prod_{0\le i\le L_2} \left(\mathrm{adm}_Y(\sigma_1 \times \tau_{L_{floor}(\sigma_1,\tau_i,Y)}) + \mathrm{adm}_Y(\sigma_2 \times \tau_i)\right) \\
&= \begin{cases} \prod_{0\le i\le L_2} \left(\mathrm{adm}_Y(\sigma_1 \times \tau_i) + \mathrm{adm}_Y(\sigma_2 \times \tau_i)\right) & L_1 > L_2 \\ \prod_{0\le i\le L_2} \left(\mathrm{adm}_Y(\sigma_1 \times \tau_{L_{floor}(\sigma_1,\tau_i,Y)}) + \mathrm{adm}_Y(\sigma_2 \times \tau_i)\right) & L_1 \le L_2 \end{cases} \\
&= \begin{cases} \mathrm{adm}_Y(\sigma_1 \times \tau_{L_2}) + \mathrm{adm}_Y(\sigma_2 \times \tau_{L_2}) & L_1 > L_2 \\ \text{False} + \mathrm{adm}_Y(\sigma_2 \times \tau_{L_2}) & L_1 \le L_2 \end{cases} \\
&= \begin{cases} \mathrm{adm}_Y(\sigma_1 \times \tau_{L_2}) + \text{False} & L_1 > L_2 \\ \text{False} + \text{False} & L_1 \le L_2 \end{cases} \\
&= \begin{cases} \mathrm{adm}_Y(\sigma_1 \times \tau_{L_2}) & L_1 > L_2 \\ \text{False} & L_1 \le L_2. \end{cases}
\end{aligned}$$

The third equation follows from the fact that with

$$L_1 = L_{\text{final}}^{L^{max}(\sigma_1,\sigma_2,J)}(\sigma_1, Y) > L_{\text{final}}^{L^{max}(\sigma_1,\sigma_2,J)}(\sigma_2, Y) = L_2$$

we have

$$L_{floor}(\sigma_1, \tau_i, Y) = i$$

for every factor in the product. After that, we use the row-wise orientation of Y to reduce the product to its last factor and use the fact that, as already mentioned above, the last non-zero block in every row of Y is inadmissible. Because $\mathrm{adm}_{YTY^T}(\sigma \times \sigma) = \text{False}$ for

all $\sigma \in P_I^{hier}$, we have

$$\mathrm{adm}_Q = \mathrm{adm}_{YTY^T}$$

and thus, finally, the statement. ∎

The conditions of this theorem are satisfied for many $\mathcal{H}$-matrices, including those for which the conditions mentioned in Theorem 3.12 and 3.14 are true, if Assumption D holds.

But what does this result mean for the admissibility of Q? Assume for simplification that A is square, and $P_{I\times J}^{streak}$ consists of the diagonal blocks on every level. Then $L_{final}^{\ell}(\sigma_i, A) = i$ holds for all i on every level, and our result shows that the entire upper triangular part of Q is inadmissible. Hence, the storage cost of such a Q as an $\mathcal{H}$-matrix using the induced admissibility function is of the same complexity as the dense version of Q. For rectangular matrices the result is similar, although not as easy to see.

Compared to the $\mathcal{H}$-matrix structures of the $\mathcal{H}$-matrices Y, R and $\overline{T}$ that result from the block Householder approach to the QR decomposition, in which we completely circumvent the calculation of Q, the $\mathcal{H}$-matrix structure of Q suffers from more serious problems regarding fill-in. The conditions from Theorem 3.18 are easily satisfied, for example in the simple (square) numerical example in Subsection 4.1.3, where using the induced admissibility function for Q leads to a comparatively bad performance of algorithms that make use of Q and adapt the $\mathcal{H}$-matrix structure during computation. This is different for Y, R and $\overline{T}$, where fill-in is also of concern, but - as Theorems 3.9, 3.14, 3.15, and especially the results in the upcoming section show - there is a wider range of (starting) $\mathcal{H}$-matrices A that do not lead to these problems.

3.3.4 Parameter Bounds

We can now bound the bandwidth - and thus indirectly the sparsity constant C_{sp} - of all $\mathcal{H}$-matrices used during the block Householder based algorithm for the QR decomposition by the bandwidth of Y.

Theorem 3.19 *Let Prerequisites A be satisfied. Define an admissibility function on* $P_{J\times J}^{hier}$ *by*

$$a(\tau \times \tau') = \prod_{\sigma \in P_I^{L^{max}(\tau,\tau',I)}} adm_Y(\sigma \times \tau) + adm_A(\sigma \times \tau')$$

for all $\tau \times \tau' \in P_{J\times J}^{hier}$. *Furthermore, using the extended admissibility function as given*

in Definition 3.6, define

$$s(\tau \times \tau') = adm_A^{ext}(\sigma(\tau) \times \tau').$$

Then it holds that

$$C_{bw}(s, R) \leq C_{bw}(adm_A, A),$$

where C_{bw} is the bandwidth constant (see Definition 2.29).

Proof We show the statement first for the rows and then for the columns:

1. Let $\tau, \tau', \tau'' \in P_J^{hier}$ be given with

   $$\text{level}(\tau) = \text{level}(\tau') = \text{level}(\tau'')$$

   and

   $$s(\tau \times \tau') = s(\tau \times \tau'') = \text{False}.$$

   We show that there are two inadmissible blocks in one row of A with the same distance to each other as $\tau \times \tau'$ and $\tau \times \tau''$. With that the $\mathcal{H}$-matrix bandwidth of R can only be as large as the $\mathcal{H}$-matrix bandwidth of A.

   If $\sigma(\tau) \in P_I^{hier}$ (this not necessarily the case because of how $\sigma(\tau)$ is defined, see Definition 3.6), then by definition of adm_A^{ext} we have

   $$\text{adm}_A(\sigma(\tau) \times \tau') = \text{adm}_A(\sigma(\tau) \times \tau'') = \text{False}.$$

   Because all blocks $\check{\tau} \in P_J^{hier}$ with $\text{level}(\check{\tau}) = \tau'$ between τ' and τ'' also induce blocks $\sigma(\tau) \times \check{\tau}$ in A and the columns in R are ordered the same as in A, we have

   $$\text{dist}_{row;R}(\tau', \tau'') = \text{dist}_{row;A}(\tau', \tau'').$$

   So let $\sigma(\tau) \notin P_I^{hier}$. There is no $\widetilde{\sigma}$ with $\sigma \subset \widetilde{\sigma}$ and

   $$\text{adm}_A(\widetilde{\sigma} \times \tau') = \text{True}$$

   because in that case

   $$s(\sigma(\tau) \times \tau') = \text{True}$$

   would hold as well. Hence, for all $\widetilde{\sigma} \in P_I^{hier}$ with $\widetilde{\sigma} \times \tau \in P_{I \times J}^{hier}$ and $\sigma(\tau) \subset \widetilde{\sigma}$, we have

   $$\text{adm}_A(\widetilde{\sigma} \times \tau') = \text{False}.$$

Now choose $\widetilde{\sigma} \in P_I^{hier}$ with the highest possible level. Then similar to before, all blocks $\check{\tau}$ between τ' and τ'' induce blocks $\widetilde{\sigma} \times \check{\tau}$ in A. Furthermore, by definition, every block $\sigma(\tau) \times \check{\tau}$ is contained in one distinct block $\widetilde{\sigma} \times \check{\tau}$. Hence, we have

$$\mathrm{dist}_{R,row}(\tau \times \tau', \tau \times \tau'') = \mathrm{dist}_{A,row}(\widetilde{\sigma} \times \tau', \widetilde{\sigma} \times \tau'')$$

in this case as well, and thus

$$C_{bw,row}(s, R) = C_{bw}(\mathrm{adm}_A, A)$$

holds.

2. Let $\tau, \tau', \tau'' \in P_J^{hier}$ with

$$\mathrm{level}(\tau) = \mathrm{level}(\tau') = \mathrm{level}(\tau'').$$

Let $\tau' < \tau''$ and

$$s(\tau' \times \tau) = s(\tau'' \times \tau) = \mathrm{False}.$$

If $\sigma(\tau'), \sigma(\tau'') \in P_I^{hier}$, then we have

$$\mathrm{adm}_A(\sigma(\tau') \times \tau) = \mathrm{adm}_A(\sigma(\tau'') \times \tau) = \mathrm{False},$$

and thus with the same arguments as in 1 we get

$$\mathrm{dist}_{R,col}(\tau' \times \tau, \tau'' \times \tau) = \mathrm{dist}_{A,col}(\sigma' \times \tau, \sigma'' \times \tau).$$

So let $\sigma(\tau'), \sigma(\tau'') \notin P_I^{hier}$. Then with the arguments in 1, there are uniquely defined $\widetilde{\sigma}', \widetilde{\sigma}'' \in P_I^{hier}$ of highest level with $\widetilde{\sigma}' \times \tau, \widetilde{\sigma}'' \times \tau \in P_{I\times J}^{hier}$ and

$$\mathrm{adm}_A(\widetilde{\sigma}', \tau) = \mathrm{False}$$
$$\mathrm{adm}_A(\widetilde{\sigma}'', \tau) = \mathrm{False}.$$

We can apply this argument to all blocks $\check{\tau}$ with $\mathrm{level}(\check{\tau}) = \tau'$ between τ' and τ'' as well. Hence, for every $\check{\tau}$ we can find a uniquely defined $\widetilde{\check{\sigma}}$ between $\widetilde{\sigma}'$ and $\widetilde{\sigma}''$ because the order of the columns in A and R is the same. Thus,

$$\mathrm{dist}_{R,col}(\tau' \times \tau, \tau'' \times \tau) = \mathrm{dist}_{A,col}(\sigma' \times \tau, \sigma'' \times \tau)$$

holds, and we have

$$C_{bw,col}(s, R) = C_{bw}(\mathrm{adm}_A, A).$$

Combining both the row-wise and column-wise results, we get the statement. ∎

Theorem 3.20 *Let Prerequisites A be true. Define an admissibility function on $P_{J\times J}^{hier}$ by*

$$a(\tau\times\tau') = \prod_{\sigma\in P_I^{L^{max}(\tau,\tau',I)}} adm_Y(\sigma\times\tau) + adm_A(\sigma\times\tau')$$

for all $\tau\times\tau'\in P_{J\times J}^{hier}$. Then it holds that

1. $C_{bw}(adm_{\overline{T}},\overline{T}) \le C_{bw}(adm_Y, Y)$,
2. $C_{bw}(adm_R, R) \le \max\{C_{bw}(adm_Y, Y), C_{bw}(adm_A, A)\}$,
3. $C_{bw}(adm_{S_1^{comb}}, S_1^{comb}) = C_{bw}(adm_{S_2^{comb}}, S_2^{comb}) \le C_{bw}(adm_R, R)$.

Proof 1. With Theorem 3.9 we have for any $\tau\times\tau'\in P_{J\times J}^{hier}$ with $\tau,\tau'\in P_J^{hier}$

$$\mathrm{adm}_{\overline{T}}(\tau\times\tau') = \prod_{\sigma\in P_I^{L^{max}(\tau,\tau',I)}} \mathrm{adm}_Y(\sigma\times\tau)\cdot\mathrm{adm}_Y(\sigma\times\tau'),$$

and thus

$$\begin{aligned}\mathrm{adm}_{\overline{T}}(\tau\times\tau') = \text{False} \;\Leftrightarrow\; &\exists\,\sigma\in P_I^{L^{max}(\tau,\tau',I)} : \mathrm{adm}_Y(\sigma\times\tau) = \mathrm{adm}_Y(\sigma\times\tau') = \text{False}\\ &\Rightarrow \mathrm{dist}_Y(\tau,\tau') + 2 \le C_{bw}(\mathrm{adm}_Y, Y).\end{aligned}$$

Hence, we have

$$C_{bw}(\mathrm{adm}_T, T) \le C_{bw}(\mathrm{adm}_Y, Y).$$

2. We want to review both factors of the lower bound for the admissibility function given by

$$\mathrm{adm}_R(\tau\times\tau') \ge \mathrm{adm}_A^{ext}(\sigma(\tau)\times\tau')\cdot a^{cw}(\tau\times\tau')$$

independently and then combine both bandwidths. For that to work, we need a block on every level of $P_{J\times J}^{hier}$ so that both factors are inadmissible. We define

$$s(\tau\times\tau') = \mathrm{adm}_A^{ext}(\sigma(\tau)\times\tau')$$

as before. We have

$$s(\tau\times\tau) = \mathrm{adm}_A^{ext}(\sigma(\tau)\times\tau) = \text{False}$$

for all $\tau\in P_J^{hier}$ because $\sigma(\tau)$ is chosen to be the row index block so that $\sigma(\tau)\times\tau$ is the

diagonal block in R and thus inadmissible. Furthermore, it holds that

$$\begin{aligned} a(\tau \times \tau) &= \prod_{\sigma \in P_I^{L^{max}(\tau,\tau',I)}} \mathrm{adm}_Y(\sigma \times \tau) + \mathrm{adm}_A(\sigma \times \tau) \\ &\leq \mathrm{adm}_Y(\sigma(\tau) \times \tau) + \mathrm{adm}_A(\sigma(\tau) \times \tau) \\ &= \mathrm{False} + \mathrm{False} = \mathrm{False} \end{aligned}$$

with the same argument. Since $\tau \times \tau$ denotes by construction the leftmost inadmissible block in R and

$$C_{bw}(a^{cw}, A) = C_{bw}(a, A)$$

because outer blocks in a^{cw} are inadmissible in a by definition, we have

$$C_{bw}(\mathrm{adm}_R, R) \leq \max\{C_{bw}(a, R), C_{bw}(\mathrm{adm}_A^{ext}, R)\}.$$

By definition of a, we have for all $\tau \times \tau' \in P_J^{hier}$ with $\tau, \tau \in P_J^{hier}$

$$a(\tau \times \tau') = \prod_{\sigma \in P_I^{L^{max}(\tau,\tau',I)}} \mathrm{adm}_Y(\sigma \times \tau) \cdot \mathrm{adm}_A(\sigma \times \tau'),$$

and thus

$$a(\tau \times \tau') = \mathrm{False} \;\Rightarrow\; \exists\, \hat{\sigma} \in P_I^{L^{max}(\tau,\tau',I)} :\; \mathrm{adm}_Y(\hat{\sigma} \times \tau) = \mathrm{adm}_A(\hat{\sigma} \times \tau') = \mathrm{False}.$$

Furthermore, we have

$$\mathrm{adm}_A(\hat{\sigma} \times \tau') = \mathrm{False} \;\Rightarrow\; \mathrm{adm}_Y(\hat{\sigma} \times \tau') = \mathrm{False}$$

for all $\hat{\sigma} \times \tau' \in P_{J \times J}^{hier}$, and thus, for a block $\tau \times \tau' \in P_{J \times J}^{hier}$ that is inadmissible in a, it holds that

$$\mathrm{dist}_Y(\tau, \tau') \leq C_{bw}(\mathrm{adm}_Y, Y).$$

Since these statements are true for any block $\tau \times \tau' \in P_{J \times J}^{hier}$, we have

$$C_{bw}(a, R) \leq C_{bw}(\mathrm{adm}_Y, Y).$$

Let $\tau \times \tau' \in P_{J \times J}^{hier}$ with $\tau, \tau' \in P_J^{hier}$. For the second factor, we apply Theorem 3.19 and get

$$C_{bw}(s, R) \leq C_{bw}(\mathrm{adm}_A, A).$$

Combining both results leads to

$$\begin{aligned} C_{bw}(\mathrm{adm}_R, R) &\leq \max\{C_{bw}(a, R), C_{bw}(s, R)\} \\ &\leq \max\{C_{bw}(\mathrm{adm}_Y, Y), C_{bw}(\mathrm{adm}_A, A)\}. \end{aligned}$$

3. By using Theorem 3.11, every inadmissible block in S_1^{comb} and S_2^{comb} is also inadmissible in R. Due to that, the bandwidth of S_1^{comb} and S_2^{comb} can be at most the bandwidth of R. ∎

For a square matrix A, we can also bound the bandwidth of Y with the bandwidth of A.

Theorem 3.21 *Let Prerequisites A be satisfied. Let A be a square matrix with $I = J$. Let $C_{bw}(adm_A, A)$ be the bandwidth of A and $C_{bw}(adm_Y, Y)$ the bandwidth of Y. Then we have*

$$C_{bw}(adm_Y, Y) \leq C_{bw}(adm_A, A).$$

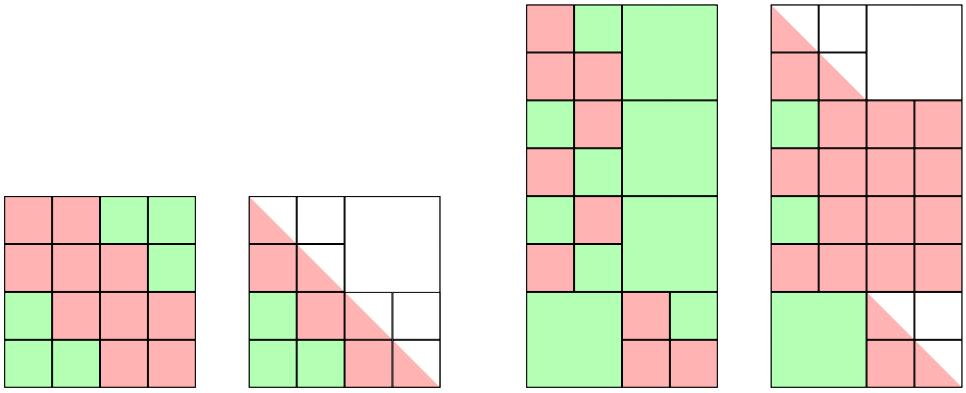
Proof The bandwidth constant of Y is given by the largest distance between an outer block (see Definition 2.27) and the last non-zero block in the corresponding row, which for square matrices has to be inadmissible in Y because it is or contains a lower triangular matrix. Both blocks are also inadmissible in A. The outer block is admissible by construction of Y, as Y has the same $\mathcal{H}$-matrix structure in its non-zero parts as A after all updates were applied. The admissibility of the last non-zero block follows directly from Assumption C because A is square. Hence, the bandwidth constant of Y is identical to or smaller than the bandwidth constant of A. ∎

Thus, in the square case, we can use the bandwidth constant of A instead of Y to bound the remaining bandwidth constants.

For a rectangular matrix A, where a permuted QR decomposition is necessary, the last block of Y in a given row is not necessarily a lower triangular - and thus inadmissible - block because the entire row can be a part of Y. Hence, the bandwidth of A cannot serve as a bound here. With Theorem 3.14, we know that the complete fill-in of the row with inadmissible blocks happens under relatively weak conditions.

3.3.5 Cost Estimate for the $\mathcal{H}$-QR Decomposition

We now have all the results we need to find a cost estimate for the entire block Householder QR decomposition for $\mathcal{H}$-matrices. As stated in Assumption B, we assume that



(a) (b) (c) (d)

Figure 3.8. Difference in the number of inadmissible blocks in Y for a square matrix in (a), corresponding factor Y in (b) and a rectangular matrix in (c), corresponding factor Y in (d).

in the low-rank factorization of all admissible blocks, the first factor has orthogonal columns. This is, depending on its creation, generally not true for a given $\mathcal{H}$-matrix. The following theorem, however, proves that such a factorization can be efficiently computed. We will call the associated process *left orthogonalization*.

Theorem 3.22 (Left orthogonalization) *Let $A \in \mathcal{H}(P^{\mathcal{H}}_{I\times J}, k, adm_A)$ be an $\mathcal{H}$-matrix with $\mathcal{H}$-block partition $P^{\mathcal{H}}_{I\times J}$. Then we can compute a left orthogonal low-rank representation of A, meaning, for every admissible block $A_{\sigma\times\tau}$ given by a low-rank factorization*

$$A_{\sigma\times\tau} = UV^T \in \mathbb{R}^{\sigma\times\tau} \quad \textit{with} \quad U \in \mathbb{R}^{\sigma\times k},\ V \in \mathbb{R}^{\tau\times k},$$

we find another low-rank factorization

$$A_{\sigma\times\tau} = \widetilde{U}\widetilde{V}^T \in \mathbb{R}^{\sigma\times\tau} \quad \textit{with} \quad \widetilde{U} \in \mathbb{R}^{\sigma\times k},\ \widetilde{V} \in \mathbb{R}^{\tau\times k},$$

where $\widetilde{U}$ has orthogonal columns, with a computational cost of at most

$$N_{lorth}(A) \leq 5k \cdot S_{\mathcal{H}}\left(P^{\ell}_{I\times J}, k\right),$$

where $S_{\mathcal{H}}\left(P^{\ell}_{I\times J}, k\right)$ denotes the storage cost of A as given in Theorem 2.40.

Proof To find the desired low-rank factorization for a given admissible block $A_{\sigma\times\tau} = UV^T \in \mathbb{R}^{\sigma\times\tau}$ with $U \in \mathbb{R}^{\sigma\times k}$ and $V \in \mathbb{R}^{\tau\times k}$, we first compute a QR decomposition of U given by

$$U = QR \quad \text{with} \quad Q \in \mathbb{R}^{\sigma\times k}, R \in \mathbb{R}^{k\times k}.$$

The computational cost for this step is bounded by $4\#\sigma k^2$ operations. We set $\widetilde{U} = Q$ and compute $\widetilde{V} \in \mathbb{R}^{\tau \times k}$ with

$$\widetilde{V} = R \cdot V,$$

which has a computational cost of at most $k^2 \cdot \#\tau$. Together the overall cost for all admissible blocks $\sigma \times \tau \in P^{\mathcal{H},+}_{I\times J}$ is bounded by

$$5k^2 \sum_{\sigma\times\tau\in P^{\mathcal{H},+}_{I\times J}} (\#\tau + \#\sigma).$$

Then our statement is true because

$$k \sum_{\sigma\times\tau\in P^{\mathcal{H},+}_{I\times J}} (\#\tau + \#\sigma)$$

is the storage cost of all admissible blocks of the matrix A. ∎

Hence, even if Assumption B is not satisfied, we can find a left orthogonal low-rank representation for a given $\mathcal{H}$-matrix in a cost-efficient manner.

Before we continue, we need another assumption. We want to ensure that the column sizes of blocks in the near-field do not differ too much. Note that, similar to Assumption A, this does not limit the type of matrix we can use. At the cost of a higher depth and some changes in the hierarchical structure, the underlying hierarchy of (block) partitionings can be changed to satisfy this assumption.

Assumption E Let $A \in \mathcal{H}(P^{\mathcal{H},A}_{I\times J}, k_A, \mathrm{adm}_A)$ be an $\mathcal{H}$-matrix and $P^{\mathcal{H},-}_{I\times J}$ its near-field. Then

$$\max_{\sigma\times\tau\in P^{\mathcal{H},-}_{I\times J}} \{\tau\} \leq 2 \cdot \min_{\sigma\times\tau\in P^{\mathcal{H},-}_{I\times J}} \{\tau\}$$

holds.

We continue by estimating the cost of all (dense) QR decompositions that happen on the highest level of the recursion. Note that the $\mathcal{H}$-matrix A in the following Theorem 3.23 is not necessarily a column leaf. We count the operations that are done during all calls to Algorithm 3.7 while computing the QR decomposition of an $\mathcal{H}$-matrix A that possibly contains more than one block column.

Theorem 3.23 (QR decomposition on the highest level) *Assume that Prerequisites A are true. Then the computational cost for all QR decompositions on the highest level computed in Algorithm 3.7 while the QR decomposition of A is computed with Algorithm 3.8 is at most*

$$N_{QR,high}(Y) \leq 8n_{min} \cdot S_{\mathcal{H}}(P_{I\times J}^{\mathcal{H},Y}, k_Y)$$

with n_{min} as given in Definition 2.22.

Proof Following Theorems 3.1 and 3.2, the $\mathcal{H}$-matrix A^{fin} that is composed of matrix blocks from A after they were fully updated by Algorithm 3.9, has in the blocks corresponding to the non-zero part of Y the same $\mathcal{H}$-matrix structure and the same ranks as Y. Furthermore, the block columns of the highest level in A^{fin}, excluding all blocks which are zero in Y, are by construction of the algorithm the block columns that Algorithm 3.7 compresses. Hence, we can use the rank k_Y of Y and the $\mathcal{H}$-block partition $P_{I\times J}^{\mathcal{H},Y}$ to describe the structure of these agglomerated block columns.

We assume that the cost for the QR decomposition of a dense $m \times n$ matrix is bounded by $4mn^2$. Then we can count the cost of the QR decomposition for an $\mathcal{H}$-matrix block column on the highest level of the recursion independently for each matrix block because the cost increases only linearly with the number of rows m. Note that larger admissible blocks get split and thus appear in more than one $\mathcal{H}$-matrix block column. With Assumption E, for a given index block $\sigma \times \tau \in P_{I\times J}^{hier}$, $\frac{2\#\tau}{n_{min}}$ is an upper bound for the number of those appearances.

Furthermore, n_{min} is - by definition and with the last part of Assumption C - an upper bound for the number of columns for each of the smaller blocks after splitting. Using this and summing over all block columns leads to a cost of less than

$$\begin{aligned}
&4 \sum_{\sigma\times\tau\in P_{I\times J}^{\mathcal{H},+}} \left(\frac{2\#\tau}{n_{min}} \cdot k_Y \cdot (n_{min})^2 \right) + 4 \sum_{\sigma\times\tau\in P_{I\times J}^{\mathcal{H},-}} \left(\#\sigma \cdot \#\tau^2\right) \\
&\leq 8k_Y \cdot n_{min} \sum_{\sigma\times\tau\in P_{I\times J}^{\mathcal{H},+}} \#\tau + 8n_{min} \cdot \sum_{\sigma\times\tau\in P_{I\times J}^{\mathcal{H},-}} (\#\sigma \cdot \#\tau) \\
&\leq 8n_{min} \cdot \left(k_Y \cdot \sum_{\sigma\times\tau\in P_{I\times J}^{\mathcal{H},+}} (\#\sigma + \#\tau) + \sum_{\sigma\times\tau\in P_{I\times J}^{\mathcal{H},-}} (\#\sigma \cdot \#\tau) \right) \\
&\leq 8n_{min} \cdot S_{\mathcal{H}}(P_{I\times J}^{\mathcal{H}}, k_Y).
\end{aligned}$$

■

Next we deal with the complexity of multiplication with T given in the reduced form $\overline{T}$. For that, we need to do some preliminary work first and use the set $P_{J\times J}^{upright}$ from

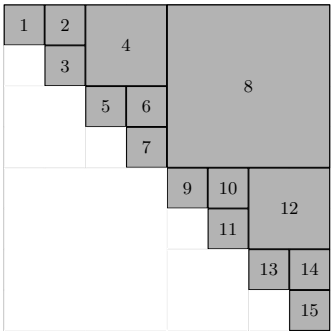


Figure 3.9. Sequence of all factors of the decomposition of T as described in Theorem 3.24.

Definition 3.4 again. Furthermore, let

$$I_{\#J}^{(\tau_1,\tau_2)} \in \mathbb{R}^{J\times J}$$

denote the matrix that is composed of the identity matrix $I_{\#J} \in \mathbb{R}^{J\times J}$, where the submatrix induced by the block $\tau_1 \times \tau_2 \in P_{J\times J}^{hier}$ is replaced by $\overline{T}_{\tau_1,\tau_2}$. We can now show that the cost of multiplication of any $\mathcal{H}$-matrix with T using the reduced form $\overline{T}$ with fast truncation (see Algorithm 2.3 and Theorem 2.44) is identical to the usual $\mathcal{H}$-matrix multiplication of the resulting $\mathcal{H}$-matrix with $\overline{T}$. As a first step, we find a decomposition of T based on Remark 2.16 that extends the observation of this remark to all higher levels, meaning blocks on the diagonal are factored further until we have reached the highest level of the corresponding hierarchy of block partitionings. See also Figure 3.9 for an illustration

Theorem 3.24 *Let Prerequisites A be true. Then there is a decomposition for the factor T given by*

$$T = \left(\prod_{1\le k\le K} I_{\overline{T}}^{(k)} \right),$$

where $I_{\overline{T}}^{(k)} = I_{\#J}^{(\tau_1,\tau_2)} \in \mathbb{R}^{J\times J}$ for some

$$\tau_1 \times \tau_2 \in P_{J\times J}^{upright} \cup \bigcup_{\tau\in P_J^{L_J}} \tau \times \tau$$

and $K \le 2^{depth(T)+1} - 1$. The following condition uniquely defines the sequence of the factors $I_{\overline{T}}^{(k)}$. For $I_{\overline{T}}^{(k)} = I_{\#J}^{(\tau_1,\tau_2)}$ and $I_{\overline{T}}^{(k')} = I_{\#J}^{(\tau_1',\tau_2')}$ with $k' > k$, one of the following is satisfied

1. $\tau_1 \subseteq \tau_1' \wedge \tau_2 < \tau_2'$,

2. $\tau_1 < \tau_1' \wedge \tau_2 < \tau_2'$,

3. $\tau_1 < \tau_1' \wedge \tau_2 \subseteq \tau_2'$.

PROOF If T is an $\mathcal{H}$-matrix that is not split further, then we only have the matrix itself as a factor in the decomposition, and the statement is true. Thus, let T be an $\mathcal{H}$-matrix that is split further, and let $\tau_1, \tau_2 \in P_J^{hier}$ satisfy $\tau_1 \cup \tau_2 = J$. Then T has a factorization given by

$$\begin{pmatrix} T_1 & T_1 Y_1^T Y_2 T_2 \\ 0 & T_2 \end{pmatrix} = \begin{pmatrix} T_1 & 0 \\ 0 & I_{\#\tau_2} \end{pmatrix} \cdot \begin{pmatrix} I_{\#\tau_1} & Y_1^T Y_2 \\ 0 & I_{\#\tau_2} \end{pmatrix} \cdot \begin{pmatrix} I_{\#\tau_1} & 0 \\ 0 & T_2 \end{pmatrix},$$

where $T_1 \in \mathbb{R}^{\tau_1 \times \tau_1}$, $T_2 \in \mathbb{R}^{\tau_2 \times \tau_2}$, $Y_1 \in \mathbb{R}^{I \times \tau_1}$, $Y_2 \in \mathbb{R}^{I \times \tau_2}$ are $\mathcal{H}$-matrices. T_1 and T_2 can, if they are also split further, be recursively factored in the same way again up until the highest level, where the diagonal blocks are stored as full matrices. By definition of $I_{\#J}^{(\tau_1,\tau_2)}$, as given before this theorem, $\overline{T}$ contains all the non-identity and non-zero submatrices of those factors. Furthermore, $P_{J\times J}^{upright}$ is the block set that induces all those submatrices. Every decomposition produces one off-diagonal block. Thus, we get at most

$$2^{\operatorname{depth}(T)} - 1$$

as many. This number is only reached if every index block in P_J^{hier} has two direct descendants or is in $P_J^{L_J}$ and has no descendants. Together with the diagonal blocks on the highest level, we have at most

$$2^{\operatorname{depth}(T)} + 2^{\operatorname{depth}(T)} - 1 = 2^{\operatorname{depth}(T)+1} - 1$$

factors in the product. Hence, together we have

$$T = \left(\prod_{1 \le k \le K} I_{\overline{T}}^{(k)} \right).$$

Let $I_{\overline{T}}^{(k)} = I_{\#J}^{(\tau_1,\tau_2)}$ and $I_{\overline{T}}^{(k')} = I_{\#J}^{(\tau_1',\tau_2')}$ with $k' > k$. During the sequential decomposition, there will be one factor $\tau_a \times \tau_b \in P_{J\times J}^{upright}$ so that one of the following conditions holds:

1. $\tau_1 \times \tau_2 \subset \tau_a \times \tau_a$ and $\tau_1' \times \tau_2' = \tau_a \times \tau_b$,

2. $\tau_1 \times \tau_2 \subset \tau_a \times \tau_a$ and $\tau_1' \times \tau_2' \subset \tau_b \times \tau_b$,

3. $\tau_1 \times \tau_2 = \tau_a \times \tau_b$ and $\tau_1' \times \tau_2' \subset \tau_b \times \tau_b$,

because otherwise $\tau_1 \times \tau_2 = \tau_1' \times \tau_2'$. In the first case, they satisfy condition 1, in the second case, they satisfy condition 2, and in the third case, they satisfy condition 3.

Thus, the second statement is also true. ∎

As our next step, we show that multiplying a low-rank $\mathcal{H}$-matrix with T using the reduced form $\overline{T}$ has the same cost as when using T directly. This is also shown in algorithmic form in Algorithm 3.11 for multiplying with T from the right. Note that this includes reusing intermediate results to keep the computational cost down.

Algorithm 3.11: Multiplication of a low-rank $\mathcal{H}$-matrix with T given in its reduced form $\overline{T}$.

Data: Low rank $\mathcal{H}$-matrix $A = U_A \cdot V_A^T \in \mathcal{H}(P_{I\times J}^{\mathcal{H},A}, k_A, \mathrm{adm}_A)$ given by dense matrices $U_A \in \mathbb{R}^{I\times k_A}$ and $V_A \in \mathbb{R}^{J\times k_A}$, $T \in \mathcal{H}(P_{J\times J}^{\mathcal{H},T}, k_T, \mathrm{adm}_T)$ given in its reduced form $\overline{T} \in \mathcal{H}(P_{J\times J}^{\mathcal{H},\overline{T}}, k_{\overline{T}}, \mathrm{adm}_{\overline{T}})$

Result: The product $A \cdot T = U_B \cdot V_B^T$ given by dense matrices $U_B \in \mathbb{R}^{I\times k_A}$ and $V_B \in \mathbb{R}^{J\times k_A}$

Function: $[U_B, V_B] = MUL_{LR,T}(A, \overline{T})$;

```
Set U_B = U_A;
if J is a column leaf cluster of P_{I×J}^{H,A} then
    V_B = V_A · T̄;
else
    Find non-trivial direct descendants τ1, τ2 ∈ P_J^hier with τ1 < τ2, τ1 ∪ τ2 = J
      and τ1 ∩ τ2 = ∅;
    Split V_A = [V_{A,τ1} V_{A,τ2}];
    V_{B,τ1} = MUL_{LR,T}(V_{A,τ1}, T̄_{τ1,τ1});
    V_{B,τ2} = V_{A,τ2} + V_{B,τ1} · T̄_{τ1,τ2};
    V_{B,τ2} = MUL_{LR,T}(V_{B,τ2}, T̄_{τ2,τ2});
    Set V_B = [V_{B,τ1} V_{B,τ2}];
end
return U_B, V_B;
```

Theorem 3.25 *Let Prerequisites A be true, and let $x \in \mathbb{R}^J$ be a vector. Using $\overline{T}$ instead of T to compute the product Tx while reusing intermediate results has the same cost as the product $\overline{T}x$. Furthermore, let $B \in \mathbb{R}^{J\times K}$ be a matrix given by its low-rank factorization*

$$B = UV^T \quad \text{with} \quad U \in \mathbb{R}^{J\times k},\ V \in \mathbb{R}^{K\times k}$$

for some $k \in \mathbb{R}$ and finite index set K. Then the cost for the multiplication of B with T using $\overline{T}$ and reusing intermediate results is identical to the cost of the operation $\overline{T} \cdot B$ and thus given by

$$k \cdot \mathcal{N}_{MV}(\overline{T}).$$

Proof If T is an $\mathcal{H}$-matrix that is not split further, then $T = \overline{T}$ holds, and the statement

is true. So let T be an $\mathcal{H}$-matrix that is split further, and let $\tau_1, \tau_2 \in P_J^{hier}$ satisfy $\tau_1 \cup \tau_2 = J$. Let

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \mathbb{R}^J \quad \text{with} \quad x_1 \in \mathbb{R}^{\tau_1},\ x_2 \in \mathbb{R}^{\tau_2}$$

be a vector and $T_1 \in \mathbb{R}^{\tau_1 \times \tau_1}$, $T_2 \in \mathbb{R}^{\tau_2 \times \tau_2}$, $Y_1 \in \mathbb{R}^{I \times \tau_1}$, $T_2 \in \mathbb{R}^{I \times \tau_2}$ $\mathcal{H}$-matrices. The dimensions of the factors as well as the number of operations in

$$\begin{pmatrix} T_1 & 0 \\ 0 & I_{\#\tau_2} \end{pmatrix} \cdot \begin{pmatrix} I_{\#\tau_1} & Y_1^T Y_2 \\ 0 & I_{\#\tau_2} \end{pmatrix} \cdot \begin{pmatrix} I_{\#\tau_1} & 0 \\ 0 & T_2 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} T_1 \cdot (x_1 + Y_1^T Y_2 \cdot T_2 x_2) \\ T_2 \cdot x_2 \end{pmatrix}$$

and

$$\begin{pmatrix} T_1 & Y_1^T Y_2 \\ 0 & T_2 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} T_1 \cdot x_1 + Y_1^T Y_2 \cdot x_2 \\ T_2 \cdot x_2 \end{pmatrix}$$

are identical when reusing $T_2 x_2$ in the first one, and thus, the cost is identical as well. We can apply this argument recursively for the entire decomposition. Thus, the cost is the same as for the usual $\mathcal{H}$-matrix-vector multiplication. For the second statement, we hence can apply Lemma 2.41 and get a cost of at most

$$k \cdot \mathcal{N}_{MV}(\overline{T}).$$

∎

Now we show a similar result for the multiplication of an arbitrary $\mathcal{H}$-matrix with T using the reduced form $\overline{T}$. The approach using Remark 2.16 as before is formalized in Algorithm 3.12. Note that for the upcoming cost estimate, we rely on using fast truncation (applied in the computation of line 7 in Algorithm 3.12) to keep the computational cost the same compared to using T directly.

Theorem 3.26 *Let Prerequisites A be satisfied. Furthermore, let there be $\mathcal{H}$-matrices $B \in \mathcal{H}(P_{J\times K}^{\mathcal{H},B}, k, adm_B)$ and $C \in \mathcal{H}(P_{J\times K}^{\mathcal{H},C}, k, adm_C)$ so that*

$$T \cdot B = C$$

holds, and adm_C is induced by the multiplication of adm_T and adm_B. Computing C using Algorithm 3.12 and fast truncation to reduce the product to rank k has the same cost as the usual $\mathcal{H}$-matrix-matrix multiplication $\overline{T} \cdot C$, meaning it is bounded by

$$N_{MM}^{fast}(\overline{T}, C, C),$$

where we used the notation as explained after Theorem 2.47.

Proof If C is a low-rank matrix, we can apply Theorem 3.25. If C is a full rank matrix, we follow the argumentation in Remark 2.16. In both cases B and C have the same structure and rank, so the cost of $\overline{T} \cdot B$ and $\overline{T} \cdot C$ are the same. So assume that B

Algorithm 3.12: Multiplication of an $\mathcal{H}$-matrix with T given in its reduced $\mathcal{H}$-matrix $\overline{T}$.

Data: $\mathcal{H}$-matrices $A \in \mathcal{H}(P_{I\times J}^{\mathcal{H},A}, k_A, \mathrm{adm}_A)$ and $T \in \mathcal{H}(P_{J\times J}^{\mathcal{H},T}, k_T, \mathrm{adm}_T)$ given in its reduced form $\overline{T} \in \mathcal{H}(P_{J\times J}^{\mathcal{H},\overline{T}}, k_{\overline{T}}, \mathrm{adm}_{\overline{T}})$
Result: The approximate product $A \cdot T = B$ given by the $\mathcal{H}$-matrix $B \in \mathcal{H}(P_{I\times J}^{\mathcal{H},B}, k_B, \mathrm{adm}_B)$
Function: $B = MUL_{\mathcal{H},T}(A, \overline{T})$

```
if J is a column leaf cluster of P^{H,A}_{I×J} then
    V_B = V_A · T̄;
else
    Find non-trivial direct descendants τ1, τ2 ∈ P^hier_J with τ1 < τ2, τ1 ∪ τ2 = J
      and τ1 ∩ τ2 = ∅;
    Split A = [A_τ1  A_τ2];
    B_τ1 = MUL_{H,T}(A_τ1, T̄_{τ1,τ1});
    Compute B_τ2 = A_τ2 + B_τ1 · T̄_{τ1,τ2};
    B_τ2 = B_τ2 = MUL_{H,T}(B_τ2, T̄_{τ2,τ2});
    Set B = [B_τ1  B_τ2];
end
return B;
```

is split further. We use the decomposition of T from Theorem 3.24 and thus get

$$C = T \cdot B = \left(\prod_{1 \le k \le K} I_{\overline{T}}^{(k)} \right) \cdot B.$$

Let $\tau \times \gamma$ be an admissible block in C. Let $\dot{\tau}$ denote the the index set of highest level so that $\tau \subseteq \dot{\tau}$ and there is $\ddot{\tau}$ with $\ddot{\tau} \neq J$ so that

$$\dot{\tau} \times \ddot{\tau} \in P_{J\times J}^{upright}.$$

Because $\tau \neq J$, this is always possible except for the last block τ_{last} on the highest level. We exclude this block for now and review it later as a separate case. Let $\dot{k}$ be the index so that

$$I_{\overline{T}}^{(\dot{k})} = I_{\#J}^{(\dot{\tau}\times\ddot{\tau})}.$$

Then

$$\begin{aligned} C &= \left(\prod_{1 \leq k \leq K} I_{\overline{T}}^{(k)} \right) \cdot B \\ &= \left(\prod_{1 \leq k < \dot{k}} I_{\overline{T}}^{(k)} \right) \cdot I_{\#J}^{(\dot{\tau} \times \ddot{\tau})} \cdot \left(\prod_{\dot{k} < k \leq K} I_{\overline{T}}^{(k)} \right) \cdot B. \end{aligned}$$

We now want to check which factors are actually relevant for the computation of $C_{\tau,\gamma}$. Let $\tau_{(k)} \times \tau'_{(k)}$ denote the corresponding block for the k-th factor $I_{\overline{T}}^{(k)}$. First review

$$\prod_{1 \leq k < \dot{k}} I_{\overline{T}}^{(k)}.$$

With Theorem 3.24 we either have $\tau_{(k)} < \dot{\tau}$ or $\tau_{(k)} \times \tau'_{(k)} \subseteq \dot{\tau} \times \dot{\tau}$. In the first case, multiplication with $I_{\overline{T}}^{(k)}$ acts like the identity for all entries of B with row index in $\dot{\tau}$ or below. The other factors can be recombined so that we multiply with the single factor

$$\begin{bmatrix} I & & \\ & T_{\dot{\tau} \times \dot{\tau}} & \\ & & I \end{bmatrix}$$

by reverting the decomposition up to the block $T_{\dot{\tau} \times \dot{\tau}}$. For the second product

$$\prod_{\dot{k} < k \leq K} I_{\overline{T}}^{(k)}$$

a factor either satisfies $\dot{\tau} \subset \tau^{(k)}$ or $\dot{\tau} < \tau^{(k)}$. Let τ_{last} denote the column index block of the last factor satisfying $\dot{\tau} \subset \tau^{(k)}$. Then all the following factors only influence the blocks below the row index set $\dot{\tau}$ and thus τ in B. Finally, we remember that $\tau_{(k)} < \tau'_{(k)}$ and especially $\tau_{(k)} \cap \tau'_{(k)} = \emptyset$. Now we can rewrite the computation of $C_{\dot{\tau},\gamma}$. Let C_2 denote all rows below $\dot{\tau}$ in the block column induced by γ in the result C after truncation to rank k, B_1 all rows above $\dot{\tau}$ in the same block column but of B. Let I_1 be the identity matrix given by the number of rows in B_1 and I_2 the identity matrix given by the number of rows in C_2. Note that the part of C described by C_2 was already computed at this point in the calculation. Then we have

$$\begin{bmatrix} B_1 \\ C_{\dot{\tau},\gamma} \\ C_2 \end{bmatrix} = \begin{bmatrix} I_1 & & \\ & T_{\dot{\tau} \times \dot{\tau}} & \\ & & I_2 \end{bmatrix} \cdot \left(\prod_{\substack{\hat{\tau} \times \hat{\tau}' \in P_{J \times J}^{upright}: \\ \dot{\tau} \subset \hat{\tau}}} \begin{bmatrix} I_1 & & & & \\ & I_{\#\dot{\tau}} & 0 & \overline{T}_{\dot{\tau},\hat{\tau}'} & 0 \\ & & & I_2 & \end{bmatrix} \right) \cdot \begin{bmatrix} B_1 \\ B_{\dot{\tau},\gamma} \\ C_2 \end{bmatrix},$$

and thus

$$C_{\dot{\tau},\gamma} = T_{\dot{\tau},\dot{\tau}} \cdot \left(B_{\dot{\tau},\gamma} + \left(\sum_{\substack{\hat{\tau}\times\hat{\tau}' \in P_{J\times J}^{upright}: \\ \dot{\tau}\subset\hat{\tau}}} \overline{T}_{\dot{\tau},\hat{\tau}'} \cdot C_{\hat{\tau}',\gamma} \right) \right).$$

Restricting ourselves to the admissible block $\tau \times \gamma$, we get

$$C_{\tau,\gamma} = T_{\tau,\dot{\tau}} \cdot \left(B_{\tau,\gamma} + \left(\sum_{\substack{\hat{\tau}\times\hat{\tau}' \in P_{J\times J}^{upright}: \\ \dot{\tau}\subset\hat{\tau}}} \overline{T}_{\tau,\hat{\tau}'} \cdot C_{\hat{\tau}',\gamma} \right) \right).$$

We need to count the operations necessary to compute the product. We assume that we truncate the result of the sum before multiplying with $T_{\tau,\dot{\tau}}$. Thus, the last product has the same cost as

$$T_{\tau,\dot{\tau}} \cdot C_{\tau,\gamma}.$$

Because $C_{\tau,\gamma}$ is admissible, we can apply Theorem 3.25 and see that the computation has the same cost as the product $\overline{T}_{\tau,\dot{\tau}} \cdot C_{\dot{\tau},\gamma}$. Thus, together with the other products, the overall cost is the same as for the operation

$$[\overline{T}_{\tau,\dot{\tau}}\ \overline{T}_{\tau,\tau_i}\ \ldots\ \overline{T}_{\tau,\tau_z}] \cdot \begin{bmatrix} C_{\dot{\tau},\gamma} \\ C_{\tau_i,\gamma} \\ \vdots \\ C_{\tau_z,\gamma} \end{bmatrix} = \overline{T}_{\tau,J} \cdot C_{J,\gamma}.$$

Note that this product does not compute the entry $C_{\tau,\gamma}$, it only has the same cost as the computation of $C_{\tau,\gamma}$.

We are nearly done, we only need to check the computation of the entries $C_{\tau_{last},\gamma}$ for $\tau_{last} \times \gamma \in P_{J\times I}^{hier,C}$. By construction $T_{\tau_{last},\tau_{last}}$ is the only non-zero block in the row induced by τ_{last} in T. Furthermore, we have $\overline{T}_{\tau_{last},\tau_{last}} = T_{\tau_{last},\tau_{last}}$ by construction. Thus

$$C_{\tau_{last},\gamma} = T_{\tau_{last},J} \cdot B_{J,\gamma} = T_{\tau_{last},\tau_{last}} \cdot B_{\tau_{last},\gamma} = \overline{T}_{\tau_{last},\tau_{last}} \cdot B_{\tau_{last},\gamma},$$

and the cost is identical. Hence, the entire statement is true. ∎

For $B \cdot T$, we can achieve the same result using a similar argument, but we omit formalizing it. We are now able to bound the cost of all updates that are computed during the QR decomposition.

Theorem 3.27 (Updates during the QR decomposition) *Let Prerequisites A be true and set*

$$\begin{aligned} k &= \max\{k_A, k_Y, k_{\overline{T}}, k_R\, n_{min}\}, \\ C_{sp} &= \max\{C_{sp}(Y), C_{sp}(R), C_{sp}(\overline{T})\}, \\ L &= \max\{L_I + 1, L_J + 1\}. \end{aligned}$$

The computational cost for all updates during the QR decomposition using Algorithm 3.8 given by the operations in Algorithm 3.9 while using fast truncation is at most

$$\begin{aligned} N_{QR,upd} \leq\ & N_{MM}^{fast}(Y^T, Y, R) + N_{MM}^{fast}(Y^T, R, R) + N_{MM}^{fast}(\overline{T}, R, R) \\ & + N_{MM}^{fast}(Y, R, Y) + N_{MM}^{fast}(Y, R, R) + N_{M+}(Y, Y, Y) + N_{M+}(R, R, R) \end{aligned}$$

or, differently bounded, at most

$$\begin{aligned} N_{QR,upd} \leq\ & C_{sp} \cdot L \cdot \big(C_1 \cdot S_{\mathcal{H}}(Y) + C_2 \cdot S_{\mathcal{H}}(R) \\ & + C_3 S_{\mathcal{H}}(\overline{T}) + C_4 \# P_{I\times J}^{\mathcal{H},Y,+} + C_5 \# P_{J\times J}^{\mathcal{H},R,+}\big) \end{aligned}$$

with

$$C_1 = 48k + 10, \quad C_2 = 120k + 8, \quad C_3 = 2, \quad C_4 = 368k^3, \quad C_5 = 920k^3.$$

Proof Using Theorem 3.5, the calculation of all updates is done as a part of three matrix-matrix products and one matrix-matrix sum. Reusing the notation from Theorem 3.5, we thus need to bound the cost of all relevant computations that are part of the products

$$Y^T \cdot A^{fin},\ T^T \cdot S_1^{comb},\ Y \cdot S_2^{comb}$$

and the sum

$$A^{fin} - Y \cdot S_2^{comb}$$

to find an upper bound for the entire update process. Because we are only interested in the cost of the computations that are actually relevant for the QR decomposition, we can use the structure of S_1^{comb} for the first and S_2^{comb} for the second product, although the results of the products are not inherently upper triangular with a zero block diagonal. For the third product and the sum we use A^{fin}.

Using Theorem 3.26 for $T \cdot S_1^{comb}$, Theorem 2.47 for all other products and Theorem 2.43 for the sum, we get

$$\begin{aligned} N_{QR,upd}\ \leq\ & N_{MM}^{fast}(Y^T, A^{fin}, S_1^{comb}) + N_{MM}^{fast}(\overline{T}, S_2^{comb}, S_2^{comb}) \\ & + N_{MM}^{fast}(Y, S_2^{comb}, A^{fin}) + N_{M+}(A^{fin}, A^{fin}, A^{fin}). \end{aligned}$$

We apply Theorems 3.9 and 3.11 to replace (the parameters of) S_1 and S_2 by R and split A^{fin} into its upper part R and its lower part, which is structurally identical to Y. We get

$$\begin{aligned} N_{QR,upd} \leq & N_{MM}^{fast}(Y^T, Y, R) + N_{MM}^{fast}(Y^T, R, R) + N_{MM}^{fast}(\overline{T}, R, R) \\ & + N_{MM}^{fast}(Y, R, Y) + N_{MM}^{fast}(Y, R, R) + N_{M+}(Y, Y, Y) + N_{M+}(R, R, R) \end{aligned}$$

and thus the first statement. For the second statement, we further estimate the bounds given in Theorem 2.47 and subsequently Theorem 2.41 to get a more compact version. First, we bound the cost for the multiplication without the truncation. We get

$$2 \cdot C_{sp} \cdot L \cdot \big((2+1+1+1) \cdot S_{\mathcal{H}}(Y) + (1+1+1+1) \cdot S_{\mathcal{H}}(R) + (1) \cdot S_{\mathcal{H}}(\overline{T})\big).$$

The truncation cost is then bounded by

$$C_{sp} \cdot L \cdot \big(24k(1S_{\mathcal{H}}(Y) + 4S_{\mathcal{H}}(R)) + 184k^3(1\#P_{I\times J}^{\mathcal{H},Y,+} + 4\#P_{J\times J}^{\mathcal{H},R,+})\big).$$

The cost for the two additions is with Theorem 2.43 bounded by

$$24kS_{\mathcal{H}}(Y) + 184k^3\#P_{I\times Y}^{\mathcal{H},Y,+} + 24kS_{\mathcal{H}}(R) + 184k^3\#P_{J\times J}^{\mathcal{H},R,+}.$$

Adding these results gives us the second statement. ∎

It should be clear that the first bound, although far from perfect, is superior, but the second provides a more direct way of seeing the cost of updates. We can use the latter to directly relate the complexity of the QR decomposition to the parameters of A. At last, the cost of the multiplication in $\overline{T}$ remains.

Theorem 3.28 (Multiplication in $\overline{T}$) *Let Prerequisites A be true and reuse the notation from Theorem 3.27. The computational cost for the calculation of $\overline{T}$ in Algorithm 3.10 is at most*

$$N_{QR,comb} \leq N_{MM}^{fast}(Y^T, Y, \overline{T})$$

or, differently bounded, at most

$$N_{QR,comb} \leq 2 \cdot C_{sp} \cdot L \cdot S_{\mathcal{H}}(Y).$$

Proof All calculations necessary to compute $\overline{T}$ are part of the $\mathcal{H}$-matrix-matrix multiplication $Y^T \cdot Y$. We apply Theorem 3.26 and thus get the first statement. For the second statement, we first use Theorem 2.41 and maximize over $\{C_{sp}(Y), C_{sp}(\overline{T})\}$ to find a bound for the computational cost of the multiplication. We get

$$4 \cdot C_{sp} \cdot L \cdot S_{\mathcal{H}}(Y).$$

The still missing cost for the fast truncation is given by

$$C_{sp} \cdot L \cdot \left(24kS_{\mathcal{H}}(\overline{T}) + 184k^3 \#P_{J\times J}^{\mathcal{H},\overline{T},+}\right).$$

By adding both results, we have an overall cost of

$$C_{sp} \cdot L \cdot \left(4S_{\mathcal{H}}(Y) + 24kS_{\mathcal{H}}(\overline{T}) + 184k^3 \#P_{J\times J}^{\mathcal{H},\overline{T},+}\right). \qquad \blacksquare$$

Now we have everything we need. The cost of Algorithm 3.8 is bounded by the cost of only $\mathcal{H}$-matrix-matrix multiplications, the cost of the left orthogonalization is bounded in Theorem 3.22, and the cost for the QR decomposition on the lowest level is bounded in Theorem 3.23.

We can collect all results up until now.

Theorem 3.29 *Let Prerequisites A be true and set*

$$\begin{aligned} k &= \max\{n_{min}, k_A, k_{\overline{T}}, k_Y, k_R\}, \\ L &= \max\{L_I, L_J\}, \\ C_{sp} &= \max\{C_{sp}(Y), C_{sp}(R), C_{sp}(\overline{T})\}. \end{aligned}$$

Then the cost for computing the QR decomposition by Algorithm 3.8 using fast truncation is bounded by

$$\begin{aligned} N_{qr} \leq C_{sp} \cdot L \cdot \big(&C_1 \cdot S_{\mathcal{H}}(A) + C_2 \cdot S_{\mathcal{H}}(Y) + C_3 \cdot S_{\mathcal{H}}(R) + C_4 \cdot S_{\mathcal{H}}(\overline{T}) \\ &+ C_5 \cdot \#P_{I\times J}^{\mathcal{H},Y,+} + C_6 \cdot \#P_{J\times J}^{\mathcal{H},R,+} + C_7 \cdot \#P_{J\times J}^{\mathcal{H},\overline{T},+}\big) \end{aligned}$$

with

$$\begin{aligned} &C_1 = 5k, \quad C_2 = 56k + 14, \quad C_3 = 120k + 8, \quad C_4 = 24k + 2, \\ &C_5 = 368k^3, \quad C_6 = 920k^3, \quad C_7 = 184k^3. \end{aligned}$$

Proof We collect our already proven cost estimates:

1. For the left orthogonalization, we use Theorem 3.22 and get an upper bound of

$$N_{lorth}(A) \leq 5k_A \cdot S_{\mathcal{H}}(Y, k_A).$$

2. For the QR decomposition on the lowest level, we use Theorem 3.23. We get

$$N_{QR,high}(Y) \leq 8n_{min} \cdot S_{\mathcal{H}}(Y).$$

3. For the updates, we use Theorem 3.27 and have

$$N_{QR,upd} \leq C_{sp} \cdot L \cdot \big(C_1 \cdot S_{\mathcal{H}}(Y) + C_2 \cdot S_{\mathcal{H}}(R) + C_3 S_{\mathcal{H}}(\overline{T}) + C_4 \# P_{I\times J}^{\mathcal{H},Y,+} + C_5 \# P_{J\times J}^{\mathcal{H},R,+} \big).$$

4. For the computations in $\overline{T}$, we use Theorem 3.28. There are no further computational costs associated with the other QR factors. The matrix R is already computed during the updates, and the block columns in Y are just recombined. Hence, we have a cost for this part of the QR decomposition of at most

$$C_{sp} \cdot L \cdot \left(4 S_{\mathcal{H}}(Y) + 24k S_{\mathcal{H}}(\overline{T}) + 184k^3 \# P_{J\times J}^{\mathcal{H},\overline{T},+} \right).$$

By combining these bounds, maximizing over some parameters and for 1 and 2 adding the originally not necessary parameter $C_{sp} \cdot L$ to achieve a more compact representation, we get the statement. ∎

Theorem 3.29 is a significant result. It means that the QR decomposition can be computed cost-efficiently if the resulting factorizations can be stored cost-efficiently. This resembles the standard result regarding the cost of the $\mathcal{H}$-matrix product. However, it still relies on the (a posteriori) knowledge of the storage cost of the resulting factors and their sparsity constants.

Our results from Subsection 3.3.4 offer some relief in that regard. At least for square matrices, the bandwidths of all resulting factors are bounded by the bandwidth of A. Additionally, as the following theorem will show, the sparsity constant can be (easily) bounded by the bandwidth constant.

Theorem 3.30 *Let $A \in \mathcal{H}(P_{I\times J}^{\mathcal{H}}, k, adm_A)$ be an $\mathcal{H}$-matrix with $\mathcal{H}$-block partition $P_{I\times J}^{\mathcal{H}}$ based on $\left(P_{I\times J}^{\ell}\right)_{\ell=0}^{L}$. Then*

$$C_{sp}(A) \leq 2 \cdot C_{bw}(A).$$

Proof We have $C_{sp}^0 = 1$ for all $\mathcal{H}$-matrices by construction. Every block on the ℓ-th level of $P_{I\times J}^{\mathcal{H}}$ has to be contained in an inadmissible block on the $(\ell-1)$-th level. Thus, there are at most $2 \cdot C_{bw}^{\ell-1}(A)$ blocks in $P_{I\times J}^{\ell} \cap P_{I\times J}^{\mathcal{H}}$ because any index set in P_J^{hier} has at most two direct descendants by Assumption A. For $0 < \ell < L$, we only have to count the admissible blocks, but that would only reduce the bound, so maximizing over all levels, we get

$$C_{sp}(A) \leq 2 \cdot C_{bw}(A).$$ ∎

As mentioned in the proof, bounding $C_{sp}(A)$ by $2 \cdot C_{bw}(A)$ will overestimate $C_{sp}^{\ell}(A)$ with $0 < \ell < L$. If $C_{bw}^{\ell}(A)$ denotes the actual number of inadmissible blocks on the ℓ-the level, we would have

$$C_{sp}^{\ell}(A) \leq 2C_{bw}^{\ell-1}(A) - C_{bw}^{\ell}(A) \text{ for } 0 < \ell < L.$$

But for $C_{sp}^{L}(A)$, we need to count all blocks anyway, even the inadmissible ones, because they are not split any further, so we do not use this observation.

For square $\mathcal{H}$-matrices we can thus replace the sparsity constant with twice the bandwidth constant of A in Theorem 3.29. This allows us to compress the result further and relate it only to the parameters of the matrix A. However, we still rely on the storage cost of the resulting $\mathcal{H}$-matrices. Depending on the actual structure of A, the resulting factors may have a considerably higher storage cost compared to A. But there is a way to overcome this problem. Due to the fact that all involved $\mathcal{H}$-matrices have the same bandwidth, the upper bound for the storage cost of A given by Theorem 2.40 is also an upper bound to the storage cost of all other factors if we use the bandwidth constant.

3.3.6 Reusing the Original Admissibility Function

The obvious solution for the massive fill-in found in Theorem 3.14 and especially Theorem 3.15 would be to refrain from using the induced admissibility function and find a suitable alternative. The naive approach would be to reuse the admissibility function of A for Y and R. Note here that we defined the admissibility function of R on the index set $J \times J$, but the admissibility function of A is defined on the index set $I \times J$. To still be able to transfer it, we use Definition 3.6 and simply set

$$\operatorname{adm}_R(\tau \times \tau') = \operatorname{adm}_A^{ext}(\sigma(\tau) \times \tau').$$

This transfer can be done indirectly in the implementation because Y and R can be saved by overwriting the original matrix A. Then R is not necessarily saved in a connected submatrix of A if A is rectangular but not square, but depending on the implementation, any block in R can directly inherit the admissibility function of the corresponding block in A when overwriting it.

There is no direct equivalent for $\overline{T}$, S_1 and S_2, so we want to suggest two alternatives. First, we set the individual blocks of T, S_1^{comb} and S_2^{comb} used at one step of the recursion to be admissible. Hence, the entire matrices would then have an admissibility function given directly by $P_{J\times J}^{upright}$. The main problem here is that the original structure of A does not influence the structures of $\overline{T}$, S_1^{comb} and S_2^{comb}. This is counterintuitive, and we will not pursue this approach any further.

An alternative that hopefully adapts better to structural changes in A would be to reuse the structure of the upper triangular part of A for $\overline{T}$, S_1^{comb} and S_2^{comb} as well. As shown in Theorems 3.11 and 3.15, the true admissibility functions of S_1^{comb} and S_2^{comb} are closely related to R and in many cases identical. Therefore, using the same admissibility function for all three makes sense. For $\overline{T}$ there is no similar argument, but nevertheless, we use the same admissibility function as for R, S_1^{comb} and S_2^{comb}. One could argue that the admissibility functions for $\overline{T}$

$$\operatorname{adm}_{\overline{T}}(\tau \times \tau') = \prod_{\sigma \in P_I^{L^{max}(\tau,\tau',I)}} \operatorname{adm}_Y(\sigma_i \times \tau) + \operatorname{adm}_Y(\sigma_i \times \tau')$$

and S_1

$$\operatorname{adm}_{S_1}(\tau \times \tau') = \prod_{\sigma \in P_I^{L^{max}(\tau,\tau',I)}} \operatorname{adm}_Y(\sigma \times \tau) + \operatorname{adm}_{\hat{A}}(\sigma \times \tau')$$

are similar enough to warrant such an approach. However, this is a rather flimsy argument in the general case because the row induced by σ in A has no zero blocks, whereas the row induced by σ in Y has possibly many zero blocks that are hence admissible. Nevertheless, it is a better approach than simply using an $\mathcal{H}_p$-matrix structure.

Aside from whether these approaches lead to useful numerical results, the question remains in what way the complexity estimates from before change. This is not our primary focus, and we would need to repeat many arguments already made in the chapter before, but we nevertheless want to give a rough sketch of the change.

Theorem 3.22 does not rely on the induced admissibility function. It simply uses the admissibility function of Y, which can thus be replaced by the admissibility function of A.

Theorem 3.23 uses the highest possible rank that can arise due to the updates during Algorithm 3.8. These calculations are no longer true because it is now possible to update an admissible block using an inadmissible block. The rank of the result could be much higher. To reuse this theorem, we need to assume that truncation to low enough bounds takes place. This is associated with further truncation costs but they are fortunately included in the update costs. These are given by $\mathcal{H}$-matrix-matrix multiplications as shown in Theorem 3.27, which do not rely directly on the induced admissibility functions, only indirectly because we use Theorem 2.47 to bound the computational cost involved.

Fortunately, Theorem 2.47 can also be replaced by a version that uses the constant C_{id} to model the influence of the different admissibility functions of the result of an $\mathcal{H}$-matrix-matrix product.

At last Theorem 3.25 and 3.26 remain. In Theorem 3.26 we show that the cost of multi-

plication using the reduced form of T given by $\overline{T}$ is the same as the usual multiplication with $\overline{T}$ under the assumption of pairwise truncation. By again replacing Theorem 2.47 with a version that uses the constant C_{id}, we could change the proof of Theorem 3.26 accordingly. Hence, this is also no obstacle.

Thus, a similar cost estimate as before is possible, although we need to be aware that this includes an upper bound to the realized ranks in the factors. For a reasonable cost, this has to be low enough. Because we no longer use the induced admissibility function, it is more likely that this leads to problems regarding the accuracy and orthogonality of the resulting QR decomposition, although we may experience some speedup.

But it is not clear that we actually see a drastic reduction in computation time due to the decrease in inadmissible blocks compared to using the induced admissibility functions. This depends on how the inadmissible blocks of A interact with each other. If we still compute many full-rank matrices that subsequently are reduced to a matrix with a low-rank factorization, we may have some improvements to the storage cost, but the computation time is still high. We will keep this in mind and get back to it in Chapter 4.

Assuming we let the truncation depend on some given accuracy and do not set an upper bound for the rank in an admissible block, replacing the induced admissibility function could have a drastic negative influence on the computation time. Suppose we need a relatively high rank to approximate admissible blocks that are inadmissible in the induced admissibility function. In that case, we have a high cost for truncation but no cost advantage due to the low-rank factorization.

Furthermore, we introduce additional inaccuracies in the resulting factors because we calculate an approximate result, whereas the induced admissibility function computes an exact one, ignoring all inaccuracies previously accrued.

We will observe this behavior in nearly all our numerical examples. Hence, the induced admissibility function can also indicate where we could expect high ranks when using another admissibility function.

3.3.7 The Rectangular Case

As seen in Theorem 3.14, the main problem in calculating a QR decomposition for a rectangular $\mathcal{H}$-matrix is the row-wise fill-in in Y. The condition mentioned in Theorem 3.14 is satisfied for many $\mathcal{H}$-matrices, so Y no longer has a sufficiently data-sparse representation as an $\mathcal{H}$-matrix. We can show that the storage cost grows as $\mathcal{O}((\#I - \#J) \cdot \#J)$ in that case. However, to simplify things, we assume that the blocks on the highest level of the hierarchical partitioning are roughly the same size and the set $P^{streak}_{I\times J}$

(see Assumption D) resembles a stretched diagonal. We formalize this in the following assumption.

Assumption F Let $A \in \mathcal{H}(P_{I\times J}^{\mathcal{H},A}, k_A, \mathrm{adm}_A)$ be an $\mathcal{H}$-matrix. Then the set $P_{I\times J}^{streak} \subset P_{I\times J}^{hier}$ as defined in Assumption D satisfies the following additional condition. For

$$s_{min} := \min_{\tau\in P_J^{L_J}} \# \left(\bigcup_{\sigma\in P_I^{L_I}:\sigma\times\tau\in P^{streak}} \sigma \right), \quad s_{max} := \max_{\tau\in P_J^{L_J}} \# \left(\bigcup_{\sigma\in P_I^{L_I}:\sigma\times\tau\in P^{streak}} \sigma \right)$$

and

$$t_{min} := \min_{\sigma\in P_I^{L_I}} \# \left(\bigcup_{\tau\in P_J^{L_J}:\sigma\times\tau\in P^{streak}} \tau \right), \quad t_{max} := \max_{\sigma\in P_I^{L_I}} \# \left(\bigcup_{\tau\in P_J^{L_J}:\sigma\times\tau\in P^{streak}} \tau \right),$$

we have

$$\max\left\{ \frac{s_{max}}{s_{min}}, \frac{t_{max}}{t_{min}} \right\} \leq M$$

for some $M \in \mathbb{R}$.

This assumption is satisfied for the hierarchical partitioning of the rows or columns when the index sets on the highest level only differ by a fixed factor independent of size $\#I$ or $\#J$, which is generally the case.

Theorem 3.31 *Let Prerequisites A be true, and assume Assumptions D and F are satisfied. Then the storage cost of Y is bounded below by*

$$S_{\mathcal{H}}(Y) \geq \frac{1}{2M^2} \cdot \left(\#I - M^2 \cdot \#J\right) \#J.$$

Proof Let $\tau \in P_J^{L_J}$ be fixed. Then all row index sets σ that are contained in

$$\bigcup_{\substack{\sigma\in P_I^{L_I}:\ \sigma\times\tau\in P^{streak} \\ \sigma\cap\sigma(\tau)=\emptyset}} \sigma$$

induce rows that are contained entirely in Y. Because the blocks $\sigma\times\tau$ that are contained in $P_{I\times J}^{streak}$ are inadmissible and the conditions of Theorem 3.14 are satisfied, these block rows consist entirely of inadmissible blocks to the right of $\sigma \times \tau$.

Because $\#\sigma(\tau) = \#\tau < t_{max}$ by construction, there are at least $s_{min} - t_{max}$ of those rows. Hence, the rows together induce at least

$$(s_{min} - t_{max}) \cdot \# \bigcup_{\hat{\tau} \in P_J^{L_J} : \hat{\tau} > \tau} \hat{\tau} > (s_{min} - t_{max}) \cdot \sum_{\hat{\tau} \in P_J^{L_J} : \hat{\tau} > \tau} t_{min}$$

matrix entries stored in inadmissible blocks. Summing this over all $\tau \in P_J^{L_J}$, we get at least

$$\begin{aligned}
(s_{min} - t_{max}) \cdot \sum_{\tau \in P_J^{L_J}} \sum_{\hat{\tau} \in P_J^{L_J} : \hat{\tau} > \tau} t_{min} &> (s_{min} - t_{max}) \frac{\#P_J^{L_J}(\#P_J^{L_J} + 1) t_{min}}{2} \\
&= (s_{min} \cdot \#P_J^{L_J} - t_{max} \cdot \#P_J^{L_J}) \frac{(\#P_J^{L_J} + 1) t_{min}}{2} \\
&\geq \left(\frac{s_{max}}{M} \cdot \#P_J^{L_J} - M \cdot t_{min} \cdot \#P_J^{L_J} \right) \frac{\#P_J^{L_J} \cdot t_{max}}{2M} \\
&\geq \left(\frac{1}{M} \cdot \#I - M \cdot \#J \right) \frac{\#J}{2M} \\
&= \frac{1}{2M^2} \cdot \left(\#I - M^2 \cdot \#J \right) \#J
\end{aligned}$$

matrix entries stored in inadmissible blocks in Y. ∎

If $P_{I \times J}^{streak}$ looks similar to a stretched diagonal, then M does not depend on the size of the matrix. If $\#I$ is large enough compared to $\#J$ to offset the subtraction of $M^2 \cdot \#J$, then the storage cost of Y grows as fast as

$$\mathcal{O}(\#I \cdot \#J).$$

That means we have no advantage in complexity compared to saving Y as a dense matrix. This is especially a problem in light of our results in Subsection 2.2.3. The cost of the entire algorithm can be bounded by the storage cost of all involved factors with additional constants, so if Y cannot be stored efficiently, the entire QR decomposition cannot be computed efficiently.

It is thus questionable whether this approach is feasible for rectangular $\mathcal{H}$-matrices satisfying the condition in Theorem 3.14.

3.4 Implementation in the Software Package H2Lib

We already gave a short, general introduction to the H2Lib in Section 2.4. In this section, we want to elaborate on some aspects specific to implementing the block Householder approach for the QR decomposition given by Algorithm 3.8.

There are three main difficulties. First, we need to implement the splitting process described at the beginning of Section 3.2 without incurring additional and unnecessary storage costs. Second, we need to efficiently define the induced structure of every $\mathcal{H}$-matrix we compute during the algorithm.

Third, we also need to compute the update in line 6 of Algorithm 3.9 correctly using the results from Theorem 3.2 so that the left factor in split admissible block does not change, and we can recombine them in Y.

The first problem can be solved by creating new `hmatrix` structs at every step where a split occurs, but we only use them to refer to already existing `hmatrix`, `amatrix` and `rkmatrix` objects or their entries. From now on, we will call these objects `hmatrix` hulls.

Hence, we create two new `hmatrix` hulls, one for the left part and one for the right part after the split. If the original `hmatrix` points to an `rkmatrix`, we also need to create two new `rkmatrix` objects but let them refer to the entries of the original `rkmatrix` to reduce the necessary storage cost. If it points to an `amatrix`, we do the same and create two new `amatrix` objects but refer to the entries of the original `amatrix`.

Suppose the original `hmatrix` itself has sons. In that case, we have to carefully look at their structure and go through the hierarchy until we can either identify left and right `hmatrix` objects or we arrive at an `amatrix` or `rkmatrix`. In the first case, we assign all `hmatrix` objects on the left to the left `hmatrix` and all the `hmatrix` objects on the right to the right `hmatrix`. In the second case, we use the already explained method to split `amatrix` and `rkmatrix` blocks. This whole process is sketched in Algorithm 3.13. Assumption A also comes into play because the entire process would still be possible but far more complicated if we could have more than two column sons.

It is important to note that on our way through the hierarchy, until we perform the splits, we further build the left and the right `hmatrix` hull. The depth of the newly formed `hmatrix` objects needs to be the same as the depth of their counterparts in the original `hmatrix` objects because otherwise the addition and multiplication routines may not work anymore. We also have to have a clear distinction between which parts of the structure are newly built for the ”new” left and right `hmatrix` objects and which belong to the ”original” `hmatrix` object. Otherwise, we cannot properly delete the `hmatrix` hulls after they are not needed anymore.

Algorithm 3.13: Vertically split $\mathcal{H}$-matrices in H2Lib.

Data: `hmatrix` struct A with row cluster σ and column cluster τ
Result: `hmatrix` structs A^{left}, A^{right}, where A^{left} is the left and A^{right} is the right part of A
Function: $[A^{left}, A^{right}] = hmatrix\_split(A)$

Create `hmatrix` hulls $A^{(left)}$ and $A^{(right)}$ with 1 column son and as many row sons as A;
if *A points to* ***rkmatrix*** *rk* **then**
  Create 2 empty `rkmatrix` objects $rk^{(left)}$, $rk^{(right)}$;
  Let $rk^{(left)}$ refer to the left part of rk and $rk^{(right)}$ to the right part of rk;
  Let $A^{(left)}$ point to $rk^{(left)}$ and $A^{(right)}$ to $rk^{(right)}$;
else if *A points to* ***amatrix*** *am* **then**
  Create 2 empty `amatrix` objects $am^{(left)}$, $am^{(right)}$;
  Let $am^{(left)}$ refer to the left part of am and $am^{(right)}$ to the right part of am;
  Let $A^{(left)}$ point to $am^{(left)}$ and $A^{(right)}$ to $am^{(right)}$;
else
  if *the column cluster of A has 2 sons τ_1 and τ_2* **then**
    for *all sons σ_i of the row cluster of A* **do**
      Let $A^{left}_{\sigma_i,\tau_1}$ refer to A_{σ_i,τ_1};
      Let $A^{right}_{\sigma_i,\tau_2}$ refer to A_{σ_i,τ_2};
    end
  else
    `// The column cluster of` A `has 1 son.`
    for *all sons σ_i of the row cluster of A* **do**
      $(A^{left}_{\sigma_i,J}, A^{right}_{\sigma_i,J}) = hmatrix\_split(A_{\sigma_i,J})$;
    end
  end
end
return $A^{(left)}$, $A^{(right)}$;

The solution to the second problem of how to implement the induced structure is a bit more complicated. First, we deal with defining the induced structure of the addition of two $\mathcal{H}$-matrices. For that, we need a function that creates a new `hmatrix` or uses an existing `hmatrix` object and refines it whenever at least one of the summands has a finer structure. By multiplication of the factors, any `rkmatrix` object can be turned into a new `amatrix` object, and all necessary splits can be done, as explained above. The process can be seen in Algorithm 3.14. If we refine an existing `hmatrix` object, we must be careful to preserve the existing entries without additional storage or computational cost.

The process for the $\mathcal{H}$-matrix multiplication is similar, but there are some differences. We recall that the block $C_{\sigma,\gamma}$ of $\mathcal{H}$-matrix product $C = A \cdot B$, where $A \in \mathcal{H}(P^{\mathcal{H}}_{I\times J}, k, \mathrm{adm}_A)$, $B \in \mathcal{H}(P^{\mathcal{H}}_{J\times K}, k, \mathrm{adm}_B)$ and $C \in \mathcal{H}(P^{\mathcal{H}}_{I\times K}, k, \mathrm{adm}_C)$, is given by

$$C_{\sigma,\gamma} = \sum_{\tau \in P^{\ell}_{J}} A_{\sigma,\tau} \cdot B_{\tau,\gamma},$$

where $0 \leq \ell \leq L_J$, and L_J is the depth of the hierarchy of partitionings of J.

Algorithm 3.14: Induced structure for the $\mathcal{H}$-matrix sum in H2Lib.

Data: `hmatrix` structs A, B and C with row `cluster` σ and column `cluster` τ
Result: `hmatrix` struct C with induced structure of the operation $C = A + B$
Function: $C = OptStructSum(A, B, C)$

if *A and B point to an* ***rkmatrix*** **then**
  `// Do nothing.`
else if *C points to an* ***amatrix*** **then**
  `// Do nothing.`
else
  if *A or B point to an* ***amatrix*** **then**
    Create new amatrix am and fill it with the data from C;
    Let C point to am;
  else if *A points to an* ***rkmatrix*** **then**
    `//` *B* `has sons.`
    If necessary, split C to have as many row and column sons as B;
    for *all sons σ_i of σ and τ_j of τ* **do**
      $OptStructSum(A, B_{\sigma_i,\tau_j}, C_{\sigma_i,\tau_j})$;
    end
  else if *B points to an* ***rkmatrix*** **then**
    `//` *A* `has sons.`
    If necessary, split C to have as many row and column sons as A;
    for *all sons σ_i of σ and τ_j of τ* **do**
      $OptStructSum(A_{\sigma_i,\tau_j}, B, C_{\sigma_i,\tau_j})$;
    end
  else
    `//` *A* `and` *B* `have sons.`
    If necessary, split C to have as many row and column sons as A and B;
    for *all sons σ_i of σ and τ_j of τ* **do**
      $OptStructSum(A_{\sigma_i,\tau_j}, B_{\sigma_i,\tau_j}, C_{\sigma_i,\tau_j})$;
    end
  end
end
return C;

After choosing the correct level ℓ, the induced admissibility of $C_{\sigma,\gamma}$ is given by a combination of the admissibilities of the factors $A_{\sigma,\tau}$ and $B_{\tau,\gamma}$. To be more precise, the product $A_{\sigma,\tau} \cdot B_{\tau,\gamma}$ is inadmissible if both $A_{\sigma,\tau}$ and $B_{\tau,\gamma}$ are inadmissible, and the sum $\sum_{\tau \in P_j^\ell} A_{\sigma,\tau} \cdot B_{\tau,\gamma}$ is inadmissible if one of the summands $A_{\sigma,\tau} \cdot B_{\tau,\gamma}$ in the entire sum is inadmissible. We formalized this in Theorem 2.31.

Algorithm 3.15: Induced structure for the $\mathcal{H}$-matrix product in H2Lib.

Data: `hmatrix` structs A with row `cluster` σ and column `cluster` τ, B with row `cluster` τ and column `cluster` γ and C with row `cluster` σ and column `cluster` γ
Result: `hmatrix` struct C with induced structure of the operation $C = A \cdot B$
Function: $C = OptStructProd(A, B, C)$

```
1  if A or B point to an rkmatrix then
       // Do nothing.
2  else if C points to an amatrix then
       // Do nothing.
3  else
4      if A and B point to an amatrix then
5          Create new amatrix am and fill it with the data from C;
6          Let C point to am;
7      else if A points to an amatrix then
           // B has sons.
8          If necessary, split C to have as many column sons as B;
9          for all sons γ_i of γ do
10             OptStructProd(A, B_{τ,γ_i}, C_{σ,γ_i});
11         end
12     else if B points to an amatrix then
           // A has sons.
13         If necessary, split C to have as many row sons as A;
14         for all sons σ_i of σ do
15             OptStructProd(A_{σ_i,τ}, B, C_{σ_i,γ});
16         end
17     else
           // A and B have sons.
18         If necessary, split C to have as many row sons as A and column sons as
             B;
19         for all sons σ_i of σ, τ_j of τ and γ_k of γ do
20             OptStructProd(A_{σ_i,τ_j}, B_{τ_j,γ_k}, C_{σ_i,γ_k});
21         end
22     end
23 end
24 return C;
```

Unfortunately, this cannot be used directly in the implementation to define the admissibility of $C_{\sigma,\gamma}$ because we have to go recursively through the structures of A and B and cannot directly access a given block on a given level.

In order to nevertheless define the induced structure of C, we start at the top level of A and B and go one by one through all products $A_{\sigma,\tau} \cdot B_{\tau,\gamma}$ that get added to a given target block $C_{\sigma,\gamma}$. Then we continue with all other target blocks. Hence, we do not check the influence of the sum of products all in one but of every product independently.

If either $A_{\sigma,\tau}$ or $B_{\tau,\gamma}$ are low-rank or $C_{\sigma,\gamma}$ is full rank, we stop because the structure of $C_{\sigma,\gamma}$ will not be changed by adding $A_{\sigma,\tau} \cdot B_{\tau,\gamma}$. If both are full rank, we set $C_{\sigma,\gamma}$ to be full rank as well. Things get more convoluted if one or both factors have further hierarchical structures. If $C_{\sigma,\gamma}$ is low-rank, we choose the structure that best suits the structure of $A_{\sigma,\tau}$ and $B_{\tau,\gamma}$. If $C_{\sigma,\gamma}$ already has a hierarchical structure, we may need to refine it. The details can be found in Algorithm 3.15. Note that this sketch of the algorithm simplifies certain aspects. The blocks $B_{J,\gamma}$ in line 9 and $A_{\sigma,J}$ in line 14 are not necessarily sons of B or, respectively, A. Hence, an `hmatrix` hull similar to Algorithm 3.13 may be necessary. Furthermore, additional auxiliary structures may be needed if C has more row sons than A or more column sons than B.

The third problem mentioned at the beginning can be solved by adding a case distinction in line 6 of Algorithm 3.9. We modify the usual $\mathcal{H}$-matrix multiplication routine to identify blocks $A_{\sigma\times\tau_2}$ in the block column $A_{I\times\tau_2}$ that were part of a larger admissible block in A. As already seen in Theorem 3.2, that means that the neighboring block $A_{\sigma\times\tau_1}$ in $A_{I\times\tau_1}$ belongs to the same larger admissible block and we can perform the update only on the right factor of $A_{\sigma\times\tau_1}$. This does not work if both $A_{\sigma\times\tau_1}$ and $A_{\sigma\times\tau_2}$ are admissible but not part of the same larger admissible block. Thus, it is not enough to check whether neighboring blocks are both admissible. To identify the correct admissible blocks, we check whether the `amatrix` object that makes up the left factor of $A_{\sigma\times\tau_1}$ references the same object as the `amatrix` object that makes up the left factor of $A_{\sigma\times\tau_2}$.

Another critical aspect of the implementation is that the matrices Y and R together overwrite the matrix A. Hence, whenever we need to use one of these matrices during the computation of the entire QR decomposition, e.g. in line 2 of Algorithm 3.9, we need to extract them from their joint matrix. We cannot copy the required parts because this would lead to unnecessary storage and computational cost. We reuse the ideas from Algorithm 3.13 and only create `hmatrix` hulls that refer to and do not copy the relevant parts.

We also need some way to remember which parts of the joint matrix belong to Y and which to R. For square matrices, this is obvious; every block below the diagonal belongs to Y, every block above the diagonal belongs to R, and the diagonal blocks on the highest level are split diagonally between Y and R. But we are generally not in this situation. During the application of the algorithm, we have a rectangular block column, and we

still need to identify what belongs to Y and what to R. If the matrix, at the beginning, was a square matrix, we could find a way to do that using the row and column clusters `rc` and `cc`, but for a rectangular starting matrix, this does not work anymore. We have to find an alternative. Our solution was to define an auxiliary structure with the same hierarchy as the original matrix and containing all necessary information. More about the further intricacies of the implementation can be found in the accompanying documentation.

Chapter 4

Numerical Results

In this chapter we will show numerical tests for the Householder-based recursive $\mathcal{H}$-QR factorization as given in Algorithm 3.8 (with auxiliary Algorithms 3.7, 3.9 and 3.10). They are based on our own publication [40], where the main focus - aside from presenting the new approach to the QR decomposition of $\mathcal{H}$-matrices - was exactly this analysis.

We choose the following types of test matrices:

1. $\mathcal{H}$-matrices resulting from a boundary-element discretization of the 2D-Laplacian and the 3D-Laplacian as described in Section 2.6 (based on the corresponding example in the H2Lib library [41]).

2. $\mathcal{H}$-matrices occurring in radial basis function (RBF) interpolation problems in $d \in \{1, 2\}$ and approximation problems in $d = 1$ spatial dimensions. We use the Gaussian function

$$f(x, y) = e^{-\varepsilon \|x-y\|_2^2}$$

   with shape parameter $\varepsilon = c_d \cdot n^{1/d}$, where n is the number of data points. We use for $d = 1$ equidistant points in $[0, 1]$ with $c_1 = 0.02$ and for $d = 2$ equidistant points in $[0, 1]^2$ with $c_2 = 0.2$. For details on RBF interpolation and approximation see our introduction in Section 2.5 and [53, 26, 82]. We also show tests for $\mathcal{H}$-matrices with $\mathcal{H}$-block partitions as for the RBF interpolation and approximation matrices but where all dense blocks as well as the factors U, V in all admissible blocks are filled with random entries.

3. Randomly filled $\mathcal{H}$-matrices with an X-like block structure of inadmissible blocks (see Figure 4.1d).

4. Randomly filled, rectangular $\mathcal{H}_p$-matrices.

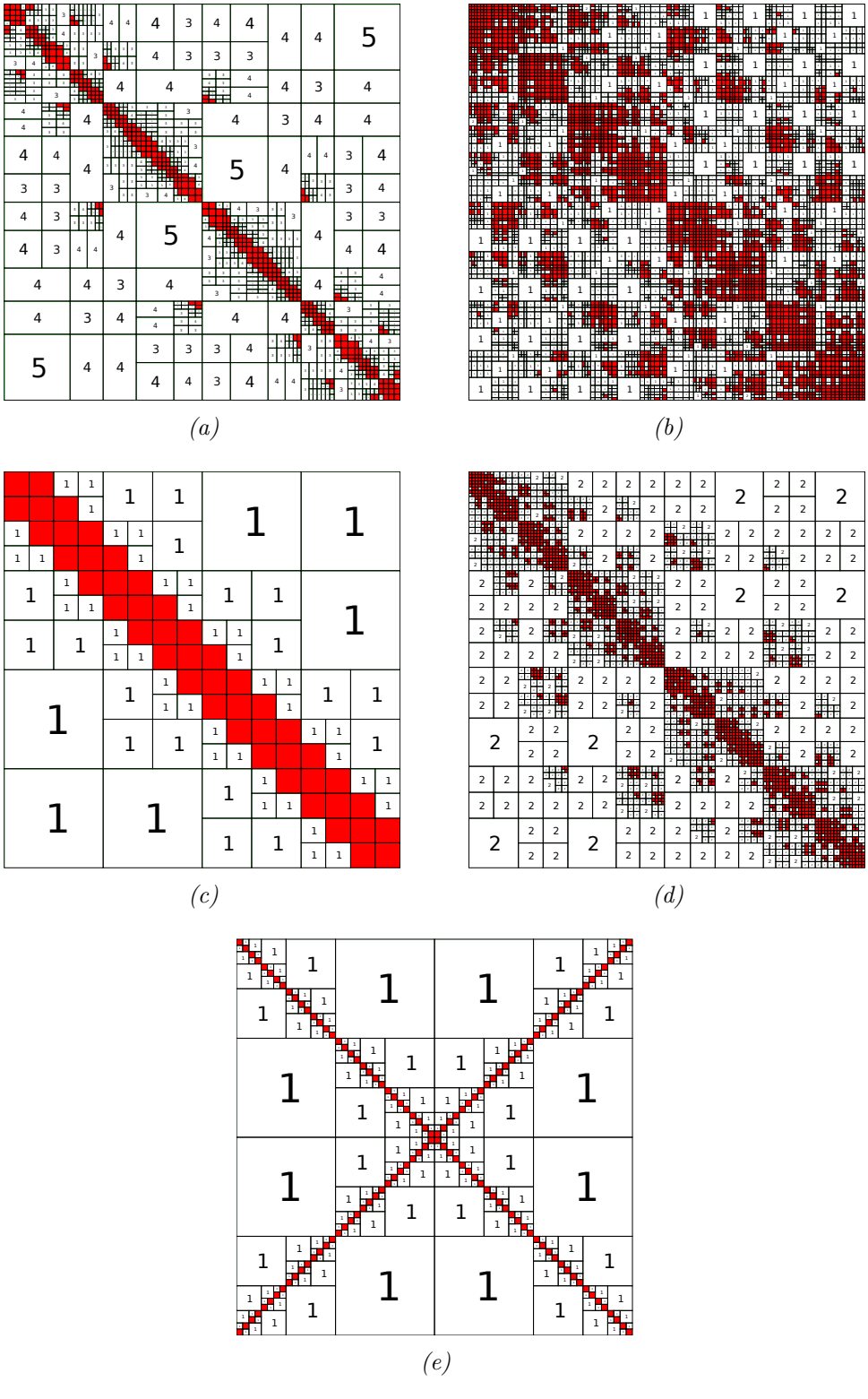


Figure 4.1. (Square) $\mathcal{H}$-matrices used in our numerical tests: (a) 2D Laplacian, (b) 3D Laplacian, (c) 1D RBF interpolation, (d) 2D RBF interpolation, (e) X-like random matrix.

We use adaptive cross approximation [9] to convert the (typically) dense matrices in the above test cases to approximating $\mathcal{H}$-matrices. Typical $\mathcal{H}$-matrix structures for our test matrices are shown in Figure 4.1.

Definition 2.23 of an $\mathcal{H}$-matrix includes the parameter k, which denotes an upper bound for the ranks of admissible blocks. Instead of using a fixed rank for all admissible blocks, we use local adaptive ranks, i. e. for a desired relative accuracy $\delta_{\mathcal{H}} \in (0,1)$, each admissible matrix block B is truncated to B_{k_B} of rank k_B such that it satisfies

$$\|B - B_{k_B}\|_2 \leq \delta_{\mathcal{H}} \|B\|_2, \tag{4.1}$$

where the rank k_B is given by $(\sigma_j, j = 1, \ldots, \min\{m,n\}$, denote the singular values of $B \in \mathbb{R}^{m \times n}$, and we set $\sigma_{\min\{m,n\}+1} := \delta_{\mathcal{H}} \sigma_1)$

$$k_B := \min\{k \in \mathbb{N}_0 \mid \sigma_{k+1} \leq \delta_{\mathcal{H}} \sigma_1\}.$$

If not noted otherwise, we use the default relative truncation accuracy of $\delta_{\mathcal{H}} = 10^{-10}$. The particular leaf size $n_{min} \in \{40, 50, 100\}$ (see Definition 2.20) that we are using in the examples is stated in the respective figure captions.

To evaluate the accuracy and orthogonality of the computed QR factors, we approximate

$$e_{approx} = \|A - QR\|_2 \qquad \text{and} \qquad e_{orth} = \|Q^T Q - I\|_2 \tag{4.2}$$

by the square root of the eigenvalue approximation obtained by 20 steps of the power iteration applied to the matrix $(A-QR)^*(A-QR)$ (or $(Q^TQ-I)^*(Q^TQ-I)$, respectively. Using more than 20 steps did not change the results in a significant way.

We compare the proposed Householder QR algorithm in $\mathcal{H}$-arithmetic to the following alternative QR algorithms:

- a Householder QR algorithm in $\mathcal{H}$-arithmetic using the explicit T instead of its reduced form $\overline{T}$,
- a Householder QR algorithm in standard (dense) arithmetic applied to the dense version of the original matrix,
- a Gram-Schmidt version for $\mathcal{H}$-matrices [10] as described in Subsection 3.1.2 (in [10], the QR decomposition based on Gram-Schmidt was generally seen as superior to the Cholesky-based QR decomposition described in Subsection 3.1.1 and to the variant based on the orthogonalization of an LU decomposition, which we described in Subsection 3.1.3 and do not test here),
- a Cholesky-based $\mathcal{H}$-QR decomposition [66] as described in Subsection 3.1.1 (only

applied to the the BEM and the random $\mathcal{H}$-matrices due to the high condition number of the RBF $\mathcal{H}$-matrices).

Using the induced admissibility function leads to new $\mathcal{H}$-matrix structures in Y, R and $\overline{T}$ representing the QR factors. In order to assess the effect of using this adaptive structure, we perform all numerical tests twice: once using the $\mathcal{H}$-block structure of A also for the matrices $Y, \overline{T}, R$ (as well as for the auxiliary matrices S_1 and S_2 in Algorithm 3.9), and once using the induced admissibility function to determine $\mathcal{H}$-structures for all computed matrices. In all subsequent figures, the results obtained using the adaptive structure are plotted as solid lines, whereas results obtained using the fixed (original) $\mathcal{H}$-matrix structure are plotted as dashed lines.

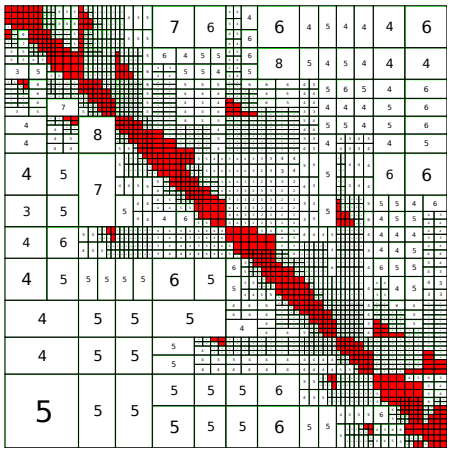
The algorithms are implemented as an extension to the H2Lib library [41]. Some routines, e. g. those for the multiplication and addition of $\mathcal{H}$-matrices, had to be rewritten in order to adjust $\mathcal{H}$-block structures "on the fly" to be consistent with the induced admissibility function. Furthermore, new (matrix-matrix) multiplication routines had to be added to incorporate the multiplication with T when only its reduced representation $\overline{T}$ is known (see Remark 2.16 and the corresponding Algorithms 3.11 and 3.12). All algorithms use calls to BLAS and LAPACK in the OpenBLAS implementation to efficiently compute the basic dense operations. In particular, we use the LAPACK routine GEQRT2 to compute the (dense) QR decomposition at the start of the recursion in the Householder QR Algorithm 3.7 (line 10). The computations were done on one core of an i5-6600K CPU with 3.50 GHz and sufficient RAM to store all used matrices.

We will use the Landau notation to state computational and storage complexities that we deduce from our numerical experiments. For better readability, we do not show all plots for all examples if they do not offer any new insights.

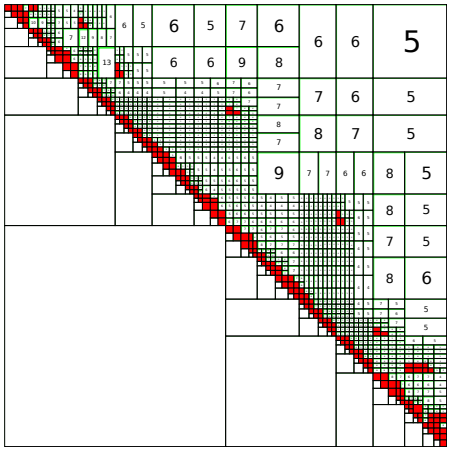
4.1 Square Matrices

4.1.1 2D Laplace Operator

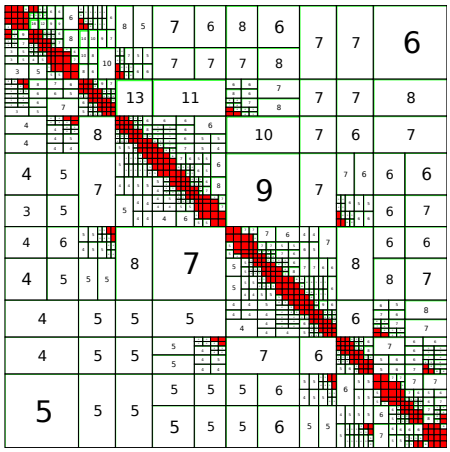
In Figure 4.2, we show the $\mathcal{H}$-matrix structures for the matrices $\overline{T}, Y$ and R that represent the QR factors of a 2D Laplacian matrix computed by the proposed Householder QR algorithm when using the adaptive $\mathcal{H}$-matrix structure ((a) and (b)) as well as when using the original $\mathcal{H}$-matrix structure ((c) and (d)). Inadmissible blocks are marked in red, whereas admissible blocks show the rank of the factored representation in this block. The induced admissibility function leads to a finer subdivision but still exhibits larger off-diagonal admissible blocks, which indicates that the underlying matrix data is amenable for an $\mathcal{H}$-matrix representation. More rigorous analyses for the storage requirements of the involved matrices will follow in Figure 4.4.



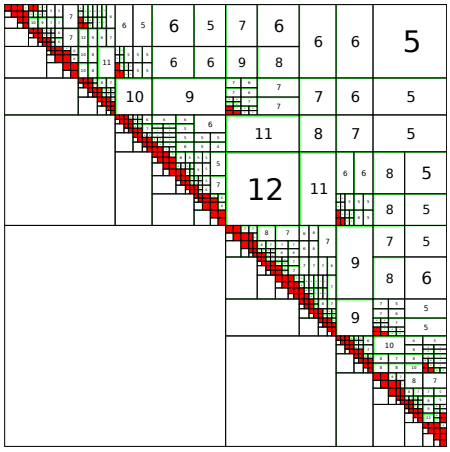
(a) Y and R.



(b) $\overline{T}$.



(c) Y and R.



(d) $\overline{T}$.

Figure 4.2. QR factors of the 2800×2800 Laplacian $\mathcal{H}$-matrix A shown in Figure 4.1(a). (a), (b) show $Y, R, \overline{T}$ with the adaptive $\mathcal{H}$-matrix structure, (c), (d) show $Y, R, \overline{T}$ with the fixed (original) $\mathcal{H}$-matrix structure of A.

In Figure 4.3, we compare various QR algorithms with respect to their computational time as well as their accuracy and orthogonality (4.2) of the computed QR factors. In the top left plot, we see that all $\mathcal{H}$-QR algorithms have a lower computational complexity than the computation in exact (dense) arithmetic, which is $\mathcal{O}(n^3)$. In this example, using the adaptive $\mathcal{H}$-matrix structure typically leads to larger computational times than using the original $\mathcal{H}$-matrix structure. A possible explanation is that the induced admissibility function may lead to a refinement of a large block that is admissible in the original $\mathcal{H}$-matrix into many small (admissible and inadmissible) subblocks such that instead of generating computational savings, one actually increases the computational time and storage requirements. This seems to be more pronounced for the Gram-Schmidt and

the Cholesky QR algorithms, which both compute an $\mathcal{H}$-matrix approximation to the orthogonal factor Q in explicit form.

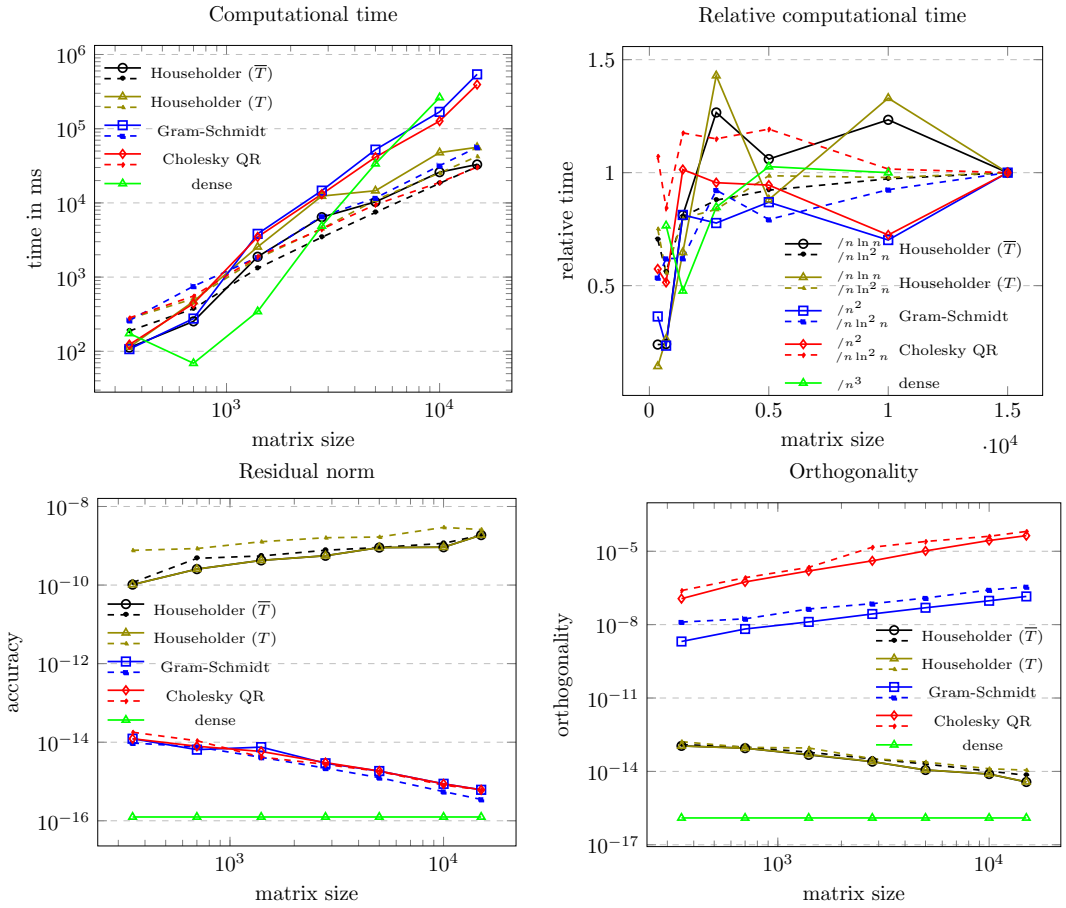


Figure 4.3. Computational time, accuracy and orthogonality for the QR factorization of 2D Laplacian $\mathcal{H}$-matrices (with $n_{min} = 50$, truncation accuracy $\delta_{\mathcal{H}} = 10^{-10}$). Continuous lines show results for the adaptive $\mathcal{H}$-matrix structure, dashed lines for the fixed (original) $\mathcal{H}$-matrix structure.

The Householder version with the reduced $\overline{T}$ is the fastest method (and in particular always faster than the one with the explicit T). In the top right plot, we show computational times scaled by their presumed complexity (and scaled by constants such that all lines end in 1). We conclude that the computational complexities of the $\mathcal{H}$-QR algorithms lie between $\mathcal{O}(n \ln n)$ (Householder ($\overline{T}$) with original $\mathcal{H}$-structure) and $\mathcal{O}(n^2)$ (Gram-Schmidt with adaptive $\mathcal{H}$-matrix structure).

The bottom left plot shows the accuracy of the QR factors. Since we use adaptive ranks with $\delta_{\mathcal{H}} = 10^{-10}$ (4.1) in our $\mathcal{H}$-arithmetic, we are satisfied that the Householder $(\overline{T}, T)$ methods have an accuracy of $\mathcal{O}(10^{-9})$. The Gram-Schmidt and Cholesky QR

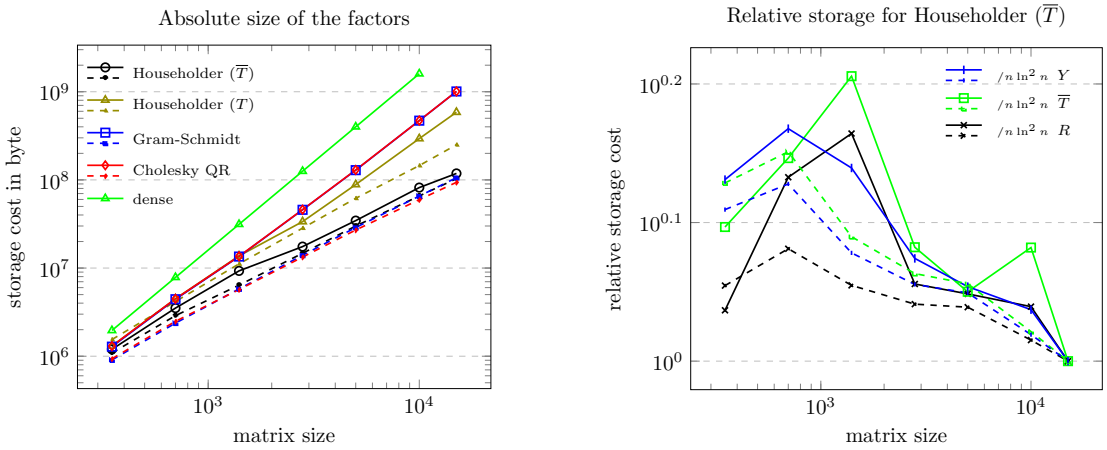


Figure 4.4. Storage requirements of the QR factors of 2D Laplacian $\mathcal{H}$-matrices (with $n_{min} = 50$, truncation accuracy $\delta_{\mathcal{H}} = 10^{-10}$). Continuous lines show results for the adaptive $\mathcal{H}$-matrix structure, dashed lines for the fixed (original) $\mathcal{H}$-matrix structure.

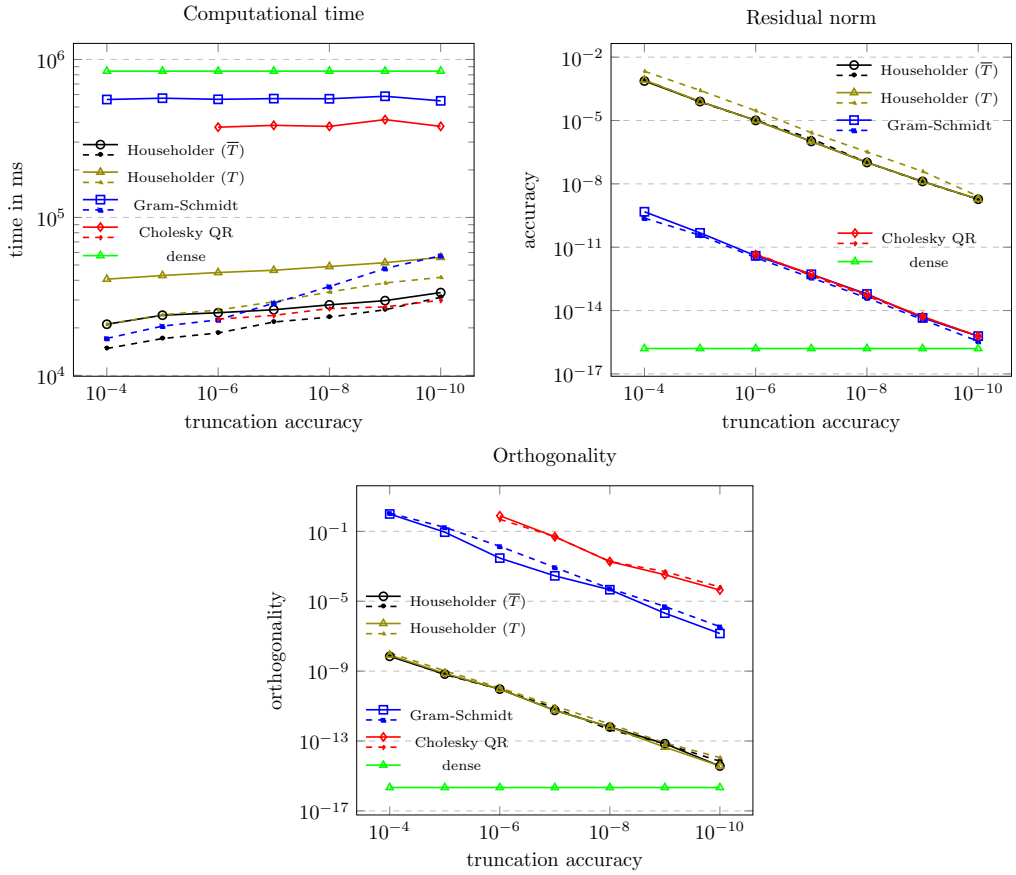


Figure 4.5. Effect of truncation accuracy $\delta_{\mathcal{H}}$ in $\mathcal{H}$-arithmetic on computational time, accuracy and orthogonality, illustrated for a 15000×15000 $2D$ Laplacian $\mathcal{H}$-matrix with leaf size 50. Continuous lines show results for the adaptive $\mathcal{H}$-matrix structure, dashed lines for the fixed (original) $\mathcal{H}$-matrix structure.

methods even have an accuracy of $\mathcal{O}(10^{-15})$. We cannot offer an explanation of why it is substantially lower and even decreases with increasing matrix size. This behavior is not repeated for our other test matrices.

The orthogonality (bottom right) shows a different picture: The Householder $(\overline{T}, T)$ QR factors retain their orthogonality for an increase in matrix size, whereas the Gram-Schmidt and Cholesky QR factorizations are less stable. In Figure 4.5, we will analyze in more detail the effect of the relative accuracy $\delta_{\mathcal{H}}$ of the $\mathcal{H}$-arithmetic on the accuracy and orthogonality of the QR factors.

In Figure 4.4, we analyze the storage requirements for the QR factors resulting from the different algorithms. The results are qualitatively analogous to those for computational times: Shown on the left, all $\mathcal{H}$-matrix QR algorithms require less storage than using dense matrices, the original $\mathcal{H}$-structure is preferable over the adaptive $\mathcal{H}$-matrix structure, and lowest storage cost are achieved for the Householder version with the reduced matrix $\overline{T}$. On the right, we show the storage costs of the individual matrices $Y, \overline{T}, R$ of the Householder $(\overline{T})$ algorithm. We do not show our plots where storage is scaled by the presumed complexity but only summarize our findings: Using the original $\mathcal{H}$-structure, the storage complexities are $\mathcal{O}(n \ln^2 n)$, and using the adaptive $\mathcal{H}$-matrix structure for Gram-Schmidt or Cholesky-QR increases the storage complexity to about $\mathcal{O}(n^{1.8})$.

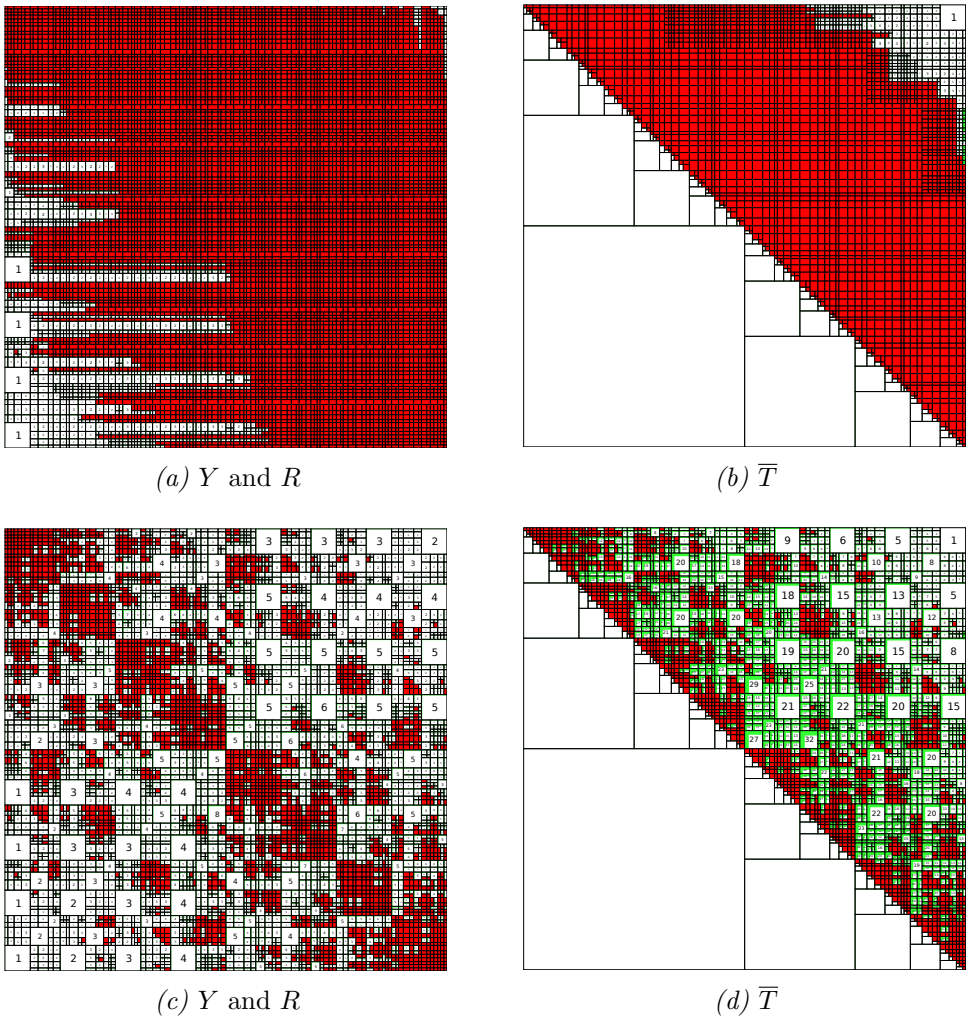
In Figure 4.5, we show the effect of the adaptive $\mathcal{H}$-accuracy $\delta_{\mathcal{H}}$ on the computational time as well as the accuracy and orthogonality of the QR factors for a matrix of fixed size. The increase in computational time, as $\delta_{\mathcal{H}}$ decreases from 10^{-4} down to 10^{-10}, appears to be moderate compared to the benefits with respect to accuracy and orthogonality. For truncation accuracies larger than 10^{-6}, Cholesky QR suffers from breakdowns. We have not included the plot for storage requirements, since it is qualitatively analogous to the (left) plot for computational time.

Interestingly, the 2D Laplace matrix does not satisfy the conditions from Subsection 3.3.2 and, as can be seen in Figure 4.2, does not exhibit the increase in inadmissible blocks in Y as described in Theorem 3.14. Hence, although the $\mathcal{H}$-matrix bandwidth is high, a QR decomposition can be computed efficiently.

4.1.2 3D Laplace Operator

In this section we present numerical results for the QR factorization of 3D boundary element Laplacian $\mathcal{H}$-matrices. Different to the results for the 2D case, we observe a strong increase in inadmissible blocks in the resulting QR factors when using the adaptive instead of the original structure in the block Householder approach. Figure 4.6 shows the resulting $\mathcal{H}$-matrices of the QR factorization for adaptive as well as original $\mathcal{H}$-matrix structures. For higher truncation accuracies this implies (very) high ranks in

many of the admissible blocks of the original structure. Both the Gram-Schmidt and the Cholesky QR approach offer no relief in this regard as they suffer from the same problems.



(a) Y and R *(b)* $\overline{T}$

(c) Y and R *(d)* $\overline{T}$

Figure 4.6. QR factors of the 3000×3000 3D Laplacian $\mathcal{H}$-matrix A shown in Figure 4.1(c). (a), (b) show $Y, R, \overline{T}$ with the adaptive $\mathcal{H}$-matrix structure, (c), (d) show $Y, R, \overline{T}$ with the fixed (original) $\mathcal{H}$-matrix structure of A. Different to the other examples we use a truncation accuracy of $\delta_H = 10^{-4}$.

Figure 4.7 shows the computational time as well as the residual and orthononality of the computed QR factors. In these tests we used the truncation accuracy $\delta_{\mathcal{H}} = 10^{-4}$ (compared to $\delta_{\mathcal{H}} = 10^{-10}$ in all other experiments), which appeared to be a good compromise between computational work and obtained accuracy. In the left plot, the slopes of the lines for results using the $\mathcal{H}$-QR factorizations appear to be smaller that the slope of the line representing the dense QR factorization, indicating a lower computational

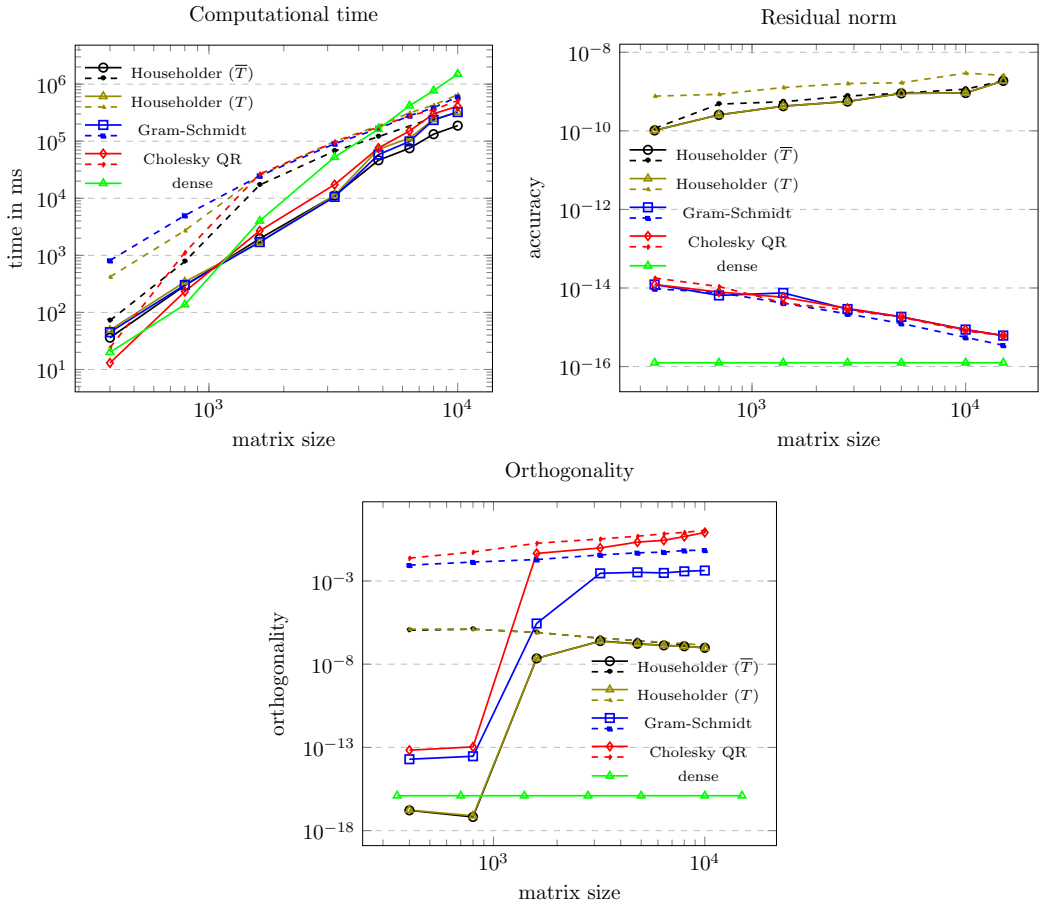


Figure 4.7. Computational time, accuracy and orthogonality for the QR factorization of 3D Laplacian $\mathcal{H}$-matrices (with $n_{\text{leaf}} = 40$, truncation accuracy $\delta_{\mathcal{H}} = 10^{-4}$). Continuous lines show results for the adaptive $\mathcal{H}$-matrix structure, dashed lines for the fixed (original) $\mathcal{H}$-matrix structure.

complexity than $\mathcal{O}(n^3)$ for the $\mathcal{H}$-variants.

In Figure 4.8, we show results for a matrix of fixed size but with varying truncation accuracy $\delta_{\mathcal{H}}$. An interesting observation is that for truncation accuracies $\delta_{\mathcal{H}} > 10^{-8}$ using the original $\mathcal{H}$-structure (shown in dashed lines) leads to higher computational times (left plot) but lower storage costs (bottom plot). This may be explained by the computation of dense blocks as intermediate results, which are truncated to low rank format only in the end. In this case, using improvements for the $\mathcal{H}$-matrix products as developed in [22] could possibly lead to faster computational times.

Otherwise, the results are as expected. The Cholesky QR approach fails for a truncation accuracy of 10^{-2} because due to the low accuracy, the arising matrices are no longer

positive definite.

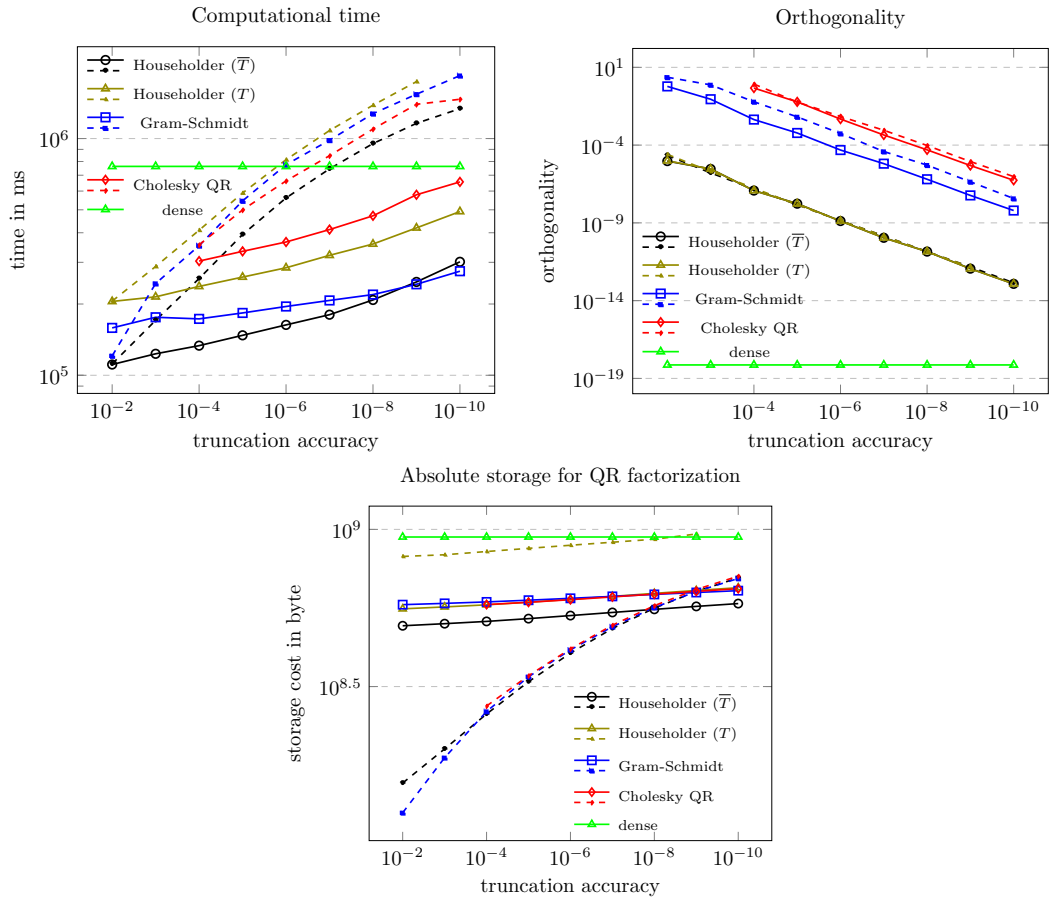
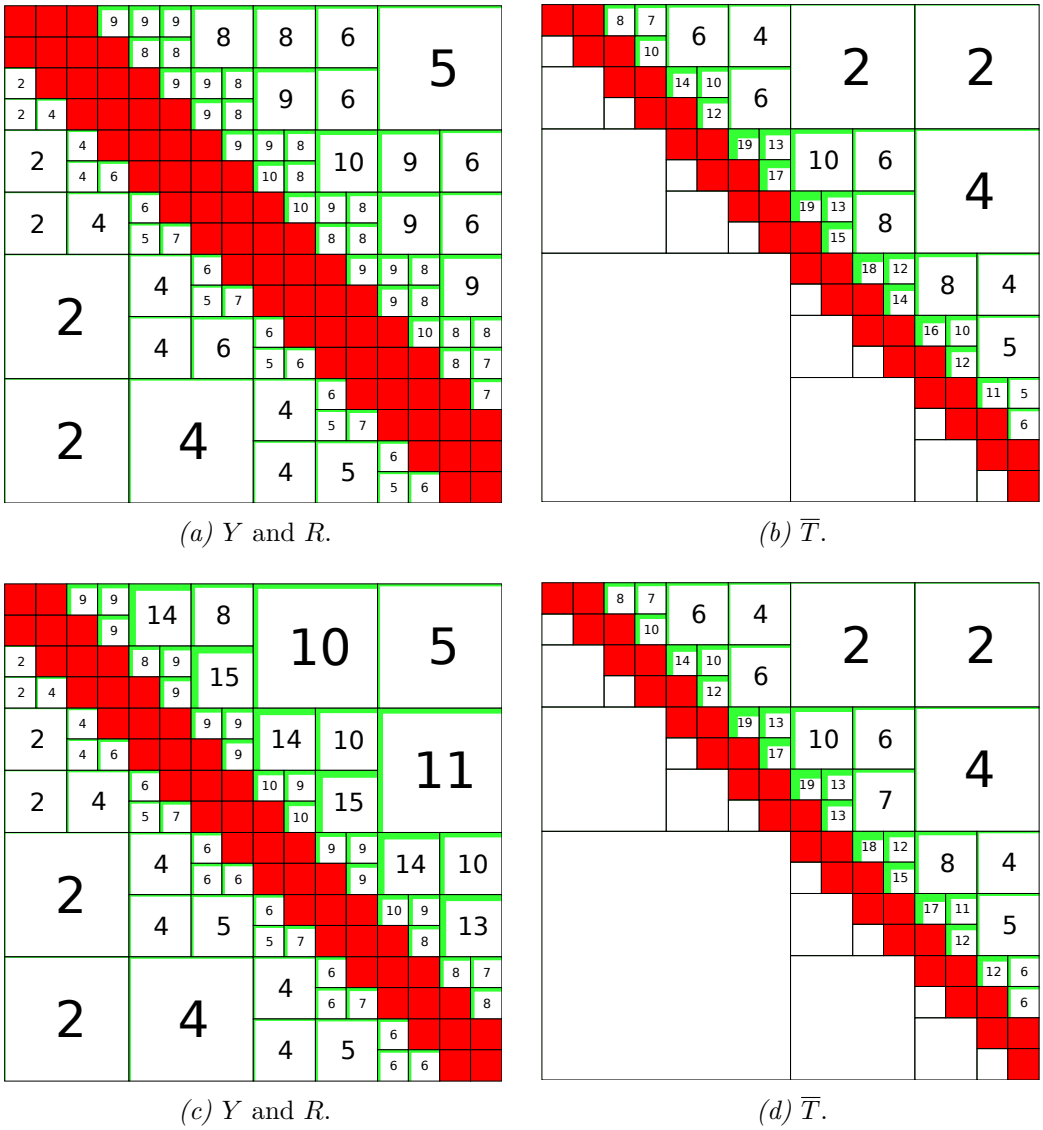


Figure 4.8. Effect of truncation accuracy $\delta_{\mathcal{H}}$ in $\mathcal{H}$-arithmetic on computational time, orthogonality and storage, illustrated for a 8000×8000 $3D$ Laplacian $\mathcal{H}$-matrix with leaf size 40. Continuous lines show results for the adaptive $\mathcal{H}$-matrix structure, dashed lines for the fixed (original) $\mathcal{H}$-matrix structure.

4.1.3 1D RBF Interpolation

In this section, we show numerical results for the QR factorization of matrices that arise in 1D RBF interpolation. While this may appear as a toy problem, there are some interesting observations that we wish to report. We begin with Figure 4.9, which shows the resulting $\mathcal{H}$-matrix structures for the matrices Y, R and $\overline{T}$ representing the QR factors. For this matrix, there are only some minor differences between those obtained for the adaptive $\mathcal{H}$-matrix structure and those in the fixed $\mathcal{H}$-matrix structure. In particular, in (a) we see four inadmissible blocks per block row on the finest level, whereas in (c) there are only three.



(a) Y and R. *(b)* $\overline{T}$. *(c)* Y and R. *(d)* $\overline{T}$.

Figure 4.9. QR factors of the 1000×1000 1D RBF interpolation $\mathcal{H}$-matrix shown in Figure 4.1(b). (a), (b) show $Y, R, \overline{T}$ with the adaptive $\mathcal{H}$-matrix structure, (c), (d) show $Y, R, \overline{T}$ with the fixed (original) $\mathcal{H}$-matrix structure.

In Figure 4.10, we show numerical results for the computational time, accuracy and orthogonality of the QR factors obtained with different QR algorithms. We see that the Householder ($\overline{T}$) algorithm is the fastest and still obtains the best results for accuracy and orthogonality among the $\mathcal{H}$-matrix QR algorithms. In fact, the factor Q computed by Gram-Schmidt is no longer orthogonal. There are no results shown for the Cholesky-QR since the Cholesky factorization failed (the RBF matrix is highly ill-conditioned). The complexity of the Householder based algorithms appears to be $\mathcal{O}(n \ln^4 n)$. We do

not include plots for the storage requirements, since they are qualitatively analogous to the computational timings. Numerical tests for different adaptive $\mathcal{H}$-accuracies, also not shown here, were qualitatively similar to the ones for the Laplace problem shown in Figure 4.5.

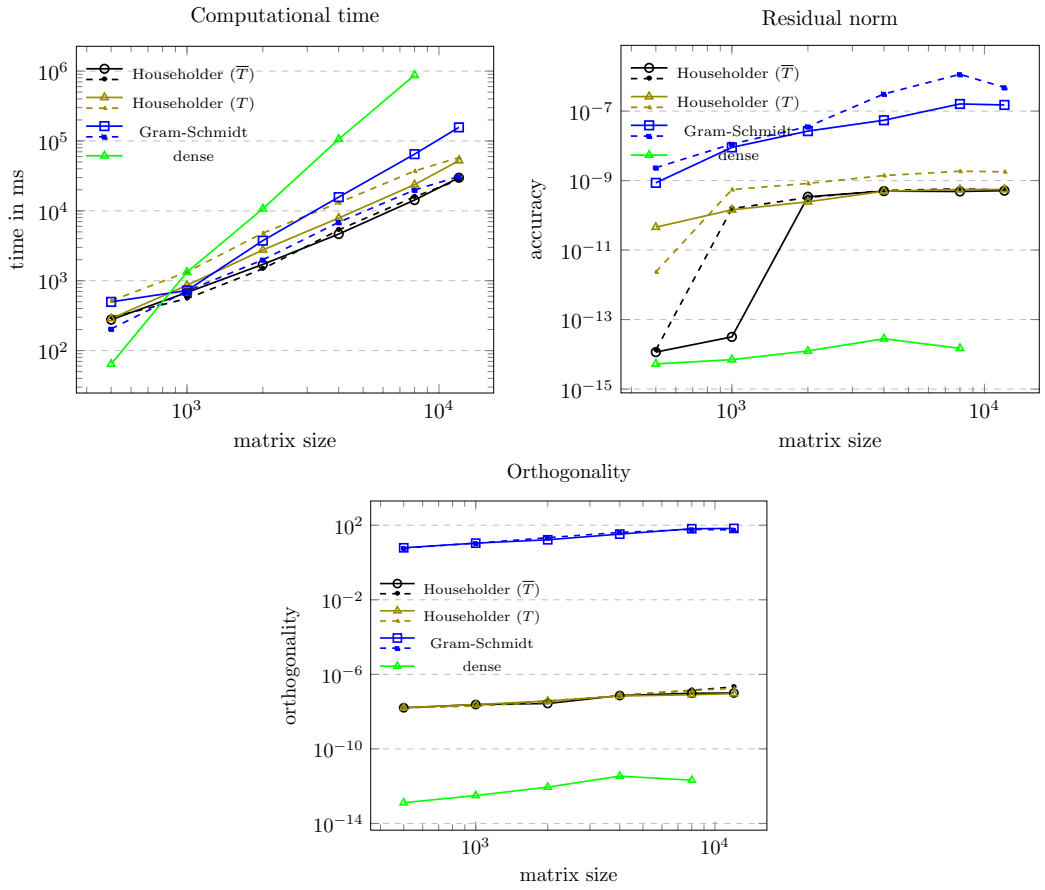
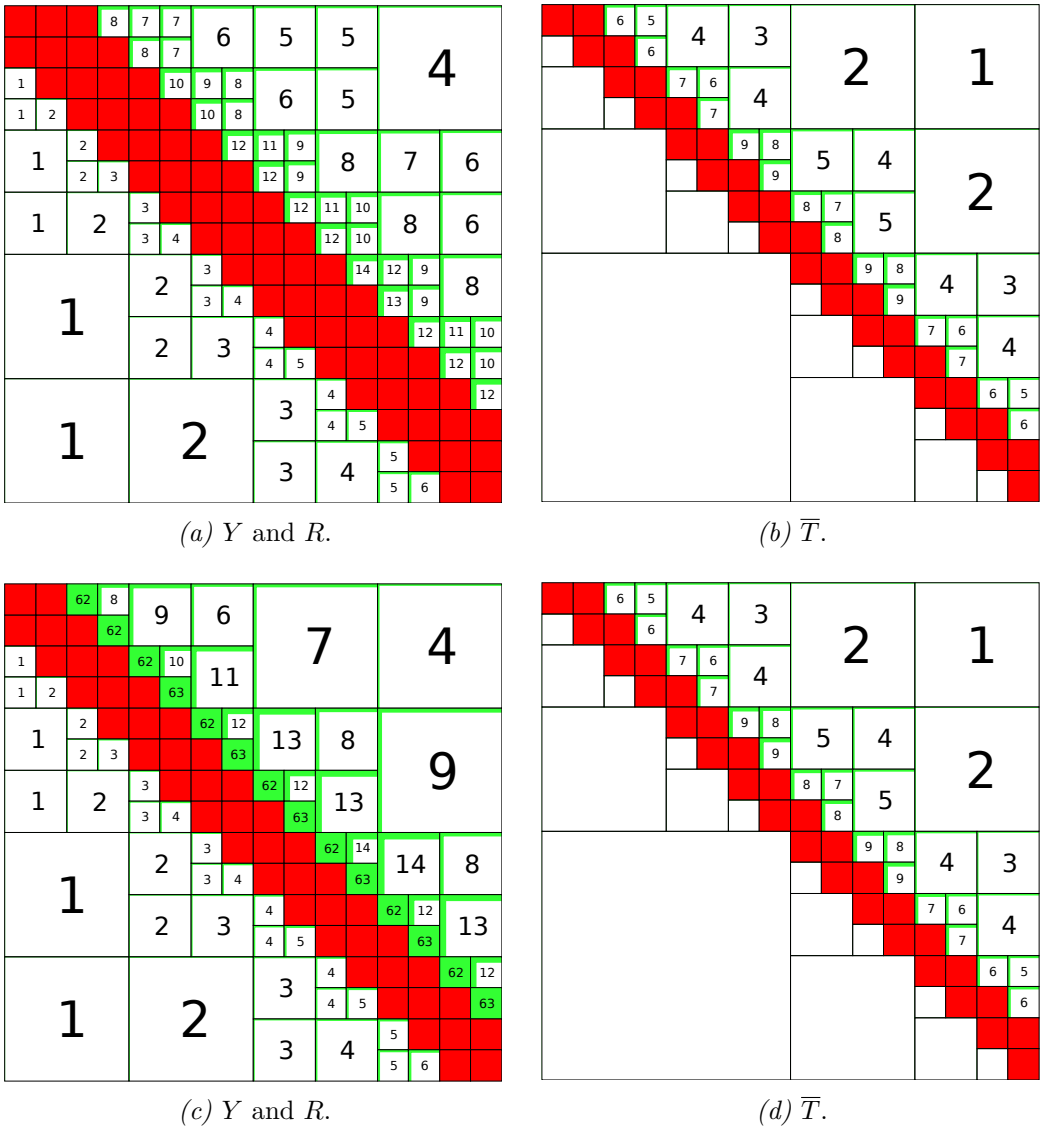


Figure 4.10. Computational time, accuracy and orthogonality for the QR factorization of 1D RBF interpolation $\mathcal{H}$-matrices (with $n_{min} = 100$, truncation accuracy $\delta_{\mathcal{H}} = 10^{-10}$). Continuous lines show results for the adaptive $\mathcal{H}$-matrix structure, dashed lines for the fixed (original) $\mathcal{H}$-matrix structure.

We now keep the $\mathcal{H}$-matrix structure of the 1D RBF interpolation problem but fill the matrix (i. e. its inadmissible subblocks and the low rank factors in admissible blocks) with random data. The resulting QR factors are shown in Figure 4.11. The main difference to the QR factors for an actual RBF interpolation matrix as shown in Figure 4.9 lies in the now much larger ranks in admissible blocks of the original $\mathcal{H}$-structure in the second upper off-diagonal block row of the matrix R shown by solid green blocks in Figure 4.11(c). In the adaptive $\mathcal{H}$-matrix structure, these blocks had been marked as inadmissible, and given their high ranks shown in (c), a treatment as admissible blocks is not efficient.



(a) Y and R. *(b)* $\overline{T}$. *(c)* Y and R. *(d)* $\overline{T}$.

Figure 4.11. QR factors of the randomized 1000×1000 1D RBF interpolation $\mathcal{H}$-matrix shown in Figure 4.1(b). (a), (b) show $Y, R, \overline{T}$ with the adaptive $\mathcal{H}$-matrix structure, (c), (d) show $Y, R, \overline{T}$ with the fixed (original) $\mathcal{H}$-matrix structure.

This is also reflected in Figure 4.12, which shows the computational time for the computation of the QR factors, their accuracy and orthogonality for the different QR algorithms applied to this matrix. Here, using the adaptive $\mathcal{H}$-matrix structure leads to a lower computational time for the Householder ($\overline{T}$) and Gram-Schmidt algorithms, with the Householder ($\overline{T}$) algorithm being the fastest one with computational complexity $\mathcal{O}(n^{1.7})$. Results for storage requirements (not shown here) are qualitatively analogous to those for the computational time. The Householder ($\overline{T}$) also yields the best results

among the $\mathcal{H}$-matrix algorithms with respect to accuracy and orthogonality. Both the Gram-Schmidt and Cholesky QR algorithms compute QR factors whose accuracy and orthogonality deteriorate with increasing matrix size, Cholesky QR being worse than Gram-Schmidt. However, they no longer fail entirely as they did for the 1D RBF interpolation matrix due to its high condition number (see Figure 4.10).

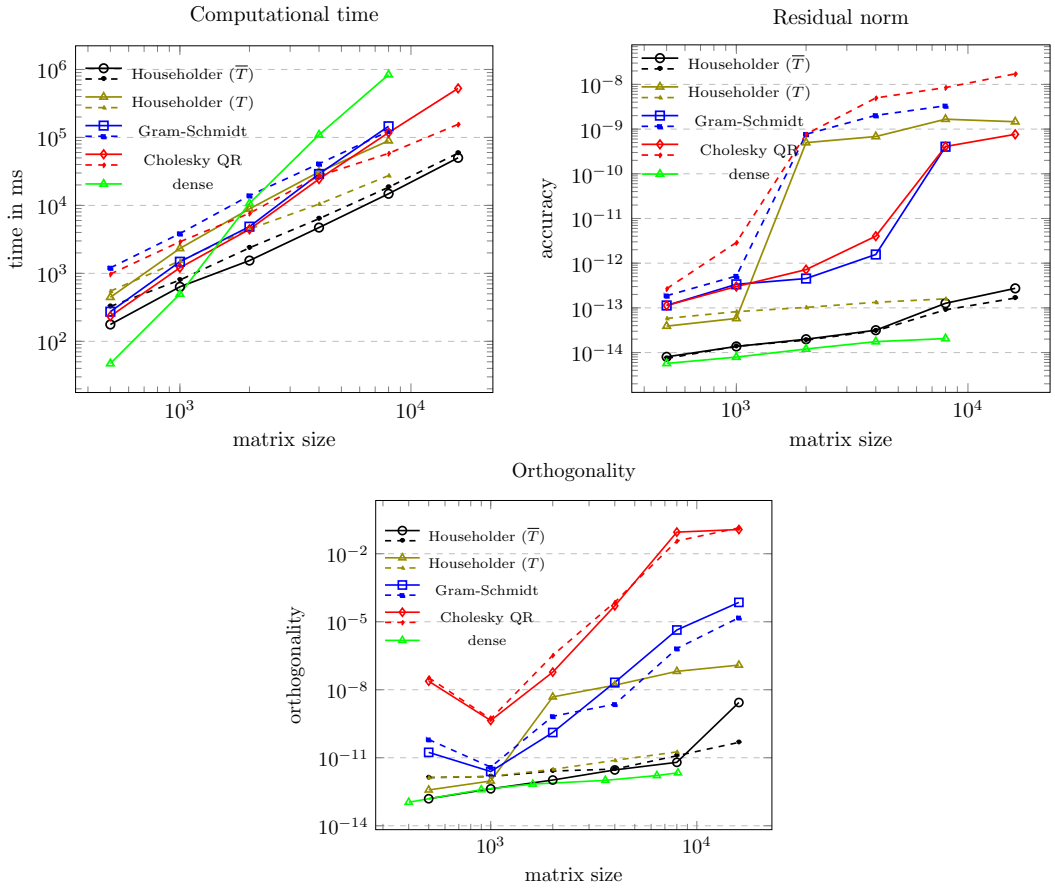
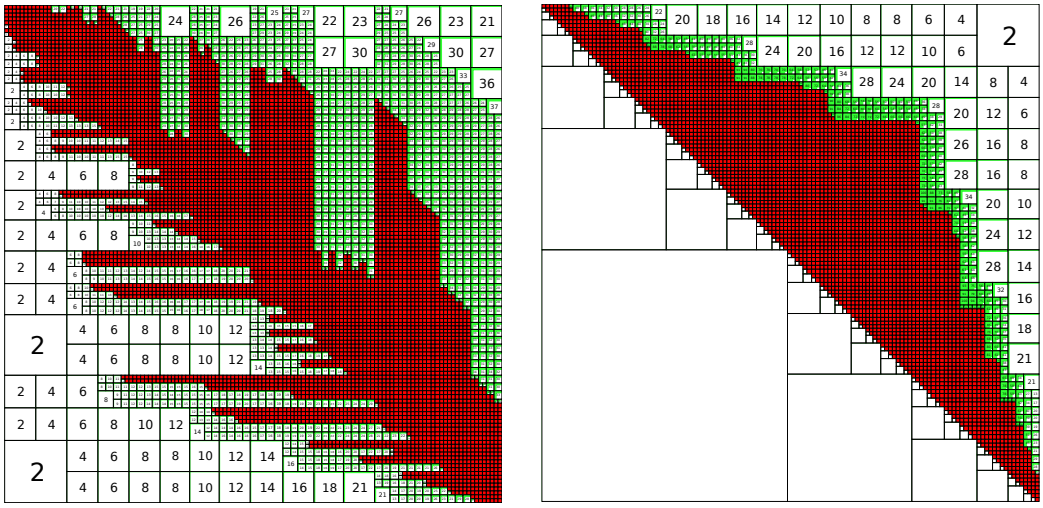


Figure 4.12. Computational time, accuracy and orthogonality for the QR factorization of randomized 1D RBF interpolation $\mathcal{H}$-matrices (with $n_{min} = 100$, truncation accuracy $\delta_{\mathcal{H}} = 10^{-10}$). Continuous lines show results for the adaptive $\mathcal{H}$-matrix structure, dashed lines for the fixed (original) $\mathcal{H}$-matrix structure.

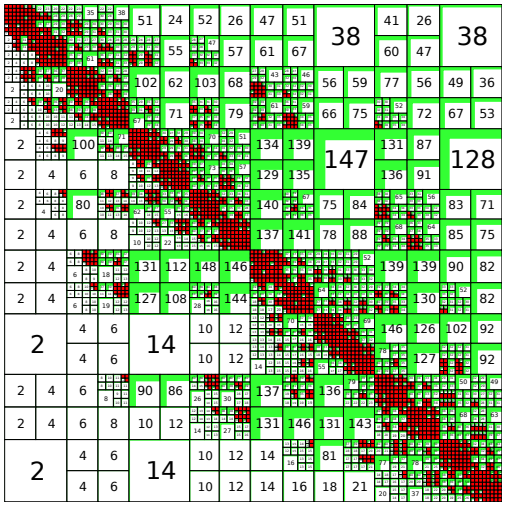
Different to the 2D Laplace case, this time, both the RBF interpolation matrix and its randomized version satisfy the conditions from the relevant Theorems in Subsection 3.3.2. However, due to the small $\mathcal{H}$-matrix bandwidth, this does not have a negative influence on the performance of the block Householder approach. However, it explains why the block Householder approach that uses the unreduced form T performs so much worse. The induced admissibility function of T contains only inadmissible blocks above the diagonal.

Furthermore, also the conditions of Theorem 3.18 are satisfied, which explains the very poor performance of the Gram-Schmidt and Cholesky QR approach that tracks the induced admissibility. Q also consists of only inadmissible blocks above the diagonal.

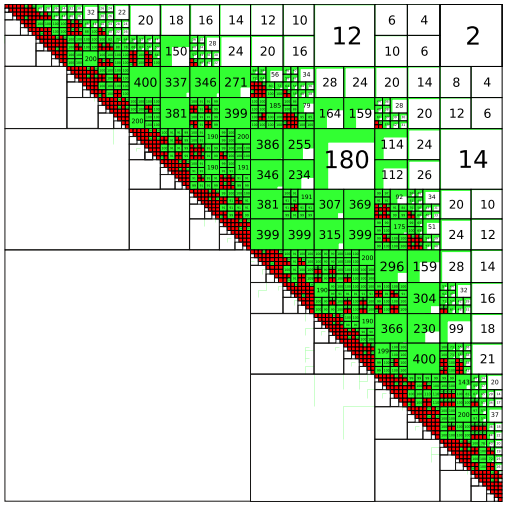


(a) Y and R.

(b) $\overline{T}$.



(c) Y and R.



(d) $\overline{T}$

Figure 4.13. QR factors of the 6400×6400 2D RBF interpolation $\mathcal{H}$-matrix shown in Figure 4.1(c). (a), (b) show $Y, R, \overline{T}$ with the adaptive $\mathcal{H}$-matrix structure, (c), (d) show $Y, R, \overline{T}$ with the fixed (original) $\mathcal{H}$-matrix structure.

4.1.4 2D RBF Interpolation

In this subsection, we show numerical results for $\mathcal{H}$-QR algorithms applied to matrices from 2D RBF interpolation problems. In Figure 4.13, we display the resulting $\mathcal{H}$-matrix structures. While using the adaptive $\mathcal{H}$-matrix structure leads to many inadmissible blocks that grow from the diagonal into the off-diagonal blocks, using the original $\mathcal{H}$-matrix leads to rather high ranks. Both observations indicate that the matrices R and $\overline{T}$ are less suitable to be approximated in $\mathcal{H}$-matrix format and the computational complexity is expected to become worse than in the previous examples.

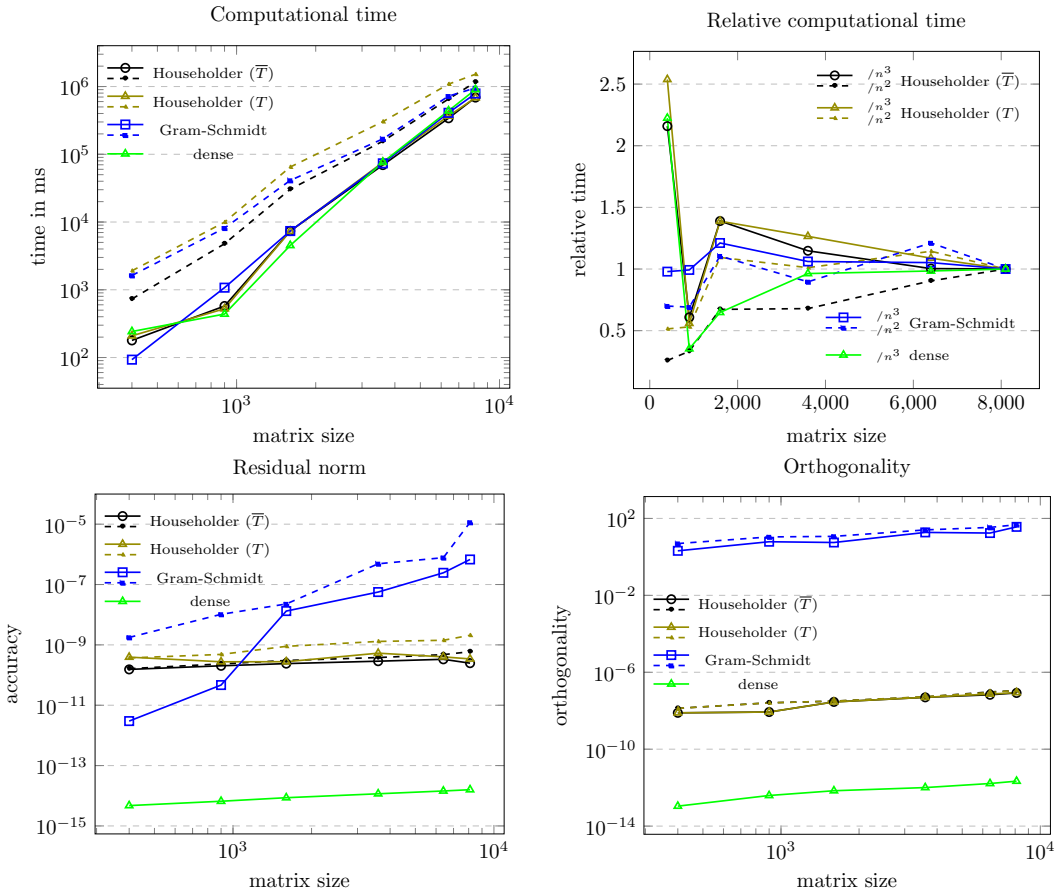
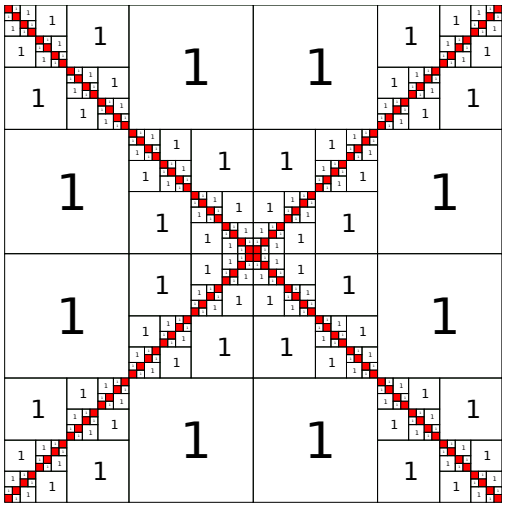


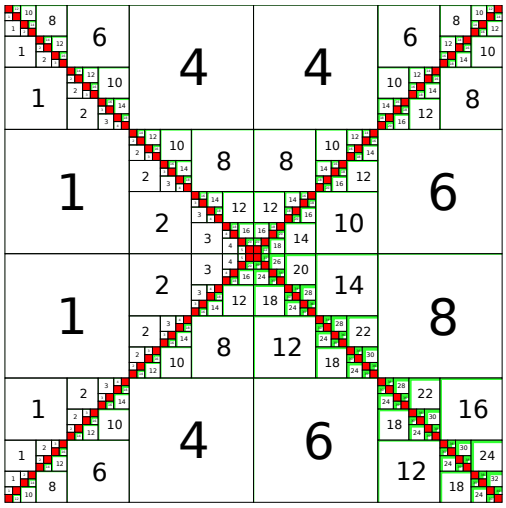
Figure 4.14. Computational time, accuracy and orthogonality for the QR factorization of $2D$ RBF interpolation $\mathcal{H}$-matrices (with $n_{min} = 40$, truncation accuracy $\delta_{\mathcal{H}} = 10^{-10}$). Continuous lines show results for the adaptive $\mathcal{H}$-matrix structure, dashed lines for the fixed (original) $\mathcal{H}$-matrix structure.

This reflects our theoretical results. The 2D RBF interpolation matrices satisfy the conditions in Subsection 3.3.2 and simultaneously have a rather large $\mathcal{H}$-matrix bandwidth. Hence, the factors in the QR decomposition likely do not allow for a good $\mathcal{H}$-matrix

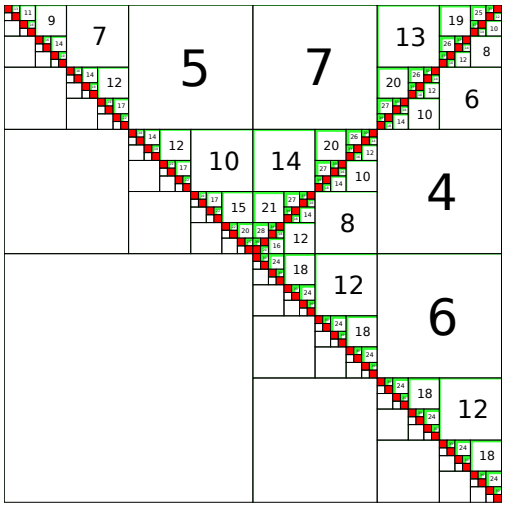
approximation.



(a) Original.



(b) Y and R.



(c) $\overline{T}$.

Figure 4.15. Random 8000 × 8000 X-like $\mathcal{H}$-matrix shown in Figure (a). (b), (c) show $Y, R, \overline{T}$ after computation by the block Householder approach.

Figure 4.14 shows the computational times to compute the QR factorization with different algorithms as well as the respective accuracy and orthogonality of the computed QR factors. In this example, using the induced admissibility function is faster than using the original $\mathcal{H}$-matrix structure. The computational complexity for the $\mathcal{H}$-algorithms using the original $\mathcal{H}$-matrix structure appears to be around $\mathcal{O}(n^2)$, whereas the complexity using the induced admissibility appears to be even higher. Results for accuracy and orthogonality confirm stability for the Householder $(T, \overline{T})$ algorithms, whereas the

Gram-Schmidt algorithm fails to compute an orthogonal matrix Q.

We do not show plots for storage requirements, since they are qualitatively analogous to the computational times. Among the matrices $Y, \overline{T}$ and R representing the QR factors in the Householder algorithm, the matrix R requires most of the storage and Y the least.

As for the 1D RBF interpolation matrices, we have repeated our numerical tests for matrices that have the $\mathcal{H}$-matrix structures of the 2D RBF interpolation matrices but are now filled with random data. We do not present those results here but just report that they did not differ in any significant way from those using actual interpolation matrices. Our conclusion is that the QR factors of the 2D RBF interpolation matrix do not benefit from any additional structure intrinsic to the data of the interpolation problem (other than given by the $\mathcal{H}$-matrix structure).

4.1.5 Randomly Filled Hierarchical Matrix with X-like Structure

We repeat our analysis for a matrix where the inadmissible blocks form an X and all entries are filled with random values. An example is shown in Figure 4.15. Different to all other examples, we do not differentiate between using the fixed (original) and the induced $\mathcal{H}$-matrix structure because they both produce the same structure in the resulting QR factors for all our methods.

As can be seen in Figure 4.16, all examined methods can compute a QR decomposition in about the same time, but the quality of the block Householder result is superior.

4.2 Rectangular Matrices

In this section we test the computation of the QR decomposition for rectangular $\mathcal{H}$-matrices. There are some differences to the square case. The factor Y is no longer a simple lower triangular matrix but a permuted lower trapezoidal matrix, where the majority of the rows have only non-zero entries. That means most of the matrix blocks above the diagonal in the combined Y, R matrix actually belong to Y and not R. See Figure 3.3 from Subsection 2.2.1 for an illustration. Furthermore, we do not have an original structure available for the factor $\overline{T}$, which is always a square matrix, where the number of rows and columns is given by the number of columns of the original matrix. In our upcoming examples, we choose the adaptive structure for $\overline{T}$ in both the version of the Householder algorithms that uses the fixed (original) structure for Y and R as well as the one that uses adaptive structure for Y and R.

If our rectangular matrix consists of (only some) block columns of our square matrices

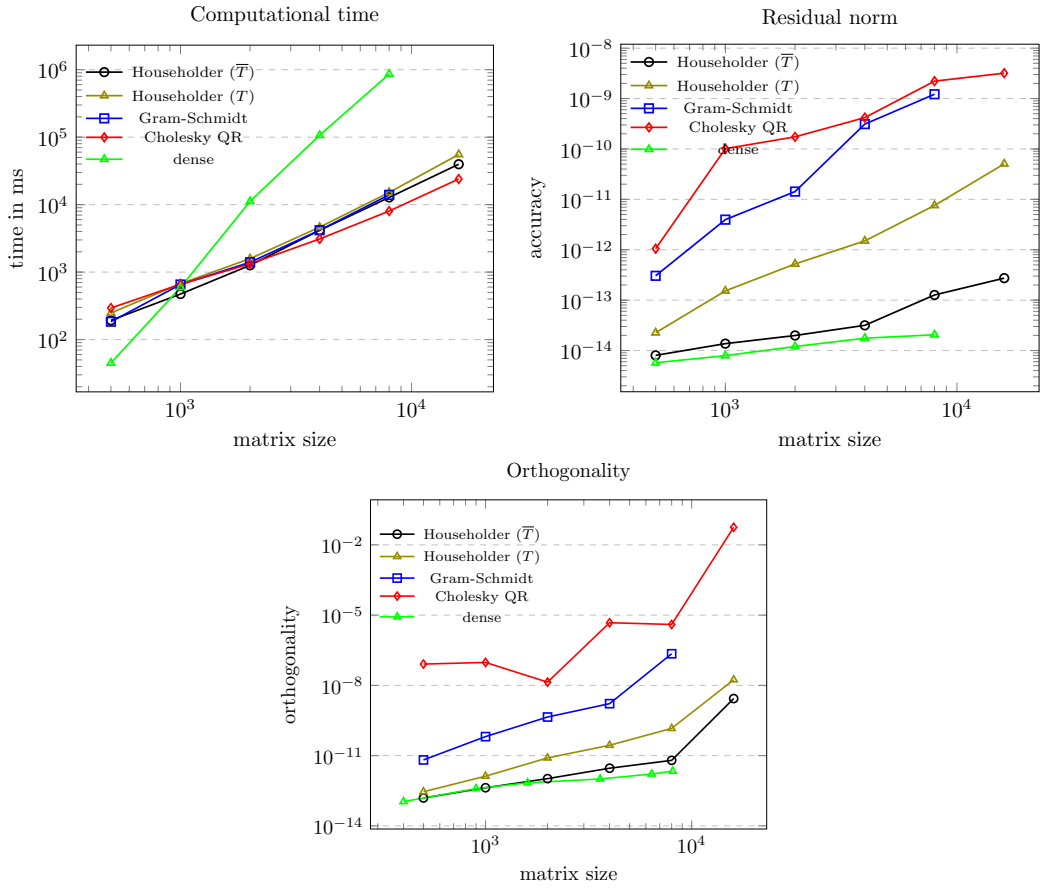


Figure 4.16. Computational time, accuracy and orthogonality for the QR factorization of X-like structured random $\mathcal{H}$-matrices (with $n_{min} = 100$, truncation accuracy $\delta_{\mathcal{H}} = 10^{-10}$).

from the previous subsections, we do not expect any further complications, at least for our block Householder approach. Due to the iterative nature of the algorithm that makes use of vertical splits, we indirectly tested this problem already in the previous section, and we omit to do it again. We will concentrate on rectangular $\mathcal{H}$-matrices, where the dense blocks are not only located in some of the block rows but in all of them. Hence, we expect certain complications regarding the efficiency of the calculations. See Subsection 3.3.7 for an overview of the possible problems.

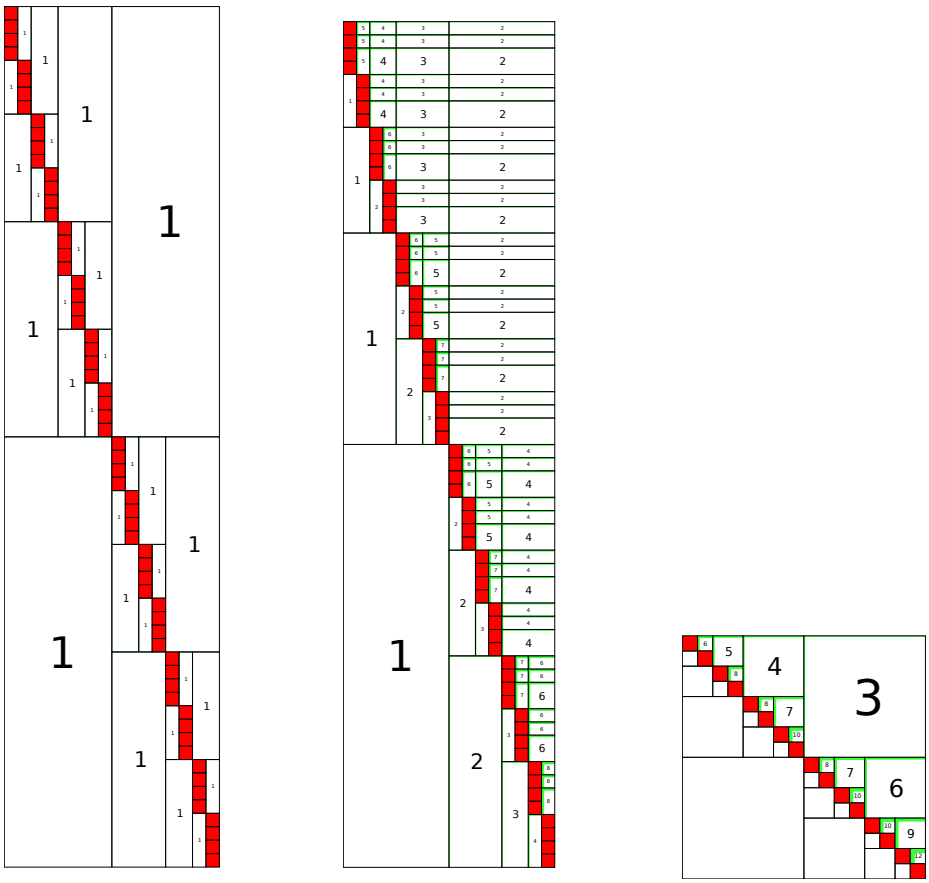
4.2.1 Random $\mathcal{H}_p$-Matrices

As our first example involving rectangular $\mathcal{H}$-matrices, we compute the QR decomposition of rectangular $\mathcal{H}_p$-matrices with random values. See Definition 2.51 for a formal

definition of square $\mathcal{H}_p$-matrices and the subsequent comments about a rectangular version. Similar to the situation in Subsection 4.1.5, we do not differentiate between using the fixed (original) and the adaptive $\mathcal{H}$-matrix structure because both result in the same structure for the QR factors.

This example was already mentioned in [60] but not explicitly computed. An illustration of such an $\mathcal{H}$-matrix can be seen in Figure 4.17.

Our numerical results show that both the block Householder approach and the Gram-Schmidt approach efficiently compute a QR decomposition of the given matrix. Interestingly, Gram-Schmidt produces a Q that is closer to an orthogonal matrix than the block Householder approach and is faster.



(a) Original. *(b)* Y and R. *(c)* $\overline{T}$.

Figure 4.17. Random 2560×640 $\mathcal{H}_p$-matrix shown in Figure (a). (b), (c) show $Y, R, \overline{T}$ after computation by the block Householder approach.

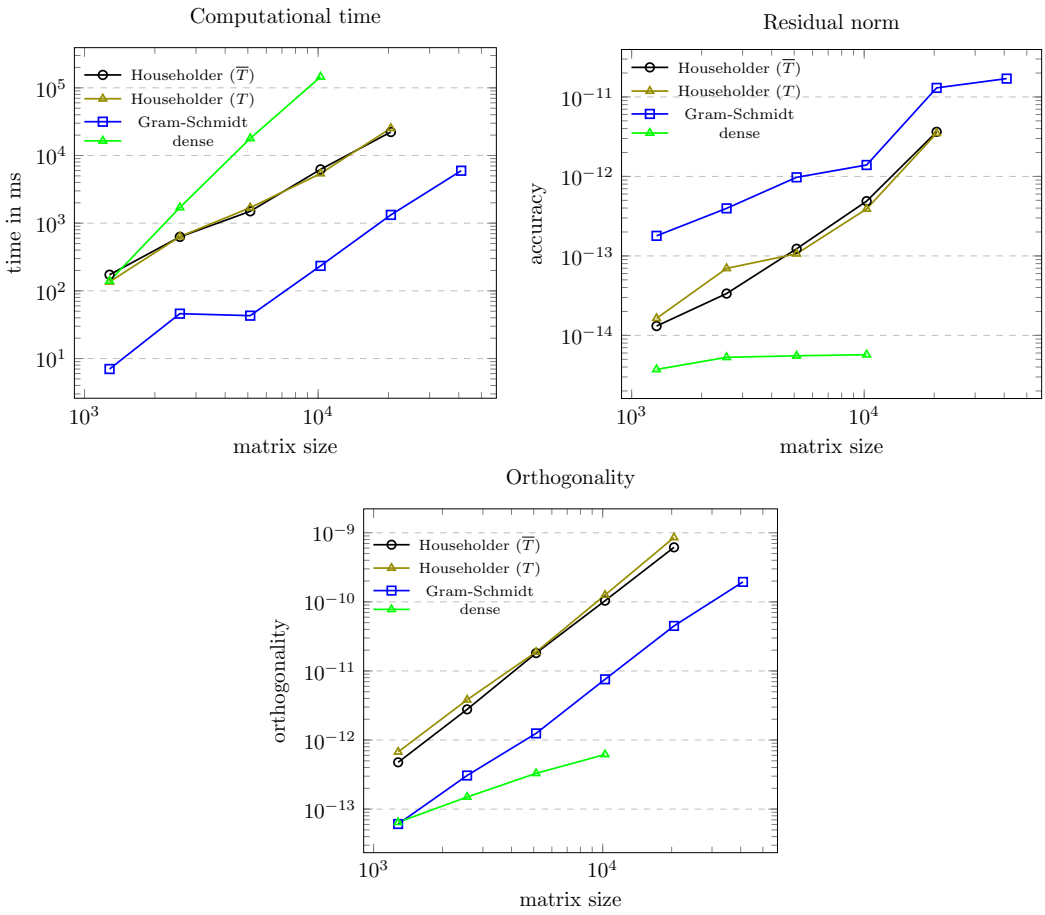


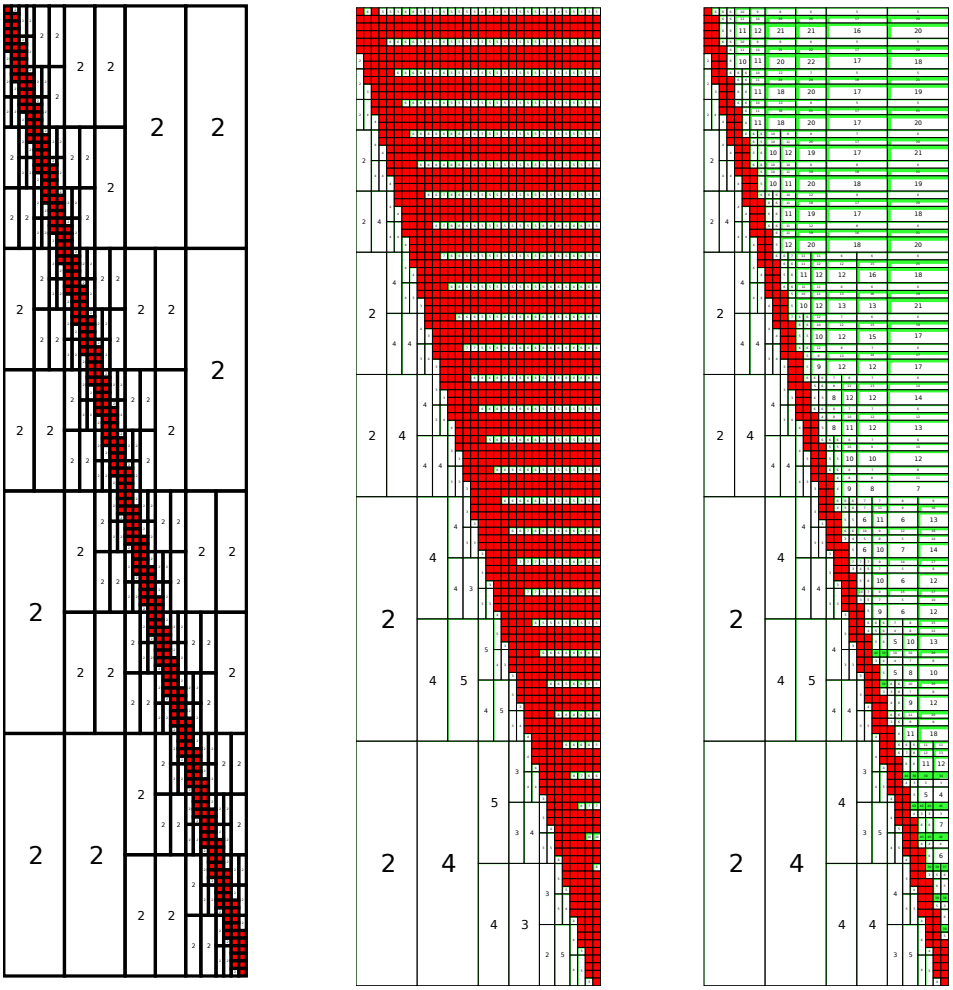
Figure 4.18. Computational time, accuracy and orthogonality for the QR factorization of randomized rectangular $\mathcal{H}_p$-matrices (with $n_{min} = 40$, truncation accuracy $\delta_{\mathcal{H}} = 10^{-10}$).

4.2.2 1D RBF Approximation

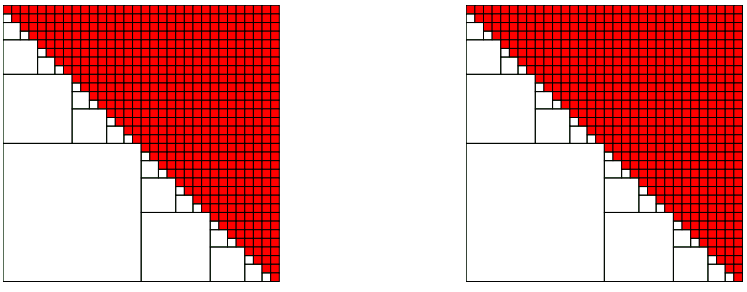
The 1D RBF approximation matrix is our first rectangular example where using the adaptive structure actually changes the $\mathcal{H}$-matrices.

The induced admissibility function predicts many inadmissible matrix blocks in the non-zero rows of Y, which reduces the overall advantage over the dense case. Simply using the original admissibility reduces the number of dense blocks that appear in the result, although we do not see any improvement in computation time, as the full matrix blocks are computed first and then truncated. But this problem could be solved with another $\mathcal{H}$-matrix multiplication technique, as can be seen in [22].

These observations are in line with our theoretical results from Subsections 3.3.2 and 3.3.7. Gram-Schmidt is faster than the block Householder approach, but this is mainly



(a) Original. *(b)* Y and R. *(c)* Y and R.



(d) $\overline{T}$. *(e)* $\overline{T}$.

Figure 4.19. QR factors of the 5120×1280 1D RBF approximation $\mathcal{H}$-matrix. (a) shows the original matrix, (b), (d) show $Y, R, \overline{T}$ using the induced admissibility, (c), (e) show $Y, R, \overline{T}$ using the original $\mathcal{H}$-block structure of A.

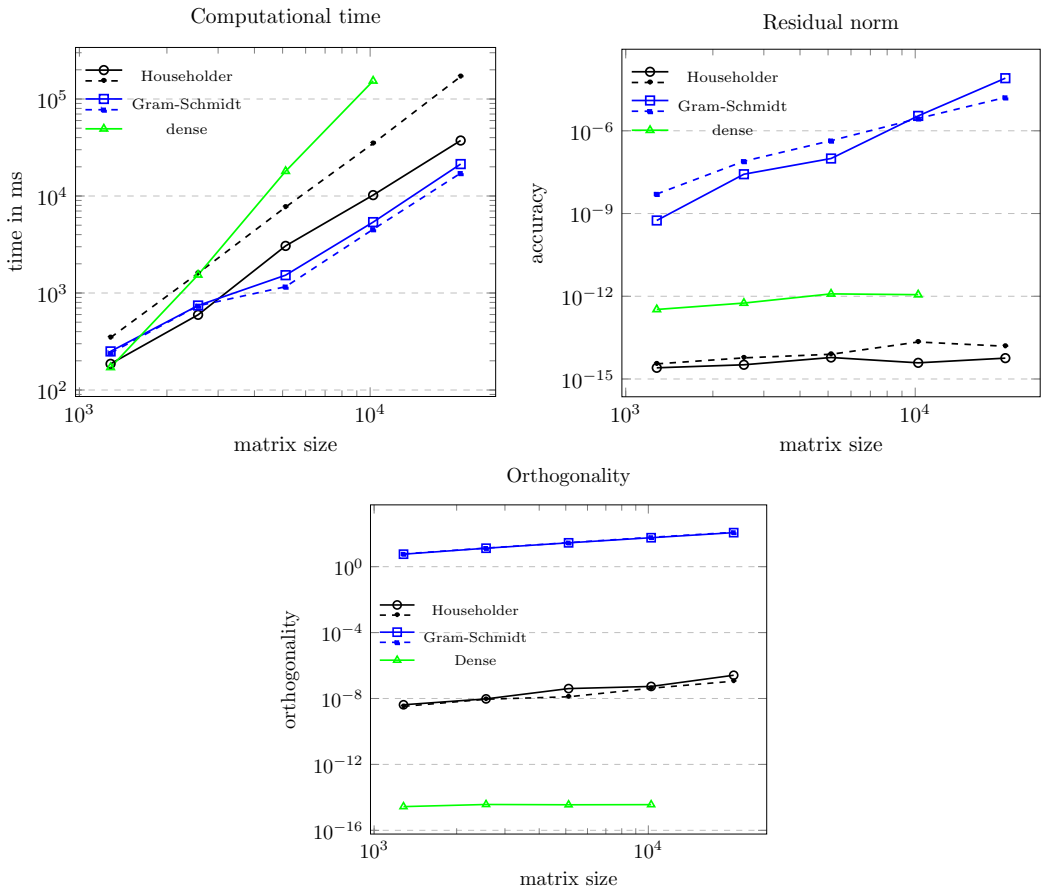
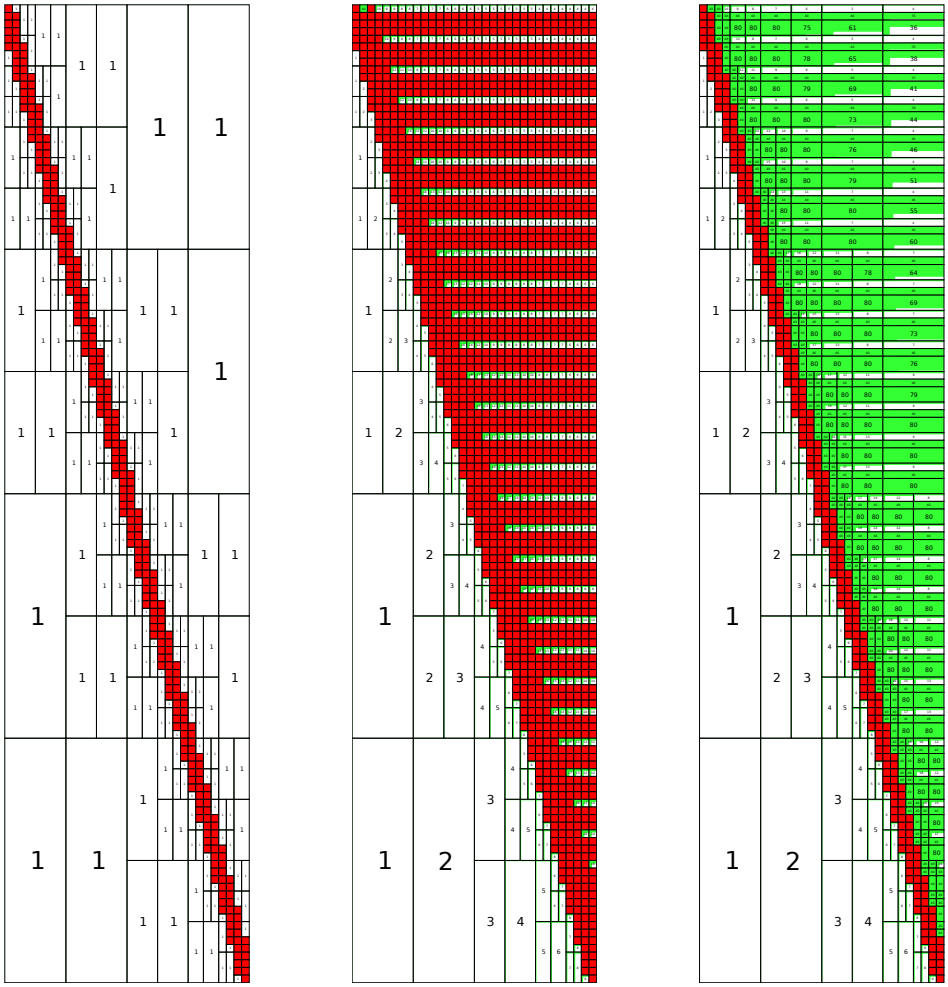


Figure 4.20. Computational time, accuracy and orthogonality for the QR factorization of 1D RBF approximation $\mathcal{H}$-matrices (with $n_{min} = 40$, truncation accuracy $\delta_{\mathcal{H}} = 10^{-10}$). Continuous lines show results for the induced admissibility function, dashed lines for reusing the original $\mathcal{H}$-structure.

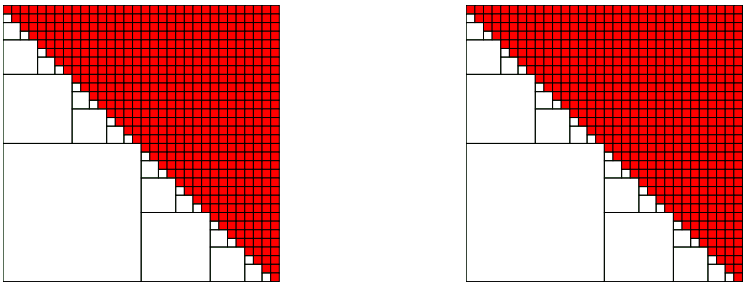
because the ranks in the resulting admissible blocks are relatively low due to the instability of the algorithm, and this goes hand in hand with a Q that is not orthogonal. Hence, the advantage in computation time comes at the expense of a poor result.

Keeping the structure and filling it with random entries, we, as Figure 4.22 shows, achieve results similar to the case of the 1D RBF interpolation. We can see that the ranks of the matrix blocks that are admissible in the fixed (original) structure, but inadmissible in the adapted structure, are much higher for the random matrix than for the RBF matrix. This is again due to inadmissible matrix blocks in the original RBF matrix that numerically do not have a full rank.

The Gram-Schmidt approach produces a much more orthogonal matrix Q than for the RBF case, which was to be expected and is consistent with the results from our RBF



(a) Original. *(b)* Y and R. *(c)* Y and R.



(d) $\overline{T}$. *(e)* $\overline{T}$.

Figure 4.21. QR factors of a randomized 5120×1280 1D RBF approximation $\mathcal{H}$-matrix. (a) shows the original matrix, (b), (d) show $Y, R, \overline{T}$ using the induced admissibility, (c), (e) show $Y, R, \overline{T}$ using the original $\mathcal{H}$-block structure of A.

interpolation examples. However, there is no longer a cost advantage over the block Householder approach, and the block Householder approach produces a QR decomposition that more accurately replicates the original matrix.

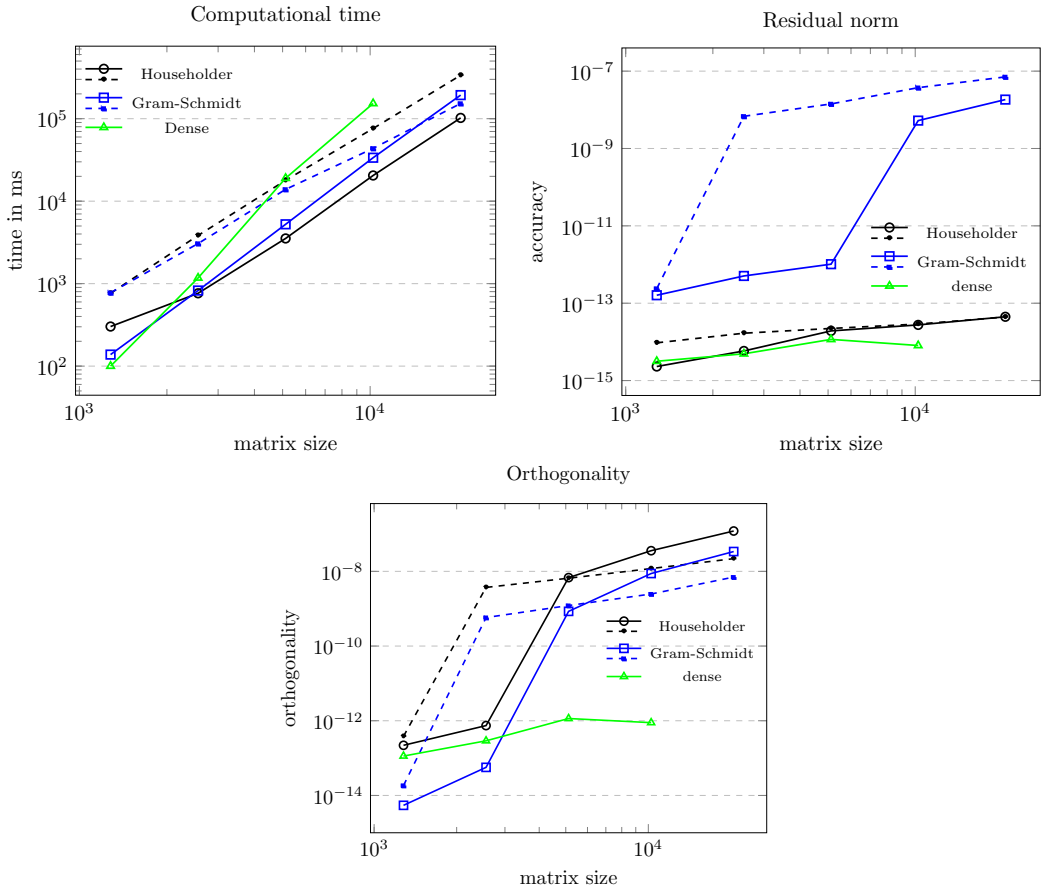


Figure 4.22. Computational time, accuracy and orthogonality for the QR factorization of randomized $2D$ RBF approximation $\mathcal{H}$-matrices (with $n_{min} = 40$, truncation accuracy $\delta_{\mathcal{H}} = 10^{-10}$). Continuous lines show results for the induced admissibility function, dashed lines for reusing the original $\mathcal{H}$-structure.

4.3 Summary

As shown by our numerical results, the newly developed method based on a Householder approach has certain advantages over the existing methods, especially in terms of orthogonality and computational times. For $\mathcal{H}$-matrices, it is generally the fastest method available and is the only $\mathcal{H}$-matrix method that provides a useful QR decomposition for the poorly conditioned RBF kernel matrices. The results thus reflect the situation of

the dense case, where a Householder approach allows for the stable computation of a QR decomposition.

In some cases, there are differences between the fixed $\mathcal{H}$-matrix structure and the adaptive $\mathcal{H}$-matrix structure versions of the algorithms, but for the block Householder algorithm, the two versions are generally comparable. For Cholesky QR and Gram-Schmidt this is different because the matrix Q has significantly more inadmissible blocks, which leads to more storage and computational cost.

Reducing the truncation accuracy reduces the computation time only moderately but drastically worsens the resulting QR decomposition.

While our proposed algorithm with Q in factored form and reduced form $\overline{T}$ is an improvement over the other algorithms from the literature that we used for comparison, it does not exhibit near-optimal complexity $\mathcal{O}(n \log^{\alpha} n)$. The reason for this is that even if the original matrix can be approximated well as a $\mathcal{H}$-matrix, the same is not necessarily true for its QR factors. This is the case for the matrix resulting from the 2D RBF interpolation problem. For the block Householder approach, there is still a slight advantage in computation time compared to the exact (dense) factorization, but this may not be enough to justify the construction of the $\mathcal{H}$-matrix in the first place. The resulting factors in the 2D RBF case either consist of many inadmissible matrix blocks when the induced admissibility function is chosen or of many admissible matrix blocks with high ranks when the original matrix structure is used.

This is even more pronounced for the 1D RBF approximation problem. For the respective matrix from 1D RBF interpolation, we do not observe the emergence of many additional inadmissible blocks. Applying the QR algorithm to the rectangular 1D approximation matrices, however, leads to a large increase of inadmissible blocks in factor Y. This corroborates our theoretical results from Subsection 3.3.7.

In general, however, the block Householder approach is an improvement over the other presented methods since the hierarchical structures in Y and $\overline{T}$ suffer from less fill-in than the explicit Q in our examples.

Chapter 5

Induced Admissibility for the $\mathcal{H}$-LU Decomposition

Our numerical results for the QR decomposition showed that using the induced admissibility function instead of simply reusing the original structure is beneficial in certain situations. This approach may also offer advantages when applied to other $\mathcal{H}$-matrix decompositions such as the LU decomposition or, more generally, to all operations on $\mathcal{H}$-matrices. We want to test this hypothesis for the LU decomposition using the algorithm first presented in [7], which is based on the following observation. Assume we have a two-by-two block representation of the (square) matrix $A \in \mathbb{R}^{n \times n}$ and its LU decomposition given by

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & \\ L_{21} & L_{22} \end{pmatrix} \cdot \begin{pmatrix} U_{11} & U_{12} \\ & U_{22} \end{pmatrix}$$

with $A_{11}, L_{11}, U_{11} \in \mathbb{R}^{n_1 \times n_1}$, $A_{12}, U_{12} \in \mathbb{R}^{n_1 \times n_2}$, $A_{21}, L_{21} \in \mathbb{R}^{n_2 \times n_1}$, $A_{22}, L_{22}, U_{22} \in \mathbb{R}^{n_2 \times n_2}$ with $n_1 + n_2 = n$. This can be rewritten as

$$\begin{aligned} A_{11} &= L_{11}U_{11}, \\ A_{12} &= L_{11}U_{12}, \\ A_{21} &= L_{21}U_{11}, \\ A_{22} - L_{21}U_{12} &= L_{22}U_{22}. \end{aligned}$$

Hence, L_{11} and U_{11} can be computed recursively by the LU decomposition of A_{11}. This result can subsequently be used to find U_{12} and L_{21} by forward and backward substitution, respectively. Finally, L_{22} and U_{22} can be (again recursively) computed by the LU decomposition of $A_{22} - L_{21}U_{12}$. This is formalized in Algorithm 5.1 for the LU decomposition itself, Algorithm 3.3 for the backward substitution and Algorithm 3.2 for the forward substitution. The latter two were already relevant for the Cholesky QR decomposition in Subsection 3.1.1 and are shown there. Apart from indirect changes

due to calls to other functions, the use of the induced admissibility function is relevant in line 8 of the Algorithm 5.1 and lines 7, 8 and 13 of the Algorithms 3.2 and 3.3. In all these cases, the arithmetic operation in the argument causes possible changes to the hierarchical structure. In Algorithm 5.1, for example, the admissibility function of the result of $A_{\sigma_2,\sigma_2} - L_{\sigma_2,\sigma_1} \cdot U_{\sigma_1,\sigma_2}$ has to be adapted before its LU decomposition is computed. The situation is similar in the respective lines of Algorithms 3.2 and 3.3.

Algorithm 5.1: LU decomposition for $\mathcal{H}$-matrices.

Data: $\mathcal{H}$-matrix $A \in \mathcal{H}(P_{I\times I}^{\mathcal{H},A}, k_A, \mathrm{adm}_A)$
Result: Lower triangular $\mathcal{H}$-matrix $L \in \mathcal{H}(P_{I\times I}^{\mathcal{H},L}, k_L, \mathrm{adm}_L)$ and upper triangular $\mathcal{H}$-matrix $U \in \mathcal{H}(P_{I\times I}^{\mathcal{H},U}, k_U, \mathrm{adm}_U)$ with $A = L \cdot U$
Function: $[L, U] = \mathcal{H}LU(A)$

1 **if** *there are no* $\sigma_1, \sigma_2 \in P_I^{hier}$ *with* $\sigma_1 \cap \sigma_2 = \emptyset$ *and* $\sigma_1 \cup \sigma_2 = \sigma$ **then**
2 | Compute the dense LU decomposition $L \cdot U = A$ with $L \in \mathbb{R}^{n,n}$, $U \in \mathbb{R}^{n,n}$;
3 **else**
4 | Find $\sigma_1, \sigma_2 \in P_I^{hier}$ with $\sigma_1 \cap \sigma_2 = \emptyset$ and $\sigma_1 \cup \sigma_2 = \sigma$;
5 | $(L_{\sigma_1,\sigma_1}, U_{\sigma_1,\sigma_1}) = \mathcal{H}LU(A_{\sigma_1,\sigma_1})$;
| // Solve $A_{\sigma_1,\sigma_2} = L_{\sigma_1,\sigma_1} \cdot U_{\sigma_1,\sigma_2}$ for U_{σ_1,σ_2}.
6 | $U_{\sigma_1,\sigma_2} = \mathcal{H}FwSub(L_{\sigma_1,\sigma_1}, A_{\sigma_1,\sigma_2})$;
| // Solve $A_{\sigma_2,\sigma_1} = L_{\sigma_2,\sigma_1} \cdot U_{\sigma_1,\sigma_1}$ for L_{σ_2,σ_1}.
7 | $L_{\sigma_2,\sigma_1} = \mathcal{H}BwSub(U_{\sigma_1,\sigma_1}, A_{\sigma_2,\sigma_1})$;
8 | $(L_{\sigma_2,\sigma_2}, U_{\sigma_2,\sigma_2}) = \mathcal{H}LU(A_{\sigma_2,\sigma_2} - L_{\sigma_2,\sigma_1} \cdot U_{\sigma_1,\sigma_2})$;
9 | Set $L = \begin{bmatrix} L_{\sigma_1,\sigma_1} & \\ L_{\sigma_2,\sigma_1} & L_{\sigma_2,\sigma_2} \end{bmatrix}$ and $U = \begin{bmatrix} U_{\sigma_1,\sigma_1} & U_{\sigma_1,\sigma_2} \\ & U_{\sigma_2,\sigma_2} \end{bmatrix}$;
10 **end**
11 **return** L, U;

In the following, we will analyze the application of the induced admissibility function to the LU decomposition in a similar way as before. First, we derive a partially induced admissibility function similar to our considerations in Subsection 3.3.1. Then we provide some simple numerical examples to show that cost and accuracy improvements can be made to the LU decomposition when the induced admissibility function is used throughout the algorithm.

5.1 Partially Induced Admissibility Function

To simplify the proofs, we only consider square and structurally symmetric $\mathcal{H}$-matrices. In the case of an $\mathcal{H}$-matrix A that is not structurally symmetric, one could use the induced structure of $A + A^T$ to get an upper bound on the induced admissibility function of A. Furthermore, we assume that every block in the hierarchical block partitioning

of A has no sons or four sons. The proofs can be transferred to matrices that do not satisfy the latter assumption, but it is less cumbersome to read it this way, and the LU decomposition is not our primary focus.

First, we need to analyze the forward and backward substitution.

Theorem 5.1 (Forward Substitution) *Let there be the following $\mathcal{H}$-matrices: $A \in \mathcal{H}(P^{\mathcal{H},A}_{I\times J}, k_A, adm_A)$, $B \in \mathcal{H}(P^{\mathcal{H},B}_{I\times J}, k_B, adm_B)$ and $L \in \mathcal{H}(P^{\mathcal{H},L}_{I\times I}, k_L, adm_L)$, where L is lower triangular. If B satisfies*

$$L \cdot B = A$$

and the admissibility function of B is induced by the $\mathcal{H}$-matrix arithmetics used in the computation of B by Algorithm 3.2, then we have for $\sigma \times \tau$

$$adm_B(\sigma \times \tau) \geq \prod_{\substack{\hat{\tau}\in P_I^{level(\tau)} \\ \hat{\tau}\leq\tau}} adm_A(\sigma \times \hat{\tau}).$$

Proof We start with the first statement. If A is an $\mathcal{H}$-matrix of depth 0, then L is inadmissible because it is a lower triangular matrix. Hence, the admissibility of B is given by the admissibility of A. Because there is only one block in every row and column, we have the statement.

Assume that the statement is true for all $\mathcal{H}$-matrices with depth k and $0 \leq k < \ell$. We show that it is also true for any $\mathcal{H}$-matrix $A \in \mathcal{H}(P^{\mathcal{H},A}_{I\times J}, k_A, \mathrm{adm}_A)$ with depth ℓ. Let σ_{left} and σ_{right} denote the non-trivial descendants of I so that $\sigma_{left} \cup \sigma_{right} = I$ and $\sigma_{left} < \sigma_{right}$ hold. Furthermore, assume that there are also non-trivial descendants of J τ_{left} and τ_{right} with $\tau_{left} \cup \tau_{right} = I$ and $\tau_{left} < \tau_{right}$, meaning we are in the first case of the inner case distinction in Algorithm 3.2. The statement is true for both blocks in the first block row and all blocks they contain because they are matrices of depth at most $\ell - 1$. We only need to check both blocks in the second block row. We begin with the first block. Set

$$S = A_{\sigma_{right},\tau_{left}} + L_{\sigma_{right},\sigma_{left}} \cdot B_{\sigma_{left},\tau_{left}}.$$

with $S \in \mathcal{H}(P^{\mathcal{H},S}_{\sigma_{right}\times\tau_{left}}, k_S, \mathrm{adm}_S)$. Let adm_S be induced by the corresponding $\mathcal{H}$-

matrix arithmetics. Then we have, by definition, for all $\sigma \times \tau \in P^{hier}_{\sigma_{right} \times \tau_{left}}$

$$\begin{aligned}
\mathrm{adm}_S(\sigma \times \tau) &= \mathrm{adm}_A(\sigma \times \tau) + \prod_{\substack{\hat{\sigma} \in P_I^{L^{max}(\sigma,\tau,I)} \\ \hat{\sigma} \subset \sigma_{left}}} (\mathrm{adm}_L(\sigma \times \hat{\sigma}) + \mathrm{adm}_B(\hat{\sigma} \times \tau)) \\
&\geq \mathrm{adm}_A(\sigma \times \tau) + \prod_{\substack{\hat{\sigma} \in P_I^{L^{max}(\sigma,\tau,I)} \\ \hat{\sigma} \subset \sigma_{left}}} (\mathrm{adm}_L(\sigma \times \hat{\sigma}) + \mathrm{adm}_A^{rw}(\hat{\sigma} \times \tau)) \\
&\geq \mathrm{adm}_A(\sigma \times \tau) + \prod_{\substack{\hat{\sigma} \in P_I^{L^{max}(\sigma,\tau,I)} \\ \hat{\sigma} \subset \sigma_{left}}} \mathrm{adm}_A^{rw(\sigma_{left},\tau_{left})}(\hat{\sigma} \times \tau) \\
&= \mathrm{adm}_A(\sigma \times \tau) + \prod_{\substack{\hat{\sigma} \in P_I^{L^{max}(\sigma,\tau,I)} \\ \hat{\sigma} \subset \sigma_{left}}} \mathrm{adm}_A^{rw}(\hat{\sigma} \times \tau).
\end{aligned}$$

S has by construction a depth less than ℓ. We replace the admissibility function adm_S by its just derived lower bound and then apply our statement, which by assumption is true for $\mathcal{H}$-matrices with depth less than ℓ. Hence, we have for all $\sigma \times \tau \in P^{hier}_{\sigma_{right} \times \tau_{left}}$

$$\begin{aligned}
\mathrm{adm}_B(\sigma \times \tau) &\geq \prod_{\substack{\widetilde{\tau} \in P_J^{L^{max}(\sigma,\tau,J)} \\ \widetilde{\tau} \subset \tau_{right}, \widetilde{\tau} \leq \tau}} \left(\mathrm{adm}_A(\sigma \times \widetilde{\tau}) + \prod_{\substack{\hat{\sigma} \in P_I^{L^{max}(\sigma,\tau,I)} \\ \hat{\sigma} \subset \sigma_{left}}} \mathrm{adm}_A^{rw}(\hat{\sigma} \times \widetilde{\tau}) \right) \\
&= \prod_{\substack{\hat{\sigma} \in P_I^{L(\sigma,\widetilde{\tau})} \\ \hat{\sigma} \leq \tau}} \mathrm{adm}_A(\sigma \times \hat{\sigma})
\end{aligned}$$

and thus our statement is true for $B_{\sigma_{right},\sigma_{left}}$. Repeating our argument for the block $B_{\sigma_{right},\sigma_{right}}$ and the case where τ does not have any descendants, finishes our proof. ∎

A similar statement for the backward substitution is also true.

Theorem 5.2 (Backward Substitution) *Let there be the following $\mathcal{H}$-matrices: $A \in \mathcal{H}(P^{\mathcal{H},A}_{J \times I}, k_A, adm_A)$, $B \in \mathcal{H}(P^{\mathcal{H},B}_{J \times I}, k_B, adm_B)$ and $U \in \mathcal{H}(P^{\mathcal{H},U}_{I \times I}, k_U, adm_U)$, where U is upper triangular. If B satisfies*

$$B \cdot U = A$$

and the admissibility function of B is induced by the $\mathcal{H}$-matrix arithmetics used in the

computation of B by Algorithm 3.3, then we have for $\tau \times \sigma' \in P_{J\times I}^{hier}$

$$adm_B(\sigma \times \sigma') \geq \prod_{\substack{\hat{\sigma}\in P_I^{level(\sigma')} \\ \hat{\sigma}\leq\sigma'}} adm_A(\hat{\sigma}\cdot\sigma').$$

Proof We apply Theorem 5.1 to

$$(B\cdot U)^T = A^T \;\Leftrightarrow\; U^T\cdot B^T = A^T.$$ ∎

Now we can make a statement about the LU decomposition. The proof itself is similar to the one before. The main difference is that not only the forward substitution is involved, and we restrict ourselves to square matrices at the beginning.

Theorem 5.3 *Let $A \in \mathcal{H}(P_{I\times I}^{\mathcal{H},A}, k_A, adm_A)$ be an $\mathcal{H}$-matrix and its structure is symmetric, meaning*

$$adm_A(\sigma\times\sigma') = adm_A(\sigma'\times\sigma)$$

for all $\sigma,\sigma' \in P_I^{hier}$ with $\sigma\times\sigma' \in P_{I\times I}^{hier}$. Let A have an LU decomposition given by lower triangular $L \in \mathcal{H}(P_{I\times I}^{\mathcal{H},L}, k_L, adm_L)$ and upper triangular $U \in \mathcal{H}(P_{I\times I}^{\mathcal{H},U}, k_U, adm_U)$. Let the admissibility functions of L and U be induced by the $\mathcal{H}$-matrix arithmetic used in their computation in Algorithm 5.1. Let $\sigma,\sigma' \in P_I^{hier}$ with $\sigma\times\sigma' \in P_{I\times I}^{hier}$. If $\sigma \leq \sigma'$, then it holds that

$$\begin{aligned} adm_L(\sigma\times\sigma') &\geq \prod_{\substack{\hat{\sigma}\in P_I^{level(\sigma')} \\ \hat{\sigma}\leq\sigma'}} adm_A(\sigma\times\hat{\sigma}) \\ &= adm_A^{cw}(\sigma\times\sigma') \end{aligned}$$

and if $\sigma \geq \sigma'$, then it holds that

$$\begin{aligned} adm_U(\sigma\times\sigma') &\geq \prod_{\substack{\hat{\sigma}\in P_I^{level(\sigma)} \\ \hat{\sigma}\leq\sigma}} adm_A(\hat{\sigma}\times\sigma) \\ &= adm_A^{rw}(\sigma\times\sigma'). \end{aligned}$$

Proof The diagonal elements of L and U are inadmissible by construction, and the

same is true for A itself by assumption. Thus, we have for L

$$\begin{aligned}
\mathrm{adm}_L(\sigma \times \sigma) &= \text{False} \\
&= \text{False} \cdot \prod_{\substack{\hat{\sigma} \in P_I^{\mathrm{level}(\sigma)} \\ \hat{\sigma} \leq \sigma}} \mathrm{adm}_A(\sigma \times \hat{\sigma}) \\
&= \mathrm{adm}_A(\sigma \times \sigma) \cdot \prod_{\substack{\hat{\sigma} \in P_I^{\mathrm{level}(\sigma)} \\ \hat{\sigma} \leq \sigma}} \mathrm{adm}_A(\sigma \times \hat{\sigma}) \\
&= \mathrm{adm}_A^{cw}(\sigma \times \sigma)
\end{aligned}$$

and for U

$$\begin{aligned}
\mathrm{adm}_U(\sigma \times \sigma) &= \text{False} \\
&= \text{False} \cdot \prod_{\substack{\hat{\sigma} \in P_I^{\mathrm{level}(\sigma)} \\ \hat{\sigma} < \sigma}} \mathrm{adm}_A(\hat{\sigma} \times \sigma) \\
&= \mathrm{adm}_A(\sigma \times \sigma) \cdot \prod_{\substack{\hat{\sigma} \in P_I^{\mathrm{level}(\sigma)} \\ \hat{\sigma} < \sigma}} \mathrm{adm}_A(\hat{\sigma} \times \sigma) \\
&= \mathrm{adm}_A^{rw}(\sigma \times \sigma).
\end{aligned}$$

The statement is thus true for all diagonal blocks that are not changed before computing their LU decomposition during the algorithm. Furthermore, the statement is true for matrices with depth 0 because it is only one diagonal block. Assume that the statement is true for all $\mathcal{H}$-matrices with depth k and $0 \leq k < \ell$. We show that it is also true for any $\mathcal{H}$-matrix with depth ℓ. Let σ_{left} and σ_{right} denote the non-trivial descendants of I so that $\sigma_{left} \cup \sigma_{right} = I$ and $\sigma_{left} < \sigma_{right}$ hold.

For the upper left diagonal element $\sigma_{left} \times \sigma_{left}$, the statement is true because it is a matrix of depth $\ell - 1$ and does not get changed before its LU decomposition is recursively computed. With Theorem 5.1, the statement is also true for the off-diagonal blocks $\sigma_{left} \times \sigma_{right}$ and $\sigma_{right} \times \sigma_{left}$, which are also at most of depth $\ell - 1$. Now let us check the bottom diagonal block $\sigma_{right} \times \sigma_{right}$. We need to show the statement for all $\sigma \times \sigma' \in P_{I \times I}^{hier}$ with $\subseteq \sigma_{right} \times \sigma_{right}$. Set

$$S = A_{\sigma_{right}, \sigma_{right}} + L_{\sigma_{right}, \sigma_{left}} \cdot U_{\sigma_{left}, \sigma_{right}} xc$$

with $S \in \mathcal{H}(P^{\mathcal{H},S}_{\sigma_{right} \times \sigma_{right}}, k_S, \mathrm{adm}_S)$. Let adm_S be induced by the corresponding $\mathcal{H}$-

matrix arithmetics. Then we have by definition

$$\mathrm{adm}_S(\sigma \times \sigma') = \mathrm{adm}_A(\sigma \times \sigma') + \prod_{\substack{\hat{\sigma} \in P_I^{L^{max}(\sigma,\sigma',I)} \\ \hat{\sigma} \subset \sigma_{left}}} \left(\mathrm{adm}_A(\sigma \times \hat{\sigma}) + \mathrm{adm}_A(\sigma' \times \hat{\sigma})\right).$$

S is of depth smaller than ℓ, so our statement is true and we have for L with $\sigma > \sigma'$

$$\begin{aligned}
\mathrm{adm}_L(\sigma \times \sigma') &\geq \prod_{\substack{\widetilde{\sigma} \in P_I^{L^{max}(\sigma,\sigma',I)} \\ \widetilde{\sigma} \subset \sigma_{right}, \widetilde{\sigma} \leq \sigma'}} \left(\mathrm{adm}_A(\sigma \times \widetilde{\sigma}) + \prod_{\substack{\hat{\sigma} \in P_I^{L(\sigma,\widetilde{\sigma})} \\ \hat{\sigma} \subset \sigma_{left}}} \left(\mathrm{adm}_A(\sigma \times \hat{\sigma}) + \mathrm{adm}_A(\widetilde{\sigma} \times \hat{\sigma})\right) \right) \\
&\geq \prod_{\substack{\widetilde{\sigma} \in P_I^{L^{max}(\sigma,\sigma',I)} \\ \widetilde{\sigma} \subset \sigma_{right}, \widetilde{\sigma} \leq \sigma'}} \left(\mathrm{adm}_A(\sigma \times \widetilde{\sigma}) + \prod_{\substack{\hat{\sigma} \in P_I^{L(\sigma,\widetilde{\sigma})} \\ \hat{\sigma} \subset \sigma_{left}}} \mathrm{adm}_A(\sigma \times \hat{\sigma}) \right) \\
&= \prod_{\substack{\hat{\sigma} \in P_I^{L(\sigma,\widetilde{\sigma})} \\ \hat{\sigma} \leq \sigma'}} \mathrm{adm}_A(\sigma \times \hat{\sigma})
\end{aligned}$$

and for U with $\sigma < \sigma'$

$$\begin{aligned}
\mathrm{adm}_U(\sigma \times \sigma') &\geq \prod_{\substack{\widetilde{\sigma} \in P_I^{L^{max}(\sigma,\sigma',I)} \\ \widetilde{\sigma} \subset \sigma_{right}, \widetilde{\sigma} \leq \sigma}} \left(\mathrm{adm}_A(\widetilde{\sigma} \times \sigma') + \prod_{\substack{\hat{\sigma} \in P_I^{L(\sigma,\widetilde{\sigma})} \\ \hat{\sigma} \subset \sigma_{left}}} \left(\mathrm{adm}_A(\widetilde{\sigma} \times \hat{\sigma}) + \mathrm{adm}_A(\sigma' \times \hat{\sigma})\right) \right) \\
&\geq \prod_{\substack{\widetilde{\sigma} \in P_I^{L^{max}(\sigma,\sigma',I)} \\ \widetilde{\sigma} \subset \sigma_{right}, \widetilde{\sigma} \leq \sigma'}} \left(\mathrm{adm}_A(\widetilde{\sigma} \times \sigma') + \prod_{\substack{\hat{\sigma} \in P_I^{L(\sigma',\widetilde{\sigma})} \\ \hat{\sigma} \subset \sigma_{left}}} \mathrm{adm}_A(\sigma' \times \hat{\sigma}) \right) \\
&= \prod_{\substack{\hat{\sigma} \in P_I^{L(\sigma,\widetilde{\sigma})} \\ \hat{\sigma} \leq \sigma'}} \mathrm{adm}_A(\hat{\sigma} \times \sigma')
\end{aligned}$$

■

As in the case of the QR decomposition, one could ask under what circumstances these admissibility functions are fully induced and not only partially induced. We will not formalize this here, but we will see in the next section that the partially induced admissibility function developed here is fully induced in our numerical example.

5.2 Numerical Results

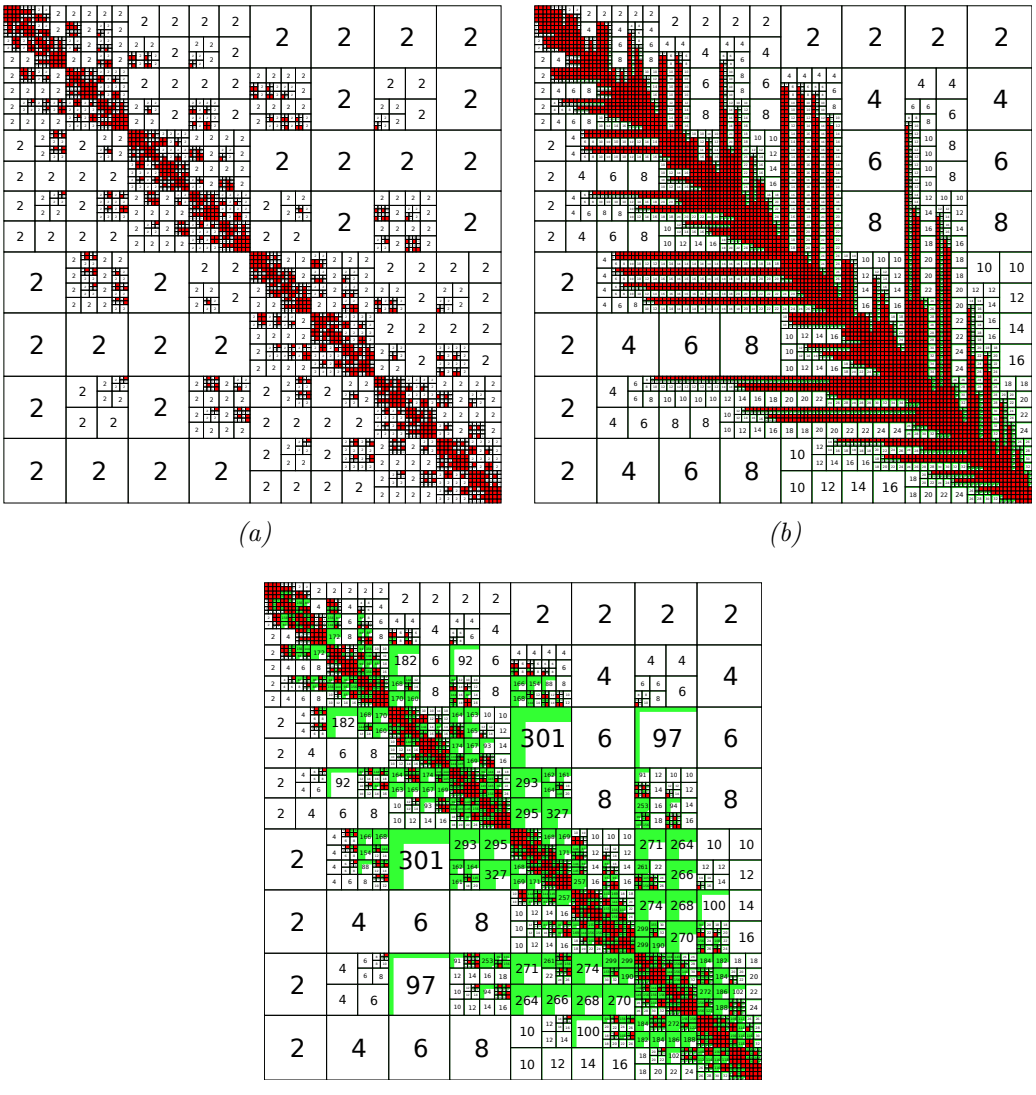


Figure 5.1. LU decomposition of a randomized 10000×10000 2D RBF interpolation $\mathcal{H}$-matrix. (a) shows the original matrix, (b) shows the LU factors with the adaptive $\mathcal{H}$-matrix structure, (c) shows the factors with the fixed (original) $\mathcal{H}$-matrix structure.

We want to compare our modified approach with the usual LU decomposition for $\mathcal{H}$-matrices. To do that, we work analogous to our numerical review of the QR decomposition in Chapter 4 but choose only one example matrix, the randomized 2D RBF interpolation matrix. To create this matrix, we use the same parameters as before,

meaning the Gaussian function

$$f(x, y) = e^{-\varepsilon \|x-y\|_2^2}$$

with parameter $\varepsilon = c_d \cdot n^{1/d}$, where n is the number of data points, $d = 2$ and equidistant points in $[0, 1]^2$ with $c_2 = 0.2$. However, we only test the randomized variant because the RBF interpolation matrix with its regular entries is highly ill-conditioned.

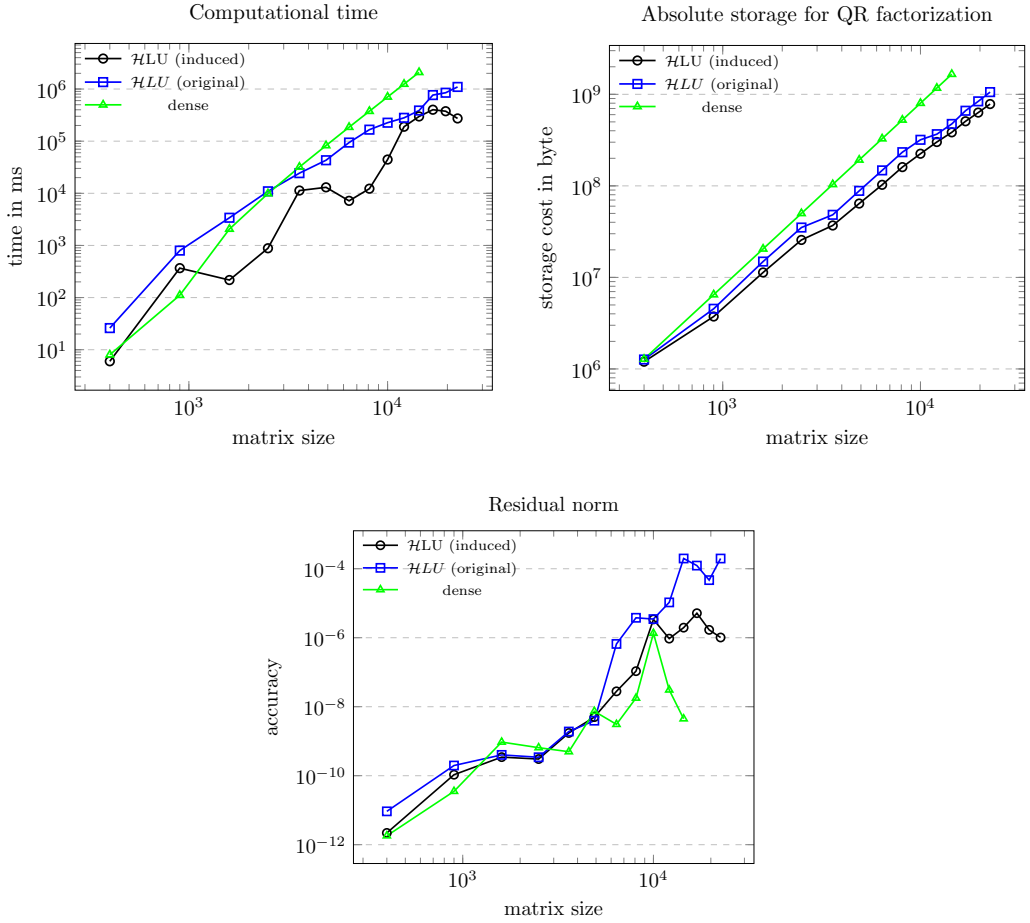


Figure 5.2. Computational time, accuracy and orthogonality for the LU factorization of randomized 2D RBF interpolation $\mathcal{H}$-matrices (with $n_{\min} = 80$, truncation accuracy $\delta_{\mathcal{H}} = 10^{-10}$).

As already explained in Chapter 4, we use adaptive cross approximation [9] to convert the (typically) dense matrices in the above test case to approximating $\mathcal{H}$-matrices, use local adaptive ranks for a desired relative accuracy and calculate

$$e_{approx} = \|A - LU\|_2$$

by the square root of the eigenvalue approximation obtained by 20 steps of the power iteration applied to the matrix $(A - LU)^*(A - LU)$ to measure the quality of the resulting LU decomposition. For comparison, we also compute the LU decomposition in standard (dense) arithmetic applied to the dense version of the original matrix. The implementation is again an extension to the H2Lib library [41].

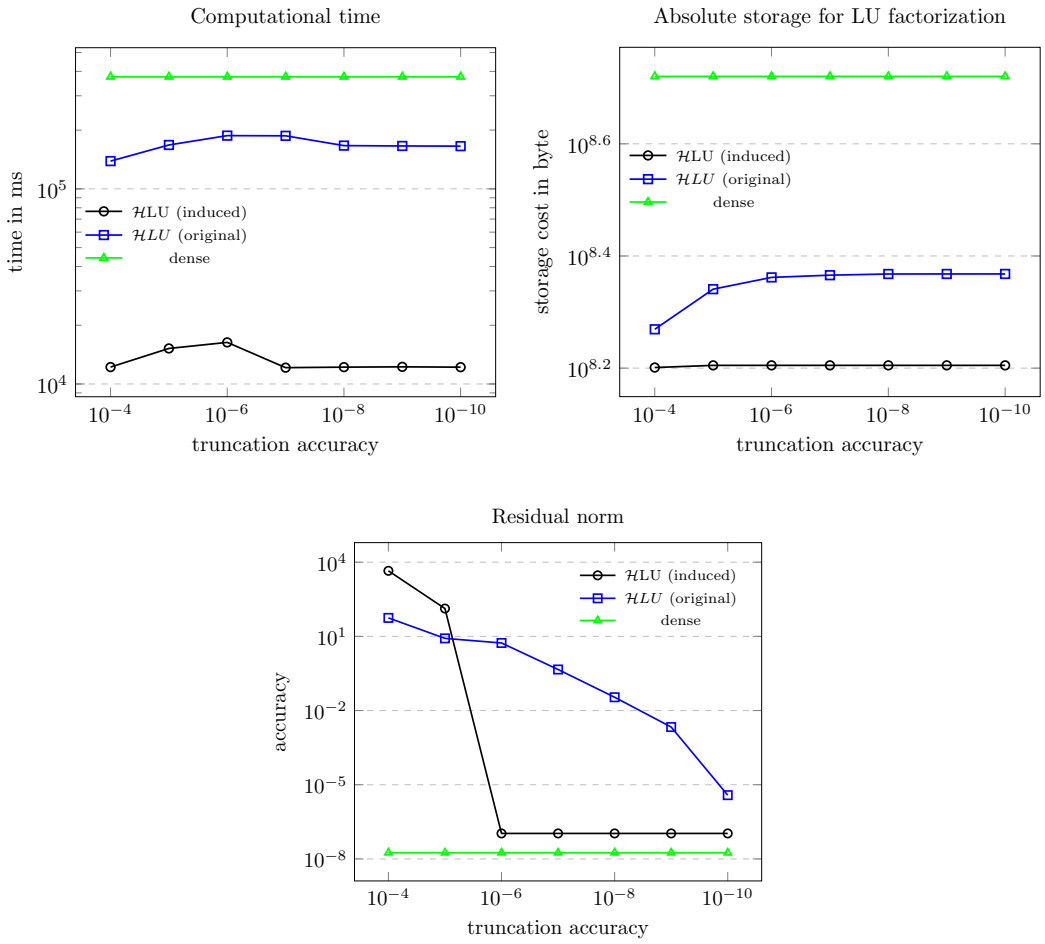


Figure 5.3. Effect of truncation accuracy $\delta_{\mathcal{H}}$ in $\mathcal{H}$-arithmetic on computational time, storage and accuracy, illustrated for a 81000×81000 randomized 2D RBF interpolation $\mathcal{H}$-matrix with leaf size 80.

Our results are generally promising. We have advantages in computational time, storage cost, and accuracy for the variant using the induced admissibility function. However, there are some major fluctuations in computational time for the $\mathcal{H}$-LU decomposition that uses the induced admissibility function. Sometimes a larger matrix is calculated faster (in absolute terms) than a smaller matrix. That is counterintuitive. The version that uses the fixed (original) structure does not show this behavior. It may have to do

with differences in the $\mathcal{H}$-matrix structure between different matrix sizes, but further investigations are warranted.

We also analyzed the effect of the adaptive $\mathcal{H}$-accuracy $\delta_{\mathcal{H}}$ on the computational time, storage as well as the accuracy of the LU factors for a matrix of fixed size. The increase in computational time as $\delta_{\mathcal{H}}$ decreases from 10^{-4} down to 10^{-10} appears to be moderate to non-existing compared to the benefits with respect to accuracy and orthogonality.

Interestingly, the advantage of the LU decomposition that uses the induced admissibility function extends even to low truncation accuracies, where one might expect a better performance or a smaller storage requirement for the variant using the original structure.

Chapter 6

Conclusion and Outlook

We have developed a new method for the QR decomposition of $\mathcal{H}$-matrices based on a block Householder approach and discussed its usefulness in comparison with existing methods for $\mathcal{H}$-matrices. Although QR decompositions cannot be computed efficiently for all types of $\mathcal{H}$-matrices, the new method extends this field and is superior (for most of our model problems) to other competing methods from the literature in terms of computational time and orthogonality of the resulting Q-factor.

One important contribution was relying on the reduced representation $\overline{T}$ of the factor T in the representation $Q = I - YTY^T$. Both our theoretical and numerical results support the idea that switching to $\overline{T}$ instead of T is an improvement. In many cases while using induced admissibility functions, $\overline{T}$ has fewer inadmissible blocks than T and cannot have more. Furthermore, the calculation of $\overline{T}$ requires fewer $\mathcal{H}$-matrix multiplications compared to the calculation of T while the cost of multiplication with T is not influenced by the change from T to $\overline{T}$. Interestingly, this is still true for dense matrices.

However, computing the QR decomposition for rectangular, non-square $\mathcal{H}$-matrices proved to be challenging. For rectangular $\mathcal{H}_p$-matrices, both the Gram-Schmidt and the block Householder approach efficiently produced a solution. Unfortunately, even a minor complication such as using 1D RBF approximation matrices, which have two additional diagonals of inadmissible blocks compared to the $\mathcal{H}_p$-matrices, led to QR factors that no longer allowed for an efficient $\mathcal{H}$-matrix approximation.

As indicated by our theoretical findings, this problem extends beyond the specific examples we have analyzed. Under certain rather weak assumptions, the QR decomposition of a rectangular, but not square, $\mathcal{H}$-matrix cannot be efficiently approximated as an $\mathcal{H}$-matrix if there is no further data sparsity not yet taken into account.

These results are valid for square $\mathcal{H}$-matrices as well but do not have such a strong impact on which $\mathcal{H}$-matrices allow for a good QR decomposition because the matrix Y

is lower triangular. Nevertheless, with Theorem 3.29 we were able to show that if an efficient $\mathcal{H}$-matrix approximation exists, whether it is square or not, we can compute it efficiently using the block Householder approach.

Using the induced admissibility function instead of reusing the original one improved the results only in some cases. It remains to be seen whether applying the improvements for the $\mathcal{H}$-matrix product suggested in [22] can further accelerate the computations.

In general, the induced admissibility function was helpful in situations where it identified inadmissible leaves (which were admissible in the original $\mathcal{H}$-matrix structure). However, if its application only resulted in smaller admissible matrix blocks, as was the case for the 2D Laplacian matrix, it could increase the computational cost. It may be desirable to develop a combination of both ideas, keeping larger admissible matrix blocks whenever possible and introducing inadmissible leaves only when necessary. Another improvement may result from also considering coarsening procedures in which either several admissible (inadmissible) blocks are agglomerated into a single admissible (inadmissible) block or inadmissible blocks are converted into admissible blocks [35].

For sparse matrices, much research has been devoted to finding suitable sparsity patterns through reordering (e.g., bandwidth reduction or nested dissection). Such considerations, especially domain-decomposition based clustering strategies [38, 37], may also be useful for the QR factorization of $\mathcal{H}$-matrices.

However, the results of this work open up several other avenues that may be worth exploring. Any sparse matrix can be seen as a $\mathcal{H}$-matrix, where the leaf size is 1 and all admissible blocks have rank 0. Therefore, all our theoretical results can be applied there as well. In particular, we have shown that any band matrix A allows a QR decomposition, where Y, $\overline{T}$ and R are band matrices again. More remarkably, even the complexity estimates can be transferred. In this case, all truncation costs can be omitted since rank 0 matrices do not generate higher ranks. Thus, a QR decomposition of a banded matrix can be computed with a complexity that depends mainly on its bandwidth. It may be worthwhile to formalize these results.

We also reworked the LU decomposition of $\mathcal{H}$-matrices to respect the induced admissibility functions and obtained some interesting results. At least for our only numerical example, the LU decomposition using the induced admissibility function had a better performance than the one using the fixed (original) structure. The $\mathcal{H}$-LU decomposition found several different applications in the literature [7, 37, 4, 61] and especially a combination with the results of [22] could lead to some improvements there.

Bibliography

[1] Sivaram Ambikasaran, Daniel Foreman-Mackey, Leslie Greengard, David W. Hogg, and Michael O'Neil. "Fast Direct Methods for Gaussian Processes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.2 (2016), pp. 252–265. DOI: 10.1109/TPAMI.2015.2448083.

[2] Jonas Ballani and Daniel Kressner. "Matrices with Hierarchical Low-Rank Structures". In: *Exploiting Hidden Structure in Matrix Computations: Algorithms and Applications : Cetraro, Italy 2015*. Basel: Springer International Publishing, 2016, pp. 161–209. DOI: 10.1007/978-3-319-49887-4_3.

[3] Prasanta Kumar Banerjee, Prasanta Kumar Banerjee, and Roy Butterfield. *Boundary element methods in engineering science*. London: McGraw-Hill (UK), 1981.

[4] Lehel Banjai and Wolfgang Hackbusch. "Hierarchical matrix techniques for low- and high-frequency Helmholtz problems". In: *IMA journal of numerical analysis* 28.1 (2008), pp. 46–79. DOI: 10.1093/imanum/drm001.

[5] Rick Beatson and Leslie Greengard. "A short course on fast multipole methods". In: *Wavelets, multilevel methods and elliptic PDEs*. Oxford: Clarendon Press, 1997, pp. 1–37.

[6] Mario Bebendorf. "Approximation of boundary element matrices". In: *Numerische Mathematik* 86.4 (2000), pp. 565–589. DOI: 10.1007/PL00005410.

[7] Mario Bebendorf. "Hierarchical LU decomposition-based preconditioners for BEM". In: *Computing* 74.3 (2005), pp. 225–247. DOI: 10.1007/s00607-004-0099-6.

[8] Mario Bebendorf. *Hierarchical matrices*. Vol. 63. Lecture Notes in Computational Science and Engineering. Springer, 2008. DOI: 10.1007/978-3-540-77147-0.

[9] Mario Bebendorf and Sergej Rjasanow. "Adaptive low-rank approximation of collocation matrices". In: *Computing* 70.1 (2003), pp. 1–24. DOI: 10.1007/s00607-002-1469-6.

[10] Peter Benner and Thomas Mach. "On the QR decomposition of $\mathcal{H}$-matrices". In: *Computing* 88.3-4 (2010), pp. 111–129. DOI: 10.1007/s00607-010-0087-y.

[11] Dario A. Bini, Stefano Massei, and Leonardo Robol. "Efficient cyclic reduction for Quasi-Birth–Death problems with rank structured blocks". In: *Applied Numerical Mathematics* 116 (2017). New Trends in Numerical Analysis: Theory, Methods, Algorithms and Applications (NETNA 2015), pp. 37–46. DOI: 10.1016/j.apnum.2016.06.014.

[12] Christian Bischof and Charles Van Loan. "The WY representation for products of Householder matrices". In: *SIAM Journal on Scientific and Statistical Computing* 8.1 (1987), pp. 2–13.

[13] Steffen Börm. "$\mathcal{H}2$-matrices–Multilevel methods for the approximation of integral operators". In: *Computing and Visualization in Science* 7.3-4 (2004), pp. 173–181.

[14] Steffen Börm. "H2-Matrix Arithmetics in Linear Complexity". In: *Computing. Archives for Informatics and Numerical Computation* 77.1 (2006), pp. 1–28. DOI: 10.1007/s00607-005-0146-y.

[15] Steffen Börm, Lars Grasedyck, and Wolfgang Hackbusch. *Hierarchical matrices - Lecture note 21/2003*. Preprint. 2003. URL: https://www.mis.mpg.de/de/publications/andere-reihen/ln/lecturenote-2103.html.

[16] Carlos A. Brebbia. *The boundary element method for engineers*. Plymouth: Pentech Press Ltd., 1978.

[17] David S. Broomhead and David Lowe. "Multivariable Functional Interpolation and Adaptive Networks". In: *Complex Systems* 2.3 (1988), pp. 321–355.

[18] Rocio Carratala-Saez, Sven Christophersen, Jose I Aliaga, Vicenc Beltran, Steffen Börm, and Enrique S Quintana-Orti. "Exploiting nested task-parallelism in the H-LU factorization". In: *Journal of Computational Science* 33 (2019), pp. 20–33. DOI: 10.1016/j.jocs.2019.02.004.

[19] Alexander H.-D. Cheng and Daisy T. Cheng. "Heritage and early history of the boundary element method". In: *Engineering analysis with boundary elements* 29.3 (2005), pp. 268–302. DOI: 10.1016/j.enganabound.2004.12.001.

[20] Martin Costabel. "Principles of boundary element methods". In: *Computer Physics Reports* 6.1 (1987), pp. 243–274. DOI: 10.1016/0167-7977(87)90014-1.

[21] Timothy A Davis and Yifan Hu. "The University of Florida sparse matrix collection". In: *ACM Transactions on Mathematical Software (TOMS)* 38.1 (2011), pp. 1–25.

[22] Jürgen Dölz, Helmut Harbrecht, and Michael D. Multerer. "On the best approximation of the hierarchical matrix product". In: *SIAM Journal on Matrix Analysis and Applications* 40.1 (2019), pp. 147–174. DOI: 10.1137/18M1189373.

[23] Carl Eckart and Gale Young. "The approximation of one matrix by another of lower rank". In: *Psychometrika* 1.3 (1936), pp. 211–218. DOI: 10.1007/BF02288367.

[24] Erik Elmroth and Fred G. Gustavson. "Applying recursion to serial and parallel QR factorization leads to better performance". In: *IBM Journal of Research and Development* 44.4 (2000), pp. 605–624.

[25] Gregory E. Fasshauer. "Positive definite kernels: past, present and future". In: *Dolomite Research Notes on Approximation* 4 (2011), pp. 21–63.

[26] Gregory E. Fasshauer and Michael McCourt. *Kernel-based Approximation Methods using MATLAB*. Vol. 19. Interdisciplinary Mathematical Sciences. Singapur: World Scientific, 2015.

[27] Gregory E. Fasshauer and Jack G. Zhang. "On choosing "optimal" shape parameters for RBF approximation". In: *Numerical Algorithms* 45.1 (2007), pp. 345–368. DOI: 10.1007/s11075-007-9072-8.

[28] Bengt Fornberg, Elisabeth Larsson, and Natasha Flyer. "Stable computations with Gaussian radial basis functions". In: *SIAM Journal on Scientific Computing* 33.2 (2011), pp. 869–892. DOI: 10.1137/09076756X.

[29] Katrijn Frederix and Marc Van Barel. "Solving a large dense linear system by adaptive cross approximation". In: *Journal of computational and applied mathematics* 234.11 (2010), pp. 3181–3195. DOI: 10.1016/j.cam.2010.02.008.

[30] Walter Gander. "Algorithms for the QR decomposition". In: *ETH Zürich Research Report* 80.02 (1980), pp. 1251–1268.

[31] Wallace Givens. "Computation of plain unitary rotations transforming a general matrix to triangular form". In: *Journal of the Society for Industrial and Applied Mathematics* 6.1 (1958), pp. 26–50. DOI: 10.1137/0106004. URL: https://doi.org/10.1137/0106004.

[32] Gene H. Golub and William Kahan. "Calculating the singular values and pseudo-inverse of a matrix". In: *Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis* 2.2 (1965), pp. 205–224. DOI: 10.1137/0702016.

[33] Gene H. Golub and Charles F. Van Loan. *Matrix computations. Johns Hopkins studies in the mathematical sciences*. Baltimore, MD: Johns Hopkins University Press, 1996.

[34] Sergei A. Goreinov, Eugene E. Tyrtyshnikov, and Nickolai L. Zamarashkin. "A theory of pseudoskeleton approximations". In: *Linear algebra and its applications* 261.1 (1997), pp. 1–21. DOI: 10.1016/S0024-3795(96)00301-1.

[35] Lars Grasedyck. "Adaptive Recompression of $\mathcal{H}$-Matrices for BEM". In: *Computing* 74.3 (2005), pp. 205–223. DOI: 10.1007/s00607-004-0103-1.

[36] Lars Grasedyck and Wolfgang Hackbusch. "Construction and arithmetics of $\mathcal{H}$-matrices". In: *Computing* 70.4 (2003), pp. 295–334. DOI: 10.1007/s00607-003-0019-1.

[37] Lars Grasedyck, Ronald Kriemann, and Sabine Le Borne. "Domain Decomposition Based $\mathcal{H}$-LU Preconditioning". In: *Numerische Mathemathik* 112 (2009), pp. 565–600. DOI: 10.1007/s00211-009-0218-6.

[38] Lars Grasedyck, Ronald Kriemann, and Sabine Le Borne. "Parallel black box $\mathcal{H}$-LU preconditioning for elliptic boundary value problems". In: *Computing and Visualization in Science* 4-6 (2008), pp. 273–291. DOI: 10.1007/s00791-008-0098-9.

[39] Leslie Greengard and Vladimir Rokhlin. "A fast algorithm for particle simulations". In: *Journal of Computational Physics* 73.2 (1987), pp. 325–348. DOI: 10.1016/0021-9991(87)90140-9.

[40] Vincent Griem and Sabine Le Borne. "A Block Householder–Based Algorithm for the QR Decomposition of Hierarchical Matrices". In: *SIAM Journal on Matrix Analysis and Applications* 45.2 (2024), pp. 847–874. DOI: 10.1137/22M1544555.

[41] *H2Lib 3.0*. www.h2lib.org. Steffen Börm et al. 2016.

[42] Wolfgang Hackbusch. "A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: Introduction to $\mathcal{H}$-matrices". In: *Computing* 62.2 (1999), pp. 89–108. DOI: 10.1007/s006070050015.

[43] Wolfgang Hackbusch. *Hierarchical Matrices: Algorithms and Analysis*. Vol. 49. Springer, 2015.

[44] Wolfgang Hackbusch and Steffen Börm. "Data-sparse approximation by adaptive $\mathcal{H}^2$-matrices". In: *Computing* 69.1 (2002), pp. 1–35. DOI: 10.1007/s00607-002-1450-4.

[45] Wolfgang Hackbusch, Boris Khoromskij, and Stefan A. Sauter. "On $\mathcal{H}^2$-Matrices". In: *Lectures on Applied Mathematics*. Berlin, Heidelberg: Springer, 2000, pp. 9–29.

[46] Wolfgang Hackbusch and Boris N Khoromskij. "A sparse $\mathcal{H}$-matrix arithmetic. Part II: Application to multi-dimensional problems". In: *Computing* 64 (2000), pp. 21–47. ISSN: 1436-5057. DOI: 10.1007/PL00021408.

[47] Wolfgang Hackbusch, Boris N. Khoromskij, and Ronald Kriemann. "Hierarchical matrices based on a weak admissibility criterion". In: *Computing* 73.3 (2004), pp. 207–243. DOI: 10.1007/s00607-004-0080-4.

[48] Nicholas J. Higham. *Accuracy and stability of numerical algorithms*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2002.

[49] Nicholas J. Higham. "Cholesky factorization". In: *Wiley Interdisciplinary Reviews: Computational Statistics* 1.2 (2009), pp. 251–254. DOI: 10.1002/wics.18.

[50] Nicholas J. Higham. *Functions of matrices: theory and computation*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2008.

[51] Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. "Kernel methods in machine learning". In: *The Annals of Statistics* 36.3 (2008), pp. 1171–1220. DOI: 10.1214/009053607000000677.

[52] Alston S. Householder. "Unitary triangularization of a nonsymmetric matrix". In: *Journal of the ACM (JACM)* 5.4 (1958), pp. 339–342. DOI: 10.1145/320941.320947.

[53] Armin Iske, Sabine Le Borne, and Michael Wende. "Hierarchical matrix approximation for kernel-based scattered data interpolation". In: *SIAM J. Sci. Comput.* 39.5 (2017), A2287–A2316. DOI: 10.1137/16M1101167.

[54] Maurice A. Jaswon and George T. Symm. *Integral equation methods in potential theory and elastostatics*. London: Academic Press, 1977.

[55] Alan Jennings. "A compact storage scheme for the solution of symmetric linear simultaneous equations". In: *The Computer Journal* 9.3 (1966), pp. 281–285. DOI: 10.1093/comjnl/9.3.281.

[56] Stephen Kirkup. "The boundary element method in acoustics: A survey". In: *Applied Sciences* 9.8 (2019), p. 1642. DOI: 10.3390/app9081642.

[57] Stephen Kirkup and Javad Yazdani. "A gentle introduction to the boundary element method in MATLAB/FREEMAT". In: *WSEAS International Conference. Proceedings. Mathematics and Computers in Science and Engineering*. 10. WSEAS. 2008.

[58] Naraparaju Kishore Kumar and Jan Schneider. "Literature survey on low rank approximation of matrices". In: *Linear and Multilinear Algebra* 65.11 (2017), pp. 2212–2244. DOI: 10.1080/03081087.2016.1267104.

[59] Daniel Kressner, Stefano Massei, and Leonardo Robol. "Low-rank updates and a divide-and-conquer method for linear matrix equations". In: *SIAM Journal on Scientific Computing* 41.2 (2019), A848–A876. DOI: 10.1137/17M1161038.

[60] Daniel Kressner and Ana Susnjara. "Fast QR decomposition of HODLR matrices". In: *arXiv: Numerical Analysis* (2018). DOI: 10.48550/ARXIV.1809.10585.

[61] Ronald Kriemann. "$\mathcal{H}$-LU factorization on many-core systems". In: *Computing and Visualization in Science* 16.3 (2013), pp. 105–117. DOI: 10.1007/s00791-014-0226-7.

[62] Sabine Le Borne. "A block Cholesky-LU-based QR factorization for rectangular matrices". In: *Numerical Linear Algebra with Applications* n/a.n/a (), e2497. DOI: 10.1002/nla.2497.

[63] Sabine Le Borne. "Factorization, symmetrization, and truncated transformation of radial basis function-GA stabilized Gaussian radial basis functions". In: *SIAM Journal on Matrix Analysis and Applications* 40.2 (2019), pp. 517–541. DOI: 10.1137/18M119207X.

[64] Sabine Le Borne. "Preconditioned nullspace method for the two-dimensional Oseen problem". In: *SIAM J. Sci. Comput.* 31 (2009), pp. 2494–2509.

[65] Sabine Le Borne and David W. Cook II. "Construction of a discrete divergence-free basis through orthogonal factorization in $\mathcal{H}$-arithmetic". In: *Computing* 81 (2007), pp. 215–238. DOI: 10.1007/s00607-007-0251-1.

[66] Michael Lintner. "Lösung der 2D Wellengleichung mittels hierarchischer Matrizen". PhD thesis. Technische Universität München, 2002.

[67] Thomas Mach. "Eigenvalue algorithms for symmetric hierarchical matrices". PhD thesis. Chemnitz University of Technology, 2012.

[68] Harry Markowitz. *Oral history interview with Harry M. Markowitz*. Charles Babbage Institute (2002). Retrieved from the University of Minnesota Digital Conservancy, https://hdl.handle.net/11299/107467.

[69] Stefano Massei, Leonardo Robol, and Daniel Kressner. "Hierarchical adaptive low-rank format with applications to discretized partial differential equations". In: *Numerical Linear Algebra with Applications* 29.6 (2022), e2448.

[70] Michael Mongillo. "Choosing basis functions and shape parameters for radial basis function methods". In: *SIAM Undergraduate Research Online* 4 (2011), pp. 190–209. DOI: 10.1137/11S010840.

[71] Vinh Phu Nguyen, Timon Rabczuk, Stéphane Bordas, and Marc Duflot. "Meshless methods: a review and computer implementation aspects". In: *Mathematics and computers in simulation* 79.3 (2008), pp. 763–813. DOI: 10.1016/j.matcom.2008.01.003.

[72] Jennifer Pestana and Tyrone Rees. "Null-Space Preconditioners for Saddle Point Systems". In: *SIAM J. on Matrix Anal. and Appl.* 37 (2016), pp. 1103–1128.

[73] Robert Schaback. "The missing Wendland functions". In: *Advances in Computational Mathematics* 34.1 (2011), pp. 67–81. DOI: 10.1007/s10444-009-9142-7.

[74] Robert Schreiber and Charles Van Loan. "A storage-efficient WY representation for products of Householder transformations". In: *SIAM Journal on Scientific and Statistical Computing* 10.1 (1989), pp. 53–57. DOI: 10.1137/0910005.

[75] Alex Schwarzenberg-Czerny. "On matrix factorization and efficient least squares solution." In: *Astronomy and Astrophysics Supplement Series* 110 (1995), p. 405.

[76] Jennifer Scott and Miroslav Tůma. *Algorithms for sparse linear systems*. Springer Nature, 2023.

[77] Matthias Seeger. "Gaussian processes for machine learning". In: *International journal of neural systems* 14.02 (2004), pp. 69–106. DOI: 10.1142/S0129065704001899.

[78] Vaclav Skala, Samsul Ariffin Abdul Karim, and Marek Zabran. "Radial basis function approximation optimal shape parameters estimation". In: *International Conference on Computational Science*. Springer. 2020, pp. 309–317. DOI: 10.1007/978-3-030-50433-5_24.

[79] Alvise Sommariva and Marco Vianello. "Numerical cubature on scattered data by radial basis functions". In: *Computing* 76.3-4 (2006), pp. 295–310. DOI: 10.1007/s00607-005-0142-2.

[80] Lloyd N. Trefethen and David Bau III. *Numerical linear algebra*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1997.

[81] Holger Wendland. "Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree". In: *Advances in computational Mathematics* 4.1 (1995), pp. 389–396. DOI: 10.1007/BF02123482.

[82] Holger Wendland. *Scattered data approximation*. Vol. 17. Cambridge Monographs on Applied and Computational Mathematics. Cambridge: Cambridge University Press, 2004.

[83] James H. Wilkinson and Christian Reinsch. *Handbook for Automatic Computation: Volume II: Linear Algebra*. Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg, 1971. ISBN: 9780387054148.

[84] Yusaku Yamamoto, Yuji Nakatsukasa, Yuka Yanagisawa, and Takeshi Fukaya. "Roundoff error analysis of the CholeskyQR2 algorithm". In: *Electronic Transactions on Numerical Analysis* 44.01 (2015), pp. 306–326.