ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
MATHICSE
Institute of Mathematics
School of Basic Sciences

MATHICSE Technical Report
Nr. 12.2018
September 2018

Fast QR decomposition of HODLR matrices

Daniel Kressner, Ana Susnjara

Address: EPFL - SB – INSTITUTE of MATHEMATICS – Math
(Bâtiment MA) Station 8 - CH-1015 - Lausanne - Switzerland

Fast QR decomposition of HODLR matrices

Daniel Kressner* Ana Šušnjara†

Abstract

The efficient and accurate QR decomposition for matrices with hierarchical low-rank structures, such as HODLR and hierarchical matrices, has been challenging. Existing structure-exploiting algorithms are prone to numerical instability as they proceed indirectly, via Cholesky decompositions or a block Gram-Schmidt procedure. For a highly ill-conditioned matrix, such approaches either break down in finite-precision arithmetic or result in significant loss of orthogonality. Although these issues can sometimes be addressed by regularization and iterative refinement, it would be more desirable to have an algorithm that avoids these detours and is numerically robust to ill-conditioning. In this work, we propose such an algorithm for HODLR matrices. It achieves accuracy by utilizing Householder reflectors. It achieves efficiency by utilizing fast operations in the HODLR format in combination with compact WY representations and the recursive QR decomposition by Elmroth and Gustavson. Numerical experiments demonstrate that our newly proposed algorithm is robust to ill-conditioning and capable of achieving numerical orthogonality down to the level of roundoff error.

1 Introduction

A HODLR (*hierarchically off-diagonal low-rank*) matrix A is defined recursively via 2×2 block partitions of the form

$$A = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right], \tag{1}$$

where the off-diagonal blocks A_{21}, A_{12} have low rank and the diagonal blocks are again HODLR matrices. The recursion is stopped once the diagonal blocks are of sufficiently small size, in the range of, say, a few hundreds. Storing the off-diagonal blocks in terms of their low-rank factors significantly reduces memory requirements and, potentially, the computational cost of operating with HODLR matrices. The goal of this work is to devise an efficient and numerically accurate algorithm for computing a QR decomposition

$$A = QR,$$

where R is an upper triangular HODLR matrix and the orthogonal matrix Q is represented in terms of its so called compact WY representation [17]: $Q = I - YTY^T$, with the identity matrix I and triangular/trapezoidal HODLR matrices T, Y.

*MATH-ANCHP, École Polytechnique Fédérale de Lausanne, Station 8, 1015 Lausanne, Switzerland. E-mail: daniel.kressner@epfl.ch.

†MATH-ANCHP, École Polytechnique Fédérale de Lausanne, Station 8, 1015 Lausanne, Switzerland. E-mail: susnjara.ana@gmail.com. The work of Ana Šušnjara has been supported by the SNSF research project *Low-rank updates of matrix functions and fast eigenvalue solvers.*

HODLR matrices constitute one of the simplest data-sparse formats among the wide range of hierarchical low-rank formats that have been discussed in the literature during the last two decades. They have proved to be effective, for example, in solving large-scale linear systems [2] and operating with multivariate Gaussian distributions [1]. In our own work [12, 20] HODLR matrices have played a central role in developing fast algorithms for solving symmetric banded eigenvalue problems. In particular, a fast variant of the so called QDWH algorithm [15, 16] for computing spectral projectors requires the QR decomposition of a HODLR matrix. It is also useful for orthonormalizing data-sparse vectors. Possibly more importantly, the QR decomposition offers a stable alternative to the LU decomposition (without pivoting) for solving linear systems with nonsymmetric HODLR matrices or to the Cholesky decomposition applied to the normal equations for solving linear least-squares problems.

For the more general class of hierarchical matrices [9], a number of approaches aim at devising fast algorithms for QR decompositions [3, 4, 13]. However, as we explain in Section 2 below, all existing approaches have limitations in terms of numerical accuracy and orthogonality, especially when A is ill-conditioned. In the other direction, when further conditions are imposed on a HODLR matrix, leading to formats such as HSS (hierarchically semi-separable) or quasi-separable matrices, then it can be possible to devise QR or URV decompositions that fully preserve the structure; see [6, 21, 22] and the references therein. This is clearly not possible for HODLR matrices: In general, Q and R do not inherit from A the property of having low-rank off-diagonal blocks. However, as it turns out, these factors can be very well *approximated* via HODLR matrices. The approach presented in this work to obtain such approximations is different from any existing approach we are aware of. It is based on the recursive QR decomposition proposed by Elmroth and Gustavson [7] for dense matrices. The key insight in this work is that such a recursive algorithm combines well with the use of the HODLR format for representing the involved compact WY representations. Demonstrated by the numerical experiments, the resulting algorithm is not only fast but it is also capable of yielding high accuracy, that is, a residual and orthogonality down to the level of roundoff error.

The rest of this paper is organized as follows. Section 2 provides an overview of HODLR matrices and the corresponding arithmetics, as well as the existing approaches to fast QR decompositions. In Section 3, we recall the recursive QR decomposition from [7] for dense matrices. The central part of this work, Section 4 combines [7] with the HODLR format. Various numerical experiments reported in Section 5 demonstrate the effectiveness of our approach. Finally, in Section 6, we sketch the extension of our newly proposed approach from square to rectangular HODLR matrices.

2 Overview of HODLR matrices and existing methods

In the following, we focus our description on *square* HODLR matrices. Although the extension of our algorithm to rectangular matrices does not require any substantially new ideas, the formal description of the algorithm would become significantly more technical. We have therefore postponed the rectangular case to Section 6.

2.1 HODLR matrices

As discussed in the introduction, a HODLR matrix $A \in \mathbb{R}^{n\times n}$ is defined by performing a recursive partition of the form (1) and requiring all occuring off-diagonal blocks to be of low rank. When this recursion is performed ℓ times, we say that A is a HODLR matrix of *level* ℓ; see Figure 1 for an illustration.

Clearly, the definition of a HODLR matrix depends on the block sizes chosen in (1) on every level of the recursion or, equivalently, on the integer partition

$$n = n_1 + n_2 + \cdots n_{2^\ell}, \tag{2}$$

defined by the sizes $n_j \times n_j$, $j = 1, \ldots, 2^\ell$, of the diagonal blocks on the lowest level of the recursion. If possible, it is advisable to choose the level ℓ and the integers n_j such that all n_j are nearly equal to a prescribed minimal block size $n_{\min}$. In the following, when discussing the complexity of operations, we assume that such a balanced partition has been chosen.

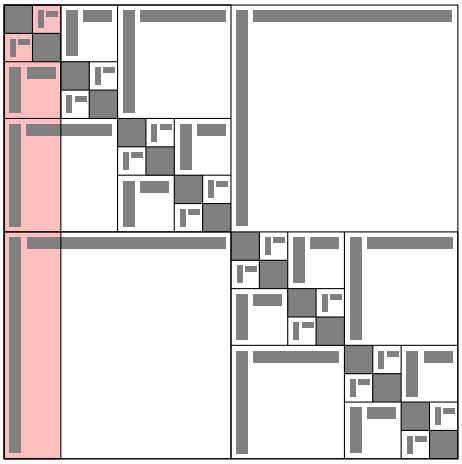


Figure 1: HODLR matrix of level $\ell = 4$. The high-lighted block column is processed recursively by our newly proposed algorithm; see Section 4. It consists of three different types of blocks: On top a level-one HODLR matrix, below a low-rank matrix entirely contained within the high-lighted block column, and two low-rank matrices extending into other block columns.

Given an integer partition (2), we define $\mathcal{H}_{n\times n}(\ell, k)$ to be the set of $n \times n$ HODLR matrices of level ℓ and rank (at most) k, that is, $A \in \mathcal{H}_{n\times n}(\ell, k)$ if every off-diagonal block of A in the recursive block partition induced by (2) has rank at most k.

A matrix $A \in \mathcal{H}_{n\times n}(\ell, k)$ admits a data-sparse representation by storing its off-diagonal blocks in terms of their low-rank factors. Specifically, letting $A|_{\text{off}} \in \mathbb{R}^{n_L \times n_R}$ denote an arbitrary off-diagonal block in the recursive partition of A, we can write

$$A|_{\text{off}} = A_L A_R, \quad A_L \in \mathbb{R}^{n_L \times k}, \quad A_R \in \mathbb{R}^{k \times n_R}. \tag{3}$$

Storing A_L and A_R instead of $A|_{\text{off}}$ for every such off-diagonal block reduces the overall memory required for storing A from $\mathcal{O}(n^2)$ to $\mathcal{O}(kn \log n)$.

We call a factorization (3) *left-orthogonal* if $A_L^T A_L = I_k$. Provided that $k \leq n_L$, an arbitrary factorization (3) can be turned into a left-orthogonal one by computing an economy-sized QR decomposition $A_L = QR$, see [8, Theorem 5.2.3], and replacing $A_L \leftarrow Q$, $A_R \leftarrow RA_R$. This described procedure requires $\mathcal{O}((n_L + n_R)k^2)$ operations.

2.1.1 Approximation by HODLR matrices

A general matrix $A \in \mathbb{R}^{n\times n}$ can be approximated by a HODLR matrix by performing low-rank truncations of the off-diagonal blocks in the recursive block partition. Specifically, letting $A|_{\text{off}}$ denote such an off-diagonal block, one computes a singular value decomposition $A|_{\text{off}} = U\Sigma V^T$

with the diagonal matrix $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots)$ containing the singular values. Letting U_k and V_k contain the first k columns of U and V, respectively, and setting $\Sigma_k = \mathrm{diag}(\sigma_1, \ldots, \sigma_k)$, one obtains a rank-$k$ approximation

$$A|_{\mathrm{off}} \approx A_L A_R \tag{4}$$

by setting $A_L = U_k$ and $A_R = \Sigma_k V_k^T$. We note that this approximation is optimal among all rank-k matrices for any unitarily invariant norm [11, Section 7.4.9]. In particular, for the matrix 2-norm, we have

$$\|A|_{\mathrm{off}} - A_L A_R\|_2 = \sigma_{k+1}. \tag{5}$$

In passing, we note that the factorization chosen in (4) is left-orthogonal.

Performing the approximation (4) for every off-diagonal block in the recursive block partition yields a HODLR matrix $A_{\mathcal{H},k} \in \mathcal{H}_{n\times n}(\ell, k)$.

In practice, the rank k is chosen adaptively and separately for each off-diagonal block $A|_{\mathrm{off}}$. Given a prescribed tolerance $\epsilon > 0$, we choose $k \equiv k_\epsilon$ to be the smallest integer such that $\sigma_{k_\epsilon+1} \leq \epsilon$. In turn, (5) implies $\|A|_{\mathrm{off}} - A_L A_R\|_2 \leq \epsilon$. The resulting HODLR approximation $A_{\mathcal{H},\epsilon}$ satisfies $\|A - A_{\mathcal{H},\epsilon}\|_2 \leq \ell\epsilon$; see, e.g., [5, Theorem 2.2].

Recompression. Most manipulations involving HODLR matrices lead to an increase of off-diagonal ranks. This increase is potentially mitigated by performing recompression. Let us consider an off-diagonal block $A|_{\mathrm{off}} = A_L A_R$, with $A_L \in \mathbb{R}^{n_L\times k_A}, A_R \in \mathbb{R}^{k_A\times n_R}$ and choose k_ϵ as explained above. If $k_\epsilon < k_A$, a rank-k_ϵ approximation reduces memory requirements while maintaining ϵ-accuracy. We use the following well-known procedure for effecting this approximation.

1. Compute $A_L = Q_1 R_1$ and $A_R^T = Q_2 R_2$, economy-sized QR decompositions of A_L and A_R^T, respectively.
2. Compute SVD $R_1 R_2^T = \tilde{U}\Sigma\tilde{V}^T$.
3. Update $A_L \leftarrow Q_1\tilde{U}_{k_\epsilon}$ and $A_R \leftarrow \Sigma_{k_\epsilon}\tilde{V}_{k_\epsilon}^T Q_2^T$.

This procedure, which will be denoted by $\mathcal{T}_\epsilon$, requires $\mathcal{O}\left((n_L + n_R)k_A^2\right)$ operations.

2.1.2 Operating with HODLR matrices

A number of operations can be performed efficiently with HODLR matrices. Table 1 lists the operations relevant in this work, together with their computational complexity; see, e.g., [9, Chapter 3] for more details. It is important to note that all operations, except for matrix-vector multiplication, are combined with low-rank truncation, as discussed above, to limit rank growth in the off-diagonal blocks. The symbol $\mathcal{H}$ signifies the inexactness due to truncations. The complexity estimates assume that all off-diagonal ranks encountered during an operation remain $\mathcal{O}(k)$.

2.2 Cholesky-based QR decomposition

This and the following sections describe three existing methods for efficiently computing the QR decomposition of a HODLR matrix. All these methods have originally been proposed for the broader class of hierarchical matrices.

Table 1: Complexity of operations with HODLR matrices: $A_1, \ldots, A_6 \in \mathcal{H}_{n\times n}(\ell, k)$, with A_3 invertible, A_4 invertible upper triangular, A_6 symmetric positive definite, $U, V \in \mathbb{R}^{n\times p}$ with $p = \mathcal{O}(k)$, and $v \in \mathbb{R}^n$.

Operation	Computational complexity
Matrix-vector multiplication: $A_1 v$	$\mathcal{O}(kn \log n)$
Matrix addition: $A_1 +_{\mathcal{H}} A_2$	$\mathcal{O}(k^2 n \log n)$
Matrix low-rank update: $A_1 +_{\mathcal{H}} UV^T$	$\mathcal{O}(k^2 n \log n)$
Matrix-matrix multiplication: $A_1 *_{\mathcal{H}} A_2$	$\mathcal{O}(k^2 n \log^2 n)$
Matrix inversion: $\mathcal{H}$-inv(A_3)	$\mathcal{O}(k^2 n \log^2 n)$
Solution of triangular matrix equation: $A_5 *_{\mathcal{H}} A_4^{-1}$	$\mathcal{O}(k^2 n \log^2 n)$
Cholesky decomposition: $\mathcal{H}$-Cholesky(A_6)	$\mathcal{O}(k^2 n \log^2 n)$

The first method, proposed by Lintner [13, 14], is based on the well-known connection between the QR and Cholesky decompositions. Specifically, letting $A = QR$ be the QR decomposition of an invertible $n \times n$ matrix A, we have

$$A^T A = R^T Q^T Q R = R^T R.$$

Thus, the upper triangular factor R can be obtained from the Cholesky decomposition of the symmetric positive definite matrix $A^T A$. The orthogonal factor Q is obtained from solving the triangular system $A = QR$. According to Table 1, these three steps (forming $A^T A$, computing the Cholesky decomposition, solving the triangular matrix equation) require $\mathcal{O}(k^2 n \log^2 n)$ operations in the HODLR format.

For dense matrices, the approach described above is well-known and often called *CholeskyQR algorithm*; see [19, Pg. 214] for an early reference. A major disadvantage of this approach, Q rapidly loses orthogonality in finite precision arithmetic as the condition number of A increases. As noted in [18], the numerical orthogonality $\|Q^T Q - I\|_2$ is usually at the level of the squared condition number $\kappa(A^T A) = \kappa(A)^2$ times the unit roundoff u. To improve its orthogonality, one can apply the CholeskyQR algorithm again to Q and update R accordingly. As shown in [23], this so called *CholeskyQR2 algorithm* results in a numerically orthogonal factor, provided that $\kappa(A)$ is at most $\mathcal{O}(\mathsf{u}^{-1/2})$.

The CholeskyQR2 algorithm for HODLR and hierarchical matrices [13] is additionally affected by low-rank truncation and may require several reorthogonalization steps to reach numerical orthogonality on the level of the truncation error, increasing the computational cost. Another approach proposed in [13] to avoid loss of orthogonality is to first compute a polar decomposition $A = QH$ and then apply the CholeskyQR algorithm to H. Because of $\kappa(H) = \kappa(A) = \sqrt{\kappa(A^T A)}$, this improves the accuracy of the CholeskyQR algorithm. On the other hand, the need for computing the polar decomposition via an iterative method, such as the sign-function iteration [10], also significantly increases the computational cost.

2.3 LU-based QR decomposition

An approach proposed by Bebendorf [3, Sec. 2.10] can be viewed as orthogonalizing a recursive block LU decomposition. Given $A \in \mathbb{R}^{n\times n}$, let us partition

$$A = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right] \tag{6}$$

and suppose that A_{11} is invertible. Setting $X = A_{21}A_{11}^{-1}$, consider the block LU decomposition

$$A = \begin{bmatrix} I & 0 \\ X & I \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} - XA_{12} \end{bmatrix}. \tag{7}$$

The first factor is orthogonalized by (1) rescaling the first block column with the inverted Cholesky factor of the symmetric positive definite matrix $I + X^T X = R_1^T R_1$ and (2) choosing the second block column as $\left[\begin{smallmatrix} -X^T \\ I \end{smallmatrix}\right]$, scaled with the inverted Cholesky factor of $I + XX^T = R_2^T R_2$. Adjusting the second factor in (7) accordingly does not change its block triangular structure. More precisely, one can prove that

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

holds. By construction, $\tilde{Q}$ is orthogonal. This procedure is applied recursively to the diagonal blocks of $\tilde{R}$. If A is a HODLR matrix corresponding to the partition (6) at every level of the recursion then all involved operations can be performed efficiently in the HODLR format. Following [3], the overall computational cost is, once again, $\mathcal{O}(k^2 n \log^2 n)$.

An obvious disadvantage of the described approach, it requires the leading diagonal block A_{11} to be well conditioned for every subproblem encountered during the recursion. This rather restrictive assumption is only guaranteed for specific matrix classes, such as well-conditioned positive definite matrices.

2.4 QR decomposition based on a block Gram-Schmidt procedure

The equivalence between the QR decomposition and the Gram-Schmidt procedure for full-rank matrices is well known. In particular, applying the modified block Gram-Schmidt procedure to the columns of A leads to the block recursive QR decomposition presented in [8, Sec. 5.2.4]. Benner and Mach [4] combined this idea with hierarchical matrix arithmetic. In the following, we briefly summarize their approach. Partitioning the economy-sized QR decomposition of A into block columns yields the relation

$$[A_1\ A_2] = [Q_1\ Q_2] \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix}. \tag{8}$$

This yields three steps for the computation of Q and R:

1. Compute (recursively) the QR decomposition $A_1 = Q_1 R_{11}$.
2. Compute $R_{12} = Q_1^T A_2$ and update $A_2 \leftarrow A_2 - Q_1 R_{12}$.
3. Compute (recursively) the QR decomposition $A_2 = Q_2 R_{22}$.

Step 2 can be implemented efficiently for a HODLR matrix that aligns with the block column partitioning (8). Steps 1 and 3 are executed recursively until the lowest level of the HODLR structure is reached. On this lowest level, it is suggested in [4, Alg. 3] to compute the QR decomposition of compressed block columns. We refrain from providing details and point out that we consider similarly compressed block columns in Section 4 below. The overall computational complexity is $\mathcal{O}(k^2 n \log^2 n)$.

The described algorithm inherits the numerical instability of Gram-Schmidt procedures. In particular, we cannot expect to obtain a numerically orthogonal factor Q in finite-precision arithmetic when A is ill-conditioned; see also the analysis in [4, Sec. 3.4].

3 Recursive WY-based QR decomposition

In this section, we recall the recursive QR decomposition by Elmroth and Gustavson [7] for a general, dense $m \times n$ matrix A with $m \geq n$. The orthogonal factor Q is returned in terms of the compact WY representation [17] of the n Householder reflectors involved in the decomposition:

$$Q = I_m - YTY^T, \tag{9}$$

where T is an $n \times n$ upper triangular matrix and Y is an $m \times n$ matrix with the first n rows in unit lower triangular form.

For $n = 1$, the matrix A becomes a column vector and we let $Q = I_m - \gamma yy^T$ be the Householder reflector [8, Sec. 5.1.2] that maps A to a scalar multiple of the unit vector. Then Q is trivially of the form (9).

For $n > 1$, we partition A into two block columns of roughly equal size:

$$A = \left[\begin{array}{c|c} A_1 & A_2 \end{array}\right], \quad A_1 \in \mathbb{R}^{m \times n_1}, \quad A_2 \in \mathbb{R}^{m \times n_2}, \quad n = n_1 + n_2.$$

By recursion, we compute a QR decomposition of the first block column

$$A_1 = Q_1 \begin{bmatrix} R_1 \\ 0 \end{bmatrix}, \quad Q_1 = I_m - Y_1 T_1 Y_1^T,$$

with $T_1 \in \mathbb{R}^{n_1 \times n_1}$, $Y_1 \in \mathbb{R}^{m \times n_1}$ taking the form explained above. The second block column A_2 is updated,

$$\tilde{A}_2 = Q_1^T A_2 = A_2 - Y_1 T_1 (Y_1^T A_2),$$

and then partitioned as

$$\tilde{A}_2 = \begin{bmatrix} A_{12} \\ A_{22} \end{bmatrix}, \quad A_{12} \in \mathbb{R}^{n_1 \times n_2}, \quad A_{22} \in \mathbb{R}^{(m-n_1) \times n_2}.$$

Again by recursion, we compute a QR decomposition of the bottom block:

$$A_{22} = Q_2 \begin{bmatrix} R_2 \\ 0 \end{bmatrix}, \quad Q_2 = I_{m-n_1} - Y_2 T_2 Y_2^T.$$

To combine the QR decompositions of the first and the updated second block column, we embedd Q_2 into the larger matrix

$$\tilde{Q}_2 = \begin{bmatrix} I_{n_1} & 0 \\ 0 & Q_2 \end{bmatrix} = I_m - \tilde{Y}_2 T_2 \tilde{Y}_2^T, \quad \tilde{Y}_2 = \begin{bmatrix} 0 \\ Y_2 \end{bmatrix}.$$

By setting

$$R = \left[\begin{array}{c|c} R_1 & A_{12} \\ \hline 0 & R_2 \end{array}\right]$$

and

$$\begin{aligned} Q &= Q_1 \tilde{Q}_2 = \left(I_m - Y_1 T_1 Y_1^T\right)\left(I_m - \tilde{Y}_2 T_2 \tilde{Y}_2^T\right) \\ &= I_m - Y_1 T_1 Y_1^T - \tilde{Y}_2 T_2 \tilde{Y}_2^T + Y_1 T_1 Y_1^T \tilde{Y}_2 T_2 \tilde{Y}_2^T \\ &= I_m - \left[\begin{array}{c|c} Y_1 & \tilde{Y}_2 \end{array}\right] \left[\begin{array}{c|c} T_1 & -T_1 Y_1^T \tilde{Y}_2 T_2 \\ \hline 0 & T_2 \end{array}\right] \left[\begin{array}{c|c} Y_1 & \tilde{Y}_2 \end{array}\right]^T, \end{aligned} \tag{10}$$

we obtain a QR decomposition

$$A = Q \begin{bmatrix} R \\ 0 \end{bmatrix}, \quad Q = I - YTY^T, \quad Y = \left[\begin{array}{c|c} Y_1 & \tilde{Y}_2 \end{array}\right], \quad T = \left[\begin{array}{c|c} T_1 & -T_1 Y_1^T \tilde{Y}_2 T_2 \\ \hline 0 & T_2 \end{array}\right].$$

Algorithm 1 summarizes the described procedure. To simplify the description, the recursion is performed down to individual columns. In practice [7], the recursion is stopped earlier: When the number of columns does not exceed a certain block size n_b (e.g., $n_b = 32$), a standard Householder-based QR decomposition is used.

Algorithm 1 Recursive block QR decomposition

Input: Matrix $A \in \mathbb{R}^{m \times n}$ with $m \geq n$.

Output: Matrices $Y \in \mathbb{R}^{m \times n}$, $T \in \mathbb{R}^{n \times n}$, $R \in \mathbb{R}^{n \times n}$, defining a QR decomposition $A = Q \begin{bmatrix} R \\ 0 \end{bmatrix}$ with $Q = I_m - YTY^T \in \mathbb{R}^{m \times m}$ orthogonal.

function $[Y, T, R] = \texttt{blockQR}(A)$
if $n = 1$ **then**
  Compute Householder reflector $I_m - \gamma y y^T$ such that $(I_m - \gamma y y^T)A = \begin{bmatrix} \rho \\ 0 \end{bmatrix}$.
  Set $Y = y$, $T = \gamma$ and $R = \rho$.
else
  Set $n_1 = \lfloor n/2 \rfloor$.
  Call $[Y_1, T_1, R_1] = \texttt{blockQR}(A(:, 1 : n_1))$.
  Update $A(:, n_1 + 1 : n) \leftarrow (I - Y_1 T Y_1^T)^T A(:, n_1 + 1 : n)$.
  Set $[Y_2, T_2, R_2] = \texttt{blockQR}(A(n_1 + 1 : m, n_1 + 1 : n))$.
  Set $\tilde{Y}_2 = \begin{bmatrix} 0 \\ Y_2 \end{bmatrix}$ and compute $T_{12} = -T_1 Y_1^T \tilde{Y}_2 T_2$.
  Return $Y = \begin{bmatrix} Y_1 & \tilde{Y}_2 \end{bmatrix}$, $T = \begin{bmatrix} T_1 & T_{12} \\ 0 & T_2 \end{bmatrix}$ and $R = \begin{bmatrix} R_1 & A(1 : n_1, n_1 + 1 : n) \\ 0 & R_2 \end{bmatrix}$.
end if
end function

4 Recursive WY-based QR decomposition of HODLR matrices

By combining the recursive block QR decomposition (Algorithm 1) with HODLR arithmetic, we will show in this section how to derive an efficient algorithm for computing the QR decomposition of a level-ℓ HODLR matrix $A \in \mathbb{R}^{n \times n}$.

The matrix processed in one step of the recursion of our algorithm takes the following form:

$$H = \begin{bmatrix} \tilde{A} \\ B \\ C \end{bmatrix}, \tag{11}$$

where:

- $\tilde{A} \in \mathbb{R}^{m \times m}$ is a HODLR matrix of level $\tilde{\ell} \leq \ell$;

- $B \in \mathbb{R}^{p\times m}$ is given in factorized form $B = B_L B_R$ with $B_L \in \mathbb{R}^{p\times r_1}$ and $B_R \in \mathbb{R}^{r_1\times m}$ for some (small) integer r_1;
- $C \in \mathbb{R}^{r_2\times m}$ for some (small) integer r_2.

To motivate this structure, it is helpful to consider the block column highlighted in Figure 1. The first block in this block column consists of a (square) level-one HODLR matrix, corresponding to the matrix $\tilde{A}$ in (11). The other three blocks are all of low rank, and the matrix B in (11) corresponds to the first of these blocks. The bottom two blocks extend into the next block column(s). For these two blocks, it is assumed that their left factors are orthonormal. These left factors are ignored and the parts of the right factors residing in the highlighted block column are collected in the matrix C in (11).

Given a matrix H of the form (11), we aim at computing, recursively and approximately, a QR decomposition of the form

$$H = Q \begin{bmatrix} R \\ 0 \end{bmatrix}, \quad Q = I - YTY^T \tag{12}$$

such that $R, T \in \mathbb{R}^{m\times m}$ are upper triangular HODLR matrices of level $\tilde{\ell}$ and the structure of Y reflects the structure of H, that is,

$$Y = \begin{bmatrix} Y_A \\ Y_B \\ Y_C \end{bmatrix}, \tag{13}$$

where: $Y_A \in \mathbb{R}^{m\times m}$ is a unit lower triangular HODLR matrix of level $\tilde{\ell}$; $Y_B \in \mathbb{R}^{p\times m}$ is in factorized form; and $Y_C \in \mathbb{R}^{r_2\times m}$.

On the highest level of the recursion, when $\tilde{\ell} = \ell$, the matrices B, C in (11) vanish, and $H = A$ is the original HODLR matrix we aim at decomposing. The QR decomposition returned on the highest level has the form (12) with both Y, T triangular level-ℓ HODLR matrices.

The computation of the QR decomposition (12) proceeds in several steps, which are detailed in the following.

Preprocessing. Using the procedure described in Section 2.1.2, we may assume that the factorization of B is normalized such that B_L has orthonormal columns. For the moment, we will discard B_L and aim at decomposing instead of H the compressed matrix

$$\tilde{H} = \begin{bmatrix} \tilde{A} \\ B_R \\ C \end{bmatrix}, \tag{14}$$

which has size $(m + r_1 + r_2) \times m$.

QR decomposition of $\tilde{H}$ on the lowest level, $\tilde{\ell} = 0$. On the lowest level of recursion, $\tilde{A}$ becomes a dense matrix. We perform a dense QR decomposition of the matrix $\tilde{H}$ defined in (14). For this purpose, one can use, for example, Algorithm 1. This yields the orthogonal factor $\tilde{Q}$ in terms of its compact WY representation, which we partition as

$$\tilde{Q} = I - \tilde{Y}T\tilde{Y}^T, \qquad \tilde{Y} = \begin{bmatrix} Y_A \\ \tilde{Y}_B \\ Y_C \end{bmatrix}, \quad Y_A \in \mathbb{R}^{m\times m}, \quad \tilde{Y}_B \in \mathbb{R}^{r_1\times m}, \quad Y_C \in \mathbb{R}^{r_2\times m}. \tag{15}$$

QR decomposition of $\tilde{H}$ on higher levels, $\tilde{\ell} \geq 1$. We proceed recursively as follows. First, $\tilde{H}$ is repartitioned as follows:

$$\tilde{H} = \begin{bmatrix} \tilde{A}_{11} & \tilde{A}_{12} \\ \tilde{A}_{21} & \tilde{A}_{22} \\ B_{R,1} & B_{R,2} \\ C_1 & C_2 \end{bmatrix}. \tag{16}$$

Here, $\tilde{A} = \begin{bmatrix} \tilde{A}_{11} & \tilde{A}_{12} \\ \tilde{A}_{21} & \tilde{A}_{22} \end{bmatrix}$ is split according to its HODLR format, that is, $\tilde{A}_{11} \in \mathbb{R}^{m_1 \times m_1}$, $\tilde{A}_{22} \in \mathbb{R}^{m_2 \times m_2}$, with $m = m_1 + m_2$, are HODLR matrices of level $\tilde{\ell} - 1$, and $\tilde{A}_{21}, \tilde{A}_{12}$ are low-rank matrices stored in factorized form.

Note that the first block column of $\tilde{H}$ in (16) has precisely the form (11) with the level of the HODLR matrix reduced by one, the low-rank block given by $\tilde{A}_{21}$ and the dense part given by $\begin{bmatrix} B_{R,1} \\ C_1 \end{bmatrix}$. This allows us to apply recursion and obtain a QR decomposition

$$\begin{bmatrix} \tilde{A}_{11} \\ \tilde{A}_{21} \\ B_{R,1} \\ C_1 \end{bmatrix} = Q_1 \begin{bmatrix} R_1 \\ 0 \end{bmatrix}, \quad Q_1 = I - Y_1 T_1 Y_1^T, \quad Y_1 = \begin{bmatrix} Y_{A,11} \\ Y_{A,21} \\ Y_{B_R,1} \\ Y_{C,1} \end{bmatrix}, \tag{17}$$

with a HODLR matrix $Y_{A,11}$ and a factorized low-rank matrix $Y_{A,21}$. We then update the second block column of $\tilde{H}$:

$$\begin{bmatrix} \hat{A}_{12} \\ \hat{A}_{22} \\ \hat{B}_{R,2} \\ \hat{C}_2 \end{bmatrix} := Q_1^T \begin{bmatrix} \tilde{A}_{12} \\ \tilde{A}_{22} \\ B_{R,2} \\ C_2 \end{bmatrix} = \begin{bmatrix} \tilde{A}_{12} - Y_{A,11} S \\ \tilde{A}_{22} - Y_{A,21} S \\ B_{R,2} - Y_{B_R,1} S \\ C_2 - Y_{C,1} S \end{bmatrix}, \tag{18}$$

where

$$S := T_1^T Y_1^T \begin{bmatrix} \tilde{A}_{12} \\ \tilde{A}_{22} \\ B_{R,2} \\ C_2 \end{bmatrix} = T_1^T (Y_{A,11}^T \tilde{A}_{12} + Y_{A,21}^T \tilde{A}_{22} + Y_{B_R,1}^T B_{R,2} + Y_{C,1}^T C_2).$$

It is important to note that each term of the sum in the latter expression is a low-rank matrix and, in turn, S has low rank. This not only makes the computation of S efficient but it also implies that the updates in (18) are of low rank and thus preserve the structure of the second block column of $\tilde{H}$.

After the update (18) has been performed, the process is completed by applying recursion to the updated second block column (18), without the first block, and obtain a QR decomposition

$$\begin{bmatrix} \hat{A}_{22} \\ \hat{B}_{R,2} \\ \hat{C}_2 \end{bmatrix} = Q_2 \begin{bmatrix} R_2 \\ 0 \end{bmatrix}, \quad Q_2 = I - Y_2 T_2 Y_2^T, \quad Y_2 = \begin{bmatrix} Y_{A,22} \\ Y_{B_R,2} \\ Y_{C,2} \end{bmatrix}. \tag{19}$$

By the discussion in Section 3, see in particular (10), combining the QR decompositions of the first and second block columns yields a QR decomposition of $\tilde{H}$:

$$\tilde{H} = \tilde{Q} \begin{bmatrix} R \\ 0 \end{bmatrix}, \quad \tilde{Q} = I - \tilde{Y} T \tilde{Y}^T \tag{20}$$

with

$$\tilde{Y} = \left[\begin{array}{c} Y_A \\ \tilde{Y}_B \\ Y_C \end{array}\right],\ Y_A = \left[\begin{array}{c|c} Y_{A,11} & 0 \\ Y_{A,21} & Y_{A,22} \end{array}\right],\ \tilde{Y}_B = \left[\begin{array}{c|c} Y_{B_R,1} & Y_{B_R,2} \end{array}\right],\ Y_C = \left[\begin{array}{c|c} Y_{C,1} & Y_{C,2} \end{array}\right]$$

and

$$R = \left[\begin{array}{c|c} R_1 & \hat{A}_{12} \\ \hline 0 & R_2 \end{array}\right], \quad T = \left[\begin{array}{c|c} T_1 & -T_1(Y_{A,21}^T Y_{A,22} + Y_{B_R,1}^T Y_{B_R,2} + Y_{C,1}^T Y_{C,2})T_2 \\ \hline 0 & T_2 \end{array}\right].$$

Note that Y_A, R, and T are triangular level-$\tilde{\ell}$ HODLR matrices, as desired.

Postprocessing. The procedure is completed by turning the obtained QR decomposition of $\tilde{H}$ into a QR decomposition of the matrix H from (11). For this purpose, we simply set $Y_B = B_L \tilde{Y}_B$ and define Y as in (13). Setting $Q = I - YTY^T$ then yields

$$\begin{aligned} Q^T H &= H - YT^T Y^T H = \operatorname{diag}(I_m, B_L, I_{r_2})\big(\tilde{H} - \tilde{Y}T^T\tilde{Y}^T\tilde{H}\big) \\ &= \operatorname{diag}(I_m, B_L, I_{r_2})\tilde{Q}^T\tilde{H} = \operatorname{diag}(I_m, B_L, I_{r_2}) \begin{bmatrix} R \\ 0 \end{bmatrix} = \begin{bmatrix} R \\ 0 \end{bmatrix}. \end{aligned}$$

Thus, we have obtained a QR decomposition of the form (12), which concludes the recursion step.

4.1 Algorithm and complexity estimates

Algorithm 2 summarizes the recursive procedure described above. A QR decomposition of a level-ℓ HODLR matrix $A \in \mathbb{R}^{n\times n}$ is obtained by applying this algorithm with $\tilde{A} = A$ and void B_L, B_R, C.

In the following, we derive complexity estimates for Algorithm 2 applied to A under the assumptions stated in Section 2.1.2. In particular, it is assumed that all off-diagonal ranks (which are chosen adaptively) are bounded by k.

Line 3. Every lower off-diagonal block of A needs to be transformed once to left-orthogonal form in the course of the algorithm. For each $\tilde{\ell}$, $1 \le \tilde{\ell} \le \ell$, there are $2^{\ell-\tilde{\ell}}$ such blocks of size $\mathcal{O}(2^{\tilde{\ell}-1}) \times \mathcal{O}(2^{\tilde{\ell}-1})$ and rank at most k. Using the procedure for left-orthogonalization explained in Section 2.1, the overall cost is

$$\sum_{\tilde{\ell}=1}^{\ell} \mathcal{O}\big(2^{\ell-\tilde{\ell}} 2^{\tilde{\ell}-1} k^2\big) = \mathcal{O}(k^2 n \log n),$$

where we used $\ell = \mathcal{O}(\log n)$.

Line 7. The QR decomposition of the compressed block column is performed for all 2^ℓ block columns on the lowest level of recursion. Each of them is of size $\mathcal{O}(\ell k) \times \mathcal{O}(1)$, because there are at most ℓ lower off-diagonal blocks intersecting with each block column. Each QR decomposition requires $\mathcal{O}(\ell k)$ operations and thus the overall cost is $\mathcal{O}(kn \log n)$.

Algorithm 2 Recursive Householder based QR decomposition for HODLR matrices (hQR)

Input: Level-$\tilde{\ell}$ HODLR matrix $\tilde{A}$, matrices B_L, B_R, C defining the matrix H in (11).
Output: Matrix Y of the form (13), upper triangular level-$\tilde{l}$ HODLR matrices T, R defining an (approximate) QR decomposition of $H = (I - YTY^T)\begin{bmatrix} R \\ 0 \end{bmatrix}$.

1: **function** $[Y, T, R] = \texttt{hQR}(\tilde{A}, B, C)$
2: **if** B_L is not orthonormal **then**
3: Compute economy-sized QR decomposition $B_L = QR$ and set $B_L \leftarrow Q, B_R \leftarrow RB_R$.
4: **end if**
5: Set $\tilde{H} = \begin{bmatrix} \tilde{A} \\ B_R \\ C \end{bmatrix}$.
6: **if** $\tilde{\ell} = 0$ **then**
7: Use Alg. 1 to compute QR decomposition $\tilde{H} = (I - YTY^T)\begin{bmatrix} R \\ 0 \end{bmatrix}$ and partition $\tilde{Y} = \begin{bmatrix} Y_A \\ \tilde{Y}_B \\ Y_C \end{bmatrix}$.
8: **else**
9: Repartition $\tilde{H} = \begin{bmatrix} \tilde{A}_{11} & \tilde{A}_{12} \\ \tilde{A}_{21} & \tilde{A}_{22} \\ B_{R,1} & B_{R,2} \\ C_1 & C_2 \end{bmatrix}$ according to the HODLR format of $\tilde{A}$.
10: Compute QR decomposition of first block column of $\tilde{H}$ recursively:
$[Y_1, T_1, R_1] = \texttt{hQR}(\tilde{A}_{11}, \tilde{A}_{21}, \begin{bmatrix} B_{R,1} \\ C_1 \end{bmatrix})$, with Y_1 defined by $Y_{A,11}$, Y_{21}, $\begin{bmatrix} Y_{B_R,1} \\ Y_{C,1} \end{bmatrix}$; see (17).
11: Compute $\tilde{S} = \mathcal{T}_{\epsilon \cdot \|A\|_2}(Y_{A,11}^T \tilde{A}_{12} + Y_{A,21}^T \tilde{A}_{22} + Y_{B_R,1}^T B_{R,2} + Y_{C,1}^T C_2)$.
12: Compute $S = T_1^T \tilde{S}$.
13: Update second block column of $\tilde{H}$: $\begin{bmatrix} \hat{A}_{12} \\ \hat{A}_{22} \\ \hat{B}_{R,2} \\ \hat{C}_2 \end{bmatrix} := \begin{bmatrix} \mathcal{T}_{\epsilon \cdot \|A\|_2}(\tilde{A}_{12} - Y_{A,11} S) \\ \tilde{A}_{22} - \mathcal{U} Y_{A,21} S \\ B_{R,2} - Y_{B_R,1} S \\ C_2 - Y_{C,1} S \end{bmatrix}$.
14: Compute QR decomposition of unreduced part of second block column of $\tilde{H}$ recursively:
$[Y_2, T_2, R_2] = \texttt{hQR}(\hat{A}_{22}, [\,], \begin{bmatrix} B_{R,2} \\ \hat{C}_2 \end{bmatrix})$, with Y_2 defined by $Y_{A,22}$, $\begin{bmatrix} Y_{B_R,2} \\ Y_{C,2} \end{bmatrix}$; see (19).
15: Compute $\tilde{T}_{12} = \mathcal{T}_\epsilon(Y_{A,21}^T Y_{A,22} + Y_{B_R,1}^T Y_{B_R,2} + Y_{C,1}^T Y_{C,2})$.
16: Compute $T_{12} = -T_1 \tilde{T}_{12} T_2$.
17: Set $T = \begin{bmatrix} T_1 & T_{12} \\ 0 & T_2 \end{bmatrix}$, and $R = \begin{bmatrix} R_1 & \hat{A}_{12} \\ 0 & R_2 \end{bmatrix}$.
18: Set $Y_A = \left[\begin{array}{c|c} Y_{A,11} & 0 \\ Y_{A,21} & Y_{A,22} \end{array}\right]$, $\tilde{Y}_B = \left[\begin{array}{c|c} Y_{B_R,1} & Y_{B_R,2} \end{array}\right]$, and $Y_C = \left[\begin{array}{c|c} Y_{C,1} & Y_{C,2} \end{array}\right]$.
19: **end if**
20: Return T, R and $Y = \begin{bmatrix} Y_A \\ B_L \tilde{Y}_B \\ Y_C \end{bmatrix}$.
21: **end function**

Lines 11–12. The computation of S involves the following operations:

1. three products of level-$(\tilde{\ell} - 1)$ HODLR matrices with low-rank matrices;
2. addition of four low-rank matrices, given in terms of their low-rank factors, combined with recompression.

The first part is effected by performing at most $3k$ matrix-vector multiplications with $\mathcal{O}(2^{\tilde{\ell}-1}) \times \mathcal{O}(2^{\tilde{\ell}-1})$ HODLR matrices. As the computation of S is performed $2^{\ell-\tilde{\ell}}$ times for every $\tilde{\ell}$, we

arrive at a total cost of

$$\sum_{\tilde{\ell}=1}^{\ell} \mathcal{O}\big(2^{\ell-\tilde{\ell}} k^2 2^{\tilde{\ell}-1} \tilde{\ell}\big) = \mathcal{O}(k^2 n \log^2 n). \tag{21}$$

In the second part, the addition is performed for matrices of size $\mathcal{O}(2^{\tilde{\ell}-1}) \times \mathcal{O}(2^{\tilde{\ell}-1})$. The first three terms in Line 11 have rank at most k but the rank of the last term $Y_{C,1}^T C_2$ can be up to $(\ell-1)k$. Letting the rank grow to $\mathcal{O}(\ell k)$ would lead to an unfavourable complexity, because the cost of recompression depends quadratically on the rank of the matrix to be recompressed. To avoid this effect, we execute $(\ell+1)k$ separate additions, each immediately followed by the application of $\mathcal{T}_{\epsilon \cdot \|A\|_2}$. Assuming that each recompression truncates to rank $\mathcal{O}(k)$, this requires $\mathcal{O}(2^{\tilde{\ell}-1} \ell k^2)$ operations. Similarly as in (21), this leads to a total cost of $\mathcal{O}(k^2 n \log^2 n)$.

Line 13. For updating the second block column of $\tilde{H}$, the following operations are performed:

1. The computation of $\hat{A}_{12}$ requires k HODLR matrix-vector multiplications, followed by low-rank recompression of a matrix of rank at most $2k$. Analogously to (21), this requires a total cost of $\mathcal{O}(k^2 n \log^2 n)$.
2. The computation of $\hat{A}_{22}$ requires (approximate) subtraction of the product of two low-rank matrices from a level-$(\tilde{\ell}-1)$ HODLR matrix. The most expensive part of this step is the recompression of the updated HODLR matrix with off-diagonal ranks at most $2k$, amounting to a total cost of

$$\sum_{\tilde{\ell}=1}^{\ell} \mathcal{O}\big(2^{\ell-\tilde{\ell}} k^2 2^{\tilde{\ell}-1} \log 2^{\tilde{\ell}-1}\big) = \mathcal{O}(k^2 n \log^2 n).$$

3. The computation of $\hat{B}_{R,2}$ and $\hat{C}_{R,2}$ involves the multiplication of a matrix with at most $k + k\ell$ rows with a low-rank matrix, which requires $\mathcal{O}(k^2 \ell 2^{\tilde{\ell}-1})$ operations each time. The total cost is thus again $\mathcal{O}(k^2 n \log^2 n)$.

Lines 15–16. The computation of the low-rank block T_{12} involves:

1. three multiplications of level-$(\tilde{\ell}-1)$ HODLR matrices with low-rank matrices;
2. addition of three low-rank matrices, given in terms of their low-rank factors, combined with recompression.

Therefore, total cost is identical with the cost for computing S: $\mathcal{O}(k^2 n \log^2 n)$.

Summary. The total cost of Algorithm 2 applied to an $n \times n$ HODLR matrix is $\mathcal{O}(k^2 n \log^2 n)$.

5 Numerical results

In this section we demonstrate the efficiency of our method on several examples. All algorithms were implemented and executed in MATLAB version R2016b on a dual Intel Core i7-5600U 2.60GHz CPU, 256 KByte of level 2 cache and 12 GByte of RAM, using a single core. Because all algorithms are rich in calls to BLAS and LAPACK routines, we believe that the use of MATLAB (instead of a compiled language) does not severely limit the predictive value of the reported timings.

The following algorithms have been compared:

CholQR The Cholesky-based QR decomposition for HODLR matrices explained in Section 2.2.

CholQR2 CholQR followed by one step of the reorthogonalization procedure explained in Section 2.2.

hQR Algorithm 2, our newly proposed algorithm.

MATLAB Call to the MATLAB function `qr`, which in turn calls the corresponding LAPACK routine for computing the QR decomposition of a general dense matrix.

Among the methods discussed in Section 2, we have decided to focus on CholQR and CholQR2, primarily because they are relatively straightforward to implement. A comparison of CholQR and CholQR2 with the other methods from Section 2 can be found in [4].

If not stated otherwise, when working with HODLR matrices we have chosen the minimal block size $n_{\min} = 250$ and the truncation tolerance $\epsilon = 10^{-10}$.

To assess accuracy, we have measured the numerical orthogonality of $Q = I - YTY^T$ and the residual of the computed QR decomposition:

$$e_{\text{orth}} = \|Q^TQ - I\|_2, \quad e_{\text{acc}} = \|QR - A\|_2. \tag{22}$$

Example 1 (**Performance for random HODLR matrices).** We first investigate the performance of our method for HODLR matrices of varying size constructed as follows. The diagonal blocks are random dense matrices and each off-diagonal block is a rank-one matrix chosen as the outer product of two random vectors. From Figure 2, one observes that the computational time of the hQR algorithm nicely matches the $\mathcal{O}(n \log^2 n)$ reference line, the complexity claimed in Section 4.1. Compared to the much simpler and as we shall see, less accurate CholQR method, our new method is approximately only two times slower, while CholQR2 is slower than hQR for $n \geq 10\,000$. Note that for $n \geq 256\,000$, the Cholesky factorization of A^TA fails to complete due to lack of (numerical) positive definiteness and, in turn, both CholQR and CholQR2 return with an error.

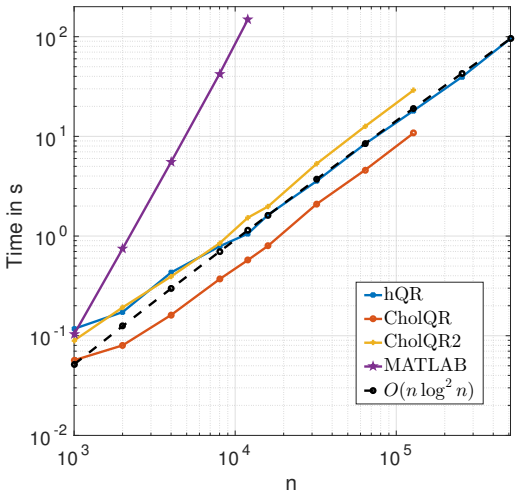


Figure 2: Example 1: Execution time vs. n for computing QR decomposition of randomly generated $n \times n$ HODLR matrices.

Table 2 provides insights into the observed accuracy for values of n for which (22) can be evaluated conveniently. As n increases, the condition number of A increases. Our method is robust to this increase and produces numerical orthogonality and a residual norm on the level of the truncation error. In contrast, the accuracy of CholQR clearly deteriorates as the condition number increases and the refinement performed by the more expensive CholQR2 cannot fully make up for this.

Table 2: Example 1. Numerical orthogonality e_{orth} and residual norm e_{acc}, see (22), of different methods for computing QR decomposition of randomly generated $n \times n$ HODLR matrices.

n	$\kappa_2(A)$	$e_{\text{orth}}^{\text{hQR}}$	$e_{\text{orth}}^{\text{CholQR}}$	$e_{\text{orth}}^{\text{CholQR2}}$	$e_{\text{acc}}^{\text{hQR}}$	$e_{\text{acc}}^{\text{CholQR}}$	$e_{\text{acc}}^{\text{CholQR2}}$
1 000	$8.7 \cdot 10^4$	$7.5 \cdot 10^{-15}$	$2.2 \cdot 10^{-9}$	$1.2 \cdot 10^{-10}$	$8.3 \cdot 10^{-13}$	$3.8 \cdot 10^{-13}$	$7.1 \cdot 10^{-11}$
2 000	$1.4 \cdot 10^5$	$1.4 \cdot 10^{-14}$	$3.4 \cdot 10^{-8}$	$2.4 \cdot 10^{-9}$	$4.4 \cdot 10^{-12}$	$1.1 \cdot 10^{-12}$	$1.3 \cdot 10^{-9}$
4 000	$1.2 \cdot 10^6$	$1.6 \cdot 10^{-13}$	$8.4 \cdot 10^{-7}$	$1.5 \cdot 10^{-8}$	$1.5 \cdot 10^{-11}$	$7.6 \cdot 10^{-12}$	$1.6 \cdot 10^{-8}$
8 000	$3.1 \cdot 10^7$	$1.9 \cdot 10^{-12}$	$5.1 \cdot 10^{-6}$	$5.7 \cdot 10^{-8}$	$1.9 \cdot 10^{-10}$	$9.1 \cdot 10^{-10}$	$3.5 \cdot 10^{-8}$
12 000	$1.5 \cdot 10^8$	$1.8 \cdot 10^{-12}$	$2.2 \cdot 10^{-4}$	$8.2 \cdot 10^{-7}$	$1.9 \cdot 10^{-10}$	$1.4 \cdot 10^{-10}$	$2.7 \cdot 10^{-7}$

Table 3 aims at clarifying whether the representation of Q in terms of its compact WY representation constitutes a disadvantage in terms of HODLR ranks. It turns out that the contrary is true; the maximal off-diagonal ranks of Y and T are significantly smaller than those of Q. Note, however, that does not translate into reduced memory consumption for the matrix sizes under consideration, because the larger off-diagonal ranks only occur in a few (smaller) off-diagonal blocks in Q.

Table 3: Example 1. Maximal off-diagonal ranks for the factors Y, T, R and $Q = I - YTY^T$ from the QR decomposition computed by hQR applied to randomly generated $n \times n$ HODLR matrices. Memory for storing Y and T as well as Q relative to memory for storing A in the HODLR format.

	Maximal ranks				Memory	
n	Y	T	Q	R	Y and T	Q
1 000	2	2	6	4	1.99	1
8 000	5	5	18	10	2	1.2
64 000	8	8	30	15	2.1	1.5
256 000	10	10	38	17	2.17	1.7

We also note that the maximal off-diagonal ranks and relative memory for Y, T, R grow slowly, possibly logarithmically, as n increases.

Example 2 (Accuracy for Cauchy matrices). In this example, we consider Cauchy matrices of size $n = 2000$, for which the entry (i, j) is given by $(x_i - y_j)^{-1}$ for $x, y \in \mathbb{R}^n$. The vectors x and y are chosen as 2000 equally spaced points from intervals I_x and I_y, respectively, additionally perturbed by $\pm 2 \cdot 10^{-2}$ with the sign chosen at random. We have used the following configurations:

- matrix A_1: intervals $I_x = [-1.25, 998.25]$ and $I_y = [-0.7, 998.9]$;
- matrix A_2: intervals $I_x = [-1.25, 998.25]$ and $I_y = [-0.45, 999.15]$;
- matrix A_3: intervals $I_x = [-1.25, 998.25]$ and $I_y = [-0.15, 999.45]$.

All three matrices are invertible but their condition numbers are different. For each i, the HODLR approximation of A_i has maximal off-diagonal rank 20. Table 4 summarizes the obtained results, which show that our method consistently attains an accuracy up to the level of truncation error. Once again, CholQR and CholQR2 fail to complete the computation for A_3, the most ill-conditioned matrix. In contrast to Example 1, the maximal off-diagonal ranks for Y and T do not grow; they are bounded by 20. The maximal off-diagonal rank for R is 32.

Table 4: Example 2. Numerical orthogonality e_{orth} and residual norm e_{acc}, see (22), of different methods for computing QR decomposition of Cauchy matrices with varying condition number.

	$\kappa_2(A_i)$	$e_{\text{orth}}^{\text{hQR}}$	$e_{\text{orth}}^{\text{CholQR}}$	$e_{\text{orth}}^{\text{CholQR2}}$	$e_{\text{acc}}^{\text{hQR}}$	$e_{\text{acc}}^{\text{CholQR}}$	$e_{\text{acc}}^{\text{CholQR2}}$
A_1	$4.8 \cdot 10^{5}$	$5.7 \cdot 10^{-11}$	$2.6 \cdot 10^{-5}$	$2.8 \cdot 10^{-11}$	$1.1 \cdot 10^{-8}$	$2.9 \cdot 10^{-9}$	$1.9 \cdot 10^{-7}$
A_2	$1.3 \cdot 10^{8}$	$3.6 \cdot 10^{-10}$	$1.3 \cdot 10^{-1}$	$3.4 \cdot 10^{-9}$	$2.3 \cdot 10^{-9}$	$6.8 \cdot 10^{-10}$	$4.6 \cdot 10^{-9}$
A_3	$2.9 \cdot 10^{12}$	$1.5 \cdot 10^{-10}$	-	-	$2.2 \cdot 10^{-9}$	-	-

Example 3 (Accuracy and orthogonality versus truncation tolerance). As our final example, we investigate the influence of the truncation tolerance on the accuracy attained by our method. For this purpose, we consider the matrix A_3 from Example 2. The truncation tolerance ϵ is varied from 10^{-2} to 10^{-20}, and the obtained results are compared with MATLAB's built-in function `qr`. Figure 3 demonstrates that the errors decrease nearly proportional with ϵ until they stagnate around $\epsilon = 10^{-14}$, below which roundoff error appears to dominate.

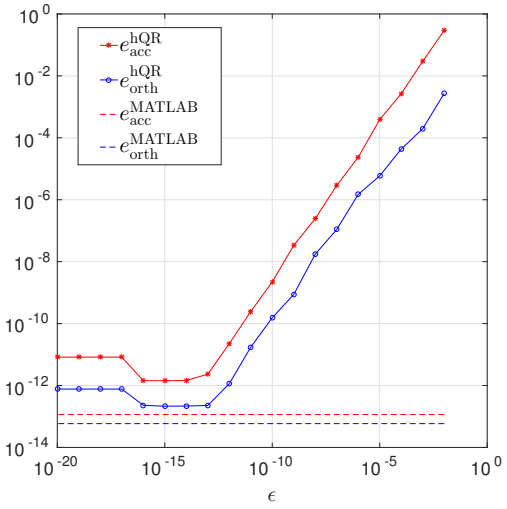


Figure 3: Example 3. Numerical orthogonality e_{orth} and residual norm e_{acc}, see (22), of hQR applied to a Cauchy matrix with condition number $\approx 10^{12}$ vs. truncation tolerance ϵ.

6 Extension to rectangular HODLR matrices

In this section, we sketch the extension of Algorithm 2 to rectangular HODLR matrices. For such matrices, one allows the diagonal blocks in the recursive partitioning (1) to be rectangular. In

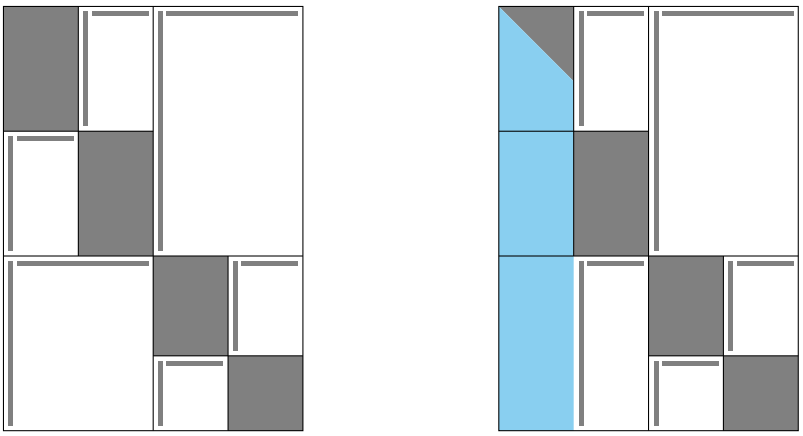
turn, the definition of a rectangular HODLR matrix $A \in \mathbb{R}^{m\times n}$ depends on two integer partitions

$$m = m_1 + m_2 + \cdots m_{2^\ell}, \quad n = n_1 + n_2 + \cdots n_{2^\ell},$$

corresponding to the sizes $m_j \times n_j$, $j = 1, \ldots, 2^\ell$, of the diagonal blocks on the lowest level of the recursion. In the following, we assume that

$$m_j \geq n_j, \quad j = 1, \ldots, 2^\ell.$$

See Figure 4 (a) for an illustration.



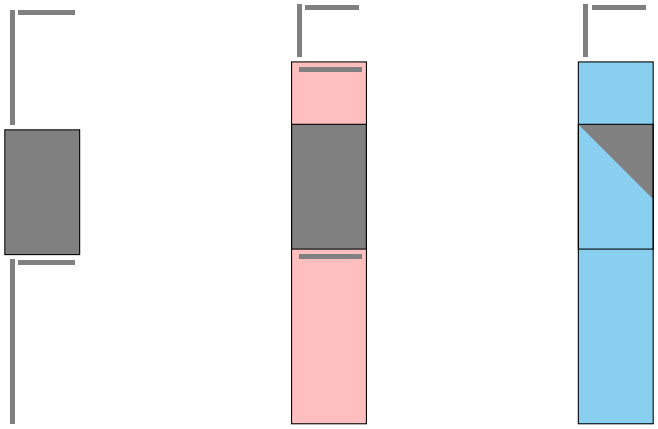
(a) *(b)*

Figure 4: (a) Rectangular HODLR matrix of level $\ell = 2$. (b) Structure after reducing the first block column. Zero parts of the matrix are colored blue.

An application appears in our work [20] on a fast spectral divide-and-conquer method, which requires the QR decomposition of an $m \times n$ matrix that results from selecting $n \approx m/2$ columns of an $m \times m$ HODLR matrix.

We now consider the application of Algorithm 2 to a rectangular HODLR matrix. This algorithm starts with reducing the first block column to upper triangular form. This first step is coherent with the structure, see Figure 4 (b), and no significant modification of Algorithm 2 is necessary. However, the same cannot be said about the subsequent steps. The transformation of the second block column (or, more precisely, its unreduced part) to upper triangular form would mix dense with low-rank blocks and in turn destroy the HODLR format. To avoid this effect, we reduce the second block column to *permuted triangular form*, such that the reduced triangular matrix replaces the dense diagonal block and all other parts become zero. This process is illustrated in Figure 5: First the low-rank blocks in the unreduced part (the part highlighted in Figure 5 (b)) are compressed. Then an orthogonal transformation is performed such that the nonzero rows are reduced to a triangular matrix situated on top of the dense block; see Figure 5 (c). In practice, this is effected by an appropriate permutation of the rows, followed by a QR decomposition and the inverse permutation. The rows of the factor Y in the compact WY representation of this transformation are permuted accordingly and, in turn, Y inherits the structure from the second block column.

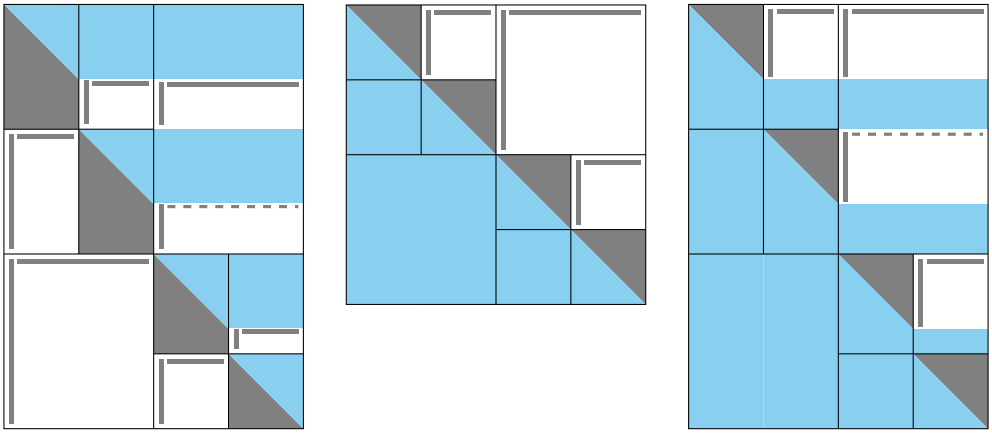
The described process is applied to each block column on the lowest level of the recursion: A permuted QR decomposition is performed such that the reduced $n_j \times n_j$ triangular matrix is situated on top of the dense diagonal block. On higher levels of the recursion, Algorithm 2 extends



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

Figure 5: Reduction of the second block column of a rectangular HODLR matrix of level $\ell = 2$. Zero parts are colored blue.

with relatively minor modifications. This modified algorithm results in a QR decomposition $A = QR$, $Q = I - YTY^T$, where Y, R are *permuted* lower trapezoidal/upper triangular matrices that inherit the HODLR format of A. The matrix T is an $n \times n$ upper triangular HODLR matrix; see Figure 6 for an illustration. Note that, in particular, R is not triangular, but it can be easily permuted to triangular form, if needed.



(a) Factor Y. *(b) Factor T.* *(c) Factor R.*

Figure 6: Illustration of factors Y, T and R of a (permuted) QR decomposition of a rectangular HODLR matrix of level $\ell = 2$. Dashed lines denote right low-rank factors that are shared with the off-diagonal blocks above the considered block.

We have collected preliminary numerical evidence that the described modified algorithm is effective at computing permuted QR decompositions of rectangular HODLR matrices. For this purpose, we have applied a dense version of the algorithm and compressed the obtained factors Y, T, R afterwards, in accordance with the format shown in Figure 6. The parameters guiding the HODLR format are identical to the default parameters in Section 5: $n_{\min} = 250$ and $\epsilon = 10^{-10}$.

Example 4 (**Performance for an invariant subspace basis).** This example illustrates the use of our algorithm for orthonormalizing a set of vectors in an application from [20]. For this purpose, we consider a tridiagonal symmetric matrix $T \in \mathbb{R}^{m\times m}$ with $m = 8\,000$, chosen such that the eigenvalues are uniformly distributed in $[-1, -10^{-1}] \cup [10^{-1}, 1]$. It turns out that the spectral projector $\Pi_{<0}$ associated with the negative eigenvalues of T can be well approximated in the HODLR format; the numerical ranks of the off-diagonal blocks are bounded by 17. We applied the method proposed in [20, Section 4.1] (with threshold parameter $\delta = 0.35$) to select a well-conditioned set of $n \approx m/2$ columns of $\Pi_{<0}$, which will be denoted by $\Pi_{<0}(:, C) \in \mathbb{R}^{m\times n}$ with the column indices C.

We applied the described modification of Algorithm 2 to orthonormalize the rectangular HODLR matrix $A = \Pi_{<0}(:, C) \in \mathbb{R}^{m\times n}$; an operation needed in [20]. The accuracy we obtained is at the level of truncation tolerance: $e_{\text{orth}} = 5.8 \cdot 10^{-12}$ and $e_{\text{acc}} = 8.9 \cdot 10^{-11}$. The algorithm is also efficient in terms of memory; see Table 5. In particular, the off-diagonal ranks of the factors Y, T and R do not grow compared to A. In this example, and in contrast to the square examples reported in Section 5, the memory is reduced when storing Y and T instead of Q.

Table 5: Examples 4 and 5. Maximal off-diagonal ranks for the factors Y, T, R and $Q = I - YTY^T$. Memory for storing Y and T as well as Q, and R relative to memory for storing A in the HODLR format.

	Maximal ranks				Memory		
	Y	T	Q	R	Y and T	Q	R
Example 4	14	14	23	11	1.5	2.1	0.87
Example 5	8	12	12	8	1.6	2.2	1.1

Example 5 (**Performance for a random rectangular HODLR matrix).** In analogy to Example 1 we generated a random $8\,000 \times 4\,000$ HODLR matrix A with off-diagonal ranks 1. We obtained $e_{\text{orth}} = 2.8 \cdot 10^{-13}$, $e_{\text{acc}} = 1.4 \cdot 10^{-11}$, and the off-diagonal ranks and the memory requirements shown in Table 5.

7 Conclusion

We have presented the hQR method, a novel, fast and accurate method for computing the QR decomposition of a HODLR matrix. Our numerical experiments indicate that hQR is the method of choice, unless one wants to sacrifice accuracy for a relatively small gain in computational time. It remains to be seen whether the developments of this work extend to the broader class of hierarchical matrices.

References

[1] S. Ambikasaran, D. Foreman-Mackey, L. Greengard, D. W. Hogg, and M. O'Neil. Fast direct methods for Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):252–265, 2016.

[2] A. H. Aminfar, S. Ambikasaran, and E. Darve. A fast block low-rank dense solver with applications to finite-element matrices. *J. Comput. Phys.*, 304:170–188, 2016.

[3] M. Bebendorf. *Hierarchical matrices*, volume 63 of *Lecture Notes in Computational Science and Engineering*. Springer-Verlag, Berlin, 2008.

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

[5] D. A. Bini, S. Massei, and L. Robol. On the decay of the off-diagonal singular values in cyclic reduction. *Linear Algebra Appl.*, 519:27–53, 2017.

[6] Y. Eidelman, I. Gohberg, and I. Haimovici. *Separable type representations of matrices and fast algorithms. Vol. 1*, volume 234 of *Operator Theory: Advances and Applications*. Birkhäuser/Springer, Basel, 2014. Basics. Completion problems. Multiplication and inversion algorithms.

[7] E. Elmroth and F. Gustavson. Applying recursion to serial and parallel QR factorization leads to better performance. *IBM J. Research & Development*, 44(4):605–624, 2000.

[8] G. H. Golub and C. F. Van Loan. *Matrix computations*. Johns Hopkins University Press, Baltimore, MD, fourth edition, 2013.

[9] W. Hackbusch. *Hierarchical matrices: algorithms and analysis*, volume 49 of *Springer Series in Computational Mathematics*. Springer, Heidelberg, 2015.

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

[11] R. A. Horn and C. R. Johnson. *Matrix analysis*. Cambridge University Press, Cambridge, second edition, 2013.

[12] D. Kressner and A. Šušnjara. Fast computation of spectral projectors of banded batrices. *SIAM J. Matrix Anal. Appl.*, 38(3):984–1009, 2017.

[13] M. Lintner. *Lösung der 2D Wellengleichung mittels hierarchischer Matrizen*. Doctoral thesis, TU München, 2002.

[14] M. Lintner. The eigenvalue problem for the 2D Laplacian in $\mathcal{H}$-matrix arithmetic and application to the heat and wave equation. *Computing*, 72(3-4):293–323, 2004.

[15] Y. Nakatsukasa, Z. Bai, and F. Gygi. Optimizing Halley's iteration for computing the matrix polar decomposition. *SIAM J. Matrix Anal. Appl.*, 31(5):2700–2720, 2010.

[16] Y. Nakatsukasa and N. J. Higham. Stable and efficient spectral divide and conquer algorithms for the symmetric eigenvalue decomposition and the SVD. *SIAM J. Sci. Comput.*, 35(3):A1325–A1349, 2013.

[17] R. Schreiber and C. F. Van Loan. A storage-efficient WY representation for products of Householder transformations. *SIAM J. Sci. Statist. Comput.*, 10(1):53–57, 1989.

[18] A. Stathopoulos and K. Wu. A block orthogonalization procedure with constant synchronization requirements. *SIAM J. Sci. Comput.*, 23(6):2165–2182, 2002.

[19] G. W. Stewart. *Introduction to matrix computations*. Academic Press [A subsidiary of Harcourt Brace Jovanovich, Publishers], New York-London, 1973. Computer Science and Applied Mathematics.

[20] A. Šušnjara and D. Kressner. A fast spectral divide-and-conquer method for banded matrices. arXiv:1801.04175 [math.NA], 2018.

[21] R. Vandebril, M. Van Barel, and N. Mastronardi. *Matrix computations and semiseparable matrices. Vol. 1.* Johns Hopkins University Press, Baltimore, MD, 2008.

[22] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li. Fast algorithms for hierarchically semiseparable matrices. *Numer. Linear Algebra Appl.*, 17(6):953–976, 2010.

[23] Y. Yamamoto, Y. Nakatsukasa, Y. Yanagisawa, and T. Fukaya. Roundoff error analysis of the CholeskyQR2 algorithm. *Electron. Trans. Numer. Anal.*, 44:306–326, 2015.

Recent publications:

INSTITUTE of MATHEMATICS
MATHICSE Group

Ecole Polytechnique Fédérale (EPFL)

CH-1015 Lausanne

2018

02.2018 ELEONORA ARNONE, LAURA AZZIMONTI, FABIO NOBILE, LAURA M. SANGALLI:
Modelling spatially dependent functional data via regression with differential regularization

03.2018 NICCOLO DAL SANTO, SIMONE DEPARIS, ANDREA MANZONI, ALFIO QUARTERONI:
Mutli space reduced basis preconditioners for parametrized Stokes equations

04.2018 MATTHIEU MARTIN, SEBASTIAN KRUMSCHEID, FABIO NOBILE:
Mutli space reduced basis preconditioners for parametrized Stokes equations
Analysis of stochastic gradient methods for PDE-Constrained optimal control problems with uncertain parameters

05.2018 VALENTINE REY, SEBASTIAN KRUMSCHEID, FABIO NOBILE:
Quantifying uncertainties in contact mechanics of rough surfaces using the Multilevel Monte Carlo method

06.2018 MICHELE PISARONI, FABIO NOBILE, PENELOPE LEYLAND:
A continuation-multilevel Monte Carlo evolutionary algorithm for robust aerodynamic shape design

07.2018 STEFANO MASSEI, MARIAROSA MAZZA, LEONARDO ROBOL:
Fast solvers for 2D fractional differential equations using rank structured matrices

08.2018 DARIO A. BINI, STEFANO MASSEI, LEONARDO ROBOL:
Quasi-Toeplitz matrix arithmetic : a Matlab toolbox

09.2018 ASSYR ABDULLE, ANDREA DI BLASIO:
A Bayesian numerical homogenization method for elliptic multiscale inverse problems

10.2018 ASSYR ABDULLE, GIACOMO ROSILHO DE SOUZA:
A local discontinuous Galerkin gradient discretization method for linear and quasilinear elliptic equations

11.2018 ASSYR ABDULLE, GIACOMO GAREGNANI:
Random time step probabilistic methods for uncertainty quantification in chaotic and geometric numerical integration

12.2018 DANIEL KRESSNER, ANA SUSNJARA:
Fast QR decomposition of HODLR matrices

***