Received: 16 September 2022 | Revised: 28 January 2023 | Accepted: 3 February 2023

DOI: 10.1002/nla.2497

RESEARCH ARTICLE

WILEY

A block Cholesky-LU-based QR factorization for rectangular matrices

Sabine Le Borne

Hamburg University of Technology, Institute of Mathematics, Hamburg, Germany

Correspondence
Sabine Le Borne, Hamburg University of Technology, Institute of Mathematics, Am Schwarzenberg-Campus 3, 21073 Hamburg, Germany.
Email: leborne@tuhh.de

Abstract

The Householder method provides a stable algorithm to compute the full QR factorization of a general matrix. The standard version of the algorithm uses a sequence of orthogonal reflections to transform the matrix into upper triangular form column by column. In order to exploit (level 3 BLAS or structured matrix) computational advantages for block-partitioned algorithms, we develop a block algorithm for the QR factorization. It is based on a well-known block version of the Householder method which recursively divides a matrix columnwise into two smaller matrices. However, instead of continuing the recursion down to single matrix columns, we introduce a novel way to compute the QR factors in implicit Householder representation for a larger block of several matrix columns, that is, we start the recursion at a block level instead of a single column. Numerical experiments illustrate to what extent the novel approach trades some of the stability of Householder's method for the computational efficiency of block methods.

KEYWORDS

block QR factorization, Householder method

1 | INTRODUCTION

Let $A \in \mathbb{R}^{m\times n}$, and let the orthogonal matrix $Q \in \mathbb{R}^{m\times m}$ and upper triangular matrix $R \in \mathbb{R}^{m\times n}$ denote the factors of the full QR factorization of A,

$$A = QR.$$

In the case $m > n$ with $\text{rank}(A) = n$, that is, a tall/skinny A with full column rank, the factors Q and R can be split to obtain a reduced (thin) QR factorization

$$A = QR = \begin{pmatrix} Y & Z \end{pmatrix} \begin{pmatrix} R_\square \\ 0 \end{pmatrix} = YR_\square, \tag{1}$$

where $Y \in \mathbb{R}^{m\times n}$ spans the range of A and $Z \in \mathbb{R}^{m\times(m-n)}$ forms a basis of the nullspace of A^T.

There exist many algorithms in the literature to compute a QR factorization of a given matrix, see for example, References 1-3 for some recent contributions (and the references therein). While some of the algorithms may be applied to any type of matrix, others try exploit special properties of the matrix such as a particular sparsity structure or a hierarchical low-rank structure. In the case of a rectangular matrix, some algorithms may only compute the reduced form while others compute the full QR factors, and some provide Q explicitly in matrix form while others provide it only implicitly, allowing its multiplication by a vector or matrix.

In this article, we are concerned with the computation of a full QR factorization for a full rank matrix $A \in \mathbb{R}^{m\times n}$ with $m \geq n$ where the orthogonal factor Q is given in factored form[1,4-7]

$$Q = I - VSV^T$$

with a unit lower triangular matrix $V \in \mathbb{R}^{m\times n}$ and an upper triangular matrix $S \in \mathbb{R}^{n\times n}$. Possible advantages of such a factored form of Q are that V and S together require less storage than Q if A is tall/skinny (assuming dense matrices) and may preserve the structure of A better than Q if A is structured.

If $A \in \mathbb{R}^{m\times 1}$ consists of only a single column, then a Householder reflection with a vector $u = (u_1, \ldots, u_m)^T$ of length $\|u\|_2 = \sqrt{2}$,

$$Q = I - uu^T = I - vsv^T \quad \text{with} \quad v := \frac{u}{u_1}, s := u_1^2, \tag{2}$$

provides such a factored form of $Q \in \mathbb{R}^{m\times m}$ directly. If $A \in \mathbb{R}^{m\times n}$ with $n > 1$, then there exists a recursive algorithm based on a columnwise division of $A = \begin{pmatrix} A_1 & A_2 \end{pmatrix}$ intotwo matrices A_1, A_2 with fewer columns than A. The recursion ends when matrices with only a single column are reached for which the QR factorization in the factored representation is given by a Householder reflection as shown in (2). We will review this recursive algorithm in Section 2.

While this recursive Householder factorization starts out as a block algorithm, it looses this character when the recursion continues down to single matrix columns. Hence we propose to combine it with a novel block QR algorithm for any matrices $A \in \mathbb{R}^{m\times \ell}$ with fewer than k columns (i.e., $\ell \leq k$) for some fixed parameter $k \in \mathbb{N}$, that is, we no longer go down to single columns. The new algorithm is not as stable as Householder reflections but it is a block algorithm which can benefit from BLAS-3 or structured matrix arithmetic. This novel algorithm and its combination with the recursive block Householder factorization constitutes our main result and is given in Section 3.

In Section 4, we show numerical results that illustrate the effect of stopping the recursion for $k \in \mathbb{N}$ and replacing it by the new block algorithm. Some conclusions follow in Section 5.

2 | REVIEW: A RECURSIVE BLOCK HOUSEHOLDER QR FACTORIZATION

The following recursive full Householder QR factorization has been introduced in Reference 8 and also used/mentioned in References 1 and 5.

The Householder QR factorization $A = QR = (I - VSV^T)R$ of a single column matrix $A \in \mathbb{R}^{m\times 1}$ is given by a Householder reflection (2). If A has more than one column, it is divided columnwise into two submatrices. The resulting blocks in the right hand side of

$$\begin{pmatrix} A_1 & A_2 \end{pmatrix} = \left(I - \begin{pmatrix} V_1 & V_2 \end{pmatrix} \begin{pmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{pmatrix} \begin{pmatrix} V_1 & V_2 \end{pmatrix}^T \right) \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \\ 0 & 0 \end{pmatrix}, \tag{3}$$

are computed by the Algorithm 1 (see also Figure 1 for an illustration of the respective blocks).

3 | MAIN RESULT: A BLOCK QR FACTORIZATION WITH FACTORED ORTHOGONAL FACTOR Q

We will follow the derivation in Reference 6 which provides a block computation of the nullspace basis Z in (1). In the following theorem, we extend this result to obtain the full orthogonal factor $Q = \begin{pmatrix} Y & Z \end{pmatrix} = I - VSV^T$ of a tall/skinny matrix $A \in \mathbb{R}^{m\times n}$, $m \geq n$, in factored form.

Algorithm 1. Recursive Householder-based QR factorization of $A \in \mathbb{R}^{m\times n}$ with orthogonal factor in the form $Q = I - VSV^T$, see (4) for notation of subblocks

Input: $A \in \mathbb{R}^{m\times n}$.
Output: $V, R \in \mathbb{R}^{m\times n}, S \in \mathbb{R}^{n\times n}$ representing the QR factors of A.
if $n = 1$ **then**
  Compute a Householder reflection $Q = I - vsv^T$ such that $R := Q^T A$ is a multiple of the first unit vector.
else
  Split A vertically into two blocks $A = \begin{pmatrix} A_1 & A_2 \end{pmatrix}$ of one or more columns.
  Compute (recursively) a QR factorization of A_1: $A_1 = \underbrace{(I - V_1 S_{11} V_1^T)}_{=:Q_1} \begin{pmatrix} R_{11} \\ 0 \end{pmatrix}$.
  Update and partition A_2: $Q_1^T A_2 =: \begin{pmatrix} R_{12} \\ \widetilde{A}_{22} \end{pmatrix}$.
  Compute (recursively) a QR factorization of $\widetilde{A}_{22}$: $\widetilde{A}_{22} = (I - \widetilde{V}_2 S_{22} \widetilde{V}_2^T) \begin{pmatrix} R_{22} \\ 0 \end{pmatrix}$.
  Compute V_2, S_{12} : $V_2 := \begin{pmatrix} 0 \\ \widetilde{V}_2 \end{pmatrix}$, $S_{12} = -S_{11} V_1^T V_2 S_{22}$.
end if
return V, S, R.

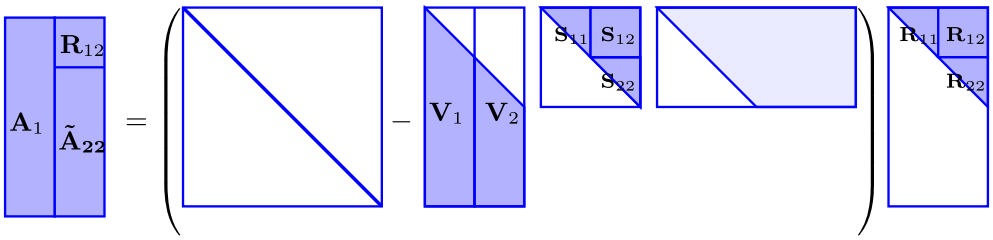

FIGURE 1 Illustration of the recursive Householder block QR factorization.

Theorem 1. *Let $A \in \mathbb{R}^{m\times n}$ with $m > n$, rank$(A) = n$, be subdivided rowwise into a square block $A_\square \in \mathbb{R}^{n\times n}$ and a remainder block $A_r \in \mathbb{R}^{(m-n)\times n}$ with a respective subdivision of its Householder QR factors $Q \in \mathbb{R}^{m\times m}$ and $R \in \mathbb{R}^{m\times n}$ with $R_\square \in \mathbb{R}^{n\times n}, Y \in \mathbb{R}^{m\times n}, Z \in \mathbb{R}^{m\times(m-n)}$ as in*

$$\begin{pmatrix} A_\square \\ A_r \end{pmatrix} = A = QR = \begin{pmatrix} Y & Z \end{pmatrix} \begin{pmatrix} R_\square \\ 0 \end{pmatrix}. \tag{4}$$

Let $L, U \in \mathbb{R}^{n\times n}$ denote the LU factors of $A_\square - R_\square$ (assuming they exist). Then, the orthogonal matrix $Q = \begin{pmatrix} Y & Z \end{pmatrix}$ has the two representations

$$Q = \begin{pmatrix} I_1 & \\ & I_2 \end{pmatrix} - \underbrace{\begin{pmatrix} L \\ A_r U^{-1} \end{pmatrix}}_{=:V} \underbrace{\left(-U R_\square^{-1} L^{-T}\right)}_{=:S} \begin{pmatrix} L^T & U^{-T} A_r^T \end{pmatrix}, \tag{5}$$

$$= \begin{pmatrix} I_1 & \\ & I_2 \end{pmatrix} - \underbrace{\begin{pmatrix} A_\square - R_\square \\ A_r \end{pmatrix}}_{=:\widetilde{V}} \underbrace{\left(-R_\square^{-1} L^{-T} U^{-T}\right)}_{=:\widetilde{S}} \begin{pmatrix} (A_\square - R_\square)^T & A_r^T \end{pmatrix} \tag{6}$$

with identity matrices $I_1 \in \mathbb{R}^{n\times n}, I_2 \in \mathbb{R}^{(m-n)\times(m-n)}$, unit lower triangular matrix $V \in \mathbb{R}^{m\times n}$ and upper triangular matrix $S \in \mathbb{R}^{n\times n}$.

Proof. Subdividing $Y = \begin{pmatrix} Y_\square \\ Y_r \end{pmatrix}$ according to the subdivision of A in (4), there hold $A_\square = Y_\square R_\square$, $A_r = Y_r R_\square$ and hence

$$Y_\square = A_\square R_\square^{-1},\ Y_r = A_r R_\square^{-1}.$$

The respective matrix block consisting of the first m columns in the right hand side of (5) reduces to

$$\begin{aligned}
&\begin{pmatrix} I_1 \\ \end{pmatrix} - \begin{pmatrix} L \\ A_r U^{-1} \end{pmatrix} \left(-U R_\square^{-1} L^{-T} \right) L^T \\
&= \begin{pmatrix} I_1 \\ \end{pmatrix} + \begin{pmatrix} LUR_\square^{-1} \\ A_r R_\square^{-1} \end{pmatrix} = \begin{pmatrix} I_1 + (A_\square - R_\square) R_\square^{-1} \\ A_r R_\square^{-1} \end{pmatrix} \\
&= \begin{pmatrix} A_\square R_\square^{-1} \\ A_r R_\square^{-1} \end{pmatrix} = Y.
\end{aligned}$$

The matrix V in (5) is a unit lower triangular matrix, and S is a product of upper triangular matrices and hence itself upper triangular.

The representation (5), in particular for the remaining $m - n$ columns, now follows since in a Householder QR factorization there holds $Q = I - VSV^T$ with upper triangular matrix S and unit lower triangular matrix V consisting columnwise of the Householder vectors that are scaled to have ones along the diagonal.[1,7]

The representation (6) follows directly from (5) in view of $\widetilde{V} = VU$ and $\widetilde{S} = U^{-1} S U^{-T}$. ∎

Remark 1. The representation (5) leads to triangular matrices V, S and hence less storage compared to $\widetilde{V}$, $\widetilde{S}$ in (6) (assuming dense matrices). However, (6) may be advantageous with respect to computational costs: $\widetilde{V}$ requires no additional computations whereas V requires the multiplication $A_r U^{-1}$, potentially destroying (blockwise low-rank) structure in A_r. Regarding the computations of S and $\widetilde{S}$, both require $R_\square^{-1} L^{-T}$. In (5), this is multiplied by U from the left, preserving the upper triangular matrix form. In (6), this is multiplied by U^{-T} from the right (performed as a triangular solve). Hence, depending on the (type of) matrix A, one variant may be preferable over the other from a computational cost aspect.

Remark 2. The block Cholesky-LU-based QR factorization in Algorithm 2 requires a Cholesky factorization of $A^T A$ in line 3. For (highly) ill-conditioned matrices A, this may cause stability problems or even a breakdown in the computation of the Cholesky factorization. One may replace the Cholesky factorization with one of the variants CholeskyQR2, LU-CholeskyQR or LU-CholeskyQR2 recently introduced in Reference 3 for the computation of a thin QR factorization. The algorithm LU-CholeskyQR first computes a pivoted LU factorization $PA = LU$ followed by a Cholesky factorization $L^T L = \hat{R}^T \hat{R}$ which is expected to be more stable than a Cholesky factorization of $A^T A$. The Cholesky factor R of $A^T A$ is then computed as $R = \hat{R} U$. The two variants CholeskyQR2 and LU-CholeskyQR2 are based on a subsequent reorthogonalization of the orthogonal factor (which in our case has first to be computed since we do not require it).

The key value/benefit of Theorem 1 lies in the fact that the QR factors can be computed using block algorithms for the required Cholesky and LU factorizations as well as the required matrix-matrix multiplications (triangular solves). The representations in Theorem 1 lead to Algorithm 2.

The decomposition (5) in Theorem 1 not only requires $A_\square - R_\square$ to be invertible but also to have an LU factorization. The decomposition (6) in Theorem 1, however, only needs $A_\square - R_\square$ to be invertible since $L^{-T} U^{-T} = (LU)^{-T} = (A_\square - R_\square)^{-T}$, that is, $L^{-T} U^{-T}$ can be replaced by the inverse of $(A_\square - R_\square)^T$. The following lemma provides a criterion for $A_\square - R_\square$ to be invertible.

Lemma 1. *Let $A \in \mathbb{R}^{m \times n}$, $m \geq n$. $A_\square - R_\square$ is invertible if the matrix Q in (4) is a rank-n modification of the identity matrix, that is, the product of n (and not fewer) Householder matrices. If $m = n$, that is, $A \in \mathbb{R}^{m \times m}$ is a square matrix, then $A_\square - R_\square$ is not invertible.*

Algorithm 2. Block Cholesky-LU-based QR factorization of $A = (A_\square^T, A_r^T)^T$ with orthogonal factor in the form $Q = I - VSV^T$, see (4) for notation of subblocks

1: **Input:** $A \in \mathbb{R}^{m\times n}$.
2: **Output:** $V \in \mathbb{R}^{m\times n}, S, R_\square \in \mathbb{R}^{n\times n}$ representing the QR factors of A.
3: Compute A^TA and its Cholesky factor $R_\square \in \mathbb{R}^{n\times n}$.
4: Compute $A_\square - R_\square$ and its LU factors $L, U \in \mathbb{R}^{n\times n}$.
5: Compute $S := -(UR_\square^{-1})L^{-T}$ (or $\widetilde{S} := -(R_\square^{-1}L^{-T})U^{-T}$).
6: Compute $V = \begin{pmatrix} L \\ A_rU^{-1} \end{pmatrix}$ (or $\widetilde{V} = \begin{pmatrix} A_\square - R_\square \\ A_r \end{pmatrix}$).
7: **return** $V, S, R_\square$ (or $\tilde{V}, \tilde{S}, R_\square$).

Algorithm 3. Recursive block QR factorization of $A \in \mathbb{R}^{m\times n}$ with orthogonal factor in the form $Q = I - VSV^T$ (3), user-specified input parameter $1 < k < n$.

1: **Input:** $A \in \mathbb{R}^{m\times n}$.
2: **Output:** $V, R \in \mathbb{R}^{m\times n}, S \in \mathbb{R}^{n\times n}$ representing the QR factors of A.
3: **if** $1 < n < k$ **then**
4: Perform Algorithm 2 to compute a QR factorization $A = (I - VSV^T)R$.
5: **else**
6: Perform Algorithm 1 to compute a QR factorization $A = (I - VSV^T)R$.
7: **end if**
8: **return** V, S, R.

Proof. Let Q be the product of n Householder matrices $Q_i = I - u_iu_i^T$ with $\|u_i\| = \sqrt{2}$ which produce zeros below the ith diagonal entry of $Q_{i-1} \ldots Q_1A$. By construction of Householder vectors, the leading $i-1$ entries of the Householder vectors $u_i \in \mathbb{R}^m$ are zero and the ith entry is nonzero. Hence u_i can be scaled such that its ith entry is one, leading to the representation $Q = I - VSV^T$ with lower unit triangular $V \in \mathbb{R}^{m\times n}$ and upper triangular matrix $S \in \mathbb{R}^{n\times n}$.[1,7] The upper square $(n \times n)$ part of V, which we denote by L, is a lower unit triangular matrix. Hence, considering only the leading n rows in $A = (I - VSV^T)R$, there holds $A_\square = \left(I_1 - LSL^T\right)R_\square$ and hence $A_\square - R_\square = -L(SL^T)$. Since S and L^T are upper triangular, $A_\square - R_\square$ has an LU factorization with $U = -SL^T$.

Now let $A \in \mathbb{R}^{m\times m}$ be a square matrix with QR factorization $A = QR$. Then

$$A_\square - R_\square = A - R = QR - R = (Q - I)R.$$

Since Q is a projection, it has an eigenvalue 1, hence $Q - I$ is singular and thus $A_\square - R_\square$ is singular as well. ∎

Remark 3. If $A_\square - R_\square$ does not have an LU factorization, one may apply a row permutation and try to compute the QR factorization $PA = QR$ by the proposed algorithm. The triangular factor $R_\square$ remains unchanged since it is the Cholesky factor of A^TA which is independent of row permutations of A. Hence, if there exists a permutation matrix P such that $\widetilde{A}_\square - R_\square$ has an LU factorization where $\widetilde{A}_\square$ denotes the leading block of PA, then the block QR algorithm may be used.

The motivation for Algorithm 2 is not to use it as a stand-alone algorithm but in combination with Algorithm 1, leading to Algorithm 3: If A has fewer than k columns for a user-defined parameter k, then the new block Algorithm 2 is used, otherwise one proceeds recursively as in the Householder-based Algorithm 1.

4 | NUMERICAL RESULTS

We have implemented Algorithm 3 in MATLAB and tested the impact of the parameter k which defines when to switch from the Householder recursion to the new block Cholesky-LU-based QR factorization. The shown results have used

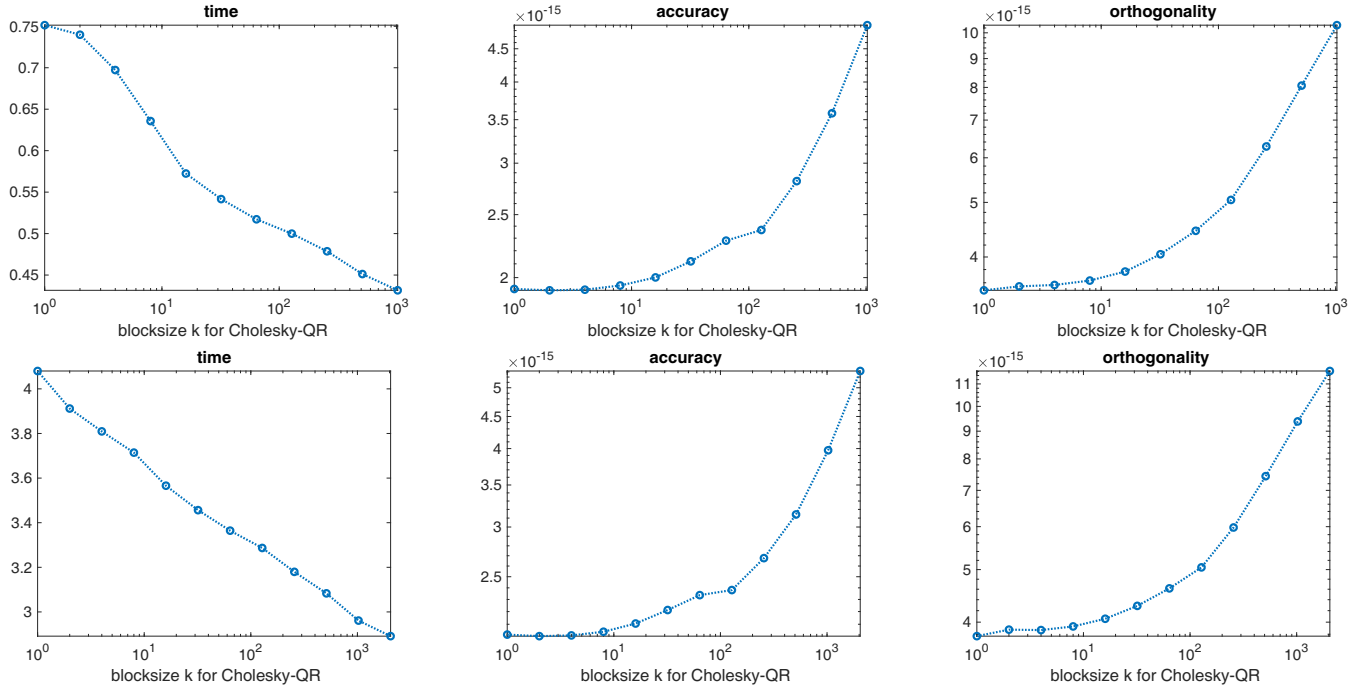

FIGURE 2 Results for a dense 2500×2500 matrix (top row) and a dense 5000×5000 matrix (bottom row), both filled with random entries.

the factorization in (6). Results using (5) were similar and are hence not shown. To facilitate the reproduction of our numerical results, we have included our MATLAB code in Appendix A.

In the following Figures 2 and 3, the x-axis refers to this parameter k, and the markers correspond to tests for $k = 2^\ell, \ell = 1, 2, \ldots$. We performed tests for dense square matrices of sizes 2500 and 5000 filled with random entries as well as for sparse matrices resulting from a five-point-stencil discretization of the 2D Laplacian on regular grids of sizes 2500 and 5041, resp. Even though the latter is a sparse matrix, we treated it as a dense one, that is, stored and computed with all its zero entries. The left plots in Figures 2 and 3 show the computational time (lowest out of 3 runs). As anticipated, the transition to the block version at the start of the recursion leads to increasing gains in computational time as the block size parameter k increases. The plots in the middle of Figures 2 and 3 show the (relative) accuracy of the factorization measured in the Frobenius norm, $\|A - QR\|_F / \|A\|_F$. As expected, this value increases as k increases. However, noting the logarithmic scale of the y-axis, the loss in accuracy appears to be moderate. The same applies to the (relative loss of) orthogonality $\|Q^T Q - I\|_F / \sqrt{n}$ shown in the right plots.

We have also performed numerical tests to compare different variants for the Cholesky factorization needed in line 3 of the block Cholesky-LU-based QR factorization in Algorithm 2. The Cholesky variants CholeskyQR2, LU-CholeskyQR and LU-CholeskyQR2 have briefly been introduced in Remark 2, for details we refer to Reference 3. Figure 4 shows the resulting relative accuracy and orthogonality of the computed QR factorization for the following four matrices:

Name	Rows	Cols	Condition number
Random	2500	2500	7990
2D Laplace	2500	2500	1053
Krylov	2500	625	$2.4e+16$
Hilbert	20	20	$1.5e+18$

A Hilbert matrix is given by $A = (\frac{1}{i+j-1})_{1 \le i,j \le n}$. For a random matrix H and a random vector r, the Krylov matrix A has been computed with columns $A(:, i) = H^i r / \|H^i r\|_2$, that is, its columns have been scaled to length 1 to avoid overflow.

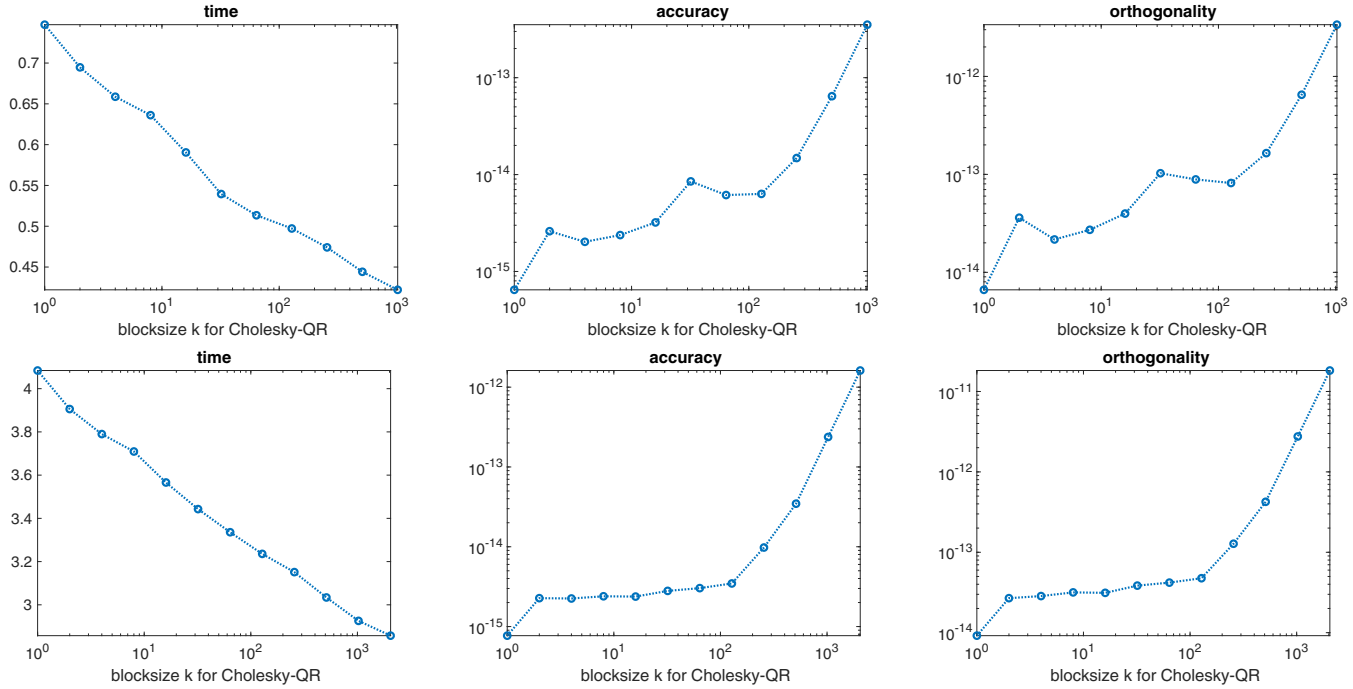

FIGURE 3 Results for a sparse 2500×2500 matrix (top row) and a sparse 5041×5041 matrix (bottom row), both representing a discretized 2D Laplace operator but treated as a dense matrix.

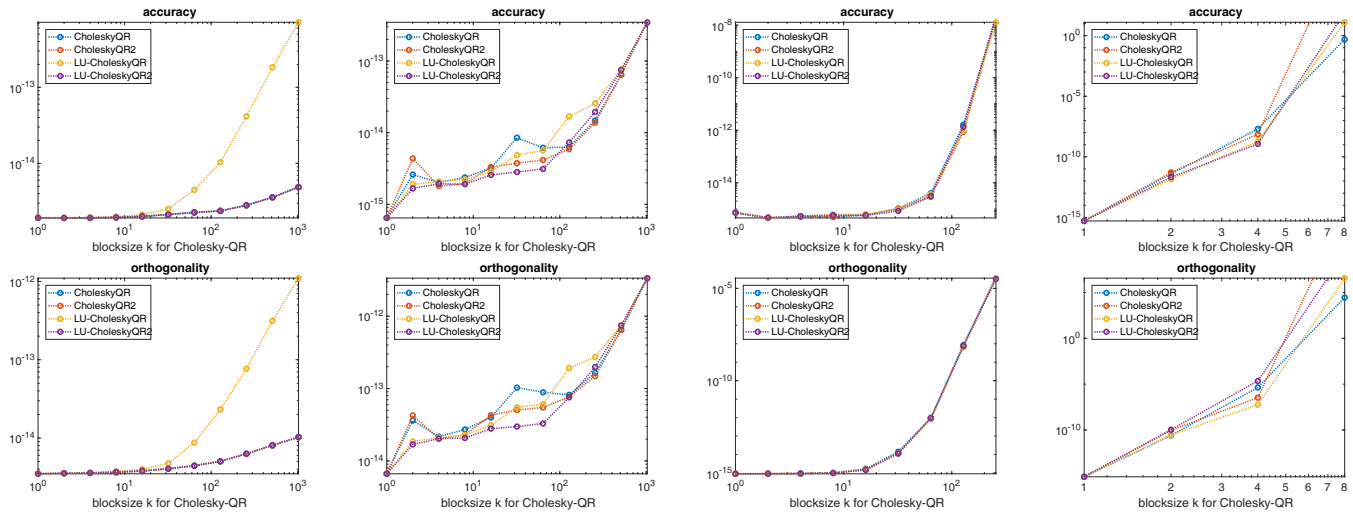

FIGURE 4 Relative accuracy (top) and orthogonality (bottom) for four test matrices (left to right): A 2500×2500 random matrix, a 2500×2500 sparse (Laplace) matrix, a 2500×625 Krylov matrix and a 20×20 Hilbert matrix.

Timings are not shown since the Cholesky factorization at the start of the recursion requires only a negligible part of the overall computational time, hence the timings hardly vary for the different variants.

For the random matrix, all variants except for LU-CholeskyQR perform comparable. The worse performance of LU-CholeskyQR can also be observed for different problem sizes (not shown here). For the sparse 2D Laplace matrix, there are some minor differences in accuracy and orthogonality but the general trend is the same for the four variants. The stabilized versions (ending with a "2") seem to perform slightly better. For the Krylov matrix, we would have to zoom in to see any differences. Compared to the random and Laplace matrix, however, we notice a faster decrease of accuracy and orthogonality due to the ill-conditioning of the Krylov matrix. The Hilbert matrix is highly ill-conditioned, and each increase of the blocksize k deteriorates the quality of the factorization significantly. For $k = 8$, the computed QR

factorization is already useless, for larger k it breaks down since the matrix A^TA whose Cholesky factorization is needed is no longer (numerically) positive definite.

5 | CONCLUSIONS

We have provided a novel approach to compute the Householder QR factors where Q is given in the form $Q = I - VSV^T$. While Algorithm 1 has already replaced BLAS-2 operations of the standard Householder QR factorization in favor of BLAS-3 operations, the recursion still goes down to a single column. While this may just slow down computations when working with dense matrices, it may cause severe difficulties when working with rank structured matrices where certain matrix blocks that are stored in a low rank format need to be split and later glued together again. The new Algorithm 2 can be performed in structured matrix arithmetic (e.g., hierarchical matrices) where structured Cholesky- and LU-factorizations are readily available.

The numerical results for dense matrices confirm the gains in computational time at the expense of some loss of accuracy and orthogonality compared to the recursive Householder-based Algorithm 1.

It remains to implement the new algorithm for structured matrices using their efficient matrix arithmetic and test the resulting computational and storage complexities.

ACKNOWLEDGMENT

Open Access funding enabled and organized by Projekt DEAL.

CONFLICT OF INTEREST STATEMENT

This study does not have any conflicts of interest to disclose.

DATA AVAILABILITY STATEMENT

Data sharing is not applicable to this article as no new data were created or analyzed in this study.

ORCID

Sabine Le Borne https://orcid.org/0000-0002-4399-4442

REFERENCES

1. Barlow J. Block modified Gram-Schmidt algorithms and their analysis. SIAM J Matrix Anal Appl. 2019;40:1257–90.
2. Carson E, Lund K, Rozloznik M, Thomas S. Block Gram-Schmidt algorithms and their stability properties. Linear Algebra Appl. 2022;638:150–95.
3. Terao T, Ozaki K, Ogita T. LU-Cholesky QR algorithms for thin QR decomposition. Parallel Comput. 2020;92:102571. https://doi.org/10.1016/j.parco.2019.102571
4. Golub G, Van Loan C. Matrix computations. 3rd. ed. London: John Hopkins; 1996.
5. Kressner D, Susnjara A. Fast QR decomposition of HODLR matrices, 2018. https://arxiv.org/abs/1809.10585
6. Le Borne S. Block computation and representation of a sparse nullspace basis of a rectangular matrix. Linear Algebra Appl. 2008;428:2455–67.
7. Schreiber R, Van Loan C. A storage-efficient WY representation for products of householder transformations. SIAM J Sci Stat Comput. 1989;10:53–7.
8. Elmroth E, Gustavson F. Applying recursion to serial and parallel QR factorization leads to better performance. IBM J Res Dev. 2000;44:605–24.

How to cite this article: Le Borne S. A block Cholesky-LU-based QR factorization for rectangular matrices. Numer Linear Algebra Appl. 2023;e2497. https://doi.org/10.1002/nla.2497

APPENDIX A. MATLAB CODE OF ALGORITHM 4

```
function [V, S, R] = householderQR(A, k, cholesky_type)
% choose variant for computing Cholesky factor of A'*A
% 1 : Cholesky
% 2 : Cholesky2
% 3 : LU-Cholesky
% 4 : LU-Cholesky2

[rows, cols] = size(A);
V = zeros(rows, cols);
R = zeros(rows, cols);
S = zeros(cols, cols);

if cols == 1
  % Householder based QR for a single column
  sigma = norm(A, 'fro');
  if A(1,1) > 0
    sigma = -sigma;
  end
  V = A;
  V(1,1) = V(1,1) - sigma;
  normV = norm(V, 'fro');
  v1 = V(1,1) / normV;
  V = 1 / V(1,1) * V;
  S = 2*v1^2;
  R(1,1) = sigma;
else
  mid = ceil(cols / 2);

  if (cols > k) || (rows == cols)
    % recursive Householder QR
    [V(:,1:mid), S(1:mid,1:mid), R(:,1:mid)] = ...
        householderQR(A(:,1:mid), k, cholesky_type);
    Aaux = S(1:mid,1:mid)' * (V(:,1:mid)' * A(:,mid+1:cols));
    R(1:mid,mid+1:cols) = ...
        A(1:mid,mid+1:cols) - V(1:mid,1:mid) * Aaux;
    A22tilde = ...
        A(mid+1:rows,mid+1:cols) - V(mid+1:rows,1:mid) * Aaux;
    [V(mid+1:rows,mid+1:cols), S(mid+1:cols,mid+1:cols), ...
        R(mid+1:rows,mid+1:cols)] = ...
        householderQR(A22tilde, k, cholesky_type);
    S(1:mid,mid+1:cols) = - S(1:mid,1:mid) * (V(:,1:mid)' ...
        * V(:,mid+1:cols)) * S(mid+1:cols,mid+1:cols);
  else
    % direct (nonrecursive) Cholesky-based QR
    if (cholesky_type == 1)      % Cholesky
      R(1:cols,1:cols) = chol(A'*A);
    elseif (cholesky_type == 2) % Cholesky2
      S = chol(A'*A);
      V = A / S;
      U = chol(V'*V);
      R(1:cols,1:cols) = U*S;
```

```
    elseif (cholesky_type == 3) % LU-Cholesky
      [L,U,P]=lu(A);
      S = chol(L'*L);
      R(1:cols,1:cols) = S*U;
    else                        % LU-Cholesky2
      [L,U,P]=lu(A);
      R(1:cols,1:cols) = chol(L'*L);
      S = R(1:cols,1:cols)*U;
      V = A / S;
      U = chol(V'*V);
      R(1:cols,1:cols) = U*S;
    end
    V = A;
    V(1:cols,:) = V(1:cols,:) - R(1:cols,:);
    S = - R(1:cols,1:cols) \ inv(V(1:cols,:))';
    %[L,U]=lu(V(1:cols,:));
  %S = - R(1:cols,1:cols) \ (U\ (L\ eye(cols)))';
  end
end
```