# High-Performance Up-and-Downdating via Householder-like Transformations

Robert A. van de Geijn
Field G. Van Zee
Department of Computer Science
The University of Texas at Austin
Austin, TX 78712
rvdg,field@cs.utexas.edu

Draft
February 1, 2010

**Abstract**

We present high-performance algorithms for up-and-downdating a Cholesky factor or QR factorization. The method uses Householder-like transformations, sometimes called hyperbolic Householder transformations, that are accumulated so that most computation can be cast in terms of high-performance matrix-matrix operations. The resulting algorithms can then be used as building blocks for an algorithm-by-blocks that allows computation to be conveniently scheduled to multithreaded architectures like multicore processors. Performance is shown to be similar to that achieved by a blocked QR factorization via Householder transformations.

## 1 Introduction

Consider the Linear Least-Squares problem that, given a matrix $A \in \mathbb{C}^{m\times n}$ with linearly independent columns and $y \in \mathbb{C}^m$, computes $x \in \mathbb{C}^n$ that minimizes $\|Ax - y\|_2$. This problem is typically solved via one of two methods:

**Method of Normal Equations:** Solve $A^H Ax = A^H y$ by computing the Cholesky factor of $A^H A$, upper triangular matrix $R$, followed by forward and backward substitution to solve $R^H Rx = A^H y$.

**QR factorization (via Householder transformations):** Compute the $QR$ factorization $A = QR$ where $Q$ is an orthogonal $m \times n$ matix and $R$ is an upper triangular $n \times n$ matrix. Solve $Rx = Q^H y$.

In this paper, we concern ourselves with the following prototypical scenario: Let rows of the appended system $\left( \begin{array}{c|c} A & y \end{array} \right)$ represent observations that have been taken, for example, over time. These observations can be partitioned into three groups: $\left( \begin{array}{c|c} A & y \end{array} \right) = \left( \begin{array}{c|c} B & b \\ \hline\hline C & c \\ \hline\hline D & d \end{array} \right)$ where the Cholesky factor corresponding to $\left( \begin{array}{c} B \\ \hline\hline D \end{array} \right)$ has already been computed: $B^H B + D^H D = R^H R$. Now, the rows of $D$ represent old data that we would like to remove while the rows of $C$ represent new data that we would like to add to the Linear Least-Squares problem. Thus, we would like to compute the Cholesky factor corresponding to $\left( \begin{array}{c} B \\ \hline\hline C \end{array} \right)$ leveraging the already computed $R$. The right-hand side has to be updated correspondingly, which is discussed in Section 6.

In [14] hyperbolic Householder transformations are reviewed for this problem and analyzed both from an algorithmic and numerical stability point of view. In that paper, references to the literature can also be found. The present paper builds on the insights in that paper and combines it with insights from other papers [1, 13, 7, 16, 8, 17, 15] that focus on aggregrating multiple Householder-like transformations into a block transformation. The contribution of the present paper is a practical high-performance algorithm for up- and/or downdating that can be implemented as a library routine using the level-3 BLAS [5].

The remainder of the paper is structured as follows. In Section 2 we discuss a family of Householder-like transformations and how to accumulate them into a block transformation. Updating and downdating are discussed separately in Sections 3 and 4, respectively, and then combined in Section 5 in which a blocked algorithm is also given. How to use an up- and/or downdated system to solve the new Linear Least-Squares problem is discussed in Section 6. A brief overview of the algorithm-by-blocks concept is given in Section 7. Performance is reported in Section 8 and concluding remarks can be found in the final section.

## 2 A Family of Householder Transformations

In the following discussion, we will let $\Sigma \in \mathbb{R}^{n\times n}$ with $\Sigma = \mathrm{diag}(1, \pm 1, \cdots, \pm 1)$ so that $\Sigma\Sigma = I$. Such a matrix is referred to as a *signature matrix*. We make the choice that the first diagonal element equals to one so as to simplify our discussion. Then, by design, $\Sigma e_0 = e_0$, where $e_0$ is the first column of the identity matrix.

**Theorem 1** *Let $w \in \mathbb{C}^n$ and $\tau = w^H \Sigma w/2 \neq 0$. Then $(I - \frac{1}{\tau}\Sigma w w^H)\Sigma(I - \frac{1}{\tau}\Sigma w w^H)^H = \Sigma$.*

**Proof:** Let $w \in \mathbb{C}^n$ and $\tau = w^H \Sigma w/2 \neq 0$. Then

$$(I - \frac{1}{\tau}\Sigma w w^H)\Sigma(I - \frac{1}{\tau}\Sigma w w^H)^H \quad = \quad \Sigma - 2\frac{1}{\tau}\Sigma w w^H \Sigma + \frac{2}{\tau}\Sigma w w^H \Sigma = \Sigma.$$

**endofproof**

When $\Sigma = I$ the $(I - \frac{1}{\tau}\Sigma w w^H)$ in the above theorem is the traditional Householder transformation or reflector. If $\Sigma = \left(\begin{array}{c||c} I & 0 \\ \hline 0 & -I \end{array}\right)$ it is referred to as a hyperbolic Householder Transformation.

**Theorem 2** *Let $x \in \mathbb{C}^n$, $\chi_0 = e_0^T x$ be its first element, $|\lambda|^2 = \sqrt{x^H \Sigma x}$ chosen so that $\bar{\lambda}\chi_0 (= \bar{\chi}_0 \lambda)$ is real, $w = x + \lambda e_0$, and $\tau = \frac{w^H \Sigma w}{2} \neq 0$. Then $(I - \frac{1}{\tau}\Sigma w w^H)^H x = -\lambda e_0$.*

**Proof:** Under the assumptions of the theorem

$$\frac{2w^H \Sigma x}{w^H \Sigma w} = \frac{2(x + \lambda e_0)^H \Sigma x}{(x + \lambda e_0)^H \Sigma (x + \lambda e_0)} = \frac{2(x^H \Sigma x + \bar{\lambda}\chi_0)}{x^H \Sigma x + 2\bar{\lambda}\chi_0 + |\lambda|^2} = 1$$

and

$$(I - \frac{1}{\tau}\Sigma w w^H)^H x = x - \frac{1}{\tau} w w^H \Sigma x = x - \frac{2w^H \Sigma x}{w^H \Sigma w}(x + \lambda e_0) = -\lambda e_0.$$

**endofproof**

**Corollary 3** *Under the assumptions of Thm. 2, if $\chi_0$ is real, then $\lambda = \pm\sqrt{x^H \Sigma x}$.*

The Cholesky factor $R$ that is being updated and/or downdated often has real diagonal elements, and vector $w$ can be normalized by dividing by a nonzero scalar, in which case the following steps provide a robust way of computing $w$ and $\lambda$ so that $w$ has a unit first element:

- $\lambda := \mathrm{sign}(\chi_0)\sqrt{x^H \Sigma x}$. $w := x + \lambda e_0$. (Note: the choice of the sign means that $\lambda$ and $\chi_0$ have the same sign, thus avoiding catastrophic cancellation that can lead to unnecessary numerical inaccuracy).

- If $\omega_0$ equals zero, then $w = e_0$ else $w := w/\omega_0$. (Here $\omega_0 = e_0^T w$ equals the first element of $w$. This step normalizes $w$ so that it has a unit first element.)

**Definition 4** *Let $x \in \mathbb{C}^n$ be such that $x^H \Sigma x \neq 0$. We define the function*

$$[\tilde{\chi}_0, w_1, \tau] := \text{GeneralHouse}\,(\Sigma, \chi_0, x_1)$$

*so that* $\left(I - \frac{1}{\tau}\Sigma w w^H\right)^H x = \tilde{\chi}_0 e_0$, *where* $x = \left(\frac{\chi_0}{x_1}\right)$, $w = \left(\frac{1}{w_1}\right)$, *and* $\tau = \frac{w^H \Sigma w}{2}$.

**Theorem 5** *Let the matrix $W_{k-1} \in \mathbb{C}^{n\times k}$ have linearly independent columns. Partition $W_{k-1}$ by columns as*

$$W_{k-1} = \left( \begin{array}{c|c|c|c} w_0 & w_1 & \cdots & w_{k-1} \end{array} \right)$$

*and let $\tau_i \neq 0$, $0 \leq i < k$. Then for $0 \leq j < k$ there exists a $j \times j$ nonsingular upper triangular matrix $T_j$ such that*

$$\left(I - \frac{1}{\tau_0}\Sigma w_0 w_0^H\right)\left(I - \frac{1}{\tau_1}\Sigma w_1 w_1^H\right)\cdots\left(I - \frac{1}{\tau_{j-1}}\Sigma w_{j-1} w_{j-1}^H\right) = \left(I - \Sigma W_{j-1} T_{j-1}^{-1} W_{j-1}^H\right)$$

*The matrices $T_j$ is given by the recurrence $T_0 = \tau_0$ and* $T_j = \left(\begin{array}{c|c} T_{j-1} & W_{j-1}^H \Sigma w_j \\ \hline 0 & \tau_j \end{array}\right)$ *for $1 \leq j < k$.*

**Proof:** Proof by induction on $j$.
**Base case.** $j = 0$: Trivially true.
**Inductive step.** Induction Hypothesis (I.H.): Assume that

$$\left(I - \frac{1}{\tau_0}\Sigma w_0 w_0^H\right)\cdots\left(I - \frac{1}{\tau_1}\Sigma w_1 w_1^H\right)\left(I - \frac{1}{\tau_{j-1}}\Sigma w_{j-1} w_{j-1}^H\right) = \left(I - \Sigma W_{j-1} T_{j-1}^{-1} W_{j-1}^H\right)$$

We need to show that $\left(I - \frac{1}{\tau_0}\Sigma w_0 w_0^H\right)\cdots\left(I - \frac{1}{\tau_{j-1}}\Sigma w_{j-1} w_{j-1}^H\right)\left(I - \frac{1}{\tau_j}\Sigma w_j w_j^H\right) = \left(I - \Sigma W_j T_j^{-1} W_j^H\right)$:

$$\begin{aligned}
\left(I - \frac{1}{\tau_0}\Sigma w_0 w_0^H\right)\cdots\left(I - \frac{1}{\tau_j}\Sigma w_j w_j^H\right) &= \left(I - \Sigma W_{j-1} T_{j-1}^{-1} W_{j-1}^H\right)\left(I - \frac{\Sigma w_j w_j^H}{\tau_j}\right) \\
&= I - \Sigma \left( \begin{array}{c|c} W_{j-1} & w_j \end{array} \right) \left(\begin{array}{c|c} T_{j-1}^{-1} & -T_{j-1}^{-1} W_{j-1}^H \Sigma w_j / \tau_j \\ \hline 0 & 1/\tau_j \end{array}\right) \left( \begin{array}{c|c} W_{j-1} & w_j \end{array} \right)^H \\
&= I - \Sigma \left( \begin{array}{c|c} W_{j-1} & w_j \end{array} \right) \left(\begin{array}{c|c} T_{j-1} & W_{j-1}^H \Sigma w_j \\ \hline 0 & \tau_j \end{array}\right)^{-1} \left( \begin{array}{c|c} W_{j-1} & w_j \end{array} \right)^H = I - \Sigma W_j T_j^{-1} W_j^H.
\end{aligned}$$

**By the Principle of Mathematical Induction** the desired result holds. **endofproof**

**Theorem 6** *Let $W \in \mathbb{C}^{n\times k}$ be a matrix with linearly independent columns such that $W^H \Sigma W$ has nonzero diagonal elements. Then there exists a unique nonsingular upper triangular matrix with real diagonal elements $T \in \mathbb{C}^{k\times k}$ such that $(I - \Sigma W T^{-1} W^H)\Sigma(I - \Sigma W T^{-1} W^H)^H = \Sigma$. This matrix $T$ satisfies $T + T^H = W^H \Sigma W$ so that $T = \text{striu}(W^H \Sigma W) + \frac{1}{2}\text{diag}(W^H \Sigma W)$.*

**Proof:** Theorem 5 provides a proof of existence. (The $w_i$ and $\tau_i$'s in that theorem equal the columns of $W$ and diagonal elements of $W^T \Sigma W$, respectively.) Now,

$$\Sigma = (I - \Sigma W T^{-1} W^H)\Sigma(I - \Sigma W T^{-1} W^H)^H = \Sigma - \Sigma W T^{-1} W^H \Sigma - \Sigma W T^{-H} W^H \Sigma + \Sigma W T^{-1} W^H \Sigma W T^{-H} W^H \Sigma$$

so that

$$0 = \Sigma(W T^{-1} W^H + W T^{-H} W^H - W T^{-1} W^H \Sigma W T^{-H} W^H)\Sigma.$$

Thus,

$$0 = W^H \Sigma 0 \Sigma W = W^H \Sigma\Sigma(W T^{-1} W^H + W T^{-H} W^H - W T^{-1} W^H \Sigma W T^{-H} W^H)\Sigma\Sigma W$$

or, equivalently, (since $W^H \Sigma\Sigma W = W^H W$ is nonsingular)

$$0 = T^{-1} + T^{-H} - T^{-1} W^H \Sigma W T^{-H}$$

from which we conclude that $T^H + T = W^H \Sigma W$.

Now, if $T$ is upper triangular and has real valued diagonal elements, then $T = \text{striu}(W^H \Sigma W) + \frac{1}{2}\text{diag}(W^H \Sigma W)$. **endofproof**

# 3 Updating

Let us consider $A = \left( \begin{array}{c} B \\ \hline\hline C \end{array} \right)$ with $A^H A = B^H B + C^H C = \tilde{R}^H \tilde{R}$, where $\tilde{R}$ is the Cholesky factor of $A^H A$. Here we will assume that both $A$ and $B$ have linearly independent columns. The question is whether, if we know that the Cholesky factor of $B^H B$ is $R$, we can cheaply compute the Cholesky factor of $A^H A$. This is known as the updating problem. We know that $\tilde{R}^H \tilde{R} = A^H A = B^H B + C^H C = R^H R + C^H C$. We also know that if $B = QR$ is a QR factorization of $B$ and $\left( \begin{array}{c} R \\ \hline\hline C \end{array} \right) = \hat{Q}\hat{R}$, then

$$A = \left( \begin{array}{c} B \\ \hline\hline C \end{array} \right) = \left( \begin{array}{c} QR \\ \hline\hline C \end{array} \right) = \left( \begin{array}{c||c} Q & 0 \\ \hline\hline 0 & I \end{array} \right) \left( \begin{array}{c} R \\ \hline\hline C \end{array} \right) = \left( \begin{array}{c||c} Q & 0 \\ \hline\hline 0 & I \end{array} \right) \hat{Q}\hat{R} = \check{Q}\hat{R}$$

so that $A = \check{Q}\hat{R}$ is a QR factorization of $A$. Because of the uniqueness of the QR factorization (modulo signs), $\hat{R} = \tilde{R}$. This then provides us with the desired Cholesky factor. We conclude that to compute $\tilde{R}$ it suffices to compute the QR factorization of $\left( \begin{array}{c} R \\ \hline\hline C \end{array} \right)$.

A stable way for computing the QR factorization of $\left( \begin{array}{c} R \\ \hline\hline C \end{array} \right)$ relies on Householder transformations: given matrix $\left( \begin{array}{c} R \\ \hline\hline C \end{array} \right)$ where $R$ is an $n \times n$ upper triangular matrix and $C \in \mathbb{C}^{m_C \times n}$, we would like to compute $\{H_0, \cdots, H_{n-1}\}$ so that

$$H_{n-1}^H \cdots H_0^H \left( \begin{array}{c} R \\ \hline\hline D \end{array} \right) = \left( \begin{array}{c} \tilde{R} \\ \hline\hline 0 \end{array} \right) \quad \text{and} \quad H_j^H = H_j = I - \frac{1}{\tau_j} u_j u_j^H \text{ with } \tau_j = u_j^H u_j/2.$$

We recognize this as the special case of the Generalized Householder Transformation where $\Sigma = I$, in other words, the classical Householder Transformation.

**Definition 7** *Let $x = \left( \begin{array}{c} \chi_0 \\ \hline x_1 \end{array} \right) \in \mathbb{C}^n$ with $\chi_0 \in \mathbb{R}$ be such that $\|x\|_2 \neq 0$. We define the function*

$$[\tilde{\chi}_0, u_1, \tau] := \text{House}(\chi_0, x_1)$$

*so that* $\left( \left( \begin{array}{c|c} 1 & 0 \\ \hline 0 & I \end{array} \right) - \frac{1}{\tau} \left( \begin{array}{c} 1 \\ \hline u_1 \end{array} \right) \left( \begin{array}{c} 1 \\ \hline u_1 \end{array} \right)^H \right) \left( \begin{array}{c} \chi_0 \\ \hline x_1 \end{array} \right) = \left( \begin{array}{c} \hat{\chi}_0 \\ \hline 0 \end{array} \right)$*, where* $\tau = \frac{1 + u_1^H u_1}{2}$.

An algorithm that, given the Cholesky factor $R$ of $A^H A$, computes the Cholesky factor of $A^H A + C^H C$ is now given in Figure 1. Here is the basic idea: Assume that the computation has progressed so that the matrices contain

$$\left( \begin{array}{c} R \\ \hline\hline C \end{array} \right) = \left( \begin{array}{c|c|c} R_{00} & r_{01} & R_{02} \\ \hline 0 & \rho & r_{12}^T \\ \hline 0 & 0 & R_{22} \\ \hline\hline 0 & c_1 & C_2 \end{array} \right).$$

In the current step a Householder transformation is computed and applied so that

$$\left[ \left( \begin{array}{c|c|c||c} I & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 \\ \hline 0 & 0 & I & 0 \\ \hline\hline 0 & 0 & 0 & I \end{array} \right) - \frac{1}{\tau} \left( \begin{array}{c} 0 \\ \hline 1 \\ \hline 0 \\ \hline\hline u_1 \end{array} \right) \left( \begin{array}{c} 0 \\ \hline 1 \\ \hline 0 \\ \hline\hline u_1 \end{array} \right)^H \right] \left( \begin{array}{c|c|c} R_{00} & r_{01} & R_{02} \\ \hline 0 & \rho & r_{12}^T \\ \hline 0 & 0 & R_{22} \\ \hline\hline 0 & c_1 & C_2 \end{array} \right) = \left( \begin{array}{c|c|c} R_{00} & r_{01} & R_{02} \\ \hline 0 & \tilde{\rho} & \tilde{r}_{12}^T \\ \hline 0 & 0 & R_{22} \\ \hline\hline 0 & 0 & \tilde{C}_2 \end{array} \right).$$

The vector $u_1$ overwrites the vector $c_1$ that it annihilates. The function that updates $r_{12}^T$ and $C_2$ given $\tau$ and $u_1$ is given by

$$\left(\tilde{r}_{12}^T, \tilde{C}_2\right) := \text{ApplyHouse}\left(\tau_1, u_1, r_{12}^T, C_2\right).$$

# 4 Downdating

Now, let us consider the alternative problem where $A = \left(\begin{array}{c} B \\ \hline\hline D \end{array}\right)$ with $A^H A = B^H B + D^H D = R^H R$, where $D \in \mathbb{C}^{m_D \times n}$ and $R \in \mathbb{C}^{n \times n}$ is the Cholesky factor of $A^H A$. The new question becomes how to compute the Cholesky factor of $B^H B$ from $R$ and $D$. This is known as the downdating problem. Let us call the desired Cholesky factor $\tilde{R}$. We know that

$$\tilde{R}^H \tilde{R} = B^H B = R^H R - D^H D = \left(\begin{array}{c} R \\ \hline\hline D \end{array}\right)^H \left(\begin{array}{c||c} I_n & 0 \\ \hline\hline 0 & -I_{m_D} \end{array}\right) \left(\begin{array}{c} R \\ \hline\hline D \end{array}\right)$$

In the remainder of this section, we will define

$$\Sigma_{n,m} = \left(\begin{array}{c||c} I_n & 0 \\ \hline\hline 0 & -I_m \end{array}\right).$$

The goal is going to be to compute a sequence of transformations, $\{S_0, S_1, \cdots, \mathbb{S}_{n-1}\}$ such that

$$S_j \Sigma_{n,m_D} S_j^H = \Sigma_{n,m_D}$$

and

$$\begin{aligned} B^H B &= \left(\begin{array}{c} R \\ \hline\hline D \end{array}\right)^H \left(\begin{array}{c||c} I_n & 0 \\ \hline\hline 0 & -I_{m_D} \end{array}\right) \left(\begin{array}{c} R \\ \hline\hline D \end{array}\right) = \left(\begin{array}{c} R \\ \hline\hline D \end{array}\right)^H S_0 \cdots S_{n-1} \Sigma_{n,m_D} S_{n-1}^H \cdots S_0^H \left(\begin{array}{c} R \\ \hline\hline D \end{array}\right) \\ &= \left(\begin{array}{c} \tilde{R} \\ \hline\hline 0 \end{array}\right)^H \Sigma_{n,m_D} \left(\begin{array}{c} \tilde{R} \\ \hline\hline 0 \end{array}\right) = \tilde{R}^H \tilde{R} \end{aligned}$$

In other words, given matrix $\left(\begin{array}{c} R \\ \hline\hline D \end{array}\right)$ where $R$ is an $n \times n$ upper triangular matrix, we would like to compute $\{S_0, \cdots, S_{n-1}\}$ so that $S_{n-1}^H \cdots S_0^H \left(\begin{array}{c} R \\ \hline\hline D \end{array}\right) = \left(\begin{array}{c} \tilde{R} \\ \hline\hline 0 \end{array}\right)$ and $S_j \Sigma_{n,m_D} S_j = \Sigma_{n,m_D}^H$.

**Definition 8** *Let* $x = \left(\begin{array}{c} \chi_0 \\ \hline x_1 \end{array}\right) \in \mathbb{C}^n$ *with* $\chi_0 \in \mathbb{R}$ *be such that* $x^H \Sigma_{1,n-1} x = |\chi_0|^2 - x_1^H x_1 \neq 0$. *We define the function*

$$[\tilde{\chi}_0, v_1, \tau] := \text{HHouse}\left(\chi_0, x_1\right)$$

*so that* $\left(\left(\begin{array}{c|c} 1 & 0 \\ \hline 0 & I \end{array}\right) - \frac{1}{\tau}\left(\begin{array}{c|c} 1 & 0 \\ \hline 0 & -I \end{array}\right)\left(\begin{array}{c} 1 \\ \hline v_1 \end{array}\right)\left(\begin{array}{c} 1 \\ \hline v_1 \end{array}\right)^H\right)\left(\begin{array}{c} \chi_0 \\ \hline x_1 \end{array}\right) = \left(\begin{array}{c} \hat{\chi}_0 \\ \hline 0 \end{array}\right)$, *where* $\tau = \frac{1 - v_1^H v_1}{2}$.

We recognize this as the special case of the Generalized Householder Transformation where $\Sigma = \Sigma_{1,n-1}$. This special case is referred to as a hyperbolic Householder Transformation in the literature.

Given the function HHouse an algorithm that, given the Cholesky factor $R$ of $A^H A$, computes the Cholesky factor of $A^H A - D^H D$ is now given in Figure 2. Here is the basic idea: Assume that the computation has progressed so that the matrices contain

$$\left(\begin{array}{c} R \\ \hline\hline D \end{array}\right) = \left(\begin{array}{c|c|c} R_{00} & r_{01} & R_{02} \\ \hline 0 & \rho & r_{12}^T \\ \hline 0 & 0 & R_{22} \\ \hline\hline 0 & d_1 & D_2 \end{array}\right).$$

**Algorithm:** $[R, C, t] := \textsc{UpDate\_unb}(R, C, t)$

**Partition** $R \rightarrow \left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array}\right)$, $C \rightarrow \left(\begin{array}{c|c} C_L & C_R \end{array}\right)$, $t \rightarrow \left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right)$

**where** $R_{TL}$ is $0 \times 0$, $C_L$ has 0 columns, $t_T$ has 0 elements

**while** $m(R_{TL}) < m(R)$ **do**

**Repartition**

$$\left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} R_{00} & r_{01} & R_{02} \\ \hline 0 & \rho_{11} & r_{12}^T \\ \hline 0 & 0 & R_{22} \end{array}\right), \left(\begin{array}{c|c} C_L & C_R \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} C_0 & c_1 & C_2 \end{array}\right), \left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right) \rightarrow \left(\begin{array}{c} t_0 \\ \hline \tau_1 \\ \hline t_2 \end{array}\right)$$

**where** $\rho_{11}$ and $\tau_1$ are scalars, and $c_1$ has 1 column

---

$[\rho_{11}, c_1, \tau_1] := \text{House}(\rho_{11}, c_1)$

$\left(\begin{array}{c} r_{12}^T \\ \hline C_2 \end{array}\right) := \text{ApplyHouse}\left(\tau_1, c_1, r_{12}^T, C_2\right)$

---

**Continue with**

$$\left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} R_{00} & r_{01} & R_{02} \\ \hline 0 & \rho_{11} & r_{12}^T \\ \hline 0 & 0 & R_{22} \end{array}\right), \left(\begin{array}{c|c} C_L & C_R \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} C_0 & c_1 & C_2 \end{array}\right), \left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right) \leftarrow \left(\begin{array}{c} t_0 \\ \hline \tau_1 \\ \hline t_2 \end{array}\right)$$

**endwhile**

Figure 1: Unblocked algorithm for updating.

**Algorithm:** $[R, D, t] := \textsc{DownDate\_unb}(R, D, t)$

**Partition** $R \rightarrow \left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array}\right)$, $D \rightarrow \left(\begin{array}{c|c} D_L & D_R \end{array}\right)$, $t \rightarrow \left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right)$

**where** $R_{TL}$ is $0 \times 0$, $D_L$ has 0 columns, $t_T$ has 0 elements

**while** $m(R_{TL}) < m(R)$ **do**

**Repartition**

$$\left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} R_{00} & r_{01} & R_{02} \\ \hline 0 & \rho_{11} & r_{12}^T \\ \hline 0 & 0 & R_{22} \end{array}\right), \left(\begin{array}{c|c} D_L & D_R \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} D_0 & d_1 & D_2 \end{array}\right), \left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right) \rightarrow \left(\begin{array}{c} t_0 \\ \hline \tau_1 \\ \hline t_2 \end{array}\right)$$

**where** $\rho_{11}$ and $\tau_1$ are scalars, and $d_1$ has 1 column

---

$[\rho_{11}, d_1, \tau_1] := \text{HHouse}(\rho_{11}, d_1)$

$\left(\begin{array}{c} r_{12}^T \\ D_2 \end{array}\right) := \text{ApplyHHouse}\left(\tau, d_1, r_{12}^T, D_2\right)$

---

**Continue with**

$$\left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} R_{00} & r_{01} & R_{02} \\ \hline 0 & \rho_{11} & r_{12}^T \\ \hline 0 & 0 & R_{22} \end{array}\right), \left(\begin{array}{c|c} D_L & D_R \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} D_0 & d_1 & D_2 \end{array}\right), \left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right) \leftarrow \left(\begin{array}{c} t_0 \\ \hline \tau_1 \\ \hline t_2 \end{array}\right)$$

**endwhile**

Figure 2: Unblocked algorithm for downdating.

In the current step a hyperbolic Householder Transformation is computed and applied so that

$$\left( \left( \begin{array}{c|c|c||c} I & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 \\ \hline 0 & 0 & I & 0 \\ \hline\hline 0 & 0 & 0 & I \end{array} \right) - \frac{1}{\tau} \left( \begin{array}{c|c|c||c} I & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 \\ \hline 0 & 0 & I & 0 \\ \hline\hline 0 & 0 & 0 & -I \end{array} \right) \left( \begin{array}{c} 0 \\ \hline 1 \\ \hline 0 \\ \hline\hline v_1 \end{array} \right) \left( \begin{array}{c} 0 \\ \hline 1 \\ \hline 0 \\ \hline\hline v_1 \end{array} \right)^H \right)^H \left( \begin{array}{c|c|c} R_{00} & r_{01} & R_{02} \\ \hline 0 & \rho & r_{12}^T \\ \hline 0 & 0 & R_{22} \\ \hline\hline 0 & d_1 & D_2 \end{array} \right)$$

$$= \left( \begin{array}{c|c|c} R_{00} & r_{01} & R_{02} \\ \hline 0 & \tilde{\rho} & \tilde{r}_{12}^T \\ \hline 0 & 0 & R_{22} \\ \hline\hline 0 & 0 & \tilde{D}_2 \end{array} \right).$$

The vector $v_1$ overwrites the vector $d_1$ that it annihilates.

# 5 Up-and-Downdating

Finally, let us consider the general problem where $A = \left( \begin{array}{c} B \\ \hline\hline C \\ \hline\hline D \end{array} \right)$ with $A^H A = B^H B + C^H C + D^H D$, where $C \in \mathbb{C}^{m_C \times n}$, $D \in \mathbb{C}^{m_D \times n}$. Let $R \in \mathbb{C}^{n \times n}$ be the Cholesky factor of $B^H B + D^H D$. The final question becomes how to compute the Cholesky factor of $B^H B + C^H C$, $\tilde{R}$, from $R$, $C$, and $D$. Clearly, one can do so by first updating and then downdating, or vise versa. We will develop an algorithm that does so in one step rather than two. We will call this the *up-and-downdating* problem.

We know that

$$\tilde{R}^H \tilde{R} = B^H B + C^H C = R^H R + C^H C - D^H D = \left( \begin{array}{c} R \\ \hline\hline C \\ \hline\hline D \end{array} \right)^H \left( \begin{array}{c||c||c} I_n & 0 & 0 \\ \hline\hline 0 & I_{m_C} & 0 \\ \hline\hline 0 & 0 & -I_{m_D} \end{array} \right) \left( \begin{array}{c} R \\ \hline\hline C \\ \hline\hline D \end{array} \right).$$

In the remainder of this section, we will define

$$\Sigma_{n,m,k} = \left( \begin{array}{c||c||c} I_n & 0 & 0 \\ \hline\hline 0 & I_m & 0 \\ \hline\hline 0 & 0 & -I_k \end{array} \right).$$

The goal is going to be to compute a sequence of transformations, $\{G_0, G_1, \cdots, G_{n-1}\}$ such that

$$G_j \Sigma_{n,m_C,m_D} G_j^H = \Sigma_{n,m_C,m_D}$$

and

$$\begin{aligned} B^H B + C^H C &= \left( \begin{array}{c} R \\ \hline\hline C \\ \hline\hline D \end{array} \right)^H \left( \begin{array}{c||c||c} I_n & 0 & 0 \\ \hline\hline 0 & I_{m_C} & 0 \\ \hline\hline 0 & 0 & -I_{m_D} \end{array} \right) \left( \begin{array}{c} R \\ \hline\hline C \\ \hline\hline D \end{array} \right) \\ &= \left( \begin{array}{c} R \\ \hline\hline C \\ \hline\hline D \end{array} \right)^H G_0 \cdots G_{n-1} \left( \begin{array}{c||c||c} I_n & 0 & 0 \\ \hline\hline 0 & I_{m_C} & 0 \\ \hline\hline 0 & 0 & -I_{m_D} \end{array} \right) G_{n-1}^H \cdots G_0^H \left( \begin{array}{c} R \\ \hline\hline C \\ \hline\hline D \end{array} \right) \\ &= \left( \begin{array}{c} \tilde{R} \\ \hline\hline 0 \end{array} \right)^H \left( \begin{array}{c||c||c} I_n & 0 & 0 \\ \hline\hline 0 & I_{m_C} & 0 \\ \hline\hline 0 & 0 & -I_{m_D} \end{array} \right) \left( \begin{array}{c} \tilde{R} \\ \hline\hline 0 \end{array} \right) = \tilde{R}^H \tilde{R} \end{aligned}$$

In other words, given matrix $\left(\begin{array}{c} R \\ \hline\hline C \\ \hline\hline D \end{array}\right)$ where $R$ is an $n \times n$ upper triangular matrix, we would like to compute $\{G_0, \cdots, G_{n-1}\}$ so that

$$G_{n-1}^H \cdots G_0^H \left(\begin{array}{c} R \\ \hline\hline C \\ \hline\hline D \end{array}\right) = \left(\begin{array}{c} \tilde{R} \\ \hline\hline 0 \\ \hline\hline 0 \end{array}\right) \quad \text{and} \quad G_j^H \left(\begin{array}{c||c||c} I_n & 0 & 0 \\ \hline\hline 0 & I_{m_C} & 0 \\ \hline\hline 0 & 0 & -I_{m_D} \end{array}\right) G_j = \left(\begin{array}{c||c||c} I_n & 0 & 0 \\ \hline\hline 0 & I_{m_C} & 0 \\ \hline\hline 0 & 0 & -I_{m_D} \end{array}\right).$$

**Definition 9** *Let* $x = \left(\begin{array}{c} \chi_0 \\ \hline x_1 \\ \hline y_1 \end{array}\right)$ *with* $\chi_0 \in \mathbb{R}$, $x_1 \in \mathbb{C}^{m_C}$ *and* $y_1 \in \mathbb{C}^{m_D}$ *be such that* $x^H \Sigma_{1,m_C,m_D} x = |\chi_0|^2 + x_1^H x_1 - y_1^H y_1 \neq 0$
*We define the function*

$$[\tilde{\chi}_0, u_1, v_1, \tau] := \text{UDHouse}\,(\chi_0, x_1, y_1)$$

*so that* $\left[ \begin{pmatrix} 1 & 0 & 0 \\ 0 & I_{m_C} & 0 \\ 0 & 0 & I_{m_D} \end{pmatrix} - \frac{1}{\tau} \begin{pmatrix} 1 & 0 & 0 \\ 0 & I_{m_C} & 0 \\ 0 & 0 & -I_{m_D} \end{pmatrix} \begin{pmatrix} 1 \\ u_1 \\ v_1 \end{pmatrix} \begin{pmatrix} 1 \\ u_1 \\ v_1 \end{pmatrix}^H \right] \begin{pmatrix} \chi_0 \\ x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} \hat{\chi}_0 \\ 0 \\ 0 \end{pmatrix}$, *where* $\tau = \frac{1 + u_1^H u_1 - v_1^H v_1}{2}$.

We recognize this as the special case of the Generalized Householder Transformation where $\Sigma = \Sigma_{n,m_C,m_D}$.

Given the function UDHouse an algorithm that, given the Cholesky factor $R$ of $B^H B + D^H D$, computes the Cholesky factor of $B^H B + C^H C$ is now given in Figure 3. Here is the basic idea: Assume that the computation has progressed so that the matrices contain

$$\left(\begin{array}{c} R \\ \hline\hline D \end{array}\right) = \left(\begin{array}{c|c|c} R_{00} & r_{01} & R_{02} \\ \hline 0 & \rho_{11} & r_{12}^T \\ \hline 0 & 0 & R_{22} \\ \hline\hline 0 & c_1 & C_2 \\ \hline\hline 0 & d_1 & D_2 \end{array}\right).$$

In the current step an up-and-downdating Householder Transformation is computed and applied so that

$$\left[ \left(\begin{array}{c|c|c||c||c} I & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & I & 0 & 0 \\ \hline\hline 0 & 0 & 0 & I & 0 \\ \hline\hline 0 & 0 & 0 & 0 & I \end{array}\right) - \frac{1}{\tau} \left(\begin{array}{c|c|c||c||c} I & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & I & 0 & 0 \\ \hline\hline 0 & 0 & 0 & I & 0 \\ \hline\hline 0 & 0 & 0 & 0 & -I \end{array}\right) \left(\begin{array}{c} 0 \\ \hline 1 \\ \hline 0 \\ \hline\hline u_1 \\ \hline\hline v_1 \end{array}\right) \left(\begin{array}{c} 0 \\ \hline 1 \\ \hline 0 \\ \hline\hline u_1 \\ \hline\hline v_1 \end{array}\right)^H \right] \left(\begin{array}{c|c|c} R_{00} & r_{01} & R_{02} \\ \hline 0 & \rho_{11} & r_{12}^T \\ \hline 0 & 0 & R_{22} \\ \hline\hline 0 & c_1 & C_2 \\ \hline\hline 0 & d_1 & D_2 \end{array}\right)$$
$$= \left(\begin{array}{c|c|c} R_{00} & r_{01} & R_{02} \\ \hline 0 & \tilde{\rho}_{11} & \tilde{r}_{12}^T \\ \hline 0 & 0 & R_{22} \\ \hline\hline 0 & 0 & \tilde{C}_2 \\ \hline\hline 0 & 0 & \tilde{D}_2 \end{array}\right).$$

The vectors $u_1$ and $v_1$ overwrite the vectors $c_1$ and $d_1$, respectively.

A blocked algorithm for up-and-downdating is now given in Figure 4. The statement

$$[R_{12}, C_2, D_2] := \text{ApplyBlkUDHouse}\,(T_1, C_1, D_1, R_{12}, C_2, D_2)$$

performs the update

$$\left(\begin{array}{c} \tilde{R}_{12} \\ \hline\hline \tilde{C}_2 \\ \hline\hline \tilde{D}_2 \end{array}\right) := \left[ \left(\begin{array}{c||c||c} I & 0 & 0 \\ \hline\hline 0 & I & 0 \\ \hline\hline 0 & 0 & I \end{array}\right) - \left(\begin{array}{c||c||c} I & 0 & 0 \\ \hline\hline 0 & I & 0 \\ \hline\hline 0 & 0 & -I \end{array}\right) \left(\begin{array}{c} I \\ \hline\hline C_1 \\ \hline\hline D_1 \end{array}\right) T_1^{-T} \left(\begin{array}{c} I \\ \hline\hline C_1 \\ \hline\hline D_1 \end{array}\right)^H \right] \left(\begin{array}{c} R_{12} \\ \hline\hline C_2 \\ \hline\hline D_2 \end{array}\right)$$

**Algorithm:** $[R, C, D, T] := \textsc{UpAndDownDate\_unb}(R, C, D, T)$

**Partition** $R \rightarrow \left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array}\right)$, $C \rightarrow \left(\begin{array}{c|c} C_L & C_R \end{array}\right)$, $D \rightarrow \left(\begin{array}{c|c} D_L & D_R \end{array}\right)$, $T \rightarrow \left(\begin{array}{c|c} T_{TL} & T_{TR} \\ \hline 0 & T_{BR} \end{array}\right)$

**where** $R_{TL}$ and $T_{TL}$ are $0 \times 0$, $C_L$ and $D_L$ have 0 columns

**while** $m(R_{TL}) < m(R)$ **do**

**Repartition**

$$\left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} R_{00} & r_{01} & R_{02} \\ \hline 0 & \rho_{11} & r_{12}^T \\ \hline 0 & 0 & R_{22} \end{array}\right), \left(\begin{array}{c|c} T_{TL} & T_{TR} \\ \hline 0 & T_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} T_{00} & t_{01} & T_{02} \\ \hline 0 & \tau_{11} & t_{12}^T \\ \hline 0 & 0 & T_{22} \end{array}\right),$$

$$\left(\begin{array}{c|c} C_L & C_R \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} C_0 & c_1 & C_2 \end{array}\right), \left(\begin{array}{c|c} D_L & D_R \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} D_0 & d_1 & D_2 \end{array}\right)$$

**where** $\rho_{11}$ and $\tau_{11}$ are scalars, $c_1$ and $d_1$ are columns

---

$[\rho_{11}, c_1, d_1, \tau_{11}] := \mathrm{UDHouse}(\rho_{11}, c_1, d_1)$

$\left[r_{12}^T, C_2, D_2\right] := \mathrm{ApplyUDHouse}\left(\tau_{11}, c_1, d_1, r_{12}^T, C_2, D_2\right)$

---

**Continue with**

$$\left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} R_{00} & r_{01} & R_{02} \\ \hline 0 & \rho_{11} & r_{12}^T \\ \hline 0 & 0 & R_{22} \end{array}\right), \left(\begin{array}{c|c} T_{TL} & T_{TR} \\ \hline 0 & T_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} T_{00} & t_{01} & T_{02} \\ \hline 0 & \tau_{11} & t_{12}^T \\ \hline 0 & 0 & T_{22} \end{array}\right),$$

$$\left(\begin{array}{c|c} C_L & C_R \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} C_0 & c_1 & C_2 \end{array}\right), \left(\begin{array}{c|c} D_L & D_R \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} D_0 & d_1 & D_2 \end{array}\right)$$

**endwhile**

Figure 3: Unblocked algorithm for up-and-downdating.

**Algorithm:** $[R, C, D, T] := \textsc{UpAndDownDate\_blk}(R, C, D, T)$

**Partition** $R \rightarrow \left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array}\right)$, $C \rightarrow \left(\begin{array}{c|c} C_L & C_R \end{array}\right)$, $D \rightarrow \left(\begin{array}{c|c} D_L & D_R \end{array}\right)$, $T \rightarrow \left(\begin{array}{c|c} T_L & T_R \end{array}\right)$

**where** $R_{TL}$ and $T_L$ are $0 \times 0$, $C_L$ and $D_L$ have 0 columns

**while** $m(R_{TL}) < m(R)$ **do**

**Determine block size** $b$

**Repartition**

$$\left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} R_{00} & R_{01} & R_{02} \\ \hline 0 & R_{11} & R_{12} \\ \hline 0 & 0 & R_{22} \end{array}\right), \left(\begin{array}{c|c} T_L & T_R \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} T_0 & T_1 & T_2 \end{array}\right),$$

$$\left(\begin{array}{c|c} C_L & C_R \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} C_0 & C_1 & C_2 \end{array}\right), \left(\begin{array}{c|c} D_L & D_R \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} D_0 & D_1 & D_2 \end{array}\right)$$

**where** $R_{11}$ **and** $T_1$ **are** $b \times b$ **,** $C_1$ **and** $D_1$ **have** $b$ **columns**

---

$[R_{11}, C_1, D_1, T_1] := \mathrm{UpAndDownDate\_unb}(R_{11}, C_1, D_1, T_1)$

$[R_{12}, C_2, D_2] := \mathrm{ApplyBlkUDHouse}\,(T_1, C_1, D_1, R_{12}, C_2, D_2)$

---

**Continue with**

$$\left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} R_{00} & R_{01} & R_{02} \\ \hline 0 & R_{11} & R_{12} \\ \hline 0 & 0 & R_{22} \end{array}\right), \left(\begin{array}{c|c} T_L & T_R \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} T_0 & T_1 & T_2 \end{array}\right),$$

$$\left(\begin{array}{c|c} C_L & C_R \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} C_0 & C_1 & C_2 \end{array}\right), \left(\begin{array}{c|c} D_L & D_R \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} D_0 & D_1 & D_2 \end{array}\right)$$

**endwhile**

Figure 4: Blocked algorithm for up-and-downdating.

where $T_1 = \text{striu}(I + C_1^H C_1 - D_1^H D_1) + \frac{1}{2}\text{diag}(I + C_1^H C_1 - D_1^H D_1)$. The submatrix $T_1$ may be computed via the following steps[1]:

$$\begin{aligned} T_1 &:= \text{triu}(I + C_1^H C_1 - D_1^H D_1) \\ T_1 &:= \text{ScaleDiagonal}\left(\frac{1}{2}, T_1\right) \end{aligned}$$

where the ScaleDiagonal operation scales the diagonal of the second argument by the first argument. With $T_1$ computed, we may perform the update as follows:

$$\begin{aligned} W &:= T_1^{-H}(R_{12} + C_1^H C_2 + D_1^H D_2) \\ \left(\begin{array}{c} \tilde{R}_{12} \\ \hline\hline \tilde{C}_2 \\ \hline\hline \tilde{D}_2 \end{array}\right) &:= \left(\begin{array}{c} R_{12} \\ \hline\hline C_2 \\ \hline\hline D_2 \end{array}\right) - \left(\begin{array}{c} I \\ \hline\hline C_1 \\ \hline\hline -D_1 \end{array}\right) W \end{aligned}$$

This blocked algorithm contains within it blocked algorithms for updating and downdating, since $D$ or $C$ can be taken to be "empty".

# 6 Solving a System

Consider the matrices $B \in m_B \times n$, $C \in rm_C \times n$, and $D \in m_D \times n$. The up-and-downdating problem starts with a matrix $R$ such that $B^H B + C^H C = R^H R$. Where does this come from? Typically, it comes from solving the linear least-squares problem

$$\min_x \left\| \begin{pmatrix} B \\ D \end{pmatrix} x - \begin{pmatrix} b_B \\ b_D \end{pmatrix} \right\|.$$

There are two standard ways of solving this problem: normal equations and QR factorization (via Householder transformations). Since we are using Generalized Householder transformations to updowndate, we restrict ourselves to the case where the original $R$ came from (the equivalent of) a QR factorization.

So, we assume that we have computed (the equivalent of) the QR factorization

$$\begin{pmatrix} B \\ D \end{pmatrix} = QR$$

and then solved

$$Rx = Q^H \begin{pmatrix} b_B \\ b_D \end{pmatrix} = b_{BD}.$$

Now, after the updowndate step, what one is interested in is solving

$$\min_{\tilde{x}} \left\| \begin{pmatrix} B \\ C \end{pmatrix} \tilde{x} - \begin{pmatrix} b_B \\ b_C \end{pmatrix} \right\|$$

which could be solved via the QR factorization

$$\begin{pmatrix} B \\ C \end{pmatrix} = \tilde{Q}\tilde{R}$$

and then solved

$$\tilde{R}\tilde{x} = \tilde{Q}^H \begin{pmatrix} b_B \\ b_C \end{pmatrix} = b_{BC}.$$

[1] In practice, we find it most convenient to compute $T_1$ at then end of the unblocked algorithm for up-and-downdating, UpAndDownDate_Unb. Note that we omit this step from the algorithm shown in Figure 3 and instead only show the storing of the $\tau$ values along the diagonal of $T_1$.

Now, we have computed up-and-downdating Householder transformations $G_j$ so that

$$\left(\begin{array}{c} \tilde{R} \\ \hline\hline 0 \\ \hline\hline 0 \end{array}\right) = G_{n-1}^H \cdots G_0^H \left(\begin{array}{c} R \\ \hline\hline C \\ \hline\hline D \end{array}\right)$$

Note that

$$\left(\begin{array}{c} b_{BC} \\ \hline\hline \tilde{b}_C \\ \hline\hline \tilde{b}_D \end{array}\right) = G_{n-1}^H \cdots G_0^H \left(\begin{array}{c} b_{BD} \\ \hline\hline b_C \\ \hline\hline b_D \end{array}\right).$$

Thus, applying the block up-and-downdating Householder transformations from the left will up-and-downdate $b_{BD}$ into $b_{BC}$. This computation may be performed via the same ApplyBlkUDHouse operation described in the previous section and used in Figure 4:

$$\begin{array}{rcl} \left[\tilde{R}, \tilde{C}, \tilde{D}, \tilde{T}\right] & := & \text{UpAndDownDate\_Blk}\,(R, C, D, T) \\ \left[b_{BC}, \tilde{b}_C, \tilde{b}_D\right] & := & \text{ApplyBlkUDHouse}\left(\tilde{T}, \tilde{C}, \tilde{D}, b_{BD}, b_C, b_D\right) \end{array}$$

At this point, $R$ and $b_{BD}$ have been up-and-downdated to $\tilde{R}$ and $b_{BC}$, respectively, and so a new solution to the system may be computed by solving

$$\tilde{R}\tilde{x} = b_{BC}.$$

# 7 An Algorithm-by-Blocks

As part of the FLAME project, we have developed and reported on algorithm-by-blocks for various linear algebra operations and how to schedule them to distributed memory as well as multithreaded parallel architectures. An algorithm-by-blocks views a matrix as a collection of submatrices (blocks), possibly hierarchically. Each block becomes a unit of data and computation with blocks become units of computation.

- In [9, 6] we give algorithms-by-tiles for out-of-core LU and QR factorization. A tile is a block that corresponds to a unit for I/O. By modifying the pivoting strategy for LU factorization and the computation of Householder transformations for QR factorization, the computation can be case in terms of operations with blocks while only increasing the operation count by a lower order term.

- In [3] a runtime system, SuperMatrix, for scheduling algorithm-by-blocks to multiple threads is introduced. Implementations of algorithms-by-blocks utilizing this runtime system are discussed in a large number of conference papers and summarized in a journal paper [12]. The idea is that the algorithm-by-blocks generates a Directed Acyclic Graph (DAG) of operations and dependencies which are then scheduled for execution by threads at runtime.

The effort focuses on solving the programmability problem: algorithms are coded in a style that closely resembles the algorithms in the figures in this paper. The algorithm-by-blocks is coded in a very similar style. By separating the generation of the DAG by the algorithm from the scheduling of that DAG, the library routine needs not change when the scheduling policy is modified.

Details of the algorithm-by-blocks for up-and-downdating are essentially identical to those of the updating algorithm-by-tiles in [6] and the QR factorization algorithm-by-blocks scheduled with SuperMatrix in [11] or PLASMA in [2], except that minor modifications are made when computing with parts of the matrix that is removed as part of the downdating. Thus, we don't give further details here and merely report performance, in the next section.

# 8 Performance

In this section, we provide performance results for various implementations of the up-and-downdating algorithm, including a high-performance algorithm-by-blocks.

All experiments were performed using double-precision floating-point arithmetic on a Dell PowerEdge R900 server consisting of four Intel "Dunnington" six-core processors, providing a total of 24 cores with a combined peak performance of 255 GFLOPs ($255 \times 10^9$ floating-point operations per second) with 96 GBytes of shared main memory. Performance experiments were gathered under the GNU/Linux 2.6.18 operating system. Source code was compiled by the Intel C/C++ Compiler, version 11.1.

In addition to reporting performance for implementations of the up-and-downdating operation, for comparison we also provide performance data for QR factorization via the UT transform, as the two operations are closely related. We report performance for the following implementations in Figures 5 and 6:

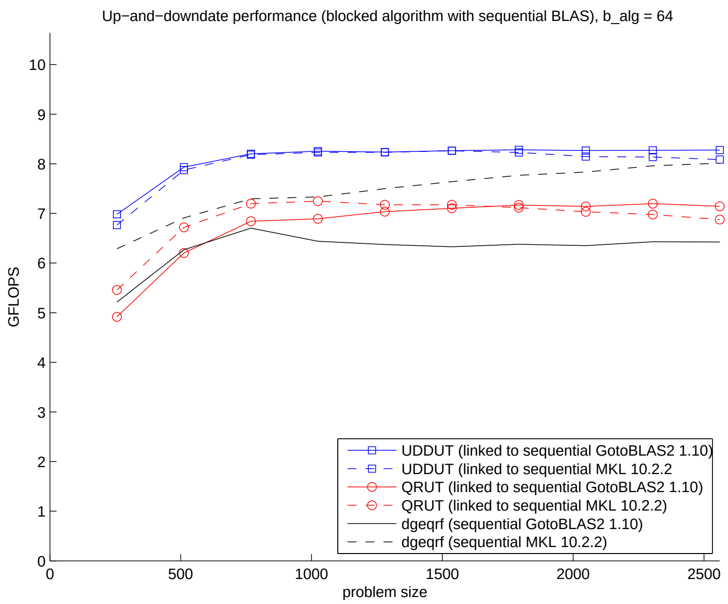
- UDDUT. A sequential implementation of the blocked algorithm for the up-and-downdating operation shown in Figure 4.
- QRUT. A sequential implementation of a blocked algorithm for a QR factorization via the UT transform [7].
- UDDUTABB. A multithreaded implementation of an algorithm-by-blocks for the up-and-downdating operation.
- QRUTABB. A multithreaded implementation of an algorithm-by-blocks for a QR factorization via the UT transform [10].
- sequential `dgeqrf`. A sequential implementation of the LAPACK QR factorization routine.
- multithreaded `dgeqrf`. A multithreaded implementation of the LAPACK QR factorization routine.

These implementations were timed in two ways: linked to a sequential build of GotoBLAS2 1.10 and linked to sequential build of Intel's MKL 10.2.2. The `dgeqrf` implementations, likewise, were obtained from both GotoBLAS2 1.10 and MKL 10.2.2. Also, parallelism was obtained from the UDDUTABB and QRUTABB via the SuperMatrix runtime system [3, 4].

Performance results are computed using an operation count of $2n^2(m_C + m_D)$ for the up-and-downdate operation and $2n^2(m - \frac{n}{3})$ for a QR factorization. This counts *useful* operations, ignoring extra operations that are performed so that the blocked algorithms can cast computation in terms of matrix-matrix multiplication. The $y$-axes of the graphs are scaled to indicate the peak performance for the number of cores utilized.

In Figure 5 (top) we report the performance of the blocked algorithms using a single core, choosing the block size equal to 64. The rates of computation achieved by the up-and-downdating algorithms is better than those achieved by the QR factorization because more computation is cast in terms of matrix-matrix multiplication. Timings for the same blocked algorithms using 24 cores and an algorithmic block size of 256 are given in the bottom graph. We note that while MKL's `dgeqrf` achieves very good performance, our implementation of the QR factorization and the up-and-downdating algorithm does not when linked to MKL's multithreaded BLAS. This is likely due to how the matrix-matrix multiplication (dgemm) is parallelized. When linked to the GotoBLAS2, the performance is much improved, although still well below peak.

In Figure 6 we report the performance of the algorithms-by-blocks. The ability to store matrices by blocks combined with a run-time system that schedules operations to threads greatly improves performance. When the storage block size ($b_{\text{store}}$) and algorithmic blocksize ($b_{\text{alg}}$) are relatively large, ramp-up is slow while the asymptotic performance is better.



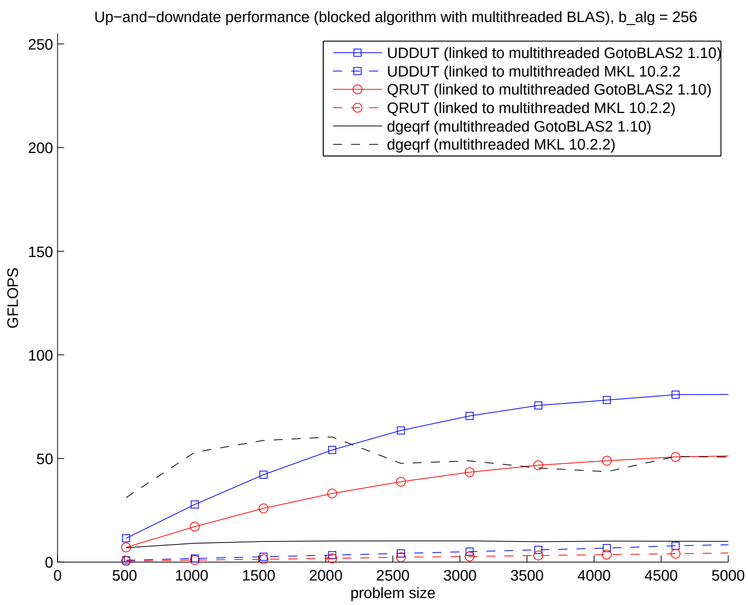
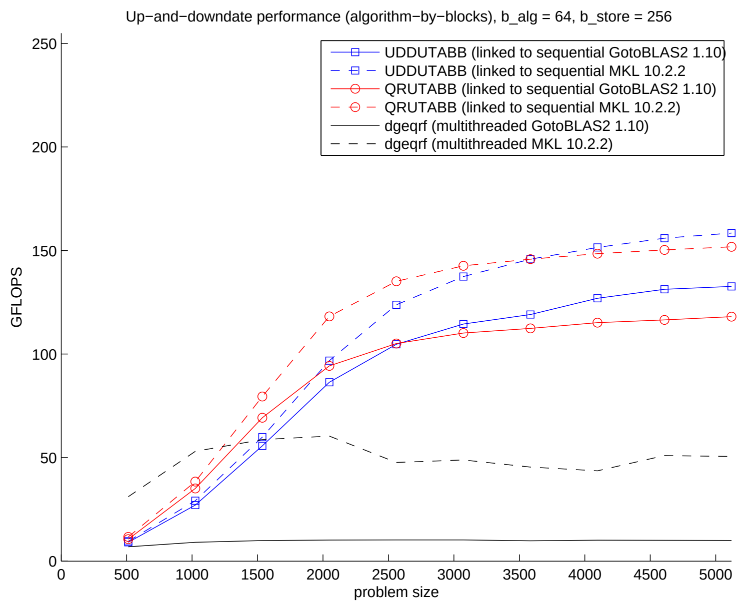




Figure 5: Performance of sequential and multithreaded implementations of the up-and-downdating operation. Top: Sequential up-and-downdating implementations compared to various sequential QR factorizations, using an algorithmic block size of 64. Bottom: Blocked algorithm for up-and-downdating linked to multithreaded BLAS compared to various multithreaded QR factorizations, with an algorithm block size of 256.



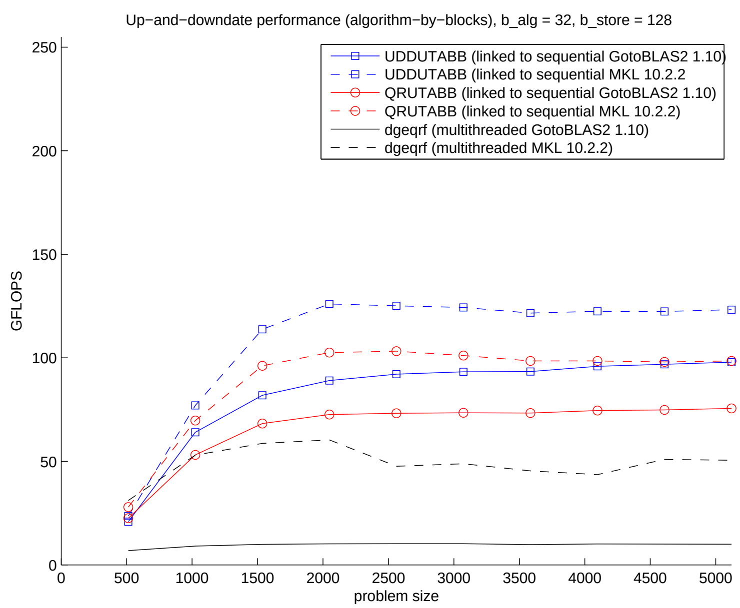




Figure 6: Multithreaded algorithm-by-blocks for up-and-downdating compared to various multithreaded QR factorizations, including the multithreaded QR factorization in MKL. Algorithmic and storage blocksizes are chosen to equal 64 and 256, respectively, in the top graph and 32 and 128 in the bottom graph.

# 9 Conclusion

In this paper, we have presented unblocked and blocked algorithms for the up- and/or downdating problem. It has been shown that blocked algorithms can be easily formulated and that high performance can be achieved.

## Acknowledgments

This research was partially sponsored by NSF grant CCF-0917167 and a grant from Microsoft.

*Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation (NSF).*

# References

[1] Christian Bischof and Charles Van Loan. The WY representation for products of Householder matrices. *SIAM J. Sci. Stat. Comput.*, 8(1):s2–s13, Jan. 1987.

[2] Alfredo Buttari, Julien Langou, Jakub Kurzak, and Jack Dongarra. Parallel tiled qr factorization for multicore architectures. *Concurr. Comput. : Pract. Exper.*, 20(13):1573–1590, 2008.

[3] Ernie Chan, Enrique Quintana-Ortí, Gregorio Quintana-Ortí, and Robert van de Geijn. SuperMatrix out-of-order scheduling of matrix operations for SMP and multi-core architectures. In *SPAA '07: Proceedings of the Nineteenth ACM Symposium on Parallelism in Algorithms and Architectures*, pages 116–126, 2007.

[4] Ernie Chan, Field G. Van Zee, Paolo Bientinesi, Enrique S. Quintana-Ortí, Gregorio Quintana-Ortí, and Robert van de Geijn. Supermatrix: A multithreaded runtime scheduling system for algorithms-by-blocks. FLAME Working Note #25 TR-07-41, The University of Texas at Austin, Department of Computer Sciences, August 2007.

[5] Jack J. Dongarra, Jeremy Du Croz, Sven Hammarling, and Iain Duff. A set of level 3 basic linear algebra subprograms. *ACM Trans. Math. Soft.*, 16(1):1–17, March 1990.

[6] Brian C. Gunter and Robert A. van de Geijn. Parallel out-of-core computation and updating the QR factorization. *ACM Transactions on Mathematical Software*, 31(1):60–78, March 2005.

[7] Thierry Joffrain, Tze Meng Low, Enrique S. Quintana-Ortí, Robert van de Geijn, and Field G. Van Zee. Accumulating householder transformations, revisited. *ACM Trans. Math. Softw.*, 32(2):169–179, 2006.

[8] C. Puglisi. Modification of the Householder method based on the compact wy representation. *SIAM J. Sci. Stat. Comput.*, 13:723–726, 1992.

[9] Enrique S. Quintana-Ortí and Robert A. van de Geijn. Updating an LU factorization with pivoting. *ACM Trans. Math. Softw.*, 35(2):1–16, 2008.

[10] Gregorio Quintana-Ortí, Enrique Quintana-Ortí, Ernie Chan, Field G. Van Zee, and Robert van de Geijn. Scheduling of QR factorization algorithms on SMP and multi-core architectures. FLAME Working Note #24 TR-07-37, The University of Texas at Austin, Department of Computer Sciences, July 2007.

[11] Gregorio Quintana-Orti, Enrique S. Quintana-Orti, Ernie Chan, Robert A. van de Geijn, and Field G. Van Zee. Scheduling of QR factorization algorithms on SMP and multi-core architectures. In *PDP '08: Proceedings of the 16th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP 2008)*, pages 301–310, Washington, DC, USA, 2008. IEEE Computer Society.

[12] Gregorio Quintana-Ortí, Enrique S. Quintana-Ortí, Robert A. van de Geijn, Field G. Van Zee, and Ernie Chan. Programming matrix algorithms-by-blocks for thread-level parallelism. *ACM Transactions on Mathematical Software*, 36(3):14:1–14:26, July 2009.

[13] Robert Schreiber and Charles Van Loan. A storage-efficient WY representation for products of Householder transformations. *SIAM J. Sci. Stat. Comput.*, 10(1):53–57, Jan. 1989.

[14] Michael Stewart and G. W. Stewart. On hyperbolic triangularization: Stability and pivoting. *SIAM Journal on Matrix Analysis and Applications*, 19(4):847–860, 1998.

[15] Xiaobai Sun. Aggregations of elementary transformations. Technical Report Technical report DUKE–TR–1996–03, Duke University, 1996.

[16] H. F. Walker. Implementation of the GMRES method using Householder transformations. *SIAM J. Sci. Stat. Comput.*, 9(1):152–163, 1988.

[17] Wen-Ming Yan and Kuo-Liang Chung. A block representation for products of hyperbolic householder transform. *Applied Mathematics Letters*, 10(1):109–112, January 1997.