Blocked Cholesky factorization updates of the Riccati recursion using hyperbolic Householder transformations

Pieter Pas and Panagiotis Patrinos

***Abstract*— Newton systems in quadratic programming (QP) methods are often solved using direct Cholesky or $LDL^\top$ factorizations. When the linear systems in successive iterations differ by a low-rank modification (as is common in active set and augmented Lagrangian methods), updating the existing factorization can offer significant performance improvements over recomputing a full Cholesky factorization. We review the hyperbolic Householder transformation, and demonstrate its usefulness in describing low-rank Cholesky factorization updates. By applying this hyperbolic Householder-based framework to the well-known Riccati recursion for solving saddle-point problems with optimal control structure, we develop a novel algorithm for updating the factorizations used in optimization solvers for optimal control. Specifically, the proposed method can be used to efficiently solve the semismooth Newton systems that are at the core of the augmented Lagrangian-based QPALM-OCP solver. An optimized open-source implementation of the proposed factorization update routines is provided as well.**

I. Introduction

In many types of numerical optimization solvers, the most computationally demanding steps are the construction and factorization of the linear system that determines a Newton step. In practice, such a system could be solved using a Cholesky decomposition $LL^\top$ or $LDL^\top$, where L is lower triangular and D diagonal. When two Newton systems in successive iterations of the solver are similar, it may be advantageous to *update* the existing factorization from the previous iteration, rather than to perform a new Cholesky decomposition in the current iterate. Active set solvers are one prominent class of methods where this is the case: from one iteration to the next, a single constraint (or a small number of constraints) is added to or removed from the active set, and this translates to the addition of a low-rank term in the Newton system [17]. Techniques for computing the updated Cholesky factor $\tilde{L}$ of the matrix $LL^\top + \sigma aa^\top$ (for a known triangular factor L and a rank-one update by a vector a) are well known: an early overview of various methods was given by Gill et al. in [5]. Their factorization update methods have seen widespread use in active set [8], [10], augmented Lagrangian [6], [9] and even interior point solvers [3]. Since a factorization update for a rank-one modification of an n-by-n matrix can be computed in $\mathcal{O}(n^2)$ operations as opposed to $\mathcal{O}(n^3)$ operations for a full Cholesky factorization, using them effectively can significantly improve solver run times.

An alternative framework for describing factorization updates is based on hyperbolic Householder transformations, a generalization of the standard orthogonal Householder transformations often used to compute QR factorizations. This framework was originally developed by Rader and Steinhardt for adding and removing data in least squares problems for signal processing applications [12]. Subsequent studies analyzed the numerical properties of hyperbolic Householder transformations for the problem of updating Cholesky factorizations [2], [14]. This Householder-based approach to factorization updates naturally generalizes to the case where multiple rank-one matrices are added or subtracted at once. That is, it provides an intuitive way of representing the low-rank updating problem

$$\tilde{L}\tilde{L}^\top = LL^\top + A\Sigma A^\top, \tag{1}$$

where we wish to find the lower triangular Cholesky factor $\tilde{L}$ of the matrix $LL^\top + A\Sigma A^\top \in \mathbb{R}^{n\times n}$, for some known Cholesky decomposition $LL^\top$ and a low-rank modification by the update matrix $A \in \mathbb{R}^{n\times m}$ and a diagonal matrix Σ. We will refer to this problem simply as the *updating* problem, even though the specific case where $\Sigma \prec 0$ is sometimes referred to as a *downdating* problem. Factorization updates of the form (1) are of interest to us, because they match exactly the type of low-rank modifications that are encountered in the inner semismooth Newton method at the core of the augmented Lagrangian-based QPALM solver [6, §4.1.2].

The hyperbolic Householder framework has two main advantages over the methods from [5]: 1. It simplifies the notation in theoretical derivations, providing a convenient mental model for reasoning about factorization updates; and 2. by merging multiple rank-one updates in a single operation, it enables high-performance practical implementations. While such blocked implementations built on top of BLAS have been shown to scale well for very large matrices [16], optimized versions for smaller matrices (such as the ones used in optimal control) are not yet available.

Contributions

The goal of the present paper is threefold:

- Provide a self-contained overview of hyperbolic Householder transformations, specifically in the context of the Cholesky factorization updating problem (1). (§II, §III)
- Present a practical use case for the hyperbolic Householder framework, using it to derive a novel method for updating the Cholesky factorizations in a solver for

This work was supported by FWO PhD grant 11M9523N; FWO projects G081222N, G0A0920N, G033822N; KU Leuven internal funding C14/24/103, C14/18/068, 12Y7622N; the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 953348; and EuroHPC Project 101118139 Inno4Scale.

The authors are with Department of Electrical Engineering (ESAT), STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics, KU Leuven, Kasteelpark Arenberg 10, 3001 Leuven, Belgium. (e-mail: {pieter.pas,panos.patrinos}@esat.kuleuven.be)

saddle-point problems with optimal control structure, based on the well-known Riccati recursion. (§IV)
- Demonstrate the practical performance of hyperbolic Householder operations using optimized, open-source C++ implementations of the presented algorithms. (§V)

The Riccati-based factorization update routines, along with their efficient implementation, offer a compelling approach for solving Newton systems in the recently proposed QPALM-OCP solver for linear-quadratic optimal control problems [9].

Notation

We use $\mathbb{L}$, $\mathbb{U}$ and $\mathbb{D}$ to denote the sets of lower triangular, upper triangular and diagonal matrices respectively. The symbol $\mathbf{e}_1$ denotes the first standard basis vector $(1\ 0\ ...\ 0)$. For vectors, x_i refers to the i-th element of x. For block matrices, A_{ij} refers to the subblock in block row i and block column j of A. In pseudocode, an empty matrix $(\,)$ represents the neutral element for block matrix concatenation. We use tildes to denote "updated" matrices. Subscripts in parentheses denote the iteration index in an algorithm, and in the context of optimal control, a superscript (for vectors) or subscript (for matrices) is used for the stage index.

II. Hyperbolic Householder transformations

A. Construction and properties

Definition 1 ($\mathcal{S}$-orthogonality): Given a diagonal matrix $\mathcal{S} \in \mathbb{D}(\mathbb{R}^n)$, we say that a matrix $\breve{Q} \in \mathbb{R}^{n\times n}$ is $\mathcal{S}$-orthogonal if

$$\breve{Q}\mathcal{S}\breve{Q}^\top = \mathcal{S}. \tag{2}$$

In the case where $\mathcal{S} = \mathrm{I}$, Definition 1 reduces to the definition of standard orthogonality, $\breve{Q}\breve{Q}^\top = \mathrm{I}$. It is easy to show that if $\breve{Q}_{(1)}$ and $\breve{Q}_{(2)}$ are both $\mathcal{S}$-orthogonal, then their product $\breve{Q}_{(1)}\breve{Q}_{(2)}$ is also $\mathcal{S}$-orthogonal.

An important application of standard Householder transformations is the introduction of zeros during e.g. QR factorization. Additionally, their orthogonality allows efficient application of their inverses. The following lemma shows that a generalized *hyperbolic* Householder transformation can be constructed, with properties similar to the standard Householder transformation: A hyperbolic Householder matrix is $\mathcal{S}$-orthogonal, and can be constructed to reduce all but the first element of a given vector to zero.

Lemma 1 (Hyperbolic Householder reflectors): Consider a matrix $\mathcal{S} \triangleq \left(\begin{smallmatrix} 1 & \\ & \Sigma \end{smallmatrix}\right) \in \mathbb{D}(\mathbb{R}^n)$ and a vector $x \in \mathbb{R}^n$ with $x^\top\mathcal{S}x > 0$. Then the matrix $\breve{Q} \in \mathbb{R}^{n\times n}$ defined below is $\mathcal{S}$-orthogonal, and satisfies $\breve{Q}^\top x = \eta\mathbf{e}_1$.

$$\breve{Q} \triangleq \mathrm{I} - \frac{2}{v^\top\mathcal{S}v}\mathcal{S}vv^\top, \quad \begin{array}{l}\text{where } v \triangleq x - \eta\mathbf{e}_1,\\ \text{and}\quad \eta \triangleq -\operatorname{sign}(x_1)\sqrt{x^\top\mathcal{S}x}.\end{array}^{1} \tag{3}$$

Proof: The $\mathcal{S}$-orthogonality of $\breve{Q}$ and the value of $\breve{Q}^\top x$ follow from straightforward algebraic manipulations.

$$\breve{Q}\mathcal{S}\breve{Q}^\top = \mathcal{S} - \tfrac{2\mathcal{S}vv^\top\mathcal{S}}{v^\top\mathcal{S}v} - \tfrac{2\mathcal{S}vv^\top\mathcal{S}}{v^\top\mathcal{S}v} + \tfrac{2\mathcal{S}v}{v^\top\mathcal{S}v}(v^\top\mathcal{S}v)\tfrac{2v^\top\mathcal{S}}{v^\top\mathcal{S}v} = \mathcal{S}.$$

Since $v^\top\mathcal{S}v = x^\top\mathcal{S}x - 2x^\top\mathcal{S}\eta\mathbf{e}_1 + \eta^2 = 2\eta^2 - 2x_1\eta = -2\eta v_1$,

$$\breve{Q}^\top x = x - \tfrac{2v^\top\mathcal{S}x}{v^\top\mathcal{S}v}v = x - \tfrac{2v^\top\mathcal{S}(v+\eta\mathbf{e}_1)}{v^\top\mathcal{S}v}v = x - v = \eta\mathbf{e}_1.$$

Finally, the choice of the sign of η ensures that $-2x_1\eta \geq 0$, thus $v^\top\mathcal{S}v > 0$, and $\breve{Q}$ is well-defined. ∎

Remark 1 (Hyperbolic Householder matrix construction): Even without knowing $\breve{Q}$, its $\mathcal{S}$-orthogonality and the fact that only the first component of $\breve{Q}^\top x$ is nonzero are sufficient to determine (up to a sign) the value of this component η: $x^\top\mathcal{S}x = x^\top\breve{Q}\mathcal{S}\breve{Q}^\top x = \eta^2$. The construction of $\breve{Q}$ in Lemma 1 is based on that of the classical Householder reflector [15, Fig. 10.1], generalized for quadratic forms using $\mathcal{S}$ rather than the Euclidean inner product.

Remark 2 (Alternative definitions): Rader and Steinhardt originally defined the following symmetric hyperbolic Householder matrix $\breve{Q}_{\mathrm{RS}} = \mathcal{S} - \frac{2}{v^\top\mathcal{S}v}vv^\top$ [12, Eq. 28]. When $\mathcal{S}$ consists only of diagonal elements ±1, which is the case considered in [12], their definition is related to (3) through the identity $\breve{Q} = \mathcal{S}\breve{Q}_{\mathrm{RS}}$, which can be seen using $\mathcal{S}^2 = \mathrm{I}$. The definition in (3) is inspired by the one used in [14, §3], and even though symmetry is lost, it has the advantage of generalizing to matrices $\mathcal{S}$ with entries different from ±1. We note that $\breve{Q}$ and η in Lemma 1 are unique only up to some signs, and similar results can be formulated if $\breve{Q}$ is multiplied by an $\mathcal{S}$-orthogonal matrix $\mathbf{diag}(\pm1\ \dots\ \pm1)$.

The hyperbolic Householder transformation introduced in Lemma 1 serves as a building block for an $\mathcal{S}$-orthogonal generalization of the QR factorization or its transpose, the LQ factorization. The $L\breve{Q}$ factorization described by the following proposition (with L lower triangular and $\breve{Q}$ $\mathcal{S}$-orthogonal) enables $\mathcal{S}$-orthogonal triangularization, a key step in performing Cholesky factorization updates.

Proposition 1 (Hyperbolic $L\breve{Q}$ factorization): Consider a matrix $\begin{pmatrix} L & A \end{pmatrix}$ with $L \in \mathbb{L}(\mathbb{R}^{n\times n})$ and $A \in \mathbb{R}^{n\times m}$, and a diagonal matrix $\Sigma \in \mathbb{D}(\mathbb{R}^m)$ for which $LL^\top + A\Sigma A^\top \succ 0$. Then there exists an $\mathcal{S}$-orthogonal matrix $\breve{Q} \in \mathbb{R}^{(n+m)\times(n+m)}$ with $\mathcal{S} = \left(\begin{smallmatrix} \mathrm{I}_n & \\ & \Sigma \end{smallmatrix}\right)$ such that

$$\begin{pmatrix} L & A \end{pmatrix}\breve{Q} = \begin{pmatrix} \tilde{L} & 0 \end{pmatrix}, \tag{4}$$

with $\tilde{L} \in \mathbb{L}(\mathbb{R}^{n\times n})$ lower triangular.

Proof: We will recursively construct $\breve{Q}$ as a product of $\mathcal{S}$-orthogonal transformations, each annihilating one row of A. The base case for size $n = 1$ is covered by Lemma 1. For $n > 1$, partition the given matrices as follows:

$$\left(\begin{array}{c|c|c} \lambda & & a^\top \\ \hline \ell & L' & A' \end{array}\right) \triangleq \begin{pmatrix} L & A \end{pmatrix}, \quad \begin{array}{l}\text{with } \lambda \in \mathbb{R}, \ell \in \mathbb{R}^{n-1},\\ L' \in \mathbb{L}(\mathbb{R}^{(n-1)\times(n-1)}),\\ a \in \mathbb{R}^m, A' \in \mathbb{R}^{(n-1)\times m}.\end{array} \tag{5}$$

From the assumption $\tilde{H} \triangleq LL^\top + A\Sigma A^\top \succ 0$, we know that the top left element of $\tilde{H}$ is strictly positive [1, Prop. 8.2.8], i.e. $x^\top\left(\begin{smallmatrix} 1 & \\ & \Sigma \end{smallmatrix}\right)x > 0$ with $x \triangleq \left(\begin{smallmatrix} \lambda \\ a \end{smallmatrix}\right)$. We can therefore invoke Lemma 1 to construct a $\left(\begin{smallmatrix} 1 & \\ & \Sigma \end{smallmatrix}\right)$-orthogonal matrix $\breve{Q}_{(n)}$ that annihilates a, i.e. $\begin{pmatrix} \lambda & a^\top \end{pmatrix}\breve{Q}_{(n)} = \begin{pmatrix} \tilde{\lambda} & 0 \end{pmatrix}$, with $\tilde{\lambda} = -\operatorname{sign}(\lambda)\sqrt{\lambda^2 + a^\top\Sigma a}$. Partition $\breve{Q}_{(n)}$ by isolating its first row and column, and embed it into a larger matrix

[1] Using $\operatorname{sign}(0) = 1$.

$\breve{Q}^{\uparrow}_{(n)} \in \mathbb{R}^{(n+m)\times(n+m)}$ which is clearly $\mathcal{S}$-orthogonal:

$$\breve{Q}^{\uparrow}_{(n)} \triangleq \begin{pmatrix} \breve{q}_{(n),11} & & \breve{q}^{\top}_{(n),12} \\ & \mathrm{I}_{n-1} & \\ \breve{q}_{(n),21} & & \breve{Q}_{(n),22} \end{pmatrix}, \quad \begin{array}{l} \text{with } \breve{q}_{(n),11} \in \mathbb{R}, \\ \breve{q}_{(n),12},\, \breve{q}_{(n),21} \in \mathbb{R}^m. \end{array}$$

Defining $\begin{pmatrix} \tilde{\ell} & \tilde{A}' \end{pmatrix} \triangleq \begin{pmatrix} \ell & A' \end{pmatrix} \breve{Q}_{(n)}$, we have that

$$\begin{pmatrix} L & A \end{pmatrix} \breve{Q}^{\uparrow}_{(n)} = \left(\begin{array}{c|c|c} \tilde{\lambda} & & \\ \hline \tilde{\ell} & L' & \tilde{A}' \end{array}\right). \tag{6}$$

$\breve{Q}_{(n)}$ reduced all nondiagonal nonzeros in the top row to zero, and computed the first column of $\tilde{L}$. To proceed, we will assume inductively that Proposition 1 holds for size $n-1$, so that we can use it to annihilate $\tilde{A}'$ and to update L'. To apply the proposition, we require $\tilde{H}' \triangleq L'L'^{\top} + \tilde{A}'\Sigma\tilde{A}'^{\top} \succ 0$. This holds, because $\tilde{H}'$ is the Schur complement of $\tilde{\lambda}^2$ in

$$\begin{aligned} \left(\begin{array}{c|c} \tilde{\lambda}^2 & \tilde{\lambda}\tilde{\ell}^{\top} \\ \hline \tilde{\lambda}\tilde{\ell} & \tilde{\ell}\tilde{\ell}^{\top} + \tilde{H}' \end{array}\right) &= \left(\begin{array}{c|c|c} \tilde{\lambda} & & \\ \hline \tilde{\ell} & L' & \tilde{A}' \end{array}\right) \mathcal{S} \left(\begin{array}{c|c|c} \tilde{\lambda} & & \\ \hline \tilde{\ell} & L' & \tilde{A}' \end{array}\right)^{\top} \\ &\overset{(6)}{=} \begin{pmatrix} L & A \end{pmatrix} \breve{Q}^{\uparrow}_{(n)} \mathcal{S} \breve{Q}^{\uparrow\top}_{(n)} \begin{pmatrix} L & A \end{pmatrix}^{\top} = \tilde{H}, \end{aligned} \tag{7}$$

where we used the $\mathcal{S}$-orthogonality of $\breve{Q}^{\uparrow}_{(n)}$ in the last step. Since $\tilde{H} \succ 0$ and $\tilde{\lambda}^2 > 0$, $\tilde{H}' = \tilde{H}/\tilde{\lambda}^2 \succ 0$ [7, Thm. 7.7.7]. Recursive application of the proposition then provides us with an $\begin{pmatrix} \mathrm{I}_{n-1} & \\ & \Sigma \end{pmatrix}$-orthogonal matrix $\breve{Q}'$ that satisfies

$$\begin{pmatrix} L' & \tilde{A}' \end{pmatrix} \breve{Q}' = \begin{pmatrix} \tilde{L}' & 0 \end{pmatrix}. \tag{8}$$

The matrix $\breve{Q}'$ can again be embedded into a larger $\mathcal{S}$-orthogonal matrix $\breve{Q}^{\uparrow}_{(n-1,1)} \triangleq \begin{pmatrix} 1 & \\ & \breve{Q}' \end{pmatrix}$. Therefore,

$$\begin{aligned} \begin{pmatrix} L & A \end{pmatrix} \underbrace{\breve{Q}^{\uparrow}_{(n)} \breve{Q}^{\uparrow}_{(n-1,1)}}_{\triangleq \breve{Q}} &\overset{(6)}{=} \left(\begin{array}{c|c|c} \tilde{\lambda} & & \\ \hline \tilde{\ell} & L' & \tilde{A}' \end{array}\right) \begin{pmatrix} 1 & \\ & \breve{Q}' \end{pmatrix} \\ &\overset{(8)}{=} \left(\begin{array}{c|c|c} \tilde{\lambda} & & \\ \hline \tilde{\ell} & \tilde{L}' & 0 \end{array}\right). \end{aligned} \tag{9}$$

Defining $\tilde{L} \triangleq \begin{pmatrix} \tilde{\lambda} & \\ \tilde{\ell} & \tilde{L}' \end{pmatrix}$ concludes the proof. ∎

Corollary 1 (Factorization updates):
Consider a Cholesky factor $L \in \mathbb{L}(\mathbb{R}^{n\times n})$ of the symmetric positive definite matrix $H \triangleq LL^{\top}$. Let $\tilde{H} \triangleq H + A\Sigma A^{\top}$ be positive definite for a given update matrix $A \in \mathbb{R}^{n\times m}$ and a diagonal matrix $\Sigma \in \mathbb{D}(\mathbb{R}^m)$. Then there exists an $\mathcal{S}$-orthogonal matrix $\breve{Q}$ with $\mathcal{S} \triangleq \begin{pmatrix} \mathrm{I}_n & \\ & \Sigma \end{pmatrix}$, such that the matrix $\tilde{L} \in \mathbb{L}(\mathbb{R}^{n\times n})$ defined below is a Cholesky factor of $\tilde{H}$.

$$\begin{pmatrix} \tilde{L} & 0 \end{pmatrix} \triangleq \begin{pmatrix} L & A \end{pmatrix} \breve{Q}. \tag{10}$$

Proof: Proposition 1 ensures existence of such $\breve{Q}$. Multiplying both sides of (10) by $\mathcal{S}$ and by their own transposes:

$$\begin{aligned} \begin{pmatrix} \tilde{L} & 0 \end{pmatrix} \mathcal{S} \begin{pmatrix} \tilde{L} & 0 \end{pmatrix}^{\top} &= \begin{pmatrix} L & A \end{pmatrix} \breve{Q} \mathcal{S} \breve{Q}^{\top} \begin{pmatrix} L & A \end{pmatrix}^{\top} \\ \overset{(2)}{\Leftrightarrow} \quad \tilde{L}\tilde{L}^{\top} &= \begin{pmatrix} L & A \end{pmatrix} \begin{pmatrix} \mathrm{I} & \\ & \Sigma \end{pmatrix} \begin{pmatrix} L & A \end{pmatrix}^{\top} = \tilde{H}. \end{aligned} \tag{11}$$

∎

There are two key parts to Corollary 1: First, *if* there exists an $\begin{pmatrix} \mathrm{I} & \\ & \Sigma \end{pmatrix}$-orthogonal transformation $\breve{Q}$ that reduces the matrix $\begin{pmatrix} L & A \end{pmatrix}$ to a lower triangular matrix $\begin{pmatrix} \tilde{L} & 0 \end{pmatrix}$, then this matrix $\tilde{L}$ is a Cholesky factor of the updated matrix $LL^{\top} + A\Sigma A^{\top}$. Second, Proposition 1 confirms that such a transformation $\breve{Q}$ *does* exist if the updated matrix is positive definite. Moreover, its proof describes the construction of $\breve{Q}$ and $\tilde{L}$.

B. Efficient representation and application of hyperbolic Householder transformations

We now discuss some practical considerations regarding optimized software routines based on (hyperbolic) Householder transformations: 1. Applying an individual hyperbolic Householder transformation for each column limits the performance of numerical implementations. Instead we will consider a blocked variant that applies the product of multiple elementary hyperbolic Householder matrices at once. 2. The choice of $\breve{Q}$ in Lemma 1 flips the sign of the first component of x. Since diagonal elements of Cholesky factors are conventionally taken to be positive, we will preserve their sign during the update by negating the first column of $\breve{Q}$ (cfr. Remark 2). 3. The Householder reflection vectors v from (3) will be normalized so their first element v_1 equals one. This simplifies the blocked variant, and allows the storage of A to be reused for the reflection vectors.

Specifically, we represent the hyperbolic Householder matrix that annihilates the i-th row of A as follows:

$$\breve{Q}_{(i)} \triangleq \begin{pmatrix} \mathrm{I}_{i-1} & & \\ & \tau^{-1}_{(i)} - 1 & -\tau^{-1}_{(i)} b^{\top}_{(i)} \\ & \mathrm{I}_{n-i} & \\ & \tau^{-1}_{(i)} \Sigma b_{(i)} & \mathrm{I} - \tau^{-1}_{(i)} \Sigma b_{(i)} b^{\top}_{(i)} \end{pmatrix}, \tag{12}$$

with $\tau^{-1}_{(i)} = \frac{2v_1}{v^{\top}\mathcal{S}v}$ and $\begin{pmatrix} 1 \\ b_{(i)} \end{pmatrix} = v_1^{-1} v$ for v defined in (3).

The recursive nature of the proof of Proposition 1 motivates a recursive expression for the product of the hyperbolic Householder matrices that is constructed in (9).

Lemma 2 (Product of hyperbolic Householder matrices): The product of matrices $\breve{Q}_{(i)}$ as defined by (12) is given by

$$\breve{Q}_{(1)} \cdots \breve{Q}_{(k)} = \begin{pmatrix} T^{-1}_{(k)} - \mathrm{I} & & -T^{-1}_{(k)} B_{(k)} \\ & \mathrm{I}_{n-k-1} & \\ \Sigma B^{\top}_{(k)} T^{-1}_{(k)} & & \mathrm{I} - \Sigma B^{\top}_{(k)} T^{-1}_{(k)} B_{(k)} \end{pmatrix},$$

where $B_{(1)} \triangleq b^{\top}_{(1)}$, $\quad T_{(1)} \triangleq \tau_{(1)}$,

$$B_{(i)} \triangleq \begin{pmatrix} B_{(i-1)} \\ b^{\top}_{(i)} \end{pmatrix}, \quad T_{(i)} \triangleq \begin{pmatrix} T_{(i-1)} & B_{(i-1)} \Sigma b_{(i)} \\ & \tau_{(i)} \end{pmatrix}.$$

Proof: By induction. The base case $\breve{Q}_{(1)}$ is trivial using (12). Then assume that Lemma 2 holds for factors 1 through k, and carry out the block matrix multiplication $\left(\breve{Q}_{(1)} \cdots \breve{Q}_{(k)}\right) \breve{Q}_{(k+1)}$. We find that this product equals

$$\left(\begin{array}{c|c|c} U_{(k+1)} - \mathrm{I} & & -U_{(k+1)} B_{(k+1)} \\ \hline & \mathrm{I}_{n-k-2} & \\ \hline \Sigma B^{\top}_{(k+1)} U_{(k+1)} & & \mathrm{I} - \Sigma B^{\top}_{(k+1)} U_{(k+1)} B_{(k+1)} \end{array}\right),$$

where we introduced $U_{(k+1)} \triangleq \begin{pmatrix} T^{-1}_{(k)} & -T^{-1}_{(k)} B_{(k)} \Sigma b_{(k+1)} \tau^{-1}_{(k+1)} \\ & \tau^{-1}_{(k+1)} \end{pmatrix}$ and $B_{(k+1)}$ as defined above. Observing that $U_{(k+1)} = T^{-1}_{(k+1)}$ completes the proof [1, Fact 2.17.1]. ∎

Remark 3: The construction in Lemma 2, using the product of the reflection vectors B and a triangular matrix U, is

based on the *compact WY representation* of standard Householder matrices developed in [13]. The insight of using a recursion in terms of $T_{(i)}$ rather than $U_{(i)}$ was first published in [11]. Note that application of $T_{(k)}^{-1}$ is implemented using back substitution; no explicit inversion is performed.

Thanks to this representation of hyperbolic Householder products, application of $\breve{Q}$ to a given matrix requires mostly matrix–matrix multiplications. This results in high arithmetic intensity, and optimal performance on modern processors. In contrast, sequential application of individual Householder transformations is generally limited by memory bandwidth.

III. Algorithms for factorization updates

The constructive proof of Proposition 1 can be leveraged to turn Corollary 1 into a practical method for performing updates of a given Cholesky factorization. This section presents two core algorithms and later combines them into an efficient blocked factorization update procedure.

A. Two core algorithms, two perspectives

Given a triangular matrix $L_{11} \in \mathbb{L}(\mathbb{R}^{n\times n})$, an update matrix $A \in \mathbb{R}^{n\times m}$ and a diagonal matrix $\Sigma \in \mathbb{D}(\mathbb{R}^m)$, Algorithm 1 computes the $({}^{\mathrm{I}}{}_{\Sigma})$-orthogonal matrix $\breve{Q}$ and the updated lower triangular matrix $\tilde{L}_{11} \in \mathbb{L}(\mathbb{R}^{n\times n})$ such that, as in Corollary 1,

$$\begin{pmatrix}\tilde{L}_{11} & 0\end{pmatrix} = \begin{pmatrix}L_{11} & A_1\end{pmatrix}\breve{Q}. \tag{13}$$

In other words, given a Cholesky factor L_{11} of H, Algorithm 1 computes a Cholesky factor $\tilde{L}_{11}$ of $\tilde{H} \triangleq H + A_1\Sigma A_1^\top$.

Algorithm 2 uses Lemma 2 to apply the matrix $\breve{Q}$ generated by Algorithm 1 to matrices $L_{21} \in \mathbb{R}^{l\times n}$ and $A_2 \in \mathbb{R}^{l\times m}$:

$$\begin{pmatrix}\tilde{L}_{21} & \tilde{A}_2\end{pmatrix} = \begin{pmatrix}L_{21} & A_2\end{pmatrix}\breve{Q}. \tag{14}$$

Remark 4 (Two viewpoints of Algorithm 1): One way to view Algorithm 1 is simply as a function that maps a given Cholesky factor and update matrix to an updated Cholesky factor, i.e. $(L,\, A,\, \Sigma) \mapsto \tilde{L} = \mathrm{chol}(LL^\top + A\Sigma A^\top)$. However, Algorithm 1 is more than a black box that performs the update: it also records all operations needed for the update in a linear transformation $\breve{Q}$ that can be applied efficiently to other matrices as well (not just to L and A). This is a powerful perspective, because it enables blocked variants of the Cholesky factorization update procedures that are both more efficient in practice, and that support the derivation of factorization updates for the Riccati recursion in Section IV.

B. Blocked algorithm

As suggested by Remark 4, a combination of Algorithms 1 and 2 can be used to develop a *blocked* variant of the factorization update procedure, where the Cholesky factors and the update matrix are partitioned as follows:

$$\left(\begin{array}{c|c} L & A \end{array}\right) = \left(\begin{array}{cc|c} L_{11} & & A_1 \\ L_{21} & L_{22} & A_2 \end{array}\right), \tag{15}$$

with $L_{11} \in \mathbb{L}(\mathbb{R}^{r\times r}), A_1 \in \mathbb{R}^{r\times m}$, for some block size r. The updated Cholesky factor $\tilde{L} = \mathrm{chol}(LL^\top + A\Sigma A^\top)$ can then be computed by iterating the following steps:

0. Partition $\left(\begin{array}{cc|c} L_{11} & & A_1 \\ L_{21} & L_{22} & A_2 \end{array}\right) \leftarrow \left(\begin{array}{c|c} L & A \end{array}\right)$;
1. Find $\breve{Q}$ such that $\begin{pmatrix}\tilde{L}_{11} & 0\end{pmatrix} = \begin{pmatrix}L_{11} & A_1\end{pmatrix}\breve{Q}$;
2. Compute $\begin{pmatrix}\tilde{L}_{21} & \tilde{A}_2\end{pmatrix} = \begin{pmatrix}L_{21} & A_2\end{pmatrix}\breve{Q}$;
3. Go to 0. with $\left(\begin{array}{c|c} L & A \end{array}\right) \leftarrow \left(\begin{array}{c|c} L_{22} & \tilde{A}_2 \end{array}\right)$.

Each iteration updates the r leftmost columns of L, and the procedure is then applied recursively to the remainder L_{22} until all columns of L have been updated (at which point L_{22} is empty). Algorithms 1 and 2 are used in tandem to perform steps 1 and 2 respectively, ensuring that $\breve{Q}$ is $({}^{\mathrm{I}}{}_{\Sigma})$-orthogonal. These two steps can be viewed as a single multiplication $\begin{pmatrix}\tilde{L}_{11} & 0 \\ \tilde{L}_{21} & \tilde{A}_2\end{pmatrix} = \begin{pmatrix}L_{11} & A_1 \\ L_{21} & A_2\end{pmatrix}\breve{Q}$. The proof for this blocked algorithm is a straightforward generalization of the proof of Proposition 1. [2].

[2]Apply Prop. 1 to $(L_{11} \mid A_1)$ instead of applying Lem. 1 to (λ, a).

Algorithm 1: Block hyperbolic Householder update

In : $L_{11} \in \mathbb{L}(\mathbb{R}^{n\times n})$, $A_1 \in \mathbb{R}^{n\times m}$, $\Sigma \in \mathbb{D}(\mathbb{R}^m)$.
Out : $\tilde{L}_{11} \in \mathbb{L}(\mathbb{R}^{n\times n})$, $B \in \mathbb{R}^{n\times m}$, $T \in \mathbb{U}(\mathbb{R}^{n\times n})$.
Result: $\begin{pmatrix}\tilde{L}_{11} & 0\end{pmatrix} = \begin{pmatrix}L_{11} & A_1\end{pmatrix}\breve{Q}$,
where $\breve{Q} = \begin{pmatrix} T^{-1} - \mathrm{I} & -T^{-1}B \\ \Sigma B^\top T^{-1} & \mathrm{I} - \Sigma B^\top T^{-1} B \end{pmatrix}$ is $({}^{\mathrm{I}}{}_{\Sigma})$-orthogonal.

Initialize $L_{(1)} \leftarrow L_{11}$ and $A_{(1)} \leftarrow A_1$
Initialize $\tilde{L}_{(0)} \leftarrow ()$ and $B_{(0)} \leftarrow ()$ and $T_{(0)} \leftarrow ()$
for $k = 1, 2, \ldots n$
  Partition $\left(\begin{array}{c|c} \lambda & \\ \hline \ell & L_{(k+1)} \end{array}\right) \leftarrow L_{(k)}$ and $\left(\begin{array}{c} a^\top \\ \hline A' \end{array}\right) \leftarrow A_{(k)}$ with $\lambda \in \mathbb{R}$, $a \in \mathbb{R}^m$
  $\alpha^2 \leftarrow a^\top \Sigma a$, $\tilde{\lambda} \leftarrow \mathrm{sign}(\lambda)\sqrt{\lambda^2 + \alpha^2}$, $\beta \leftarrow \lambda + \tilde{\lambda}$
  $\tau_{(k)}^{-1} \leftarrow \frac{2\beta^2}{\alpha^2 + \beta^2}$, $b_{(k)} \leftarrow \beta^{-1} a$ ▷ Construct $\breve{Q}_{(k)}$
  $w \leftarrow \tau_{(k)}^{-1}\left(\ell + A'\Sigma b_{(k)}\right)$
  $\left(\tilde{\ell} \;\middle|\; A_{(k+1)}\right) \leftarrow \left(w - \ell \;\middle|\; A' - w\, b_{(k)}^\top\right)$ ▷ Apply $\breve{Q}_{(k)}$
  $\tilde{L}_{(k)} \leftarrow \left(\begin{array}{c|c} \tilde{L}_{(k-1)} & \tilde{\lambda} \\ & \tilde{\ell} \end{array}\right)$ ▷ Build updated Cholesky factor
  $B_{(k)} \leftarrow \begin{pmatrix} B_{(k-1)} \\ \hline b_{(k)}^\top \end{pmatrix}$, $T_{(k)} \leftarrow \left(\begin{array}{c|c} T_{(k-1)} & B_{(k-1)}\Sigma b_{(k)} \\ \hline & \tau_{(k)} \end{array}\right)$
$\tilde{L}_{11} = \tilde{L}_{(n)}$ and $B = B_{(n)}$ and $T = T_{(n)}$

Algorithm 2: Block hyperbolic Householder apply

In : $L_{21} \in \mathbb{R}^{l\times n}$, $A_2 \in \mathbb{R}^{l\times m}$, $\Sigma \in \mathbb{D}(\mathbb{R}^m)$,
$B \in \mathbb{R}^{n\times m}$, $T \in \mathbb{U}(\mathbb{R}^{n\times n})$.
Out : $\tilde{L}_{21} \in \mathbb{R}^{l\times n}$, $\tilde{A}_2 \in \mathbb{R}^{l\times m}$.
Result: $\begin{pmatrix}\tilde{L}_{21} & \tilde{A}_2\end{pmatrix} = \begin{pmatrix}L_{21} & A_2\end{pmatrix}\breve{Q}$,
with $\breve{Q} = \begin{pmatrix} T^{-1} - \mathrm{I} & -T^{-1}B \\ \Sigma B^\top T^{-1} & \mathrm{I} - \Sigma B^\top T^{-1} B \end{pmatrix}$.

$W \leftarrow \left(L_{21} + A_2\Sigma B^\top\right) T^{-1}$
$\tilde{A}_2 \leftarrow A_2 - WB$
$\tilde{L}_{21} \leftarrow W - L_{21}$

In the case where $\Sigma = \pm\mathrm{I}$, this algorithm requires $mn^2 + \frac{r-1}{4}n^2 + \frac{r-1}{2}mn - \frac{r^2-3r+2}{4}n$ fused multiply-add operations, $\frac{1}{2}n^2 + mn + \frac{1}{2}nr + \mathcal{O}(n)$ multiplications, $\frac{1}{2}n^2 + \mathcal{O}(n)$ additions, $2n$ divisions, and n square roots. For $r = 1$, this amounts to $mn^2 + n^2 + mn + \mathcal{O}(n)$ operations in total.

IV. Application to the Riccati recursion

As a relevant application of Cholesky factorization updates, we consider the solution of the structured Newton systems that arise when applying an augmented Lagrangian method (ALM) such as QPALM-OCP to a linear-quadratic optimal control problem (OCP) with inequality constraints, as described in [9]. We reuse the OCP definition from [9, Eq. 1], and define the Jacobians of the dynamics $F_j \triangleq \begin{pmatrix} B_j & A_j \end{pmatrix}$, and of the inequality constraints $G_j \triangleq \begin{pmatrix} D_j & C_j \end{pmatrix}$ and $G_N \triangleq C_N$. The Hessian matrices of the stage cost functions $l_j(u^j, x^j)$ are denoted by $H_j^l \triangleq \begin{pmatrix} R_j & S_j \\ S_j^\top & Q_j \end{pmatrix}$, and $H_N^l \triangleq Q_N$.

In each inner iteration of QPALM-OCP, the Newton system [9, Eq. 16] needs to be solved at the current iterate x. This is equivalent to solving the following equality constrained quadratic program with optimal control structure:

$$\begin{aligned} \underset{\Delta u, \Delta x}{\textbf{minimize}} \quad & \textstyle\sum_{j=0}^{N-1} l_j^{\mathcal{J}(x)}(\Delta u^j, \Delta x^j) + l_N^{\mathcal{J}(x)}(\Delta x^N) \\ \textbf{subject to} \quad & \Delta x^{j+1} = A_j \Delta x^j + B_j \Delta u^j + e^j(x), \\ & \Delta x^0 = x_{\text{init}} - x^0, \end{aligned} \tag{16}$$

where the Newton step is given by the minimizer $(\Delta u, \Delta x)$. The cost functions $l_j^{\mathcal{J}(x)}$ are quadratic with $\nabla^2 l_j^{\mathcal{J}(x)} = H_j^{\mathcal{J}}(x) \triangleq H_j^l + G_j^\top \Sigma_j^{\mathcal{J}(x)} G_j$. [9, Eq. 18]. The diagonal ALM penalty matrices $\Sigma_j^{\mathcal{J}(x)}$ depend on the active set $\mathcal{J}(x)$ [9, Eq. 14] at the current iterate x: entries $(\Sigma^{\mathcal{J}(x)})_{ii}$ are set to a positive penalty parameter if they correspond to an active inequality constraint, and zero otherwise. Assuming that only a small number of constraints enter or leave the active set between successive iterates x and $\tilde{x}$, the matrices $H_j^{\mathcal{J}}$ undergo a low-rank update $H_j^{\mathcal{J}}(\tilde{x}) = H_j^{\mathcal{J}}(x) + G_j^\top(\tilde{\Sigma}_j^{\mathcal{J}} - \Sigma_j^{\mathcal{J}})G_j$, where the penalty matrices $\Sigma_j^{\mathcal{J}} \triangleq \Sigma_j^{\mathcal{J}(x)}$ and $\tilde{\Sigma}_j^{\mathcal{J}} \triangleq \Sigma_j^{\mathcal{J}(\tilde{x})}$ differ in a small number of entries only. This motivates using update routines to avoid full factorizations at each iteration.

The original implementation of QPALM-OCP uses a Schur complement method for solving the Newton system, with factorization updates based on [5] and the Sherman–Morrison formula. In the present paper, we instead consider the well-known Riccati recursion for solving problems of the form (16). We make use of the factorized variant proposed in [4]: the factorization procedure is given in Algorithm 3, with some ALM-specific modifications. Using this factorization, the Newton step can be recovered as described in [4, Alg. 2].

We now present Algorithm 4, a novel method for efficiently updating the Cholesky factors computed by Algorithm 3: Given the factors $L_{uu,j}, L_{xu,j}, L_{xx,j}$ of matrices H_j computed by Algorithm 3 for the iterate x with penalties $\Sigma_j^{\mathcal{J}}$, Algorithm 4 obtains the updated factors $\tilde{L}_{uu,j}, \tilde{L}_{xu,j}, \tilde{L}_{xx,j}$ that can be used to compute the Newton step at a new iterate $\tilde{x}$ with penalties $\tilde{\Sigma}_j^{\mathcal{J}}$. Imporantly, it does so without actually executing Algorithm 3 at $\tilde{x}$, resulting in a lower run time.

Proof: Denote by $\tilde{H}_j$ the matrices that would be computed on line 4 of Algorithm 3, if applied at $\tilde{x}$ with $\tilde{\Sigma}_j^{\mathcal{J}}$. We will show inductively that Algorithm 4 computes the factors $\begin{pmatrix} \tilde{L}_{uu,j} & \\ \tilde{L}_{xu,j} & \tilde{L}_{xx,j} \end{pmatrix} \triangleq \mathrm{chol}(\tilde{H}_j)$: As the induction hypothesis, we state that the cost-to-go matrices $P_j = L_{xx,j} L_{xx,j}^\top$ undergo a low-rank update between x and $\tilde{x}$: $\tilde{P}_j = \tilde{L}_{xx,j}\tilde{L}_{xx,j}^\top = P_j + \Phi_j \mathcal{S}_j \Phi_j^\top$ for some Φ_j and $\mathcal{S}_j$. At the final stage N, we have $\tilde{P}_N = P_N + G_N^\top(\tilde{\Sigma}_N^{\mathcal{J}} - \Sigma_N^{\mathcal{J}})G_N$, covering the base case with $\Phi_N = G_N^\top$ and $\mathcal{S}_N = \tilde{\Sigma}_N^{\mathcal{J}} - \Sigma_N^{\mathcal{J}}$. This update is readily performed by Algorithm 1 to obtain $\tilde{L}_{xx,N}$. For all previous stages $j < N$, there are two terms in the definition of matrix H_j (Alg. 3, line 4) that may change between x and $\tilde{x}$: the second term due to changes to the penalty matrix $\tilde{\Sigma}_j^{\mathcal{J}}$ in the current stage, and the third term due to changes to the cost-to-go matrix $\tilde{P}_{j+1} = \tilde{L}_{xx,j+1}\tilde{L}_{xx,j+1}^\top$ in the next stage. We have that $\tilde{H}_j = H_j + G_j^\top(\tilde{\Sigma}_j^{\mathcal{J}} - \Sigma_j^{\mathcal{J}})G_j + F_j^\top(\tilde{P}_{j+1} - P_{j+1})F_j$. Using the induction hypothesis $\tilde{P}_{j+1} = P_{j+1} + \Phi_{j+1}\mathcal{S}_{j+1}\Phi_{j+1}^\top$, we have $\tilde{H}_j = H_j + \Upsilon_j \mathcal{S}_j \Upsilon_j^\top$, with $\Upsilon_j \triangleq \begin{pmatrix} F_j^\top \Phi_{j+1} & G_j^\top \end{pmatrix}$ and $\mathcal{S}_j \triangleq \mathrm{blkdiag}(\mathcal{S}_{j+1}, \tilde{\Sigma}_j^{\mathcal{J}} - \Sigma_j^{\mathcal{J}})$. To compute the Cholesky factorization of $\tilde{H}_j$, we rely on the blocked algorithm outlined in Section III-B, as written out in detail on lines 5 to 7 of Algorithm 4. Finally, line 7 implies that $\tilde{P}_j = \tilde{L}_{xx,j}\tilde{L}_{xx,j}^\top = L_{xx,j}L_{xx,j}^\top + \Phi_j \mathcal{S}_j \Phi_j^\top = P_j + \Phi_j \mathcal{S}_j \Phi_j^\top$, completing the proof. ∎

Algorithm 3: Riccati factorization [4, Alg. 3]

In : $\{H_j^l\}_{j=0}^{N-1}, Q_N$ (Hessians of OCP cost), $\{F_j\}_{j=0}^{N-1}, \{G_j\}_{j=0}^{N}$ (dynamics & constraint Jacobians), $\{\Sigma_j^{\mathcal{J}}\}_{j=0}^{N}$ (ALM penalty factors).

Out : $\{L_{uu,j}\}_{j=0}^{N-1}, \{L_{xu,j}\}_{j=0}^{N-1}, \{L_{xx,j}\}_{j=0}^{N}$ (Cholesky factors).

1 $P_N \leftarrow Q_N + G_N^\top \Sigma_N^{\mathcal{J}} G_N$
2 $L_{xx,N} \leftarrow \mathrm{chol}(P_N)$
3 **for** $j = N-1, N-2, \ldots 0$
4 $\quad H_j \leftarrow H_j^l + G_j^\top \Sigma_j^{\mathcal{J}} G_j + F_j^\top L_{xx,j+1} L_{xx,j+1}^\top F_j$
5 $\quad \begin{pmatrix} L_{uu,j} & \\ L_{xu,j} & L_{xx,j} \end{pmatrix} \leftarrow \mathrm{chol}(H_j)$

V. Performance of optimized updating routines

The blocked algorithm from Section III-B has been implemented in C++, with highly optimized and vectorized microkernels for Algorithms 1 and 2. The source code has been made available on https://github.com/kul-optec/hyhound. Figure 1 compares the run times of this implementation to those of a full factorization without low-rank updates, for various ranks and using different block sizes. [3]

For small modification ranks m, the hyperbolic Householder factorization updates are over four times faster than a full factorization. For $m > 2$, the blocked variants ($r = 4, 8$) are significantly faster than the non-blocked variant ($r = 1$). This motivates the use of blocked hyperbolic Householder transformations over the column-wise methods from [5].

[3] Experiments were carried out on an Intel Core i7-11700 at 2.5 GHz (fixed), using v2025.0 of the Intel MKL.

Algorithm 4: Riccati factorization updates

In : $\{F_j\}_{j=0}^{N-1}, \{G_j\}_{j=0}^{N}$ (dynamics & constraint Jacobians), $\{\Sigma_j^{\mathcal{J}}\}_{j=0}^{N}, \{\tilde{\Sigma}_j^{\mathcal{J}}\}_{j=0}^{N}$ (old and new penalty factors), $\{L_{uu,j}\}_{j=0}^{N-1}, \{L_{xu,j}\}_{j=0}^{N-1}, \{L_{xx,j}\}_{j=0}^{N}$ (old Cholesky factors).

Out : $\{\tilde{L}_{uu,j}\}_{j=0}^{N-1}, \{\tilde{L}_{xu,j}\}_{j=0}^{N-1}, \{\tilde{L}_{xx,j}\}_{j=0}^{N}$ (updated Cholesky factors).

$\Phi_N \leftarrow G_N^\top, \quad S_N \leftarrow \tilde{\Sigma}_N^{\mathcal{J}} - \Sigma_N^{\mathcal{J}}$

$\begin{pmatrix} \tilde{L}_{xx,N} & 0 \end{pmatrix} \leftarrow \begin{pmatrix} L_{xx,N} & \Phi_N \end{pmatrix} \breve{Q}_{x,N}$ using Alg. 1, where $\breve{Q}_{x,N}$ is $\begin{pmatrix} \mathrm{I} & \\ & S_N \end{pmatrix}$-orthogonal

for $j = N-1, N-2, \ldots 0$

  $\begin{pmatrix} \Upsilon_{u,j} \\ \Upsilon_{x,j} \end{pmatrix} \leftarrow \begin{pmatrix} F_j^\top \Phi_{j+1} & G_j^\top \end{pmatrix}, \quad S_j \leftarrow \begin{pmatrix} S_{j+1} & \\ & \tilde{\Sigma}_j^{\mathcal{J}} - \Sigma_j^{\mathcal{J}} \end{pmatrix}$

  $\begin{pmatrix} \tilde{L}_{uu,j} & 0 \end{pmatrix} \leftarrow \begin{pmatrix} L_{uu,j} & \Upsilon_{u,j} \end{pmatrix} \breve{Q}_{u,j}$ using Alg. 1, where $\breve{Q}_{u,j}$ is $\begin{pmatrix} \mathrm{I} & \\ & S_j \end{pmatrix}$-orthogonal

  $\begin{pmatrix} \tilde{L}_{xu,j} & \Phi_j \end{pmatrix} \leftarrow \begin{pmatrix} L_{xu,j} & \Upsilon_{x,j} \end{pmatrix} \breve{Q}_{u,j}$ using Alg. 2

  $\begin{pmatrix} \tilde{L}_{xx,j} & 0 \end{pmatrix} \leftarrow \begin{pmatrix} L_{xx,j} & \Phi_j \end{pmatrix} \breve{Q}_{x,j}$ using Alg. 1, where $\breve{Q}_{x,j}$ is $\begin{pmatrix} \mathrm{I} & \\ & S_j \end{pmatrix}$-orthogonal

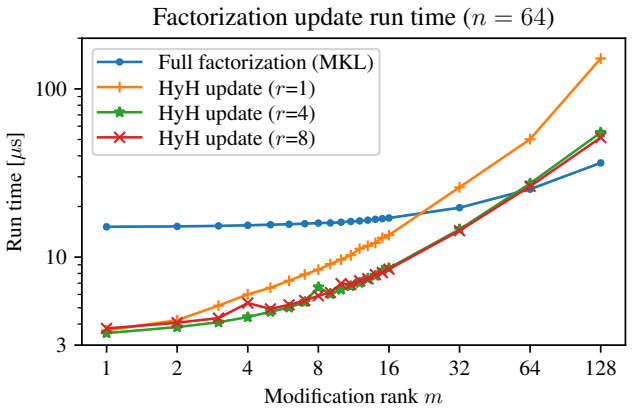


Fig. 1: Absolute run time of different factorization update methods, for $L \in \mathbb{R}^{64\times64}, A \in \mathbb{R}^{64\times m}$. *Full factorization* refers to adding $A\Sigma A^\top$ to the previous matrix H (`dsyrk`) and performing a fresh Cholesky factorization (`dpotrf`). *HyH update* refers to the blocked hyperbolic Householder-based algorithm from §III-B, for various block sizes r.

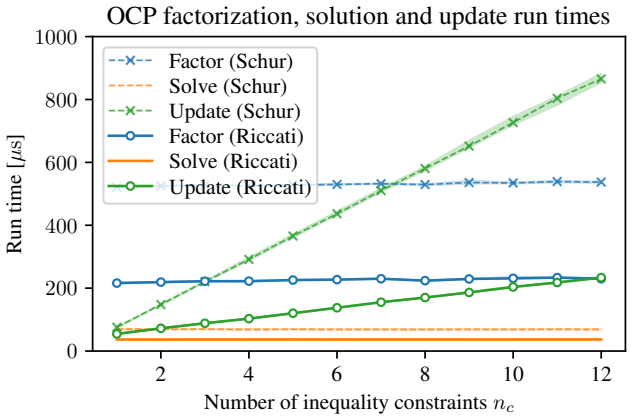


Fig. 2: Run time comparison of the individual factorization, solution, and factorization update steps when solving (16), for the Schur complement method versus the Riccati recursion from §IV. OCP dimensions: $N = 24, n_x = 24, n_u = 8$.

Finally, we compare the performance of the Schur complement method from [9] and the Riccati recursion algorithms presented in Section IV when applied to the semismooth Newton system (16) of QPALM-OCP. The system is solved for randomly generated benchmark OCPs with varying numbers of inequality constraints n_c. As a benchmark, the penalty factors of 25% of the inequality constraints (selected at random) are changed between x and $\tilde{x}$, resulting in larger modification ranks as the number of constraints increases.

Figure 2 demonstrates that the Riccati-based solver significantly outperforms the Schur complement approach. In particular, when combined with the proposed factorization update procedure, it provides an efficient method for solving successive Newton systems with an optimal control structure.

References

[1] D. S. Bernstein, *Matrix mathematics: theory, facts, and formulas*, 2nd ed. Princeton, N.J: Princeton university press, 2009.

[2] A. W. Bojanczyk and A. O. Steinhardt, "Stability Analysis of a Householder-Based Algorithm for Downdating the Cholesky Factorization," *SIAM Journal on Scientific and Statistical Computing*, vol. 12, no. 6, pp. 1255–1265, Nov. 1991.

[3] J. Frey, S. Di Cairano, and R. Quirynen, "Active-Set based Inexact Interior Point QP Solver for Model Predictive Control," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6522–6528, 2020.

[4] G. Frison and J. B. Jørgensen, "Efficient implementation of the Riccati recursion for solving linear-quadratic control problems," in *2013 IEEE International Conference on Control Applications (CCA)*, Aug. 2013, pp. 1117–1122.

[5] P. E. Gill, G. H. Golub, W. Murray, and M. A. Saunders, "Methods for modifying matrix factorizations," *Mathematics of Computation*, vol. 28, no. 126, pp. 505–535, 1974.

[6] B. Hermans, A. Themelis, and P. Patrinos, "QPALM: a proximal augmented lagrangian method for nonconvex quadratic programs," *Mathematical Programming Computation*, vol. 14, no. 3, pp. 497–541, Sept. 2022.

[7] R. A. Horn and C. R. Johnson, *Matrix analysis*, 2nd ed. Cambridge, New York: Cambridge University Press, 2012.

[8] C. Kirches, H. G. Bock, J. P. Schlöder, and S. Sager, "A factorization with update procedures for a KKT matrix arising in direct optimal control," *Mathematical Programming Computation*, vol. 3, no. 4, pp. 319–348, Dec. 2011.

[9] K. F. Lowenstein, D. Bernardini, and P. Patrinos, "QPALM-OCP: A Newton-Type Proximal Augmented Lagrangian Solver Tailored for Quadratic Programs Arising in Model Predictive Control," *IEEE Control Systems Letters*, pp. 1–1, 2024.

[10] I. Nielsen and D. Axehill, "Low-Rank Modifications of Riccati Factorizations for Model Predictive Control," *IEEE Transactions on Automatic Control*, vol. 63, no. 3, pp. 872–879, Mar. 2018.

[11] C. Puglisi, "Modification of the Householder Method Based on the Compact WY Representation," *SIAM Journal on Scientific and Statistical Computing*, vol. 13, no. 3, pp. 723–726, May 1992.

[12] C. Rader and A. Steinhardt, "Hyperbolic householder transformations," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, no. 6, pp. 1589–1602, Dec. 1986.

[13] R. Schreiber and C. Van Loan, "A Storage-Efficient WY Representation for Products of Householder Transformations," *SIAM Journal on Scientific and Statistical Computing*, vol. 10, no. 1, pp. 53–57, Jan. 1989.

[14] M. Stewart and G. W. Stewart, "On Hyperbolic Triangularization: Stability and Pivoting," *SIAM Journal on Matrix Analysis and Applications*, vol. 19, no. 4, pp. 847–860, Oct. 1998.

[15] L. N. Trefethen and I. Bau, David, *Numerical Linear Algebra*, ser. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics, Jan. 1997.

[16] R. A. Van De Geijn and F. G. Van Zee, "High-performance up-and-downdating via householder-like transformations," *ACM Trans. Math. Softw.*, vol. 38, no. 1, pp. 4:1–4:17, Dec. 2011.

[17] S. J. Wright, "Applying new optimization algorithms to more predictive control," Argonne National Lab. (ANL), Argonne, IL (United States), Tech. Rep. MCS-P-561-0196; CONF-960178-1, Mar. 1996.