LOGO

Robust Condensing for LPV-MPC with Asymptotically Constant Complexity

Gionata Cimini, Alberto Bemporad, *Fellow, IEEE*

***Abstract*— This paper presents a highly efficient method for robustly eliminating equality constraints in optimization problems arising from Model Predictive Control (MPC). Such a *condensing* operation is often implemented to reduce the number of optimization variables through the direct elimination of state variables by substitution, a procedure which is prone to numerical instability and whose complexity scales linearly with the prediction horizon. We propose a novel blocked QR decomposition algorithm for condensing MPC problems in a numerically robust manner. The method is applicable when the system dynamics, described by either linear or linearized models, are time-invariant over the prediction horizon. The method effectively handles unstable dynamics and exhibits numerical convergence properties that result in an asymptotically *constant* computational complexity regardless of the horizon length. We benchmark our method against state-of-the-art algorithms for robust condensing of dynamic optimization problems, demonstrating significantly superior results. For sufficiently long horizons, the proposed algorithm can even outperform direct state elimination in terms of speed.**

***Index Terms*— Model predictive control, adaptive control, QR decomposition, robust variable elimination, linear parameter-varying systems, embedded optimization.**

I. Introduction

Present days are witnessing an unprecedented interest in the use of embedded Model Predictive Control (MPC) across various industrial fields, including automotive [1], [2], robotics [3], [4], energy conversion [5], and medical devices [6]. At its core, an MPC scheme solves a constrained optimization problem over a finite horizon in real-time at each sample step. The ability to formulate control problems as optimization ones, to easily deal with nonlinear systems and explicitly consider system's constraints, are definitely contributing to this interest [7], [8]. Moreover, modern applications must address control requirements that do not simply translate into stabilizing the system around a set-point, but rather require optimized dynamic operations for which MPC is particularly suited [9], [10]. The formulation and stability theory for MPC has reached a mature stage and substantial ongoing research focuses on deriving efficient algorithms for its implementation [7], [11]. This is especially true for embedded applications, where solving repeatedly an optimization problem is most often challenging due to the combination of high sampling rates and scarce computational resources [12], [13].

G. Cimini is with ODYS S.r.l., Milano, Italy. Email: `gionata.cimini@odys.it`.
A. Bemporad is with the IMT School for Advanced Studies Lucca, Italy. Email: `alberto.bemporad@imtlucca.it`

In industrial applications, the most common MPC formulation involves a quadratic cost function subject to affine inequalities on inputs and states and linear equalities. The latter express the system dynamics, which can result from linearizing a nonlinear model around a nominal trajectory [14]. At each time step, MPC requires solving the resulting Quadratic Programming (QP) problem, or multiple QPs if a Sequential Quadratic Programming (SQP) approach is adopted [15]. Hence, the research on efficient QP solvers is flourishing [16]–[18]. With less equality constraints than optimization variables, the problem can be reformulated in a lower dimensional space where the equalities are removed [19]. This process is often referred to as variable elimination or *condensing*. Solving lower dimensional problems is definitely appealing for computational reasons [20], and recent literature is showing interest also into further reducing the number of optimization variables at the expense of solution accuracy [21], [22].

In contrast with classical Linear Time-Invariant (LTI) MPC, where the model is fixed and the condensing operation can be performed offline, in Linear Parameter-Varying (LPV) MPC the model is allowed to change at each controller execution – due to changes of external parameters or updating the linearization of the model – and kept constant over the prediction horizon; in Linear Time-Varying (LTV) MPC the model can even change at each prediction step, for example when linearizing around a predicted nominal trajectory. In all cases, and especially when performed online, the condensing process is nearly as crucial as the QP solver itself. Indeed, the numerical properties of the condensed QP problem are of utmost importance and highly depend on the condensing method. For example, replacing predicted states with a linear function of the inputs, which we will refer to as *standard condensing*, can lead to numerical instabilities, especially when the linear model is unstable [23]. Alternatives to the elimination of state variables have been investigated, to either preserve some sparsity patterns in the lower dimensional problem [24], [25], or to favor specific structure-exploiting solvers [26]–[28]. In general, numerically robust approaches for condensing are considered cumbersome [19, Chapter 15.3].

Most of the condensing methods mentioned above focus on reducing the overall execution time and often neglect the numerical properties of the reduced QP problem. Additionally, some of those algorithms are only suited for LTI-MPC approaches. In [23], we investigated an approach based on a specialized QR decomposition for the *robust condensing* of LTV formulations. Reducing the dimension of the optimization problem in a robust manner is very important because: $i)$

common methods, particularly standard condensing, easily blow up numerically in the presence of unstable linear models, $ii)$ QP solvers, especially first-order methods, usually require fewer iterations to reach the same accuracy if the QP problem is well conditioned. Numerical weaknesses are amplified in embedded MPC, because the scarce computational resources often impose to work in single precision arithmetic [12]. The work in [23] shows that the additional computational burden due to a QR decomposition can be vastly mitigated with a tailored factorization algorithm.

A. Contribution

Strongly encouraged by the results in [23], in this paper we answer the question on whether it is possible to further decrease the computational cost of condensing in LPV-MPC, exploiting the fact that the linearized state-space matrices are constant in prediction, but still can change at every execution of the controller. Many systems fall under this framework, and the sparse structure with repeated matrices along the horizon opens up to further efficient methods with respect to the more generic LTV-MPC formulation. We develop a novel blocked QR decomposition algorithm for the elimination of equality constraints in the LPV-MPC problem, or, more generically, in problems that exhibit the same structure for the equality constraints.

Although both the approach in [23] and the one presented here provide a QR decomposition, they are profoundly different in their mechanics. Here, not only we exploit the structured sparsity of the matrix to factorize, but also formulate the iterations as reduced-order QR decompositions of the block-columns, therefore obtaining considerably lower execution times. Moreover, the approach in [23] is restricted to MPC problems, while the one proposed here applies to matrices with a more generic structure.

A second contribution of this paper is to demonstrate that the recursive factorization of the block-columns converge to a structure that makes the execution-time *constant* with respect to horizon length, after a certain convergence tolerance is met. This means that not only the method is superior to alternative robust condensing approaches, but it is also appealing when numerical robustness is not the only concern. Indeed, we show that it can be even faster than standard condensing, whose solely benefit is the fast execution speed, making the method particularly appealing for embedded LPV-MPC applications.

The remainder of this paper is organized as follows. We first describe the robust condensing idea in Section II and recall blocked QR decomposition algorithms. We then derive a novel algorithm for structured optimization problems in Section III and investigate its convergence properties. The relevance of the algorithm for LPV-MPC problems is detailed in Section IV. Finally, in Section V we present different numerical benchmarks where our algorithm is compared to state-of-the-art methods, and to more standard approaches.

B. Notation and definitions

Given a matrix $A \in \mathbb{R}^{m\times n}$, made of $q \times t$ blocks and such that $m = qm_q$, $n = tn_t$, we indicate by $A_i \in \mathbb{R}^{m\times n_t}$ the i-th column block of A, and by $A_{i,j} \in \mathbb{R}^{m_q\times n_t}$ the i-th row and j-th column block of A. Given the set $\mathcal{I} = \{i_1, \dots, i_r\} \subset \mathbb{N}$, $r \leq n$, $A_{\mathcal{I}}$ denotes the matrix $\begin{bmatrix} A_{i_1} & \dots & A_{i_r}\end{bmatrix} \in \mathbb{R}^{m\times rn_t}$, obtained by horizontally stacking the block-columns indexed by $\mathcal{I}$. Let $\mathcal{J} \subset \mathbb{N}$ be another finite set, then $A_{\mathcal{I},\mathcal{J}}$ represents the matrix obtained by horizontally and vertically stacking the blocks indexed respectively by $\mathcal{I}$ and $\mathcal{J}$. Given a matrix A, $\|A\|$ indicates any norm, $\|A\|_{\mathrm{F}}$ its Frobenius norm, $\|A\|_{\mathrm{F}}{}^2 = \sum_{i=1}^{n}\sum_{j=1}^{m} A_{i,j}^2$, $\|A\|_\infty = \max|A_{i,j}|$ its ∞-norm, with indexes $i = 1, \dots, m$, $j = 1, \dots, n$ intended here as the matrix entries. The null space of A is denoted by $\ker(A)$, and $\sigma(A)$, $\lambda(A)$ indicate the vectors of, respectively, the singular values and the eigenvalues of A sorted in non-increasing order. The spectral radius of A will be indicated by $\rho(A) = \max_{i=1,\dots,n}|\lambda_i(A)|$, and $\kappa(A)$ is the condition number of A, here defined by the Euclidean norm and hence $\kappa(A) = \sigma_1(A)\sigma_n(A)^{-1}$. Additionally, $n_+(A)$ and $n_-(A)$ represent the positive and negative index of inertia of A. The matrix $\mathbf{0}_{m\times n}$ is the null matrix with m rows and n columns, whereas $\mathbf{0}_n$ is the null matrix with n columns and an appropriate number of rows if $n > 0$, and the empty matrix if $n = 0$. The matrix $I_n \in \mathbb{R}^{n\times n}$ is the identity matrix of dimension n. We denote by $\mathcal{U}(a, b)$, $a, b \in \mathbb{R}$, $b > a$, the uniform distribution of real numbers in the interval $[a, b]$

II. Problem statement

Consider the minimization of an objective function $f : \mathbb{R}^m \to \mathbb{R}$ subject to a set of linear equality constraints:

$$\min_z f(z) \quad \text{subject to } A'z = b \tag{1}$$

where $A \in \mathbb{R}^{m\times n}$ is full column rank, with $m \geq n$. A typical approach to handle (1) is *variable elimination* [29], where a pair of matrices $E \in \mathbb{R}^{m\times n}$ and $Z \in \mathbb{R}^{m\times(m-n)}$ is chosen such that the following properties hold:

$$\det\left(\begin{bmatrix} E & Z\end{bmatrix}\right) \neq 0, \quad A'Z = \mathbf{0} \tag{2}$$

hence the columns of Z span the null-space of A', and $A'E$ is non-singular. Let $z = Z\bar{z} + Ee$ represent the set of solutions to $A'z = \mathbf{0}$, with $e \in \mathbb{R}^n$ and $\bar{z} \in \mathbb{R}^{m-n}$. From (2) we can derive the following parametrization

$$e = (A'E)^{-1}b, \;\; \bar{z} \text{ free.} \tag{3}$$

The unconstrained optimization problem

$$\min_{\bar{z}} f(Z\bar{z} + s)$$

in $m-n$ variables, with $s = E(A'E)^{-1}b$, is equivalent to (1). Clearly, the approach remains valid also if the minimization is subject to additional inequality constraints.

Among the numerous methods to derive E and Z, the computation of an orthogonal basis is preferable for numerical stability [29]. Consider the QR decomposition of A:

$$A = Q\begin{bmatrix} R \\ \mathbf{0}'_{m-n}\end{bmatrix}, \quad Q = \begin{bmatrix} Q_1 & Q_2\end{bmatrix} \tag{4}$$

where Q is an orthogonal matrix, $Q'Q = I_m$, $Q_1 \in \mathbb{R}^{m\times n}$, $Q_2 \in \mathbb{R}^{m\times(m-n)}$, and $R \in \mathbb{R}^{n\times n}$ is upper triangular. As

Algorithm 1 Blocked QR decomposition

Input: Matrix $A \in \mathbb{R}^{m\times n}$ formed by $p \times q$ blocks.

```
1: $n_q = n/q$;
2: $Q \leftarrow I_m$;
3: $R \leftarrow \mathbf{0}_{m\times n}$;
4: for $i = 1,\dots,q$ do
5:     $\hat{Q}^{(i)}\begin{bmatrix} R'_{i,i} & \mathbf{0}'_{(p-i+1)m_q-n_q}\end{bmatrix}' = \begin{bmatrix} A'_{i,i} & \dots & A'_{p,i}\end{bmatrix}'$;
6:     for $j = i+1,\dots,q$ do
7:         $\begin{bmatrix} R_{i,j} \\ A_{i+1,j} \\ \vdots \\ A_{p,j}\end{bmatrix} \leftarrow \hat{Q}^{(i)\prime}\begin{bmatrix} A_{i,j} \\ A_{i+1,j} \\ \dots \\ A_{p,j}\end{bmatrix}$;
8:     end for
9:     $Q \leftarrow Q \underbrace{\begin{bmatrix} I_{(i-1)n_q} & \mathbf{0}'_{(i-1)n_q} \\ \mathbf{0}_{(i-1)n_q} & \hat{Q}^{(i)}\end{bmatrix}}_{Q^{(i)}}$;
10: end for
```

Output: Orthogonal matrix $Q \in \mathbb{R}^{m\times m}$ and upper triangular matrix $R \in \mathbb{R}^{n\times n}$ such that $QR = A$.

shown in [23], selecting $E \equiv Q_1$ and $Z \equiv Q_2$ not only satisfies conditions (2), but also guarantees that the vector $s = Q_1(R^{-1})'b \equiv Q_1 e$ solves the minimum norm problem:

$$s = \arg\min_z \|z\|^2 \qquad \text{subject to } A'z = b \tag{5}$$

The original m-dimensional optimal solution to problem (2) can be reconstructed by computing $z = Q_2\bar{z} + s$.

Numerical robustness comes at a cost. Traditional QR decomposition has a $\mathcal{O}(mn^2)$ complexity, which is more expensive than most of the alternatives, such as LU, Cholesky or interpolative decomposition. Hence deriving algorithms that reduce the execution time of variable elimination for a specific class of problems is of utmost interest. The elements of A below its diagonal can be nullified by several methods, most commonly the Gram-Schmidt procedure, Householder reflections, and Givens rotations [30]. Regardless of the method chosen to introduce zeros, the computations to derive (4) can be reorganized to exploit the underlying structure of the matrices, especially on modern computing architectures. Methods that favor vectorized operations [31] are commonly referred to as *blocked*, while those optimized also for parallel execution [32] are typically described as *tiled*, and most often rely on block structures for operations on the single tiles. Both these algorithmic optimizations, blocking and tiling, tend to require more *flops* (floating-point operations) than standard sequential algorithms, but are essential for enabling matrix-matrix multiplications. Such *level-3* operation is the preferred choice in high-performance computing due to better memory utilization and high computational intensity [33], [34].

We will first review a generic and iterative *blocked QR* decomposition, [30], [35], which will be the backbone of the novel algorithm derived in this paper. We stress that our motivation to study blocked algorithms does not stem from having access to high-performance computing or specialized hardware – a rare exception, especially in embedded applications. Rather, we are driven by the opportunity to exploit the inherent block structure of the problem itself. As we explore in detail in the next sections, this structure offers a path to designing blocked algorithms that are extremely efficient, independent of whether we can perform matrix-matrix operations using dedicated hardware.

The considered blocked QR algorithm is summarized by Algorithm 1. We consider matrix A in (4) as subdivided into $p \times q$ blocks, such that:

$$A = \begin{bmatrix} A_{1,1} & \dots & A_{1,q} \\ \vdots & \ddots & \vdots \\ A_{p,1} & \dots & A_{p,q}\end{bmatrix} \tag{6}$$

with $A_{i,j} \in \mathbb{R}^{m_q\times n_q}$, $\forall i,j$, and we also introduce the full block-column notation $A = \begin{bmatrix} A_1 & A_2 & \dots & A_q\end{bmatrix}$, where $A_i \in \mathbb{R}^{m\times n_q}$. Note that, in order to have a compact block notation, we assume that $(m \mod p) = 0$ and $(n \mod q) = 0$. However, Algorithm 1 can be trivially extended when this condition does not hold, as well as to blocks of non-homogeneous row/column dimensions. At each iteration i, the algorithm triangularize a subset of the i-th block-column, specifically $A_{\mathcal{I}_i,i}$ with $\mathcal{I}_i = \{i,\dots,p\}$, by performing the QR decomposition

$$\hat{Q}^{(i)}\begin{bmatrix} \hat{R}^{(i)\prime} & \mathbf{0}'_{(p-i+1)m_q-n_q}\end{bmatrix}' = A_{\mathcal{I}_i,i} \tag{7}$$

with $\hat{Q}^{(i)} \in \mathbb{R}^{(pm_q-(i-1)n_q)\times(pm_q-(i-1)n_q)}$ and $\hat{R}^{(i)} \in \mathbb{R}^{n_q\times n_q}$. From this point forward, the superscript notation $X^{(i)}$ denotes the matrix X computed at iteration i. The i-th diagonal block of R can be directly assigned in Step 5 since $\hat{R}^{(i)} \equiv R_{i,i}$. From orthogonality of $\hat{Q}^{(i)}$, triangularizing $A_{\mathcal{I}_i,i}$ is equivalent to multiply it by $\hat{Q}^{(i)\prime}$. Therefore, the rest of the unfactorized sub-matrix $A_{\mathcal{I}_i,\mathcal{J}_i}$ with $\mathcal{J}_i = \{i+1,\dots,q\}$ undergoes the same orthogonal transformation, which completes the computation of the i-th block-row of R. The matrix-matrix multiplication $\hat{Q}^{(i)\prime}A_{\mathcal{I}_i,\mathcal{J}_i}$ is performed, block-column wise, in Step 7, since that allows structure exploiting optimizations. In Algorithm 1 we assume that R is allocated in dedicated memory. However, it is also possible to reformulate the steps to construct R on the memory space of A, offering a more concise interpretation of the major algorithm iteration. Indeed, let use define the orthogonal matrix $Q^{(i)} \in \mathbb{R}^{m\times m}$ as

$$Q^{(i)} = \begin{bmatrix} I_{(i-1)n_q} & \mathbf{0}'_{(i-1)n_q} \\ \mathbf{0}_{(i-1)n_q} & \hat{Q}^{(i)}\end{bmatrix}. \tag{8}$$

Then, Algorithm 1 is mathematically equivalent to the iterative update $A \leftarrow Q^{(i)\prime}A$, and the Q in (4) can be computed with right-accumulation $Q = Q^{(1)}Q^{(2)}\dots Q^{(q)}$ as Step 9.

Remark 2.1: The decomposition in Step 5 of Algorithm 1 could be performed through Householder reflections, which can be accumulated in the so-called WY representation, with $W, Y \in \mathbb{R}^{m\times n}$ such that $Q = I_m - WY'$, and Y a unit lower trapezoidal matrix containing the Householder vectors [36]. This would increase the amount of level-3 operations of the algorithm.

III. Efficient Block QR decomposition

We derive an efficient QR decomposition algorithm for a matrix A that exhibits the following specific structure. Consider the matrices $S_x \in \mathbb{R}^{n_q \times n_q}$, $S_y \in \mathbb{R}^{r_q \times n_q}$, $S_z \in \mathbb{R}^{n_q \times n_q}$, and let $m_q = n_q + r_q$. Matrix $A \in \mathbb{R}^{m \times n}$ is partitioned into $q \times q$ blocks of dimension $m_q \times n_q$ such that:

$$A = \begin{bmatrix} S_y & & & 0 & \\ S_z & S_x & & & \\ & S_y & & & \\ & S_z & & & \\ & & \ddots & S_x & \\ & & & S_y & \\ & & & S_z & S_x \\ & 0 & & & S_y \\ & & & & S_z \end{bmatrix} \tag{9}$$

with $m = qm_q$ and $n = qn_q$. We note that, as long as S_y is not the empty matrix ($r_q > 0$), A is not in block Toepliz form, for which results on the asymptotic behaviour of eigenvalues would exist [37]. Matrix A in (9) can be rewritten in the following compact block-column form:

$$A_1 = \begin{bmatrix} S_y' & S_z' & \mathbf{0}'_{m_q(q-1)} \end{bmatrix}' \tag{10a}$$

$$A_i = \begin{bmatrix} \mathbf{0}'_{m_q(i-2)+r_q} & S_x' & S_y' & S_z' & \mathbf{0}'_{m_q(q-i)} \end{bmatrix}' \tag{10b}$$
$$i = 2, \dots, q.$$

This section adopts a generic formulation, but in Section IV we will explain the relevance and implications of (10) for MPC problems.

Consider a generic matrix $\tilde{Y} := \begin{bmatrix} Y' & \mathbf{0}_{n \times r} \end{bmatrix}'$ with $Y \in \mathbb{R}^{m \times n}$, full column rank, and $m \geq n$. Clearly

$$\begin{bmatrix} \Phi & \mathbf{0}'_m \\ \mathbf{0}_m & I_r \end{bmatrix} \begin{bmatrix} R \\ \mathbf{0}'_r \end{bmatrix} = \tilde{Y} \tag{11}$$

is a QR decomposition of $\tilde{Y}$, with $\Phi \in \mathbb{R}^{m \times m}$, $\Phi'\Phi = I$, $R \in \mathbb{R}^{n \times n}$. We can apply (11) to the decomposition of $A_{\mathcal{I}_i, i}$ in Step 5 of Algorithm 1, when the matrix to factorize is in the form of (9). We have that

$$\hat{Q}^{(i)} = \begin{bmatrix} \hat{Q}^{(i)}_{1,1} & \mathbf{0}'_{n_q + ir_q} \\ \mathbf{0}_{n_q + ir_q} & I_{m_q(q-i)} \end{bmatrix} \tag{12}$$

where $\hat{Q}^{(i)}_{1,1} \in \mathbb{R}^{(n_q + ir_q) \times (n_q + ir_q)}$. From now on, we assume $\hat{Q}^{(i)}_{1,1}$ in (12) to be partitioned as

$$\hat{Q}^{(i)}_{1,1} = \begin{bmatrix} T^{(i)} & \overbrace{V_1^{(i)} \quad V_2^{(i)}}^{V^{(i)}} \end{bmatrix} \tag{13}$$

with $T^{(i)} \in \mathbb{R}^{(n_q + ir_q) \times n_q}$, $V_1^{(i)} \in \mathbb{R}^{(n_q + ir_q) \times r_q}$ and $V_2^{(i)} \in \mathbb{R}^{(n_q + ir_q) \times (i-1)r_q}$. We present two lemmas that will be instrumental in proving the main theorem of this section. The first simplifies the outer factorization in Step 5, while the second streamlines the update of the un-factorized matrix in Step 7, specifically when Algorithm 1 is applied to (9).

Lemma 3.1: Consider a full-column rank matrix $X \in \mathbb{R}^{m_x \times n}$, $m_d := m_x - n > 0$, and its QR decomposition

$$\begin{bmatrix} \Phi_1 & \Phi_2 \end{bmatrix} \begin{bmatrix} R_x \\ \mathbf{0}'_{m_d} \end{bmatrix} = X \tag{14}$$

with $R_x \in \mathbb{R}^{n \times n}$, $\Phi_1 \in \mathbb{R}^{m_x \times n}$, $\Phi_2 \in \mathbb{R}^{m_x \times m_d}$, $\Phi'\Phi = I$. Given the matrix $Y := \begin{bmatrix} D' & X' \end{bmatrix}'$ with $D \in \mathbb{R}^{m_d \times n}$, its QR decomposition $Q_y \begin{bmatrix} R_y' & \mathbf{0}_{2m_d} \end{bmatrix}' = Y$, $R_y \in \mathbb{R}^{n \times n}$, $Q_y \in \mathbb{R}^{(m_x + m_d) \times (m_x + m_d)}$, $Q_y' Q_y = I$, can be computed from the reduced-order (m_x instead of $m_x + m_d$) decomposition

$$\begin{bmatrix} \Gamma_{1,1} & \Gamma_{1,2} \\ \Gamma_{2,1} & \Gamma_{2,2} \end{bmatrix} \begin{bmatrix} R_d \\ \mathbf{0}'_{m_d} \end{bmatrix} = \begin{bmatrix} D \\ R_x \end{bmatrix} \tag{15}$$

with $R_d \in \mathbb{R}^{n \times n}$, $\Gamma_{1,1} \in \mathbb{R}^{m_d \times n}$, $\Gamma_{1,2} \in \mathbb{R}^{m_d \times m_d}$, $\Gamma_{2,1} \in \mathbb{R}^{n \times n}$, $\Gamma_{2,2} \in \mathbb{R}^{n \times m_d}$, $\Gamma'\Gamma = I$. Matrices Q_y, R_y can be then reconstructed as:

$$R_y \equiv R_d \tag{16a}$$

$$Q_y = \begin{bmatrix} \Gamma_{1,1} & \Gamma_{1,2} & \mathbf{0}_{m_x \times m_d} \\ \Phi_1 \Gamma_{2,1} & \Phi_1 \Gamma_{2,2} & \Phi_2 \end{bmatrix}. \tag{16b}$$

Proof: See Appendix A. ■

Lemma 3.1 shows that, given X and Y, there exists $Z_x \in \ker(X')$, such that $Z_y := \begin{bmatrix} Z_{y,1} & Z_{y,2} \end{bmatrix} \in \ker(Y')$, $Z_{y_1} \in \mathbb{R}^{m \times (n + m_d)}$ and $Z_{y_2} = \begin{bmatrix} \mathbf{0}_{m_d} & Z_x' \end{bmatrix}'$.

Lemma 3.2: Consider Algorithm 1 applied to a matrix $A \in \mathbb{R}^{m \times n}$ partitioned with q block-columns as in (10). The triangularization of block column A_j, $j = 1, \dots, q-1$ updates exclusively block-column A_{j+1}.

Proof: See Appendix B. ■

As a result of Lemma 3.2, Algorithm 1 applied to (9) sees the for-loop of Step 6 collapsing into a single iteration with $j = i+1$, for $i = 1, \dots, q-1$. The following theorem sets the basis for the derivation of a novel QR decomposition algorithm for matrices A as in (9).

Theorem 3.1: Let $A \in \mathbb{R}^{qm_q \times qn_q}$ be a full-column rank matrix with structure (9), and $Q \begin{bmatrix} R' & \mathbf{0}' \end{bmatrix}' = A$ its QR decomposition. Consider a blocked factorization algorithm, with block-column size n_q, and an orthonormal matrix $\Gamma^{(i)} \in \mathbb{R}^{m_q \times m_q}$, $\Gamma^{(i)\prime}\Gamma^{(i)} = I$, partitioned as follows:

$$\Gamma^{(i)} = \begin{bmatrix} \Gamma^{(i)}_{1,1} & \Gamma^{(i)}_{1,2} \\ \Gamma^{(i)}_{2,1} & \Gamma^{(i)}_{2,2} \end{bmatrix} \tag{17}$$

with $\Gamma^{(i)}_{1,1} \in \mathbb{R}^{r_q \times n_q}$, $\Gamma^{(i)}_{1,2} \in \mathbb{R}^{r_q \times r_q}$, $\Gamma^{(i)}_{2,1} \in \mathbb{R}^{n_q \times n_q}$ and $\Gamma^{(i)}_{2,2} \in \mathbb{R}^{n_q \times r_q}$, $i = 1, \dots, n_q$. Matrices Q and R can be iteratively constructed by performing n_q reduced-order (m_q instead of $m_q(q - i + 1)$) factorizations $\Gamma^{(i)} \begin{bmatrix} \hat{R}^{(i)\prime} & \mathbf{0}_{r_q} \end{bmatrix}' = \tilde{A}^{(i)} \in \mathbb{R}^{m_q \times n_q}$, where:

$$\tilde{A}^{(1)} = \begin{bmatrix} S_y' & S_z' \end{bmatrix}' \tag{18a}$$

$$\tilde{A}^{(i)} = \begin{bmatrix} S_x' \prod_{j=0}^{i-2} \Gamma^{(j)}_{2,1} \Gamma^{(i-1)}_{2,2} & \hat{R}^{(i-1)\prime} \end{bmatrix}', \; i = 2, \dots, n_q \tag{18b}$$

with $\hat{R}^{(i)} \in \mathbb{R}^{n_q \times n_q}$ and $\Gamma^{(0)}_{2,1} \equiv I_{n_q}$.

Proof: From Lemma 3.2 we know that A_{i+1} is the only block-column of the un-factorized A being updated by the triangularization of A_i, at the i-th iteration of Algorithm 1. Considering (10), we can consequently rewrite Step 7 as:

$$R_{i,i+1} \leftarrow T^{(i)\prime}\mathring{S}_x^{(i)} \tag{19a}$$

$$\begin{bmatrix} A_{i+1,i+1} \\ \vdots \\ A_{q,i+1} \end{bmatrix} \leftarrow \begin{bmatrix} V^{(i)\prime}\mathring{S}_x^{(i)} \\ S_y \\ S_z \\ \mathbf{0}'_{m_q(q-i-1)} \end{bmatrix} \tag{19b}$$

with $i = 1, \dots, n_q - 1$, and $\mathring{S}_x^{(i)} := \begin{bmatrix} \mathbf{0}'_{ir_q} & S'_x \end{bmatrix}'$. Let us define $Y^{(i)} \in \mathbb{R}^{(n_q+ir_q)\times n_q}$ as

$$Y^{(i)} = \begin{cases} \begin{bmatrix} S'_y & S'_z \end{bmatrix}' & \text{if } i = 1 \\ \begin{bmatrix} \mathring{S}_x^{(i-1)\prime}V^{(i-1)} & S'_y & S'_z \end{bmatrix}' & \text{if } i = 2, \dots, n_q \end{cases}. \tag{20}$$

Since (19b) is the partition being factorized at iteration $i+1$, it follows that Algorithm 1 operation reduces exclusively to the n_q decompositions:

$$\hat{Q}_{1,1}^{(i)} \begin{bmatrix} \hat{R}^{(i)} \\ \mathbf{0}'_{ir_q} \end{bmatrix} = Y^{(i)}, \quad i = 1, \dots, n_q, \tag{21}$$

where the null blocks of (19b) have been omitted due to (11). We aim to prove, by induction, that $Y^{(i+1)}$ can be equivalently expressed as

$$Y^{(i+1)} = \begin{bmatrix} V_1^{(i)\prime}\mathring{S}_x^{(i)} \\ Y^{(i)} \end{bmatrix}. \tag{22}$$

Consider the definition of $V^{(i)}$ in (13); since $V^{(1)} \equiv V_1^{(1)}$, then (22) clearly holds for $i = 1$. For the induction step, assume $Y^{(i)} = \begin{bmatrix} \mathring{S}_x^{(i-1)\prime}V_1^{(i-1)} & Y^{(i-1)\prime} \end{bmatrix}'$ holds true. We can then apply Lemma 3.1 to $Y^{(i)}$ and obtain $\hat{Q}_{1,1}^{(i)} = \begin{bmatrix} T^{(i)} & V^{(i)} \end{bmatrix}$ and $\hat{R}^{(i)}$ from the decomposition

$$\begin{bmatrix} \Gamma_{1,1}^{(i)} & \Gamma_{1,2}^{(i)} \\ \Gamma_{2,1}^{(i)} & \Gamma_{2,2}^{(i)} \end{bmatrix} \begin{bmatrix} \hat{R}^{(i)} \\ \mathbf{0}'_{r_q} \end{bmatrix} = \begin{bmatrix} V_1^{(i-1)\prime}\mathring{S}_x^{(i-1)} \\ \hat{R}^{(i-1)} \end{bmatrix} \tag{23}$$

such that

$$T^{(i)} = \begin{bmatrix} \Gamma_{1,1}^{(i)\prime} & \Gamma_{2,1}^{(i)\prime}T^{(i-1)\prime} \end{bmatrix}' \tag{24a}$$

$$V_1^{(i)} = \begin{bmatrix} \Gamma_{1,2}^{(i)\prime} & \Gamma_{2,2}^{(i)\prime}T^{(i-1)\prime} \end{bmatrix}' \tag{24b}$$

$$V_2^{(i)} = \begin{bmatrix} \mathbf{0}_{(i-1)r_q\times r_q} & V^{(i-1)\prime} \end{bmatrix}' \tag{24c}$$

with $T^{(0)} \equiv I_{n_q}$ and $V^{(0)}$ the empty matrix. From (24c), $V_2^{(i)\prime}\mathring{S}_x^{(i)} \equiv V^{(i-1)\prime}\mathring{S}_x^{(i-1)}$ holds, which replaced into (20) leads to

$$Y^{(i+1)} = \begin{bmatrix} \begin{bmatrix} V_1^{(i)\prime} \\ V_2^{(i)\prime} \end{bmatrix}' \mathring{S}_x^{(i)} \\ S_y \\ S_z \end{bmatrix} = \begin{bmatrix} V_1^{(i)\prime}\mathring{S}_x^{(i)} \\ V^{(i-1)\prime}\mathring{S}_x^{(i-1)} \\ S_y \\ S_z \end{bmatrix} = \begin{bmatrix} V_1^{(i)\prime}\mathring{S}_x^{(i)} \\ Y^{(i)} \end{bmatrix}, \tag{25}$$

proving that the recursive relation (22) holds for all values of i. From (24b) it follows that

$$V_1^{(i)\prime}\mathring{S}_x^{(i)} = \left(\Gamma_{2,2}^{(i)} \prod_{j=0}^{i-1} \Gamma_{2,1}^{(j)}\right)' S_x \tag{26}$$

and the expression (18) for $\tilde{A}^{(i)}$ is derived by combining (20), (23) and (26). Note that $\tilde{A}^{(i)}$ is guaranteed to be full column rank for all i. For $i = 1$ it follows from the rank and sparsity properties of A, while for $i > 1$ that is guaranteed by the iterative column sub-block $\hat{R}^{(i-1)}$.

It remains to prove that the iterative, reduced-order, QR decomposition

$$\Gamma^{(i)} \begin{bmatrix} \hat{R}^{(i)} \\ \mathbf{0}'_{r_q} \end{bmatrix} = \tilde{A}^{(i)}, \quad i = 1, \dots, n_q \tag{27}$$

can be used to ultimately construct $Q\begin{bmatrix} R' & \mathbf{0}' \end{bmatrix}' = A$. For what concerns R, (27) already computes the block diagonal elements, since $\hat{R}^{(i)} \equiv R_{i,i}$. Moreover, from Lemma 3.2 we know that $R_{i,i+1}$, are the only non-zero off-diagonal blocks, which, considering (24a), can be computed as

$$R_{i,i+1} = T^{(i)\prime}\mathring{S}_x^{(i)} = \left(\prod_{j=1}^{i} \Gamma_{2,1}^{(j)}\right)' S_x, \quad i = 1, \dots, q-1. \tag{28}$$

Matrix Q is built by right-accumulation of $Q^{(i)}$, which exclusively depends on $\hat{Q}_{1,1}^{(i)}$, see (8) and (12). The terms $\hat{Q}_{1,1}^{(i)}$, $i = 1, \dots, q$ are derived from (24). This proves that we can form Q and R through a block-column algorithm that iterates on the fixed dimensional decompositions (27), with $\tilde{A}^{(i)}$ as in (18). ■

The results of Lemma 3.2 and Theorem 3.1 are sufficient to develop an efficient blocked QR demposition algorithm for (9). We refer to this novel algorithm as `QR-AMPC` for the relevance it has in the field of *adaptive* MPC, as explained in Section IV.

A. Numerical convergence of `QR-AMPC`

Before presenting the full `QR-AMPC` algorithm, we investigate the numerical properties of the iterative update (18). First, we introduce a lemma necessary to apply later the monotone convergence theorem to the $\|\hat{R}^{(i)^{-1}}\|_F$ sequence.

Lemma 3.3: Consider matrices $\Psi \in \mathbb{R}^{n\times n}$, invertible, and $W \in \mathbb{R}^{m\times n}$, and the QR decomposition

$$Q \begin{bmatrix} R \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} W \\ \Psi \end{bmatrix} \tag{29}$$

then $\|\Psi^{-1}\|_{\mathrm{F}} \geq \|R^{-1}\|_{\mathrm{F}}$.

Proof: See Appendix C. ■

The following theorem provides the main result of this section. We demonstrate that the sequence of matrices $\tilde{A}^{(i)}$ in (18) converges, independently of the numerical characteristics of S_x, S_y and S_z matrices.

Theorem 3.2: Consider $\tilde{A}^{(i)} \in \mathbb{R}^{m_q\times n_q}$, full column rank matrix with $m_q \geq n_q$ and $r_q := m_q - n_q > 0$, defined by the set of recursive QR decompositions:

$$\Gamma^{(i)} \begin{bmatrix} \hat{R}^{(i)} \\ \mathbf{0}'_{r_q} \end{bmatrix} = \tilde{A}^{(i)} = \begin{bmatrix} \Gamma_p^{(i)} \\ \hat{R}^{(i-1)} \end{bmatrix} \tag{30}$$

where $\Gamma_p^{(i)} := \left(\prod_{j=0}^{i-2}\Gamma_{2,1}^{(j)}\Gamma_{2,2}^{(i-1)}\right)' S_x$, $\Gamma^{(i)}$ partitioned according to (17) and $\tilde{A}^{(0)}$ that can take any value. Then, matrices $\Gamma^{(i)}$ and $\tilde{A}^{(i)}$ enjoy the following convergence property:

$$\lim_{i\to\infty}\Gamma^{(i)} = \begin{bmatrix}\mathbf{0}_{n_q} & I_{r_q}\\ I_{n_q} & \mathbf{0}_{r_q}\end{bmatrix} \tag{31a}$$

$$\lim_{i\to\infty}\tilde{A}^{(i)} = \begin{bmatrix}\mathbf{0}'_{r_q}\\ \bar{R}\end{bmatrix} \tag{31b}$$

with $\bar{R} \in \mathbb{R}^{n_q\times n_q}$ a constant upper triangular matrix.

Proof: Let $X \in \mathbb{R}^{m\times n}$ be a block matrix, partitioned into sub-matrices $X_{i,j} \in \mathbb{R}^{m_i\times n_j}$, the following property holds for any norm:

$$\|X_{i,j}\| \le \|X\| \le \sum_{i_1,j_1}\|X_{i_1,j_1}\|\,, \quad \forall i,\, j \tag{32}$$

which comes from the extension of triangle inequality to block matrices. Analogously to norms induced by inner products, the Frobenius norm is invariant to orthogonal matrices, hence:

$$\|QX\|_{\mathrm{F}}^2 = \mathrm{tr}(QXX'Q') = \mathrm{tr}(Q'QXX') = \|X\|_{\mathrm{F}}^2$$

where the second equality comes from the invariance to cyclic permutations of the trace operation. We can therefore write

$$\left\|\hat{R}^{(i)}\right\|_{\mathrm{F}} = \left\|\begin{bmatrix}\Gamma_p^{(i)}\\ \hat{R}^{(i-1)}\end{bmatrix}\right\|_{\mathrm{F}} \tag{33}$$

from which, by applying (32), we get

$$\|\hat{R}^{(i-1)}\|_{\mathrm{F}} \le \|\hat{R}^{(i)}\|_{\mathrm{F}}\,.$$

We can also apply Lemma 3.3 to (30) and derive

$$\|\hat{R}^{(i-1)^{-1}}\|_{\mathrm{F}} \ge \|\hat{R}^{(i)^{-1}}\|_{\mathrm{F}}\,. \tag{34}$$

Therefore, we have shown that the sequence $\|\hat{R}^{(i)}\|_{\mathrm{F}}$ is non-decreasing and the sequence $\|\hat{R}^{(i)^{-1}}\|_{\mathrm{F}}$ is non-increasing. Moreover, from Theorem 3.1 we know that $\tilde{A}^{(i)}$ is full-column rank, hence $\hat{R}^{(i)}$ is invertible for all i.

Given an invertible matrix $X \in \mathbb{R}^{n\times n}$, we have that

$$\|X\|_{\mathrm{F}}^2 = \sum_{i=1}^{n}\sigma_i^2(X) \tag{35a}$$

$$\sigma_i(X^{-1}) \equiv \sigma_{n-i+1}(X)^{-1}, \quad i = 1,\ldots,n. \tag{35b}$$

From the invertibility of $\hat{R}^{(i)}$ there exists a positive α such that

$$\lim_{i\to\infty}\sigma_{n_q}(\hat{R}^{(i)^{-1}}) = \lim_{i\to\infty}\sigma_1(\hat{R}^{(i)})^{-1} \ge \alpha > 0 \tag{36}$$

applies, and $\lim_{i\to\infty}\sum_{j=1}^{n_q}\sigma_j(\hat{R}^{(i)^{-1}}) \ge n_q\alpha$ consequently holds. From (35b) we can derive that $\lim_{i\to\infty}\|\hat{R}^{(i)^{-1}}\|_{\mathrm{F}} \ge \sqrt{n_q\alpha}$. Hence the sequence of real numbers $\|\hat{R}^{(i)^{-1}}\|_{\mathrm{F}}$ is non-increasing and bounded below by a positive value. From the monotone convergence theorem it converges to the finite and strictly positive limit $\underline{\psi} = \lim_{i\to\infty}\|\hat{R}^{(i)^{-1}}\|_{\mathrm{F}}$. From (36) there exits a $\beta > 0$ such that $\lim_{i\to\infty}\sigma_1(\hat{R}^{(i)}) \le \beta$, and $\lim_{i\to\infty}\sum_{j=1}^{n_q}\sigma_j(\hat{R}^{(i)}) \le n_q\beta$ easily follows. That provides an upper bound for the non-decreasing sequence $\|\hat{R}^{(i)}\|_{\mathrm{F}}$, which in turn converges to the finite limit $\bar{\psi} = \lim_{i\to\infty}\|\hat{R}^{(i)}\|_{\mathrm{F}}$.

By applying (32) to (33) we obtain $\|\hat{R}^{(i)}\|_{\mathrm{F}} \le \|\Gamma_p^{(i)}\|_{\mathrm{F}} + \|\hat{R}^{(i-1)}\|_{\mathrm{F}}$ and since $\lim_{i\to\infty}\|\hat{R}^{(i)}\|_{\mathrm{F}} = \bar{\psi} > 0$ then

$$\lim_{i\to\infty}\|\Gamma_p^{(i)}\|_{\mathrm{F}} = 0 \tag{37}$$

follows. A matrix with 0 norm is the null matrix, hence

$$\lim_{i\to\infty}\tilde{A}^{(i)} = \begin{bmatrix}\mathbf{0}'_{r_q}\\ \hat{R}^{(i-1)}\end{bmatrix}. \tag{38}$$

Given $X = \begin{bmatrix}\mathbf{0}' & Y'\end{bmatrix}'$ with Y a square upper triangular matrix, a valid QR decomposition of X can be directly derived applying the backward identity, namely:

$$Q\begin{bmatrix}R\\ \mathbf{0}\end{bmatrix} = \begin{bmatrix}\mathbf{0} & I\\ I & \mathbf{0}\end{bmatrix}\begin{bmatrix}Y\\ \mathbf{0}\end{bmatrix} = \begin{bmatrix}\mathbf{0}\\ Y\end{bmatrix} = X \tag{39}$$

We can combine (30) with (38) and (39) to prove the existence of an $\bar{R} \in \mathbb{R}^{n_q\times n_q}$ such that $\lim_{i\to\infty}\Gamma^{(i)}$ is the backward identity and (31) holds true. ■

Building on Theorem 3.1, we have established that a sequence of n_q reduced-order factorizations is sufficient to decompose A. Theorem 3.2 further demonstrates that the components associated with the decompositions exhibit numerical convergence. In Section III-B we will detail how to utilize such property to design an early stopping criterion for a computational complexity independent of q.

In the remaining, we aim to link the decay of $\|\Gamma_p^{(i)}\|_{\mathrm{F}}$ to the components of A. First, we provide norm equivalence results for sub-blocks of orthogonal matrices.

Lemma 3.4: Let $Q \in \mathbb{R}^{m\times m}$ be an orthogonal matrix block partitioned as

$$Q = \begin{bmatrix}Q_{1,1} & Q_{1,2}\\ Q_{2,1} & Q_{2,2}\end{bmatrix} \tag{40}$$

with $Q_{1,1} \in \mathbb{R}^{r_q\times m_2}$, $Q_{1,2} \in \mathbb{R}^{r_q\times r_q}$, $Q_{2,1} \in \mathbb{R}^{m_2\times m_2}$ and $Q_{2,2} \in \mathbb{R}^{m_2\times r_q}$. Then $\|Q_{1,1}\|_{\mathrm{F}} = \|Q_{2,2}\|_{\mathrm{F}}$ for any r_q, m_2 such that $m = r_q + m_2$.

Proof: See Appendix D. ■

Since $\Gamma_{2,1}^{(i)} \in \mathbb{R}^{n_q\times n_q}$ is a sub-block of an orthogonal matrix, $\|\Gamma_{2,1}^{(i)}\|_{\mathrm{F}} \le \sqrt{n_q}$ holds for all $i \ge 1$ (recall that $\Gamma_{2,1}^{(0)} \equiv I_{n_q}$). Moreover, the multiplication by a sub-block of an orthogonal matrix does not increase the norms, hence have that

$$\|\prod_{j=0}^{i-2}\Gamma_{2,1}^{(j)}\|_{\mathrm{F}} \le \|\Gamma_{2,1}^{(1)}\|_{\mathrm{F}} \le \sqrt{n_q}. \tag{41}$$

By replacing (41) into the definition of $\Gamma_p^{(i)}$, we obtain:

$$\|\Gamma_p^{(i)}\| \le \sqrt{n_q}\,\|S_x\|_{\mathrm{F}}\,\|\Gamma_{2,2}^{(i-1)}\|_{\mathrm{F}}\,. \tag{42}$$

Since $\Gamma_{1,1}^{(i)} = \Gamma_p^{(i)}R^{(i)^{-1}}$, we use the result of Lemma 3.4, to replace $\Gamma_{2,2}^{(i-1)}$ and get

$$\|\Gamma_p^{(i)}\|_{\mathrm{F}} \le \zeta^{(i)}\,\|\Gamma_p^{(i-1)}\|_{\mathrm{F}} \tag{43a}$$

$$\zeta^{(i)} = \sqrt{n_q}\,\|S_x\|_{\mathrm{F}}\,\|\hat{R}^{(i-1)^{-1}}\|_{\mathrm{F}}\,. \tag{43b}$$

Consider $\|\hat{R}^{(1)^{-1}}\|_{\mathrm{F}}^2 = \sum_{i=1}^{n_q} \sigma_i(\tilde{A}^{(1)})^{-2} \leq n\sigma_{n_q}(\tilde{A}^{(1)})^{-2}$, and the fact that $\|\hat{R}^{(i)^{-1}}\|_{\mathrm{F}}$ has been proven to be non-increasing, we can easily derive

$$\zeta^{(i)} \leq \sqrt{n_q}\,\|S_x\|_{\mathrm{F}}\,\|\hat{R}^{(1)^{-1}}\|_{\mathrm{F}} \leq n_q^{\frac{3}{2}}\sigma_1(S_x)\sigma_{n_q}(\begin{bmatrix} S_y' & S_z'\end{bmatrix}')^{-1}. \tag{44}$$

Given (43a), the bound in (44) can be used to design problem-specific scaling of A in order to guarantee exponential decay of $\|\Gamma_p^{(i)}\|_{\mathrm{F}}$ since block $i = 1$.

B. QR−AMPC algorithm

By exploiting the results of Lemma 3.2 and Theorem 3.1, we derive the novel blocked QR decomposition algorithm QR−AMPC for computing $Q = \begin{bmatrix} E & Z\end{bmatrix}$ and R, of a given A as in (9). In order to formalize the steps of the algorithm, let us introduce a block partitioning of the matrices $T^{(i)}$ and $V_1^{(i)}$ in (24), such that:

$$T^{(i)} = \begin{bmatrix}\Theta_1', \ldots, \Theta_i', \bar{\Theta}'\end{bmatrix}'$$
$$V_1^{(i)} = \begin{bmatrix}\Upsilon_1', \ldots, \Upsilon_i', \bar{\Upsilon}'\end{bmatrix}'$$

with $\Theta_i \in \mathbb{R}^{r_q\times n_q}$, $\bar{\Theta} \in \mathbb{R}^{n_q\times n_q}$, $\Upsilon_i \in \mathbb{R}^{r_q\times r_q}$, $\bar{\Upsilon} \in \mathbb{R}^{n_q\times r_q}$. Matrices E and Z are derived as blocked matrices formed both by $q\times q$ blocks of dimensions respectively $m_q\times n_q$ for E and $m_q\times(m_q-n_q)$ for Z. It is worth noticing that E is a block upper triangular matrix, and Z is a block lower triangular matrix, which favours the utilization and storage of the factorized matrices.

As stated previously, we are interested in QR−AMPC for its relevance in the field of adaptive MPC, although it can be used in any application exhibiting the structure (9). The steps of QR−AMPC are formalized in Algorithm 2.

Remark 3.1: Step 7 of QR−AMPC requires the computation of a full QR decomposition of the full column rank matrix $\tilde{A}^{(i)} \in \mathbb{R}^{m_q\times n_q}$. Both Givens rotations and Householder reflectors can exploit the structured sparsity coming from the lower submatrix being upper triangular. In this work, we have implemented a customized Householder transformation that works on shortened reflectors, whose length depends on the column index of $\tilde{A}^{(i)}$.

Building on the results of Theorem 3.2, QR−AMPC can be equipped with an early stopping criterion, so to compute approximated matrices E, Z and R such that $\begin{bmatrix} E & Z\end{bmatrix}\begin{bmatrix} R' & \mathbf{0}_{m-n}\end{bmatrix}' \sim A$. If $\|\Gamma_p^{(i)}\|_{\mathrm{F}} = 0$, $\Gamma^{(i)}$ is the backward identity (see (39)), which replaced into (24a)-(24b) leads to:

$$T^{(i)} = \begin{bmatrix}\mathbf{0}_{r_q}' & T^{(i-1)\prime}\end{bmatrix}', \quad V_1^{(i)} = \begin{bmatrix} I_{r_q} & \mathbf{0}_{n_q+(i-1)r_q}'\end{bmatrix}'.$$

For a prescribed convergence tolerance $\epsilon_c \geq 0$, we verify the occurrence of numerical convergence by checking

$$\|\Gamma_p^{(i)}\|_{\mathrm{F}} \equiv \|\bar{\Upsilon}' S_x\|_{\mathrm{F}} \leq \epsilon_c. \tag{45}$$

Suppose that condition (45) is checked at the end of each main for-loop in QR−AMPC, and let $i_c = i$ be the current index for which (45) holds true. The execution of QR−AMPC can be then halted at the end of step i_c and the remaining blocks of the approximated E, Z and R can be explicitly constructed by simple shifted assignments as follows:

$$R_{j,j} = R_{i_c,i_c} \tag{46a}$$
$$R_{j,j+1} = R_{i_c,i_c+1} \tag{46b}$$
$$E_j = \begin{bmatrix}\mathbf{0}_{(j-i_c)m_q}' & E_{1:i_c,i_c}' & \mathbf{0}_{(q-j)m_q}'\end{bmatrix}' \tag{46c}$$
$$Z_{q-j+1} = \begin{bmatrix}\mathbf{0}_{(q-j)m_q}' & Z_{q-i_c+1:q,i_c}' & \mathbf{0}_{(j-i_c)m_q}'\end{bmatrix}' \tag{46d}$$
$$j = i_c+1, \ldots, q.$$

Algorithm 2 QR−AMPC: efficient block QR decomposition

Input: Matrices S_x, $S_z \in \mathbb{R}^{n_q\times n_q}$ and $S_y \in \mathbb{R}^{r_q\times n_q}$, forming $A \in \mathbb{R}^{m\times n}$ as in Equation (9), with $m_q = n_q + r_q$, $m = qm_q$ and $n = qn_q$.

```
1:  $\tilde{A} \leftarrow \begin{bmatrix} S_y' & S_z'\end{bmatrix}'$;
2:  $R \leftarrow \mathbf{0}_{n\times n}$;
3:  $E \leftarrow \mathbf{0}_{m\times n}$;
4:  $Z \leftarrow \mathbf{0}_{m\times(m-n)}$;
5:  $T \leftarrow I_{r_q}$;
6:  for $i = 1, \ldots, q$ do
7:      $\Gamma\begin{bmatrix} R_{i,i} \\ \mathbf{0}_{r_q}\end{bmatrix} = \tilde{A}$, with $\Gamma$ partitioned as in (17);
8:      $V_1 \leftarrow \begin{bmatrix}\Gamma_{1,2}' & \Gamma_{2,2}'T'\end{bmatrix}' \equiv \begin{bmatrix}\Upsilon_1', \ldots, \Upsilon_i', \bar{\Upsilon}'\end{bmatrix}'$;
9:      $T \leftarrow \begin{bmatrix}\Gamma_{1,1}' & \Gamma_{1,2}'T'\end{bmatrix}' \equiv \begin{bmatrix}\Theta_1', \ldots, \Theta_i', \bar{\Theta}'\end{bmatrix}'$;
10:     for $j = 1, \ldots, i-1$ do
11:         $Z_o \leftarrow Z_{q-i+j+1,\,q-i+2\,:\,q-i+j+1}$;
12:         $E_{j,i} \leftarrow Z_o\begin{bmatrix}\Theta_1' & \ldots & \Theta_j'\end{bmatrix}'$;
13:         $Z_{q-i+j,r} \leftarrow Z_o\begin{bmatrix}\Upsilon_1' & \ldots & \Upsilon_j'\end{bmatrix}'$;
14:     end for
15:     $E_{i,i} \leftarrow \begin{bmatrix}\Theta_i' & \bar{\Theta}'\end{bmatrix}'$;
16:     $Z_{q,q-i+1} \leftarrow \begin{bmatrix}\Upsilon_i' & \bar{\Upsilon}'\end{bmatrix}'$;
17:     if $i < q$ then
18:         $R_{i,i+1} \leftarrow \bar{\Theta}' S_x$;
19:         $\tilde{A} \leftarrow \begin{bmatrix} S_x'\bar{\Upsilon} & R_{i,i}'\end{bmatrix}'$;
20:     end if
21: end for
```

Output: Orthogonal matrix $Q = \begin{bmatrix} E & Z\end{bmatrix}$, with $E \in \mathbb{R}^{m\times n}$ $Z \in \mathbb{R}^{m\times(m-n)}$, and upper triangular matrix $R \in \mathbb{R}^{n\times n}$ such that $ER = A$ and $ZA' = \mathbf{0}$.

Note that, with $\epsilon_c = 0$, we get the exact decomposition, that is QR−AMPC running for q steps. We stress that, despite the approximation induced by an $\epsilon_c > 0$, $\det(\begin{bmatrix} E, Z\end{bmatrix}) \neq 0$ holds by construction from (46). If we neglect the computational cost of the memory copies needed for (46), the numerical convergence of QR−AMPC makes it a constant decomposition algorithm in q after (45) is met.

Example 3.1: To get a preliminary idea of the effectiveness of QR−AMPC, we compare it with respect to the generic blocked QR decomposition (Algorithm 1), to which its mechanics are inspired. The numerical benchmarks consists in factorizing matrices A of different dimensions, but all with the structure described by (9), in order to derive R and $Q = \begin{bmatrix} E & Z\end{bmatrix}$ such that (4) holds. The test bench is a 12th Gen Intel® Core™ i9-12900H @2.50 GHz. The algorithm is

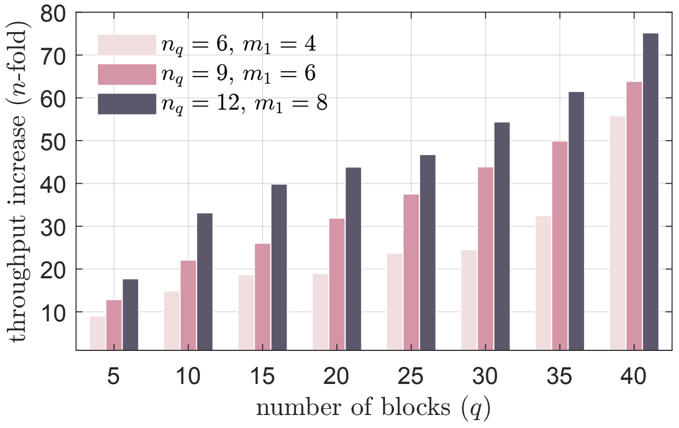

Fig. 1. Throughput increase enabled by QR-AMPC on several $\boldsymbol{A}$ matrices in the form of (9), spanning different block sizes and different number of blocks. The throughput increase (or execution time reduction) is computed as $\boldsymbol{t}$(QR-AMPC)/$\boldsymbol{t}$(Algorithm 1). The convergence tolerance is set to $\boldsymbol{\epsilon_c = 0}$ as we are interested in comparing the exact algorithm.

originally coded in MATLAB®, from which C code is auto-generated and then compiled with -O2 flag. Unless differently specified, this is the setup of all the algorithms utilized in this paper, especially in Section V. Moreover, whenever an execution time measurement is involved, that is acquired as the shifted geometric mean over 100 executions of the same code [38]. We introduce the notation $t(l)$, which stands for the turnaround time (in ms) of the algorithm defined by the label l. In this example we consider three sets of dimensions, $r_q \in \{4, 6, 8\}$ and $n_q = 1.5r_q$, and for each one we record t(QR-AMPC) and t(Algorithm 1) for matrices formed by $q \in \{5j \mid j = 1, \ldots, 8\}$. For this test we are interested only in the exact factorization, therefore we set $\epsilon_c = 0$. Figure 1 presents the results, demonstrating a remarkable reduction in execution time for QR-AMPC, ranging from 10-fold to 80-fold.

Example 3.2: We evaluate the impact of algorithm convergence on decomposition accuracy and execution time. We select a specific problem dimension, $r_q = 6$ and $n_q = 9$, with $q = 40$, and generate a set of 100 random tuples (S_x, S_y, S_z) for each one of the following setups:

- Setup 1: $S_x, S_y, S_z \sim \mathcal{U}(-1, 1)$;
- Setup 2: $S_x \sim \mathcal{U}(-0.1, 0.1)$, $S_y, S_z \sim \mathcal{U}(-1, 1)$;
- Setup 3: $S_x \sim \mathcal{U}(-1, 1)$, $S_y, S_z \sim \mathcal{U}(-10, 10)$.

The objective of the different matrix entries distributions is to evaluate their impact on the numerical convergence. In fact, from (44), one can easily infer that the upper bound directly depends on the norm of S_x, and inversely depends on the norms of S_y and S_z. Each random matrix $A \in \mathbb{R}^{360\times240}$ if factorized with QR-AMPC, whose execution is halted at different steps $i_c < q$, thus applying (46) for $q - i_c$ steps. We collect the reconstruction error

$$\zeta_r(Q, R, A) = \left\| Q \begin{bmatrix} R' & \mathbf{0}_{m-n} \end{bmatrix}' - A \right\|_2 (1 + \|A\|_2)^{-1}$$

and the total execution time, which includes the exact factorization of the first i_c blocks and direct copy of the remaining $q - i_c$ blocks with (46). We recall that QR-AMPC is normally halted according to the tolerance check (45), while for the sake of this test we select directly the step i_c. Figure 2 collects the results, highlighting the convergence rates for the different values assumed by the tuple (S_x, S_y, S_z). For instance, if a reconstruction error of 10^{-8} is deemed acceptable for a given application, the algorithm converges, on average, in 24 steps for “Setup 1”, in 12 steps for “Setup 2”, and in 9 steps for “Setup 3”. The bound for the latter holds actually on the full set of 100 matrices. If convergence is achieved, then for any larger q the matrices are factorized in a *constant execution time*. We show such a time in the bottom part of Figure 2 for different halting steps.

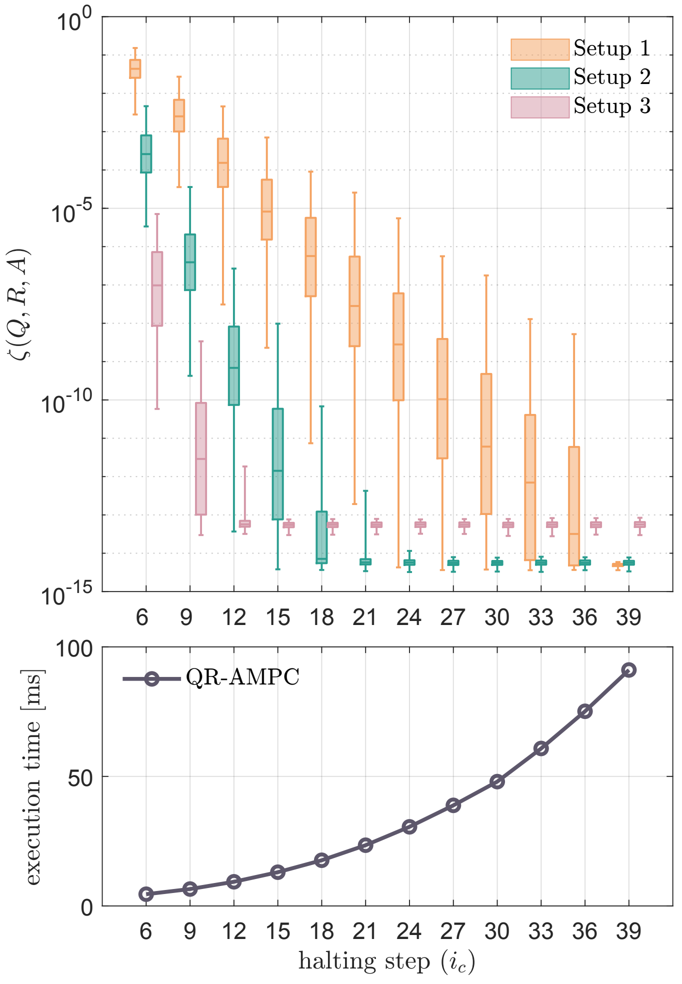

Fig. 2. Reconstruction error $\boldsymbol{\zeta(Q, R, A)}$ and execution time $\boldsymbol{t}$(QR-AMPC) for the factorization of random matrices $\boldsymbol{A}$ with $\boldsymbol{r_q = 6}$, $\boldsymbol{n_q = 9}$, $\boldsymbol{q = 40}$, made by tuples $\boldsymbol{(S_x, S_y, S_z)}$ drawn from different uniform distribution setups. The algorithm is stopped at increasing iteration steps $\boldsymbol{i_c}$ showing the impact of the different setups on the numerical convergence.

IV. Implications for Model Predictive Control

Consider the following Linear Parameter-Varying (LPV) discrete-time state-space model

$$x(k+1) = \mathcal{A}(\theta(k))x(k) + \mathcal{B}(\theta(k))u(k) \tag{47a}$$

$$y(k) = \mathcal{C}(\theta(k))x(k) \tag{47b}$$

where $x \in \mathbb{R}^{n_x}$, $y \in \mathbb{R}^{n_y}$, and $u \in \mathbb{R}^{n_u}$ are the system states, measured outputs, and manipulated inputs, respectively,

$\theta(k) \in \mathbb{R}^{n_\theta}$ is a vector of parameters affecting the dynamics, and k is the sample step. The LPV form (47) is of interest because $i)$ a physical system can naturally exhibit LPV dynamics, $ii)$ those can be the result of linearizing and discretizing a nonlinear model:

$$\frac{\mathrm{d}x_c(t)}{\mathrm{d}t} = f_x(x_c(t), u_c(t)) \tag{48a}$$
$$y_c(t) = f_y(x_c(t)) \tag{48b}$$

at time t around nominal values $\bar{x}_c(t)$, $\bar{u}_c(t)$ that are assumed constant in prediction, with x_c, y_c, u_c the continuous-time counterparts of x, y, u. For MPC design, we consider the following finite-time optimal control problem:

$$\min_{\{u_i, x_{i+1}\}_{i=0}^{p-1}} \quad \sum_{i=0}^{p-1} \ell_i(x_{i+1}, u_i, \theta(k)) \tag{49a}$$
$$\text{s.t.} \quad x_{i+1} = \mathcal{A}(\theta(k))x_i + \mathcal{B}(\theta(k))u_i \tag{49b}$$
$$y_i = \mathcal{C}(\theta(k))x_i \tag{49c}$$
$$u_i \in \mathbb{U}, \ \ y_i \in \mathbb{Y} \tag{49d}$$
$$x_0 = x(k) \tag{49e}$$

where p is the prediction horizon, $\ell_i(x_{i+1}, u_i, \theta(k))$ is any linear or quadratic cost function, $\mathbb{U} \in \mathbb{R}^{n_u}$ and $\mathbb{Y} \in \mathbb{R}^{n_y}$ are polyhedral sets that impose constraints on inputs and outputs respectively. The above LPV-MPC formulation is also referred to as *adaptive* MPC in [39]. Problem (49) has $m = p(n_x + n_u)$ variables, and if we define the optimizer vector as $z = \begin{bmatrix} u_0' & x_1' & \dots & u_{p-1}' & x_p' \end{bmatrix}'$, then (49b) can be rewritten in the compact linear system form $A'z = b$, $A \in \mathbb{R}^{m \times n}$ as in (9), $n = pn_x$, $b \in \mathbb{R}^n$, with:

$$S_x \equiv -\mathcal{A}(\theta(k)), \ S_y \equiv -\mathcal{B}(\theta(k)), \ S_z \equiv I_{n_x} \tag{50a}$$
$$b = \begin{bmatrix} (\mathcal{A}(\theta(k))x_0(k))' & \mathbf{0}'_{(p-1)n_x} \end{bmatrix}'. \tag{50b}$$

Therefore, (49) can be cast into an equivalent optimal control problem in $m - n$ variables by applying the parametrization $z = Z\bar{z} + s$, see (3), with $\bar{z} \in \mathbb{R}^{m-n}$ and E, Z, and R computed with QR-AMPC applied to matrix A defined by (50a). Clearly $n_q \equiv n_x$, $r_q \equiv n_u$ hold, and $q \equiv p$ meaning that the blocking pattern of the QR decomposition corresponds to the prediction horizon.

In Section III we have shown that the block QR decomposition has a constant runtime in q, however the solution $s = E(R^{-1})'b$ to the linear system $A'z = b$ has still to be computed. Here, we show that for an MPC optimization problem also the computation of s converges numerically, leading to an *overall condensing routine that is constant with respect to the prediction horizon.*

Let $s \in \mathbb{R}^m$ be decomposed in p stacked vectors, such that $s_i \in \mathbb{R}^{n_x+n_u}$, $i = 1, \dots, p$. Recall that R consists of $p \times p$ blocks of dimension $n_x \times n_x$ of which only the diagonal and 1st upper off-diagonal blocks are nonzero, and that E is an upper block-triangular matrix consisting of $p \times p$ blocks of dimension $(n_x + n_u) \times n_x$. Let us also introduce $\bar{s} \in \mathbb{R}^{pn_x}$ as the solution of the linear system $R'\bar{s} = b$, partitioned in p stacked vectors, such that $\bar{s}_i \in \mathbb{R}^{n_x}$, $i = 1, \dots, p$ are the constituting blocks. Given the above mentioned sparsity patterns, and b as in (50b), we have that s can be efficiently computed as:

$$R_{1,1}'\bar{s}_1 = \mathcal{A}(\theta(k))x(k) \tag{51a}$$
$$R_{j,j}'\bar{s}_j = -R_{j-1,j}'\bar{s}_{j-1}, \qquad j = 2, \dots, p \tag{51b}$$
$$s_i = \sum_{j=i}^{p} E_{i,j}\bar{s}_j, \qquad i = 1, \dots, p. \tag{51c}$$

Theorem 4.1: Let $A \in \mathbb{R}^{m \times n}$ be a matrix defined by (9) and (50a), $b \in \mathbb{R}^n$ a vector as in (50b), $m = p(n_x + n_u)$, $n = pn_x$ and let $\begin{bmatrix} E & Z \end{bmatrix} \begin{bmatrix} R' & \mathbf{0}' \end{bmatrix}' = A$ be the QR decomposition of A. Assume $\rho(\mathcal{A}(\theta(k))) \leq 1$. Then, the vector $s \in \mathbb{R}^m$, such that $s = E(R^{-1})'b$ with $A's = b$, enjoys the following convergence property:

$$\lim_{i \to \infty} s_i = \mathbf{0}_{n_x+n_u} \tag{52}$$

with $s_i \in \mathbb{R}^{n_x+n_u}$ the i-th slice of s.

Proof: We start by proving that

$$\sigma_1(R_{p,p}) \geq \dots \geq \sigma_1(R_{2,2}) \geq \sigma_1(R_{1,1}) > 1. \tag{53}$$

Consider QR-AMPC steps. We know that $\tilde{A}^{(1)} = \begin{bmatrix} S_y' & S_z' \end{bmatrix}' = \begin{bmatrix} \mathcal{B}(\theta(k))' & I_{n_x} \end{bmatrix}'$. If we impose $\Psi \triangleq I_{n_x}$ and $W \triangleq \mathcal{B}(\theta(k))$ in (29), we can follow considerations analogous to those used in the proof of Lemma 3.3, in particular the ones to get to (76). Under the assumption that $\mathcal{B}(\theta(k)) \neq \mathbf{0}$ we get

$$\sigma_1(R_{1,1}) > \sigma_1(I_{n_x}) = 1. \tag{54}$$

For the steps $j = 2, \dots, p$, since $\tilde{A}^{(j)}$ contains the sub-block $R_{j-1,j-1}$, see (30), applying again (76) leads to

$$\sigma_1(R_{j,j}) \geq \sigma_1(R_{j-1,j-1}). \tag{55}$$

Combining (54) and (55) proves (53). Since for an invertible matrix $X \in \mathbb{R}^{n \times n}$, $\sigma_i(X^{-1}) \equiv \sigma_{n-i+1}(X)^{-1}$ holds for all i, it can be shown that

$$\sigma_1(R_{p,p}^{-1}) \leq \dots \leq \sigma_1(R_{2,2}^{-1}) \leq \sigma_1(R_{1,1}^{-1}) < 1. \tag{56}$$

Now recall that $R_{i,i+1} = \prod_{j=i}^{1} \Gamma_{2,1}^{(j)} S_x$, with $i = 1, \dots, p-1$. Clearly, $\sigma_1(\Gamma_{2,1}^{(j)}) \leq 1$ holds for a sub-block of a unitary matrix, and hence $\rho(R_{i,i+1}) \leq \rho(S_x)$. In the context of MPC, we impose $S_x \triangleq \mathcal{A}(\theta(k))$ and thus, given $\rho(\mathcal{A}(\theta(k))) \leq 1$, we get

$$\rho\left((R_{j,j}^{-1})'R_{j-1,j}\right) < 1, \quad j = 2, \dots, p \tag{57}$$

which, from a simple manipulation of the linear system (51b), proves that $\lim_{j \to \infty} \bar{s}_j = \mathbf{0}_{n_x}$. The slice s_i is a linear function of $\bar{s}_j$, with $j \geq i$, see (51c), therefore $\lim_{j \to \infty} s_j = \mathbf{0}_{n_x}$ directly follows. ■

In Theorem 4.1, the (marginal) stability of the linear system, $\rho(\mathcal{A}(\theta(k))) \leq 1$, is only a sufficient condition for s_i to approach 0. Indeed, condition (57) very likely holds, starting from a certain j, even in presence of unstable systems. From (56) and (28), respectively, we can derive that

$$\sigma_1(R_{i,i}^{-1}) < 1, \qquad i = 1, \dots, p \tag{58a}$$
$$\sigma_1(R_{j,j+1}) \leq \sigma_1(R_{j-1,j}), \qquad j = 2, \dots, p. \tag{58b}$$

Hence, the spectral norm of both matrices involved in (57) is non-increasing, one of which strictly bounded in the unit circle. Moreover, given $X = \begin{bmatrix} D' & Y' \end{bmatrix}'$, $\sigma_1(X) \geq \max(\sigma_1(D), \sigma_1(Y))$ holds,

$$\sigma_1(R_{1,2}) \leq \sigma_1(\Gamma^{(1)}_{2,1})\sigma_1(\mathcal{A}(\theta(k))) \tag{59a}$$
$$\sigma_1(R_{2,2})^{-1} \leq \max(\sigma_1(\Gamma^{(1)}_{2,2}\mathcal{A}(\theta(k))), \sigma_1(R_{1,1}))^{-1} \tag{59b}$$

which means that a large eigenvalue in $\mathcal{A}(\theta(k))$ does enlarge the upper bound on the $R_{i,i+1}$ sequence, but it also shrinks the upper bound on the $(R^{-1}_{i,i})'$ sequence. Despite being likely, s_i approaching 0 is not guaranteed in case of an unstable system. For instance, a large tolerance ϵ_c can cause QR-AMPC to halt before properties (58) and (59) have a sufficient impact on (57). Therefore, when solving for (51) we propose the adoption of a second tolerance, $\epsilon_s \geq 0$, used to halt the p iterations in (51) by checking $\|s_i\|_\infty \leq \epsilon_s$. Similarly to i_c, let i_s be the index such that $\|s_{i_s+j}\|_\infty \leq \epsilon_s$, $j = 0, \dots, p - i_s$ holds.

A. QR-AMPC *implementation*

If the LPV formulation (47) originates from the linearization of a nonlinear model, at step k one could linearize (48) around $x(k)$ and get $b = \mathbf{0}$ as a consequence of $x_0 = \mathbf{0}$ in (50b). In this case the linear system $A'z = b$ would admit the trivial solution $s = \mathbf{0}$. Hence, not only the computation of $s = E(R^{-1})'b$ would be completely avoided, but also QR-AMPC could be modified to get a computationally cheaper and less memory-demanding implementation that exclusively computes matrix Z. Clearly, the selection of ϵ_s would be unnecessary in this case.

An additional crucial consideration regards the approximation introduced by a possible early stopping of the method due to numerical convergence. By construction, the column blocks of A' are increasingly ordered from the current to the last prediction step. Therefore the steps that are affected by the approximation are the ones that are furthest in the future, on which there is obviously more uncertainty.

We finally stress that QR-AMPC requires only the state dynamics to stay constant over the prediction horizon. The output equation can indeed vary and be, for instance, the result of successive linearizations at different nominal points. We could therefore replace (49c) with $y_i = \mathcal{C}_i x_i$, $i = 0, \dots, p-1$, despite that being a less frequent, but more generic, formulation.

B. *QP formulation*

We consider the following widely-used MPC formulation based on a quadratic objective function. Let $W^u_i \in \mathbb{R}^{n_u \times n_u}$, $W^\delta_i \in \mathbb{R}^{n_u \times n_u}$, and $W^y_i \in \mathbb{R}^{n_y \times n_y}$ be some weight matrices on inputs, input rates, and outputs, respectively. At step k, the i-th stage cost, $i = 0, \dots, p-1$, is

$$\begin{aligned}\ell_i(x_{k+i}, u_i, \theta(k)) = \|W^u_i(u_i - \bar{u}_i)\|_2^2 + \|W^\delta_i(u_i - u_{i-1})\|_2^2 + \\ + \|W^y_i(\mathcal{C}(\theta(k))x_{i+1} - \bar{y}_i)\|_2^2\end{aligned} \tag{60}$$

with $u_{-1} = u(k-1)$ and $\begin{bmatrix}\bar{u}_0, \dots \bar{u}_{p-1}\end{bmatrix}$, $\begin{bmatrix}\bar{y}_0, \dots \bar{y}_{p-1}\end{bmatrix}$ some references trajectories on inputs and states. The optimization problem (49) with the quadratic cost (60) can be cast into the following QP problem:

$$\min_z \quad \frac{1}{2}z'Hz + h'z \tag{61a}$$
$$\text{s.t.} \quad Az = b \tag{61b}$$
$$Gz \leq g \tag{61c}$$

with $H \in \mathbb{R}^{m \times m}$ the symmetric and positive semi-definite Hessian matrix, $h \in \mathbb{R}^m$ the linear cost term, $G \in \mathbb{R}^{m_c \times m}$ and $g \in \mathbb{R}^{m_c}$ the matrices collecting all the m_c constraints imposed by $\mathbb{U}$ and $\mathbb{Y}$ along the horizon. Under the parametrization $z = Z\bar{z} + s$ introduced in (3), (61) turns into the following reduced dimensional QP in $m_r = m - n$ variables

$$\min_{\bar{z}} \ \frac{1}{2}\bar{z}'H_r\bar{z} + h_r'\bar{z}, \quad \text{s.t. } G_r\bar{z} \leq g_r \tag{62}$$

where $H_r \in \mathbb{R}^{m_r \times m_r}$, $h_r \in \mathbb{R}^{m_r}$, $G_r \in \mathbb{R}^{m \times m_r}$ and $g_r \in \mathbb{R}^{m_r}$ are computed as:

$$H_r = Z'HZ \tag{63a}$$
$$h_r = 2H's + h \tag{63b}$$
$$G_r = GZ \tag{63c}$$
$$g_r = Gs + g. \tag{63d}$$

In (61)–(63), we have dropped the dependence from k for simplicity. In the context of MPC, although somewhat improperly, the non-condensed formulation (61) is often addressed as *sparse*, referring in particular to the sparsity patterns of the Hessian and constraint matrices, whereas (62) as *dense*.

C. *Standard condensing*

Here we give a brief overview about standard condensing, the most common approach when it comes to reducing the number of variables in an MPC optimization problem. We are going to use it for comparison in Section V. The idea is to replace the predicted states x_k with the explicit solution formula

$$x_k = \mathcal{A}(\theta(k))^k x_0 + \sum_{i=0}^{k-1} \mathcal{A}(\theta(k))^i \mathcal{B}(\theta(k)) u_{k-i}, \tag{64}$$

hence obtaining an optimization problem where $\bar{z}$ corresponds to the vector of predicted inputs.

Equation (64) imposes the coordinate transformation $z = Z\bar{z} + s$ with

$$Z = \begin{bmatrix} I_{n_u} & 0 & \dots & \dots & 0 \\ \mathcal{B} & 0 & \dots & \dots & 0 \\ 0 & I_{n_u} & 0 & \dots & 0 \\ \mathcal{AB} & \mathcal{B} & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & 0 & I_{n_u} \\ \mathcal{A}^{p-1}\mathcal{B} & \mathcal{A}^{p-2}\mathcal{B} & \dots & \mathcal{AB} & \mathcal{B} \end{bmatrix}, \ s = \begin{bmatrix} 0 \\ \mathcal{A} \\ 0 \\ \mathcal{A}^2 \\ \vdots \\ 0 \\ \mathcal{A}^p \end{bmatrix} x_0 \tag{65}$$

where we have dropped the dependence from $\theta(k)$ for simplicity. Condensing with (65) easily blows up numerically in case there exists any $\lambda_i(\mathcal{A}(\theta(k))) > 1$, $i = 1, \dots, n_x$.

V. Numerical Results

We benchmark the proposed method against state-of-the-art alternatives. First, we propose a comparison with respect to the most efficient robust condensing method available in the literature. Then, we show that the proposed method can be a valid replacement for standard condensing even when numerical robustness is not a concern. Finally, we test the algorithm in a typical MPC application in combination with different well-known QP solvers.

A. Comparison with robust condensing methods

When numerical robustness *is* a concern, QR decompositions are the preferred way for condensing (49). In [23] we proposed a very efficient, non-blocked, QR decomposition algorithm (named QR-MPC) specifically tailored for the sparsity pattern of matrix (9). We showed increased robustness with respect to pre-stabilization, and reduced computational burden (up to an order of magnitude) with respect to structure-unaware QR decomposition. Differently from what we present here, that algorithm assumes that the state dynamics can change over the horizon, namely $x_{i+1} = \mathcal{A}_i x_i + \mathcal{B}_i u_i$, $i = 0, \ldots, p-1$, as a result, for instance, of successive linearization along a nominal trajectory. Hence, QR-AMPC is less versatile than QR-MPC in terms of problem formulation, but it is much more efficient for the class of problems (49). Both algorithms provide a QR decomposition of A'. However, the two algorithms largely differ for the adopted blocked strategy and recursive mechanics of the reduced-order QR decomposition, as well as their convergence properties.

The comparison setup is close to the one of Example 3.1. We consider a set of matrices A to be factorized, formed by randomly generated $\mathcal{A}$ and $\mathcal{B}$ matrices with $n_u \in \{4, 6, 8\}$, $n_x = 1.5 n_u$ and $p \in \{5j \,|\, j = 1, \ldots, 8\}$. The results are shown is Figure 3. Here we are interested in showing a comparison about the *exact* factorization routines, hence we set $\epsilon_c = 0$ for QR-AMPC. Note that the numerical accuracy of the two methods is equivalent within machine precision, with a reconstruction error of A that is lower than 10^{-13} in both cases. The benchmark shows that QR-AMPC is anyway a much better alternative than the state-of-the-art QR-MPC when we are dealing with LPV systems. In fact even the exact decomposition leads to a significant throughput improvement, which widens as the prediction horizon increases. With $p = 40$ there is already an improvement of an order of magnitude.

B. Comparison with standard condensing

Standard and robust condensing clearly target different needs in the context of MPC problems. The former is computationally very fast and easy to code. The latter is considered a necessary execution overhead to address possible numerical instabilities. However, the convergence properties of QR-AMPC discovered in this work can be a game changer in case of sufficiently long horizons, making robust condensing faster than the standard one. We show this by comparing equality elimination with QR-AMPC, considering two different tolerances $\epsilon_c \equiv \epsilon_s \in \{10^{-5}, 10^{-8}\}$, and with standard

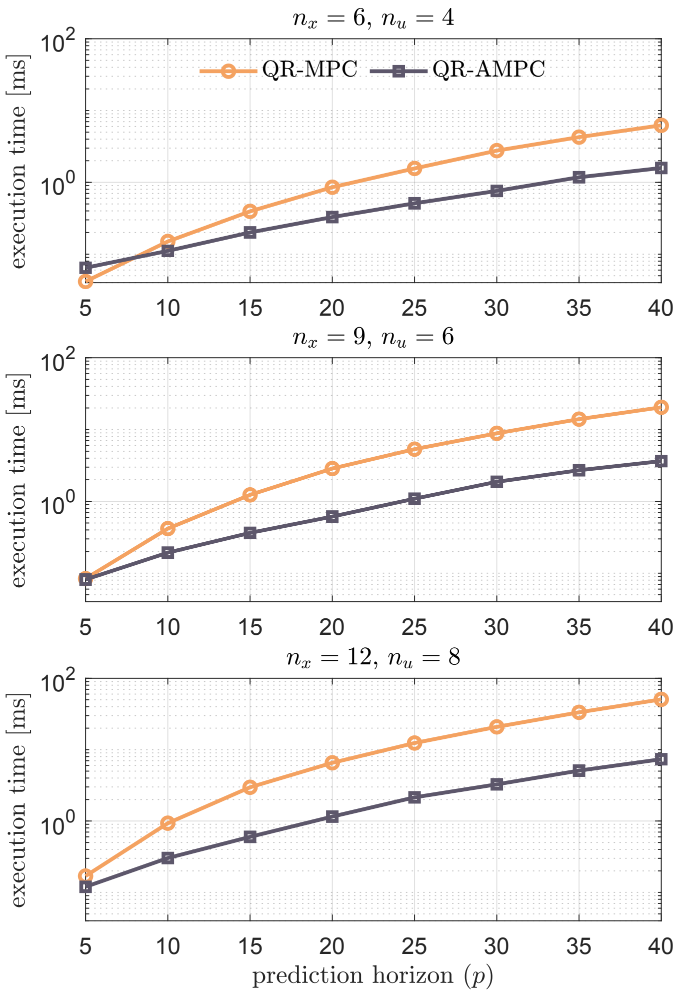

Fig. 3. Throughput comparison for QR-MPC and QR-AMPC for several $\boldsymbol{A}$ matrices with structure (9), constructed from randomly generated $\boldsymbol{\mathcal{A}(\theta(k))}$ and $\boldsymbol{\mathcal{B}(\theta(k))}$ matrices. We set $\boldsymbol{\epsilon_c} = \mathbf{0}$ for QR-AMPC so to compare the two exact factorization methods.

condensing in Section IV-C. The two approaches are compared in terms of execution time required to fully form the elements of the parametrization (3), on 100 problems constructed from random (stable) $\mathcal{A}(\theta(k))$, $\mathcal{B}(\theta(k)) \in \mathcal{U}(-100, 100)$ and $\mathcal{C}(\theta(k)) = I_{n_x}$ with $n_x = 9$, $n_u = 6$. We set $p \in \{10, 20, 40, 60, 80, 100\}$, and evaluate the KKT residual for the different values of ϵ_c when solving QPs with cost function (60) defined by $W_i^y = 10 I_{n_x}$, $W_i^u = 0.1 I_{n_u}$ and $W_i^\delta = \mathbf{0}$, $i = 0, \ldots, p-1$. Figure 4 collects the results. The execution time, which includes exclusively the time to form (62), shows that at $p = 60$, QR-AMPC with the lower accuracy is already faster than standard condensing, and at $p = 100$ the throughput gap is on average ~ 4 and ~ 21 times, respectively for the two tolerances. The execution time is a distribution for QR-AMPC and a single average value for standard condensing because it is influenced by the distributions of convergence indexes i_c and i_s, which are shown in the bottom plot.

Remark 5.1: The benchmark of this section is based on the time acquisition of the full implementation of QR-AMPC, and also excludes the time needed to actually solve the QP problem. As discussed in IV-A some throughput can be saved

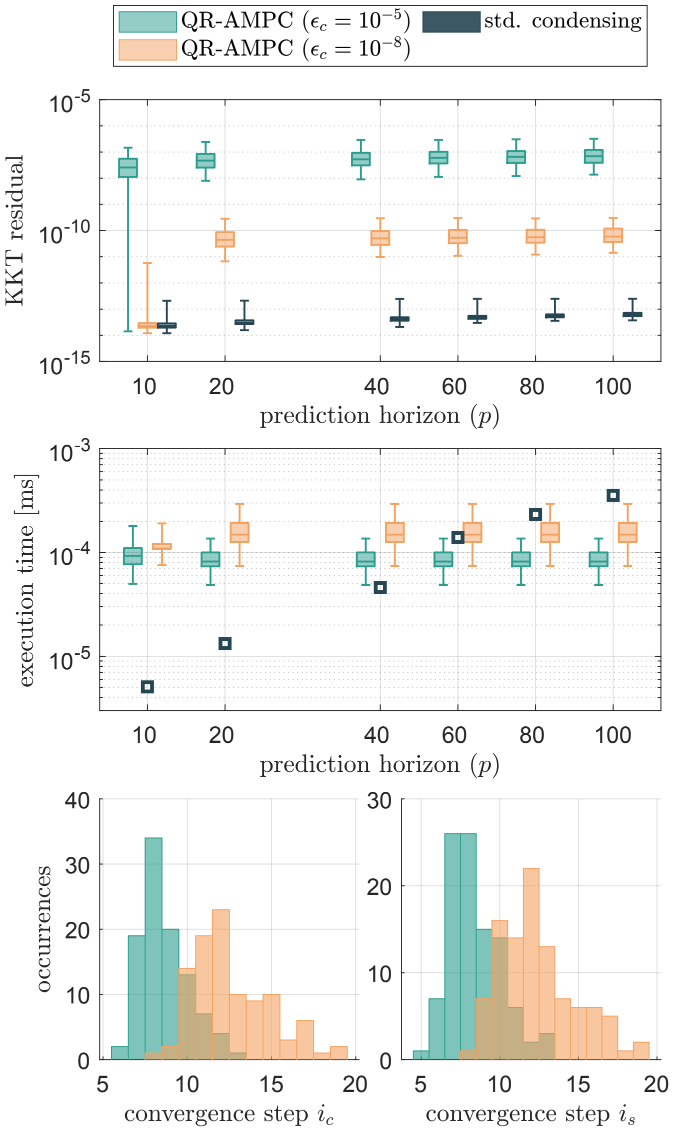

Fig. 4. Comparison between standard condensing and QR-AMPC with two convergence tolerances. Top plot shows the optimality error, mid plot the execution time of condensing, and bottom row the distribution of i_c and i_s on the 100 problems solved.

(in QR-AMPC) with a specific selection of linearization points. Moreover, given that the reduced QPs originating from robust condensing are better conditioned by construction, the time to solve the QP could be sensibly less than with standard condensing, especially with first order methods. For instance, in [23], the authors compare solution quality metrics for QP problems solved using the Alternating Direction Method of Multipliers (ADMM), applied to both standard and robust (QR) condensing. With a fixed number of iterations, the optimizer error under robust condensing improves by over an order of magnitude. Consequently, the crossing point in the execution time between the two methods, when considering the overall optimization routine, may occur at shorter prediction horizons in practice.

TABLE I
PARAMETERS OF CSTR MODEL IN EQUATIONS (66)

Parameter	Value	Unit
F	1	m^3/h
V	1	m^3
R	1.985875	$\mathrm{kcal}/(\mathrm{kmol}\cdot\mathrm{K})$
ΔH	-5960	kcal/kmol
E	11843	kcal/kmol
k_0	34393800	1/h
ρC_p	500	$\mathrm{kcal}/(\mathrm{m}^3\cdot\mathrm{K})$
UA	150	$\mathrm{kcal}/(\mathrm{K}\cdot\mathrm{h})$

C. Application to an MPC problem

The continuous stirred tank reactor (CSTR) [40, p. 563] is a classical benchmark problem for MPC [39], and it is a perfect candidate to test the performance of a robust condensing method. Indeed, it exhibits strongly nonlinear dynamics with respect to one of the states, and the linearized dynamics can be open-loop unstable during transitions between different operating conditions.

The system has $n_x = 2$ states, the concentration of the reactant C_A [$\mathrm{kmol/m^3}$] and the temperature inside the reactor T_r [K], and $n_u = 1$ input which is the jacket coolant temperature T_c [K]. The inlet feed stream temperature T_f [K], and the reactant concentration in it, C_{Af} [$\mathrm{kmol/m^3}$], are treated as measured disturbances. By defining $T_f^-(t) \triangleq T_f(t) - T_r(t)$ and $T_c^-(t) \triangleq T_r(t) - T_c(t)$, the nonlinear model derived from mass balance and energy conservation principles is given by

$$\frac{\mathrm{d}C_A(t)}{\mathrm{d}t} = \frac{F}{V}(C_{Af}(t) - C_A(t)) - r(t) \tag{66a}$$
$$\frac{\mathrm{d}T_r(t)}{\mathrm{d}t} = \frac{F}{V}T_f^-(t) - \frac{\Delta H}{\rho C_p}r(t) - \frac{UA}{\rho C_p V}T_c^-(t) \tag{66b}$$

with $r(t) = k_0 e^{\frac{-E}{RT(t)}} C_A(t)$, E the activation energy, R the Boltzmann ideal gas constant, k_0 a non-thermal factor, A the coolant interface area, ΔH, C_p and ρ the coefficients for heat per mole of the reaction, heat capacity, and density. The value of the parameters, and their units, are collected in Table I. Further details about the plant model can be found in [39], [40].

The control goal is to manipulate T_c so to regulate C_A to a desired set-point $\bar{C}_A$. The prediction horizon is set to $p = 20$, and the set-point $\bar{C}_A$ is previewed for 5 steps, namely $\bar{C}_{A,k+j}$, $j = 1,\dots,5$ is known at step k. The nonlinear dynamics are linearized around $u(k-1)$ and $x(k)$ with a sample time $T_s = 0.5\,\mathrm{h}$, and discretized by first-order Euler approximation. We assume that the measured disturbances are constant, $C_{Af} = 10\,\mathrm{kmol/m^3}$ and $T_f = 298.15\,\mathrm{K}$. States and inputs under the regulation of a robustly condensed MPC are shown in Figure 5. The reactant concentration correctly follows the reference trajectory, and the coolant temperature complies with constraints imposed on its value and increments, which are respectively:

$$285.15 \le T_{c,k+i} \le 312.15, \tag{67a}$$
$$-2 \le T_{c,k+i} - T_{c,k+i-1} \le 2 \tag{67b}$$

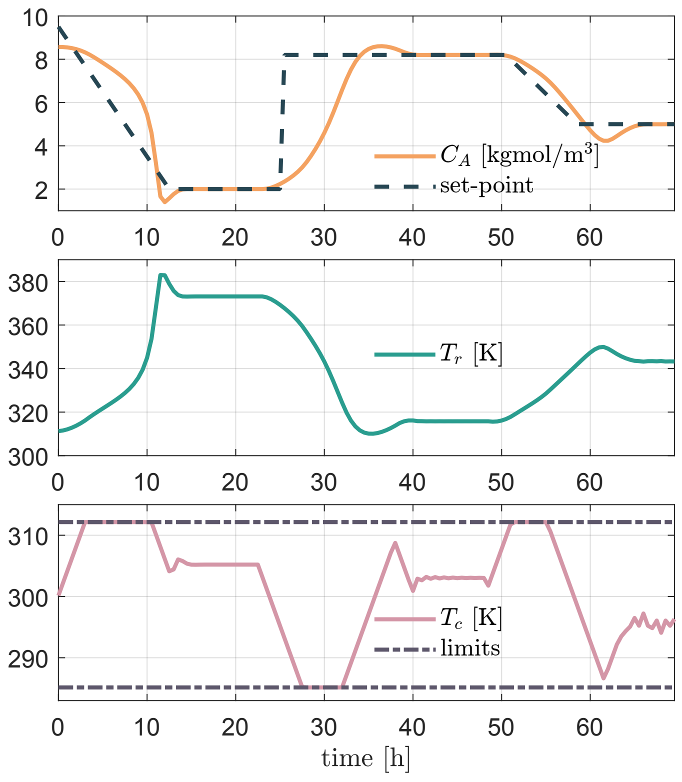

Fig. 5. Closed-loop behaviour of CSTR system under robustly condensed LPV-MPC operation. Top plot shows the desired set-point tracking for reactant concentration, mid plot the reactor temperature evolution, and bottom plot the manipulated coolant temperature, subject to constraints (67)

with $i = 0, \dots, p-1$ and $T_{c,k-1} = T_c(k-1)$. Hereafter, we test different solvers and formulations, but the differences in term of control performance are not noticeable and therefore the results in Figure 5 are assumed to be equivalent for all the flavours of QR-AMPC application that will follow.

First, we consider standard condensing. The CSTR model can be open-loop unstable and Figure 6 shows, on the left, the evolution of the eigenvalues of $\mathcal{A}(k)$ for the control scenario of Figure 5. The unstable behaviour is clearly excited during the simulation, and the right plot shows the corresponding $\kappa(H_r)$, that is the condition number of the reduced Hessian in (62) obtained with both robust (QR) and standard condensing. The latter blows up numerically in the presence of unstable eigenvalues, making the robust approach mandatory. Even when the linearized dynamics are stable, $\kappa(H_r)$ is more than two orders of magnitude larger, which may have negative impacts on the convergence and accuracy of the QP algorithm used.

Lastly, we want to analyze whether condensing is a valid option for MPC in the first place. In fact, one could solve directly (61), and the issue of numerical blow-up caused by equalities elimination would be highly mitigated. Moreover, (61) is much sparser than (62), and many efficient sparse QP solvers exist [17], [41]. Yet, at the same time, very efficient dense solvers are available [16], [42] and the typical small dimension of problems that arise in MPC usually makes it more convenient to work in a reduced number of variables

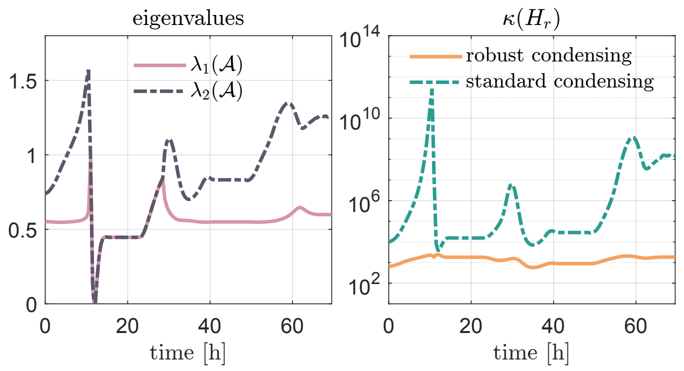

Fig. 6. Evolution of the eigenvalues of $\boldsymbol{\mathcal{A}}(\boldsymbol{k})$ on the left, clearly crossing into the unstable region for prolonged time. On the right the evolution of $\boldsymbol{\kappa}(\boldsymbol{H_r})$, where $\boldsymbol{H_r}$ is computed from (63) with $\boldsymbol{Z}$ obtained with both robust and standard condensing. The latter blows numerically in correspondence of unstable eigenvalues.

rather than exploiting the problem sparsity. That is especially true in embedded MPC, where dimensions are smaller and libraries or hardware accelerations that are necessary for an efficient sparse approach are not available. We show this by considering two well known and open-source QP solvers, osqp [41] and qpOASES [42], and the commercial solver ODYS QP [16]. The former has been specifically developed for sparse formulations, while the latter two are designed for dense problems. All solvers run on the simulation scenario of Figure 5, with default settings, once solving QPs in the sparse form (61) and once in the dense form (62), which is condensed by means of QR-AMPC with $\epsilon_c = 10^{-6}$. Figure 7 shows the KKT residual on the original problem and the execution time of the six configurations, with the dense versions obviously including both the time to condense and to actually solve the QP problem. The KKT residual is small enough to make the closed-loop results indistinguishable from each other, confirming that, as far as numerical robustness is of concern, robust condensing or sparse formulation are both valid approaches. The throughput comparison is far more interesting. Consider qpOASES first. As expected, the impact of working on a reduced dimensional QP is huge as the solver is not able to exploit the sparse structure of (61). Condensing, robustly, make the whole QP solution routine faster by ~ 25.9 times on average, and ~ 49.5 times in the worst-case. Also ODYS QP is positively affected by condensing, with an average and worst-case speed up of ~ 6.2 and ~ 12.2 times, respectively. Less expected are results regarding osqp. Contrary to the prevailing narrative that sees the non-condensed formulation compulsory if the QP solver handles well sparse problems, here we show that an efficient condensing can overturn the outcome. Even if obviously less impacted, also osqp benefits from the approach presented in this paper. The execution time is faster by ~ 1.8 times on average, and ~ 6.6 times in the worst-case, despite the solver being specifically designed for sparse QPs. We finally note that ODYS QP stands out for its execution speed and accuracy when addressing both sparse and dense formulations, with an average and worst-case throughput gain of ~ 14 and ~ 50 times if compared to the best competitor.

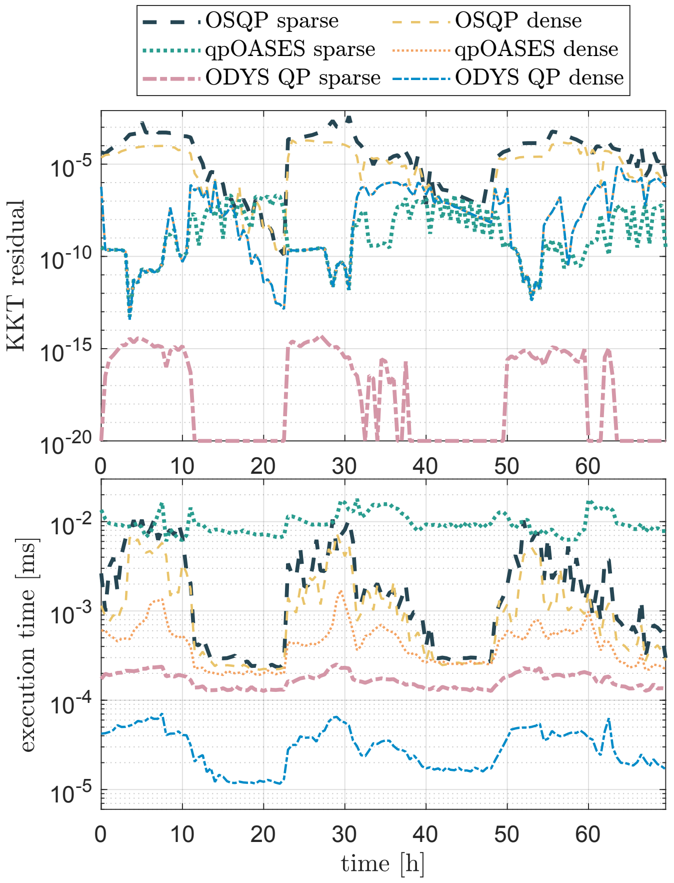

Fig. 7. Optimality error and execution time of `osqp`, `qpOASES` and `ODYS QP` solver running on the simulation scenario of Figure 5, and solving both sparse and dense equivalent QP formulations. The dense setup is condensed with `QR-AMPC` with $\boldsymbol{\epsilon_c = 10^{-6}}$ and the execution time includes both the time to condense the QP and to actually solve it. The solvers are running with default settings.

VI. Conclusions

This paper has addressed the topic of *condensing* optimization problems that are subject to equality constraints in a numerically robust, yet efficient, manner. We specifically targeted the structure of problems that arise from LPV-MPC, where the linearized matrices change at each time-step, but remain constant in prediction. The importance of numerical robustness is not to be overlooked. The most common alternative, that is standard condensing through states elimination, may easily blow up numerically even if the linearized system is slightly unstable and the prediction horizon is long enough. For such LPV-MPC formulations, we have derived a novel algorithm based on blocked QR decomposition which is shown to outperform the execution time of state-of-the-art robust condensing methods. The topic is of great interest for embedded MPC, where the throughput decrease that commonly accompanies a robust approach can easily become a key limitation. The proposed method is not only intrinsically efficient, but also exhibits numerical convergence properties that we have fully investigated and exploited. For sufficiently long horizons, the algorithm can be early stopped and the parametrization explicitly constructed. The result is a condensing routine that is not only robust, but even more efficient than the widely adopted state elimination. Finally, we have shown that QP solvers specifically developed for either sparse or dense formulations benefit from the proposed algorithm, increasing their average and worst-case throughput.

References

[1] W. Liu, M. Hua, Z. Deng, Z. Meng, Y. Huang, C. Hu, S. Song, L. Gao, C. Liu, B. Shuai, A. Khajepour, L. Xiong, and X. Xia, “A systematic survey of control techniques and applications in connected and automated vehicles,” *IEEE Internet of Things Journal*, vol. 10, no. 24, pp. 21 892–21 916, 2023.
[2] A. Norouzi, H. Heidarifar, H. Borhan, M. Shahbakhti, and C. R. Koch, “Integrating machine learning and model predictive control for automotive applications: A review and future directions,” *Engineering Applications of Artificial Intelligence*, vol. 120, p. 105878, 2023.
[3] M. M. Sotaro Katayama and Y. Tazaki, “Model predictive control of legged and humanoid robots: models and algorithms,” *Advanced Robotics*, vol. 37, no. 5, pp. 298–315, 2023.
[4] A. Saviolo, J. Frey, A. Rathod, M. Diehl, and G. Loianno, “Active learning of discrete-time dynamics for uncertainty-aware model predictive control,” *IEEE Transactions on Robotics*, vol. 40, pp. 1273–1291, 2024.
[5] I. Harbi, J. Rodriguez, E. Liegmann, H. Makhamreh, M. L. Heldwein, M. Novak, M. Rossi, M. Abdelrahem, M. Trabelsi, M. Ahmed, P. Karamanakos, S. Xu, T. Dragičević, and R. Kennel, “Model-predictive control of multilevel inverters: Challenges, recent advances, and trends,” *IEEE Transactions on Power Electronics*, vol. 38, no. 9, pp. 10 845–10 868, 2023.
[6] A. Chakrabarty, E. Healey, D. Shi, S. Zavitsanou, F. J. Doyle, and E. Dassau, “Embedded model predictive control for a wearable artificial pancreas,” *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2600–2607, 2020.
[7] D. Q. Mayne, “Model predictive control: Recent developments and future promise,” *Automatica*, vol. 50, no. 12, pp. 2967–2986, 2014.
[8] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017.
[9] J. Köhler, M. A. Müller, and F. Allgöwer, “Analysis and design of model predictive control frameworks for dynamic operation—An overview,” *Annual Reviews in Control*, vol. 57, p. 100929, 2024.
[10] G. Cimini, M. Gatti, D. Bernardini, A. Bemporad, C. Audas, and C.-G. Dussap, “Towards supervisory model predictive control for circular life support systems in long-term space missions,” *Life Sciences in Space Research*, vol. 47, pp. 105–123, 2025.
[11] M. Schwenzer, M. Ay, T. Bergs, and D. Abel, “Review on model predictive control: an engineering perspective,” *The International Journal of Advanced Manufacturing Technology*, vol. 117, no. 5, pp. 1327–1349, Nov 2021.
[12] G. Cimini, D. Bernardini, S. Levijoki, and A. Bemporad, “Embedded model predictive control with certified real-time optimization for synchronous motors,” *IEEE Transactions on Control Systems Technology*, vol. 29, no. 2, pp. 893–900, 2021.
[13] G. Cimini and A. Bemporad, “Exact complexity certification of active-set methods for quadratic programming,” *IEEE Transactions on Automatic Control*, vol. 62, no. 12, pp. 6094–6109, 2017.
[14] D. Mayne, J. Rawlings, and M. Diehl, *Model Predictive Control: Theory and Design*, 2nd ed. Madison,WI: Nob Hill Publishing, LCC, 2018.
[15] D. Kouzoupis, G. Frison, A. Zanelli, and M. Diehl, “Recent advances in quadratic programming algorithms for nonlinear model predictive control,” *Vietnam Journal of Mathematics*, vol. 46, no. 4, pp. 863–882, Dec 2018.
[16] G. Cimini, A. Bemporad, and D. Bernardini, “ODYS QP Solver,” ODYS S.r.l. (https://odys.it/qp), 2017.
[17] R. Schwan, Y. Jiang, D. Kuhn, and C. N. Jones, “PIQP: A Proximal Interior-Point Quadratic Programming Solver,” in *2023 62nd IEEE Conference on Decision and Control (CDC)*, 2023, pp. 1088–1093.
[18] 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, Sep 2022.
[19] J. Nocedal and S. Wright, *Numerical Optimization*, ser. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006.

[20] M. Muehlebach and R. D'Andrea, "A Method for Reducing the Complexity of Model Predictive Control in Robotics Applications," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2516–2523, 2019.

[21] R. Schurig, A. Himmel, and R. Findeisen, "Geometric Data-Driven Dimensionality Reduction in MPC with Guarantees," 2024.

[22] G. Pan and T. Faulwasser, "NMPC in active subspaces: Dimensionality reduction with recursive feasibility guarantees," *Automatica*, vol. 147, p. 110708, 2023.

[23] A. Bemporad and G. Cimini, "Variable Elimination in Model Predictive Control Based on K-SVD and QR Factorization," *IEEE Transactions on Automatic Control*, vol. 68, no. 2, pp. 782–797, 2023.

[24] T. Dang, K. Ling, and J. Maciejowski, "Banded null basis and ADMM for embedded MPC," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 13 170–13 175, 2017.

[25] J. Jerez, E. Kerrigan, and G. Constantinides, "A sparse and condensed QP formulation for predictive control of LTI systems," *Automatica*, vol. 48, no. 5, pp. 999 – 1002, 2012.

[26] D. Kouzoupis, R. Quirynen, J. Frasch, and M. Diehl, "Block condensing for fast nonlinear mpc with the dual newton strategy," *IFAC-PapersOnLine*, vol. 48, no. 23, pp. 26–31, 2015, 5th IFAC Conference on Nonlinear Model Predictive Control NMPC 2015.

[27] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. v. Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "acados—a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, vol. 14, no. 1, pp. 147–183, Mar 2022.

[28] C. Kirches, H. G. Bock, J. P. Schlöder, and S. Sager, "Complementary condensing for the direct multiple shooting method," in *Modeling, Simulation and Optimization of Complex Processes: Proceedings of the Fourth International Conference on High Performance Scientific Computing, March 2-6, 2009, Hanoi, Vietnam.* Springer, 2012, pp. 195–206.

[29] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York, NY, USA: Springer, 2006.

[30] C. v. L. G. H. Golub, *Matrix Computations*, 4th ed. JHU Press, 2013.

[31] J. L. Barlow, "Block Modified Gram–Schmidt Algorithms and Their Analysis," *SIAM Journal on Matrix Analysis and Applications*, vol. 40, no. 4, pp. 1257–1290, 2019.

[32] M. R. Apriansyah and R. Yokota, "Parallel QR Factorization of Block Low-rank Matrices," *ACM Transactions on Mathematical Software*, vol. 48, no. 3, p. 1–28, Sep. 2022.

[33] J. J. Dongarra, J. Du Croz, S. Hammarling, and I. S. Duff, "A set of level 3 basic linear algebra subprograms," *ACM Trans. Math. Softw.*, vol. 16, no. 1, p. 1–17, mar 1990.

[34] L. S. Blackford, A. Petitet, R. Pozo, K. Remington, R. C. Whaley, J. Demmel, J. Dongarra, I. Duff, S. Hammarling, G. Henry *et al.*, "An updated set of basic linear algebra subprograms (BLAS)," *ACM Transactions on Mathematical Software*, vol. 28, no. 2, pp. 135–151, 2002.

[35] W. Jalby and B. Philippe, "Stability Analysis and Improvement of the Block Gram–Schmidt Algorithm," *SIAM Journal on Scientific and Statistical Computing*, vol. 12, no. 5, pp. 1058–1073, 1991.

[36] 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, 1989.

[37] R. A. Davis, K.-S. Lii, and D. N. Politis, *Asymptotic Distribution of Eigenvalues of Block Toeplitz Matrices*. New York, NY: Springer New York, 2011, pp. 163–164.

[38] P. J. Fleming and J. J. Wallace, "How not to lie with statistics: the correct way to summarize benchmark results," *Commun. ACM*, vol. 29, no. 3, p. 218–221, Mar. 1986.

[39] A. Bemporad, M. Morari, and N. Ricker, *Model Predictive Control Toolbox for MATLAB*. The Mathworks, Inc., 2020, http://www.mathworks.com/access/helpdesk/help/toolbox/mpc/.

[40] B. Bequette, *Process Dynamics: Modeling, Analysis and Simulation*. Prentice-Hall, 1998.

[41] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: an operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020.

[42] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[43] Y. Wang and S. Zheng, "The converse of weyl's eigenvalue inequality," *Advances in Applied Mathematics*, vol. 109, pp. 65–73, 2019.

APPENDIX

A. Proof of Lemma 3.1

If we consider the QR decomposition (14), Y can be expressed as follows:

$$Y = \begin{bmatrix} D \\ X \end{bmatrix} = \underbrace{\begin{bmatrix} I_{m_d} & \mathbf{0}_n & \mathbf{0}_{m_d} \\ \mathbf{0}'_{m_x} & \Phi_1 & \Phi_2 \end{bmatrix}}_{\tilde{\Phi}} \underbrace{\begin{bmatrix} D \\ R_x \\ \mathbf{0}'_{m_d} \end{bmatrix}}_{\tilde{Y}}. \tag{68}$$

Since X is full-column rank, so is $\tilde{Y} \in \mathbb{R}^{(m_x+m_d)\times n}$, and $\tilde{\Phi} \in \mathbb{R}^{(m_x+m_d)\times(m_x+m_d)}$ is orthogonal by construction. The QR factorization of $\tilde{Y}$ can be derived by combining (11) and (15):

$$\underbrace{\begin{bmatrix} \Gamma_{1,1} & \Gamma_{1,2} & \mathbf{0}'_{m_d} \\ \Gamma_{2,1} & \Gamma_{2,2} & \mathbf{0}'_n \\ \mathbf{0}_n & \mathbf{0}_{m_d} & I_{m_d} \end{bmatrix}}_{\tilde{Q}} \begin{bmatrix} R_d \\ \mathbf{0}'_{m_d} \\ \mathbf{0}'_{m_d} \end{bmatrix} = \tilde{Y}. \tag{69}$$

By replacing (69) into (68) we get:

$$Q_y \begin{bmatrix} R_y \\ \mathbf{0}'_{2m_d} \end{bmatrix} = \tilde{\Phi}\tilde{Q} \begin{bmatrix} R_d \\ \mathbf{0}'_{2m_d} \end{bmatrix} = Y \tag{70}$$

where Q_y is orthogonal, since $(\tilde{\Phi}\tilde{Q})'(\tilde{\Phi}\tilde{Q}) = \tilde{Q}'\tilde{\Phi}'\tilde{\Phi}\tilde{Q} = \tilde{Q}'\tilde{Q} = I$ holds. The factorization (16) easily follows, which proves the Lemma.

B. Proof of Lemma 3.2

Let us define $\hat{A}^{(i,j)} \in \mathbb{R}^{(q-i+1)m_q\times n_q}$ the result of the block-column update of Step 7 at the i-th iteration of Algorithm 1, with $i=1,\ldots,q-1$, and $j=i+1,\ldots,q$ such that:

$$\hat{A}^{(i,j)} = \hat{Q}^{(i)\prime} A_{\mathcal{I}_i,j} \tag{71}$$

with $\mathcal{I}_i = \{i,\ldots,q\}$. If we replace the definition of $\hat{Q}^{(i)}$ and the i-th block column of A, respectively equations (10) and (12), into (71) we get:

$$\hat{A}^{(i,i+1)} = \begin{bmatrix} Q_{1,1}^{(i)\prime} \begin{bmatrix} \mathbf{0}'_{ir_q} \\ S_x \end{bmatrix} \\ S_y \\ S_z \\ \mathbf{0}'_{m_q(q-i-1)} \end{bmatrix}, \quad \hat{A}^{(i,j_1)} = \begin{bmatrix} Q_{1,1}^{(i)\prime}\, \mathbf{0}'_{ir_q+n_q} \\ \mathbf{0}_{r_q+(j_1-i-2)m_q} \\ S_x \\ S_y \\ S_z \\ \mathbf{0}'_{m_q(q-j_1)} \end{bmatrix} \tag{72}$$

with $j_1 = i+2,\ldots,q$. Clearly, $\hat{A}^{(i,j_1)} = \hat{Q}^{(i)\prime} A_{\mathcal{I}_i,j_1} \equiv A_{\mathcal{I}_i,j_1}$ which proves that A_i triangularization only updates A_{i+1} block-column.

C. Proof of Lemma 3.3

Consider the matrix $\Psi_1 = \begin{bmatrix} w & \Psi' \end{bmatrix}'$ obtained by prepending w', $w \in \mathbb{R}^n$, to Ψ. We know that

$$\sigma_i(\Psi_1) = \sqrt{\lambda_i(\Psi_1'\Psi_1)} = \sqrt{\lambda_i(\Psi'\Psi + ww')} \tag{73}$$

with $i = 1,\ldots,n$. Given two Hermitian matrices $X, Y \in \mathbb{R}^{n\times n}$, denote $p = n_+(Y)$ and $q = n_-(Y)$. Then, from Weyl's theorem [43], we have

$$\lambda_{i+q}(X) \le \lambda_i(X+Y) \le \lambda_{i-p}(X),\ i = 1,\ldots,n \tag{74}$$

with $\lambda_i(X) = \infty$ for $i < 1$ and $\lambda_i(X) = -\infty$ for $i > n$. In case of a rank-1 perturbation, since $Y = yy'$ with $y \in \mathbb{R}^n$ is positive-semidefinite, it is easy to show that (74) reduces to:

$$\lambda_i(X) \leq \lambda_i(X + yy') \leq \lambda_{i-1}(X),\ i = 1, \dots, n \tag{75}$$

that is the eigenvalues of $(X + yy')$ *interlace* those of X. Inequality (75) applies to (73) because $\Phi'\Phi$ is symmetric. The matrix $B \triangleq \begin{bmatrix} W' & \Psi' \end{bmatrix}'$ being factorized in (29) can be obtained by iteratively prepending m rows to Ψ, thus the left side of inequality (75) holds iteratively, from which we get $\lambda_i(\Psi'\Psi) \leq \lambda_i(B'B)$. Moreover, given Ψ invertible, we know that $\sigma_n(\Psi) > 0$, and $\sigma(R) \equiv \sigma(B)$, and hence we can derive that $\sigma_i(\Psi) \leq \sigma_i(R)$, from which

$$\sigma_i^2(\Psi) \leq \sigma_i^2(R),\ i = 1, \dots, n \tag{76}$$

easily follows. Combining (35) with (76) leads to

$$\|\Psi^{-1}\|_F = \sqrt{\sum_{i=1}^{n} \frac{1}{\sigma_i^2(\Psi)}} \geq \sqrt{\sum_{i=1}^{n} \frac{1}{\sigma_i^2(R)}} = \|R^{-1}\|_F \tag{77}$$

which proves the theorem. Note that $\|\Psi^{-1}\|_2 = 1/\sigma_1(\Psi) \geq 1/\sigma_1(R) = \|R^{-1}\|_2$ holds analogously.

D. Proof of Lemma 3.4

From the definition of the Frobenius norm $\|X\|_\mathrm{F} = \sqrt{\sum_{i,j} |x_{i,j}|^2}$, with $x_{i,j}$ the i-th row and j-th column element of X, it is easy to derive that

$$\left\| \begin{bmatrix} X_1 & \dots & X_n \end{bmatrix} \right\|_\mathrm{F} = \sqrt{\sum_{i=1}^{n} \|X_i\|_\mathrm{F}^2} \tag{78}$$

with X_i being column blocks of matrix X. The same relation (78) holds true also in case of a partitioning of X in row blocks. Hence, for matrix (40) we can write

$$\left\| \begin{bmatrix} Q_{1,1} \\ Q_{2,1} \end{bmatrix} \right\|_\mathrm{F} = \sqrt{\|Q_{1,1}\|_\mathrm{F}^2 + \|Q_{2,1}\|_\mathrm{F}^2} \tag{79a}$$

$$\left\| \begin{bmatrix} Q_{2,1} & Q_{2,2} \end{bmatrix} \right\|_\mathrm{F} = \sqrt{\|Q_{2,1}\|_\mathrm{F}^2 + \|Q_{2,2}\|_\mathrm{F}^2}. \tag{79b}$$

Additionally, the Frobenius norm satisfies the relation $\|X\|_\mathrm{F} = \sqrt{\mathrm{tr}(AA')}$, and if we define $Q_c \in \mathbb{R}^{m \times n_c}$ and $Q_r \in \mathbb{R}^{n_r \times m}$ as any given subset of n_c columns and n_r rows of an orthonormal matrix $Q \in \mathbb{R}^{m \times m}$, respectively, we then have

$$\|Q_c\|_\mathrm{F} = \sqrt{\mathrm{tr}(Q_c'Q_c)} = \sqrt{\mathrm{tr}(I_{n_c})} = \sqrt{n_c} \tag{80a}$$

$$\|Q_r\|_\mathrm{F} = \sqrt{\mathrm{tr}(Q_rQ_r')} = \sqrt{\mathrm{tr}(I_{n_r})} = \sqrt{n_r}. \tag{80b}$$

From (80) we deduce that the Frobenius norm of block-columns and block-rows of an orthonormal matrix depends exclusively from the dimension of the block. Combining (79) and (80) leads to

$$\sqrt{\|Q_{2,1}\|_\mathrm{F}^2 + \|Q_{2,2}\|_\mathrm{F}^2} = \sqrt{\|Q_{1,1}\|_\mathrm{F}^2 + \|Q_{2,1}\|_\mathrm{F}^2} \tag{81}$$

from which we can derive that $\|Q_{1,1}\|_\mathrm{F} = \|Q_{2,2}\|_\mathrm{F}$.

Gionata Cimini received his Master's degree cum laude in Computer and Automation Engineering in 2012, and his Ph.D. cum laude in Information Engineering in 2017 from Università Politecnica delle Marche, Italy. He has been a Guest Ph.D. Scholar with the IMT School for Advanced Studies Lucca, and he was the recipient of the Best Ph.D. Thesis in Systems and Control Engineering, Italy, 2017. He held Research Assistant positions with the Information Department, Università Politecnica delle Marche and with the Automotive Research Center at University of Michigan, USA. He worked as an advanced control engineer for General Motors Company, Detroit, USA. Since 2017 he works at ODYS S.r.l., Milano, Italy, where he is currently the Technical Lead for Numerical Optimization. He has published more than 50 papers in the areas of embedded optimization, model predictive control, and their application to automotive, aerospace, and energy domains, and he is the coinventor of 4 international patents. He is the author or co-author of commercial software and algorithms for real-time optimization, model predictive control and machine learning, some of which are currently used for automotive mass production.

Alberto Bemporad received his Master's degree cum laude in Electrical Engineering in 1993 and his Ph.D. in Control Engineering in 1997 from the University of Florence, Italy. In 1996/97 he was with the Center for Robotics and Automation, Department of Systems Science & Mathematics, Washington University, St. Louis. In 1997-1999 he held a postdoctoral position at the Automatic Control Laboratory, ETH Zurich, Switzerland, where he collaborated as a Senior Researcher until 2002. In 1999-2009 he was with the Department of Information Engineering of the University of Siena, Italy, becoming an Associate Professor in 2005. In 2010-2011 he was with the Department of Mechanical and Structural Engineering of the University of Trento, Italy. Since 2011 he has been a Full Professor at the IMT School for Advanced Studies Lucca, Italy, where he served as the Director of the institute from 2012 to 2015. He spent visiting periods at Stanford University, the University of Michigan, and Zhejiang University. In 2011 he co-founded ODYS S.r.l., a company specialized in developing model predictive control systems for industrial production. He has published more than 400 papers in the areas of model predictive control, hybrid systems, optimization, and automotive control, and is the co-inventor of 21 patents. He is the author or coauthor of various software packages for model predictive control design and implementation, including the Model Predictive Control Toolbox (The Mathworks, Inc.) and the Hybrid Toolbox for MATLAB. He was an Associate Editor of the IEEE Transactions on Automatic Control during 2001-2004 and Chair of the Technical Committee on Hybrid Systems of the IEEE Control Systems Society from 2002 to 2010. He received the IFAC High-Impact Paper Award for the 2011-14 triennial, the IEEE CSS Transition to Practice Award in 2019, the 2021 SAE Environmental Excellence in Transportation Award, the 2024 Beale–Orchard-Hays Prize for Excellence in Computational Mathematical Programming, and an ERC Advanced Research Grant in 2024. He is an IEEE Fellow since 2010 and an IFAC Fellow since 2025.