Variable Elimination in Model Predictive Control Based on K-SVD and QR Factorization

A. Bemporad and G. Cimini

***Abstract*—For linearly constrained least-squares problems that depend on a vector of parameters, this paper proposes techniques for reducing the number of involved optimization variables. After first eliminating equality constraints in a numerically robust way by QR factorization, we propose a technique based on singular value decomposition (SVD) and unsupervised learning, that we call K-SVD, and neural classifiers to automatically partition the set of parameter vectors in K nonlinear regions in which the original problem is approximated by using a smaller set of variables. For the special case of parametric constrained least-squares problems that arise from model predictive control (MPC) formulations, we propose a novel and very efficient QR factorization method for eliminating equality constraints. Together with SVD or K-SVD, the method provides a numerically robust alternative to standard condensing and move blocking, and to other complexity reduction methods for MPC based on basis functions. We show the good performance of the proposed techniques in numerical tests and in a problem of linearized MPC of a nonlinear benchmark process.**

***Index Terms*—Constrained least squares, model predictive control, variable elimination, singular value decomposition, unsupervised learning.**

I. Introduction

Several problems in engineering can be reformulated as parametric constrained least-squares (pCLS) problems in which the matrices defining the cost function and the equality and inequality constraints depend on a set of parameters. Examples range from control engineering, in particular model predictive control (MPC) [1], filtering and smoothing [2], financial engineering [3]–[5], to mention a few. A common feature in such applications is that the pCLS problem must be solved quickly and in a numerically robust way, often in simple embedded devices. This asks for methods that can reformulate, and possibly approximate, the problem by reducing the number of optimization variables, and that aim at good numerical conditioning, so to ease the numerical procedure employed to solve the reduced problem.

In particular, in MPC formulations the equality constraints are usually eliminated by substituting the predicted state as a function of the applied inputs, a.k.a. *condensing* [6], that, as we will discuss later, can lead to very badly conditioned problems. In addition, the number of optimization variables is typically reduced by using *move blocking* [7]–[10]. This corresponds to keeping the input signal constant ("blocked") between pre-specified prediction steps; typically the input signal is free to move during the first few prediction steps and then kept constant. The blocking scheme could be also modified in real-time, such as with the heuristic method proposed in [11]. Blocking moves, however, complicate the recursive structure of the problem, therefore making efficient solution methods conceived for the non-condensed problem [12] not directly applicable. Moreover, move blocking may prevent the application of involved blocking strategies such as the aforementioned [11].

The so-called *partial condensing* [13] lies in between the condensed and non-condensed forms. It is a reformulation of the MPC optimization problem in which just a subset of the equality constraints are eliminated by replacing some of the states with the corresponding state responses. Such a reformulated quadratic programming (QP) problem has a *block sparse* structure, which reduces the number of optimization variables while still allowing the exploitation of efficient sparse linear algebra. For a benchmark comparison between partial condensed and non-condensed forms, the reader is referred to [14], where improvements are shown for open-source solvers commonly used in sparse problems.

Alternatives to the standard condensing methods have been investigated with the aim of preserving the sparsity of the lower dimensional problem, so to exploit tailored solvers. This is typically denoted as *sparse condensing*. For instance, in [15] the authors propose the use of Turnback algorithm [16] to find a banded null basis, and suggest to perform a open-loop simulation with a zero input excitation to get a particular solution. In [17], a banded formulation is obtained by computing the deadbeat feedback gain, under the assumption of null controllability. The method proposed in [18] relaxes such an assumption and the basis for equality constraints is computed through deadbeat responses. One drawback of these approaches is that in case of unstable models the generated open-loop solution may contain numbers that exponentially increase with the prediction horizon.

Principal Component Analysis (PCA) was also proposed for input trajectory parameterization in MPC formulations. In [19], the authors propose to construct a basis from the principal components of the Hessian of the condensed form. Alternatively, such a basis can be derived from the principal components of the open-loop response matrix, as in [20], which however limits its application to the only case where standard condensing is used to eliminate equality constraints. A different approach, proposed in [21], is to apply PCA to a matrix that collects the snapshots of previous control inputs already applied to the real plant, or to a simulation model. A drawback of all the above methods is that, if the matrices of the linear prediction model are time-varying, PCA must be

A. Bemporad is with the IMT School for Advanced Studies Lucca, Italy. Email: alberto.bemporad@imtlucca.it

G. Cimini is with ODYS S.r.l., Italy. Email: gionata.cimini@odys.it

performed in real-time, making the execution of the overall MPC algorithm computationally intractable in many applications. Additionally, the method does not extend to pCLS problems in which the equality constraints are eliminated by using more general numerical methods, like the ones described in [22, Ch. 20–22].

A. Contribution

This paper addresses the issue of reducing the number of optimization variables in pCLS problems by eliminating equality constraints, and of further eliminating degrees of freedom, possibly sacrificing optimality, to simplify the resulting optimization problem. First, we propose a method based on a computationally efficient QR factorization of the matrix associated with equality constraints, that in case of pCLS problems arising from MPC is a much more numerically stable method to eliminate equalities than standard condensing, in particular when the dynamics given by the prediction model are unstable.

A second contribution is an offline procedure based on PCA, unsupervised learning, and multiclass classification to reduce the number of remaining optimization variables. We call the unsupervised learning procedure K-SVD, as it is an extension of SVD to determine K different sets of principal directions, rather than just one as in standard linear PCA. As each sample vector is associated with a corresponding parameter vector, K-SVD determines a nonlinear partition of the parameter space into regions, and provides a set of principal directions in each region. When K-SVD is applied to pCLS problems, each sample vector is an optimal solution of the pCLS problem for a corresponding value of the parameter vector, and the method returns K different approximate reformulations of the pCLS problem.

For pCLS problems arising from MPC, we also address issues of recursive feasibility of the proposed scheme, in order to make sure that reducing the number of free optimization variables does not lead to infeasible reduced-order problems, which is particularly important in an MPC setting.

Finally, we provide evidence of the benefits of the proposed methods in numerical experiments.

B. Notation

Given a finite set $I = \{i_1, \dots, i_N\}$, $\mathrm{card}(I)$ denotes its number N of elements (cardinality). Given a vector $v \in \mathbb{R}^n$, $\|v\|_2$ is the Euclidean norm of v, $|v|$ is the component-wise absolute value of v, $\left[v\right]_+$ is the projection of v onto the non-negative orthant, and $\mathrm{diag}(v)$ is the diagonal matrix of $\mathbb{R}^{n\times n}$ formed from v. Given a matrix $A \in \mathbb{R}^{n\times m}$, $\|A\|_F$ is the Frobenius norm of A, $\|A\|_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{m} A_{ij}^2$, $\ker(A)$ is the null space of A, $\mathrm{Im}(A)$ the range space of A, and $\mathrm{rank}(A)$ the rank of A. The sub-matrix of A obtained by collecting its entries whose row-index ranges from i to j and column-index from h to k is denoted by $A_{i:j,k:h}$. We define as $\kappa(A) = \|A\|\|A^+\|$ the generalized condition number of A with respect to the spectral norm, with A^+ the Moore-Penrose inverse of A. The matrix $0_m \in \mathbb{R}^{n\times m}$ is the zero matrix with m columns, or the zero row vector if $n = 1$. 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 $[a, b]$, and by $\mathcal{D}(c, d)$, $c, d \in \mathbb{N}_0$, $d > c$ the uniform discrete distribution in the interval $[c, d]$.

II. Problem formulation

Consider the following parameter-dependent constrained least-squares (CLS) problem

$$\begin{aligned} \min_{z} \quad & \frac{1}{2}\|A(\theta)z - b(\theta)\|_2^2 && \text{(1a)}\\ \text{s.t.} \quad & C(\theta)z = e(\theta) && \text{(1b)}\\ & G(\theta)z \le g(\theta) && \text{(1c)} \end{aligned}$$

where $z \in \mathbb{R}^\ell$ is the optimization vector and $\theta \in \mathbb{R}^p$ is the vector of parameters defining the problem instance, $A(\theta) \in \mathbb{R}^{n_c\times\ell}$, $b(\theta) \in \mathbb{R}^{n_c}$, $C(\theta) \in \mathbb{R}^{n_e\times\ell}$, $e(\theta) \in \mathbb{R}^{n_e}$, $G(\theta) \in \mathbb{R}^{n_i\times\ell}$, and $g(\theta) \in \mathbb{R}^{n_i}$. For simplicity, we assume that $\mathrm{rank}(C(\theta)) = n_e$, $\forall\theta \in \mathbb{R}^p$, although this assumption can be easily relaxed (see Remark 2.1 below). Our goal is to find numerically efficient and robust ways to solve (1), including methods to approximate the problem so that it can be solved with respect to a *reduced number of optimization variables*. From now on, for simplicity of notation we will drop the dependence on θ where obvious.

A. Variable reduction by elimination of equality constraints

We start reducing the number of variables by eliminating the equality constraints (1b). Several methods exist to handle equality-constrained least squares problems [22, Ch. 20–22]. We consider the following numerically-robust variable-elimination method based on the QR factorization

$$C' = QR, \quad Q = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}, \quad R = \begin{bmatrix} R_1 \\ 0 \end{bmatrix} \qquad (2)$$

where Q is an orthogonal matrix, $Q'Q = I$, $Q_1 \in \mathbb{R}^{\ell\times n_e}$, $Q_2 \in \mathbb{R}^{\ell\times(\ell-n_e)}$, and R is upper triangular, with $R_1 \in \mathbb{R}^{n_e\times n_e}$. The columns of matrix Q_2 provide a basis of $\ker(C)$, as

$$CQ_2 = \begin{bmatrix} R_1' & 0 \end{bmatrix}\begin{bmatrix} Q_1' \\ Q_2' \end{bmatrix}Q_2 = \begin{bmatrix} R_1' & 0 \end{bmatrix}\begin{bmatrix} 0 \\ I \end{bmatrix} = 0$$

Consider now the change of variables

$$y = \begin{bmatrix} \bar{s} \\ s \end{bmatrix}, \quad \bar{s} = Q_1'z, \quad s = Q_2'z$$

Since $z = Qy$, we get $Cz = R'Q'Qy = \begin{bmatrix} R_1' & 0 \end{bmatrix} y = R_1'\bar{s} = e$. As we assumed $\mathrm{rank}(C) = n_e$, matrix R_1 is nonsingular and hence

$$\bar{s} = (R_1')^{-1}e, \ s \text{ free} \qquad (3)$$

where $s \in \mathbb{R}^{\ell-n_e}$ is the vector of remaining optimization variables. By substituting

$$\begin{aligned} z &= Q_2 s + \bar{z} && \text{(4a)}\\ \bar{z} &\triangleq Q_1\bar{s} && \text{(4b)} \end{aligned}$$

the following CLS problem *without* equality constraints

$$\begin{aligned} \min_{s} \quad & \frac{1}{2}\|AQ_2 s - (b - A\bar{z})\|_2^2 && \text{(5a)}\\ \text{s.t.} \quad & GQ_2 s \le g - G\bar{z} && \text{(5b)} \end{aligned}$$

is equivalent to (1), where in (5) all constant matrices/vectors are in general a function of θ. Clearly, $\bar{z}$ satisfies (1b).

The QR elimination method described above enjoys the following property:

Proposition 2.1: Vector $\bar{z}$ solves the minimum norm problem

$$\bar{z} = \arg\min_{z} \|z\|^2 \quad \textit{subject to } Cz = e \tag{6}$$

Proof: The optimality conditions from Problem (6) are $2z + C'z_D = 0$, $Cz = e$, which gives $z_D = -2(CC')^{-1}e$, and hence $z = C'(CC')^{-1}e = QR(R'R)^{-1}e = Q_1R_1R_1^{-1}(R_1')^{-1}e = \bar{z}$. ■

The orthogonal unit vectors given by the columns of $\begin{bmatrix} Q_1 & Q_2 \end{bmatrix}$ and the associated coordinate transformation $z = \bar{z} + Q_2 s$ are ideal for numerical robustness. Consider a generic coordinate transformation $z = Ys_y + Zs$, where Y and Z denote any matrices whose columns form a basis for $\mathrm{Im}(C)$ and $\ker(C)$, respectively. As CY is non-singular, any vector $z = Y(CY)^{-1}e + Zs$ satisfies (1b), and one must select Y in such a way that CY is well conditioned. Proposition 2.1 shows that if $Y(CY)^{-1}e$ solves (6) then $Y(CY)^{-1} = C'(CC')^{-1}$ which makes the condition number of CY independent from the basis selection. The pair (Y, Z) such that Proposition 2.1 holds true is not unique. Among the possible choices, letting $(Y, Z) \triangleq (Q_1, Q_2)$ guarantees also the best bound on *i)* $\kappa(CY)$ as $\kappa(CQ_1) = \kappa(R_1) = \kappa(C)$, *ii)* $\kappa(AZ)$ as $\kappa(AQ_2) \leq \kappa(A)$ if $n_c \geq \ell$ and A is full column rank, *iii)* $\kappa(GZ)$ as $\kappa(GQ_2) \leq \kappa(G)$ if $n_i \geq \ell$ and G is full row rank. We note that AZ and GZ are the matrices forming the reduced CLS problem (5), therefore the coordinate transformation $z = \bar{z} + Q_2 s$ improves the numerical robustness of both variable elimination and CLS problem optimization.

Remark 2.1: The variable elimination method for equality constraints can be easily generalized to the case in which $\mathrm{rank}(C) = n_3 < n_e$. Let $C'P = Q\begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix}$ be a rank-revealing QR factorization of C', with $R_{11} \in \mathbb{R}^{n_3 \times n_3}$, $R_{22} \in \mathbb{R}^{(n_e - n_3)\times(n_e - n_3)}$ and $P \in \mathbb{R}^{n_e \times n_e}$ a permutation matrix such that $||R_{2,2}|| \leq \epsilon$, with ϵ an arbitrary small tolerance. Let $P = \begin{bmatrix} P_1 & P_2 \end{bmatrix}$ with $P_1 \in \mathbb{R}^{n_e \times n_3}$ and $P_2 \in \mathbb{R}^{n_e \times (n_3 - n_e)}$. Then, equality constraints can be eliminated by replacing the fixed variables in (3) with $\bar{s} = (R_{11}')^{-1}P_1'e$. Similarly, the method can be also generalized to the less common case in which $\ell \leq n_e$ and, possibly, $\mathrm{rank}(C) < n_e$. ■

III. Variable reduction by SVD

After eliminating equality constraints, we want to attempt to further reduce the complexity of solving (5) by lowering the number of optimization variables from n to m, possibly at the price of introducing suboptimality and infeasibility (we will handle feasibility issues in Section III-B). To this end, let us consider a basis $\Phi = [\phi_1 \ \dots \ \phi_m] \in \mathbb{R}^{n \times m}$ and a constant vector $\phi_0 \in \mathbb{R}^n$ that we use to parameterize the vector s of free variables as an affine function of a new vector $v \in \mathbb{R}^m$ of free optimization variables

$$s = \phi_0 + \sum_{i=1}^{m} \phi_i v_i = \phi_0 + \Phi v \tag{7}$$

We assume that matrix Φ is full column rank, otherwise some degrees of freedom v_i would be wasted.

In the presence of inequality constraints (1c), after substituting (7) and optimizing with respect to v, we would like to recover the solution s^* of the original inequality-constrained least-squares problem (5) at best. However, we also want to recover the exact solution of (1) when all inequality constraints (1c) are inactive and $m < n$. Therefore, for any given new value of θ, before applying (7) we first compute the unconstrained solution of (5a)

$$s_u^* = (Q_2'A'AQ_2)^{-1}Q_2'(b - A\bar{z}) \tag{8}$$

and check if constraints (5b) are satisfied for $s = s_u^*$. In case they are, we can immediately retrieve the optimal solution $z_u^* = \bar{z} + Q_2 s_u^*$. Otherwise, we substitute (7) in (5) and solve the reduced CLS problem

$$v^* = \arg\min_{v \in \mathbb{R}^m} \quad \tfrac{1}{2}\|AQ_2\Phi v - (b - Az_0)\|_2^2 \tag{9a}$$

$$\text{s.t.} \quad GQ_2\Phi v \leq g - Gz_0 \tag{9b}$$

where $z_0 = \bar{z} + Q_2\phi_0$, and then set

$$z^* = Q_2\Phi v^* + z_0 \tag{10}$$

A. Choice of basis

Let us focus on finding a basis Φ that enables us to reconstruct s^* under inequality constraints (5b) at best when the unconstrained solution s_u^* in (8) is infeasible. Due to the dependence of (1) on the vector θ of parameters, s^* is also a function of θ.

A standard approach for dimensionality reduction is to resort to principal component analysis (PCA) to identify an optimal basis Φ for a given set of values θ_i of the parameter vector, $i = 1, \dots, M$, as follows. For each θ_i, let s_i^* be the constrained solution of (5) when $\theta = \theta_i$. Let $\bar{s}_m = \frac{1}{M}\sum_{i=1}^{M} s_i^*$ be the mean of the collected vectors s_i^*, and let $S = [s_1^* - \bar{s}_m \ \dots \ s_M^* - \bar{s}_m]'$, $S \in \mathbb{R}^{M \times n}$. Consider the economy-size singular value decomposition (SVD) of S

$$S = U\Sigma V' \tag{11}$$

where $\Sigma = \mathrm{diag}([\sigma_1 \ \dots \ \sigma_n]')$, $\sigma_i \geq \sigma_j$ for $i \leq j$. Matrix $V = [V_1 \ \dots \ V_n] \in \mathbb{R}^{n \times n}$ is orthogonal, $V'V = I$, and provides the principal directions. Matrix $U \in \mathbb{R}^{M \times n}$ collects n columns of an orthogonal matrix, such that $U_i\Sigma$ are the principal components of $s_i^* - \bar{s}_m$ along the principal directions V_1, …, V_m, and U_i is the i-th row of U. Given the number m of degrees of freedom we want to allow in the constrained least-squares problem (9b) by restricting s as in (7), we set

$$\Phi = [V_1 \ \dots \ V_m] \tag{12a}$$

$$\phi_0 = \bar{s}_m \tag{12b}$$

Note that, as suggested above, problem (9) is not solved when the solution s_u^* is already feasible with respect to the constraints (9b), and therefore the basis vectors in (12) will not be used in such a case. Hence, to form matrix S we only need to collect samples θ_i such that the corresponding unconstrained optimizer s_u^* violates (9b). To this end, we select vectors θ_i such that the corresponding optimal Lagrange multipliers λ_i associated with the inequality constraints (5b) are above a certain threshold $\epsilon_\lambda > 0$.

B. Feasibility

Because of the reduction of the number of degrees of freedom, even if problem (1) is feasible it might happen that (9) is infeasible. In this section we discuss a few ways to address this issue.

First, assuming that we have a feasible vector z_f available with respect to the constraints (1b)–(1c), or equivalently a solution s_f satisfying (5b), we want to be able to obtain z_f from the parameterization (7) in spite of the reduction of the number of degrees of freedom. To achieve this, one can set $\phi_0 = s_f$ instead of ϕ_0 as in (12b) in order to guarantee that $v = 0$ is a feasible solution. A more flexible approach is to add $\bar{s}_m - s_f$ in the basis and parameterize s as

$$s = s_f + v_0(\bar{s}_m - s_f) + \sum_{i=1}^{m} v_i \phi_i \tag{13}$$

where $v_0 \in \mathbb{R}$ is an extra degree of freedom. Again, the resulting reduced CLS problem is feasible for $v = 0$, where now $v \in \mathbb{R}^{m+1}$, and maintains the optimal principal component decomposition (12) for $v_0 = 1$.

A second approach is to replace (1c) with the following soft constraints

$$G(\theta)z \le g(\theta) + V_g(\theta)\zeta \tag{14}$$

where $V_g(\theta) \in \mathbb{R}^{n_i \times n_\zeta}$ is a matrix whose entries are all zero except one per row which is positive. The vector of *slack variables* $\zeta \in \mathbb{R}^{n_\zeta}$ is penalized by changing (1a) to

$$\min_z \frac{1}{2} \left\| \begin{bmatrix} A(\theta) & 0 \\ 0 & \Lambda_\zeta(\theta) \end{bmatrix} \begin{bmatrix} z \\ \zeta \end{bmatrix} - \begin{bmatrix} b(\theta) \\ 0 \end{bmatrix} \right\|_2^2 \tag{15}$$

where $\Lambda_\zeta(\theta) \in \mathbb{R}^{n_\zeta \times n_\zeta}$. In this case, constraints (14) become

$$G(\theta)Q_2(\theta)\Phi v \le g(\theta) - G(\theta)z_0 + V_g(\theta)\zeta \tag{16}$$

When the original CLS problem is formulated with soft-constraints (14), an alternative approach is to adopt the parameterization introduced in (12) and keep both v and ζ as optimization variables. In case ζ is a vector ($n_\zeta > 1$), the number of slacks can be also arbitrarily shrunk to $\bar{n}_\zeta$, $1 \le \bar{n}_\zeta < n_\zeta$, by right multiplying matrix $V_g(\theta)$ by a binary matrix E_ζ, $E_\zeta \in \{0,1\}^{n_\zeta \times \bar{n}_\zeta}$, with all columns of E_ζ being nonzero. For instance, for the minimum value $\bar{n}_\zeta = 1$, we set $E_\zeta = [1 \ \dots \ 1]'$.

Remark 3.1: Problem (15) can be modified to account for a linear penalty on ζ, so to achieve an exact penalty function on ζ [23, Sect. 14.3]. Let $\gamma_\zeta(\theta)$ a vector of (positive and large enough) linear weights on ζ and $\Gamma_\zeta(\theta) = \mathrm{diag}(\gamma_\zeta(\theta))$. If $\Lambda_\zeta(\theta)$ is invertible, (15) can be changed to

$$\min_z \frac{1}{2} \left\| \begin{bmatrix} A(\theta) & 0 \\ 0 & \Lambda_\zeta(\theta) \end{bmatrix} \begin{bmatrix} z \\ \zeta \end{bmatrix} - \begin{bmatrix} b(\theta) \\ \Lambda_\zeta^{-1}(\theta)'\Gamma_\zeta(\theta)e_\zeta \end{bmatrix} \right\|_2^2 \tag{17}$$

with $e_\zeta = -[1 \dots 1]'$, and solved subject to (16) and the additional constraints $\zeta_i \ge 0$, $i = 1, \dots, n_\zeta$.

Example 3.1: Consider a parameter-dependent CLS problem with $n = 20$, $n_\theta = 3$, $n_c = 20$, $A(\theta) \equiv A$, $b(\theta) = b + F\theta$, with the entries of A, b, and F randomly generated from a normal distribution, and $G(\theta) = [I \ -I]'$, $g_i(\theta) = [1 \dots 1]'$ (box constraints). After generating $M = 5000$ random samples of θ, such that $\theta_i \sim \mathcal{U}(-1, 1)$, and collecting the corresponding optimal solutions we apply the SVD decomposition (12) for values of m ranging between 1 and $n - 1$. Then, we generate further $N_{\mathrm{val}} = 500$ vectors θ for validation. In both cases, in collecting the parameter vectors θ_i we discard those whose associated optimal dual variables are all smaller than $\epsilon_\lambda = 10^{-5}$, to ensure that at least one inequality constraint is active at optimum. For each θ, we solve the reduced CLS problem (9) with respect to $v \in \mathbb{R}^m$ and to the additional slack variable $\zeta \in \mathbb{R}$ as in (17) with $\Lambda_\zeta = 10^5$, $\Gamma_\zeta = 10^5$, and $V_g = [1 \ \dots \ 1]'$ in (14).

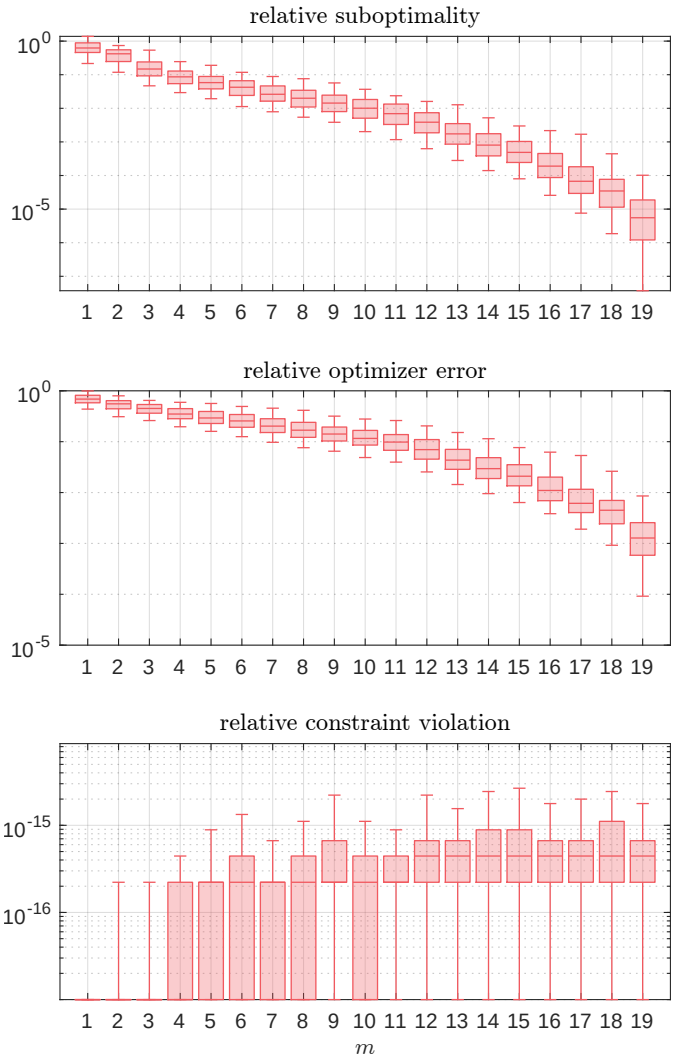

Fig. 1. Distribution of the errors induced by using (12), as a function of m: relative optimality error (top plot), relative difference of optimizers (mid plot), relative maximum violation of the inequality constraints (bottom plot) over 500 validation tests.

Figure 1 shows the results obtained in terms of the relative optimality error $\frac{\|Az^* - b - F\theta\| - \|Az_r^* - b - F\theta\|}{\|Az^* - b - F\theta\|}$, the relative error $\frac{\|z^* - z_r^*\|}{\|z^*\|}$, and the maximum relative constraint violation $\max_i \left\{ \frac{G_i z_r^* - g_i}{|g_i|} \right\}$, where all norms are Euclidean norms. As expected, the higher the number m of basis vectors selected in order of decreasing eigenvalue, the better is the quality of the approximation z_r. In particular the relative optimizer error is

Algorithm 1 K-means in optimizer space.

Input: Optimizer samples $s_1^*, \dots, s_M^* \in \mathbb{R}^n$, number K of clusters, number m of basis elements.

1. Create random partition $I_1, \dots, I_K$, $\cup_{i=1}^K I_i = \{1, \dots, M\}$, $I_i \cap I_j = \emptyset$, $\forall i, j = 1, \dots, K$, $i \neq j$;
2. **repeat**:
2.1. **for** $j = 1, \dots, K$ **do**:
2.1.1. $M_j \leftarrow$ number of elements of I_j;
2.1.2. $\phi_0^j \leftarrow \frac{1}{M_j} \sum_{i \in I_j} s_i^*$;
2.2. **for** $i = 1, \dots, M$ **do**:
2.2.1. Reassign s_i^* to cluster j such that

$$j = \arg \min_{j \in \{1, \dots, K\}} \|s_i^* - \phi_0^j\|_2^2 \tag{18}$$

3. **until** convergence;
4. **for** $j = 1, \dots, K$ **do**:
4.1. Compute basis $\Phi_j \in \mathbb{R}^{n \times m}$ using SVD of matrix S_j obtained by collecting $s_i^* - \phi_0^j$ for $i \in I_j$;
5. **end**.

Output: Clusters $I_1, \dots, I_K$, corresponding basis $\Phi_1, \dots, \Phi_K$ and offset vectors $\phi_0^1, \dots, \phi_0^K$.

Algorithm 2 K-SVD for parameter-dependent PCA.

Input: Optimizer samples $s_1^*, \dots, s_M^* \in \mathbb{R}^n$, number m of basis elements, initial clusters $I_1, \dots, I_K$, $\cup_{i=1}^K I_i = \{1, \dots, M\}$, $I_i \cap I_j = \emptyset$, $\forall i, j = 1, \dots, K$, $i \neq j$.

1. **repeat**:
1.1. **for** $j = 1, \dots, K$ **do**:
1.1.1. $M_j \leftarrow$ number of elements of I_j;
1.1.2. $\phi_0^j \leftarrow \frac{1}{M_j} \sum_{i \in I_j} s_i^*$;
1.1.3. Compute basis $\Phi_j \in \mathbb{R}^{n \times m}$ using the economy-size SVD

$$S_j = [U_1^j \; U_2^j] \begin{bmatrix} \Sigma_1^j & 0 \\ 0 & \Sigma_2^j \end{bmatrix} [\Phi_j \; V_2^j] \tag{19}$$

where the rows of S_j are $(s_i^* - \phi_0^j)'$, $i \in I_j$;
1.2. **for** $i = 1, \dots, M$ **do**:
1.2.1. Reassign s_i^* to cluster j such that

$$j = \arg \min_{j \in \{1, \dots, K\}} \left\{ \min_v \|s_i^* - \Phi_j v - \phi_0^j\|_2^2 \right\} \tag{20}$$

2. **until** convergence;
3. **end**.

Output: Clusters $I_1, \dots, I_K$, corresponding bases $\Phi_1, \dots, \Phi_K$ of orthogonal vectors and offset vectors $\phi_0^1, \dots, \phi_0^K$.

monotonically decreasing with respect to m, as reducing m to $m - k_1$ is equivalent to imposing $v_i = 0$, $i = m - k_1 + 1, \dots, m$ in (7). Therefore a reduction to $m - k_2$ with $k_2 < k_1$ leads to a larger variance explained. ■

IV. Generalization to parameter-dependent bases

The method described in Section III determines a basis Φ to use for all parameter vectors $\theta \in \mathbb{R}^p$. In order to gain more flexibility, we propose to make Φ a piecewise-constant function of θ by using unsupervised learning and multiclass classification methods, as we describe next.

Assume M samples s_i^* have been collected for the corresponding set $\Theta = \{\theta_i\}_{i=1}^M$ of parameter vectors, as described in Section III, and that we only allow K different matrices $\Phi_1, \dots, \Phi_K \in \mathbb{R}^{n \times m}$ and vectors $\phi_0^1, \dots, \phi_0^K \in \mathbb{R}^n$ for the parameterization of s. A first method is to simply perform K-means [24, Algorithm 14.1] on Θ to get K clusters and repeat the approach of Section III on each cluster. Then, the clusters can be separated for example by using piecewise linear separation (the Voronoi diagram of the centroid of the clusters, robust linear programming [25], or one of the efficient algorithms proposed in [26]) in order to define a function that, for each given $\theta \in \mathbb{R}^p$, returns the corresponding basis Φ_j and ϕ_0^j to use. The main drawback of this approach is that clustering would be done only based on the Euclidean distance between two parameter vectors, independently on the values of the corresponding optimizer s^*.

Alternatively, one can perform K-means on the set $\{s_i^*\}_{i=1}^M$ to get K clusters $I_1, \dots, I_k$ of indices, $\cup_{i=1}^K I_i = \{1, \dots, M\}$, $I_i \cap I_j = \emptyset$, $\forall i, j = 1, \dots, K$, $i \neq j$, as described in Algorithm 1. This is a classical K-means algorithm with random initialization of the clusters I_j, $j = 1, \dots, K$, whose convergence is tested when the set $\{I_j\}$ of clusters is not changing between two consecutive iterations. After executing Algorithm 1, we compute a matrix Φ_j for each cluster by SVD. The cluster indices $\{I_j\}$ determined by Algorithm 1 also induce a partition of Θ in corresponding K sets. Finding a (nonlinear) separation function in the θ-space $\mathbb{R}^p$ is a *multiclass classification* problem [27] (or *binary classification* if $K = 2$), that is the problem of determining a function $\psi : \mathbb{R}^p \to \{1, \dots, K\}$ that associates to any vector θ its corresponding class $j = \psi(\theta)$.

The main drawback of running K-means in the space of optimization vectors s is that a small *distance* between two optimizers s_i^*, s_j^* does not necessarily implies that they can be well approximated by the same basis. For example, $s_i^* = [10 \; 0 \dots \; 0]'$ is much "closer" to $s_j^* = [100 \; 0 \dots \; 0]'$ than to $s_h^* = [10 \; 1 \dots \; 0]'$, as s_i^*, s_j^* are both multiple of the same vector. In light of the above considerations, we propose the variant of K-means described in Algorithm 2, that we call K-SVD algorithm, to perform clustering of the index set $\{1, \dots, M\}$ by reassigning each vector s_i^* to the cluster I_j whose current basis Φ_j and offset ϕ_0 best represent it in a least-squares sense.

K-SVD requires an initial partition of $\{1, \dots, M\}$ in K clusters. A possible approach is to create random clusters $I_1, \dots, I_K$ as in Step 1 of Algorithm 1, or to get $I_1, \dots, I_K$ by fully executing the K-means Algorithm 1 on $s_1^*, \dots, s_M^*$.

The following result proves that Algorithm 2 is indeed an algorithm, in that it terminates in a finite number of steps to a local minimum of the problem of finding the K "best" bases.

Theorem 4.1: Let $M > n > m$. K-SVD converges in a finite number of steps to a local minimum of the following

optimization problem

$$\min_{J,\{\Phi_j,\phi_0^j\}_{j=1}^K} \quad \sum_{i=1}^{M} \min_{v} \|s_i^* - \Phi_{J(i)} v - \phi_0^{J(i)}\|_2^2 \tag{21a}$$

$$\text{s.t.} \quad \phi_0^j = \frac{1}{M_j} \sum_{i \in I_j} s_i^*, \ j = 1, \dots, K \tag{21b}$$

where $J \in \{1, \dots, K\}^M$ is a sequence of cluster labels, namely $J(i) = j$ implies that s_i^* belongs to cluster #j, $I_j \triangleq \{i \in \{1, \dots, M\} : \ J(i) = j\}$, $M_i = \text{card}(I_j)$, and where the columns of each matrix Φ_j are orthogonal.

Proof: We want to show that Algorithm 2 is a coordinate-descent method that solves problem (21a) by iterating between minimizing w.r.t. $\{\phi_0^j\}_{j=1}^K$, then $\{\Phi^j\}_{j=1}^K$, and then J.

Consider first the label vector J and $\{\Phi^j\}_{j=1}^K$ fixed. Step 1.1.2 determines the vectors $\{\phi_0^j\}_{j=1}^K$ such that the equality constraints (21b) are satisfied. Now consider both the label vector J and $\{\phi_0^j\}_{j=1}^K$ fixed. We need to solve (21a) with respect to the bases $\{\Phi^j\}_{j=1}^K$, which is equivalent to solving the following problem

$$\min_{\{\Phi_j\}_{j=1}^K} \sum_{j=1}^{K} \left(\sum_{i \in I_j} \min_{v} \|s_i^* - \Phi_j v - \phi_0^j\|_2^2 \right) \tag{22}$$

Clearly problem (22) is separable in the K independent minimization problems

$$\min_{\Phi, \{v_i\}_{i \in I_j}} \sum_{i \in I_j} \|\bar{s}_i - \Phi v_i\|_2^2 \tag{23}$$

where $v_i \in \mathbb{R}^m$ and we have introduced the simplified notation $\bar{s}_i \triangleq s_i^* - \phi_0^{J(i)}$. Let $\bar{S} = [\bar{s}_{i_1} \ \dots \ \bar{s}_{i_{M_j}}]'$, $\bar{S} \in \mathbb{R}^{M_j \times n}$, be the matrix collecting the M_j samples $\bar{s}_i'$ of cluster j as its rows, where $\{i_1, \dots, i_{M_j}\} = I_j$, and let $D = [v_{i_1} \ \dots \ v_{i_{M_j}}]'$ be the corresponding matrix of optimal coordinates, $D \in \mathbb{R}^{M_j \times m}$. By Eckart-Young-Mirsky theorem [28], the following matrix optimization problem

$$\hat{S} = \arg \min_{\text{rank } \hat{S} \leq m} \|\bar{S} - \hat{S}\|_F^2 \tag{24}$$

is solved by $\hat{S} = U_1 \Sigma_1 V_1'$, where $U \Sigma V'$ is the SVD decomposition of $\bar{S}$, $U = [U_1 \ U_2 \ U_3]$, $U_1 \in \mathbb{R}^{M_j \times m}$, $U_2 \in \mathbb{R}^{M_j \times n-m}$, $U_3 \in \mathbb{R}^{M_j \times (M_j - n)}$, $U'U = I$, $\Sigma = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \\ 0 & 0 \end{bmatrix}$ is the matrix of singular values, $\Sigma \in \mathbb{R}^{M_j \times n}$, and $\Sigma_1 \in \mathbb{R}^{m \times m}$, $\Sigma_2 \in \mathbb{R}^{(n-m) \times (n-m)}$, while $V = [V_1 \ V_2]$, $V_1 \in \mathbb{R}^{n \times m}$ $V_2 \in \mathbb{R}^{n \times (n-m)}$, $V'V = I$. By setting $\Phi_j = V_1$ and $D = U_1 \Sigma_1 \in \mathbb{R}^{M_j \times m}$, we get

$$\|\bar{S} - \hat{S}\|_F^2 = \sum_{i \in I_j} \|\bar{s}_i - \Phi_j v_i\|_2^2$$

and therefore Φ_j is a matrix of $\mathbb{R}^{n \times m}$ with orthogonal columns that solves problem (23).

For fixed $\{\Phi^j\}_{j=1}^K$ and $\{\phi_0^j\}_{j=1}^K$, clearly (20) determines the vector J of labels that minimizes (21a), as for each sample s_i^* the basis Φ_j and bias ϕ_0^j are chosen that give the least value of $\min_v \|s_i^* - \Phi_j v - \phi_0^j\|_2^2$.

Having shown that K-SVD is a coordinate-descent algorithm, the cost function (21a) is monotonically non-increasing at each iteration and lower-bounded by zero, so it converges asymptotically. Moreover, as the number of possible combinations J are finite, Algorithm 2 always terminates after a finite number of steps, under the assumption that in case of multiple optima at Step 1.2.1 the optimizer J is always chosen in accordance with a predefined criterion, and that the SVD algorithm used at Step 1.1.3 returns the same matrices U, Σ, V for the same given input matrix $\bar{S}$. ■

Due to the finite termination property shown in Theorem 4.1, a termination criterion at Step 2 is to stop when the label vector J does not change after one iteration. Note that Algorithm 2 is only guaranteed to convergence to a local minimum, whether this is also a global one depends on the initial guess J. Moreover, the decrease of the cost function in (21a) can be easily monitored. In fact, from the proof of Eckart-Young-Mirsky theorem for the Frobenius norm, we have that for a given label sequence I the corresponding optimal cost (22) is

$$\sum_{j=1}^{K} \left(\sum_{i \in I_j} \min_{v} \|s_i^* - \Phi_j v - \phi_0^j\|_2^2 \right) = \sum_{j=1}^{K} \sum_{h=m+1}^{n} (\Sigma_{2ii}^j)^2,$$

that is a value that can be immediately computed as a byproduct of the SVDs in (19).

After running K-SVD, the samples θ_i get also labelled by the index sets $I_1, \dots, I_K$. Several multiclass classification methods can be then used to determine the classification function ψ, such as multicategory proximal support vector machines [29] or neural networks [30]. In this paper we use K one-to-all neural classifiers $\psi_i : \mathbb{R}^p \to [0, 1]$, $i = 1, \dots, K$, and then define the nonlinear separation function $\psi : \mathbb{R}^p \to [0, 1]$ such that $\psi(\theta) = \arg\max_{j=1 \dots K} \{\psi_j(\theta)\}$.

In embedded applications the hyper-parameter K is most often dictated by the limitations imposed by the controller unit, in terms of the memory needed to store the K parameterizations and the throughput required to compute ϕ_i, $i = 1, \dots, K$. However, if the application allows tuning K, one can use any heuristic for cluster analysis, for instance the "elbow method".

Remark 4.1: A justification for associating a basis Φ, ϕ_0 to a region of the space of parameters θ stems from the multiparametric analysis of problem (5). For instance, in case matrices A and G do not depend on θ and $b(\theta)$, $g(\theta)$ are affine, (5) is a multiparametric quadratic programming (mpQP) problem [31], whose optimal solution $s^*(\theta)$ is piecewise affine. Therefore, $s(\theta) = H_j \theta + h_j$ on each polyhedral region P_j of the set of parameters θ get partitioned by the mpQP algorithm, $j = 1, \dots, n_p$. If the parametric solution $s(\theta)$ were known, $n_p \leq K$, $m \geq p$, for all $\theta \in P_j$ the parameterization (7) would reproduce the optimal solution if one computes the QR factorization of $H_j = Q_j R_j$ and sets $\Phi_j = [Q_j \ 0_{p-m}]$, $\phi_0^j = h_j$, and $v = [(R_j \theta)' \ 0_{p-m}]'$. ■

Remark 4.2: In K-SVD one may generalize and assign a different number of basis elements to each of the K clusters. The proof of Theorem 4.1 immediately extends to such a more general case, as in (21), as the proof is independent on the number m_j of columns each matrix Φ_j has, as long as those numbers m_j are fixed.

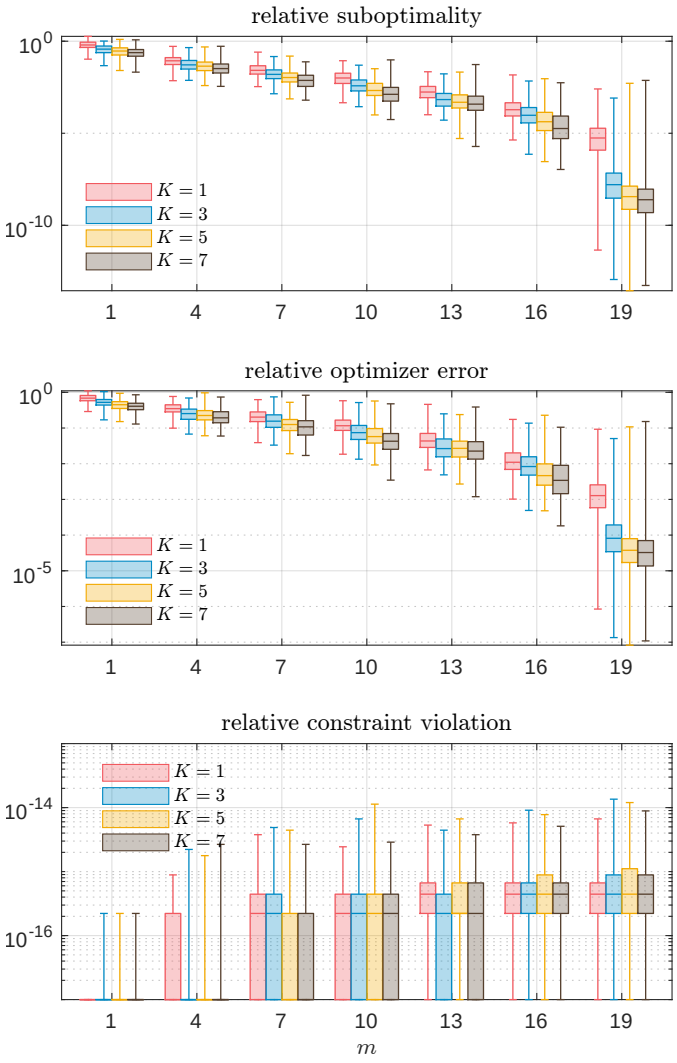

Fig. 2. Distribution of the errors induced by using Algorithm 2 and one-to-all neural classifiers: relative optimality error (top plot), relative difference of optimizers (mid plot), relative maximum violation of the inequality constraints (bottom plot) over 500 validation tests. The case $K = 1$ corresponds to using a single SVD decomposition and to the results shown in Figure 1.

As a further improvement one could also replace the Euclidean distance in (20) with any other reassignment criterion, that can be tailored to modeling assumptions specific for the optimization problem at hand. The theory and methods described in the paper will remain the same. ■

Example 4.1: We consider again the parametric constrained least-squares problem defined in Example 3.1. Using the same $M = 5000$ samples, we run K-SVD for different values of K (number of clusters) and m (number of basis vectors), where the case $K = 1$ corresponds to the single SVD decomposition (11). The clusters generated by K-SVD are separated by training K neural one-to-all classifiers. Each classifier is a neural network composed by 2 layers of 3 neurons each with sigmoidal activation function $\frac{1}{1+e^{-x}}$, cascaded by a sigmoidal output function, corresponding to 28 coefficients. For each cluster h, $h = 1, \ldots, K$, if the training sample θ_i belongs to that cluster then the corresponding label $j_i = 1$, otherwise $j_i = 0$. The standard cross-entropy loss $-(j \log(\hat{j}) + (1-j)\log(1-\hat{j}))$ is used for training the network using the batch nonlinear programming solver implemented in the ODYS Deep Learning Toolset [32]. The total CPU time to run K-SVD and train the neural classifiers in MATLAB R2020a ranges between 12 and 26 seconds on an Intel Core i9-10885H 2.40 GHz machine, with roughly 0.4% to 1.8% of the time spent to run K-SVD.

The obtained results in terms of relative optimality error, relative error, and maximum relative constraint violation are shown in Figure 2. As expected, the quality of the approximation z_r increases if more basis vectors m and partitions K are available. Moreover, increasing K up to 5 or 7 is comparable or superior for some dimensions than adding 3 basis vectors to the $K = 1$ approach. That is an interesting result because increasing the number of optimization variables of the pCLS problem (9) for improving the performance is computationally more intense than evaluate reasonably small neural classifiers while maintaining m unaltered. ■

A. Preservation of selected optimization variables

The approximation method described in the previous sections aims at approximating the reduced vector s of variables obtained after eliminating the equality constraints (1b). On the other hand, an approximation error Δs with respect to s^* propagates on the original equality-constrained vector z as $Q_2 \Delta s$, see (4a). It might be desirable to reduce the approximation error on certain components of z of main interest, say without loss of generality the first $k \leq m$ components $z_1 = [z^1 \ \ldots \ z^k]'$ of z.

Using weighted low-rank approximation methods instead of SVD in Step 4.1 of Algorithm 1 or in Step 1.1.3 of K-SVD to take into account matrix Q_2 in (4) would not be convenient, as it would involve iterative procedures [33]. Instead, we propose to compute a different QR factorization than in (2).

Let $z = \left[\begin{smallmatrix} z_1 \\ z_2 \end{smallmatrix}\right]$, $z_2 = [z^1 \ \ldots \ z^k]'$, $z_2 \in \mathbb{R}^{\ell-k}$. Accordingly, the equality constraints $Cz = e$ become $C_1 z_1 + C_2 z_2 = e$. Let us compute the QR factorization of C_2'

$$C_2' = \bar{Q} \begin{bmatrix} \bar{R}_1 \\ 0 \end{bmatrix}, \quad \bar{Q} = [\bar{Q}_1 \ \bar{Q}_2] \tag{25}$$

and consider the change of variables

$$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & \bar{Q}' \end{bmatrix} z, \quad z = \begin{bmatrix} I & 0 \\ 0 & \bar{Q} \end{bmatrix} y$$

Then, by further splitting $y_2 = [s_3' \ s_2']'$, $s_3 \in \mathbb{R}^{n_e}$, $s_2 \in \mathbb{R}^{\ell-k-n_e}$, we get $e = Cz = C_1 y_1 + [\bar{R}_1' \ 0]\bar{Q}'[0 \ \bar{Q}]\left[\begin{smallmatrix} y_1 \\ y_2 \end{smallmatrix}\right] = C_1 z_1 + \bar{R}_1' s_3$ from which we obtain

$$s_3 = (\bar{R}_1')^{-1}(e - C_1 z_1) \tag{26a}$$

$$s = \begin{bmatrix} z_1 \\ s_2 \end{bmatrix} \text{ free} \tag{26b}$$

Finally, we get the following transformation from s to z

$$z = \begin{bmatrix} y_1 \\ \bar{Q} y_2 \end{bmatrix} = \begin{bmatrix} z_1 \\ \bar{Q}_1 s_3 + \bar{Q}_2 s_2 \end{bmatrix} = Q_2 s + \bar{z} \tag{27}$$

where

$$Q_2 \triangleq \begin{bmatrix} I & 0 \\ -\bar{Q}_1 (\bar{R}_1')^{-1} C_1 & \bar{Q}_2 \end{bmatrix}, \quad \bar{z} = \begin{bmatrix} 0 \\ \bar{Q}_1 (\bar{R}_1')^{-1} e \end{bmatrix}$$

Therefore, we can apply K-SVD or Algorithm 1 to the samples s_i^* obtained from (26b) after substituting z as in (27) in the constrained parametric LS problem. In order to emphasize that the components z_1 of s are better approximated than the remaining components s_2, we can replace the samples s_i^* with $\begin{bmatrix} \tau z_1^* \\ s_2^* \end{bmatrix}$ in computing the bias terms and SVDs, where $\tau > 1$ is a scalar parameter. After executing K-SVD (or Algorithm 1), the bias term ϕ_0 (or ϕ_0^j) needs to be scaled back by dividing its first k components by τ.

V. Variable-reduction methods for MPC

We now want to adopt the methods developed in the previous sections to address the pCLS problems that arise specifically in model predictive control formulations, and to further refine such methods to exploit the particular structure of those problems.

Let us assume that the dynamics of the process are modeled by the linear state-space model

$$x_{k+1} = \mathcal{A}_k(\theta)x_k + \mathcal{B}_k(\theta)u_k \tag{28}$$

with $\mathcal{A} \in \mathbb{R}^{n_x \times n_x}$ and $\mathcal{B} \in \mathbb{R}^{n_x \times n_u}$. Most often, model (28) is obtained by linearizing a nonlinear model of the process around a nominal trajectory, and x_k, u_k represent the predicted deviations from such a trajectory. Typically the nominal state trajectory is the one obtained by applying the sequence of manipulated variables optimized at the previous MPC execution, starting from the current state.

Let $z = [u_0' \; x_1 \; \dots \; u_{T-1} \; x_T]'$ be the vector of optimization variables collecting the sequence of manipulated variables $u_k \in \mathbb{R}^{n_u}$ and the corresponding state variables $x_k \in \mathbb{R}^{n_x}$ over a prediction horizon of future T steps, with $z \in \mathbb{R}^\ell$, $\ell = T(n_x + n_u)$. Vector $\theta \in \mathbb{R}^p$ collects the current estimate $x(t)$ of the plant to control, the current (and possibly future) reference signals to track, and other parameters affecting the prediction model, the performance index, and the constraints.

The equality constraints (1b) embedding model (28) have the following band-matrix form

$$\begin{bmatrix} \mathcal{B}_0 & -I & 0 & & \dots & & 0 \\ 0 & \mathcal{A}_1 & \mathcal{B}_1 & -I & 0 & \dots & 0 \\ \vdots & & & \vdots & & & \vdots \\ 0 & \dots & & 0 & \mathcal{A}_{T-1} & \mathcal{B}_{T-1} & -I \end{bmatrix} z = \begin{bmatrix} -\mathcal{A}_0 x_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{29}$$

where in (29) we have omitted the possible dependence of $\mathcal{A}_k$, $\mathcal{B}_k$ on θ for simplicity. Constraints (29) are a special case of (1c) in which $n_e = Tn_x$, $\ell = T(n_x + n_u)$. We assume that $\mathrm{rank}(C) = Tn_x$.

In MPC problems, the cost function (1a) is usually defined as

$$\begin{aligned} A(\theta) &= \mathrm{blockdiag}(R_{u_0}, R_{x_1}, \dots, R_{u_{(T-1)}}, R_{x_T}) \\ b(\theta) &= \begin{bmatrix} R_{u0}u_{r0} \\ R_{x1}x_{r1} \\ \vdots \\ R_{u(T-1)}u_{r(T-1)} \\ R_{xT}x_{rT} \end{bmatrix} \end{aligned} \tag{30}$$

where $\mathrm{blockdiag}()$ is the block-diagonal matrix of its arguments, $A(\theta) \in \mathbb{R}^{\ell \times \ell}$, $b(\theta) \in \mathbb{R}^\ell$, $\|R_{u_k}(\bar{u}_k - u_k)\|_2^2$ is the penalty on inputs and $\bar{u}_k \in \mathbb{R}^{n_u}$ is the input reference, and similarly $\|R_{x_k}(\bar{x}_k - x_k)\|_2^2$ penalizes the deviation of the states from their reference $\bar{x}_k \in \mathbb{R}^{n_x}$. Penalties on outputs $\|R_{y_k}(\bar{y}_k - y_k)\|_2^2$, with $y_k = \mathcal{C}_k x_k$, $y_k \in \mathbb{R}^{n_y}$, are a special case in which $R_{x_k} = R_{y_k}\mathcal{C}_k$, and $R_{x_k}\bar{x}_k$ is replaced by $R_{y_k}\bar{y}_k$ in (30). Mixed input/state costs involving both u_k, x_k could be also considered, which would lead to having off-diagonal blocks in $A(\theta)$. Note that the weights in (30) may be rectangular matrices, for example in case some input is not weighted R_{u_k} would have less than n_u rows.

The inequality constraints (1c) can collect n_i constraints on inputs, input increments, states, and any other linear constraint involving a linear combination of such variables. In most practical MPC applications, it is customary to treat some of the inequality constraints as *soft*, by introducing a vector $\zeta \in \mathbb{R}^{n_\zeta}$ of additional slack variables as in (14).

In deploying MPC laws in embedded platforms, solving problem (1) efficiently in real time poses two main challenges:

- C1. How to deal with equality constraints (1b) and possibly eliminate them in a numerically robust way, reducing the number of optimization variables (excluding slacks) from ℓ to $n = Tn_u$;
- C2. How to further reduce the number of degrees of freedom from n to m, $m < n$, possibly sacrificing optimality in favor of lighter computations.

Regarding (C1), the pCLS problem (1) is typically reformulated in the so-called *condensed form* by replacing the predicted states x_k with the corresponding prediction

$$x_k = \prod_{i=0}^{k-1} \mathcal{A}_i x_0 + \sum_{i=0}^{k-1} \prod_{j=i+1}^{k-1} \mathcal{A}_j \mathcal{B}_i u_i \tag{31}$$

where $\prod_{i=j}^{j+k} \mathcal{A}_i = \mathcal{A}_{j+k} \dots \mathcal{A}_j$ if $k \geq 1$, or $\mathcal{A}_j$ if $k = 0$, or the identity I_{n_x} if $k < 0$. Such a condensing procedure allows one to recast (1) to a smaller pCLS problem with only n (or $n + n_\zeta$) variables. The main drawback of the condensing (31) is its potentially poor numerical robustness: for example in case of unstable dynamics the substitution (31) easily blows up numerically.

A possible remedy is to prestabilize the dynamical system (28) by setting

$$u_k = \mathcal{K}_k(\theta)x_k + u_k^c \tag{32}$$

and then treat u_0^c, …, u_{T-1}^c as new optimization variables. While this can be easily accomplished offline for linear time-invariant (LTI) systems, in case of linear parameter-varying (LPV) or linear time-varying (LTV) systems the stabilizing set of feedback gains $\{\mathcal{K}_k\}_{k=0}^{T-1}$ must be computed online for the given value of θ. A way to compute such gains is to solve a finite-horizon unconstrained linear-quadratic (LQ) optimal control problem using the Riccati iterations

$$\begin{aligned} \mathcal{P}_k &= \mathcal{Q}_k - \mathcal{A}_k'\mathcal{P}_{k+1}\mathcal{B}_k(\mathcal{R}_k + \mathcal{B}_k'\mathcal{P}_{k+1}\mathcal{B}_k)^{-1}\mathcal{B}_k'\mathcal{P}_{k+1}\mathcal{A}_k \\ &\quad + \mathcal{A}_k'P_{k+1}\mathcal{A}_k \\ \mathcal{K}_k &= -(\mathcal{R}_k + \mathcal{B}_k'\mathcal{P}_k\mathcal{B}_k)^{-1}\mathcal{B}_k'\mathcal{P}_k\mathcal{A}_k \end{aligned} \tag{33}$$

for $k = T-1, \dots, 0$, where $\mathcal{Q}_k = R'_{x_k} R_{x_k}$, $\mathcal{R}_k = R'_{u_k} R_{u_k}$, and $\mathcal{P}_T = R'_{x_T} R_{x_T}$.

Solution methods based on the *non-condensed form*, in which the states $x_1, \dots, x_T$ are kept as optimization variables, have the drawback of requiring general purpose sparse linear-algebra libraries to solve (1) efficiently, or ad-hoc algorithms exploiting the special structure (29), for example when using interior-point methods [12]. A drawback of such an approach comes when one wants to address the second challenge (C2), that is to reduce the number of optimization variables to simplify the mathematical programming problem to solve online, which alters the structure of (29).

The variable elimination method in Section II-A, based on the QR factorization of $C'(\theta)$, offers a numerically robust way to cope with (C1), even when the dynamics (28) is unstable, while preserving the possibility to further reduce the degrees of freedom by means of advanced algorithms such as K-SVD.

The next example shows the numerical properties of the different variable elimination methods mentioned above.

Example 5.1: Consider random linear systems with $n_x = 5$ states, $n_u = 3$ inputs, and prediction horizons $T \in \{10, 20, 30, 40\}$. We first consider *stable* systems, with eigenvalues $\lambda(\mathcal{A})$ such that $\lambda_i \sim \mathcal{U}(0.499, 0.999)$, $i = 1, \dots, n_x$, and then *unstable* systems with real eigenvalues $\lambda_i \sim \mathcal{U}(1, 1.25)$, $i = 1, \dots, n_x$. For each prediction horizon T, a total of 1000 stable and unstable random systems are generated. The weight matrices R_{u_i}, R_{x_i}, $i = 1, \dots, T$ for input and state penalties are constant along the horizon, and change for each problem with their structure chosen such that $\text{rank}(A) \sim \mathcal{D}(\lfloor \frac{Tn_x}{3} \rfloor + Tn_u, \ell)$, and their non-zero elements drawn from $\mathcal{U}(1, 10)$. Figure 3 shows the condition number of the Hessian matrix $A'_r A_r$ associated with the LS problem

$$\min_{s} \ \frac{1}{2}\|A_r s - b_r\|_2^2 \quad \text{s.t. } G_r s \le g_r \tag{34}$$

obtained from (1) by eliminating Tn_x variables by standard condensing (31) ($s = [u'_0 \ \dots \ u'_{T-1}]'$), by prestabilization using dynamic programming (33) ($s = [(u^c_0)' \ \dots \ (u^c_{T-1})']'$, see (32)), and by QR factorization (2) (s such that $z = \bar{z} + Q_2 s$). The relation $\kappa(A'_r A_r) = \kappa^2(A)$ clearly holds. While standard condensing numerically explodes (even for moderately unstable systems), both the LQ prestabilizer and the QR factorization deal effectively with unstable systems, although the latter is considerably more robust. ■

Note that advanced pre-conditioning methods for $C(\theta)$, among which we cite geometric mean scale [34], can be used to improve the numerical robustness of the methods in Example 5.1. However, such an improvement is only marginal and, more importantly, standard condensing would blow up numerically anyway.

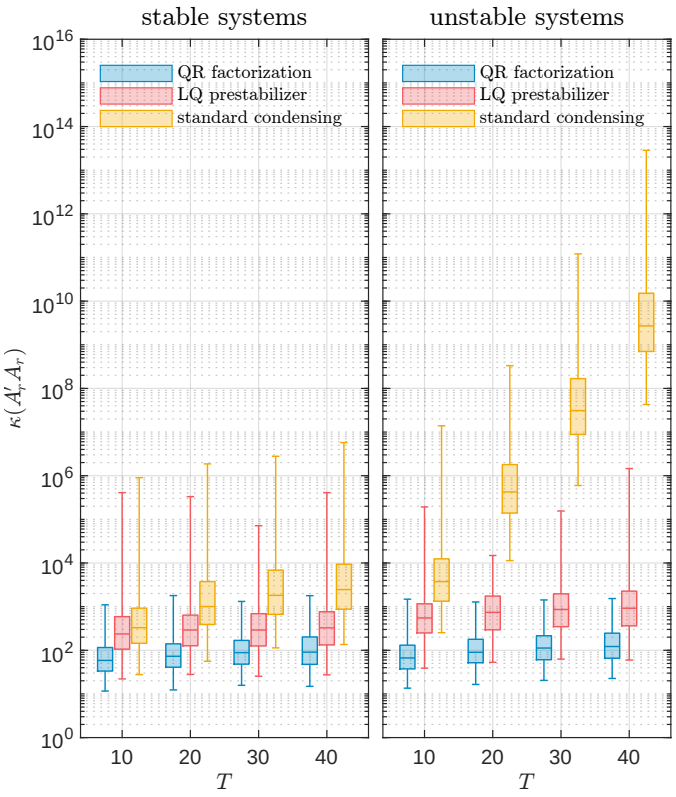

Fig. 3. Distributions of the condition number of $A'_r A_r$, the Hessian matrix of the reduced LS problem after eliminating equality constraints (1b), comparing different condensing techniques. The distribution for each prediction horizon is obtained on a set of 1000 random linear systems. Both stable (*left*) and unstable (*right*) systems are tested, the latter with real eigenvalues between 1 and 1.25.

A. Efficient elimination of equality constraints

When applied to an MPC setting, the numerical robustness of the elimination method in Section II-A comes at the expense of a substantially less efficient MPC routine if compared to (31). A standard QR decomposition algorithm, with full Q computation, introduces an $\mathcal{O}(5T^3 n_x^3)$ complexity term. In this section we present an algorithm tailored to computing the QR decomposition of $C'(\theta)$ very efficiently. Afterwards, we derive a rigorous analysis of the flops (floating-point operations) required by the standard and the proposed QR condensing.

We start by noting that matrix $C(\theta)$ is very sparse. In such a scenario it is preferred to annihilate the columns of R_1 by means of Givens rotations, which selectively introduce zeros below the diagonal, one element at a time [35], [36]. Let $x = \begin{bmatrix} a & b \end{bmatrix}'$ be a vector with $\|x\|_2 = r > 0$; a Givens rotation is a 2-by-2 matrix G_r such that $G_r \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}$. It is easy to show that the Q and R_1 factors of the decomposition of $C'(\theta)$ are sparse as well and such that

$$R_1 = \begin{bmatrix} U_1 & \tilde{R}_1 & 0 & 0 \\ 0 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \tilde{R}_{T-1} \\ 0 & \dots & 0 & U_T \end{bmatrix} \tag{35a}$$

$$\begin{bmatrix} Q_1 \,|\, Q_2 \end{bmatrix} = \begin{bmatrix} Q_{1,1} & \dots & Q_{1,T} \,|\, Q_{2,1} & \dots & Q_{2,T} \end{bmatrix} \tag{35b}$$

where $U_i \in \mathbb{R}^{n_x \times n_x}$ is an upper triangular matrix, $\tilde{R}_j \in \mathbb{R}^{n_x \times n_x}$ is dense, $Q_{1,i} = \begin{bmatrix} \tilde{Q}'_{1,i} & \tilde{U}'_i & 0 \end{bmatrix}'$, $Q_{2,i} = \begin{bmatrix} 0 & L'_i & \tilde{Q}'_{2,i} \end{bmatrix}'$, with $\tilde{Q}_{1,i} \in \mathbb{R}^{(n_u i + n_x(i-1)) \times n_x}$ dense, $\tilde{U}_i \in \mathbb{R}^{n_x \times n_x}$ upper triangular, $L_i \in \mathbb{R}^{n_u \times n_u}$ lower triangular, $\tilde{Q}_{2,i} \in \mathbb{R}^{(n_u + (n_x + n_u)(T-i)) \times n_u}$ dense, and $i = 1, \dots, T$, $j = 1, \dots, T-1$. The sparsity pattern of (29) and (35) provides the basis for developing Algorithm 3, that we call QR-

MPC. Its computational analysis is described by the following lemma, which highlights the efficiency of the method.

Lemma 5.1: Let $C \in \mathbb{R}^{Tn_x \times \ell}$ *be the matrix of equality constraints for problem* (1) *defined as in* (29). *QR-MPC computes the QR decomposition* $\begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_1 \\ 0 \end{bmatrix} = C'$ *used for the numerically robust condensing of problem* (1)*, and enjoys the following properties:*

i) The serial complexity χ_{QR} *of QR-MPC, namely the total number of flops required for termination without any parallel computation, is*

$$\chi_{QR} = T^3(n_x^2 n_u + n_x n_u^2) + 3T^2 n_u n_x(2n_x + n_u + 1) + \mathcal{O}(T) \tag{36}$$

ii) QR-MPC saves a total of

$$\chi_{QRS} - \chi_{QR} = T^3(5n_x^3 + 11n_x^2 n_u + 5n_x n_u^2) - \mathcal{O}(T^2) \tag{37}$$

flops, with χ_{QRS} *the complexity of a standard non-economy QR decomposition algorithm relying on Givens rotations for columns annihilation.*

Proof: i) The complexity of the factorization can be expressed as the sum of terms $\chi_{\text{QR}} = \chi(R) + \chi(Q)$, where we denote by $\chi(Y)$ the complexity of computing matrix Y. Clearly $\chi(R)$ comes from the iterative execution of Step 3.4.3, and $\chi(Q)$ from Step 3.4.4. Let $mp(2n-1)$ be the complexity of the matrix vector product $M = Y\bar{D}$ with $Y \in \mathbb{R}^{m \times n}$, $\bar{D} \in \mathbb{R}^{n \times p}$, then

$$\chi(R) = \sum_{j=1}^{Tn_x} \sum_{i=j+1}^{r_1} 6\Big(n_x \min\big(\lfloor \tfrac{j-1}{n_x} \rfloor + 2, T\big) - j + 1\Big) \tag{38}$$

Let us define the integral operators $\mathcal{I}_0(n) \triangleq \int_0^n \tau\, d\tau$, $\mathcal{I}_1(n) \triangleq \int_0^n (\tau+1)\, d\tau$ and $\mathcal{I}_2(n) \triangleq \int_0^n (\tau+1)^2\, d\tau$; we can get rid of the *minimum* and *integer part* operations in (38) by rearranging the summations such that

$$\begin{aligned}
\chi(R) &= 6\sum_{k=1}^{T}\sum_{j=1}^{n_x}\sum_{i=1}^{kn_u}(2n_x - j + 1) - 6\sum_{j=1}^{n_x}\sum_{i=1}^{Tn_u}(n_x - j + 1) \approx \\
&\approx 6n_u\Big(\mathcal{I}_1(T)\,(2n_x^2 - \mathcal{I}_0(n_x)) - T(n_x^2 - \mathcal{I}_0(n_x))\Big) = \\
&= \frac{9}{2}T^2 n_x^2 n_u
\end{aligned} \tag{39}$$

On the other hand, the complexity of $\chi(Q)$ is

$$\chi(Q) = \sum_{j=1}^{Tn_x} \sum_{i=j+1}^{r_1} 6\big(j + n_u(1 + \lfloor \tfrac{j-1}{n_x} \rfloor) - (n_x + n_u)\lfloor \tfrac{i-j-1}{n_u} \rfloor\big) \tag{40}$$

Let us define $w = n_x + n_u$, by expanding the summations in (40) similarly to the contribution of $\chi(R)$, $\chi(Q)$ can be rewritten as

$$\begin{aligned}
\chi(Q) &= 6\sum_{k=1}^{T}\sum_{j=1}^{n_x}\sum_{h=1}^{k}\sum_{i=1}^{n_u}\big((k-h)w + n_u + j\big) \approx \\
&\approx 6n_u\sum_{k=1}^{T}\Big(k^2 n_x w - k n_x^2 + k\mathcal{I}_1(n_x) - w\sum_{j=1}^{n_x}\mathcal{I}_0(k)\Big) \approx \\
&\approx 6n_u\big(\frac{n_x}{2}w\mathcal{I}_2(T) + n_x(1 - \frac{n_x}{2})\mathcal{I}_1(T)\big) = \\
&= T^3(n_x^2 n_u + n_x n_u^2) + 3Tn_x n_u\big(T(\frac{n_x}{2} + n_u + 1) + (n_u + 3)\big)
\end{aligned} \tag{41}$$

From (39) and (41) we get that $\chi_{\text{QR}} = \chi(R) + \chi(Q) = T^3(n_x^2 n_u + n_x n_u^2) + 3T^2 n_u n_x(2n_x + n_u + 1) + \mathcal{O}(T)$, which proves $i)$.

ii) Let $C \in \mathbb{R}^{m \times n}$ be a full-rank matrix, the serial complexity χ_{QRS} of a standard QR decomposition algorithm based on Givens rotations, and forming Q matrix explicitly (non-economy version) is

$$\chi_{\text{QRS}} = \sum_{j=1}^{n}\sum_{i=j+1}^{m} 6(m + n - j + 1) \approx 6n(m^2 - m - \frac{n^2}{6}) \tag{42}$$

By replacing $m = T(n_x + n_u)$ and $n = Tn_x$ in (42) we obtain

$$\chi_{\text{QRS}} = T^3(5n_x^3 + 6n_x n_u^2 + 12n_x^2 n_u) - 6T^2(n_x^2 + n_u n_x) \tag{43}$$

From (36) and (43) it follows that $\chi_{\text{QRS}} - \chi_{\text{QR}} = T^3(5n_x^3 + 11n_x^2 n_u + 5n_x n_u^2) - \mathcal{O}(T^2)$, which proves $ii)$. ■

Theorem 5.1: Let (1a)–(1b) be the equality constrained least-squares formulation of an MPC problem with n_x states, n_u inputs, a prediction horizon of T steps, $C(\theta)$, $e(\theta)$ as in (29), and $A(\theta)$, $b(\theta)$ as in (30). Let

$$\min_s \ \frac{1}{2}\|A_r(\theta)s - b_r(\theta)\|_2^2 \tag{44}$$

be the reformulation of (1a)–(1b) in *condensed form* after removing equalities (1b), where $s \in \mathbb{R}^{Tn_u}$, $A_r(\theta) \in \mathbb{R}^{\ell \times Tn_u}$ and $b_r(\theta) \in \mathbb{R}^{\ell}$. Then, the serial complexity for computing $A_r(\theta)$, $b_r(\theta)$ is

i) χ_s in case of *standard condensing*, with

$$\chi_s = T^2(n_x^2 n_u - \frac{n_x n_u}{2}) + \mathcal{O}(T) \tag{45}$$

ii) χ_q in the case of *QR condensing*, with

$$\begin{aligned}
\chi_q = {} & T^3(n_x^2 n_u + n_x n_u^2) + T^2(n_x^2(6n_u + 2) + n_u^2(3n_x + 1) + \\
& + 2n_u n_x - 2(n_x + n_u)) + \mathcal{O}(T)
\end{aligned} \tag{46}$$

Proof: i) We first note that the states replacement (31) is equivalent to impose the coordinate transformation $z = Fs + f$ with $F \in \mathbb{R}^{\ell \times Tn_u}$, $f \in \mathbb{R}^{\ell}$ defined as

$$F = \begin{bmatrix}
I_{n_u} & 0 & \dots & \dots & 0 \\
\mathcal{B} & 0 & \dots & \dots & 0 \\
0 & I_{n_u} & 0 & \dots & 0 \\
\mathcal{A}\mathcal{B} & \mathcal{B} & 0 & \dots & 0 \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
0 & \dots & \dots & 0 & I_{n_u} \\
\mathcal{A}^{T-1}\mathcal{B} & \mathcal{A}^{T-2}\mathcal{B} & \dots & \mathcal{A}\mathcal{B} & \mathcal{B}
\end{bmatrix}, \ f = \begin{bmatrix}
0 \\ \mathcal{A} \\ 0 \\ \mathcal{A}^2 \\ \vdots \\ 0 \\ \mathcal{A}^T
\end{bmatrix} x_0 \tag{47}$$

where in (47) we have omitted the dependence of $\mathcal{A}$ and $\mathcal{B}$ from k, and therefore the condensed problem corresponds to (5) after having replaced $Q_2 \leftarrow F$ and $\bar{z} \leftarrow f$. Clearly $CF = 0$ holds, and f solves (1b). The complexity χ_1^s associated with forming f is equivalent to T matrix-vector products of dimension n_x, thus

$$\chi_1^s = Tn_x(2n_x - 1) \tag{48}$$

Let $M = Y\bar{D}$ be a matrix-vector product with $Y \in \mathbb{R}^{n_x \times n_x}$, $\bar{D} \in \mathbb{R}^{n_x \times n_u}$; then, constructing F has a complexity χ_2^s equivalent to $T-1$ repetitions of such product, that is

$$\chi_2^s = (T-1)n_x n_u(2n_x - 1) \tag{49}$$

The cost χ_3^s for computing $A(\theta)F$ comes instead from $\sum_{i=1}^{T} i$ repetitions of $M = Y\bar{D}$, such that

$$\chi_s^3 = T^2(n_x^2 n_u - \frac{n_x n_u}{2} + T n_x n_u(2n_x - 1) \quad (50)$$

and the complexity of $b - F\bar{z}$ is

$$\chi_4^s = T n_x(2n_x - 1) + \ell \quad (51)$$

We finally prove $i)$ by simply computing $\chi_s = \sum_{i=1}^{4} \chi_i^s$.

$ii)$ From Lemma 5.1 we know the complexity of factorizing $C(\theta)$, and thus we are left with computing the flops required to form the cost function (5a). We can exploit the sparsity of R_1 in (35a) when solving the linear system $\bar{s} = (R_1')^{-1}e$ by forward substitution. That is equivalent to run the update step $\bar{s}_i \leftarrow (e_i - R'_{i,k:i-1}\bar{s}_{k:i-1})/R_{i,i}$, with $k = 1 + \max(0, n_x\lfloor\frac{i-1}{n_x}\rfloor - 1)$ and $i = 1, \dots, n$, which takes

$$\chi_1^q = n_x^2(3T - 2) + T n_x \quad (52)$$

flops to be executed. For the complexity χ_2^q of matrix-vector product $\bar{z} = Q_1\bar{s}$ we exploit instead the sparsity of Q_1 in (35b). Let us recall that the flops required for computing $c = Yd$ with Y sparse, is well approximated by $2n_z$, with n_z the number of non-zero elements of Y. Then, we get the complexity

$$\chi_2^q = n_x^2(T^2 + 2T - 1) + n_x(n_u(T^2 - 1) - 2T) \quad (53)$$

The structure of $A(\theta)$ in (30) is first used to derive the complexity χ_3^q for computing $b(\theta) - A(\theta)\bar{z}$ that is

$$\chi_3^q = 2(n_u^2 + n_x^2)T + \ell \quad (54)$$

and then, combined with Q_2 sparsity, to derive the complexity χ_4^q of the matrix-vector product $A(\theta)Q_2$, which is

$$\chi_4^q = T^2(n_x^2 + n_u^2 - 2(n_x + n_u)) + T(2(n_u^2 + n_x^2) - n_u - n_x) \quad (55)$$

Computing $\chi_q = \chi_{\text{QR}} + \sum_{i=1}^{4} \chi_i^q$ proves $ii)$. ■

Remark 5.1: The memory allocation of QR-MPC can be reduced by overwriting the upper triangular part of $C(\theta)'$ with R_1, thus saving the allocation space for R. This is easily achieved by computing the plane rotation on the pair $(C(\theta)_{k,i-1}, C(\theta)_{k,i})$, and applying it to $C'(\theta)$ at Step 3.4.3. Note that any robust computation of the c and s factors, see [37] for instance, can be used in place of Steps 3.4.1 and 3.4.2 to avoid over/underflow. In addition, QR-MPC factorization can be modified to take explicitly into account possible zero-entries of $\mathcal{A}$ and $\mathcal{B}$. If $|C_{k,i-1}| < \epsilon_0$, we get the rotation $G_r = \left[\begin{smallmatrix} 0 & -1 \\ 1 & 0 \end{smallmatrix}\right]$ which corresponds to a signed swap of $C(\theta)$ and Q columns, saving therefore the flops needed for the matrix-vector products of Steps 3.4.3 and 3.4.4. ■

Theorem 5.1 proves the serial complexity of reformulating the cost function (1a) in the presence of the equality constraints (1b), when either *standard* or *QR* condensing is adopted. The flops for reformulating inequalities (1c) are instead neglected because, most often, in MPC applications states and inputs are constrained by simple bounds, that is $G = \begin{bmatrix} I_\ell & -I_\ell \end{bmatrix}'$, in which case computing (5b) costs approximately n_i flops, regardless of the condensing method used. If instead constraints involving the linear combination of states and/or inputs are present, for instance output constraints, the cost

Algorithm 3 QR-MPC factorization for efficient condensing.

Input: Matrix $C(\theta) \in \mathbb{R}^{Tn_x \times \ell}$ of equality constraints (1b), n_u, n_x, T, zero-detection tolerance ϵ_0.

1. $Q \leftarrow I_\ell$;
2. $R \leftarrow C(\theta)'$;
3. **for** $j = 1, \dots, Tn_x$ **do**:
   3.1. $k \leftarrow \lfloor\frac{j-1}{n_x}\rfloor$;
   3.2. $r_1 \leftarrow j + n_u(k+1)$;
   3.3. $r_2 \leftarrow n_x \min(k+2, T)$;
   3.4. **for** $i = r_1, \dots, j+1$ **do**:
      3.4.1. $q_1 \leftarrow 1 + (n_x + n_u)\lfloor\frac{i-j-1}{n_u}\rfloor$;
      3.4.2. **if** $|R_{i,k}| > \epsilon_0$ **do**:
         3.4.1. $c \leftarrow R_{i-1,k}/\sqrt{R_{i-1,k}^2 + R_{i,k}^2}$;
         3.4.2. $s \leftarrow -R_{i,k}/\sqrt{R_{i-1,k}^2 + R_{i,k}^2}$;
         3.4.3. $R_{i-1:i,j:r_2} \leftarrow \begin{bmatrix} c & s \\ -s & c \end{bmatrix}' R_{i-1:i,j:r_2}$;
         3.4.4. $Q_{q_1:r_1,i-1:i} \leftarrow Q_{q_1:r_1,i-1:i} \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$;
4. **end**.

Output: Orthogonal matrix $Q \in \mathbb{R}^{\ell \times \ell}$ and upper triangular matrix $R \in \mathbb{R}^{\ell \times n_e}$ such that $QR = C(\theta)'$.

of reformulating all the closed half-spaces defining a specific constraint along the prediction horizon is $n_u(2n_p - 1)\sum_{i=1}^{T} i$ flops, assuming the worst-case in which the constraint is imposed at each time step, where $n_p \in \{1, \dots, n_x + n_u\}$ is the number of variables involved in the definition of the constraint itself.

We finally note that the computational assessment of Theorem 5.1 remains valid even in case a vector $\zeta \in \mathbb{R}^{n_\zeta}$ of slack variables is introduced for softening (some of) the constraints. Indeed, when reformulating the pCLS problem (1) such that $\begin{bmatrix} z' & \zeta' \end{bmatrix}'$ is the extended vector of optimization variables, equalities (1b) are replaced by

$$\begin{bmatrix} C & 0_{n_\zeta} \end{bmatrix} \begin{bmatrix} z \\ \zeta \end{bmatrix} = e \quad (56)$$

and hence the QR factorization of C' in (2) becomes

$$\begin{bmatrix} C' \\ 0_{n_\zeta} \end{bmatrix} = \begin{bmatrix} Q & 0 \\ 0 & I_{n_\zeta} \end{bmatrix} \begin{bmatrix} R \\ 0'_{n_\zeta} \end{bmatrix} \quad (57)$$

From (57) we know that the extended optimizer vector for the pCLS problem (5) without equality constraints is $\begin{bmatrix} s' & \zeta' \end{bmatrix}'$, which means slack variables are kept as free variables, and one can derive the reduced pCLS by applying QR-MPC to (1) and extend the optimizer vector afterwards.

Example 5.2: From Example 5.1 it is clear that a numerically robust method to eliminate equalities is *mandatory* when the system dynamics are unstable. Indeed, condensing by means of (31) blows-up numerically even in double precision. When instead $\mathcal{A}_k$, $k = 1, \dots, T$ are stable, standard condensing is appealing because of the reduced throughput required to compute the (F, f) pair and form (34), see the results of Theorem 5.1. Here we want to show that even in this scenario, a robust elimination of equalities may still

be valuable despite the increase in computational complexity. Consider parameters n_x, n_u, $\text{rank}(A)$, λ_j, $\mathcal{A}_k$, $\mathcal{B}_k$, $R_{u_{k-1}}$, R_{x_k}, $k=1,\dots,T$, $j=1,\dots,n_x$ defined according to the random pCLS setup of Example 5.1, and restricted to the sole case of stable dynamics. We enforce box constraints on all the inputs, that is $u^- \leq u_k \leq u^+$, $k=1,\dots,T$, with $u^-_{(j)} \sim \mathcal{U}(-\delta,\delta)$, $u^+_{(j)} \sim \mathcal{U}(u^-_{(j)}, u^-_{(j)}+\delta)$, $j=1,\dots,n_u$ and $\delta=1$. We also constrain the states such that $x^- \leq x_i \leq x^+$, $i=1,\dots,T$ and

$$x^-_{(j)} = \begin{cases} \sim \mathcal{U}(-\delta,\delta) & \text{if } x^-_{(j)} \text{ is imposed} \\ -\infty & \text{otherwise} \end{cases}$$
$$x^+_{(j)} = \begin{cases} \sim \mathcal{U}(x^-_{(j)}, x^-_{(j)}+\delta) & \text{if } x^+_{(j)} \text{ is imposed}, x^-_{(j)} \neq -\infty \\ \sim \mathcal{U}(-\delta,\delta) & \text{if } x^+_{(j)} \text{ is imposed}, x^-_{(j)} \equiv -\infty \\ \infty & \text{otherwise} \end{cases} \tag{58}$$

with $j=1,\dots,n_x$. For each problem we randomly select which upper and lower bounds on the states are enforced in such a way that the total number of constraints is $n_i = 2n_u + m_x$, with $m_x \sim \mathcal{D}(\lfloor \frac{n_x}{2} \rfloor, \lfloor \frac{4n_x}{3} \rfloor)$. We consider two procedures for solving such generated MPC problems. The first, denoted "MPC$_{\text{STD}}$" is based on standard condensing and solves the reduced pCLS problem (34) constructed by applying the variable transformation $z = Fs + f$, see (47). In the second one, which is denoted by "MPC$_{\text{QR}}$", one must first factorize C' (by means of the QR-MPC algorithm), compute the transformation $z = Q_2 s + Q_1 (R_1')^{-1} e$, and compute the matrices defining (34), which is ultimately solved. In both cases we solve the reduced pCLS problem for s by means of the Alternating Direction Method of Multipliers (ADMM), the reader is referred to [38] for mathematical details. Let us define s^{j+1} and h^{j+1} the sequential primal updates on the directions s and h at the j-th iteration, and w^{j+1} the dual update, the steps of an over-relaxed ADMM applied to (34) are:

$$\begin{aligned} s^{j+1} &= (A_r' A_r + \rho G_r' G_r)^{-1} (A_r' b_r - \rho G_r' (s^j + w^j - g_r)) \\ h^{j+1} &= \left[\alpha_1 h^j - \alpha(G_r s^{j+1} - g_r) - w^j\right]_+ \\ w^{j+1} &= w^j + \alpha(G_r s^{j+1} + h^{j+1} - g_r) + \alpha_1 (h^{j+1} - h^j) \end{aligned} \tag{59}$$

with $\alpha \in (0,2)$ the relaxation parameter, and $\rho > 0$ the weight on the augmented Lagrangian. We consider a *single precision* floating-point implementation of MPC$_{\text{STD}}$ and MPC$_{\text{QR}}$, with ADMM running for a fixed amount $p=200$ of iterations. The quality of the solution reached after p iterations is evaluated in terms of the optimizer error $\mu_o(z^p)$ and the violation $\mu_f(z^p)$ of the constraints with respect to the original problem (1), defined as:

$$\mu_o(z^p) = \frac{\|z^* - z^p\|}{\|z^*\|}, \quad \mu_f(z^p) = \max\left\{ J \begin{bmatrix} |Cz^p - e| \\ (Gz^p - g)_+ \end{bmatrix} \right\} \tag{60}$$

where $J = \text{diag}(d_J)^{-1}$, $d_J = [\|[C_1\ e_1]\|_2\ \dots\ \|[C_{n_e}\ e_{n_e}]\|_2\ \|[G_1\ g_1]\|_2\ \dots\ \|[G_{n_i}\ g_{n_i}]\|_2]'$. Clearly we have that $z^p = Fs^p + f$ for MPC$_{\text{STD}}$, and $z^p = Q_2 s^p + Q_1 (R_1')^{-1} e$ for MPC$_{\text{QR}}$. Figure 4 shows the distribution of optimality and feasibility when solving $n_t = 1000$ random MPC problems for each T. The bottom plot shows also the total computational

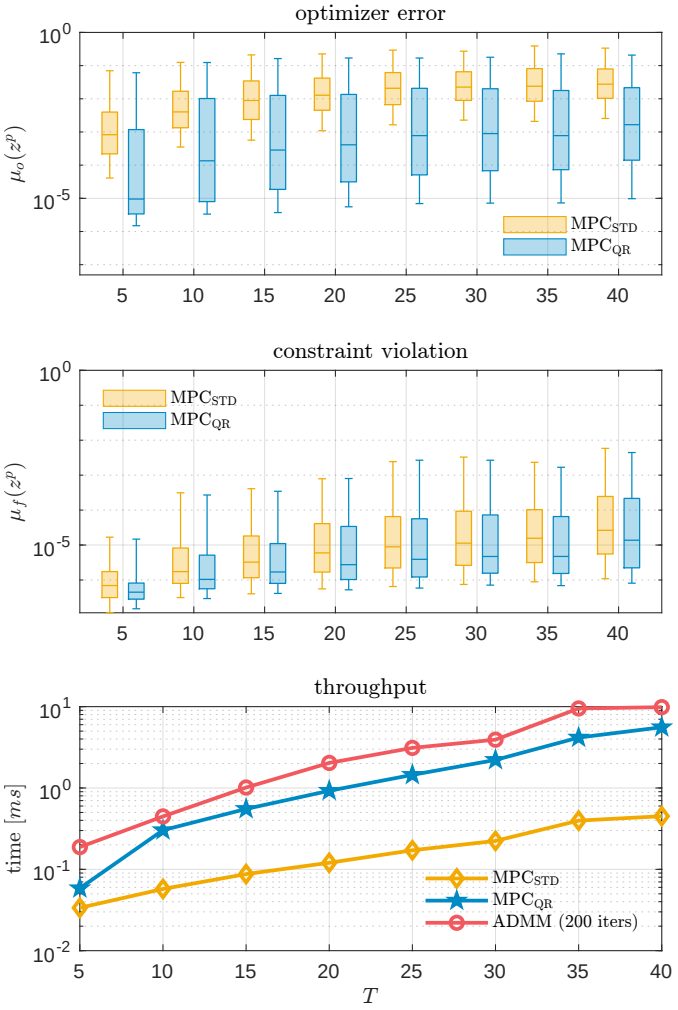

Fig. 4. Comparison MPC$_{\text{STD}}$ and MPC$_{\text{QR}}$ in terms of solution quality and computational load on a set of $n_t = 1000$ random *stable* MPC problems. Algorithms are coded in single precision floating-point arithmetic, and the reduced pCLS (34) is solved by a fixed number of ADMM iterations ($p = 200$), see (59). Distribution of the optimizer error (*top*), distribution of the maximum constraint violation (*mid*) and shifted geometric mean ($h_t = 10$) of the time required to both condense and solve (with ADMM) the reduced problem (*bottom*) are reported.

load of both MPC routines, obtained as the *shifted geometric mean*

$$\nu(t) = \left(\prod_{i=1}^{n_t} (t_i + h_t) \right)^{1/n_t} \tag{61}$$

where $t \in \mathbb{R}^{n_t}$ is the array collecting the execution time over the n_t different CLS problems, and h_t is a shift parameter. We assume $h_t = 10$ in this example. MPC$_{\text{QR}}$ improves both the quality metrics for the optimal solution, especially the optimizer error which is reduced by more than one order of magnitude. Such results undoubtedly show that the robust elimination is not only an option to deal with unstable systems, but it is beneficial even when standard condensing is reliable. The price for such robustness is an increase in the total throughput of the MPC routine, as shown in the bottom plot of the figure. We stress that in our setup the computational load for solving (34) is independent from the method used to construct the reduced LS problem, because the optimization

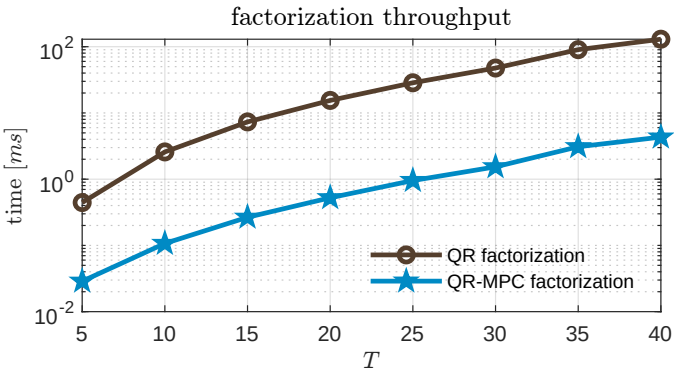

Fig. 5. Computational evaluation of the proposed variable elimination method. QR-MPC (*blue*) is compared to a standard (*brown*) non-economy QR factorization, see [36, Algorithm 5.2.4]. The time shown for each prediction horizon T is the shifted geometric mean (61), with $h_t = 10$, computed over $n_t = 1000$ random CLS problems.

problem has the same dimension and it is solved by running a fixed amount of ADMM iterations. This means that the time difference is only due to the increased complexity of computing the variable transformation and forming (34). The relative impact on the total time is therefore determined by the overhead of the optimization algorithm used to solve (34). Moreover, algorithms whose convergence rate depends on $\kappa(A_r)$ will converge faster for an MPC$_{\text{QR}}$ implementation, with a consequent shrinking of the computational gap.

Lastly, we highlight the efficiency of the QR-MPC algorithm by comparing its throughput with respect to a standard non-economy QR decomposition based on Givens rotations, see [36, Algorithm 5.2.4]. Figure 5 shows the timing results on the same set of MPC problems used in Figure 4. Note that for the implementation of QR-MPC we have taken into account throughput and memory efficiency ideas proposed in Remark 5.1. This is the building block of the MPC$_{\text{QR}}$ routine and it is indeed the main contribution to the computation time shown by the bottom plot of Figure 4. By comparing the two figures we can conclude that without QR-MPC the computational gap between MPC$_{\text{STD}}$ and MPC$_{\text{QR}}$ would have been worse by up to an order of magnitude. ■

B. Reduction of the number of variables

In MPC applications, it is often observed that reducing the number of degrees of freedom to $m < n$ does not compromise closed-loop performance, while it instead simplifies the online optimization problem. As mentioned in Section I, a way commonly used in MPC to reduce the number of optimization variables is *move blocking*, which consists of keeping the input signal u_k constant between prediction steps k_i and $k_{i+1} - 1$, $i = 0, \dots, m_u$, with m_u the number of steps where the input signal is free to change, $k_0 = 0$ and $k_{m_u} = T$. In this way, the number of optimization variables is reduced from n to $m = m_u n_u$. Frequently, $k_i = i$ for $i = 0, \dots, m_u - 1$, i.e., the input signal is free to move within a *control horizon* of m_u steps and then is frozen afterwards until the end of the prediction horizon T. For example, having a control horizon $m_u \leq 5$ is enough in most MPC problems, due to the receding-horizon mechanism of MPC. Reference governors [39], [40] are an extreme case in which, thanks to the presence of a prestabilizing feedback loop, using the control horizon $m_u = 1$ to optimize the reference signals achieves the task of fulfilling constraints and maintaining good closed-loop performance.

Basis functions have been suggested to reduce the number of degrees of freedom by parameterizing the input sequence as

$$u_k = \sum_{i=1}^{m} v_i \phi_i(k) \tag{62}$$

where $\{\phi_i(\cdot)\}_{i=1}^{m}$ is a given basis. Blocking moves can be seen as a special case in which the basis functions are the unit-step function $\mathbb{1}(k - k_i) = 1$ for $k \geq k_i$, or zero otherwise,

$$u_k = \sum_{i=1}^{m} v_i \, \mathbb{1}(k - k_i) \tag{63}$$

The use of Laguerre polynomials to parameterize the input sequence was investigated in [41], which also examines other choices of orthogonal functions. Laguerre polynomials were also used to parameterize both the input and state sequences [42], in a way that these are invariant to time shifts and hence amenable for warm-starting the new optimization problem with the shifted solution of the previous problem.

Note that the use of basis functions as in (7) becomes necessary when the free optimization variables s have no system theoretical meaning, such as in case the QR method described in Section V-A is used to eliminate equality constraints. For such a method, there is no direct and intuitive approach like blocking moves to reduce the number of free variables, unless blocking-moves are introduced before removing equality constraints.

As we work in discrete-time over a finite horizon of T steps, "basis functions" are nothing else than just vectors $\phi_i \in \mathbb{R}^n$. Therefore, finding a basis of m elements to parameterize the input sequence $\{u_k\}_{k=0}^{T-1}$ is equivalent to finding a matrix $\Phi \in \mathbb{R}^{Tn_u \times m}$ with full column-rank. The SVD-based method described in Section III, or alternatively its generalization in Section IV, can be immediately applied for this task. The approach described in Section IV-A can be also applied to preserve the components of the optimization vector corresponding to the applied command input u_0 as much as possible with respect to the original solution of (1).

Variable reduction through PCA of the condensed Hessian was also proposed to simultaneously reduce the optimizer size and limit the ill-conditioning effects of unstable systems. However, this needs an online SVD when A or C depend on θ, similarly to [19]–[21], and the factorization is performed on an ill-conditioned matrix.

Regarding the feasibility of the reduced pCLS problem, we can apply the techniques presented in Section III-B. In particular, if a feasible input and state trajectory z_f is available, then the corresponding vector s_f can computed and one can set $\phi_0 = s_f$ or use the parameterization (13).

VI. Examples of application to MPC problems

We apply the methods developed in the previous sections to the classical benchmark problem of MPC of a continuous stirring tank reactor (CSTR) [7]. The system has $n_x = 2$ states

performance index	value	MPC setting
$J_{\text{exact},2}$	434.5290	$n=2$, exact solution
$J_{\text{exact},3}$	458.1742	$n=3$, exact solution
$J_{\text{exact},4}$	437.4142	$n=4$, exact solution
$J_{\text{exact},5}$	399.0286	$n=5$, exact solution
$J_{\text{exact},8}$	325.1723	$n=8$, exact solution
$J_{\text{exact},20}$	276.3555	$n=20=T$, exact solution
$J_{\text{svd},2}$	2335.6096	$n=20$, $m=2$, $K=1$
$J_{\text{svd},3}$	1338.1678	$n=20$, $m=3$, $K=1$
$J_{\text{svd},4}$	303.8875	$n=20$, $m=4$, $K=1$
$J_{\text{ksvd},2}$	290.7679	$n=20$, $m=2$, $K=10$
$J_{\text{ksvd},3}$	278.5647	$n=20$, $m=3$, $K=10$
$J_{\text{ksvd},4}$	276.9783	$n=20$, $m=4$, $K=10$

TABLE I
CLOSED-LOOP PERFORMANCE INDEX J COMPUTED AS IN (64) FOR DIFFERENT MPC SOLUTION METHODS.

(the temperature of the reactor T_r [K] and the concentration C_A [kmol/m^3] of reactant A) and $n_u = 1$ input (coolant temperature T_c [K]), see [43]. The goal is to make C_A track a given reference C_A^{ref} by applying a linear parameter-varying MPC with sample time $T_s = 0.5$ h. Accordingly, the nonlinear dynamics are linearized around the current state $x(k)$ and previous input $u(k-1)$, then converted to discrete-time by first-order Euler approximation.

We assume that the feed-stream concentration C_{A_f} and temperature T_f, that are measured disturbances, are kept constant, namely C_{A_f}=10 kmol/m^3 and T_f = 298.15 K. From now on, units will be omitted where obvious. The following constraints are imposed on the manipulated input T_c:

$$285.15 \le T_{c,k} \le 312.15$$
$$-2 \le T_{c,k} - T_{c,k-1} \le 2$$

In order to handle constraints and weights on input increments $\Delta u_k = u_k - u_{k-1}$, the 2$^{\text{nd}}$-order model is extended with the additional dynamics $u_k = u_{k-1} + \Delta u_k$. We choose a prediction horizon $T = 20$, preview on the reference signal of five steps (namely, $C_{A,k}^{\text{ref}}, \dots, C_{A,k+5}^{\text{ref}}$ are known at time step k), weight matrix $R_{\delta u} = 0.04$ on input increments, $R_x = \left[\begin{smallmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{smallmatrix}\right]$ on the extended state $[x_k' \ u_{k-1}]'$.

Move blocking to n free control moves is obtained by setting $\Delta u_k = 0$ for all $k = n, n+1, \dots, T-1$. Table I shows the closed-loop performance results obtained for different values of the control horizon n, starting from the steady-state initial condition $u_{-1} = 298.15$ and $x_0 = [311.267 \ 8.5695]'$, quantified by the following index

$$J = \sum_{k=0}^{N} (C_A(k) - C_{A,k}^{ref})^2 + 0.04(T_{c,k} - T_{c,k-1})^2 \qquad (64)$$

where N is the total number of closed-loop simulation steps.

The closed-loop trajectories corresponding to $n = T = 20$ and $n = 3$ are depicted in Figure 6, corresponding to the costs $J_{\text{exact},20}$ and $J_{\text{exact},3}$ reported in Table I, respectively.

In order to reduce the number of optimization variables from $n = T = 20$ to $m < T$, we generate a dataset of $M = 10{,}000$ samples of vector $\theta = [T_{r,k} \ C_{A,k} \ T_{c,k-1} \ C_{A,k}^{\text{ref}}, \ \dots, \ C_{A,k+5}^{\text{ref}}]'$. The samples are generated by running a closed-loop simulation under the MPC controller designed above with random step signal sampled from the uniform distribution between 2 and 9

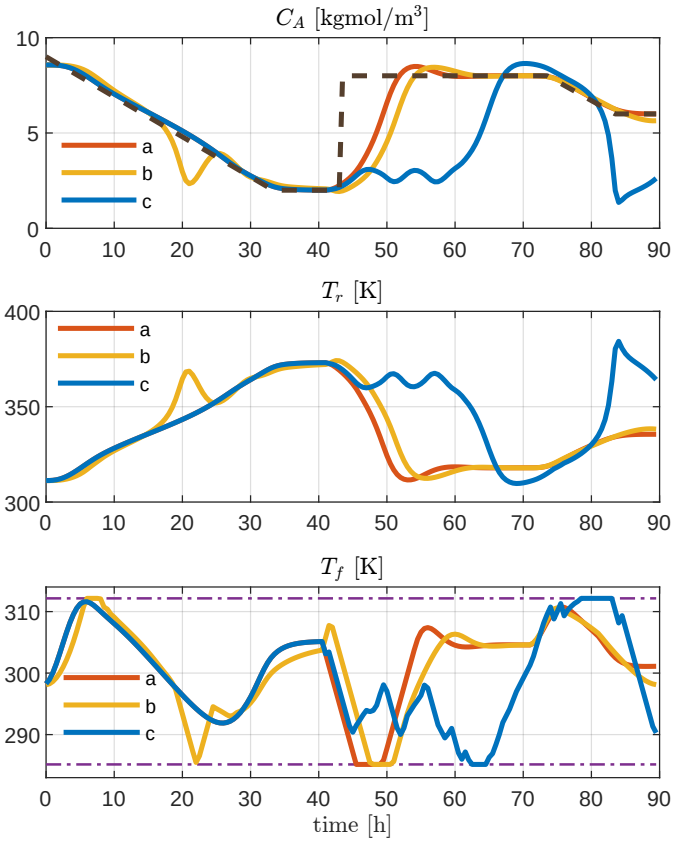

Fig. 6. Closed-loop MPC results: (a) exact solution with control horizon $n = T = 20$; (b) exact solution with control horizon $n = 3$ (i.e., blocking moves); and (c) single SVD on the full problem ($n = T$) with $m = 3$. The results obtained by applying K-SVD ($K = 10$) for $m = 2, 3, 4$ and single SVD for $m = 4$ on the full problem ($n = T$) are almost indistinguishable from plot (a). The dashed line in the top plot is the reference trajectory for C_A, the dot-dashed lines in the bottom plot are the limits imposed on T_f.

kgmol/m^3 as reference signal C_A^{ref}, where at each sample step the set-point has probability 1% of switching. For each sample θ_i, the QR factorization (25) is applied to the MPC problem formulation with $n = T = 20$ as a constrained least-squares problem, so that the sequence $u_i^* = [\Delta u_k^* \ \dots \Delta u_{k+T-1}^*]_i'$ of optimal input increments is computed as in (4a) from the solution $z_i^* = [z_{1i}^* \ s_{2i}^*]'$ of the reduced-order problem (5). This is augmented with a slack variable ζ as in (16) to avoid any infeasibility, which is heavily penalized by adding $(10^5\zeta)^2$ in the least-squares problem. As suggested in Section IV-A, SVDs are applied to samples $\left[\begin{smallmatrix} \tau z_{1i}^* \\ s_{2i}^* \end{smallmatrix}\right]$, with $\tau = 20$ in order to favor the reconstruction of the first input increment z_1^*.

We first compute a single SVD decomposition on the generated dataset to find a basis Φ and offset ϕ_0 for $m = 2, 3, 4$. The closed-loop simulation results obtained by minimizing with respect to $v \in \mathbb{R}^2$ are also reported in Figure 6, corresponding to the performance indices $J_{\text{svd},2}$, $J_{\text{svd},3}$, and $J_{\text{svd},4}$ reported in Table I, respectively.

Next, we apply the K-SVD approach, running Algorithm 2 with $K = 10$ and $m = 2, 3, 4$. For each $j = 1, \dots, K$, we train a one-to-all neural classifier with two hidden layers of 6 neurons each and sigmoidal activation function, cascaded by a sigmoidal output function, corresponding to 109 coefficients

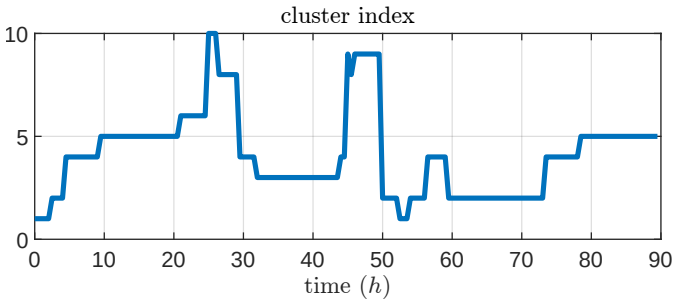

Fig. 7. Index of basis chosen when applying MPC based on K-SVD with $K = 10$, $m = 3$.

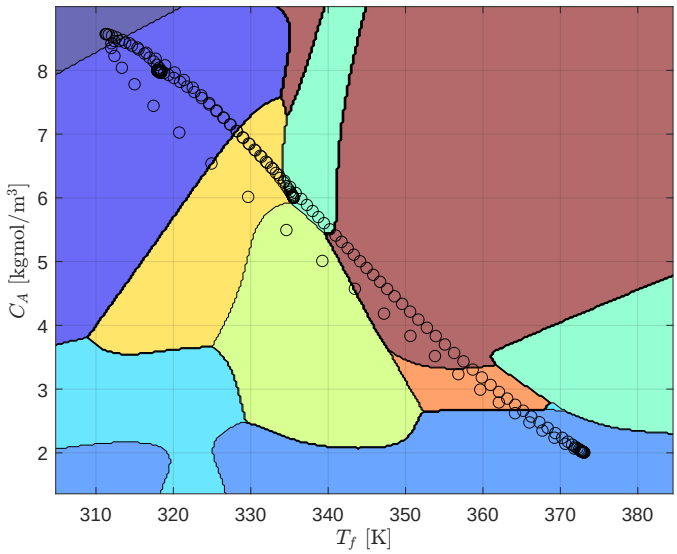

Fig. 8. Section in the (T_r, C_A) space of the five-dimensional partition induced by the neural classifiers for separating the clusters identified by K-SVD with $K = 10$, $m = 3$. The values encountered during the closed-loop simulation are shown as black circles.

to learn per classifier. The cross-entropy loss is used during training, which is accomplished in a total of about 4 minutes on the same machine used in Example 4.1 using the ODYS Deep Learning Toolset [32].

The corresponding closed-loop simulation results are almost indistinguishable from the benchmark performance of exact solution with control horizon $n = T = 20$ depicted in Figure 6, as confirmed by the performance indices $J_{\text{ksvd},2}$, $J_{\text{ksvd},3}$, and $J_{\text{ksvd},4}$ reported in Table I. It is apparent that reducing the number of degrees of freedom by K-SVD provides much better results that by a single SVD and by move blocking: for example, with $m = 3$ degrees of freedom we have $J_{\text{ksvd},3} = 276.9783 \approx J_{\text{exact}}$, $J_{\text{svd},3} = 1338.1678$, $J_{\text{exact},3} = 458.1742$. For $K = 10$ and $m = 3$, the index of the basis function chosen at each step by selecting the neural classifier with the largest value is depicted in Figure 7, while Figure 8 shows the section in the (T_r, C_A) space of the partition in $\mathbb{R}^9$ induced by the classifiers, for the remaining coordinates set to $u_{k-1} = 300.24$, $C^{\text{ref}}_{A,k} \approx \ldots \approx C^{\text{ref}}_{A,k+5} \approx 5.62$ (these values are the average values computed on the training dataset). The figure also shows the values encountered during the closed-loop simulation reported in Figure 6.

Finally, we test how the MPC algorithm performs without SVD approximations for the same number of degrees of freedom, that is by setting the control horizon $n = m < T$. The results for $n = m = 3$ are also shown in Figure 6, corresponding to the closed-loop index $J_{\text{exact},3}$ of Table I.

VII. Conclusions

In this paper we have investigated numerical methods for reducing the number of variables in pCLS problems, such as those encountered in MPC formulations. The K-SVD algorithm is a general method that extends linear PCA to handle parameter-dependent sample vectors, still keeping the resulting approximation of the vectors a linear combination of the principal components, so that the reduced problem remains a pCLS. We have also shown that the QR factorization is a much more numerically-stable approach than standard condensing, and provided a specialized QR-based equality elimination scheme that exploits the special structure of pCLS' arising from MPC.

Further research will be devoted to address issues of recursive feasibility and closed-loop stability of MPC schemes in which the degrees of freedom are reduced by using basis functions. Moreover, although we have addressed least-squares problems, our approach can be extended to other parameter-dependent optimization problems, such as to QP's, which would be an immediate extension, but also to other convex and nonconvex problem classes.

References

[1] D. Mayne, J. Rawlings, and M. Diehl, *Model Predictive Control: Theory and Design*, 2nd ed. Madison,WI: Nob Hill Publishing, LCC, 2018.

[2] C. Rao, J. Rawlings, and D. Mayne, "Constrained state estimation for nonlinear discrete-time systems: Stability and moving horizon approximations," *IEEE Trans. Automatic Control*, vol. 48, no. 2, pp. 246–258, 2003.

[3] J. Skaf and S. Boyd, "Multi-period portfolio optimization with constraints and transaction costs," in *Working Manuscript*, 2009, https://web.stanford.edu/~boyd/papers/dyn_port_opt.html.

[4] M. Graf Plessen, L. Puglia, T. Gabbriellini, and A. Bemporad, "Dynamic option hedging with transaction costs: A stochastic model predictive control approach," *Int. J. Robust Nonlinear Control*, vol. 29, pp. 5058–5077, 2019.

[5] S. Boyd, E. Busseti, S. Diamond, R. Kahn, K. Koh, P. Nystrup, and J. Speth, "Multi-period trading via convex optimization," *Foundations and Trends® in Optimization*, vol. 3, no. 1, pp. 1–76, 2017.

[6] G. Frison and J. Jørgensen, "A fast condensing method for solution of linear-quadratic control problems," in *52nd IEEE Conference on Decision and Control*, 2013, pp. 7715–7720.

[7] 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/.

[8] R. Cagienard, P. Grieder, E. Kerrigan, and M. Morari, "Move blocking strategies in receding horizon control," *Journal of Process Control*, vol. 17, no. 6, pp. 563–570, 2007.

[9] R. Gondhalekar and J. Imura, "Least-restrictive move-blocking model predictive control," *Automatica*, vol. 46, no. 7, pp. 1234–1240, 2010.

[10] R. Shekhar and C. Manzie, "Optimal move blocking strategies for model predictive control," *Automatica*, vol. 61, pp. 27–34, 2015.

[11] T. Schwickart, H. Voos, M. Darouach, and S. Bezzaoucha, "A flexible move blocking strategy to speed up model-predictive control while retaining a high tracking performance," in *Proc. European Control Conference*, 2016, pp. 764–769.

[12] S. Wright, "Efficient convex optimization for linear MPC," in *Handbook of Model Predictive Control*. Springer, 2019, pp. 287–303.

[13] D. Axehill, "Controlling the level of sparsity in MPC," *Systems & Control Letters*, vol. 76, pp. 1 – 7, 2015.

[14] 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, 2018.

[15] 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.

[16] M. Berry, M. Heath, I. Kaneko, M. Lawo, R. Plemmons, and R. Ward, "An algorithm to compute a sparse basis of the null space," *Numerische Mathematik*, vol. 47, no. 4, pp. 483–504, 1985.

[17] 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.

[18] J. Yang, T. J. Meijer, V. S. Dolk, B. d. Jager, and W. P. M. H. Heemels, "A system-theoretic approach to construct a banded null basis to efficiently solve MPC-based QP problems*," in *Proc. 59th IEEE Conf. on Decision and Control*, 2019, pp. 1410–1415.

[19] C. Ong and Z. Wang, "Reducing variables in model predictive control of linear system with disturbances using singular value decomposition," *Systems & Control Letters*, vol. 71, pp. 62–68, 2014.

[20] O. Rojas, G. Goodwin, M. Seròn, and A. Feuer, "An SVD based strategy for receding horizon control of input constrained linear systems," *Int. J. Robust and Nonlinear Control*, vol. 14, no. 13-14, pp. 1207–1226, 2004.

[21] J. Unger, M. Kozek, and S. Jakubek, "Reduced order optimization for model predictive control using principal control moves," *Journal of Process Control*, vol. 22, no. 1, pp. 272 – 279, 2012.

[22] C. Lawson and R. Hanson, *Solving least squares problems*. SIAM, 1974, vol. 161.

[23] R. Fletcher, *Practical methods of optimization*, 2nd ed. Wiley, 2000.

[24] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning*, 2nd ed. New York: Springer, 2009.

[25] K. Bennett and O. Mangasarian, "Multicategory discrimination via linear programming," *Optimization Methods and Software*, vol. 3, pp. 27–39, 1994.

[26] V. Breschi, D. Piga, and A. Bemporad, "Piecewise affine regression via recursive multiple least squares and multicategory discrimination," *Automatica*, vol. 73, pp. 155–162, Nov. 2016.

[27] M. Aly, "Survey on multiclass classification," Caltech, USA, Tech. Rep., 2005.

[28] G. Golub, A. Hoffman, and G. Stewart, "A generalization of the Eckart-Young-Mirsky matrix approximation theorem," *Linear Algebra and Its Applications*, vol. 88, pp. 317–327, 1987.

[29] G. Fung and O. Mangasarian, "Multicategory proximal support vector machine classifiers," *Machine Learning*, vol. 59, pp. 77–97, 2005.

[30] G. Ou and Y. Murphey, "Multi-class pattern classification using neural networks," *Pattern Recognition*, vol. 40, no. 1, pp. 4–18, 2007.

[31] A. Bemporad, M. Morari, V. Dua, and E. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

[32] ODYS S.r.l., "Deep Learning Toolset," https://odys.it/deep-learning.

[33] N. Srebro and T. Jaakkola, "Weighted low-rank approximations," in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 2003, pp. 720–727.

[34] R. Fourer, "Solving staircase linear programs by the simplex method, 1: Inversion," *Mathematical Programming*, vol. 23, no. 1, pp. 274–313, Dec 1982.

[35] J. Wilkinson, *The Algebraic Eigenvalue Problem*, ser. Monographs on numerical analysis. Clarendon Press, 1988.

[36] G. G. C. van Loan, *Matrix Computations*, 4th ed. JHU Press, 2013.

[37] D. Bindel, J. Demmel, W. Kahan, and O. Marques, "On computing givens rotations reliably and efficiently," *ACM Transactions on Mathematical Software*, vol. 28, no. 2, pp. 206–238, Jun 2002.

[38] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[39] A. Bemporad, A. Casavola, and E. Mosca, "A predictive reference governor for constrained control systems," *Computers in Industry*, vol. 36, pp. 55–64, 1998.

[40] E. Gilbert, I. Kolmanovsky, and K. T. Tan, "Discrete-time reference governors and the nonlinear control of systems with state and control constraints," *Int. J. Robust Nonlinear Control*, vol. 5, no. 5, pp. 487–504, 1995.

[41] B. Khan and J. Rossiter, "Alternative parameterisation within predictive control: a systematic selection," *International Journal of Control*, vol. 86, no. 8, pp. 1397–1409, 2013.

[42] 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.

[43] D. Seborg, D. Mellichamp, T. Edgar, and F. D. III, *Process Dynamics and Control*. John Wiley & Sons, 2010.

Alberto Bemporad received his Master's degree 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 is Full Professor at the IMT School for Advanced Studies Lucca, Italy, where he served as the Director of the institute in 2012-2015. He spent visiting periods at Stanford University, University of Michigan, and Zhejiang University. In 2011 he cofounded ODYS S.r.l., a company specialized in developing model predictive control systems for industrial production. He has published more than 350 papers in the areas of model predictive control, hybrid systems, optimization, automotive control, and is the co-inventor of 16 patents. He is 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 in 2002-2010. He received the IFAC High-Impact Paper Award for the 2011-14 triennial and the IEEE CSS Transition to Practice Award in 2019. He is an IEEE Fellow since 2010.

Gionata Cimini received his Master's degree in Computer and Automation Engineering in 2012 and his Ph.D. degree in Information Engineering, in 2017, from Università Politecnica delle Marche, Italy. He has been a guest Ph.D. scholar at the IMT School for Advanced Studies Lucca, Italy, and a visiting Ph.D. scholar at University of Michigan, USA. He held research assistant positions at the Information Department, Università Politecnica delle Marche and at the Automotive Research Center, University of Michigan. In 2016, he was a contract employee at General Motors Company, USA. In 2017 he joined ODYS S.r.l., where he is currently the technical manager for numerical optimization. His research interests include model predictive control, embedded optimization, and their application to problems in the automotive, aerospace, and power electronics domains.