2024 European Control Conference (ECC)
June 25-28, 2024. Stockholm, Sweden

Structural Exploitation for the Homogeneous Reformulation of Model Predictive Control Problems

Jonas Hall[1,2] and Arvind U Raghunathan[1]

Abstract— **Algorithms for solving Quadratic Programs (QPs) are indispensable in the research of Model Predictive Control (MPC) of linear dynamical systems. In a recent paper, Raghunathan [1] proposed a novel Homogeneous Quadratic Program (HQP) formulation that can determine optimality and infeasibility of QPs under assumptions that are readily satisfied for MPC. In this paper, we develop a structure exploiting factorization for the linear systems that occur when solving the QPs arising in MPC using the HQP formulation. We have developed a C++ framework (QOACH-MPC) that abstracts the structure exploiting factorization from the algorithm implementation, and makes it convenient for implementing and testing algorithms for MPC. Currently, QOACH-MPC implements an Interior Point Method (IPM) and Semismooth Newton Method (SNM) for solving the HQP, where the step computation exploits the structure in MPC. We demonstrate how our framework can be leveraged to produce a mixed algorithmic strategy for solving the closed-loop MPC problem.**

I. Introduction

Model Predictive Control (MPC) has become the predominant choice for control of constrained linear dynamical systems. Central to MPC is the solution of Quadratic Programs (QPs) that compute a control action guaranteeing the system's stability while satisfying the limits on states and controls. This has motivated research in the development of algorithms based on Interior Point Methods (IPMs) [2], [3], active-set methods [4], [5], splitting methods [6], [7], [8], and Semismooth Newton Methods (SNMs) [9]. The methods differ in their computational complexity per iteration, ability to leverage solutions at the previous time step, and iteration complexity. However, most of the algorithms can benefit from exploiting the structure that is inherent in QP instances arising in MPC. To this end, Riccati recursion has been used to reduce the computational effort involved in the factorization of the linear systems arising in the context of MPC [10], [2]. This allows the computational effort involved per iteration to scale linearly instead of cubically in the number of time steps in MPC.

QP algorithms are designed to guarantee convergence to an optimal solution when the problem is feasible. However, feasibility is not always guaranteed. In real-time applications such as MPC, it is imperative for QP algorithms to recognize infeasibility and produce a certificate of infeasibility. In this context, not all QP algorithms are capable of returning such a certificate. For example, the standard IPM in [2] cannot produce a certificate of infeasibility. On the other hand, IPMs based on the Homogeneous Self-Dual embedding such as the one in [3] can return a certificate of infeasibility. The active-set methods [4], [5], some splitting method [8] and SNMs [9] can also return a certificate of infeasibility.

Raghunathan [1] introduced a novel approach of reformulating the QP into a so-called Homogeneous Quadratic Program (HQP). The HQP has one additional nonnegative variable compared to the original QP, is always feasible, and can be shown to inherit the convexity properties of the QP under very mild assumptions. The HQP always has an optimal solution. Further, optimality and infeasibility of a QP can be inferred by inspecting the solution of the HQP. The key advantage is that any algorithm based on HQP can now handle optimality and infeasibility of QPs robustly. In a follow-up work [11], the authors presented an IPM with predictor-corrector step for solving the HQP, and implemented the algorithm in Julia [12]. The results demonstrated that the effectiveness of the HQP formulation based IPM in determining optimality and infeasibility of QP. Leveraging these works, Raghunathan and Hall [13] have developed QOACH (Quadratic Optimization Algorithms for Convex problems based on Homogenization), a C++ implementation for solving QPs based on the HQP reformulation. QOACH abstracts the specifics of the factorizations in the step computations away from the algorithm implementation. QOACH currently implements an infeasible IPM and SNM for solving QP and employs sparse linear algebra for factorizations in the step computation. These implementations have been tested on a number of publicly available QP instances.

In this paper, we extend QOACH to exploit the structure that is inherent in QP instances arising in MPC. A key departure of our approach from [2] is that we employ the HQP formulation for MPC. In addition, we allow the dynamical systems to also feature algebraic variable at each time step in the MPC. These algebraic variables could be slacks variables inserted into inequality constraints at each time step of the MPC or they could be naturally present in the dynamics of the system. We extend the Riccati recursion developed in [10], [2] for factorization of the matrices that arise in the step computation for dynamical systems with algebraic variables. QOACH-

[1]Mitsubishi Electric Research Laboratories, 201 Broadway, Cambridge, MA 02139, USA
[2]Division of Systems Engineering, Boston University, USA
hallj@bu.edu, raghunathan@merl.com

MPC allows to easily switch between different algorithms while continuing to exploit the structure inherent in the MPC. In a numerical experiment we demonstrate that switching between SNM and IPM can be beneficial when solving a sequence of Optimal Control Problems (OCPs).

The remainder of this paper is organized as follows. §II introduces the MPC problem formulation. We present the homogeneous reformulation of the QP arising in MPC in §III. §IV describes the IPM and SNM algorithms using the HQP. The Riccati recursion for exploiting structure is described in §V. §VI presents the computational results and conclusions are presented in §VII.

A. Notation

The notation $\mathbb{R}$ denotes the set of reals, $v \in \mathbb{R}^n$ denotes a real-valued vector of dimension n, and $\mathrm{diag}(v) \in \mathbb{R}^{n\times n}$ denotes the diagonal matrix with entries of v on the diagonal. The i-th component of v is denoted as $[v]_i$. For a matrix $M \in \mathbb{R}^{n\times m}$ we refer to the element at i-th row and j-th column by $[M]_{ij}$. For $a, b \in \mathbb{R}$, $0 \le a \perp b \ge 0$ represents the complementarity constraint $a, b \ge 0, ab = 0$. For vectors $u, v \in \mathbb{R}^n$, the notation $0 \le u \perp v \ge 0$ denotes the component-wise complementarity constraint $0 \le [u]_i \perp [v]_i \ge 0$. For $a, b \in \mathbb{R}$, the Fischer-Burmeister function is denoted as $\phi(a,b) = a + b - \sqrt{a^2+b^2}$. For vectors $u, v \in \mathbb{R}^n$, $\Phi : \mathbb{R}^{n\times n} \to \mathbb{R}^n$ denotes the componenent-wise Fischer-Burmeister function, i.e., $[\Phi(u,v)]_i = \phi([u]_i, [v]_i)$. The space of $n \times n$ real, symmetric matrix is denoted as $\mathbb{S}^n$. For a matrix $A \in \mathbb{S}^n$, the notation $A\ (\succeq) \succ 0$ denotes the positive (semi)definiteness of the matrix.

II. Problem Formulation

We are interested in solving the discrete-time finite horizon Optimal Control Problem (OCP)

$$\begin{aligned}
\min_{x_k,u_k,y_k} \ & \sum_{k=0}^{N-1} \left(\ell_k^x(x_k) + \ell_k^u(u_k) + \ell_k^y(y_k)\right) \\
& + \ell_N^x(x_N) + \ell_N^y(y_N) && \text{(1a)}\\
\text{s.t.} \ & x_0 = \hat{x}_0, && \text{(1b)}\\
& -x_{k+1} + A_k x_k + B_k u_k + C_k y_k = g_k, && \text{(1c)}\\
& D_k x_k + E_k u_k + F_k y_k = h_k, && \text{(1d)}\\
& D_N x_N + F_N y_N = h_N, && \text{(1e)}\\
& (\underline{x}_k, \underline{u}_k, \underline{y}_k) \le (x_k, u_k, y_k) \le (\overline{x}_k, \overline{u}_k, \overline{y}_k), && \text{(1f)}\\
& (\underline{x}_N, \underline{y}_N) \le (x_N, y_N) \le (\overline{x}_N, \overline{y}_N), && \text{(1g)}
\end{aligned}$$

where $k = 0, 1, \ldots, N-1$ is the time index, $x_k \in \mathbb{R}^{n_x}$ is the state at time k, $u_k \in \mathbb{R}^{n_u}$ is the control action during the kth control interval, and $y_k \in \mathbb{R}^{n_y}$ is the algebraic variables at time k. The bar quantities $\underline{x}_k, \overline{x}_k \in \mathbb{R}^{n_x}$ represent fixed lower and upper bounds for the state variables x_k (analogously for u_k and y_k), and $g_k \in \mathbb{R}^{n_x}$, $h_k, h_N \in \mathbb{R}^{n_y}$ define the affine offsets for the dynamics. The convex quadratic stage and terminal cost terms are defined by $\ell_k^x(x_k) = \frac{1}{2}x_k^\top Q_k^x x_k + (q_k^x)^\top x_k$ for $k = 0, \ldots, N$ and $\ell_k^u(u_k)$, $\ell_k^y(y_k)$, $\ell_N^y(y_N)$ are defined analogously.

A. Assumptions

Throughout the remainder of this paper we make the following assumptions:

(M1) The stage cost Hessians Q_k^x, Q_k^y are positive semidefinite and crucially Q_k^u is positive definite.

(M2) The matrices F_k are invertible for all k.

Assumption (M1) is standard in MPC. Assumption (M2) is not restrictive. This holds for example when the discrete-time system results from the time discretization of index-1 Differential Algebraic Equation (DAE) [14].

III. Homogeneous Reformulation

For ease of exposition, we view (1) as a QP

$$\begin{aligned}
\min_{z} \ & \frac{1}{2} z^\top Q z + q^\top z && \text{(2a)}\\
\text{s.t.} \ & Az = b, && \text{(2b)}\\
& \underline{z} \le z \le \bar{z}, && \text{(2c)}
\end{aligned}$$

where $z = (x_0, u_0, y_0, \ldots, x_{N-1}, u_{N-1}, y_{N-1}, x_N, y_N)$. The remaining parameters $Q, q, A, b, \underline{z}, \overline{z}$ in (2) can be readily inferred from (1). Q and A are sparse and highly structured matrices. Exploiting this sparsity pattern is well understood and discussed in detail in §V.

The Homogeneous Slacked QP (HSQP) reformulation [11] of (2) is given by

$$\begin{aligned}
\min_{z, s^l, s^u, \tau} \ & \frac{1}{2} z^\top Q z + \tau q^\top z + \frac{1}{2}\theta\tau^2 - \theta\tau && \text{(3a)}\\
\text{s.t.} \ & Az = \tau b, && \text{(3b)}\\
& z - s^l = \tau \underline{z}, && \text{(3c)}\\
& z + s^u = \tau \bar{z}, && \text{(3d)}\\
& \tau, s^l, s^u \ge 0, && \text{(3e)}
\end{aligned}$$

where $\theta \in \mathbb{R}$ is a parameter controlling the convexity of the reformulation (see Theorem 1). The HSQP introduces a nonnegative variable τ which multiplies the right hand side of the equality constraints (2b) and the bounds (2c) rendering the constraints in HSQP to be homogeneous. Thus, $(z, s^l, s^u, \tau) = 0$ is always a feasible solution of (3) regardless of the feasibility of (2). The nonnegative slack variables s^l and s^u are introduced in order to convert the homogenizations of (2c) into equality constraints. Raghunathan et al [11] showed that a solution of (2) or its infeasibility can be inferred from solving (3) if the following assumptions hold.

(A1) The matrix A has full row rank.

(A2) The matrix Q is positive definite on the null space of A i.e., $v^T Q v > 0$ for all v with $Av = 0$.

We now show that these assumptions hold in the given MPC setting.

Lemma 1. *Suppose Assumptions (M1)-(M2) hold for MPC (1), then QP (2) satisfies Assumptions (A1)-(A2).*

Proof. From Assumption (M2), the algebraic variables y_k can be eliminated for each k using (1d) to obtain the dynamical system in terms of states and controls alone. Subsequently, the states x_k can also be eliminated using (1b)-(1c). Since equality constraints provide unique pivots to eliminate a set of variable the equality constraints are full row rank, this proves that Assumption (A1) holds. The elimination of x_k, y_k described above results in an inequality constrained optimization problem in the controls u_k. The Hessian of the objective function for the inequality constrained problem is positive definite due to Assumption (M1). This implies that Assumption (A2) holds, completing the proof. $\square$

Using Lemma 1, we can restate the main result from [11] relating HSQP and QP in the following.

Theorem 1. *Suppose Assumptions (M1)-(M2) hold for (1). Choose the parameter $\theta > 0$ to satisfy*

- *$\theta + 2\theta^\star > 0$ where $\theta^\star$ is the optimal value of the QP without bounds, i.e. (2a)-(2b)*
- *θ is chosen large enough to satisfy*

$$\begin{bmatrix} d_z \\ d_\tau \end{bmatrix}^T \begin{bmatrix} Q & q \\ q^T & \theta \end{bmatrix} \begin{bmatrix} d_z \\ d_\tau \end{bmatrix} > 0 \,\forall\, (d_z, d_\tau) : Ad_z - bd_\tau = 0.$$

Then, the following hold

(i) QP (2) has an optimal solution $x^\star$ iff HSQP (3) has an optimal solution $(\hat{z}, \hat{s}^l, \hat{s}^u, \hat{\tau})$ with $\hat{\tau} > 0$.

(ii) QP (2) is infeasible iff HSQP (3) has an optimal solution $(\hat{z}, \hat{s}^l, \hat{s}^u, \hat{\tau})$ with $\hat{\tau} = 0$.

Proof. From Lemma 1 we have that Assumptions (A1)-(A2) hold for QP (2). Theorems 1-2 in [11] are applicable and be used to prove the claim. $\square$

Theorem 1 provides the basis for solving QP by developing algorithms that find a minimizer of HSQP. In [11] the authors described an infeasible Interior Point Method (IPM) with Mehrotra's predictor-corrector step computation in Julia. Specifically, the authors derived a block elimination strategy for the linear system in the IPM applied to HSQP such that the computational effort per iteration is similar to that of IPMs applied to QPs [15]. In QOACH [13] we recognized that the block elimination strategy can be applied to any method that solves HSQP through the application of Newton's method to the first-order stationary conditions. We leveraged this observation to implement an IPM and a Semismooth Newton Method (SNM) for solving HSQP (3). We provide a description of a general algorithm for solving HSQP (3) and the step computation in §IV.

A. Computation of θ

We conclude this section with a brief discussion on the computation of θ, for details we refer to [1]. The main effort in the computation of $\theta^\star = \frac{1}{2}(z^{\text{unc}})^\top Q z^{\text{unc}} + q^\top z^{\text{unc}}$, where z^{unc} is the solution of

$$\begin{pmatrix} Q & A^\top \\ A & 0 \end{pmatrix} \begin{pmatrix} z^{\text{unc}} \\ \lambda^{\text{unc}} \end{pmatrix} = \begin{pmatrix} -q \\ b \end{pmatrix}. \tag{4}$$

In what follows we show that computing the step direction requires solving systems similar to the one above. To this end, the computational cost of obtaining $\theta^\star$ amounts to one additional iteration.

IV. Algorithmic Framework for Solving HSQP

We now outline a generic framework for solving HSQP. The first-order stationary conditions for HSQP are

$$R_d(\boldsymbol{v}, \boldsymbol{\lambda}, \boldsymbol{\nu}) := \tag{5a}$$
$$\begin{bmatrix} Qz + A^\top \lambda + q\tau + \lambda^l + \lambda^u \\ q^\top z - b^\top \lambda - \underline{z}^\top \lambda^l - \overline{z}^\top \lambda^u + \theta\tau - \nu^\tau - \theta \\ -\lambda^l - \nu^l \\ +\lambda^u - \nu^u \end{bmatrix} = 0,$$

$$R_p(\boldsymbol{v}, \boldsymbol{\lambda}, \boldsymbol{\nu}) := \begin{bmatrix} Az - b\tau \\ z - s^l - \underline{z}\tau \\ z + s^u - \overline{z}\tau \end{bmatrix} = 0, \tag{5b}$$

$$R_c(\boldsymbol{v}, \boldsymbol{\lambda}, \boldsymbol{\nu}) := \begin{bmatrix} 0 \le s^l \perp \nu^l \ge 0 \\ 0 \le s^u \perp \nu^u \ge 0 \\ 0 \le \tau \perp \nu^\tau \ge 0 \end{bmatrix}, \tag{5c}$$

where $\lambda, \lambda^l, \lambda^u$ are multipliers for the equality constraints (3b)-(3d), respectively, ν^l, ν^u, ν^τ are multipliers for the bounds on s^l, s^u, τ, respectively, and $\boldsymbol{v} := (z, s^l, s^u, \tau)$, $\boldsymbol{\lambda} := (\lambda, \lambda^l, \lambda^u)$, and $\boldsymbol{\nu} := (\nu^l, \nu^u, \nu^\tau)$.

Different algorithms for solving HSQP (3) can be viewed as differing in how complementarity constraints (5c) are handled. IPMs [15] satisfy bounds on s^l, s^u, τ and $\boldsymbol{\nu}$ by enforcing that all iterates are positive and solve for

$$R_c^{\text{IPM}}(\boldsymbol{v}, \boldsymbol{\lambda}, \boldsymbol{\nu}) := \begin{bmatrix} \text{diag}(s^l)\nu^l \\ \text{diag}(s^u)\nu^u \\ \tau\nu^\tau \end{bmatrix} = 0. \tag{6}$$

On the other hand, SNMs such as [9], choose to solve

$$R_c^{\text{SNM}}(\boldsymbol{v}, \boldsymbol{\lambda}, \boldsymbol{\nu}; \rho, \boldsymbol{\nu}^{\text{prox}}) :=$$
$$\begin{bmatrix} \Phi(s^l, \nu^l) + \rho \cdot (\nu^l - \nu^{l,\text{prox}}) \\ \Phi(s^u, \nu^u) + \rho \cdot (\nu^u - \nu^{u,\text{prox}}) \\ \phi(\tau, \nu^\tau) + \rho \cdot (\nu^\tau - \nu^{\tau,\text{prox}}) \end{bmatrix} = 0, \tag{7}$$

where Φ and ϕ are defined in §I-A, $\rho > 0$ is the so-called proximal parameter and $\boldsymbol{\nu}^{\text{prox}} := (\nu^{l,\text{prox}}, \nu^{u,\text{prox}}, \nu^{\tau,\text{prox}})$ is the proximal point for the bound multipliers. In contrast to [9], we do not add a proximal term to the conditions in (5a)-(5b). Similar to typical proximal point algorithms, we approximately solve a sequence of proximal problems and update the proximal point $\boldsymbol{\nu}^{\text{prox}}$ until HSQP (3) is solved.

QOACH implements an IPM with a Mehrotra predictor-corrector method [15] for computing the step at each iteration. As for the SNM, QOACH computes a Newton step of the system (5a), (5b), and (7), and uses a linesearch filter mechanism [16] for globalization. The key ingredient is the computation of the step by solving the linear system derived from linearization of the first-order stationary conditions: (5a)-(5b),(6) for IPM; and (5a)-(5b),(7) for SNM. The linear system has identical structure in both algorithms as described next.

A. Step Computation

IPM and SNM at an iterate $(\hat{\boldsymbol{v}}, \hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\nu}})$ solve

$$Q\Delta z + A^\top \Delta\lambda + q\Delta\tau + \Delta\lambda^l + \Delta\lambda^u = -R_{d,z}, \tag{8a}$$
$$A\Delta z - b\Delta\tau = -R_{p,e}, \tag{8b}$$

$$q^\top \Delta z - b^\top \Delta\lambda - \underline{z}^\top \Delta\lambda^l - \overline{z}^\top \Delta\lambda^u \tag{8c}$$
$$+\theta\Delta\tau - \Delta\nu^\tau = -R_{d,\tau}, \tag{8d}$$
$$w_\tau \Delta\tau + w_\nu \Delta\nu^\tau = -R_{c,\tau}, \tag{8e}$$
$$\Delta z - \Delta s^l - \underline{z}\Delta\tau = -R_{p,l}, \tag{8f}$$
$$W^l \Delta s^l + W^l_\nu \Delta\nu_l = -R_{c,l}, \tag{8g}$$
$$-\Delta\lambda^l - \Delta\nu^l = -R_{d,l}, \tag{8h}$$
$$\Delta z + \Delta s^u - \overline{z}\Delta\tau = -R_{p,u}, \tag{8i}$$
$$W^u \Delta s^l + W^u_\nu \Delta\nu_l = -R_{c,u}, \tag{8j}$$
$$\Delta\lambda^u - \Delta\nu^u = -R_{d,u}, \tag{8k}$$

where $(w_\tau, w_\nu) = \partial_{(\tau,\nu^\tau)}\varphi(\hat{\tau}, \hat{\nu}^\tau)$, and W^l, W^l_ν, W^u, W^u_ν are diagonal matrices defined as

$$\big([W^l]_{ii}, [W^l_\nu]_{ii}\big) = \partial_{([s^l]_i,[\nu^l]_i)}\varphi([\hat{s}^l]_i, [\hat{\nu}^l]_i),$$
$$\big([W^u]_{ii}, [W^u_\nu]_{ii}\big) = \partial_{([s^u]_i,[\nu^u]_i)}\varphi([\hat{s}^u]_i, [\hat{\nu}^u]_i),$$

where the function $\varphi(a, a') = aa'$ for IPM, and $\varphi(a, a') = \phi(a, a') + \rho a'$ for SNM. Note that the right hand side in (8) is chosen as for a Newton step for sake of exposition. We prove the property that enables the block elimination approach in [11]. The block elimination strategy can be applied to any choice of the right hand side.

Lemma 2. *Consider a generic iterate $(\hat{\boldsymbol{v}}, \hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\nu}})$, and let us assume in the case of IPM that $\hat{s}^l, \hat{s}^u, \hat{\tau}, \hat{\boldsymbol{\nu}} > 0$. Then, $w_\tau, w_\nu > 0$ and $W^l, W^l_\nu, W^u, W^u_\nu$ are positive diagonal matrices.*

Proof. IPM enforces $\hat{s}^l, \hat{s}^u, \hat{\tau}, \hat{\boldsymbol{\nu}} > 0$ and the claim holds. For SNM, the partial $\partial_{(a,a')}\varphi(a, a')$ is given by: $(1 - a/\sqrt{a^2 + (a')^2}, 1 + \rho - a'/\sqrt{a^2 + (a')^2})$ if $(a, a') \neq 0$ and $\big(1 - 1/\sqrt{2}, 1 + \rho - 1/\sqrt{2}\big)$ otherwise. Since $\rho > 0$, the claim holds. $\square$

We outline briefly the block elimination strategy in [11]. We begin by eliminating $\Delta\lambda^l, \Delta^u$ in terms of $\Delta\nu^l, \Delta\nu^u$ using (8h), (8k). Lemma 2 establishes invertibility of W^l_ν, W^u_ν and thus enables elimination of $\Delta\nu^l, \Delta\nu^u$ in terms of $\Delta s^l, \Delta s^u$ using (8g), (8j). Further, $\Delta s^l, \Delta s^u$ can be eliminated in terms of $\Delta z, \Delta\tau$ using (8f), (8i). Finally, again due to Lemma 2, we can eliminate $\Delta\nu^\tau$ using (8e). The block eliminations reduce the linear system (8) to

$$\begin{bmatrix} Q + \Sigma & A^\top & q \\ A & 0 & -b \\ q^\top & -b^\top & \theta + \sigma \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta\lambda \\ \Delta\tau \end{bmatrix} = \begin{bmatrix} \tilde{R}_{d,z} \\ -R_{p,e} \\ \tilde{R}_{d,\tau} \end{bmatrix}, \tag{9}$$

where Σ is a diagonal matrix with positive entries on the diagonal, $\sigma > 0$, and $\tilde{R}_{d,z}, R_{p,e}, \tilde{R}_{d,\tau}$ are appropriately defined using the elimination strategy. The system (9) is solved by eliminating $(\Delta z, \Delta\lambda)$ in terms of $\Delta\tau$ using the first two block equations as

$$\begin{bmatrix} Q + \Sigma & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta\lambda \end{bmatrix} = \begin{bmatrix} \tilde{R}_{d,z} \\ -R_{p,e} \end{bmatrix} - \begin{bmatrix} q \\ -b \end{bmatrix} \Delta\tau. \tag{10}$$

The matrix on the left-hand side is factorized and used to backsolve for two vectors on the right-hand side in (10). The solution of backsolves can then be substituted in the third block equation in (9) to obtain a scalar equation in $\Delta\tau$ alone. Starting with the determination of $\Delta\tau$ we can proceed to recover $(\Delta z, \Delta\lambda)$ and the rest of the quantities in (8). QOACH performs a LDL^T factorization of the matrix in (10) using Intel MKL Pardiso [17]. We show in §V how the structure of the MPC constraints (1b)-(1e) defining the constraint matrix A can be leveraged to reduce the cost for performing the factorization.

B. Initialization

We conclude this section with a discussion on the computation of an initial iterate for the IPM and SNM algorithms. We can leverage the solution of (4) to compute an initial iterate. For IPM, certain quantities of the initial iterate are required to satisfy strict positivity. Accordingly, we set the initial iterate as

$$\tau^0 = \frac{\theta + \sqrt{\theta^2 + 4(2\theta^\star + \theta)\mu^0}}{2(2\theta^\star + \theta)}, \nu^{\tau,0} = \frac{\mu^0}{\tau^0}, \tag{11a}$$
$$z^0 = \tau^0 z^{\text{unc}}, \lambda^0 = \tau^0 \lambda^{\text{unc}}, \lambda^{l,0} = 0, \lambda^{u,0} = 0, \tag{11b}$$
$$s^{l,0} = \max(1.0, z^0 - \tau^0 \underline{z}), [\nu^{l,0}]_i = \frac{\mu^0}{[s^{l,0}]_i}, \tag{11c}$$
$$s^{u,0} = \max(1.0, \tau^0 \overline{z} - z^0), [\nu^{u,0}]_i = \frac{\mu^0}{[s^{u,0}]_i}, \tag{11d}$$

where μ^0 is an initial choice of the barrier parameter. Note that $\tau^0 > 0$ if θ is chosen to satisfy the first assumption in Theorem 1. For the SNM, there is no requirement for any of the iterates to be positive and hence, the slacks and bound multipliers are initialized as follows $s^{l,0} = z^0 - \tau^0 \underline{z}, s^{u,0} = \tau^0 \overline{z} - z^0, \nu^{l,0} = 0, \nu^{u,0} = 0$.

V. Structural Exploitation

The main effort in the step computation of the IPM or SNM is the solution of a linear system of the form

$$\begin{bmatrix} Q + \Sigma & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} dz \\ d\lambda \end{bmatrix} = \begin{bmatrix} r \\ f \end{bmatrix}, \tag{12}$$

where (r, f) represent one of the vectors on the right hand side of (10). In this section we introduce the notation $(dz, d\lambda)$ in order to prevent confusion with the Newton step $(\Delta z, \Delta\lambda)$. By exploiting the structure of A inherited from the system's dynamics (1b)-(1e), we can show that $(dz, d\lambda)$ in (12) is the primal-dual solution of the equality constrained QP

$$\min_{dx_k, du_k, dy_k} \sum_{k=0}^{N-1} (dx_k, du_k, dy_k)^\top Q^\Sigma_k (dx_k, du_k, dy_k)$$
$$+ (dx_N, dy_N)^\top Q^\Sigma_N (dx_N, dy_N)$$

$$- \sum_{k=0}^{N-1} r_k^\top (dx_k, du_k, dy_k) - r_N^\top (dx_N, dy_N) \tag{13a}$$

$$\text{s.t.} \quad dx_0 = f_{-1}^d \tag{13b}$$

$$- dx_{k+1} + A_k dx_k + B_k du_k + C_k dy_k = f_k^d \tag{13c}$$

$$D_k dx_k + E_k du_k + F_k dy_k = f_k^c \tag{13d}$$

$$D_N x_N + F_N y_N = f_N^c \tag{13e}$$

where r_k represents the components in r corresponding to (dx_k, du_k, dy_k) for stage k, and r_N corresponds to (dx_N, dy_N). In a similar manner, $f_{-1}^d, f_k^d, f_k^c, f_N^c$ are the components in f corresponding to the right hand sides for the equality constraints in (1b)-(1e), respectively. Further, $Q_k^\Sigma = Q_k + \Sigma_k$.

Let $-d\lambda_{-1}^d$, $d\lambda_k^d$, $d\lambda_k^c$, $d\lambda_N^c$ denote the multipliers for the constraints in (13b)-(13e), respectively. Then $d\lambda$ in (12) corresponds to the optimal values of the multipliers for the equality constraints in (13). Using the introduced notation and the structure in (13) we can recast the step computation in (12) as

$$- \begin{bmatrix} \mathbb{I} \\ 0 \\ 0 \end{bmatrix} d\lambda_{k-1}^d + Q_k^\Sigma \begin{bmatrix} dx_k \\ du_k \\ dy_k \end{bmatrix} + \begin{bmatrix} D_k^\top \\ E_k^\top \\ F_k^\top \end{bmatrix} d\lambda_k^c + \begin{bmatrix} A_k^\top \\ B_k^\top \\ C_k^\top \end{bmatrix} d\lambda_k^d = r_k \tag{14a}$$

$$- \begin{bmatrix} \mathbb{I} \\ 0 \end{bmatrix} d\lambda_{N-1}^d + Q_N^\Sigma \begin{bmatrix} x_N \\ y_N \end{bmatrix} + \begin{bmatrix} D_N^\top \\ F_N^\top \end{bmatrix} d\lambda_N^c = r_N \tag{14b}$$

$$(13b) - (13e) \tag{14c}$$

for $k = 0, \ldots, N-1$ (note that we chose the sign of $d\lambda_{-1}^d$ such that (14a) extends to $k = 0$). Here, $\mathbb{I}$ denotes the identity matrix. Consider (14b) and (13e) in block matrix form

$$\begin{array}{c} \begin{bmatrix} -\mathbb{I} & Q_N^{\Sigma,x} & & D_N^\top \\ & & Q_N^{\Sigma,y} & F_N^\top \\ & D_N & F_N & \end{bmatrix} \\ \qquad\quad \Psi_N \qquad\qquad \Phi_N \end{array} \begin{bmatrix} d\lambda_{N-1}^d \\ dx_N \\ dy_N \\ d\lambda_N^c \end{bmatrix} = \begin{bmatrix} r_N^x \\ r_N^y \\ f_N^c \end{bmatrix}. \tag{15}$$

From Assumption (M2) we have that F_N^{-1} exists. It can be easily verified that Φ_N^{-1} is given by

$$\Phi_N^{-1} = \begin{bmatrix} 0 & F_N^{-1} \\ F_N^{-\top} & -F_N^{-\top} Q_N^{\Sigma,y} F_N^{-1} \end{bmatrix}. \tag{16}$$

Using (16), we can eliminate $(dy_N, d\lambda_N^c)$ from (15) in terms of dx_N as

$$\begin{bmatrix} dy_N \\ d\lambda_N^c \end{bmatrix} = \begin{bmatrix} F_N^{-1}(f_N^c - D_N dx_N) \\ F_N^{-\top} r_N^y + F_N^{-\top} Q_N^{\Sigma,y} F_N^{-1}(-f_N^c + D_N dx_N) \end{bmatrix}. \tag{17}$$

Plugging (17) into the first block equation in (15) yields the Riccati form

$$-d\lambda_{N-1}^d + \Pi_N dx_N = \pi_N, \tag{18}$$

where

$$\Pi_N = Q_N^{\Sigma,x} - \Psi_N^\top \Phi_N^{-1} \Psi_N, \tag{19a}$$

$$\pi_N = r_N^x - \Psi_N^\top \Phi_N^{-1} \begin{bmatrix} r_N^y \\ f_N^c \end{bmatrix}. \tag{19b}$$

Further, simplifying the right hand side of (19a) yields that $\Pi_N = Q_N^{\Sigma,x} + D_N^\top F_N^{-\top} Q_N^{\Sigma,y} F_N^{-1} D_N$ which shows that $\Pi_N \succeq 0$. Following [2], we show that the form

$$-d\lambda_{k-1}^d + \Pi_k dx_k = \pi_k \tag{20}$$

holds for $0 \leq k \leq N$ and $\Pi_k \succeq 0$ by induction starting from $k = N$ which we have shown to be true. Suppose, (20) hold for $k+1$. Consider the constraints in (14a) in conjunction with (20) for $k+1$

$$\begin{array}{c} \begin{bmatrix} -\mathbb{I} & Q_k^{\Sigma,x} & & A_k^\top & & & D_k^\top \\ & & Q_k^{\Sigma,u} & B_k^\top & & & E_k^\top \\ & A_k & B_k & & -\mathbb{I} & C_k & \\ & & & -\mathbb{I} & \Pi_{k+1} & & \\ & & & C_k^\top & & Q_k^{\Sigma,y} & F_k^\top \\ & D_k & E_k & & & F_k & \end{bmatrix} \\ \qquad\quad \Psi_k \qquad\qquad\qquad \Phi_k \end{array}$$

$$\begin{bmatrix} d\lambda_{k-1}^d \\ dx_k \\ du_k \\ d\lambda_k^d \\ dx_{k+1} \\ dy_k \\ d\lambda_k^c \end{bmatrix} = \begin{bmatrix} r_k^x \\ r_k^u \\ f_k^d \\ \pi_{k+1} \\ r_k^y \\ f_k^c \end{bmatrix}. \tag{21}$$

We eliminate $(dy_k, d\lambda_k^c)$ from (21) using the last two block rows

$$\begin{bmatrix} dy_k \\ d\lambda_k^c \end{bmatrix} = \begin{bmatrix} F_k^{-1} \\ -F_k^{-\top} Q_k^{\Sigma,y} F_k^{-1} \end{bmatrix} (-D_k dx_k - E_k du_k + f_k^c) + \begin{bmatrix} 0 \\ F_k^{-\top} \end{bmatrix} (-C_k^T d\lambda_k^d + r_k^y)$$

to obtain

$$\begin{bmatrix} -\mathbb{I} & \tilde{Q}_k^x & \tilde{Q}_k^{xu} & \tilde{A}_k^\top & \\ & \tilde{Q}_k^{ux} & \tilde{Q}_k^u & \tilde{B}_k^\top & \\ & \tilde{A}_k & \tilde{B}_k & & -\mathbb{I} \\ & & & -\mathbb{I} & \Pi_{k+1} \end{bmatrix} \begin{bmatrix} -d\lambda_{k-1}^d \\ dx_k \\ du_k \\ d\lambda_k^d \\ dx_{k+1} \end{bmatrix} = \begin{bmatrix} \tilde{r}_k^x \\ \tilde{r}_k^u \\ \tilde{f}_k^d \\ \pi_{k+1} \end{bmatrix} \tag{22}$$

where

$$\begin{bmatrix} \tilde{Q}_k^x & \tilde{Q}_k^{xu} \\ \tilde{Q}_k^{ux} & \tilde{Q}_k^u \end{bmatrix} = \begin{bmatrix} Q_k^{\Sigma,x} & \\ & Q_k^{\Sigma_u} \end{bmatrix} + \begin{bmatrix} D_k^\top \\ E_k^\top \end{bmatrix} (F_k^{-\top} Q_k^{\Sigma,y} F_k^{-1}) \begin{bmatrix} D_k & E_k \end{bmatrix} \tag{23}$$

$$\tilde{A}_k = A_k - C_k F_k^{-1} D_k$$

$$\tilde{B}_k = B_k - C_k F_k^{-1} E_k$$

$$\begin{bmatrix} \tilde{r}_k^x \\ \tilde{r}_k^u \\ \tilde{f}_k^d \end{bmatrix} = \begin{bmatrix} r_k^x \\ r_k^u \\ f_k^d \end{bmatrix} - \begin{bmatrix} D_k^\top (F_k^{-\top} r_k^y - F_k^{-\top} Q_k^{\Sigma,y} F_k^{-1} f_k^c) \\ E_k^\top (F_k^{-\top} r_k^y - F_k^{-\top} Q_k^{\Sigma,y} F_k^{-1} f_k^c) \\ C_k F_k^{-1} f_k^c \end{bmatrix}$$

Using the last two block rows in (22) we can eliminate $(d\lambda_k^d, dx_{k+1})$ in terms of (dx_k, du_k) to obtain

$$\begin{bmatrix} d\lambda_k^d \\ dx_{k+1} \end{bmatrix} = \begin{bmatrix} \Pi_{k+1} \\ \mathbb{I} \end{bmatrix} (\tilde{A}_k dx_k + \tilde{B}_k du_k - \tilde{f}_k^d) - \begin{bmatrix} \pi_{k+1} \\ 0 \end{bmatrix}. \tag{24}$$

Substituting for $(d\lambda_k^d, dx_{k+1})$ in the first two block rows of (22) and simplify to obtain

$$\begin{bmatrix} -\mathbb{I} & \widehat{Q}_k^x & \widehat{Q}_k^{xu} \\ & \widehat{Q}_k^{ux} & \widehat{Q}_k^u \end{bmatrix} \begin{bmatrix} d\lambda_{k-1}^d \\ dx_k \\ du_k \end{bmatrix} = \begin{bmatrix} \widehat{r}_k^x \\ \widehat{r}_k^u \end{bmatrix} \tag{25}$$

where

$$\begin{bmatrix} \widehat{Q}_k^x & \widehat{Q}_k^{xu} \\ \widehat{Q}_k^{ux} & \widehat{Q}_k^u \end{bmatrix} = \begin{bmatrix} \tilde{Q}_k^x & \tilde{Q}_k^{xu} \\ \tilde{Q}_k^{ux} & \tilde{Q}_k^u \end{bmatrix} + \begin{bmatrix} \tilde{A}_k^\top \\ \tilde{B}_k^\top \end{bmatrix} \Pi_{k+1} \begin{bmatrix} \tilde{A}_k^\top \\ \tilde{B}_k^\top \end{bmatrix}^\top \tag{26a}$$

$$\widehat{r}_k^x = \tilde{r}_k^x + \tilde{A}_k^\top(\Pi_{k+1}\tilde{f}_k^d + \pi_{k+1}) \tag{26b}$$

$$\widehat{r}_k^u = \tilde{r}_k^u + \tilde{B}_k^\top(\Pi_{k+1}\tilde{f}_k^d + \pi_{k+1}) \tag{26c}$$

Finally, from (25) we obtain from eliminating du_k using the second block equation that

$$\Pi_k = \widehat{Q}_k^x - \widehat{Q}_k^{xu}(\widehat{Q}_k^u)^{-1}\widehat{Q}_k^{ux} \tag{27a}$$

$$\pi_k = \widehat{r}_k^x - \widehat{Q}_k^{xu}(\widehat{Q}_k^u)^{-1}\widehat{r}_k^u \tag{27b}$$

The inverse of $\widehat{Q}_k^u$ exists since the matrices in (26a) and (23) are formed by the addition of low rank matrices to the original Hessian matrices $Q_k^{\Sigma,x}$ and $Q_k^{\Sigma,u}$. Further, $\Pi_k \succeq 0$ since this is obtained as Schur-complement of the matrix in (26a). Finally, the block elimination procedure serves as proof that Φ_k is invertible as claimed before. The main computational effort is the inversion of a dense matrix $\widehat{Q}_k^u$ of size n_u. The steps involved in solving the Riccati recursion are summarized in Algorithm 1.

Algorithm 1: Riccati recursion for solving (12)

```
   Data: $(r, f)$
   Result: $(dz, d\lambda)$
1  Compute $\Pi_k, \pi_k$ for $k = N, \dots, 0$ via (19), (27);
2  Set $dx_0 = f_{-1}^d$;
3  Set $d\lambda_{-1}^d = \Pi_0 dx_0 - \pi_0$;
4  for $k = 0, \dots, N-1$ do
5      Set $du_k = (\widehat{Q}_k^u)^{-1}(\widehat{r}_k^u - \widehat{Q}^{ux} dx_k)$;
6      Set $dx_{k+1} = \tilde{A}_k dx_k + \tilde{B}_k du_k - \tilde{f}_k^d$;
7      Set $d\lambda_k^d = \Pi_{k+1} dx_{k+1} - \pi_{k+1}$;
8      Set $dy_k = F_k^{-1}(f_k^c - D_k dx_k - E_k du_k)$;
9      Set $d\lambda_k^c = F_k^{-\top}(r_k^c - C_k^\top d\lambda_k^d - Q_k^{\Sigma,y} dy_k)$;
10 Compute $(dy_N, d\lambda_N^c)$ using (17).
```

VI. Numerical Results

All results in this section were obtained using a 12th Generation Intel Core i9 processor with 3.2GHz and 125GiB RAM under Linux operating system Ubuntu 20.04.

We first compare the introduced methods to state-of-the-art solvers: the open-source SNM solver FBstab [9], which utilizes the Riccati recursion to solve the MPC problem (1); and the commercial solver Gurobi [18].

A. Servo Motor Control

Consider the continuous time servo motor model [19]

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -128 & -2.5 & 6.4 & 0 \\ 0 & 0 & 0 & 1 \\ 128 & 0 & -6.4 & -10.2 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1282 & 0 & -64 & 0 \end{bmatrix} x,$$

with output and input constraints $|y_2| \leq 78.5$Nm and $|u| \leq 220$V, respectively. We conduct a similar experiment as was done in [9]: we solve discretized reference tracking OCPs, where the time horizon N is varied from 1 to 1000.

The results are depicted in Figure 1 and Table I. We find that the IPM achieves comparable results to the state-of-the-art solver FBstab and slightly slower results compared to Gurobi (at most a factor 3-4). For $N \in \{3, 4\}$, the initial iterate of the SNM is found to be optimal. The initial iterate (see §IV-B) is chosen as the optimal solution of the equality constrained QP neglecting the bounds, which resembles the Linear Quadratic Regulator (LQR), and is optimal if it is feasible for the bounds. For larger N, the SNM is slower by about a factor of 3 compared to the IPM for large horizons, which is due to the increased number of iterations taken (see Table I). Note that FBstab requires about twice as many Newton iterations compared to the IPM, yet it has comparable solution times. This is due to the two backsolves (10) required for step computations in HSQP, which makes an HSQP iterate about twice as expensive as a QP iterate. We further find that for this example the implemented Riccati recursion with dense linear algebra does not lead to significant speed-ups over solving the MPC problem as a sparse QP.

B. From OCP to MPC

The above results discuss the performance for solving a single OCP rather than a series of such as is done in MPC. Oftentimes the OCP does not change much from one time instance to the next, in which case we expect the utilization of the current solution to reduce the computational effort required to obtain its successor. For any sequence of optimization problems this is known as warm-starting. Unfortunately, some algorithms are difficult to warm-start, e.g., the discussed IPM algorithm, whereas other methods are able to recycle the previous solution easily, e.g., the discussed SNM algorithm. Nonetheless, we have seen above that IPM outperforms SNM in the

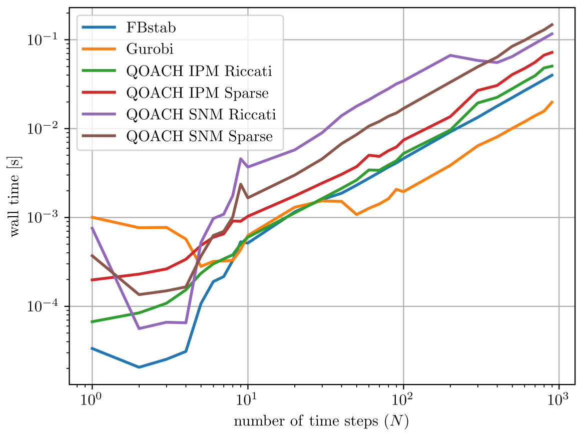

Fig. 1: Comparing computational timings of the introduced method to state-of-the-art solvers.

TABLE I: Comparing the number of iterates required for each solver dependent on the number of stages N. The respective computation times are depicted in Figure 1. The number of iterates for the respective QOACH Riccati and sparse implementations are identical.

	IPM		SNM	
N	QOACH	Gurobi	QOACH	FBstab
3	5	10	0	2
4	6	11	0	2
10	12	12	30	23
100	12	12	34	24

OCP setting, since it often requires significantly fewer Newton iterations. Furthermore, the IPM method tends to be more robust towards poor conditioning, whereas the iterations within an SNM framework may stagnate. Ideally, for an MPC application we desire a method that is capable of both: robust convergence properties while exploiting the knowledge of the current solution in the computation of the next.

In what follows we analyze the effect of warm-starting versus cold-starting for both methods. Cold-starting QOACH does not leverage the previous iterate but rather utilizes the initialization as discussed in §IV-B. As noted in §VI-A, the initial iterate of the SNM can readily provide the optimal solution. Clearly, the situation is quite different for IPM since we require an initialization with positive slacks. As opposed to the LQR-like initialization, the warm-started method uses a shift initialization as is commonly done in MPC [20]. For IPM we initialize via a convex combination between the shifted solution and the initialization used for cold starts, which prevents initialization with vanishing slacks.

Figure 2 depicts the computational times required for solving the closed-loop servo motor as introduced in §VI-A using the cold-start. The respective warm-started solutions are depicted in Figure 3. In both figures, the left plot shows the original reference tracking problem and we find that warm-starting makes little to no difference for IPM. The right figures show results for the same reference tracking problem but with included terminal inequality constraints $-\varepsilon \leq y(T) \leq \varepsilon$ (see Figure 4). We find that warm-starting significantly improves the performance of SNM and slightly improves the performance of IPM. Note that the difference between the cold-started and the warm-started SNM is almost an order of magnitude.

The observations above indicate that it would be beneficial to use a cold-started IPM at the first time step, and otherwise rely on a warm-started SNM method. The flexibility of QOACH enables us to switch between these algorithms easily. The results in Figure 3 show the IPM-SNM method inherits the computational benefits of both methods.

VII. Conclusion and Future Work

In this paper we introduced the structural exploitation of MPC problems when such are solved via the homogeneous QP framework [1]. We showed that standard assumptions in MPC satisfy the assumptions required for the homogenization. We then specified the IPM and SNM step computations within HSQP tailored to MPC via a Riccati recursion. In a numerical experiment we report comparable results to state-of-the-art solvers, and discuss the benefit of switching between IPM and SNM during the MPC feedback loop.

Future work aims at extending the HSQP framework further, both by integration of other solution methods and forms of structural exploitation. For instance, one line of work could consist of integrating an active set strategy, or the structural exploitation in MPC could be extended to a stochastic MPC setting.

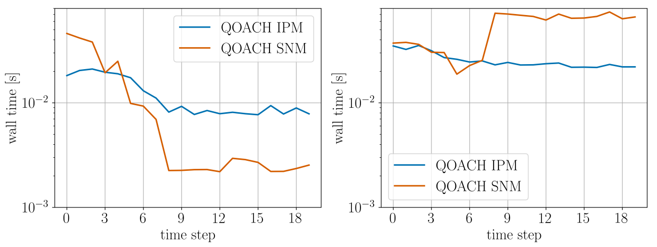

Fig. 2: Plotting the wall time required to obtain the cold-started MPC solution at each time step. The right results includes the terminal constraint $-\varepsilon \leq y(T) \leq \varepsilon$, which is omitted on the left.

References

[1] A. U. Raghunathan, "Homogeneous formulation of convex quadratic programs for infeasibility detection," in *2021 60th IEEE Conference on Decision and Control (CDC)*, pp. 968–973, 2021.

[2] C. V. Rao, S. J. Wright, and J. B. Rawlings, "Application of interior-point methods to model predictive control," *Journal of optimization theory and applications*, vol. 99, pp. 723–757, 1998.

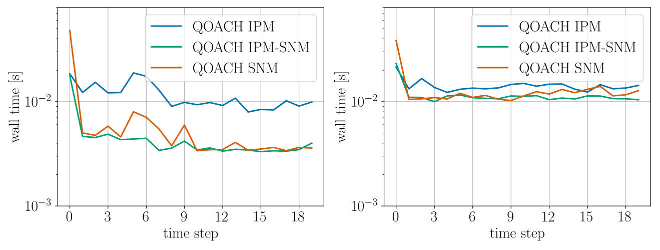

Fig. 3: Plotting the wall time required to obtain the warm-started MPC solution at each time step. The right results includes the terminal constraint $-\varepsilon \leq y(T) \leq \varepsilon$, which is omitted on the left. The mixed method utilizes cold-started IPM in the first time step and warm-started SNM for the remaining time steps.

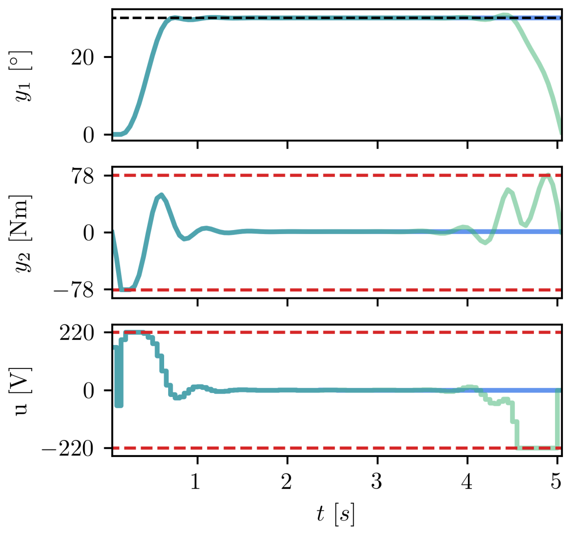

Fig. 4: Open loop servo motor OCP trajectories with (green) and without (blue) terminal constraint $y(T) \approx 0$.

[3] A. Domahidi, E. Chu, and S. Boyd, "ECOS: An socp solver for embedded systems," in *2013 European Control Conference (ECC)*, pp. 3071–3076, 2013.

[4] R. Bartlett and L. Biegler, "QPSchur: A dual, active-set, schur-complement method for large-scale and structured convex quadratic programming," *Optimization and Engineering*, vol. 7, p. 5–32, 2006.

[5] H. Ferreau, H. Bock, and M. Diehl, "An online active set strategy to overcome the limitations of explicit MPC," *International Journal of Robust and Nonlinear Control*, vol. 18, no. 8, p. 816–830, 2008.

[6] P. Patrinos, L. Stella, and A. Bemporad, "Douglas-rachford splitting: Complexity estimates and accelerated variants," in *53rd IEEE Conference on Decision and Control*, pp. 4234–4239, 2014.

[7] P. Giselsson and S. Boyd, "Diagonal scaling in douglas-rachford splitting and admm," in *53rd IEEE Conference on Decision and Control*, pp. 5033–5039, 2014.

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

[9] D. Liao-McPherson and I. Kolmanovsky, "FBstab: A proximally stabilized semismooth algorithm for convex quadratic programming," *Automatica*, vol. 113, p. 108801, 2020.

[10] T. Glad and H. Jonson, "A method for state and control constrained linear quadratic control problems," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1583–1587, 1984.

[11] A. U. Raghunathan, D. Jha, and D. Romeres, "Homogeneous infeasible interior point method for convex quadratic programs," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, pp. 7571–7578, IEEE, 2022.

[12] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM Review*, vol. 59, no. 1, pp. 65–98, 2017.

[13] A. U. Raghunathan and J. Hall, "Homogeneous formulation for a class of convex quadratic programs." under preparation, 2023.

[14] K. E. Brenan, S. L. Campbell, and L. R. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. Society for Industrial and Applied Mathematics, 1995.

[15] S. Wright, *Primal-Dual Interior-Point Methods*. SIAM, 1997.

[16] N. I. M. Gould, S. Leyffer, and P. L. Toint, "A multidimensional filter algorithm for nonlinear equations and nonlinear least-squares," *SIAM Journal on Optimization*, vol. 15, no. 1, pp. 17–38, 2004.

[17] O. Schenk and K. Gartner, "Solving unsymmetric sparse systems of linear equations with PARDISO," *J. of Future Generation Computer Systems*, vol. 20, no. 3, pp. 475–487, 2004.

[18] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2023.

[19] A. Bemporad and E. Mosca, "Fulfilling hard constraints in uncertain linear systems by reference managing," *Automatica*, vol. 34, no. 4, pp. 451–461, 1998.

[20] M. Diehl, H. J. Ferreau, and N. Haverbeke, "Efficient numerical methods for nonlinear MPC and moving horizon estimation," *Nonlinear model predictive control: towards new challenging applications*, pp. 391–417, 2009.