On the Implementation of a Preconditioned Riccati Recursion based Primal-Dual Interior-Point Algorithm for Input Constrained Optimal Control Problems*

Wahlgreen, Morten Ryberg; Jørgensen, John Bagterp

Published in:
IFAC-PapersOnLine

Link to article, DOI:
10.1016/j.ifacol.2022.07.468

Publication date:
2022

Document Version
Publisher's PDF, also known as Version of record

Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 55-7 (2022) 346–351

On the Implementation of a Preconditioned Riccati Recursion based Primal-Dual Interior-Point Algorithm for Input Constrained Optimal Control Problems ⋆

Morten Ryberg Wahlgreen * **John Bagterp Jørgensen** *

* *Department of Applied Mathematics and Computer Science, Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark.*

Abstract:
We present a preconditioned interior-point algorithm tailored for input constrained quadratic programmings (QPs) arising in optimal control problems (OCPs). The implicit approach to OCPs results in large sparse QPs, which we utilized by a tailored Riccati recursion algorithm. The Riccati recursion algorithm requires the solution of a set of small dense linear sub-systems of equations. The proposed preconditioner is an easily invertible diagonal matrix, which we apply in every linear sub-system of equations. We solve a target tracking OCP for a linearized modified quadruple tank system in Matlab. The computational results indicate that ill-conditioning in the sub-systems are reduced and that the additional CPU time for preconditioning is negligible. Additionally, the paper presents a detailed description of the proposed algorithm and serves as an implementation guide for the algorithm.

Keywords: Interior-point method, Quadratic programming, Optimal Control Problem, Riccati recursion, Preconditioning

1. INTRODUCTION

Interior-point methods have been an integrated part of optimization since the 1960s (Forsgren et al., 2002; Wright, 2004). In the 1960s, interior-point methods were mainly applied in problems with nonlinear constraints. Today, the applications span many different types of problems including linear programming (LP), quadratic programming (QP), and nonlinear programming (NLP) (Astfalk et al., 1992; Vanderbei, 1999; Byrd et al., 1999; Nocedal and Wright, 1999; Gertz and Wright, 2003). Furthermore, LPs, QPs, and NLPs are an essential component in development of advanced control algorithms such as model predictive control (MPC) (Rawlings et al., 1994). Interior-point methods suffer from unavoidable ill-conditioning as the iterations approach the solution (Murray, 1971). Preconditioning approaches are often proposed in relation to inexact interior-point methods due to the inherent need of well-conditioned systems in iterative solvers for linear systems (Bergamaschi et al., 2004; Shahzad et al., 2010; Cui et al., 2019). However, proposed preconditioning in relation to exact methods is sparse.

In linear model predictive control (LMPC), QPs arise in the form of optimal control problems (OCPs) (Borrelli et al., 2009). Usually either explicit or implicit approaches are applied to express OCPs as QPs. Explicit approaches result in small dense QPs, while implicit methods result in large sparse QPs (Shahzad et al., 2010). In the implicit case, Riccati recursion has been proposed as an efficient sparse solver with linear complexity in the control horizon (Rao et al., 1998; Frison and Jørgensen, 2013). The Riccati recursion based interior-point method suffers from unavoidable ill-conditioning as all other interior-point methods. We propose a diagonal preconditioner for Riccati recursion based interior-point methods at the cost of negligible additional CPU time. We suggest that preconditioning at the cost of negligible CPU time improve the overall quality of the algorithm.

In this paper, we propose a preconditioned interior-point method for input constrained QPs arising in OCPs. We introduce the well-known Riccati recursion for solution of structured linear systems and propose a preconditioner for the Riccati based interior-point method. We present a detailed description of the proposed interior-point method, where we emphasize how to exploit the structure of box-constraints to reduce CPU time. As such, the paper serves as an implementation guide for a well-conditioned Riccati recursion based interior-point method for QPs in OCPs. The results in the paper are based on a Matlab implementation of the proposed interior-point method. We demonstrate applications of the implementation on a target tracking OCP for a linearized modified quadruple tank system.

The Matlab implementation is an essential building block in the development of a Riccati recursion based sequential quadratic programming (SQP) algorithm for NLPs arising in nonlinear model predictive control (NMPC). Additionally, it serves as a prototype for an C implementation of a LMPC, which can be applied for parallel Monte Carlo simulation of closed loop systems (Wahlgreen et al., 2021).

⋆ Corresponding author: J. B. Jørgensen (E-mail: jbjo@dtu.dk).

10.1016/j.ifacol.2022.07.468

The remaining part of the paper is organized as follows. Section 2 introduces the main parts of the primal-dual interior-point algorithm. Section 3 presents the proposed preconditioner. Section 4 presents the Riccati recursion algorithm. Section 5 presents an example application of our interior-point algorithm. Section 6 presents our conclusion.

2. PRIMAL-DUAL INTERIOR-POINT ALGORITHM

This section presents a detailed description of the primal-dual interior-point algorithm for general box-constrained QPs. The algorithm is a predictor-corrector with scaled KKT-violation convergence criterion. We specialize the algorithm to OCPs in Section 4.

2.1 Quadratic programming

We consider box-constrained QPs on the form,

$$\begin{aligned} \min_{x} \quad & f(x) = \frac{1}{2}x^T H x + g^T x, && \text{(1a)} \\ s.t. \quad & A^T x = b, && \text{(1b)} \\ & l \leq x \leq u, && \text{(1c)} \end{aligned}$$

where $H \in \mathbb{R}^{n\times n}$, $g \in \mathbb{R}^n$, $A \in \mathbb{R}^{n\times m_e}$, $b \in \mathbb{R}^{m_e}$, $l \in \mathbb{R}^n$, $u \in \mathbb{R}^n$, and $x \in \mathbb{R}^n$ are the decision variables.

The Lagrangian function for (1) is,

$$\begin{aligned} \mathcal{L}(x,y,z_l,z_u) =& \frac{1}{2}x^T H x + g^T x - y^T(A^T x - b) \\ & - z_l^T(x-l) - z_u^T(u-x), \end{aligned} \tag{2}$$

where $y \in \mathbb{R}^{m_e}$, $z_l \in \mathbb{R}^n$, and $z_u \in \mathbb{R}^n$ are the Lagrange multipliers for the equality constraints, lower bound constraints, and upper bound constraints, respectively.

The first order KKT-conditions for (1) are,

$$\begin{aligned} Hx + g - Ay - z_l + z_u &= 0, && \text{(3a)} \\ b - A^T x &= 0, && \text{(3b)} \\ s_l + l - x = 0, \quad s_u + x - u &= 0, && \text{(3c)} \\ s_{l,i} z_{l,i} = 0, \quad s_{u,i} z_{u,i} &= 0, && \text{(3d)} \\ (z_l, z_u, s_l, s_u) &\geq 0, && \text{(3e)} \end{aligned}$$

where $i = 1, ..., n$ and the slack variables, s_l and s_u, are

$$s_l = x - l \geq 0, \qquad s_u = u - x \geq 0. \tag{4}$$

We write the KKT-conditions, (3), as the nonlinear system of equations,

$$\begin{aligned} \begin{bmatrix} r_L \\ r_A \\ r_{B_l} \\ r_{B_u} \\ r_{S_l Z_l} \\ r_{S_u Z_u} \end{bmatrix} = \begin{bmatrix} Hx + g - Ay - z_l + z_u \\ b - A^T x \\ s_l + l - x \\ s_u + x - u \\ S_l Z_l e \\ S_u Z_u e \end{bmatrix} = 0, && \text{(5a)} \\ (z_l, z_u, s_l, s_u) \geq 0, && \text{(5b)} \end{aligned}$$

where $Z_l = \text{diag}(z_l)$, $Z_u = \text{diag}(z_u)$, $S_l = \text{diag}(s_l)$, $S_u = \text{diag}(s_u)$, and e is a vector of ones of proper dimension. We apply Newton's method to solve (5), which yields the following linear system of equations for the Newton search direction,

$$\begin{bmatrix} H & -A & -I & I & 0 & 0 \\ -A^T & 0 & 0 & 0 & 0 & 0 \\ -I & 0 & 0 & 0 & I & 0 \\ I & 0 & 0 & 0 & 0 & I \\ 0 & 0 & S_l & 0 & Z_l & 0 \\ 0 & 0 & 0 & S_u & 0 & Z_u \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z_l \\ \Delta z_u \\ \Delta s_l \\ \Delta s_u \end{bmatrix} = - \begin{bmatrix} r_L \\ r_A \\ r_{B_l} \\ r_{B_u} \\ r_{S_l Z_l} \\ r_{S_u Z_u} \end{bmatrix}. \tag{6}$$

We introduce the equivalent system,

$$\begin{bmatrix} H & -A & -C & 0 \\ -A^T & 0 & 0 & 0 \\ -C^T & 0 & 0 & I \\ 0 & 0 & S & Z \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ \Delta s \end{bmatrix} = - \begin{bmatrix} r_L \\ r_A \\ r_C \\ r_{SZ} \end{bmatrix}, \tag{7}$$

with $C = [I, -I]$, $Z = \text{diag}([Z_l; Z_u])$, $S = \text{diag}([S_l; S_u])$, $\Delta z = [\Delta z_l; \Delta z_u]$, $\Delta s = [\Delta s_l; \Delta s_u]$, $r_C = [r_{B_l}; r_{B_u}]$, and $r_{SZ} = [r_{S_l Z_l}; r_{S_u Z_u}]$. We point out that (7) only serves as notation, while (6) is the preferred form for implementation as it exploits the identity structure of C.

The solution of (7), $(\Delta x, \Delta y, \Delta z, \Delta s)$, serves as a step direction for the algorithm. In each iteration, $[l]$, the algorithm performs the step,

$$(x, y, z, s) \leftarrow (x, y, z, s) + \eta\alpha(\Delta x, \Delta y, \Delta z, \Delta s), \tag{8}$$

where $\eta = 0.995$ and the step-size, α, ensures $(z, s) \geq 0$.

2.2 Predictor-corrector

Our algorithm applies the Mehtrotra predictor-corrector method (Mehrotra, 1992). The method considers the following form of (7),

$$\begin{bmatrix} H & -A & -C & 0 \\ -A^T & 0 & 0 & 0 \\ -C^T & 0 & 0 & I \\ 0 & 0 & S & Z \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ \Delta s \end{bmatrix} = - \begin{bmatrix} r_L \\ r_A \\ r_C \\ \bar{r}_{SZ} \end{bmatrix}, \tag{9}$$

where $\bar{r}_{SZ}$ varies in the predictor and corrector phase. In the predictor phase, we set $\bar{r}_{SZ} = r_{SZ}$ and denote the solution to (9), $(\Delta x^{aff}, \Delta y^{aff}, \Delta z^{aff}, \Delta s^{aff})$. This direction is called the affine direction. In the corrector phase, we set $\bar{r}_{SZ} = r_{SZ} + \Delta S^{aff} \Delta Z^{aff} - \sigma\mu e$ and denote the solution $(\Delta x, \Delta y, \Delta z, \Delta s)$. We compute the duality gap, μ, and centering parameter, σ, as,

$$\begin{aligned} \mu^{aff} &= \frac{(z + \alpha^{aff}\Delta z)^T (s + \alpha^{aff}\Delta s^{aff})}{m}, \\ \mu &= \frac{s^T z}{m}, \qquad \sigma = \left(\frac{\mu^{aff}}{\mu}\right)^3, \end{aligned} \tag{10}$$

where $m = 2n$ for box-constrained QPs (1).

2.3 Augmented form

We write (9), i.e., (6), in the augmented form,

$$\begin{bmatrix} \bar{H} & -A \\ -A^T & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = - \begin{bmatrix} -\bar{r}_L \\ r_A \end{bmatrix}, \tag{11}$$

where

$$\bar{H} = H + D_l + D_u, \tag{12}$$

$$\begin{aligned} \bar{r}_L =& -r_L + (S_l^{-1} Z_l)(r_{B_l} - Z_l^{-1} \bar{r}_{S_l Z_l}) \\ & - (S_u^{-1} Z_u)(r_{B_u} - Z_u^{-1} \bar{r}_{S_u Z_u}), \end{aligned} \tag{13}$$

with $D_l = \text{diag}(z_l / s_l)$ and $D_u = \text{diag}(z_u / s_u)$. The variables $\bar{r}_{S_l Z_l}$ and $\bar{r}_{S_u Z_u}$ are,

$$\text{Predictor:} \quad \begin{matrix} \bar{r}_{S_l Z_l} = r_{S_l Z_l}, \\ \bar{r}_{S_u Z_u} = r_{S_u Z_u}, \end{matrix} \tag{14}$$

$$\text{Corrector:} \quad \begin{matrix} \bar{r}_{S_l Z_l} = r_{S_l Z_l} + \Delta S_l^{aff} \Delta Z_l^{aff} - \sigma\mu e, \\ \bar{r}_{S_u Z_u} = r_{S_u Z_u} + \Delta S_u^{aff} \Delta Z_u^{aff} - \sigma\mu e, \end{matrix} \tag{15}$$

where $\Delta Z_l^{aff} = \text{diag}(\Delta z_l^{aff})$, $\Delta Z_u^{aff} = \text{diag}(\Delta z_u^{aff})$, $\Delta S_l^{aff} = \text{diag}(\Delta s_l^{aff})$, and $\Delta S_u^{aff} = \text{diag}(\Delta s_u^{aff})$.

We compute Δz_l, Δz_u, Δs_l, and Δs_u as,

$$\Delta z_l = (S_l^{-1} Z_l)(r_{B,l} - Z_l^{-1} \bar{r}_{S_l Z_l}) - (S_l^{-1} Z_l)\Delta x, \tag{16a}$$

$$\Delta z_u = (S_u^{-1} Z_u)(r_{B,u} - Z_u^{-1} \bar{r}_{S_u Z_u}) + (S_u^{-1} Z_u)\Delta x, \tag{16b}$$

$$\Delta s_l = -Z_l^{-1} \bar{r}_{S_l Z_l} - Z_l^{-1} S_l \Delta z_l, \tag{16c}$$

$$\Delta s_u = -Z_u^{-1} \bar{r}_{S_u Z_u} - Z_u^{-1} S_u \Delta z_u. \tag{16d}$$

We point out that the structure of box-constraints result in cheap diagonal matrix operations exploited as element-wise vector-vector operations.

2.4 Fraction-to-the-boundary

We compute the step size, $0 < \alpha \leq 1$, after solution of (11) in both the predictor and corrector phase. We apply the fraction-to-the-boundary rule,

$$\begin{bmatrix} z \\ s \end{bmatrix} + \alpha \begin{bmatrix} \Delta z \\ \Delta s \end{bmatrix} \geq \kappa \begin{bmatrix} z \\ s \end{bmatrix}, \tag{17}$$

for $0 \leq \kappa << 1$ and $\kappa \to 0$ as the iteration number, $[l]$, increases. For $\kappa = 0$, the fraction-to-boundary rule ensures $(z, s) \geq 0$. With $\kappa > 0$, the rule (17) strictly satisfies $(z, s) > 0$ with a z or s proportional step-back from the boundary. The rule, (17), is similar to the rule applied in IPOPT (Wächter and Biegler, 2006).

In the predictor phase, we set $\kappa = 0$ to get maximum information from the affine step direction. In the corrector phase, we set $\kappa = \min(1 - \eta, \mu^{aff})$, which ensures that $\kappa \to 0$ as the algorithm reaches the solution.

2.5 Convergence criterion

The algorithm converges once the first order KKT-conditions, (3), are satisfied. Numerically, we consider the scaled KKT-violation, ξ,

$$\begin{aligned} \xi = \max \Big(& \tilde{r}_L ||r_L||_\infty, \tilde{r}_A ||r_A||_\infty, \tilde{r}_B ||r_{B_l}||_\infty, \\ & \tilde{r}_B ||r_{B_u}||_\infty, ||r_{S_l Z_l}||_\infty, ||r_{S_u Z_u}||_\infty \Big), \end{aligned} \tag{18}$$

where

$$\tilde{r}_L = \max(1, ||H||_\infty, ||g||_\infty, ||A||_\infty)^{-1}, \tag{19a}$$

$$\tilde{r}_A = \max(1, ||A^T||_\infty, ||b||_\infty)^{-1}, \tag{19b}$$

$$\tilde{r}_B = \max(1, ||l||_\infty, ||u||_\infty)^{-1}. \tag{19c}$$

Convergence is achieved when $\xi < \epsilon$, where $\epsilon > 0$ is the tolerance.

3. DIAGONAL PRECONDITIONING

We consider the system matrix in augmented form, (11),

$$M = \begin{bmatrix} \bar{H} & -A \\ -A^T & 0 \end{bmatrix}, \tag{20}$$

where we recall that $\bar{H} = H + D_l + D_u$. The elements of D_l and D_u approach either 0 or ∞ as the interior-point algorithm approaches the solution. As such, M becomes increasing ill-conditioned towards later iterations of the algorithm.

We propose a diagonal preconditioning matrix to improve conditioning of the system matrix, M. Consider the precondition matrix,

$$P_{i,j}(A) = \begin{cases} A_{i,i} + \gamma_i & i = j \\ 0 & \text{otherwise} \end{cases}, \tag{21}$$

related to an arbitrary matrix, $A \in \mathbb{R}^{n \times n}$, where $\gamma_i \geq 0$. The preconditioner, $P(A)$, is easily invertible by inverting each diagonal element. Additionally, it includes a regulation parameter, γ_i, intended to avoid 0-division when inverting $P(A)$. We let $P = P(M)$ denote the preconditioner for M in (20) and choose $\gamma_i = 1.0$ for all i. The preconditioned system becomes,

$$\left(P^{-1} M\right) \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = -P^{-1} \begin{bmatrix} -\bar{r}_L \\ r_A \end{bmatrix}. \tag{22}$$

4. OPTIMAL CONTROL

In this section, we introduce the Riccati recursion algorithm for solution of structured linear systems of equations. We apply the preconditioner, (21), to the subsystems of equations solved in the Riccati recursion based solver.

4.1 Optimal Control Problem

We consider the input box-constrained OCP on the form,

$$\min_{\{u,x\}} \quad \phi = l_0(u_0) + \sum_{k=1}^{N-1} l_k(x_k, u_k) + l_N(x_N), \tag{23a}$$

$$s.t. \quad x_0 = \hat{x}_0, \tag{23b}$$

$$x_{k+1} = A_k^T x_k + B_k^T u_k + b_k, \tag{23c}$$

$$u_{\min,k} \leq u_k \leq u_{\max,k}, \tag{23d}$$

where $\{u, x\} = \{u_k, x_{k+1}\}_{k=0}^{N-1}$ and

$$l_0(u_0) = \frac{1}{2} u_0^T R_0 u_0 + r_0^T u_0 + \rho_0, \tag{24}$$

$$\begin{aligned} l_k(x_k, u_k) = & \frac{1}{2} \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} Q_k & M_k \\ M_k^T & R_k \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} \\ & + \begin{bmatrix} q_k \\ r_k \end{bmatrix}^T \begin{bmatrix} x_k \\ u_k \end{bmatrix} + \rho_k, \end{aligned} \tag{25}$$

$$l_N(x_N) = \frac{1}{2} x_N^T P_N x_N + p_N^T x_N + \rho_N. \tag{26}$$

The OCP (23) is a QP of the form (1) with,

$$x = \begin{bmatrix} u_0 & x_1 & u_1 & x_2 & \cdots & u_{N-1} & x_N \end{bmatrix}^T, \tag{27a}$$

$$H = \begin{bmatrix} R_0 & & & & & & \\ & Q_1 & M_1 & & & & \\ & M_1^T & R_1 & & & & \\ & & & \ddots & & & \\ & & & & Q_{N-1} & M_{N-1} & \\ & & & & M_{N-1}^T & R_{N-1} & \\ & & & & & & P_N \end{bmatrix}, \tag{27b}$$

$$g = \begin{bmatrix} r_0 & q_1 & r_1 & \cdots & q_{N-1} & r_{N-1} & q_N \end{bmatrix}^T, \tag{27c}$$

$$A = \begin{bmatrix} -B_0^T & I & & & & \\ & -A_1^T & -B_1^T & I & & \\ & & \ddots & \ddots & \ddots & \\ & & & -A_{N-1}^T & -B_{N-1}^T & I \end{bmatrix}^T, \quad (27d)$$

$$b = \begin{bmatrix} \tilde{b}_0 & b_1 & \cdots & b_{N-1} \end{bmatrix}^T, \quad (27e)$$

$$l = \begin{bmatrix} u_{\min,0} & -\infty & u_{\min,1} & \cdots & u_{\min,N-1} & -\infty \end{bmatrix}^T, \quad (27f)$$

$$u = \begin{bmatrix} u_{\max,0} & \infty & u_{\max,1} & \cdots & u_{\max,N-1} & \infty \end{bmatrix}^T, \quad (27g)$$

where $\tilde{b}_0 = b_0 + A_0^T x_0$.

4.2 Riccati recursion based interior-point method for OCPs

We consider the augmented system, (11), in the interior-point algorithm for the OCP (23) (for $N = 3$),

$$\left[\begin{array}{ccccccc|ccccccc} \bar{R}_0 & & & & & & & B_0 & & & & & & \\ & Q_1 & M_1 & & & & & -I & A_1 & & & & & \\ & M_1^T & \bar{R}_1 & & & & & & B_1 & & & & & \\ & & & Q_2 & M_2 & & & & -I & A_2 & & & & \\ & & & M_2^T & \bar{R}_2 & & & & & B_2 & & & & \\ & & & & & P_3 & & & & -I & & & & \\ \hline B_0^T & -I & & & & & & & & & & & & \\ & A_1^T & B_1^T & -I & & & & & & & & & & \\ & & & A_2^T & B_2^T & -I & & & & & & & & \end{array}\right] \left[\begin{array}{c} \Delta u_0 \\ \Delta x_1 \\ \Delta u_1 \\ \Delta x_2 \\ \Delta u_2 \\ \Delta x_3 \\ \hline \Delta y_0 \\ \Delta y_1 \\ \Delta y_2 \end{array}\right] = - \left[\begin{array}{c} \bar{r}_0 \\ q_1 \\ \bar{r}_1 \\ q_2 \\ \bar{r}_2 \\ p_3 \\ \hline \tilde{b}_0 \\ b_1 \\ b_2 \end{array}\right], \quad (28)$$

where $\bar{R}_k = R_k + D_{l,k} + D_{u,k}$, due to (12), (27f), and (27g). We compute the right hand side of the KKT system, (28), exactly as the right hand side of (11).

We exploit the sparse structure of (28) with a Riccati recursion based linear equation solver with linear complexity in the horizon, N (Rao et al., 1998). Algorithm 1 and 2 presents the factorization and solution phase of the Riccati recursion algorithm (Jørgensen, 2004). Notice that Riccati recursion requires solution of a set of small dense subsystems of linear equations.

4.3 Preconditioned Riccati recursion

Unavoidable ill-conditioning of the interior-point method arises in $\bar{R}_k$. Consequently, $R_{e,k}$, becomes increasing ill-conditioned. We propose to apply the diagonal preconditioner, (21), to improve conditioning of the sub-systems containing $R_{e,k}$. Consequently, the preconditioned systems are,

$$K_k = -\tilde{R}_{e,k}^{-1}\left[\tilde{P}_k^{-1}(M_k^T + B_k P_{k+1} A_k^T)\right], \quad (29a)$$

$$a_k = -\tilde{R}_{e,k}^{-1}\left[\tilde{P}_k^{-1}(r_k + B_k(P_{k+1} b_k + p_{k+1}))\right], \quad (29b)$$

$$a_0 = -\tilde{R}_{e,0}^{-1}\left[\tilde{P}_0^{-1}(r_0 + B_0(P_1 b_0 + p_1))\right], \quad (29c)$$

where $\tilde{R}_{e,k} = \tilde{P}_k^{-1} R_{e,k}$ and $\tilde{P}_k = P(R_{e,k})$.

4.4 Algorithm

Algorithm 3 presents an implementation guide of the preconditioned Riccati recursion based interior-point method.

5. RESULTS

This section presents our results based on an example application. We implement the proposed Riccati recursion based primal-dual interior-point algorithm in Matlab and consider a linearized modified quadruple tank system.

Algorithm 1: Riccati factorization

Input: $\{\bar{R}_k, Q_k, M_k, A_k, B_k\}_{k=0}^{N-1}$, P_N.

1. Compute,

$$R_{e,k} = \bar{R}_k + B_k P_{k+1} B_k^T, \quad (30a)$$

$$K_k = -R_{e,k}^{-1}(M_k^T + B_k P_{k+1} A_k^T), \quad (30b)$$

$$P_k = Q_k + A_k P_{k+1} A_k^T - K_k^T R_{e,k} K_k, \quad (30c)$$

for $k = N-1, N-2, ..., 1$ and

$$R_{e,0} = \bar{R}_0 + B_0 P_1 B_0^T. \quad (31)$$

Return: $\{R_{e,k}, P_{k+1}\}_{k=0}^{N-1}$, $\{K_k\}_{k=1}^{N-1}$.

Algorithm 2: Riccati solution

Input: $\{Q_k, M_k, A_k, B_k, R_{e,k}, P_{k+1}\}_{k=0}^{N-1}$, $\{K_k\}_{k=1}^{N-1}$.

1. Compute,

$$a_k = -R_{e,k}^{-1}(\bar{r}_k + B_k(P_{k+1} b_k + p_{k+1})), \quad (32a)$$

$$p_k = q_k + A_k(P_{k+1} b_k + p_{k+1}) + K_k^T(\bar{r}_k + B_k(P_{k+1} b_k + p_{k+1})), \quad (32b)$$

for $k = N-1, N-2, ..., 1$ and

$$a_0 = -R_{e,0}^{-1}(\bar{r}_0 + B_0(P_1 \tilde{b}_0 + p_1)). \quad (33)$$

2. Compute the solution, $\{\Delta u_k, \Delta x_{k+1}\}_{k=0}^{N-1}$,

$$\Delta u_0 = a_0, \quad (34a)$$

$$\Delta x_1 = B_0^T \Delta u_0 + \tilde{b}_0, \quad (34b)$$

and

$$\Delta u_k = K_k \Delta x_k + a_k, \quad (35a)$$

$$\Delta x_{k+1} = A_k^T \Delta x_k + B_k^T \Delta u_k + b_k, \quad (35b)$$

for $k = 1, 2, ..., N-1$.

3. Compute the Lagrange multipliers, $\{\Delta y_k\}_{k=0}^{N-1}$,

$$\Delta y_{N-1} = P_N \Delta x_N + p_N, \quad (36a)$$

$$\Delta y_{k-1} = A_k \Delta y_k + Q_k \Delta x_k + M_k \Delta u_k + q_k, \quad (36b)$$

for $k = N-1, N-2, ..., 1$.

Return: $\{\Delta u_k, \Delta x_{k+1}, \Delta y_k\}_{k=0}^{N-1}$.

5.1 Modified quadruple tank system

We apply a deterministic nonlinear model for the mass balances of the quadruple tank system of the form (Azam and Jørgensen, 2015),

$$\dot{x}(t) = f(t, x(t), u(t), d(t), p), \quad (37a)$$

$$z(t) = h(x(t)), \quad (37b)$$

where $x \in \mathbb{R}^4$ is the four masses, $u \in \mathbb{R}^2$ is the two flow rates, $d \in \mathbb{R}^2$ is the two disturbance flows in the top tanks, p are the parameters, and $z \in \mathbb{R}^2$ is the heights in the bottom tanks. We linearize the model at the steady state, $x_s = [2110.2; 1761.2; 680.6; 394.0]$ [g], achieved for $u_s = [250; 325]$ [cm^3/s] and $d_s = [100; 100]$ [cm^3/s]. The output steady state is $z_s = [55.51; 46.33]$ [cm]. Additionally, we compute the exact discretization of the linear model with sampling time, $T_s = 15$ [s]. The result is a linear state space model,

$$x_{k+1} = A_k x_k + B_k u_k + E_k d_k, \quad (38a)$$

$$z_k = C_{z,k} x_k, \quad (38b)$$

Algorithm 3: Preconditioned Riccati recursion based primal-dual interior-point algorithm

Input: H, g, A, b, l, u (as in (27)), x_0, ϵ.

- Initialize: $y = 0$, $z_l = 1$, $z_u = 1$, $s_l = 1$, $s_u = 1$.
- Calculate the scaled KKT-violation, ξ, in (18).

while $\xi > \epsilon$ **do**

1. **Predictor phase:**
   - Setup augmented system (11) with (14).
   - Compute factorization, $\{R_{e,k}, P_{k+1}\}_{k=0}^{N-1}$ and $\{K_k\}_{k=1}^{N-1}$, with Algorithm 1.
   - Solve the augmented system, (11), with Algorithm 2 for Δx^{aff} and Δy^{aff}.
   - Compute Δz_l^{aff}, Δz_u^{aff}, Δs_l^{aff}, and Δs_u^{aff} in (16).
   - Compute the affine step size, α^{aff}, with (17).
   - Compute the duality gap and centering parameter in (10).
2. **Corrector phase:**
   - Setup right-hand side of (11) with (15).
   - Solve the augmented system (11) with Algorithm 2 for Δx and Δy.
   - Compute Δz_l, Δz_u, Δs_l, and Δs_u in (16).
   - Compute the step size, α, with (17).
3. Update $(x, y, z_l, z_u, s_l, s_u)$ according to (8).
4. Calculate the scaled KKT-violation, ξ, in (18).

Return: x, y, z_l, z_u, s_l, s_u.

where

$$A_k = \begin{bmatrix} 0.8659 & 0 & 0.1246 & 0 \\ 0 & 0.8659 & 0 & 0.1246 \\ 0 & 0 & 0.8659 & 0 \\ 0 & 0 & 0 & 0.8659 \end{bmatrix}, \tag{39a}$$

$$B_k = \begin{bmatrix} 9.7793 & 0.3926 \\ 0.2944 & 8.3822 \\ 0 & 5.5882 \\ 4.1911 & 0 \end{bmatrix}, \; E_k = \begin{bmatrix} 0.9814 & 0 \\ 0 & 0.9814 \\ 13.970 & 0 \\ 0 & 13.970 \end{bmatrix}, \tag{39b}$$

$$C_{z,k} = \begin{bmatrix} 0.0026 & 0 & 0 & 0 \\ 0 & 0.0026 & 0 & 0 \end{bmatrix}, \tag{39c}$$

for all k.

We consider the target tracking OCP,

$$\min_{x,u} \quad \phi = \frac{1}{2}\sum_{k=1}^{N} ||z_k - \bar{z}_k||_{Q_z}^2 + \frac{1}{2}\sum_{k=0}^{N-1} ||u_k - \bar{u}_k||_{Q_u}^2, \tag{40a}$$

$$s.t. \quad x_0 = \hat{x}_0, \tag{40b}$$

$$x_{k+1} = A_k x_k + B_k u_k + E_k d_k, \tag{40c}$$

$$z_k = C_z x_k, \tag{40d}$$

$$u_{\min,k} \leq u_k \leq u_{\max,k}, \tag{40e}$$

where $k = 0, ..., N-1$. In the general form, (23), we have $Q_k = C_{z,k}^T Q_z C_{z,k}$, $M_k = 0$, $R_k = Q_u$, $P_N = C_{z,N}^T Q_z C_{z,N}$, $q_k = -(Q_z C_z)^T \bar{z}_k$, $p_N = -(Q_z C_z)^T \bar{z}_N$, $r_k = -Q_u \bar{u}_k$, and $b_k = E_k d_k$ for all k. Additionally, we use $Q_z = I$, $Q_u = 0$, $u_{\min,k} = [0; 0]$, $u_{\max,k} = [500; 500]$, and $d_k = [100; 100]$ for all k, with initial condition, $x_0 = x_s$, discrete horizon, $N = 200$, and a variable target, $\bar{z}_k$, over the horizon. We state the OCP in deviation variables due to the linearization.

Figure 1 shows the solution to the OCP, (40). The OCP is solved with the proposed interior-point algorithm.

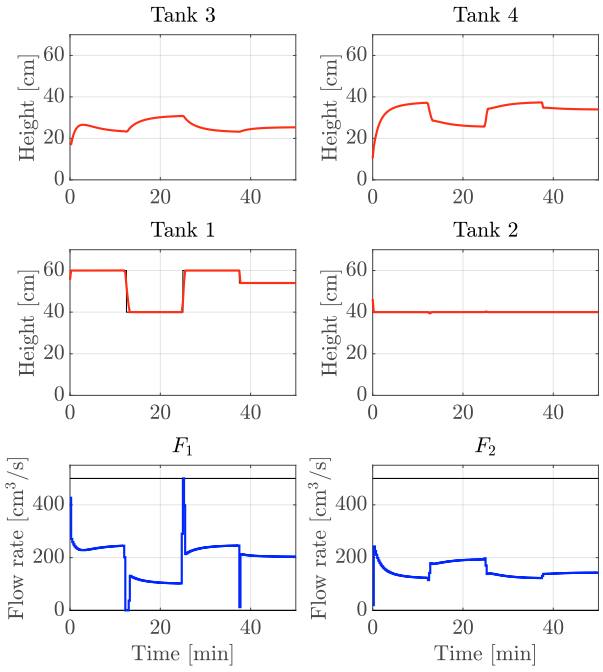

Fig. 1. Solution to target tracking OCP for linearized modified quadruple tank system with $N = 200$ and a variable target, $\bar{z}$. The controller is able to track the target in tank 1 and 2 (black line).

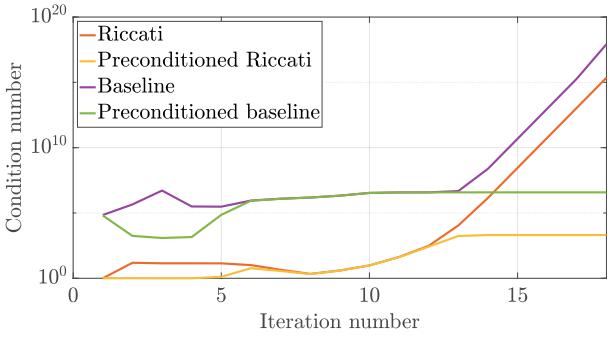

Fig. 2. Condition number of the system matrix in each iteration of the interior-point algorithm with four modes.

5.2 Condition number

We consider the interior-point algorithm in four modes,

- Baseline,
- Riccati,
- Preconditioned Baseline,
- Preconditioned Riccati,

where baseline means dense solution of the augmented system, (11). We compare the condition number of the system matrix with and without preconditioning at each iteration. For the Riccati solver, we consider the worst case condition number of the sub-system matrices. Fig. 2 presents the results. It is evident that the preconditioner improves the conditioning of the system matrices in both the baseline and Riccati recursion based interior-point algorithm.

5.3 CPU time

We apply the interior-point algorithm to solve the OCP for increasing control horizon, N. Fig. 3 presents the CPU

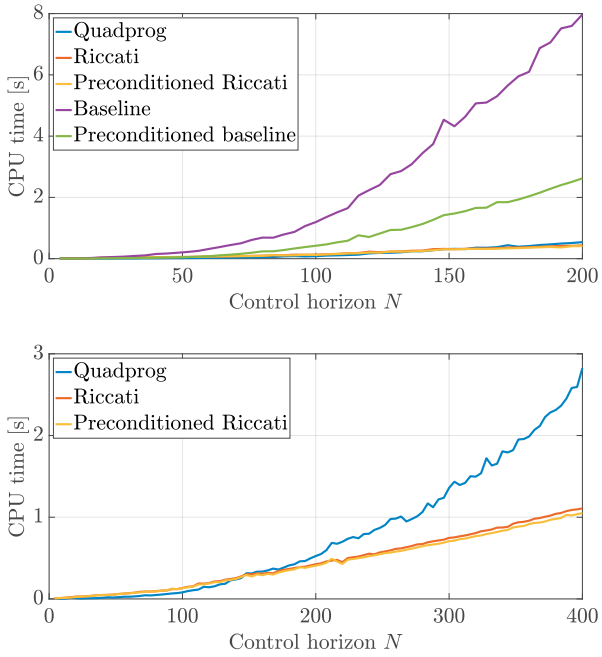

Fig. 3. CPU time for `quadprog` and interior-point algorithm. Top: Baseline and Riccati. Bottom: Riccati.

time for the interior-point method and `quadprog`. We observe that the Riccati based algorithm scales linearly in N and it is evident that the Riccati based algorithm outperforms the other algorithms for large N as expected. Additionally, the preconditioning does not increase CPU time rather it decreases the CPU time in these experiments. We point out that the implicit approach to the OCP results in a large sparse QP. Dense solvers like `quadprog` benefit from explicit approaches that produce small dense QPs. Thus, a fair CPU time comparison would require an explicit approach for `quadprog`. Such comparison is out of scope of this paper.

6. CONCLUSION

The paper presents a preconditioned Riccati recursion based interior-point algorithm tailored for QPs arising in input constrained OCPs. We implement the interior-point algorithm in Matlab and solve an OCP for target tracking of a linearized modified quadruple tank system. The results show that the diagonal preconditioner improves conditioning of the linear sub-systems of equations in the Riccati recursion.

This paper contains a detailed description of the proposed algorithm and serves as an implementation guide.

REFERENCES

Astfalk, G., Lustig, I., Marsten, R., and Shanno, D. (1992). The Interior-Point Method for Linear-Programming. *IEEE Software*, 9(4), 61–68.

Azam, S.N.M. and Jørgensen, J.B. (2015). Modeling and simulation of a modified quadruple tank system. *IEEE International Conference on Control Systems, Computing and Engineering (ICCSCE)*, 365–370.

Bergamaschi, L., Gondzio, J., and Zilli, G. (2004). Preconditioning Indefinite Systems in Interior Point Methods for Optimization. *Computational Optimization and Applications*, 28(2), 149–171.

Borrelli, F., Pekar, J., Baotić, M., and Stewart, G. (2009). On the Computation of Linear Model Predictive Control laws. *IEEE Conference on Decision and Control (CDC)*.

Byrd, R.H., Hribar, M.E., and Nocedal, J. (1999). An Interior Point Algorithm for Large-scale Nonlinear Programming. *SIAM Journal on Optimization*, 9(4), 877–900.

Cui, Y., Morikuni, K., Tsuchiya, T., and Hayami, K. (2019). Implementation of Interior-point Methods for LP based on Krylov Subspace Iterative Solvers with Inner-iteration Preconditioning. *Computational Optimization and Applications*, 74(1), 143–176.

Forsgren, A., Gill, P.E., and Wright, M.H. (2002). Interior Methods for Nonlinear Optimization. *SIAM Review*, 44(4), 525–597.

Frison, G. and Jørgensen, J.B. (2013). Efficient Implementation of the Riccati Recursion for Solving Linear-Quadratic Control Problems. *IEEE International Conference on Control Applications (CCA), Hyderabad, India*, 1117–1122.

Gertz, E.M. and Wright, S.J. (2003). Object-Oriented Software for Quadratic Programming. *ACM Transactions on Mathematical Software*, 29(1), 58–81.

Jørgensen, J.B. (2004). *Moving Horizon Estimation and Control*. Ph.D. thesis, Technical University of Denmark.

Mehrotra, S. (1992). On The Implementation of a Primal-dual Interior Point Method. *SIAM Journal on Optimization*, 2(4), 575–601.

Murray, W. (1971). Analytical expressions for the eigenvalues and eigenvectors of the Hessian matrices of barrier and penalty functions. *Journal of Optimization Theory and Applications*, 7(3), 189–196.

Nocedal, J. and Wright, S.J. (1999). *Numerical Optimization*. Springer.

Rao, C.V., Wright, S.J., and Rawlings, J.B. (1998). Application of Interior-Point Methods to Model Predictive Control. *Journal of Optimization Theory and Applications*, 99(3), 723–757.

Rawlings, J., Meadows, E., and Muske, K. (1994). Nonlinear Model Predictive Control: A Tutorial and Survey. *IFAC Advanced Control and Chemical Processes, Kyoto, Japan*.

Shahzad, A., Kerrigan, E.C., and Constantinides, G.A. (2010). A Fast Well-conditioned Interior Point Method for Predictive Control. *IEEE Conference on Decision and Control (CDC), Atlanta, USA*.

Vanderbei, R.J. (1999). LOQO:an interior point code for quadratic programming. *Optimization Methods and Software*, 11(1-4), 451–484.

Wahlgreen, M.R., Reenberg, A.T., Nielsen, M.K., Rydahl, A., Ritschel, T.K.S., Dammann, B., and Jørgensen, J.B. (2021). A High-Performance Monte Carlo Simulation Toolbox for Uncertainty Quantification of Closed-loop Systems. *IEEE Conference on Decision and Control (CDC), Accepted*.

Wright, M.H. (2004). The interior-point revolution in optimization: history, recent developments, and lasting consequences. *American Mathematical Society*, 42, 39–56.

Wächter, A. and Biegler, L.T. (2006). On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming. *Mathematical Programming*, 106(1), 25–57.