Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Active-Set based Inexact Interior Point QP Solver for Model Predictive Control

Jonathan Frey [*,**] **Stefano Di Cairano** [*] **Rien Quirynen** [*]

[*] *Mitsubishi Electric Research Laboratories, Cambridge, MA, USA.*
[**] *Department IMTEK, University of Freiburg, Germany.*

Abstract: Interior point methods are applicable to a large class of problems and can be very reliable for convex optimization, even without a good initial guess for the optimal solution. Active-set methods, on the other hand, are often restricted to linear or quadratic programming but they have a lower computational cost per iteration and superior warm starting properties. The present paper proposes an approach for improving the numerical conditioning and warm starting properties of interior point methods, based on an active-set identification strategy and inexact Newton-type optimization techniques. In addition, we show how this reduces the average computational cost of the linear algebra operations in each interior point iteration. We developed an efficient C code implementation of the active-set based interior point method (ASIPM) and show that it can be competitive with state of the art solvers for a standard case study of model predictive control stabilizing an inverted pendulum on a cart.

Keywords: Optimization algorithms, Interior point methods, Model predictive control

1. INTRODUCTION

Optimization based control and estimation techniques, such as model predictive control (MPC) and moving horizon estimation (MHE), have experienced an increasing amount of interest from industry (Rawlings et al., 2017). At each sampling instant, MPC solves an optimal control problem (OCP), in which a particular cost function is minimized subject to continuity conditions and inequality constraints. A block-sparse quadratic program (QP) structure arises in linear or linear time-varying formulations. A similarly structured QP defines the subproblems within sequential quadratic programming (SQP) for nonlinear optimal control (Gros et al., 2016).

In this paper, we aim to solve the following convex constrained linear-quadratic optimal control problem (OCP)

$$\min_{X,U} \quad \sum_{k=0}^{N-1} \frac{1}{2} \begin{bmatrix} x_k \\ u_k \end{bmatrix}^\top \begin{bmatrix} Q_k & S_k^\top \\ S_k & R_k \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + \begin{bmatrix} q_k \\ r_k \end{bmatrix}^\top \begin{bmatrix} x_k \\ u_k \end{bmatrix} \tag{1a}$$

$$+ \frac{1}{2} x_N^\top Q_N x_N + q_N^\top x_N \tag{1b}$$

$$\text{s.t.} \quad x_0 = \hat{x}_0, \tag{1c}$$

$$x_{k+1} = a_k + A_k x_k + B_k u_k, \ k = 0, \dots, N-1, \tag{1d}$$

$$d_k \geq D_k^{\mathrm{x}} x_k + D_k^{\mathrm{u}} u_k, \qquad k = 0, \dots, N, \tag{1e}$$

where $x_k \in \mathbb{R}^{n_{\mathrm{x}}}$ is the state vector, $u_k \in \mathbb{R}^{n_{\mathrm{u}}}$ is the vector of control inputs, $Q_k \in \mathbb{R}^{n_{\mathrm{x}} \times n_{\mathrm{x}}}$, $S_k \in \mathbb{R}^{n_{\mathrm{u}} \times n_{\mathrm{x}}}$ and $R_k \in \mathbb{R}^{n_{\mathrm{u}} \times n_{\mathrm{u}}}$ are the cost function weight matrices, and N is the prediction horizon. Further, we denote the predicted state and control trajectories as $X := [x_0^\top, \dots, x_N^\top]^\top$ and $U := [u_0^\top, \dots, u_{N-1}^\top]^\top$, respectively. The constraints in QP (1) include the system dynamics with $A_k \in \mathbb{R}^{n_{\mathrm{x}} \times n_{\mathrm{x}}}$, $B_k \in \mathbb{R}^{n_{\mathrm{x}} \times n_{\mathrm{u}}}$, $a_k \in \mathbb{R}^{n_{\mathrm{x}}}$, inequality constraints with $D_k^{\mathrm{x}} \in \mathbb{R}^{n_{\mathrm{c},k} \times n_{\mathrm{x}}}$, $D_k^{\mathrm{u}} \in \mathbb{R}^{n_{\mathrm{c},k} \times n_{\mathrm{u}}}$ and an initial value condition where $\hat{x}_0 \in \mathbb{R}^{n_{\mathrm{x}}}$ denotes the current state estimate.

A real-time implementation of MPC requires the solution of the block-structured QP (1) within a specified time period, typically on embedded control hardware with limited computational resources and a relatively small amount of available memory (Ferreau et al., 2017; Di Cairano and Kolmanovsky, 2018). Generally, there is a trade-off between solvers that make use of second-order information and require only few but computationally complex iterations, including active-set strategies (Ferreau et al., 2014; Quirynen and Cairano, 2019) and interior point algorithms (Domahidi and Perez, 2014; Frison et al., 2014), versus first-order methods that are of low complexity but may require many more iterations, e.g., ADMM (Raghunathan and Di Cairano, 2015) and other gradient or splitting-based methods (Ferreau et al., 2017). Similar to the software tools `FORCES` (Domahidi and Perez, 2014) and `HPMPC` (Frison et al., 2014), we focus on interior point methods that exploit the block-structured sparsity.

When solving a set of QPs with a varying number of active constraints (Bartlett et al., 2000), interior point methods (IPMs) typically provide a smaller variation in the number of iterations compared to active-set (AS) solvers. However, each IPM iteration is typically more computationally expensive than an AS one, because AS solvers exploit low-rank factorization updates (Wright, 1996). In MPC applications, the average computational cost of AS methods can be further reduced by warm starting, which is generally more complicated for IPMs (Shahzad et al., 2010b). As a consequence, when solving QPs, IPMs may outperform AS in worst-case performance but they may have a longer average computation time.

The aim of this paper is to reduce the average computational cost per IPM iteration and to improve the warm starting capabilities of IPMs, while avoiding an undesirable increase of their worst-case computational perfor-

mance. We extend prior work on IPM-tailored inexact Newton-type techniques (Shahzad et al., 2010a) and warm starting strategies (Shahzad et al., 2010b; Zanelli et al., 2017) for IPMs. Here, we propose an inexact Newton-type implementation that results in a numerically robust active-set identification strategy and allows for using low-rank factorization update techniques and tailored warm starting procedures in the IPM. We present an efficient C code implementation of the resulting active-set based interior point method (`ASIPM`) and show that it can be competitive with state of the art solvers for a case study of MPC stabilizing an inverted pendulum on a cart.

The paper is organized as follows. Section 2 briefly introduces the standard IPM. Next, Section 3 proposes the novel inexact Newton-type algorithm and Section 4 describes its block-sparse structure exploitation. Section 5 discusses tailored warm starting strategies. The `ASIPM` solver code and results of the MPC case study are presented in Section 6 and 7. Section 8 concludes the paper.

2. INTERIOR POINT METHODS FOR LINEAR-QUADRATIC OPTIMAL CONTROL

We introduce the compact notation for the Hessian H, and matrices F and G, respectively, denoting the equality (1d) and inequality constraints (1e), as follows

$$H := \begin{bmatrix} Q_0 & S_0^\top & \dots & \mathbb{0} \\ S_0 & R_0 & \dots & \mathbb{0} \\ \vdots & \vdots & \ddots & \\ \mathbb{0} & \mathbb{0} & \dots & Q_N \end{bmatrix}, \quad G := \begin{bmatrix} D_0^{\mathrm{x}} & D_0^{\mathrm{u}} & \dots & \mathbb{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{0} & \mathbb{0} & \dots & D_N^{\mathrm{x}} \end{bmatrix},$$

$$F := \begin{bmatrix} -\mathbb{1} & \mathbb{0} & \mathbb{0} & \mathbb{0} & \dots & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ A_0 & B_0 & -\mathbb{1} & \mathbb{0} & \dots & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \mathbb{0} & \dots & \mathbb{0} & \mathbb{0} & \mathbb{0} & A_{N-1} & B_{N-1} & -\mathbb{1} \end{bmatrix}. \tag{2}$$

We define the gradient vector $h := [q_0^\top, r_0^\top, \dots, q_N^\top]^\top$, and the constraint vectors $f := -[\hat{x}_0^\top, a_0^\top, \dots, a_{N-1}^\top]^\top$ and $g := [d_0^\top, \dots, d_N^\top]^\top$. Based on this compact notation, we write the relaxed Karush-Kuhn-Tucker (KKT) conditions

$$Hz + F^\top \lambda + G^\top \mu + h = 0, \tag{3a}$$
$$Fz - f = 0, \tag{3b}$$
$$Gz - g + s = 0, \tag{3c}$$
$$MS\mathbf{1} - \tau \mathbf{1} = 0, \tag{3d}$$
$$\mu, s \geq 0, \tag{3e}$$

where we introduced $z := [x_0^\top, u_0^\top, \dots, x_N^\top]^\top$, the Lagrange multipliers λ and μ for the equality and inequality constraints, respectively, and we defined $\mathbf{1} := [1, \dots, 1]^\top \in \mathbb{R}^{n_{\mathrm{ieq}}}$ and the diagonal matrices $M := \operatorname{diag}(\mu)$ and $S := \operatorname{diag}(s)$. In (3c), we introduced slack variables $s \geq 0$ for the inequality constraints $Gz \leq g$. In (3d), we smoothened the complementarity conditions using the barrier parameter τ. A primal-dual IPM solves the smoothened system of KKT conditions in (3) based on a Newton-type method for a sequence of values of the barrier parameter $\tau \to 0$. More specifically, in the k^{th} iteration of the IPM, the search direction is obtained by solving the linear system

$$\begin{bmatrix} H & F^\top & G^\top & \mathbb{0} \\ F & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ G & \mathbb{0} & \mathbb{0} & \mathbb{1} \\ \mathbb{0} & \mathbb{0} & S^k & M^k \end{bmatrix} \begin{bmatrix} \Delta z^k \\ \Delta \lambda^k \\ \Delta \mu^k \\ \Delta s^k \end{bmatrix} = - \begin{bmatrix} r_z^k \\ r_\lambda^k \\ r_\mu^k \\ r_s^k \end{bmatrix}, \tag{4}$$

where the right-hand side is given by

$$r_z^k = Hz^k + F^\top \lambda^k + G^\top \mu^k + h, \quad r_\mu^k = Gz^k - g + s^k, \tag{5a}$$
$$r_\lambda^k = Fz^k - f, \quad r_s^k = M^k S^k \mathbf{1} - \tau^k \mathbf{1}. \tag{5b}$$

We eliminate the slack variables Δs^k from system (4) and introduce the notation $w \in \mathbb{R}^{n_{\mathrm{ieq}}}$, $w_i := \frac{s_i}{\mu_i}$ and $W := \operatorname{diag}(w)$. This results in the reduced form of the linearized KKT system

$$\begin{bmatrix} H & F^\top & G^\top \\ F & \mathbb{0} & \mathbb{0} \\ G & \mathbb{0} & -W^k \end{bmatrix} \begin{bmatrix} \Delta z^k \\ \Delta \lambda^k \\ \Delta \mu^k \end{bmatrix} = - \begin{bmatrix} r_z^k \\ r_\lambda^k \\ \bar{r}_\mu^k \end{bmatrix}, \tag{6}$$

where $\bar{r}_\mu^k = r_\mu^k - M^{k^{-1}} r_s^k$ and $\Delta s^k = -M^{k^{-1}}(S^k \Delta \mu^k + r_s^k)$.

3. INEXACT NEWTON-TYPE METHOD FOR IMPROVED NUMERICAL CONDITIONING

In the following, we propose a novel inexact Newton-type implementation of the IPM that has following advantages over standard implementations:

(1) Improved numerical conditioning for the approximated KKT matrix by bounding of the w-values.
(2) Classification of inequality constraints leads to a reduced computational cost by restricting to a smaller set of potentially active constraints (see Section 4).
(3) Tailored warm starting strategies that exploit the constraint classification procedure (see Section 5).

3.1 Classification of Inequality Constraints in IPM

We follow and extend the idea of δ-inactive constraints in (Shahzad et al., 2010a) and propose a new classification based on $w_i := \frac{s_i}{\mu_i}$ for each $i = 1, \dots, n_{\mathrm{ieq}}$. For inequality constraints that are strictly active at the optimal solution, $s_i \to 0$ and $\mu_i > 0$ such that $w_i \to 0$ for $k \to \infty$. For inactive inequality constraints, $\mu_i \to 0$ and $s_i > 0$ such that $w_i \to \infty$ for $k \to \infty$. Thus, the w-values become increasingly small and large for active and inactive inequality constraints, respectively, which highlights the numerical ill-conditioning that must be tackled when implementing IPMs (Shahzad et al., 2010a).

Based on lower and upper bound values $w_{\min} \ll w_{\max}$, we classify constraints into the following three categories:

(1) *inactive*: constraints that are likely to be inactive at the solution, with index set $\mathcal{I}_{\mathrm{in}} := \{\, i \mid w_i \geq w_{\max} \,\}$.
(2) *active*: constraints that are likely to be active at the solution, with index set $\mathcal{I}_{\mathrm{act}} := \{\, i \mid w_i \leq w_{\min} \,\}$.
(3) *guessing*: constraints that are neither in the active nor in the inactive range, with index set $\mathcal{I}_{\mathrm{g}}$.

All inequality constraints can be classified in exactly one of these categories, i.e., the three index sets are disjoint and $\mathcal{I}_{\mathrm{act}} \cup \mathcal{I}_{\mathrm{g}} \cup \mathcal{I}_{\mathrm{in}} = \{1, \dots, n_{\mathrm{ieq}}\}$. The w-values at different iterations of our IPM, with cold started initial guess, and their classification are shown in Figure 1. The wide range of w-values when the IPM is near convergence leads to numerical ill-conditioning.

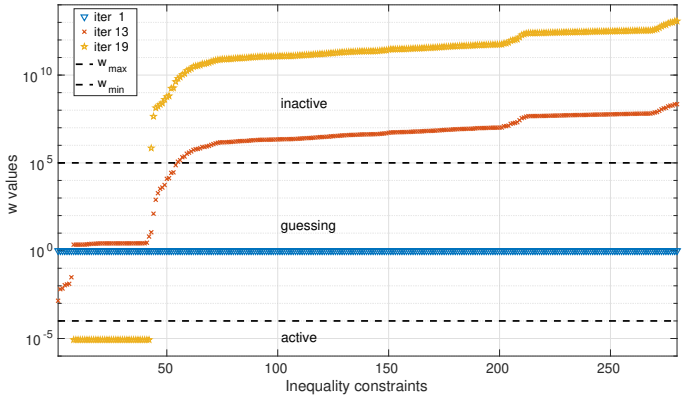

Fig. 1. Distribution and classification of w-values for particular iterations of a cold started IPM call.

3.2 Inexact IPM: KKT Matrix Approximation

Based on the above classification of constraints, we present an approximation of the KKT matrix leading to our proposed inexact IPM implementation that improves the numerical conditioning, reduces the computational cost and allows effective warm starting strategies. Instead of the exact linearized KKT system in (6), our proposed inexact Newton-type implementation solves the following system to compute an approximated search direction

$$\left[\begin{array}{cccc|c} H & F^\top & G_{\text{act}}^\top & G_{\text{g}}^\top & \mathbb{0} \\ F & -\varepsilon\mathbb{1} & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ G_{\text{act}} & \mathbb{0} & -\varepsilon\mathbb{1} & \mathbb{0} & \mathbb{0} \\ G_{\text{g}} & \mathbb{0} & \mathbb{0} & -W_{\text{g}}^k & \mathbb{0} \\ \hline G_{\text{in}} & \mathbb{0} & \mathbb{0} & \mathbb{0} & -W_{\text{in}}^k \end{array}\right] \left[\begin{array}{c} \Delta z^k \\ \Delta\lambda^k \\ \Delta\mu_{\text{act}}^k \\ \Delta\mu_{\text{g}}^k \\ \Delta\mu_{\text{in}}^k \end{array}\right] = - \left[\begin{array}{c} r_z^k \\ r_\lambda^k \\ \bar{r}_{\mu,\text{act}}^k \\ \bar{r}_{\mu,\text{g}}^k \\ \bar{r}_{\mu,\text{in}}^k \end{array}\right], \tag{7}$$

where the inequality constraints and w-values have been reordered and grouped together according to their category, i.e., we have defined the diagonal matrices $W_{\text{in}}^k := W_{i\in\mathcal{I}_{\text{in}}}^k$, $W_{\text{act}}^k := W_{i\in\mathcal{I}_{\text{act}}}^k$ and $W_{\text{g}}^k := W_{i\in\mathcal{I}_{\text{g}}}^k$. Similarly, we split G and $\bar{r}_\mu$ into the corresponding blocks.

The approximations in the KKT matrix of each inexact Newton-type iteration (7) are:

- neglecting contributions of $G_{\text{in}}^\top \Delta\mu_{\text{in}}^k$ to the first row of optimality conditions, because the corresponding Lagrange multiplier values are sufficiently small;
- lower bounding of the diagonal block matrices in (7) that correspond to the equality constraints and active inequality constraints, i.e., $W_{\text{act}}^k \approx \varepsilon\mathbb{1}$.

The latter lower bounding results in an augmented Lagrangian type regularization of the linearized KKT system (Malyshev et al., 2018). The lower bound $w_{\min}$ can typically be chosen equal to the value for the regularization parameter, i.e., $w_{\min} = \varepsilon$. The accuracy of the approximation, and therefore the convergence rate of the Newton-type method, can be manipulated by choosing appropriate values for $w_{\min}$ and $w_{\max}$. However, a detailed analysis of the convergence properties for the proposed inexact Newton-type IPM is part of ongoing research.

4. TAILORED SPARSITY EXPLOITING DIRECT LINEAR SYSTEM SOLUTION

The main computational cost of each IPM iteration is the computation of the search direction by solving the linearized KKT system (4) for which many different approaches have been proposed, e.g., in (Domahidi and Perez, 2014; Frison et al., 2014; Shahzad et al., 2010a). Here, we propose a specific implementation that is tailored to the structure of the inexact Newton system in (7).

4.1 Direct Solution of Reduced Linear System

We start with the relatively standard numerical elimination of the Lagrange multiplier values by inverting the nonsingular diagonal matrices in (7). This results in the reduced linear system to compute the inexact search direction of the primal variables

$$\underbrace{\left[H + \frac{1}{\varepsilon}F^\top F + \frac{1}{\varepsilon}G_{\text{act}}^\top G_{\text{act}} + G_{\text{g}}^\top W_{\text{g}}^{k^{-1}} G_{\text{g}}\right]}_{:=\mathcal{M}^k} \Delta z^k = -\bar{r}_z^k, \tag{8}$$

where $\bar{r}_z^k = r_z^k + \frac{1}{\varepsilon}F^\top r_\lambda^k + \frac{1}{\varepsilon}G_{\text{act}}^\top \bar{r}_{\mu,\text{act}}^k + G_{\text{g}}^\top W_{\text{g}}^{k^{-1}} \bar{r}_{\mu,\text{g}}^k$. When considering (7), the reduced linear system does not depend on inactive inequality constraints, due to neglecting contributions of $G_{\text{in}}^\top \Delta\mu_{\text{in}}^k$ to the first row of optimality conditions. In addition, the corresponding search directions for the Lagrange multipliers can be computed as

$$\Delta\lambda^k = \frac{1}{\varepsilon}\left(r_\lambda^k + F\Delta z^k\right), \qquad \Delta\mu_{\text{act}}^k = \frac{1}{\varepsilon}\left(\bar{r}_{\mu,\text{act}}^k + G_{\text{act}}\Delta z^k\right), \tag{9a}$$

$$\Delta\mu_{\text{g}}^k = W_{\text{g}}^{k^{-1}}\left(\bar{r}_{\mu,\text{g}}^k + G_{\text{g}}\Delta z^k\right), \; \Delta\mu_{\text{in}}^k = W_{\text{in}}^{k^{-1}}\left(\bar{r}_{\mu,\text{in}}^k + G_{\text{in}}\Delta z^k\right). \tag{9b}$$

4.2 Block-tridiagonal Cholesky Factorization

The inexact coefficient matrix $\mathcal{M}^k$ in (8) is positive definite and has a block-tridiagonal sparsity structure due to the block-sparse Hessian and constraint matrices in (2). A block-tridiagonal Cholesky factorization is used to solve the corresponding linear system. The dominating term in the computational cost of the factorization is

$$N\left(\frac{7}{3}n_{\text{x}}^3 + 4n_{\text{x}}^2 n_{\text{u}} + 2n_{\text{x}}n_{\text{u}}^2 + \frac{1}{3}n_{\text{u}}^3\right), \tag{10}$$

where terms that are quadratic or linear in n_{x} and n_{u} and those independent of N are dropped. The resulting block-structured backsolves require

$$N\left(4\,n_{\text{x}}^2 + 6\,n_{\text{x}}n_{\text{u}} + 2\,n_{\text{u}}^2\right) \tag{11}$$

floating point operations. It is clear that the arithmetic cost for the factorization in (10) is one order of magnitude larger than the cost of using it to solve the linear system (8) in (11). In addition, as discussed also in (Malyshev et al., 2018), the arithmetic cost of this particular block-tridiagonal Cholesky factorization is equal to that of the factorized Riccati recursion that was proposed originally in (Frison and Jørgensen, 2013). For our particular inexact IPM implementation, we can exploit the problem structure to further reduce the computational complexity of this block-tridiagonal Cholesky factorization.

4.3 Block Updates of Reverse Cholesky Factorization

First, not all of the blocks in the block-tridiagonal coefficient matrix $\mathcal{M}^k$ in (8) are updated from one inexact IPM iteration to the next, due to our proposed constraint classification and KKT matrix approximation strategy in Section 3. More specifically, a particular block remains unchanged if none of the inequality constraints within the corresponding stage of the prediction horizon $k = 0, \dots, N$ are in the *guessing* range. Second, due to the difference between prediction model and actual system and the stabilizing effect of the cost function in MPC, the active-set changes are more likely to occur for stages at the beginning of the prediction horizon. Similar to the dual Newton strategy in `qpDUNES` (Frasch et al., 2015), we use the reverse variant of the block-tridiagonal Cholesky factorization. We denote the last block that has been changed as $k = N_{\text{up}}$ and reuse the factorization for all blocks $k = N_{\text{up}} + 1, \dots, N$. Therefore, only the first $N_{\text{up}} \leq N$ blocks of the reverse Cholesky factorization need to be updated, resulting in an arithmetic cost of

$$N_{\text{up}} \left(\frac{7}{3} n_{\text{x}}^3 + 4 n_{\text{x}}^2 n_{\text{u}} + 2 n_{\text{x}} n_{\text{u}}^2 + \frac{1}{3} n_{\text{u}}^3 \right) \tag{12}$$

instead of the total cost from (10). As illustrated by Figure 2, these low-rank block-structured factorization updates are particularly effective in combination with the warm starting strategies that are described in Section 5.

Remark 1. In addition to the reuse of the last $N - N_{\text{up}}$ blocks, we could perform rank-1 block-structured factorization updates in order to efficiently take into account the changes that correspond to a relatively small amount of inequality constraints over a large range of control stages.

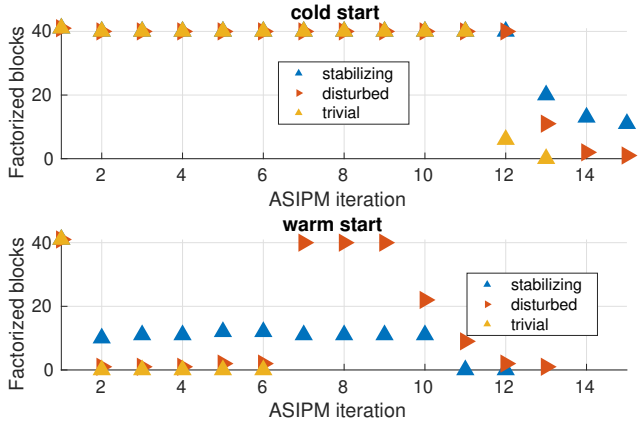

Fig. 2. Number of factorized blocks within `ASIPM` for pendulum stabilization benchmark: cold start (top) and warm start (bottom) in the nominal case, an example with disturbance, and for a trivial problem when only bounds on slack variables are active.

4.4 Structure Exploitation for Auxiliary Control Variables

In real-world applications of MPC Di Cairano and Kolmanovsky (2018), it is difficult to ensure the satisfaction of state-dependent inequality constraints $h(x) \leq 0$ in closed loop due to model mismatch and disturbances. Therefore, soft constraint reformulations $h(x) \leq s$ with slack variables $s \geq 0$ and exact penalty terms are typically used (Fletcher, 1987). Even when no state-dependent inequality constraints are active at the optimal solution, the simple bound constraints on the slack variables would be active at each control stage to impose each of the slack variables to be equal to zero. Depending on the $w_{\min}$ value, this could mean that N_{up} is large and many of the blocks are updated in each IPM iteration.

However, an additional sparsity structure arises in the block-tridiagonal coefficient matrix $\mathcal{M}^k$ in case of a simple bound on an optimization variable that does not appear in any cross-terms in the quadratic cost function, nor in any other inequality constraints that are not inactive or in any of the equality constraints. The latter results in a contribution on the diagonal of the coefficient matrix while the off-diagonal elements are zero for that row and column, corresponding to an update of the Cholesky factorization that does not scale with any of the dimensions in the problem. Therefore, simple bounds on slack variables can be ignored in the determination of the number of blocks N_{up} to be updated, when all of the corresponding soft constraints are inactive in that particular IPM iteration.

5. WARM STARTING OF ACTIVE-SET BASED INTERIOR POINT METHOD

As illustrated by Figure 2, there is little variation in the number of iterations for a cold started IPM when solving a trivial versus a difficult QP. Instead, warm starting for our proposed inexact IPM solver can be particularly effective because it results in the combined effect of

- reducing the average, and maybe even the worst case, number of IPM iterations to reach convergence;
- reducing the average, and often also the worst case, computational cost per iteration due to the reuse of blocks in the reverse Cholesky factorization.

5.1 Dual Warm Starting with Constraint Inactivation

The issue that limits warm starting in IPMs is that convergence can be (much) slower due to the non-smooth complementarity conditions $s_i \mu_i = 0$ for $i = 1, \dots, n_{\text{ieq}}$, when the active set for the initial guess is different from the optimal active set. Therefore, it is generally not a good idea to initialize an IPM with the shifted solution to the QP at the previous time step. Instead, we propose to perform a constraint *inactivation* procedure to the time shifted solution before using it as an initial guess:

(1) Compute time shifted solution guess $\left(z^0, \lambda^0, \mu^0, s^0\right)$ from the QP solution at the previous time step.
(2) Scale the initial values for the slack variables upward $\tilde{s}^0 = \kappa\, s^0$ with a scaling factor $\kappa \gg 1$.
(3) Compute barrier parameter value $\tau^0 = \frac{\tilde{s}^{0\top} \mu^0}{n_{\text{ieq}}}$.
(4) Use resulting initial guess $\left(z^0, \lambda^0, \mu^0, \tilde{s}^0, \tau^0\right)$ for IPM.

This procedure corresponds to *inactivating* the inequality constraints that were active at the previous time step by increasing the values for the corresponding slack variables. The idea is to choose the value of the scaling factor $\kappa \gg 1$ sufficiently large, but not too large, such that all active constraints move back into the guessing range

instead. This avoids the risk of making a mistake in believing an inactive constraint to be active. In addition, the resulting increase in the initial guess for the barrier parameter results in a relaxed and therefore smoothened set of complementarity conditions.

Unlike the warm starting strategy that was proposed in (Shahzad et al., 2010b), the above procedure initializes the dual variables with the shifted solution at the previous time step and preserves the general distribution of w-values from one IPM call to the next (see Figure 1).

5.2 Warm Starting based on Smooth Relaxed Solution

Based on the same concepts that motivated the previous warm starting strategy, an alternative idea is to use a shifted trajectory of a smooth relaxed solution to the QP at the previous time step instead of the optimal solution. More specifically, one needs to store an inexact relaxed solution, for which a particular set of conditions is satisfied, that can be used as a warm start to the next problem. For example, the set of values $\left(z^k, \lambda^k, \mu^k, s^k\right)$ can be stored for a particular IPM iteration for which the conditions $\tau^k \leq \tau_{\min}$ and $\|r^k\| \leq r_{\min}$ hold, where $\|r^k\|$ corresponds to the norm of the right-hand side in (4).

5.3 Tailored Constraint Reactivation Strategy

The two warm starting strategies described above aim at avoiding convergence issues, when making the wrong decision on which of the constraints are active, due to the non-smoothness of the complementarity conditions. However, convergence can still be slow due to inequality constraints that are wrongly identified to be inactive in the initial solution guess. Therefore, we include an additional constraint *reactivation* procedure that can be used in case of bad initializations, e.g., due to disturbances.

The first step is to accurately detect a bad initialization of the IPM, which we propose to address by monitoring the step sizes at successive iterations. If the step size is below a particular threshold $\alpha_{\min}$ and the step size has decreased over a particular number of iterations $\bar{n}_{\mathrm{d}}$, then the reactivation procedure is invoked. In the latter case, the Lagrange multiplier values for the inactive inequality constraints are scaled upward in order to reactivate a particular percentage $0 < \gamma \leq 1$ of the constraints in the inactive range. The parameters $\alpha_{\min}$, $\bar{n}_{\mathrm{d}}$ and γ must be properly selected, otherwise the method may result in a considerable increase in the computational cost, since it can lead to the computation of many more Cholesky blocks as illustrated in Figure 2. We outline the resulting constraint reactivation procedure in Algorithm 1.

Algorithm 1 Constraint reactivation procedure for IPM

Input: Current value n_{d}, $\bar{n}_{\mathrm{d}}$, $\alpha_{\min}$, and γ.
if Decreasing step size $\alpha^k < \alpha^{k-1}$ **then**
  $n_{\mathrm{d}} \leftarrow n_{\mathrm{d}} + 1$.
  if $n_{\mathrm{d}} \geq \bar{n}_{\mathrm{d}}$ and $\alpha < \alpha_{\min}$ **then** ▷ Reactivate
    # of constraints to reactivate: $n_{\mathrm{a}} = \lceil \gamma \, |\mathcal{I}_{\mathrm{in}}| \rceil$.
    Compute $\beta > 1$ to reactivate n_{a} constraints.
    $\mu_i \leftarrow \beta \, \mu_i$, $i \in \mathcal{I}_{\mathrm{in}}$.
  end if
else
  $n_{\mathrm{d}} \leftarrow 0$. ▷ Reset counter
end if

Algorithm 2 Active-set based Inexact IPM for MPC

Warm starting strategy from Section 5.1 or 5.2.
while $\mathtt{max}(\tau^k, \|r^k\|) > \mathtt{tol}$ **do**
  Call reactivation procedure in Algorithm 1.
  N_{up}-block update of sparse factorization for $\mathcal{M}^k$.
  Form and solve reduced linear system in (8).
  Compute dual search direction step in (9b).
  Step size α^k to ensure positivity constraints.
  $(z, \lambda, \mu, s)^k \leftarrow (z, \lambda, \mu, s)^{k-1} + \alpha^k \Delta \, (z, \lambda, \mu, s)^k$.
end while
Output: Next optimal control input values $u_0^\star$.

6. IMPLEMENTATION DETAILS AND SOFTWARE

The overall description of our proposed `ASIPM` implementation can be found in Algorithm 2. Next, we briefly describe the C code software implementation before presenting the numerical simulation results.

6.1 Tailored Heuristics for Inexact Active-set based IPM

In addition to the algorithmic techniques that have been presented in the previous three sections, there are multiple tailored heuristics that make our `ASIPM` algorithm implementation effective in practice. For example, an adaptive step-size selection strategy is used to increasingly favor taking the maximum step size that satisfies the positivity constraints for slack variables and Lagrange multipliers. In addition, the proposed `ASIPM` algorithm switches between a standard IPM versus a predictor-corrector implementation, as originally proposed in (Mehrotra, 1992), depending on the relative computational cost of the block-tridiagonal Cholesky factorization in (12) versus the linear system solution in (11). In fact, the predictor-corrector IPM can provide a computational speedup if and only if the linear system solution is relatively cheap.

6.2 `ASIPM` Solver Implementation: Self-contained C code

We prepared a preliminary self-contained C code implementation of the `ASIPM` solver for fast MPC applications, which does not rely on external libraries, e.g., for performing linear algebra routines. It is known that such operations can be performed more efficiently using advanced Basic Linear Algebra Subprogram (BLAS) libraries. For instance, the open-source `BLASFEO` (Frison et al., 2017) framework provides hardware-tailored optimized dense BLAS routines, outperforming most alternative tools for small-to-medium-scale matrix dimensions that occur typically for applications of embedded optimization. However, external libraries may not be supported by micro-controller platforms for embedded implementation. Thus, our C code implementation is written such that the solver can easily be embedded in prototyping platforms as well as in micro-controllers that are used in real-world mechatronic systems (Di Cairano and Kolmanovsky, 2018). Despite this, our simple, self-contained C code implementation will be shown to remain very competitive with state of the art QP solvers, thanks to exploiting its active-set strategy for reduced block-structured factorization updates and tailored warm starting techniques.

7. NUMERICAL SIMULATION RESULTS: MPC ON AN INVERTED PENDULUM

We consider the standard case study of an inverted pendulum on a cart, and present closed-loop numerical simulation results of a linearized MPC for stabilization.

7.1 Randomized Benchmark of Linear MPC Simulations

The $n_x = 4$ state variables include the position $x_C(t)$ and velocity $v_C(t)$ of the cart, and the angle $\theta(t)$ and angular velocity $\dot{\theta}(t)$ of the pendulum. The $n_u = 2$ control variables are the force $u_F(t)$ that is applied to the mass at the end of the pendulum and the slack variable $s(t) \geq 0$. The latter is used to reformulate state constraints on the position and velocity of the cart into soft constraints

$$x_C - s \leq x_{\max}, \qquad -x_{\max} \leq x_C + s, \tag{13a}$$
$$v_C - s \leq v_{\max}, \qquad -v_{\max} \leq v_C + s, \tag{13b}$$
$$-u_{\max} \leq u_F \leq u_{\max}, \qquad 0 \leq s. \tag{13c}$$

We use a prediction horizon of $N = 40$ control intervals and a sampling time of $T_s = 0.1$ s for the MPC controller. More details on the OCP formulation for the inverted pendulum example can be found in (Quirynen et al., 2014).

We created a randomized benchmark of 100 test scenarios of linearized MPC, based on a steady state linearization of the nonlinear system dynamics for the inverted pendulum, with different initial state values and external disturbances of the applied control input values. The disturbances are simulated as an instantaneous external force that is applied once for the duration of one sampling period in the middle of each closed-loop simulation, to the mass at the end of the pendulum. The initial state values consist of zero velocities and random position values that are uniformly generated in the union of the intervals $[-1, -0.5]\, x_{\max} \cup [0.5, 1]\, x_{\max}$ and $[-1, -0.5]\, \theta_{\max} \cup [0.5, 1]\, \theta_{\max}$, respectively. Similarly, the randomized disturbance values Δu are uniformly sampled in the union of the intervals $[-1, -0.5]\, u_{\max} \cup [0.5, 1]\, u_{\max}$.

7.2 Numerical Results: Comparison of MPC Solvers

The timing results for the benchmark of 100 randomized closed-loop linear MPC simulations are presented in Figure 3. We compare a range of state of the art QP solvers for MPC against our proposed `ASIPM` algorithm, including `qpOASES` (Ferreau et al., 2014), `OSQP` (Stellato et al., 2017), `PRESAS` (Quirynen and Cairano, 2019), `qpDUNES` (Frasch et al., 2015) and `HPIPM` (Frison and Diehl, 2020). Note that the results in Figure 3 are based on warm starting and using a solution tolerance of 10^{-6} for each of the QP solvers. The figure shows a colored rectangular area that indicates the minimum and maximum number of iterations and corresponding computation time for solving one QP, while additionally the average value is indicated by a black line. It can be observed that our proposed `ASIPM` solver has the best average performance and it additionally outperforms most of the other solvers in best- and worst-case performance. Even though one cannot make conclusions from one benchmark case study, these preliminary results are aligned with the aim of `ASIPM` to provide the best-case performance of a structure-exploiting active-set solver like `PRESAS` in combination with the worst-case performance of a highly optimized IPM such as `HPIPM`.

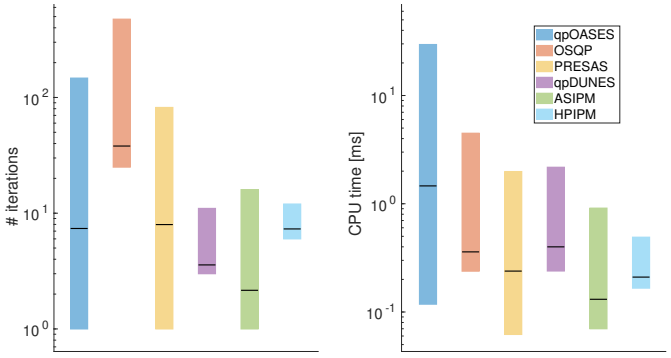

Fig. 3. Minimum, average and maximum number of iterations and timing results for a range of solvers on the benchmark of 100 randomized closed-loop linear MPC simulations on the inverted pendulum example. [1]

7.3 Warm Starting of Active-Set based IPM

The effectiveness of our proposed warm-starting strategies are illustrated in Table 1 that shows the computational results of the `ASIPM` solver on the benchmark of 100 randomized closed-loop linear MPC simulations, when using a cold start, the dual warm-starting method with constraint inactivation from Section 5.1 or the smooth relaxed initialization that was presented in Section 5.2. Due to our tailored constraint reactivation strategy, described in Algorithm 1, the proposed warm starting procedures preserve the worst-case performance of the cold started solver while resulting in a considerable improvement of its average computational performance.

Table 1. Timing results for the `ASIPM` solver, with a cold start or two tailored warm-start initialization strategies on the benchmark of 100 randomized closed-loop MPC simulations. [1]

	# IPM iterations			CPU time [ms]		
`ASIPM`	min	mean	max	min	mean	max
cold	10.0	12.0	16.0	0.56	0.70	0.99
dual	2.0	3.5	18.0	0.12	0.21	1.30
relaxed	1.0	2.2	16.0	0.08	0.16	1.12

7.4 Detailed Timing Results on Raspberry Pi 2 Platform

Unlike the computational timing results in Figure 3 and Table 1, which have been obtained on a relatively powerful computer, next we present detailed timing results for an ARM Cortex-A7 processor in the Raspberry Pi 2. While they are not embedded processors by themselves, such Raspberry Pis use ARM cores with computational capabilities on the order of high-end microprocessors that can be used for embedded control applications in several industries (Di Cairano and Kolmanovsky, 2018). Table 2 illustrates detailed timing results of the QP solvers for different linearized MPC simulations on the ARM Cortex-A7 processor, corresponding to different initial conditions θ_0 for the pendulum angle. It can be observed that our `ASIPM` solver has the best average performance and it additionally outperforms most state of the art solvers in worst-case performance.

[1] Computation times were obtained on a powerful computer that is equipped with an Intel(R) Xeon(R) CPU E3-1505M v5 @ 2.80GHz.

Table 2. Average and worst-case computation times (ms) for MPC of an inverted pendulum ($T_s = 50$ ms and $N = 50$), with varying initial conditions for the pendulum angle, on an ARM Cortex-A7 processor in the Raspberry Pi 2 computing platform. [2]

	$\theta_0 = 0.16$		$\theta_0 = 0.20$		$\theta_0 = 0.24$	
	time [ms] (mean/max)	# iter (mean/max)	time [ms] (mean/max)	# iter (mean/max)	time [ms] (mean/max)	# iter (mean/max)
qpOASES	8.90/37.72	0.2/5	19.30/57.51	2.8/10	59.40/162.66	13.3/40
PRESAS	3.19/31.37	1.3/20	5.01/47.47	2.6/30	15.79/122.50	8.4/77
qpDUNES	3.78/19.65	2.1/5	5.50/27.27	2.5/7	6.91/27.29	2.8/8
HPIPM	22.35/27.04	10.2/12	24.10/31.45	10.3/12	23.84/29.57	11.3/14
ASIPM	2.81/21.79	1.2/12	4.41/25.17	1.9/12	5.29/27.77	2.1/13

8. CONCLUSIONS

We presented a structure-exploiting implementation of an IPM that is tailored to MPC. Based on an active-set identification strategy, we proposed an inexact Newton-type algorithm that allows block-based updates to a reverse Cholesky factorization and tailored warm-started solver initialization strategies. We illustrated the computational performance of the QP solver against a range of state of the art software packages that are tailored to MPC for a self-contained C code implementation of our IPM. In addition, we showed that the solver is suitable for embedded computational hardware with relatively limited resources and using, e.g., single-precision arithmetics.

REFERENCES

Bartlett, R., Wächter, A., and Biegler, L. (2000). Active Set vs. Interior Point Strategies for Model Predictive Control. In *Proc. ACC*, 4229–4233. Chicago, Il.

Di Cairano, S. and Kolmanovsky, I.V. (2018). Real-time optimization and model predictive control for aerospace and automotive applications. In *Proc. American Control Conf.*, 2392–2409.

Domahidi, A. and Perez, J. (2014). FORCES professional. embotech GmbH (`http://embotech.com`).

Ferreau, H.J., Almer, S., Verschueren, R., Diehl, M., Frick, D., Domahidi, A., Jerez, J.L., Stathopoulos, G., and Jones, C. (2017). Embedded optimization methods for industrial automatic control. In *IFAC World Congr.*

Ferreau, H.J., Kirches, C., Potschka, A., Bock, H.G., and Diehl, M. (2014). qpOASES: a parametric active-set algorithm for quadratic programming. *Math. Progr. Comp.*, 6(4), 327–363.

Fletcher, R. (1987). *Practical Methods of Optimization*. Wiley, Chichester, 2nd edition.

Frasch, J.V., Sager, S., and Diehl, M. (2015). A parallel quadratic programming method for dynamic optimization problems. *Math. Progr. Comp.*, 7(3), 289–329.

Frison, G. and Diehl, M. (2020). HPIPM: a high-performance quadratic programming framework for model predictive control. *arXiv:2003.02547*.

Frison, G. and Jørgensen, J.B. (2013). Efficient implementation of the Riccati recursion for solving linear-quadratic control problems. In *2013 IEEE Inter. Conf. Control Appl. (CCA)*, 1117–1122.

Frison, G., Kouzoupis, D., Zanelli, A., and Diehl, M. (2017). BLASFEO: Basic linear algebra subroutines for embedded optimization. *arXiv:1704.02457*.

Frison, G., Sorensen, H.B., Dammann, B., and Jørgensen, J.B. (2014). High-performance small-scale solvers for linear model predictive control. In *Proc. European Control Conf. (ECC)*, 128–133.

Gros, S., Zanon, M., Quirynen, R., Bemporad, A., and Diehl, M. (2016). From linear to nonlinear MPC: bridging the gap via the real-time iteration. *International Journal of Control*.

Malyshev, A., Quirynen, R., Knyazev, A., and Cairano, S.D. (2018). A regularized Newton solver for linear model predictive control. In *2018 European Control Conference (ECC)*, 1393–1398.

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

Quirynen, R., Vukov, M., Zanon, M., and Diehl, M. (2014). Autogenerating microsecond solvers for nonlinear MPC: a tutorial using ACADO integrators. *Optimal Control Applications and Methods*, 36, 685–704.

Quirynen, R. and Cairano, S.D. (2019). PRESAS: Block-structured preconditioning of iterative solvers within a primal active-set method for fast MPC. *arXiv preprint arXiv:1912.02122*.

Raghunathan, A.U. and Di Cairano, S. (2015). ADMM for convex quadratic programs: Q-linear convergence and infeasibility detection. *arXiv:1411.7288*.

Rawlings, J.B., Mayne, D.Q., and Diehl, M.M. (2017). *Model Predictive Control: Theory, Computation, and Design*. Nob Hill, 2nd edition edition.

Shahzad, A., Kerrigan, E.C., and Constantinides, G.A. (2010a). A fast well-conditioned interior point method for predictive control. In *49th IEEE Conference on Decision and Control (CDC)*, 508–513.

Shahzad, A., Kerrigan, E., and Constantinides, G. (2010b). A warm-start interior-point method for predictive control. Technical report, Imperial College London.

Stellato, B., Banjac, G., Goulart, P., Bemporad, A., and Boyd, S. (2017). OSQP: An operator splitting solver for quadratic programs. *ArXiv e-prints*.

Wright, S. (1996). Applying new optimization algorithms to model predictive control. In J. Kantor, C. Garcia, and B. Carnahan (eds.), *Fifth Intern. Conf. on Chemical Process Control – CPC V*, 147–155.

Zanelli, A., Quirynen, R., Jerez, J., and Diehl, M. (2017). A homotopy-based nonlinear interior-point method for NMPC. In *Proc. IFAC World Congr.* Toulouse, France.

[2] The Raspberry Pi 2 uses a BCM2836 SoC with a 900 MHz 32-bit quad-core ARM Cortex-A7 processor, with 256 KB shared L2 cache.