

An Execution-time-certified QP Algorithm for ℓ_1 penalty-based Soft-constrained MPC [★]

Liang Wu ^{a,1}, Richard D. Braatz ^a

^aMassachusetts Institute of Technology, Cambridge, MA 02139, USA

Abstract

Providing an execution time certificate and handling possible infeasibility in closed-loop are two pressing requirements of Model Predictive Control (MPC). To simultaneously meet these two requirements, this paper uses an ℓ_1 penalty-based soft-constrained MPC formulation and innovatively transforms the resulting non-smooth QP into a box-constrained QP, which is solved by our previously proposed *direct* and execution-time certified algorithm with only *dimension-dependent* (data-independent), *simple-calculated* and *exact* number of iterations (Wu and Braatz (2023)). This approach not only overcomes the limitation of our previously proposed algorithm (Wu and Braatz (2023)), only applicable to input-constrained MPC, but also enjoys *exact recovery* feature (exactly recover the same solution when the original problem is feasible) of ℓ_1 penalty-based soft-constrained MPC formulation without suffering numerical difficulty of the resulting non-smoothness. Other various real-time QP applications, not limited to MPC, would also benefit from our QP algorithm with execution-time certificate and global feasibility.

Key words: Iteration complexity; Execution time certificate; Model predictive control; Infeasibility; Soft constraints;

1 Introduction

Model Predictive Control (MPC) has proven successful in numerous industrial applications, owing to its capability to handle inputs and state constraints within the control formulation. After the MPC design phase, MPC is then deployed into production embedded processors such as Programmable logic controllers (PLC) and Microcontrollers. Generally, deploying a real-time MPC requires an online Quadratic Programming (QP) solver for computing the control signals, except for simple problems that can compute an offline explicit form of MPC control law (Bemporad *et al.* (2002)).

Deploying a real-time MPC on embedded processors could lead to either: 1) control inputs failing to return on time, or 2) control inputs being empty upon return. Clearly, both scenarios will render the process open-loop, thus resulting in safety concerns, particularly in safety-critical process systems. The former arises due to the lack of an execution time certificate, meaning that the worst-case (maximum) execution time of the real-time

MPC must be theoretically guaranteed to be smaller than the sampling time in closed-loop. If the execution time of the real-time MPC exceeds the sampling time, control inputs fail to return on time. The latter arises from the possible infeasibility of MPC problems, wherein unknown disturbances or modeling errors may drive the plant state into a region where real-time MPC problems become infeasible. In such instances, MPC returns an empty solution.

Moreover, the execution time certificate of real-time MPC problems also serves as a valuable guide during the MPC design phase. For example, our previous execution-time-certified algorithm has *only dimension-dependent* (data-independent) and *simple-calculated* time complexity, which can help trade-off between prediction horizon (leading to problems with different dimensions) and sampling time (ensuring compliance with the execution-time certificate requirement), instead of performing extensive simulations or heavy calibration work (Forgione *et al.* (2020)) to satisfy the requirement of execution time certificates.

To avoid possible MPC controller failure from infeasibility issue, a common approach is to use soft-constraint MPC schemes which introduce slack variables to soften the constraints and add a penalty term of correspond-

[★] This paper was not presented at any IFAC conference.

Email addresses: liangwu@mit.edu (Liang Wu), braatz@mit.edu (Richard D. Braatz).

¹ Corresponding author.

ing slack variables into the MPC cost function. One desired feature, for a soft-constrained MPC scheme, is to exactly recover the same solution when the original hard-constrained MPC problem is feasible. To achieve this *exact recovery*, the ℓ_2 quadratic penalty method requires that the penalty parameter has to be infinite. The *exact penalty function* method, such as the ℓ_1 -penalty method (Kerrigan and Maciejowski (2000)), avoids this issue and only requires the penalty parameter to be finite and larger than the dual norm of the corresponding optimal Lagrange multiplier. This exact recovery power of the *exact penalty function* method stems from the introduction of non-smoothness to the associated QP (Kerrigan and Maciejowski (2000)). However, this introduces numerical challenges, not to mention the execution time certificate for the resulting non-smooth QP.

1.1 Related work

This paper aims to address the above two issues simultaneously, that is, providing the execution-time certificate and handling possible infeasibility.

Execution time generally includes average and worst-case (maximum) execution time as MPC involves a sequence of QP problems with different problem data. Most studies endeavors focus on the development of algorithms that possess a faster average execution time, see Wang and Boyd (2009); Ferreau *et al.* (2014); Stellato *et al.* (2020); Wu and Bemporad (2023b,a). However, the worst-case execution time matters a lot more than the average execution time when deploying MPC in embedded systems, as the sampling time is chosen to be larger than the worst-case execution time instead of the average execution time. The worst-case execution time should be obtained from theory rather than from extensive closed-loop simulations. This theoretical certificate of execution time has garnered considerable interest in recent years and remains an ongoing research area (Richter *et al.* (2011); Bemporad and Patrinos (2012); Giselsson (2012); Cimini and Bemporad (2017, 2019); Arnström and Axehill (2019); Arnström *et al.* (2020); Arnström and Axehill (2021); Okawa and Nonaka (2021); Wu and Braatz (2023); Wu *et al.* (2024a,b)).

All these works assume that *the adopted computation platform performs a fixed number of floating-point operations ([flops]) in constant time units*, that is,

$$\text{execution time} = \frac{\text{total [flops] required by algorithm}}{\text{[flops] processed per second}} [\text{s}],$$

where one flop is defined to be one multiplication, subtraction, addition, or division of two floating-point numbers. Calculating the total [flops] is equal to analyzing the number of iterations if a single iteration performs

fixed [flops]. Therefore, all these works analyze the worst-case iteration complexity of their proposed algorithms.

In Giselsson (2012); Richter *et al.* (2011); Bemporad and Patrinos (2012), first-order methods were used to solve the dual problem of linear MPC and their worst-case iteration complexity is not only conservative but also dependent on the data of the optimization problem. This *data-dependent* iteration complexity is an issue for nonlinear MPC (via online linearized MPC such as the Real-Time Iteration (RTI) scheme (Gros *et al.* (2020))). In such cases, the MPC problem data changes at each sampling time, making it impossible to derive the maximum execution time of all MPC problems in closed-loop.

In Cimini and Bemporad (2017, 2019); Arnström and Axehill (2019); Arnström *et al.* (2020); Arnström and Axehill (2021), active-set based methods were used and their worst-case iteration complexity certification relies on the *computationally complicated and expensive* off-line worst-case partial enumeration technique (called WCPE-MPC). This cost limits their use in nonlinear MPC (via online linearized MPC) problems. A similar situation occurs for the N-step algorithm of Okawa and Nonaka (2021), which has the worst-case iteration complexity in the problem dimension N . The N-step algorithm requires a *modified N-step vector* which is found by solving a linear program. Hence, the N-step algorithm is also not suitable for online linearized MPC problems.

To summarize, we pursue a QP algorithm with *data-independent* (only *dimension-dependent*) and *simple-calculated* iteration complexity (does not require any computationally complicated and expensive techniques for calculation.)

Our previous work (Wu and Braatz (2023)) for the first time proposed a *direct* box-constrained QP (Box-QP) algorithm with exact iteration complexity,

$$\mathcal{N} = \left\lceil \frac{\log(\frac{2n}{\epsilon})}{-2 \log(\frac{\sqrt{2n}}{\sqrt{2n} + \sqrt{2} - 1})} \right\rceil + 1,$$

which only depends on problem dimension n , where ϵ is a constant specifying the stopping criterion (e.g., 1×10^{-6}). We denote this algorithm as being *direct*, a concept borrowed from direct methods of solving linear equations (such as the QR decomposition and Cholesky decomposition) with exact and only *dimension-dependent* [flops]. We then extend this approach to develop execution-time certified nonlinear input-constrained MPC algorithms via data-driven Koopman operator (Wu *et al.* (2024a)) and RTI scheme (Wu *et al.* (2024b)) (linear time-varying model MPC resulting to time-changing problem data).

However, our *direct* Box-QP algorithm is only limited to

input-constrained MPC problems, and not applicable to general MPC problems with input and state constraints. Meantime, general MPC problems may suffer possible infeasibility in closed-loop operation.

1.2 Contribution

This article for the first time simultaneously and elegantly addresses the above two issues: 1) provides an execution-time certificate for general MPC problems subject to input and state constraints; 2) using the non-smooth ℓ_1 penalty method to soften general MPC problems with input and state constraints, aiming to handle possible infeasibility. We first prove that the non-smooth ℓ_1 penalty-based soft-constrained MPC can be equivalently transformed into Box-QP. Therefore, our previous *direct* Box-QP algorithm with *only dimension-dependent* (data-independent), *simple-calculated* and *exact* iteration complexity, is applied. This procedure results in an execution-time certified QP algorithm for soft-constrained MPC.

The purpose of this note is not to discuss how to ensure closed-loop stability of ℓ_1 soft-constrained MPC; instead, our focus is on developing an execution-time-certified QP algorithm with the capability of handling infeasibility, for emerging various real-time QP applications, not limited to MPC.

1.3 Notations

The vector of all ones is denoted by $e = (1, \dots, 1)^\top$. $\lceil x \rceil$ maps x to the least integer greater than or equal to x . For $z, y \in \mathbb{R}^n$, let $\text{col}(z, y) = [z^\top, y^\top]^\top$ and $\max(z, y) = (\max(z_1, y_1), \max(z_2, y_2), \dots, \max(z_n, y_n))^\top$. $\mathbb{R}^n$ denotes the space of n -dimensional real vectors, $\mathbb{R}_{++}^n$ is the set of all positive vectors of $\mathbb{R}^n$. For a vector $z \in \mathbb{R}^n$, its Euclidean norm is $\|z\| = \sqrt{z_1^2 + z_2^2 + \dots + z_n^2}$, $\|z\|_1 = \sum_{i=1}^n |z_i|$, $\|z\|_\infty = \max_i |z_i|$, $\text{diag}(z) : \mathbb{R}^n \rightarrow \mathbb{R}^{n \times n}$ maps an vector z to its corresponding diagonal matrix, and $z^2 = (z_1^2, z_2^2, \dots, z_n^2)^\top$. Given two arbitrary vectors $z, y \in \mathbb{R}_{++}^n$, their Hadamard product is $zy = (z_1 y_1, z_2 y_2, \dots, z_n y_n)^\top$, $\left(\frac{z}{y}\right) = \left(\frac{z_1}{y_1}, \frac{z_2}{y_2}, \dots, \frac{z_n}{y_n}\right)^\top$, $\sqrt{z} = (\sqrt{z_1}, \sqrt{z_2}, \dots, \sqrt{z_n})^\top$ and $z^{-1} = (z_1^{-1}, z_2^{-1}, \dots, z_n^{-1})^\top$.

2 Problem Formulation

A general linear (or online linearized) MPC problem with input and state constraints is commonly formulated using the condensing construction approach (Jerez *et al.* (2011)), as the general QP

$$\begin{aligned} \min_{y \in \mathbb{R}^m} \quad & \frac{1}{2} y^\top Q(t) y + y^\top (F(t)x(t)) \\ \text{s.t.} \quad & G(t)y \leq g(t) + S(t)x(t) \end{aligned} \quad (1)$$

where y denotes the control inputs; the symmetric positive definite matrix $Q(t) \in \mathbb{R}^{m \times m}$, $F(t) \in \mathbb{R}^{m \times n_x}$, the full row rank matrix $G(t) \in \mathbb{R}^{n \times m}$, $g(t) \in \mathbb{R}^n$, $S(t) \in \mathbb{R}^{n \times n_x}$ are given; $x(t) \in \mathbb{R}^{n_x}$ denotes the feedback states (or past input-output data if using an input-output model and reference signals) at the sampling time t . Note that the optimization problem data $\{Q(t), F(t), G(t), g(t), S(t), x(t)\}$ may all vary over time t as in the case where the RTI scheme (Gros *et al.* (2020)) is used for nonlinear MPC problems.

As we discussed before, the QP (1) may be infeasible when a disturbance or modeling (linearization approximation) errors drive the system state $x(t)$ into a region that the set $G(t)y \leq g(t) + S(t)x(t)$ of the QP (1) is empty. To handle the possible infeasibility, this paper adopts the ℓ_1 penalty approach to soften the QP (1) to make it always feasible. The ℓ_1 -penalty method reformulates (1) as the unconstrained non-smooth penalty minimization,

$$\begin{aligned} \min_y \quad & \frac{1}{2} y^\top Q(t) y + y^\top (F(t)x(t)) \\ & + \rho \| \max(0, G(t)y - g(t) - S(t)x(t)) \|_1 \end{aligned} \quad (2)$$

where $\rho \in \mathbb{R}_{++}$ denotes the penalty parameter. Compared to the most frequently used ℓ_2 quadratic penalty method, the ℓ_1 penalty method is better because it does not require the penalty parameter ρ to be very large, and it has the *exact recovery* feature if the penalty parameter ρ is large enough according to the following lemma,

Lemma 1 (See (Fletcher, 2000, Thm. 14.3.1)) Suppose the original QP (1) has an optimal solution y^* (we have $G(t)y^* \leq g(t) + S(t)x(t)$) and the penalty parameter $\rho > \|\zeta^*\|_\infty$, then the solution of (2) is equal to the solution of (1), where ζ^* denotes the corresponding optimal Lagrange multiplier.

The slight difference from (Fletcher, 2000, Thm. 14.3.1) is that we adopt the ℓ_1 norm and the dual of $\|\cdot\|_1$ is $\|\cdot\|_\infty$. In Kerrigan and Maciejowski (2000); Fletcher (2000), they also mentioned that the ℓ_∞ penalty (that is, $\rho \| \max(0, G(t)y - g(t) - S(t)x(t)) \|_\infty$) can be used, but only the ℓ_1 penalty can allow us to transform the non-smooth QP (2) to an equivalent Box-QP. Although Lemma 1 tells how to choose the penalty parameter ρ in theory, it is impractical as the optimal Lagrange multiplier ζ^* is unknown. In Kerrigan and Maciejowski (2000), they presented a method to calculate a lower bound of the penalty parameter ρ via solving a sequence of linear programs, which is complicated and requires choosing a bounded polytope for $x(t)$. A practical way is to regard the penalty parameter ρ as the MPC cost weights to tune in particular for online linearized MPC scenarios.

MPC distinguishes each constraint as being either hard or soft. The hard constraints are from the physical lim-

its of actuators (the control inputs), which should never be violated. The soft constraints are user-specified state constraints, which are allowed to be violated by a small amount if there is no feasible solution if the constraints are hard. The above penalty formulation (2) could generate a solution that does not violate the soft (state) constraints but violates the hard (control input) constraints.

To avoid this physically invalid solution, this paper innovatively uses the penalty parameter vector instead of the penalty parameter scalar as

$$\min_y \frac{1}{2} y^\top Q(t)y + y^\top (F(t)x(t)) + \text{diag}(\rho) \|\max(0, G(t)y - g(t) - S(t)x(t))\|_1 \quad (3)$$

where $\rho \in \mathbb{R}_{++}^n$ denotes the penalty parameter vector. As the ℓ_1 norm can lead to sparseness to the solution, we can choose larger penalty parameters for the hard constraints to ensure that the hard constraints will not be violated before the soft constraints.

Next we introduce slack variables $w = \max(0, G(t)y - g(t) - S(t)x(t))$, thus $w \geq 0$ and $\text{diag}(\rho) \|\max(0, G(t)y - g(t) - S(t)x(t))\|_1 = \rho^\top w$. Therefore, we can reformulate the nonsmooth (3) into the smooth QP,

$$\begin{aligned} \min_{y, w} \quad & \frac{1}{2} y^\top Q(t)y + y^\top (F(t)x(t)) + \rho^\top w \\ \text{s.t.} \quad & w \geq 0 \\ & w \geq G(t)y - g(t) - S(t)x(t) \end{aligned} \quad (4)$$

and its Lagrangian function $\mathcal{L}(y, w, \tilde{z}, \hat{z})$ is given by

$$\begin{aligned} \mathcal{L}(y, w, \tilde{z}, \hat{z}) = & \frac{1}{2} y^\top Q(t)y + y^\top (F(t)x(t)) + \rho^\top w \\ & - \tilde{z}^\top w - \hat{z}^\top (w - G(t)y + g(t) + S(t)x(t)) \end{aligned}$$

where $\tilde{z}, \hat{z} \in \mathbb{R}^n$ are the vector of Lagrange multipliers associated with the inequalities in (4).

Then, its Karush–Kuhn–Tucker (KKT) conditions are

$$\begin{aligned} Q(t)y + F(t)x(t) + G(t)^\top \hat{z} &= 0 \\ \rho - \tilde{z} - \hat{z} &= 0 \\ w \geq 0, \tilde{z} \geq 0, w_i \tilde{z}_i &= 0, \forall i = 1, 2, \dots, n \\ \delta \geq 0, \hat{z} \geq 0, \delta_i \hat{z}_i &= 0, \forall i = 1, 2, \dots, n \end{aligned} \quad (5)$$

where $\delta = w - G(t)y + g(t) + S(t)x(t)$ are the introduced slack variable. As $Q(t)$ is symmetric positive definite, the decision variable y can be represented by $\hat{z}$, that is,

$$y = -Q(t)^{-1}(F(t)x(t) + G(t)^\top \hat{z}). \quad (6)$$

Then the KKT condition (5) is transformed into

$$\begin{aligned} M\hat{z} + r - \delta + w &= 0 \\ \delta \geq 0, \hat{z} \geq 0, \delta_i \hat{z}_i &= 0, \forall i = 1, 2, \dots, n \\ w \geq 0, \rho - \hat{z} \geq 0, w_i(\rho_i - \hat{z}_i) &= 0, \forall i = 1, 2, \dots, n \end{aligned} \quad (7)$$

where $M = G(t)Q(t)^{-1}G(t)^\top$, $r = G(t)Q(t)^{-1}F(t)x(t) + g(t) + S(t)x(t)$.

Clearly, the reformulated KKT condition (7) is the KKT condition of the box-constrained QP (Box-QP),

$$\begin{aligned} \min_{\hat{z}} \quad & \frac{1}{2} \hat{z}^\top M\hat{z} + \hat{z}^\top r \\ \text{s.t.} \quad & 0 \leq \hat{z} \leq \rho \end{aligned} \quad (8)$$

Summarizing the above analysis and based on the same KKT condition of problem (3) and (8), we obtain the following Theorem,

Theorem 1 *Suppose that Q is symmetric positive definite, the non-smooth penalty minimization Problem 3 is equivalent to the Box-QP (8) and their optimal solution relationship is described by Eqn. (6).*

Then, we scale the Box-QP (8) to the box constraint $[-e, e]$. Applying the coordinate transformation

$$z = \text{diag}\left(\frac{2}{\rho}\right)\hat{z} - e$$

results in the equivalent Box-QP

$$\begin{aligned} \min_z \quad & \frac{1}{2} z^\top H z + z^\top h \\ \text{s.t.} \quad & -e \leq z \leq e \end{aligned} \quad \begin{aligned} (9a) \\ (9b) \end{aligned}$$

where $H = \text{diag}(\rho)M\text{diag}(\rho)$ and $h = \text{diag}(\rho)M(\rho + 2r) = H(e + 2\frac{r}{\rho})$.

Corollary 1 *The optimal solution y^* of the non-smooth penalty minimization Problem 3 can be recovered by the optimal solution z^* of the scaled Box-QP (9) with*

$$y^* = -Q(t)^{-1}\left(F(t)x(t) + \frac{1}{2}G(t)^\top(\rho z^* + \rho)\right). \quad (10)$$

3 Execution-time Certified QP Algorithm

Based on the path-following full-Newton IPM, our previous work (Wu and Braatz (2023)) proposed a *direct* optimization algorithm to solve the Box-QP (9). Its KKT

conditions are

$$Hz + h + \gamma - \theta = 0 \quad (11a)$$

$$z + \alpha - e = 0 \quad (11b)$$

$$z - \omega + e = 0 \quad (11c)$$

$$\gamma\phi = 0 \quad (11d)$$

$$\theta\psi = 0 \quad (11e)$$

$$(\gamma, \theta, \alpha, \omega) \geq 0 \quad (11f)$$

A positive parameter τ was introduced by the path-following IPM to replace (11d) and (11e) by

$$\gamma\phi = \tau^2 e, \quad (12a)$$

$$\theta\psi = \tau^2 e. \quad (12b)$$

As τ tends to 0, the path $(z_\tau, \gamma_\tau, \theta_\tau, \phi_\tau, \psi_\tau)$ converges to a solution of (11). The feasible variants of the path-following IPM algorithm boast the best theoretical $O(\sqrt{n})$ iteration complexity (Wright (1997)). Therefore, our algorithm is based on feasible IPM in which all iterates are required to be strictly in the feasible set

$$\mathcal{F}^0 := \{(z, \gamma, \theta, \phi, \psi) \mid (11a)-(11c) \text{ satisfied}, (\gamma, \theta, \phi, \psi) > 0\}.$$

3.1 Strictly feasible initial point

In our earlier work (Wu and Braatz (2023)), a cost-free initialization strategy was innovatively proposed to find a strictly feasible initial point that also satisfies the specific conditions. A strictly feasible initial point is

$$\begin{aligned} z^0 &= 0, \\ \gamma^0 &= \|h\|_\infty - \frac{1}{2}h, \\ \theta^0 &= \|h\|_\infty + \frac{1}{2}h, \\ \phi^0 &= e, \\ \psi^0 &= e, \end{aligned}$$

where $\|h\|_\infty = \max\{|h_1|, |h_2|, \dots, |h_n|\}$. It is straightforward to see that the above initial point strictly lies in $\mathcal{F}^0$.

Remark 1 (Initialization strategy) *If $h = 0$, the optimal solution of Problem 9 is $z^* = 0$; in the case of $h \neq 0$, first scale the objective (9a) (which does not change the optimal solution) as*

$$\min_z \frac{1}{2} z^\top \left(\frac{2\lambda}{\|h\|_\infty} H \right) z + z^\top \left(\frac{2\lambda}{\|h\|_\infty} h \right).$$

With the definitions $\tilde{H} = \frac{1}{\|h\|_\infty} H$ and $\tilde{h} = \frac{1}{\|h\|_\infty} h$, $\|\tilde{h}\|_\infty = 1$ and (11a) can be replaced by

$$2\lambda\tilde{H}z + 2\lambda\tilde{h} + \gamma - \theta = 0,$$

and the initial points

$$\begin{aligned} z^0 &= 0, \\ \gamma^0 &= 1 - \lambda\tilde{h}, \\ \theta^0 &= 1 + \lambda\tilde{h}, \\ \phi^0 &= e, \\ \psi^0 &= e, \end{aligned} \quad (13)$$

can be adopted, where

$$\lambda = \frac{1}{\sqrt{n+1}}.$$

It is straightforward to verify that (13) lies in $\mathcal{F}^0$. The reason to use the scale factor $\frac{2\lambda}{\|h\|_\infty}$ is to make the initial point satisfy the neighborhood requirements, e.g., see (Wu and Braatz, 2023, Lemma 4).

Notably, one main contribution of our previous work (Wu and Braatz (2023)) is to *remove the assumption* of previous works that the initial point is strictly feasible and located in a narrow neighborhood of the central path. It is the first time proposing a *cost-free* and *data-independent* initialization strategy.

3.2 Newton direction

Denote $v := \text{col}(\gamma, \theta) \in \mathbb{R}^{2n}$ and $s := \text{col}(\phi, \psi) \in \mathbb{R}^{2n}$. Then replace (12a) and (12b) by $vs = \tau^2 e$ to obtain the new complementary condition

$$\sqrt{vs} = \sqrt{\tau^2 e}. \quad (14)$$

From *Remark 1*, $(z, v, s) \in \mathcal{F}^0$ and a direction $(\Delta z, \Delta v, \Delta s)$ can be obtained by solving the system of linear equations

$$2\lambda\tilde{H}\Delta z + \Omega\Delta v = 0, \quad (15a)$$

$$\Omega^\top \Delta z + \Delta s = 0, \quad (15b)$$

$$\sqrt{\frac{s}{v}}\Delta v + \sqrt{\frac{v}{s}}\Delta s = 2(\tau e - \sqrt{vs}), \quad (15c)$$

where $\Omega = [I, -I] \in \mathbb{R}^{n \times 2n}$. Letting

$$\Delta\gamma = \frac{\gamma}{\phi}\Delta z + 2\left(\sqrt{\frac{\gamma}{\phi}}\tau e - \gamma\right), \quad (16a)$$

$$\Delta\theta = -\frac{\theta}{\psi}\Delta z + 2\left(\sqrt{\frac{\theta}{\psi}}\tau e - \theta\right), \quad (16b)$$

$$\Delta\phi = -\Delta z, \quad (16c)$$

$$\Delta\psi = \Delta z, \quad (16d)$$

reduces (15) into a more compact system of linear equations,

$$\begin{aligned} & \left(2\lambda\tilde{H} + \text{diag}\left(\frac{\gamma}{\phi}\right) + \text{diag}\left(\frac{\theta}{\psi}\right)\right)\Delta z \\ &= 2\left(\sqrt{\frac{\theta}{\psi}}\tau e - \sqrt{\frac{\gamma}{\phi}}\tau e + \gamma - \theta\right). \end{aligned} \quad (17)$$

3.3 Iteration complexity and algorithm implementation

Denote $\beta := \sqrt{vs}$ and define the proximity measure

$$\xi(\beta, \tau) = \frac{\|\tau e - \beta\|}{\tau}. \quad (18)$$

Lemma 2 (See Wu and Braatz (2023)) *Let $\xi := \xi(\beta, \tau) < 1$. Then the full Newton step is strictly feasible, i.e., $v_+ > 0$ and $s_+ > 0$.*

Lemma 3 (See Wu and Braatz (2023)) *After a full Newton step, let $v_+ = v + \Delta v$ and $s_+ = s + \Delta s$, then the duality gap is*

$$v_+^\top s_+ \leq (2n)\tau^2.$$

Lemma 4 (See Wu and Braatz (2023)) *Suppose that $\xi = \xi(\beta, \tau) < 1$ and $\tau_+ = (1 - \eta)\tau$ where $0 < \eta < 1$. Then*

$$\xi_+ = \xi(\beta_+, \tau_+) \leq \frac{\xi^2}{1 + \sqrt{1 - \xi^2}} + \frac{\eta\sqrt{2n}}{1 - \eta}.$$

Furthermore, if $\xi \leq \frac{1}{\sqrt{2}}$ and $\eta = \frac{\sqrt{2}-1}{\sqrt{2n}+\sqrt{2}-1}$, then $\xi_+ \leq \frac{1}{\sqrt{2}}$.

Lemma 5 (See Wu and Braatz (2023)) *The value of $\xi(\beta, \tau)$ before the first iteration is denoted as $\xi^0 = \xi(\beta^0, (1 - \eta)\tau^0)$. If $(1 - \eta)\tau^0 = 1$ and $\lambda = \frac{1}{\sqrt{n+1}}$, then $\xi^0 \leq \frac{1}{\sqrt{2}}$ and $\xi(\beta, w) \leq \frac{1}{\sqrt{2}}$ are always satisfied.*

Lemma 6 (See Wu and Braatz (2023)) *Let $\eta = \frac{\sqrt{2}-1}{\sqrt{2n}+\sqrt{2}-1}$ and $\tau^0 = \frac{1}{1-\eta}$, Algorithm 1 exactly requires*

$$\mathcal{N} = \left\lceil \frac{\log(\frac{2n}{\epsilon})}{-2\log(\frac{\sqrt{2n}}{\sqrt{2n}+\sqrt{2}-1})} \right\rceil + 1 \quad (19)$$

iterations, the resulting vectors being $v^\top s \leq \epsilon$.

Theorem 2 *In Algorithm 1, Step 1 requires $\frac{1}{3}m^3 + \frac{1}{2}m^2 + \frac{1}{6}m$ [flops], Step 2 requires $nm^2 + nm + 2mn^2$ [flops] for computing H and $2m^2 + 2mn + 4n + 2n^2$ [flops] for computing h , Step 3 requires n [flops], Step*

4 requires $5n + 3$ [flops], Step 5 requires $\mathcal{N}(1 + \frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n + 2n^2 + 10n + 5n)$ [flops], Step 6 requires $2n + 2mn + 2m + 2m^2$ [flops].

For linear time-invariant MPC problems, Step 1 and the H computation of Step 2 can be offline cached, and thus excluded from the online computation cost.

Algorithm 1 An execution-time certified algorithm for QP (1)

Input: the problem data $\{Q(t), F(t)x(t), G(t), g(t), S(t)x(t)\}$ at current sampling time t , the penalty parameter vector $\rho \in \mathbb{R}_{++}^n$, and the stopping tolerance ϵ (e.g., 1×10^{-6}). Then, the required exact number of iterations is

$$\mathcal{N} = \left\lceil \frac{\log(\frac{2n}{\epsilon})}{-2\log(\frac{\sqrt{2n}}{\sqrt{2n}+\sqrt{2}-1})} \right\rceil + 1.$$

1. Cache $Q(t)^{-1}$ via Cholesky decomposition;
2. $H \leftarrow \text{diag}(\rho)G(t)Q(t)^{-1}G(t)^\top \text{diag}(\rho)$, $h \leftarrow H(e + \frac{2}{\rho}G(t)Q(t)^{-1}F(t)x(t) + g(t) + S(t)x(t))$;
3. **if** $\|h\|_\infty = 0$, $z \leftarrow 0$, **goto** Step 6, **otherwise**,
4. $(z, \gamma, \theta, \phi, \psi)$ are initialized from (13) where $\lambda \leftarrow \frac{1}{\sqrt{n+1}}$, $\eta \leftarrow \frac{\sqrt{2}-1}{\sqrt{2n}+\sqrt{2}-1}$ and $\tau \leftarrow \frac{1}{1-\eta}$;
5. **for** $k = 1, 2, \dots, \mathcal{N}$ **do**
 - 5.1. $\tau \leftarrow (1 - \eta)\tau$;
 - 5.2. solve (17) for Δz by using the Cholesky decomposition method with one forward substitution and one backward substitution;
 - 5.3. calculate $(\Delta\gamma, \Delta\theta, \Delta\alpha, \Delta\omega)$ from (16);
 - 5.4. $z \leftarrow z + \Delta z$, $\gamma \leftarrow \gamma + \Delta\gamma$, $\theta \leftarrow \theta + \Delta\theta$, $\alpha \leftarrow \alpha + \Delta\alpha$, $\omega \leftarrow \omega + \Delta\omega$;
- end**
6. **return**

$$y^* = -Q(t)^{-1} \left(F(t)x(t) + \frac{1}{2}G(t)^\top(\rho z + \rho) \right)$$

4 Numerical example

To illustrate a problem with potential infeasibility, consider a double integrator example $\ddot{x} = 10000u$. Discretizing at 100 Hz (the sampling time $\Delta t = 0.01$ s), gives the discrete-time system,

$$\begin{bmatrix} x_1(t+1) \\ x_2(t+1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)$$

where $x_1(t)$ denotes the position, $x_2(t)$ denotes the velocity, and $u(t)$ denotes the acceleration at time step t . The physical limit of the control input, the acceleration

$u(t)$, is given $[-1, 1]$, and the desired state constraint is $x_1(t) \geq -1$, which could result in infeasibility due to the input constraint $[-1, 1]$. For example, if the current state is $(x_1(t), x_2(t)) = (0, -2)$, the resulting MPC problem is infeasible because $x_1(t)$ will violate the state constraint $x_1(t) \geq -1$ no matter what the control input $u(t) \in [-1, 1]$ is.

The soft-constrained MPC is used to avoid the possible infeasibility. The control input constraints are from physical limits and thus have priority over the state constraints. To achieve this priority, different weights are chosen for the input and state constraints. Here we choose two sets of the penalty parameter vector $\rho = (\rho_{\text{hard}}e, \rho_{\text{soft}}e)$:

- 1) $\rho_{\text{hard}} = 10, \rho_{\text{soft}} = 10$,
- 2) $\rho_{\text{hard}} = 100, \rho_{\text{soft}} = 10$.

The prediction horizon is set as 10, the cost matrices for the states and the terminal state are both set as $\text{diag}(1, 1)$, and the weight matrix for the input is set as 0.1.

Before the closed-loop simulation, we already know the dimension of the scaled Box-QP problem, that is, $n = 30$. By adopting the stopping criterion $\epsilon = 1 \times 10^{-6}$, we know that Algorithm 1 will exactly perform

$$\mathcal{N} = \left\lceil \frac{\log(\frac{2 \times 30}{1 \times 10^{-6}})}{-2 \log(\frac{\sqrt{2 \times 30}}{\sqrt{2 \times 30} + \sqrt{2} - 1})} \right\rceil + 1 = 173.$$

iterations. Further, by Theorem 2 we can exactly calculate the total [flops] required by Algorithm 1. In our settings, $m = 10, n = 30$ and Step 1 and the H computation of Step 2 are offline cached, thus Algorithm 1 takes a total 0.002×10^{-9} [flops] which leads to the execution time 0.002 s on a machine with 1 GFLOP/s computing power (a trivial requirement for most processors today). Thus, we can get a certificate that the execution time will be less than the sampling time $\Delta t = 0.01$ s.

We executed Algorithm 1 in MATLAB2023a via a C-mex interface, and the closed-loop simulation was performed on a contemporary MacBook Pro with 2.7 GHz 4-core Intel Core i7 processors and 16GB RAM. The number of iterations is exactly 173, which agrees with the theoretical value. The practical execution time never exceeds 0.0015 s (by running experiments multiple times), which is less than $\Delta t = 0.01$ s.

The closed-loop simulation results are depicted in Fig. 1, which shows that – in our ℓ_1 penalty-based soft-constrained MPC setting – if we choose the same penalty parameters for hard and soft constraints, the control input violates the physical actuator limits. If a larger penalty parameter is used for hard constraints, the con-

trol input is always bounded in the physical actuator limits $[-1, 1]$.

5 Conclusion

This article extends our previous execution-time-certified Box-QP algorithm (Wu and Braatz (2023)) for input-constrained MPC by using an ℓ_1 penalty to soften general MPC problems with input and state constraints. The use of the ℓ_1 penalty not only allows us to transform soft-constrained MPC problems into Box-QP problems but also enjoys *exact recovery*. Therefore, our approach can simultaneously provide execution time certificates and handle possible infeasibility for general MPC problems with input and state constraints in closed-loop.

Our execution-time-certified QP algorithm with the capability of handling infeasibility can be applied to other various real-time QP applications, such as constrained Differential Dynamic Programming (DDP) (Xie *et al.* (2017)), constrained Control Lyapunov policy (Wang and Boyd (2010)), and Control Barrier Function based QP (Ames *et al.* (2016); Xiao and Belta (2021)).

Acknowledgements

This research was supported by the U.S. Food and Drug Administration under the FDA BAA-22-00123 program, Award Number 75F40122C00200.

References

- Ames, A. D., X. Xu, J. W. Grizzle and P. Tabuada (2016). ‘Control barrier function based quadratic programs for safety critical systems’. *IEEE Transactions on Automatic Control* **62**(8), 3861–3876.
- Arnström, D., A. Bemporad and D. Axehill (2020). ‘Complexity certification of proximal-point methods for numerically stable quadratic programming’. *IEEE Control Systems Letters* **5**(4), 1381–1386.
- Arnström, D. and D. Axehill (2019). Exact complexity certification of a standard primal active-set method for quadratic programming. In ‘58th IEEE Conference on Decision and Control’. pp. 4317–4324.
- Arnström, D. and D. Axehill (2021). ‘A Unifying Complexity Certification Framework for Active-Set Methods for Convex Quadratic Programming’. *IEEE Transactions on Automatic Control* **67**(6), 2758–2770.
- Bemporad, A. and P. Patrinos (2012). ‘Simple and certifiable quadratic programming algorithms for embedded linear model predictive control’. *IFAC Proceedings Volumes* **45**(17), 14–20.
- Bemporad, A., M. Morari, V. Dua and E. N. Pistikopoulos (2002). ‘The explicit linear quadratic regulator for constrained systems’. *Automatica* **38**(1), 3–20.

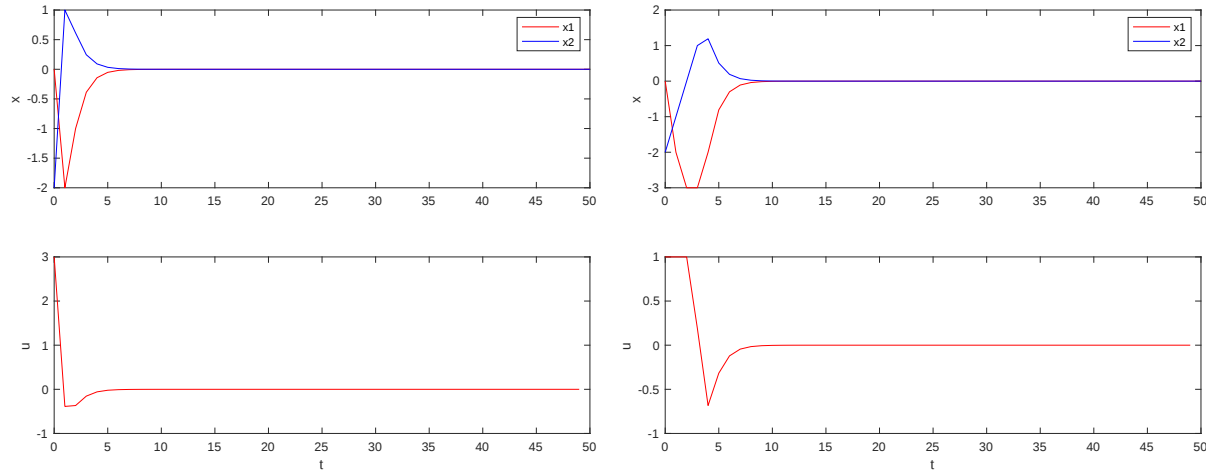


Fig. 1. Closed-loop simulation of the discrete-time double integrator with ℓ_1 penalty-based soft-constrained MPC. Left: 1) $\rho_{\text{hard}} = 10, \rho_{\text{soft}} = 10$. Right: 2) $\rho_{\text{hard}} = 100, \rho_{\text{soft}} = 10$.

- Cimini, G. and A. Bemporad (2017). ‘Exact complexity certification of active-set methods for quadratic programming’. *IEEE Transactions on Automatic Control* **62**(12), 6094–6109.
- Cimini, G. and A. Bemporad (2019). ‘Complexity and convergence certification of a block principal pivoting method for box-constrained quadratic programs’. *Automatica* **100**, 29–37.
- Ferreau, H. J., C. Kirches, A. Potschka, H. G. Bock and M. Diehl (2014). ‘qpOASES: A parametric active-set algorithm for quadratic programming’. *Mathematical Programming Computation* **6**, 327–363.
- Fletcher, R. (2000). *Practical Methods of Optimization*. John Wiley & Sons, Chichester, England.
- Forgione, M., D. Piga and A. Bemporad (2020). ‘Efficient calibration of embedded MPC’. *IFAC-PapersOnLine* **53**(2), 5189–5194.
- Giselsson, P. (2012). Execution time certification for gradient-based optimization in model predictive control. In ‘51st IEEE Conference on Decision and Control’. pp. 3165–3170.
- Gros, S., M. Zanon, R. Quirynen, A. Bemporad and M. Diehl (2020). ‘From linear to nonlinear MPC: Bridging the gap via the real-time iteration’. *International Journal of Control* **93**(1), 62–80.
- Jerez, J. L., E. C. Kerrigan and G. A. Constantinides (2011). A condensed and sparse QP formulation for predictive control. In ‘50th IEEE Conference on Decision and Control and European Control Conference’. pp. 5217–5222.
- Kerrigan, E. C. and J. M. Maciejowski (2000). Soft constraints and exact penalty functions in model predictive control. In ‘Proceedings of the UKACC International Conference’. Cambridge, UK.
- Okawa, I. and K. Nonaka (2021). ‘Linear complementarity model predictive control with limited iterations for box-constrained problems’. *Automatica* **125**, 109429.
- Richter, S., C. N. Jones and M. Morari (2011). ‘Computational complexity certification for real-time MPC with input constraints based on the fast gradient method’. *IEEE Transactions on Automatic Control* **57**(6), 1391–1403.
- Stellato, B., G. Banjac, P. Goulart, A. Bemporad and S. Boyd (2020). ‘OSQP: An operator splitting solver for quadratic programs’. *Mathematical Programming Computation* **12**(4), 637–672.
- Wang, Y. and S. Boyd (2009). ‘Fast model predictive control using online optimization’. *IEEE Transactions on Control Systems Technology* **18**(2), 267–278.
- Wang, Y. and S. Boyd (2010). ‘Fast evaluation of quadratic control-Lyapunov policy’. *IEEE Transactions on Control Systems Technology* **19**(4), 939–946.
- Wright, S. J. (1997). *Primal-dual Interior-point Methods*. SIAM, Philadelphia, PA.
- Wu, L. and A. Bemporad (2023a). ‘A construction-free coordinate-descent augmented-Lagrangian method for embedded linear MPC based on ARX models’. *IFAC-PapersOnLine* **56**(2), 9423–9428.
- Wu, L. and A. Bemporad (2023b). ‘A Simple and Fast Coordinate-Descent Augmented-Lagrangian Solver for Model Predictive Control’. *IEEE Transactions on Automatic Control* **68**(11), 6860–6866.
- Wu, L. and R. D. Braatz (2023). ‘A direct optimization algorithm for input-constrained MPC’. *arXiv preprint arXiv:2306.15079*.
- Wu, L., K. Ganko and R. D. Braatz (2024a). ‘Time-certified input-constrained NMPC via Koopman operator’. *arXiv preprint arXiv:2401.04653*.
- Wu, L., K. Ganko, S. Wang and R. D. Braatz (2024b). ‘An execution-time-certified Riccati-based IPM algorithm for RTI-based input-constrained NMPC’. *arXiv preprint arXiv:2402.16186*.
- Xiao, W. and C. Belta (2021). ‘High-order control barrier functions’. *IEEE Transactions on Automatic Control* **67**(7), 3655–3662.
- Xie, Z., C. K. Liu and K. Hauser (2017). Differential dynamic programming with nonlinear constraints. In ‘IEEE International Conference on Robotics and Automation’. pp. 695–702.