2024 IEEE 63rd Conference on Decision and Control (CDC)
December 16-19, 2024. MiCo, Milan, Italy

An Execution-time-certified Riccati-based IPM Algorithm for RTI-based Input-constrained NMPC

Liang Wu[1], Krystian Ganko[1], Shimin Wang[1], Richard D. Braatz[1], Fellow, IEEE

***Abstract*— Establishing an execution time certificate in deploying model predictive control (MPC) is a pressing and challenging requirement. As nonlinear MPC (NMPC) results in nonlinear programs, differing from quadratic programs encountered in linear MPC, deriving an execution time certificate for NMPC seems an impossible task. Our prior work [1] introduced an input-constrained MPC algorithm with the exact and only *dimension-dependent* (*data-independent*) number of floating-point operations ([flops]). This paper extends it to input-constrained NMPC problems via the real-time iteration (RTI) scheme, which results in *data-varying* (but *dimension-invariant*) input-constrained MPC problems. Therefore, applying our previous algorithm can certify the execution time based on the assumption that processors perform fixed [flops] in constant time. As the RTI-based scheme generally results in MPC with a long prediction horizon, this paper employs the efficient factorized Riccati recursion, whose computational cost scales linearly with the prediction horizon, to solve the Newton system at each iteration. The execution-time certified capability of the algorithm is theoretically and numerically validated through a case study involving nonlinear control of the chaotic Lorenz system.**

***Index Terms*— Nonlinear model predictive control, real-time iteration, iteration complexity, Riccati recursion, execution time certificate**

2024 IEEE 63rd Conference on Decision and Control (CDC) | USGov | DOI: 10.1109/CDC56724.2024.10886531

I. Introduction

Model predictive control (MPC) is a model-based optimal control technique, which at each sampling time, handles the current feedback state information by solving an online constrained optimization formulated from a dynamical prediction model and user-specified constraints and objectives.

Hence, a question arises—can the adopted MPC algorithm finish the optimization task handling the current feedback state information before the next feedback state information arrives? An execution-time certificate that guarantees that the MPC returns a control action before the next sampling time, is a necessity when deploying MPC in production environments. In addition, theoretical execution time analysis can guide the trade-off of MPC settings, such as sampling time, adopted model complexity, and prediction horizon length, instead of using extensive simulations such as with the heavy calibration work of embedded MPC [2].

This work was supported by the U.S. Food and Drug Administration under the FDA BAA-22-00123 program, Award Number 75F40122C00200. Krystian Ganko was also supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Department of Energy Computational Science Graduate Fellowship under Award Number DE-SC0022158.

[1]Massachusetts Institute of Technology, Cambridge, MA 02139, USA, {liangwu,kkganko,bellewsm,braatz}@mit.edu

This execution-time certificate requirement has garnered increasing attention within recent years and is still an active research area [3]–[7]. All these works analyze the worst-case iteration bound of their proposed algorithms to derive a worst-case execution time based on the assumption that *the adopted computation platform performs a fixed number of floating-point operations ([flops]) in constant time*. That is,

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

This article also follows this assumption to certify the execution time of input-constrained nonlinear MPC (NMPC) problems.

Linear MPC is formulated using a linear process model, which leads to solving a convex quadratic program (QP) that can be efficiently solved using methods such as interior-point methods (IPM) [8], active set methods [9], and first-order methods [10]–[12]. NMPC instead adopts a nonlinear model, which results in a nonlinear program (NLP). NLPs not only have a higher computational burden but are also nearly impossible to derive the worst-case of required [flops] for their solution. For this reason, previous works [3]–[7] on executive-time certificates focused on linear MPC problems. A straightforward idea to develop an execution-time certified NMPC algorithm is to apply these previous algorithms [3]–[7] to some nonlinear-to-linear transformation techniques in the MPC field, such as successive online linearization or real-time iteration (RTI) [13].

However, this idea is impractical because online linearized MPC problems (online linearization or RTI scheme-based MPC) have time-changing problem data, e.g., the time-changing Hessian matrix. In [3]–[5], their derived computation complexity analysis is *data-dependent*, namely depending on the data of the resulting optimization problem. In [6], [7], their worst-case computation complexity certification relies on the complicated and computation-heavy (thus offline), which also limits their use in online linearized MPC problems. Thus, to the best of the authors' knowledge, no work extends these algorithms [3]–[7] to NMPC problems.

Our recent work [1] for the first time proposes a box-constrained QP (Box-QP) algorithm with an exact and only *dimension-dependent* (*data-independent*) number of iterations $\mathcal{N} = \left\lceil \frac{\log \frac{2n}{\epsilon}}{-2\log \frac{\sqrt{2n}}{\sqrt{2n}+\sqrt{2}-1}} \right\rceil + 1$, where n denotes the problem dimension and ϵ denotes the constant (such as 1×10^{-6}) stopping criterion. Therefore, our algorithm is ideally suited for online linearized-based NMPC schemes to certify the execution time.

Unlike the successive online linearization scheme, which only linearizes at the current point, the RTI scheme linearizes the whole previous trajectory and results in a linear time-varying model along the prediction horizon, thus providing a better nonlinear model approximation. More importantly, the high-approximation RTI scheme does not increase the computational time thanks to the *Preparation-Feedback Split*, in which the computation time is only marginally larger than linear MPC. Easy-to-understand tutorials with a detailed proof of nominal stability are available [13]–[15].

Therefore, this paper adopts the RTI scheme for nonlinear input-constrained MPC problems and then applies our previous algorithm [1] can certify the execution time by analyzing the total required [flops]. Our previous work [16] also employed the data-driven Koopman operator, which lifts the nonlinear system to a higher but linear system [17] for nonlinear input-constrained MPC problems to certify the execution time by using our previous algorithm [1]. Compared to the data-driven Koopman operator, the advantage of the RTI-based scheme is the ability to use existing first-principles continuous-time models from physical laws, such as found in chemical engineering and robotics. As such, this work is a complementary approach to applications where a first-principles continuous-time model is available.

A. Contributions

This work, for the first time, develops an execution-time-certified algorithm for RTI-based input-constrained NMPC problems. The RTI-based scheme performs well, with a small sampling time and a long time horizon [13], which results in MPC problems with a long prediction horizon. Then we employ an efficient factorized Riccati recursion characterized by a computational cost that scales linearly with the prediction horizon, facilitating the solution of the Newton system at each iteration. Additionally, as a byproduct, the utilized factorized Riccati recursion eliminates the need for computing the Hessian matrix, thereby reducing both the computational time and the implementation complexity of the preparation phase.

B. Notation

$\mathbb{R}^n$ denotes the space of n-dimensional real vectors, $\mathbb{R}^n_{++}$ is the set of all positive vectors of $\mathbb{R}^n$, and $\mathbb{N}_+$ is the set of positive integers. For a vector $z \in \mathbb{R}^n$, its Euclidean norm is $\|z\| = \sqrt{z_1^2 + z_2^2 + \cdots + z_n^2}$, $\|z\|_1 = \sum_{i=1}^n |z_i|$, $\mathrm{diag}(z) : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ maps an vector z to its corresponding diagonal matrix, and $z^2 = (z_1^2, z_2^2, \cdots, z_n^2)^\top$. A function is defined as $\|x\|_Q^2 = x^\top Q x$. Given two arbitrary vectors $z, y \in \mathbb{R}^n_{++}$, their Hadamard product is $zy = (z_1y_1, z_2y_2, \cdots, z_ny_n)^\top$, $\left(\frac{z}{y}\right) = \left(\frac{z_1}{y_1}, \frac{z_2}{y_2}, \cdots, \frac{z_n}{y_n}\right)^\top$, $\sqrt{z} = \left(\sqrt{z_1}, \sqrt{z_2}, \cdots, \sqrt{z_n}\right)^\top$. The vector of all ones is denoted by $e = (1, \cdots, 1)^\top$. $\lceil x \rceil$ maps x to the least integer greater than or equal to x. For $z, y \in \mathbb{R}^n$, let $\mathrm{col}(z, y) = [z^\top, y^\top]^\top$.

II. RTI-BASED INPUT-CONSTRAINED NMPC

In this article, we consider the tracking input-constrained NMPC problem $\mathrm{NLP} \equiv \mathrm{NLP}(\hat{x}_t, \mathbf{x}_t^{\mathrm{ref}}, \mathbf{u}_t^{\mathrm{ref}})$:

$$
\begin{aligned}
\mathrm{NLP} \triangleq \arg\min\ & \tfrac{1}{2}\|x_{t,N} - x_{t,N}^{\mathrm{ref}}\|^2_{W_N} \\
& + \sum_{k=0}^{N-1} \tfrac{1}{2}\|x_{t,k} - x_{t,k}^{\mathrm{ref}}\|^2_{W_x} + \tfrac{1}{2}\|u_{t,k} - u_{t,k}^{\mathrm{ref}}\|^2_{W_u} \\
\text{s.t.}\quad & x_{t,0} = \hat{x}_t, \\
& x_{t,k+1} = F(x_{t,k}, u_{t,k}),\ k \in \mathbb{Z}_0^{N-1}, \\
& \underline{u} \le u_{t,k} \le \bar{u}, \qquad\qquad k \in \mathbb{Z}_0^{N-1},
\end{aligned}
\tag{1}
$$

where N is the prediction horizon length; $\hat{x}_t$ is the feedback state at time t; $x_{t,k} \in \mathbb{R}^{n_x}$ and $u_{t,k} \in \mathbb{R}^{n_u}$ are the kth state and control input along the prediction horizon length at time t, respectively; $\mathbf{x}_t^{\mathrm{ref}}$ and $\mathbf{u}_t^{\mathrm{ref}}$ are the given reference trajectories at time t; and $F(\cdot)$ denotes the discrete-time nonlinear dynamics, which is obtained from continuous-time nonlinear dynamics via the Runge-Kutta 4 (RK4) method. The constraints $[\underline{u}, \bar{u}]$ come from the physical limitations on the control inputs (e.g., actuators).

Apart from using sequential quadratic programming methods to solve $\mathrm{NLP}(\hat{x}_t, \mathbf{x}_t^{\mathrm{ref}}, \mathbf{u}_t^{\mathrm{ref}})$, a well-known and successful technique is the RTI scheme. This scheme reduces the computational time of an NMPC problem to that of linear MPC, while providing nominal stability [15]. The key ingredients of RTI-based NMPC are

i) *Single Full Newton Step:*
At the time t, assuming that a good initial guess $(\mathbf{x}_t^{\mathrm{guess}}, \mathbf{u}_t^{\mathrm{guess}})$ is given, the full Newton step

$$(\mathbf{x}_t, \mathbf{u}_t) \leftarrow (\mathbf{x}_t^{\mathrm{guess}}, \mathbf{u}_t^{\mathrm{guess}}) + (\Delta\mathbf{x}_t, \Delta\mathbf{u}_t)$$

provides an excellent approximation of the fully converged NMPC solution, where $(\Delta\mathbf{x}_t, \Delta\mathbf{u}_t)$ is the solution of the QP problem $\mathrm{QP}_{\mathrm{NMPC}} \equiv \mathrm{QP}_{\mathrm{NMPC}}(\hat{x}_t, \mathbf{x}_t^{\mathrm{guess}}, \mathbf{u}_t^{\mathrm{guess}}, \mathbf{x}_t^{\mathrm{ref}}, \mathbf{u}_t^{\mathrm{ref}})$ in (2).
Here the trajectories $\Delta\mathbf{x}_t = (\Delta x_{t,0}, \cdots, \Delta x_{t,N})$ and $\Delta\mathbf{u}_t = (\Delta u_{t,0}, \cdots, \Delta u_{t,N})$ are the deviation between the system trajectories $\mathbf{x}_t, \mathbf{u}_t$ and the guess trajectories $\mathbf{x}_t^{\mathrm{guess}}$, $\mathbf{u}_t^{\mathrm{guess}}$,

$$\Delta x_{t,k} = x_{t,k} - x_{t,k}^{\mathrm{guess}}, \Delta u_{t,k} = u_{t,k} - u_{t,k}^{\mathrm{guess}}, k \in \mathbb{Z}_0^{N-1}$$

and $A_{t,k}, B_{t,k}, r_{t,k}$ stem from linearizing the discrete-time nonlinear dynamic F at time t,

$$
\begin{aligned}
A_{t,k} &= \left.\frac{\partial F(x,u)}{\partial x}\right|_{x_{t,k}^{\mathrm{guess}}, u_{t,k}^{\mathrm{guess}}}, && k \in \mathbb{Z}_0^{N-1}, \\
B_{t,k} &= \left.\frac{\partial F(x,u)}{\partial u}\right|_{x_{t,k}^{\mathrm{guess}}, u_{t,k}^{\mathrm{guess}}}, && k \in \mathbb{Z}_0^{N-1}, \\
r_{t,k} &= F(x_{t,k}^{\mathrm{guess}}, u_{t,k}^{\mathrm{guess}}) - x_{t,k+1}^{\mathrm{guess}}, && k \in \mathbb{Z}_0^{N-1}.
\end{aligned}
\tag{3}
$$

ii) *Shifting Initialization based on Previous Solution:*
A very good initial guess at time t can be constructed from the shifting of a good solution obtained at the previous time $t-1$. Let $(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})$ be the solution of $\mathrm{QP}_{\mathrm{NMPC}}$ at time $t-1$, then the initial guess

$$\text{QP}_{\text{NMPC}}(\hat{x}_t, \mathbf{x}_t^{\text{guess}}, \mathbf{u}_t^{\text{guess}}, \mathbf{x}_t^{\text{ref}}, \mathbf{u}_t^{\text{ref}}) \triangleq \arg\min \tfrac{1}{2}\|\Delta x_{t,N}\|^2_{W_N} + \Delta x_{t,N}^\top W_N^\top (x_{t,N}^{\text{guess}} - x_{t,N}^{\text{ref}}) + \tfrac{1}{2}\sum_{k=0}^{N-1} \|\Delta x_{t,k}\|^2_{W_x} + \Delta x_{t,k}^\top W_x^\top (x_{t,k}^{\text{guess}} - x_{t,k}^{\text{ref}}) + \tfrac{1}{2}\|\Delta u_{t,k}\|^2_{W_u} + \Delta u_{t,k}^\top W_u^\top (u_{t,k}^{\text{guess}} - u_{t,k}^{\text{ref}}) \quad (2)$$

$$\text{s.t.} \quad \Delta x_{t,0} = \hat{x}_t - x_{t,0}^{\text{guess}},$$
$$\Delta x_{t,k+1} = A_{t,k}\Delta x_{t,k} + B_{t,k}\Delta u_{t,k} + r_{t,k}, \quad k \in \mathbb{Z}_0^{N-1},$$
$$\underline{u} - u_{t,k}^{\text{guess}} \le \Delta u_{t,k} \le \bar{u} - u_{t,k}^{\text{guess}}, \quad k \in \mathbb{Z}_0^{N-1}.$$

$(\mathbf{x}_t^{\text{guess}}, \mathbf{u}_t^{\text{guess}})$ is constructed from

$$\begin{aligned} x_{t,k}^{\text{guess}} &= x_{t-1,k+1}, \quad k \in \mathbb{Z}_0^{N-1}, \\ u_{t,k}^{\text{guess}} &= u_{t-1,k+1}, \quad k \in \mathbb{Z}_0^{N-2}, \\ u_{t,N-1}^{\text{guess}} &= u_{t,N-2}^{\text{guess}} = u_{t-1,N-1}, \\ x_{t,N}^{\text{guess}} &= f(x_{t,N-1}^{\text{guess}}, u_{t,N-1}^{\text{guess}}). \end{aligned} \quad (4)$$

Note that shifting the given reference trajectories $(\mathbf{x}_t^{\text{ref}}, \mathbf{u}_t^{\text{ref}})$ to construct the initial guess could lead to poor performance when the actual system trajectory is not in the neighborhood of the reference.

iii) *Preparation-Feedback Split:*
The RTI-based NMPC and the linear MPC both only require solving one QP problem online, but their difference is that the QP of linear MPC is constructed offline once whereas the resulting QP of RTI-based NMPC is constructed online. To further reduce the computational delay, RTI-based NMPC splits the online computation into two phases,
- *A preparation phase*, completing the part of computations related to the QP_{NMPC} construction as much as possible before the arrival of the feedback state $\hat{x}_t$;
- *A feedback phase*, finishing the QP construction and solving the QP_{NMPC} upon the feedback state $\hat{x}_t$ arriving.

In this manner, the computational time of RTI-based NMPC is only marginally larger than linear MPC.

A. Computing the sensitivities $\{A_{t,k}, B_{t,k}, r_{t,k}\}_{k=0}^{N-1}$ from continuous-time systems

In the context of NMPC, the nonlinear model is usually derived from physical laws as a continuous-time nonlinear differential equation,

$$\dot{x}(t) = f(x(t), u(t)). \quad (5)$$

In this case, computing the sensitivities $\{A_{t,k}, B_{t,k}, r_{t,k}\}_{k=0}^{N-1}$ from (3) is not straightforward. For example, the discrete-time nonlinear dynamic F is obtained from continuous-time f via the RK4 method to preserve both integration accuracy and computation efficiency. Here, we illustrate how to compute the sensitivities $\{A_{t,k}, B_{t,k}, r_{t,k}\}_{k=0}^{N-1}$ without resorting to defining F explicitly. Denote

$$f_x(x(t), u(t)) = \left.\frac{\partial f(x,u)}{\partial x}\right|_{x=x(t), u=u(t)},$$
$$f_u(x(t), u(t)) = \left.\frac{\partial f(x,u)}{\partial u}\right|_{x=x(t), u=u(t)},$$

Procedure 1 Computing $A_{t,k}, B_{t,k}, r_{t,k}$

Input: $x_{t,k}^{\text{guess}}, u_{t,k}^{\text{guess}}, x_{t,k+1}^{\text{guess}}$, the number of discretization steps N_s, the discretization time $t_i = \frac{\Delta t}{N_s}$, $f_x(\cdot)$ and $f_u(\cdot)$.

1. $x = x_{t,k}^{\text{guess}}, u = u_{t,k}^{\text{guess}}, A = I, B = 0$
2. **for** $j = 1, \ldots, N_s$ **do**
2.1. $\kappa(1) \leftarrow f(x, u)$
2.2. $[\kappa_x(1), \kappa_u(1)] \leftarrow f_x(x, u)[A, B] + [0, f_u(x, u)]$
2.3. $\kappa(2) \leftarrow f(x + \frac{1}{2}t_i\kappa(1), u)$
2.4. $[\kappa_x(2), \kappa_u(2)] \leftarrow f_x(x + \frac{1}{2}t_i\kappa(1), u)[A + \frac{1}{2}t_i\kappa_x(1), B + \frac{1}{2}t_i\kappa_u(1)] + [0, f_u(x + \frac{1}{2}t_i\kappa(1), u)]$
2.5. $\kappa(3) \leftarrow f(x + \frac{1}{2}t_i\kappa(2), u)$
2.6. $[\kappa_x(3), \kappa_u(3)] \leftarrow f_x(x + \frac{1}{2}t_i\kappa(2), u)[A + \frac{1}{2}t_i\kappa_x(2), B + \frac{1}{2}t_i\kappa_u(2)] + [0, f_u(x + \frac{1}{2}t_i\kappa(2), u)]$
2.7. $\kappa(4) \leftarrow f(x + t_i\kappa(3), u)$
2.8. $[\kappa_x(4), \kappa_u(4)] \leftarrow f_x(x + t_i\kappa(3), u)[A + t_i\kappa_x(3), B + t_i\kappa_u(3)] + [0, f_u(x + t_i\kappa(3), u)]$
2.9. $x \leftarrow x + \frac{t_i}{6}(\kappa(1) + 2\kappa(2) + 2\kappa(3) + \kappa(4))$
2.10. $[A, B] \leftarrow [A, B] + \frac{t_i}{6}([\kappa_x(1), \kappa_u(1)] + 2[\kappa_x(2), \kappa_u(2)] + 2[\kappa_x(3), \kappa_u(3)] + [\kappa_x(4), \kappa_u(4)])$
3. **end**.
4. $A_{t,k} \leftarrow A, B_{t,k} \leftarrow B, r_{t,k} \leftarrow x - x_{t,k+1}^{\text{guess}}$

Output: $A_{t,k}, B_{t,k}, r_{t,k}$.

and consider the N_s integration steps over the sampling time Δt resulting in $t_i = \frac{\Delta t}{N_s}$ discretization time, Procedure 1 shows how to compute the kth sensitivities $A_{t,k}, B_{t,k}, r_{t,k}$ from the continuous-time dynamic f via the RK4 method. Procedure 1 needs to be executed N times to calculate all $\{A_{t,k}, B_{t,k}, r_{t,k}\}_{k=0}^{N-1}$ in the *preparation phase*.

Lemma 1: Let m_f, m_{f_x}, m_{f_u} denote the flops required by the evaluation of $f(\cdot), f_x(\cdot), f_u(\cdot)$, respectively. Then the flops required by the computation of the sensitivities $\{A_{t,k}, B_{t,k}, r_{t,k}\}_{k=0}^{N-1}$ is

$$N \times (n_x + n_u + N_s(4m_f + 4m_{f_x} + 4m_{f_u} + 8n_x^3 + 8n_x^2 n_u + 10n_x^2 + 10n_x n_u + 16n_x) + n_x^2 + n_x n_u + n_x).$$

B. QP_{NMPC} construction

By using a condensing construction that eliminates the states, QP_{NMPC} (2) can be formulated as a box-constrained QP (Box-QP). Our time-certified IPM algorithm (see Section III) is tailored for the Box-QP with the unit box constraint $[-e, e]$. After generating $\{A_{t,k}, B_{t,k}, r_{t,k}\}_{k=0}^{N-1}$, we first scale

the control inputs $\Delta u_{t,k}$ subject to $[-e, e]$, denoted as $\Delta\bar{u}_{t,k}$, through

$$\Delta u_{t,k} = D\Delta\bar{u}_{t,k} + d_{t,k}, \tag{6}$$

where

$$\begin{aligned} D &= \tfrac{1}{2}\operatorname{diag}(\bar{u} - \underline{u}), \\ d_{t,k} &= \tfrac{1}{2}(\bar{u} + \underline{u}) - u_{t,k}^{\text{guess}}, \quad k \in \mathbb{Z}_0^{N-1}. \end{aligned} \tag{7}$$

Then, the associated terms are updated as

$$\begin{aligned} \bar{B}_{t,k} &= B_{t,k}D, && k \in \mathbb{Z}_0^{N-1}, \\ \bar{r}_{t,k} &= r_{t,k} + B_{t,k}d_{t,k}, && k \in \mathbb{Z}_0^{N-1}. \end{aligned} \tag{8}$$

Define $z \triangleq \operatorname{col}(\Delta\bar{u}_{t,0}, \cdots, \Delta\bar{u}_{t,N-1}) \in \mathbb{R}^n$, where $n = Nn_u$, $\bar{Q} \triangleq \operatorname{diag}(W_x, \cdots, W_x, W_N)$, $\bar{R} \triangleq \operatorname{diag}(DW_uD, \cdots, DW_uD)$, and

$$S \triangleq \begin{bmatrix} \bar{B}_{t,0} & 0 & \cdots & 0 \\ A_{t,0}\bar{B}_{t,0} & \bar{B}_{t,1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \prod_{k=0}^{N-1} A_{t,k}\bar{B}_{t,0} & \prod_{k=0}^{N-2} A_{t,k}\bar{B}_{t,1} & \cdots & \bar{B}_{t,N-1} \end{bmatrix}. \tag{9}$$

Then construct QP_{NMPC} (2) as the scaled Box-QP

$$z^* = \arg\min_z J(z) = \tfrac{1}{2}z^\top Hz + z^\top h \tag{10a}$$

$$\text{s.t. } -e \le z \le e, \tag{10b}$$

where the Hessian matrix is given by

$$H = \bar{R} + S^\top \bar{Q} S, \tag{11}$$

and the gradient vector is given by

$$h = S^\top \bar{Q} g + \begin{bmatrix} DW_u(\frac{1}{2}(\bar{u} + \underline{u}) - u_{t,0}^{\text{ref}}) \\ \vdots \\ DW_u(\frac{1}{2}(\bar{u} + \underline{u}) - u_{t,N-1}^{\text{ref}}) \end{bmatrix}, \tag{12}$$

in which the vector $g = g_1 + g_2$ and

$$g_1 \triangleq \mathbf{x}_t^{\text{guess}} - \mathbf{x}_t^{\text{ref}} + \begin{bmatrix} \bar{r}_{t,0} \\ A_{t,1}\bar{r}_{t,0} + \bar{r}_{t,1} \\ \vdots \\ \sum_{k=1}^{N-1} A_{t,k}\bar{r}_{t,0} + \cdots + \bar{r}_{t,N-1} \end{bmatrix}, \tag{13a}$$

$$g_2 \triangleq \begin{bmatrix} A_{t,0} \\ A_{t,1}A_{t,0} \\ \vdots \\ \prod_{k=0}^{N-1} A_{t,k} \end{bmatrix} (\hat{x}_t - x_{t,0}^{\text{guess}}) \tag{13b}$$

Lemma 2: The computation of $\{\bar{B}_{t,k}, \bar{r}_{t,k}\}_{k=0}^{N-1}$ requires $N(3n_xn_u + n_x)$ flops, the computation of S requires $(N^2 - N)n_xn_u^2$ flops, the computation of H requires $(2N^3 + N^2 + N)n_x^2n_u + Nn_u^2$ flops, the computation of g_1 requires $2Nn_x + 2(N-1)n_x^2$ flops, the computation of g_2 requires $n_x + 2Nn_x^2$ flops, and the computation of h requires $2Nn_x^2 + (N^2 + N)n_xn_u + (N^2 - N)n_x + n_u + N(2n_u + n_u^2)$ flops.

Remark 1: By Lemma 2, the computation of H dominates, especially when N is considerably larger than n_x and n_u. Consequently, circumventing the computation of H has the potential to diminish the computational time of the *preparation phase*. The subsequent section demonstrates that the Riccati recursion technique not only eliminates the need to compute H but also lowers the cost associated with the Newton step.

III. Time-certified Riccati-based IPM Algorithm

Based on the path-following full-Newton IPM, our recent work [1] proposed an *direct* optimization algorithm to solve the Box-QP (10). Its Karush–Kuhn–Tucker (KKT) conditions are

$$Hz + h + \gamma - \theta = 0, \tag{14a}$$

$$z + \alpha - e = 0, \tag{14b}$$

$$z - \omega + e = 0, \tag{14c}$$

$$\gamma\phi = 0, \tag{14d}$$

$$\theta\psi = 0, \tag{14e}$$

$$(\gamma, \theta, \alpha, \omega) \ge 0. \tag{14f}$$

A positive parameter τ was introduced by the path-following IPM to replace (14d) and (14e) by

$$\gamma\phi = \tau^2 e, \tag{15a}$$

$$\theta\psi = \tau^2 e. \tag{15b}$$

As τ tends to 0, the path $(z_\tau, \gamma_\tau, \theta_\tau, \phi_\tau, \psi_\tau)$ converges to a solution of (14). The feasible variants of the path-following IPM algorithm boast the best theoretical $O(\sqrt{n})$ iteration complexity [18]. Therefore, our algorithm is based on feasible IPM in which all iterates are required to be strictly in the feasible set

$$\mathcal{F}^0 \triangleq \big\{(z, \gamma, \theta, \phi, \psi) \big| \text{ (14a)-(14c) hold}, (\gamma, \theta, \phi, \psi) > 0\big\}.$$

A. Strictly feasible initial point

In our recent work [1], a cost-free initialization strategy was proposed to find a strictly feasible initial point that also satisfies the specific conditions. A strictly feasible initial point is

$$z^0 = 0, \gamma^0 = \|h\|_\infty - \tfrac{1}{2}h, \theta^0 = \|h\|_\infty + \tfrac{1}{2}h, \phi^0 = e, \psi^0 = e,$$

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

Remark 2 (Initialization strategy): For $h = 0$, the optimal solution of problem (10) is $z^* = 0$; for $h \neq 0$, first scale the objective (10a) (which does not change the optimal solution) as

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

With the definitions $\tilde{H} = \frac{1}{\|h\|_\infty}H$ and $\tilde{h} = \frac{1}{\|h\|_\infty}h$, $\|\tilde{h}\|_\infty = 1$ and (14a) can be replaced by

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

and the initial points

$$z^0 = 0,\ \gamma^0 = 1 - \lambda\tilde{h},\ \theta^0 = 1 + \lambda\tilde{h},\ \phi^0 = e,\ \psi^0 = e \quad (16)$$

can be adopted, where $\lambda = \frac{1}{\sqrt{n+1}}$. It is straightforward to verify that (16) 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 [1, Lemma 4].

B. Newton direction

Denote $v = \mathrm{col}(\gamma, \theta) \in \mathbb{R}^{2n}$ and $s = \mathrm{col}(\phi, \psi) \in \mathbb{R}^{2n}$. Then replace (15a) and (15b) by $vs = \tau^2 e$ to obtain the new complementary condition,

$$\sqrt{vs} = \sqrt{\tau^2 e}. \quad (17)$$

From *Remark 2*, $(z, v, s) \in \mathcal{F}^0$ and a direction $(\Delta z, \Delta v, \Delta s)$ can be obtained by solving the system of linear equations,

$$2\lambda\tilde{H}\Delta z + \Omega\Delta v = 0, \quad (18a)$$

$$\Omega^\top \Delta z + \Delta s = 0, \quad (18b)$$

$$\sqrt{\frac{s}{v}}\Delta v + \sqrt{\frac{v}{s}}\Delta s = 2(\tau e - \sqrt{vs}), \quad (18c)$$

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

$$\Delta\gamma = \frac{\gamma}{\phi}\Delta z + 2\left(\sqrt{\frac{\gamma}{\phi}}\tau e - \gamma\right), \quad (19a)$$

$$\Delta\theta = -\frac{\theta}{\psi}\Delta z + 2\left(\sqrt{\frac{\theta}{\psi}}\tau e - \theta\right), \quad (19b)$$

$$\Delta\phi = -\Delta z, \quad (19c)$$

$$\Delta\psi = \Delta z \quad (19d)$$

reduces (18) into a more compact system of linear equations,

$$\left(2\lambda\tilde{H} + \mathrm{diag}\left(\frac{\gamma}{\phi}\right) + \mathrm{diag}\left(\frac{\theta}{\psi}\right)\right)\Delta z = 2\left(\sqrt{\frac{\theta}{\psi}}\tau e - \sqrt{\frac{\gamma}{\phi}}\tau e + \gamma - \theta\right) \quad (20)$$

C. Implementation of the Newton step via factorized Riccati recursion

Solving the linear system (20) typically requires $O(N^3 n_u^3)$ flops. In general, the RTI-based NMPC scheme demonstrates effective performance provided that the sampling time Δt is short and the time prediction horizon T_p is long [13]. This often leads to a prolonged prediction horizon length $N = \frac{T_p}{\Delta t}$, e.g., of $20, 40$, and 60 in practical applications.

This article adopts the Riccati recursion to solve the linear system (20) efficiently when the ratio $\frac{n_x}{n_u}$ is not large and the prediction horizon N is long. Subsequently, we illustrate how the Riccati recursion not only minimizes the flops needed to solve (20) but also circumvents the computation of H.

The linear system (20) is equivalent to the formulation

$$\min\ \frac{1}{2}\Delta z^\top\left(2\lambda\tilde{H} + \mathrm{diag}\left(\frac{\gamma}{\phi}\right) + \mathrm{diag}\left(\frac{\theta}{\psi}\right)\right)\Delta z - 2\Delta z^\top\left(\sqrt{\frac{\theta}{\psi}}\tau e - \sqrt{\frac{\gamma}{\phi}}\tau e + \gamma - \theta\right). \quad (21)$$

Let $\mathrm{col}(\Delta\hat{u}_0, \cdots, \Delta\hat{u}_{N-1}) \triangleq \Delta z$, $\mathrm{col}(\Delta\hat{x}_1, \cdots, \Delta\hat{x}_N) \triangleq S\Delta z$, $\Delta\hat{x}_0 = 0$, and $\hat{Q}_{k+1} \triangleq \frac{2\lambda}{\|h\|_\infty}W_x, k \in \mathbb{Z}_0^{N-2}$, $\hat{Q}_N \triangleq \frac{2\lambda}{\|h\|_\infty}W_N$, for $k \in \mathbb{Z}_0^{N-1}$. Exploiting the structure of S in (9) and H in (11), gives that

$$\hat{R}_k \triangleq \frac{2\lambda}{\|h\|_\infty}DW_uD + \mathrm{diag}\left(\frac{\gamma}{\phi} + \frac{\theta}{\psi}\right)_{kn_u+1:(k+1)n_u},$$

$$\hat{g}_k = -2\left(\sqrt{\frac{\theta}{\psi}}\tau e - \sqrt{\frac{\gamma}{\phi}}\tau e + \gamma - \theta\right)_{kn_u+1:(k+1)n_u}.$$

The above formulation (21) is equivalent to the unconstrained linear quadratic regulator problem,

$$\begin{aligned}\min\ &\sum_{k=0}^{N-1}\left(\tfrac{1}{2}\|\Delta\hat{u}_k\|^2_{\hat{R}_k} + \hat{g}_k^\top\Delta\hat{u}_t + \tfrac{1}{2}\|\Delta\hat{x}_{k+1}\|^2_{\hat{Q}_{k+1}}\right)\\ \text{s.t.}\quad &\Delta\hat{x}_0 = 0,\\ &\Delta\hat{x}_{k+1} = A_{t,k}\Delta\hat{x}_k + \bar{B}_{t,k}\Delta\hat{u}_k,\quad k \in \mathbb{Z}_0^{N-1},\end{aligned} \quad (22)$$

which can be solved efficiently by Riccati recursion. Instead of adopting the classical Riccati recursion, this article adopts the factorized Riccati recursion to further reduce the computational cost [19], e.g., as shown in Procedure 2.

Procedure 2 Implementation of the Newton step via factorized Riccati recursion

Input: $\{A_{t,k}, \bar{B}_{t,k}, \hat{Q}_k, \hat{R}_k, \hat{g}_k\}_{k=0}^{N-1}, \hat{Q}_N$

$L_N \leftarrow \mathrm{chol}(\hat{Q}_N)$
$p_N \leftarrow 0$
for $k = N-1, \cdots, 0$ **do**
  $[L_k^\top\bar{B}_{t,k}, L_k^\top A_{t,k}] \leftarrow L_{k+1}^\top \cdot_{\mathrm{dtrmm}} [\bar{B}_{t,k}, A_{t,k}]$
  $\mathcal{O} \leftarrow [L_k^\top\bar{B}_{t,k}, L_t^\top A_{t,k}]^\top \cdot_{\mathrm{dsyrk}} [L_t^\top\bar{B}_{t,k}, L_k^\top A_{t,k}]$
  $\begin{bmatrix}\Lambda_k & \\ M_k & L_k\end{bmatrix} \leftarrow \mathrm{chol}_L\left(\mathcal{O} + \begin{bmatrix}\hat{R}_k & \\ & \hat{Q}_k\end{bmatrix}\right)$
  $q_k \leftarrow (\Lambda_k\Lambda_k^\top)^{-1}(\bar{B}_{t,k}^\top p_{k+1} + \hat{g}_k)$
  $p_k \leftarrow A_{t,k}^\top p_{k+1} - M_k\Lambda_k^\top q_k$
end
$\Delta\hat{x}_0 \leftarrow 0$
for $k = 0, \cdots, N-1$ **do**
  $\Delta\hat{u}_k \leftarrow -\Lambda_k^{-\top}M_k^\top\Delta\hat{x}_k - q_k$
  $\Delta\hat{x}_{k+1} \leftarrow A_{t,k}\Delta\bar{x}_k + \bar{B}_{t,k}\Delta\hat{u}_k$
end
Output: $\Delta z \leftarrow \mathrm{col}(\Delta\bar{u}_0, \cdots, \Delta\bar{u}_{N-1})$

Lemma 3 (See [19]): Procedure 2 requires a total of

$$N(\tfrac{7}{3}n_x^3 + 4n_x^2n_u + 2n_xn_u^2 + \tfrac{1}{3}n_u^3) + N(8n_x^2 + 8n_xn_u + 2n_u^2)$$

[flops].

D. Iteration complexity and algorithm implementation

The complete RTI-based input-constrained MPC scheme is summarized as Algorithm 3. Next, we derive the iteration complexity. Denote $\beta \triangleq \sqrt{vs}$ and define the proximity measure as

$$\xi(\beta, \tau) = \tfrac{\|\tau e - \beta\|}{\tau}. \quad (23)$$

Lemma 4 (See [1]): Let $\xi := \xi(\beta,\tau) < 1$. Then the full Newton step is strictly feasible, i.e., $v_+ > 0$ and $s_+ > 0$.

Lemma 5 (See [1]): After a full Newton step, let $v_+ = v + \Delta v$ and $s_+ = s + \Delta s$; then the duality gap is

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

Lemma 6 (See [1]): Suppose that $\xi = \xi(\beta,\tau) < 1$ and $\tau_+ = (1-\eta)\tau$ where $0 < \eta < 1$. Then,

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

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

Lemma 7 (See [1]): 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 $\xi^0 \le \frac{1}{\sqrt{2}}$ and $\xi(\beta,w) \le \frac{1}{\sqrt{2}}$ are always satisfied.

Lemma 8 (See [1]): Let $\eta = \frac{\sqrt{2}-1}{\sqrt{2n}+\sqrt{2}-1}$ and $\tau^0 = \frac{1}{1-\eta}$, Algorithm 3 exactly requires

$$\mathcal{N} = \left\lceil \frac{\log\frac{2n}{\epsilon}}{-2\log\frac{\sqrt{2n}}{\sqrt{2n}+\sqrt{2}-1}} \right\rceil + 1 \tag{24}$$

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

Lemmas 1, 2, 3, 8 imply the totals in Theorem 1.

Theorem 1: In the *preparation phase* of Algorithm 3, Step 1 takes $Nn_x + Nn_u + m_f$ [flops], Step 2 takes $N(n_x+n_u+N_s(4m_f+4m_{f_x}+4m_{f_u}+8n_x^3+8n_x^2n_u+10n_x^2+10n_xn_u+16n_x)+n_x^2+n_xn_u+n_x)$ [flops], Step 3 takes $n_u+Nn_u+N(2n_xn_u+n_x)+(N^2-N)n_xn_u^2+2Nn_x+2(N-1)n_x^2$ [flops]; In the *feedback phase* of Algorithm 3, Step 1 takes $n_x+4Nn_x^2+(N^2+N)n_xn_u+(N^2-N)n_x+n_u+N(2n_u+n_u^2)$ [flops], Step 2 takes Nn_u [flops], Step 3 takes $5Nn_u+3$ [flops], Step 4 takes $\mathcal{N}(1+N(\frac{7}{3}n_x^3+4n_x^2n_u+2n_xn_u^2+\frac{1}{3}n_u^3)+N(8n_x^2+8n_xn_u+2n_u^2)+15Nn_u+5n_x)$ [flops], Step 5 takes n_x [flops], Step 6 takes $N(2n_u+n_x^2+n_xn_u+2n_x)$ [flops], Step 7 takes $(N+1)n_x+Nn_u$[flops].

IV. Numerical Example: Chaotic Lorenz Stabilization

The Lorenz system is a well-known nonlinear dynamical system to model convective hydrodynamic flows [20]. The consequence of chaos is that, under certain parameterizations, the trajectories of the deterministic Lorenz system become arbitrarily sensitive to small perturbations in the initial conditions. Slightly different trajectories separate vastly over long horizons, which is popularly coined as the *butterfly effect*. In this work, we address the stabilization task of applying affine control signals to each coordinate of the Lorenz system,

$$\begin{aligned}\dot{x} &= \sigma(y-x)+u_x,\\ \dot{y} &= x(\rho-z)-y+u_y,\\ \dot{z} &= xy-\beta z+u_z,\end{aligned} \tag{25}$$

where (x,y,z) are the states, and $(u_x,u_y,u_z) \in [-3,3]^3$ are the control inputs. The parameters $\sigma = 10$, $\rho = 28$, $\beta = 8/3$ are chosen so that 25 exhibits chaotic behavior

Algorithm 3 A time-certified IPM algorithm for RTI-based input-constrained MPC (2) at sampling time t

Preparation phase: performed over the time interval $[t-\Delta t, t]$

Input: previous solution $(\mathbf{x}_{t-1},\mathbf{u}_{t-1})$, reference $(\mathbf{x}_t^{\text{ref}},\mathbf{u}_t^{\text{ref}})$, continuous-time dynamic $f(\cdot)$, and its partials $f_x(\cdot), f_u(\cdot)$;

1. Construct $(\mathbf{x}_t^{\text{guess}},\mathbf{u}_t^{\text{guess}})$ according to (4) from the previous solution $(\mathbf{x}_{t-1},\mathbf{u}_{t-1})$;
2. Calculate $\{A_{t,k},B_{t,k},r_{t,k}\}_{k=0}^{N-1}$ by performing Procedure 1 $\forall k \in \mathbb{Z}_0^{N-1}$;
3. Calculate the scaling factors $D,\{d_{t,k}\}_{k=0}^{N-1}$ from (7), $\{\bar{B}_{t,k},\bar{r}_{t,k}\}_{k=0}^{N-1}$ from (8), S from (9), and g_1 from (13a)

return $D,\{d_{t,k}\}_{k=0}^{N-1}$, $\{A_{t,k},\bar{B}_{t,k},r_{t,k}\}_{k=0}^{N-1}$ and S, g_1

Feedback phase: performed at time t upon arrival of $\hat{x}_t$

Input: the feedback states $\hat{x}_t$, D, $\{d_{t,k}\}_{k=0}^{N-1}$, $\{A_{t,k},\bar{B}_{t,k},\bar{r}_{t,k}\}_{k=0}^{N-1}$, S, g_1, the stopping tolerance ϵ; the required exact number of iterations $\mathcal{N} = \left\lceil \frac{\log\frac{2n}{\epsilon}}{-2\log\frac{\sqrt{2n}}{\sqrt{2n}+\sqrt{2}-1}} \right\rceil + 1$.

1. Calculate g_2 from (13b) and h from (12) based on $\hat{x}_t, S, g_1$;
2. **if** $\|h\|_\infty = 0$, $z \leftarrow 0$ and go to Step 5; **otherwise**,
3. Initialize $(z,\gamma,\theta,\phi,\psi)$ from (16) where $\lambda \leftarrow \frac{1}{\sqrt{n+1}}$, $\eta \leftarrow \frac{\sqrt{2}-1}{\sqrt{2n}+\sqrt{2}-1}$ and $\tau \leftarrow \frac{1}{1-\eta}$;
4. **for** $i = 1,2,\cdots,\mathcal{N}$ **do**

4.1. $\tau \leftarrow (1-\eta)\tau$;
4.2. Obtain Δz by performing Procedure 2;
4.3. Calculate $(\Delta\gamma,\Delta\theta,\Delta\alpha,\Delta\omega)$ from (19);
4.4. $z \leftarrow z+\Delta z$, $\gamma \leftarrow \gamma+\Delta\gamma$, $\theta \leftarrow \theta+\Delta\theta$, $\alpha \leftarrow \alpha+\Delta\alpha$, $\omega \leftarrow \omega+\Delta\omega$;

5. $\Delta x_{t,0} \leftarrow \hat{x}_t - x_{t,0}^{\text{guess}}$;
6. for $k = 0,1,\cdots,N-1$

$$\Delta u_{t,k} \leftarrow Dz_{kn_u+1:kn_u+n_u} + d_{t,k}$$
$$\Delta x_{t,k+1} \leftarrow A_{t,k}\Delta x_{t,k} + \bar{B}_{t,k}z_{kn_u+1:kn_u+n_u} + \bar{r}_{t,k}$$

7. Apply the full Newton step $(\mathbf{x}_t,\mathbf{u}_t) \leftarrow (\mathbf{x}_t^{\text{guess}},\mathbf{u}_t^{\text{guess}}) + (\Delta\mathbf{x}_t,\Delta\mathbf{u}_t)$

return NMPC solution $(\mathbf{x}_t,\mathbf{u}_t)$

with two strange attractors, $(\pm\sqrt{\beta(\rho-1)},\pm\sqrt{\beta(\rho-1)},\rho-1)$. Stabilizing the chaotic system to one of its attractors $(6\sqrt{2},6\sqrt{2},27)$ is a nontrivial task due to its fractal-like trajectories and heightened sensitivity to small perturbations. Here, the desired reference is $x_{t,k}^{\text{ref}} = \text{col}(6\sqrt{2},6\sqrt{2},27)$, $u_{t,k}^{\text{ref}} = \text{col}(0,0,0), k \in \mathbb{Z}_0^{N-1}$. The weight matrices were chosen as $W_N = W_x = I$ and $W_u = 0.1I$ for the states and control inputs, respectively. The sampling time is $\Delta t = 0.01$ s, and the time prediction horizon is $T_p = 0.2$ s, so the prediction horizon length is $N = \frac{T_p}{\Delta t} = 20$. Additionally, we opt for $N_s = 2$ integration steps to calculate the sensitivities.

Before closed-loop simulation, we can exactly calculate the flops required by the *preparation phase* and *feedback phase* of Algorithm 3 at each sampling time. The dimension

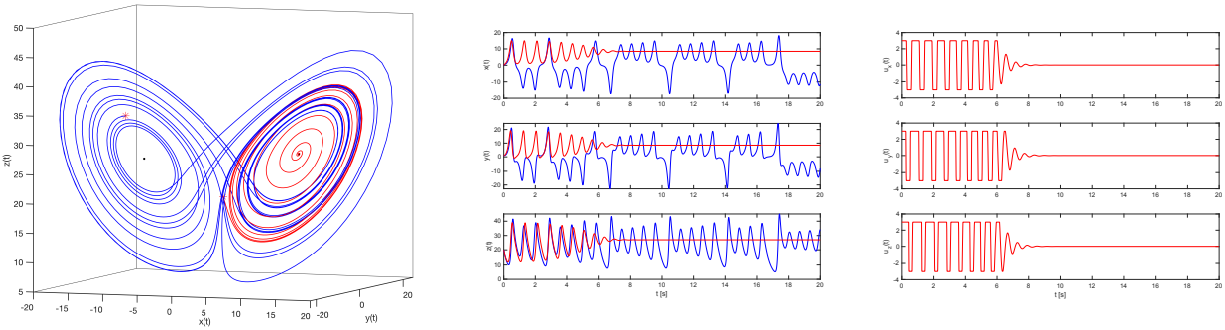

Fig. 1. Closed-loop simulation of the chaotic Lorenz system with the RTI-based input-constrained MPC. The blue and red lines represent the results of uncontrolled and MPC controlled, respectively. Left: phase trajectory. Middle: time evolution of the three coordinates. Right: the three control inputs.

of the Box-QP (10) is $n = 3{\times}N = 3{\times}20 = 60$, and we adopt the stopping criteria $\epsilon = 10^{-6}$, so Algorithm 3 will exactly perform $\mathcal{N} = \left\lceil \frac{\log \frac{2\times 60}{10^{-6}}}{-2\log \frac{\sqrt{2\times 60}}{\sqrt{2\times 60}+\sqrt{2}-1}} \right\rceil + 1 = 252$ iterations in the *feedback phase*. The Lorenz system flux expressions $f(\cdot)$, $f_x(\cdot)$, and $f_u(\cdot)$ take $m_f = 10$, $m_{f_x} = 4$, and $m_{f_u} = 0$ (since in $f_x(\cdot)$ only 4 elements vary and $f_u(\cdot) = [1, 1, 1]^\top$) [flops], respectively. Thus, by Theorem 1, the *preparation phase* requires a total of $4.05{\times}10^4$ [flops] and the *feedback phase* requires a total of $2.23{\times}10^6$ [flops]. Altogether, the algorithm execution requires a wall time of 0.0025 s on a personal laptop with 1 Gflop/s computing power. Hence, the execution time certificate that the execution time will be less than the adopted sampling time of $\Delta t = 0.01$ s, is obtained.

Algorithm 3 is executed in MATLAB2023a via a C-mex interface, and the closed-loop simulation was performed on a MacBook Pro with 2.7 GHz 4-core Intel Core i7 processors and 16GB RAM. The number of iterations is exactly 252, and the execution time of the two phases adds up to about 0.002 s which is less than $\Delta t = 0.01$ s. The closed-loop simulation results are depicted in Fig. 1, which illustrates that the MPC effectively stabilizes the chaotic Lorenz system to the specified attractor, in stark contrast to the *butterfly effect* trajectories of the uncontrolled case. The three control inputs do not violate $[-3, 3]$ and eventually converge to zeros.

V. Conclusion

This article proposes an execution-time-certified algorithm for RTI-based input-constrained NMPC problems. To address the MPC setting with a long prediction horizon that RTI-based NMPC often results in, the factorized Riccati recursion, whose computation cost scales linearly to the prediction horizon, is used to efficiently solve the Newton system at each iteration. In the future, we will continue work on extensions to general execution-time-certified NMPC algorithms including both input and state constraints.

References

[1] L. Wu and R. D. Braatz, "A direct optimization algorithm for input-constrained MPC," *IEEE Transactions on Automatic Control*, 2024, in press,arXiv:2306.15079.

[2] M. Forgione, D. Piga, and A. Bemporad, "Efficient calibration of embedded MPC," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 5189–5194, 2020.

[3] S. Richter, C. N. Jones, and M. Morari, "Computational complexity certification for real-time MPC with input constraints based on the fast gradient method," *IEEE Transactions on Automatic Control*, vol. 57, no. 6, pp. 1391–1403, 2011.

[4] A. Bemporad and P. Patrinos, "Simple and certifiable quadratic programming algorithms for embedded linear model predictive control," *IFAC Proceedings Volumes*, vol. 45, no. 17, pp. 14–20, 2012.

[5] P. Giselsson, "Execution time certification for gradient-based optimization in model predictive control," in *51st IEEE Conference on Decision and Control*, 2012, pp. 3165–3170.

[6] G. Cimini and A. Bemporad, "Exact complexity certification of active-set methods for quadratic programming," *IEEE Transactions on Automatic Control*, vol. 62, no. 12, pp. 6094–6109, 2017.

[7] D. Arnström and D. Axehill, "A unifying complexity certification framework for active-set methods for convex quadratic programming," *IEEE Transactions on Automatic Control*, vol. 67, no. 6, pp. 2758–2770, 2021.

[8] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 267–278, 2009.

[9] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, pp. 327–363, 2014.

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

[11] L. Wu and A. Bemporad, "A simple and fast coordinate-descent augmented-Lagrangian solver for model predictive control," *IEEE Transactions on Automatic Control*, vol. 68, no. 11, pp. 6860–6866, 2023.

[12] ——, "A construction-free coordinate-descent augmented-Lagrangian method for embedded linear MPC based on ARX models," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 9423–9428, 2023.

[13] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, "From linear to nonlinear MPC: Bridging the gap via the real-time iteration," *International Journal of Control*, vol. 93, no. 1, pp. 62–80, 2020.

[14] M. Diehl, "Real-time Optimization for Large Scale Nonlinear Processes," Ph.D. dissertation, Universität Heidelberg, Germany, 2001.

[15] M. Diehl, R. Findeisen, F. Allgöwer, H. G. Bock, and J. P. Schlöder, "Nominal stability of real-time iteration scheme for nonlinear model predictive control," *IEE Proceedings-Control Theory and Applications*, vol. 152, no. 3, pp. 296–308, 2005.

[16] L. Wu, K. Ganko, and R. D. Braatz, "Time-certified Input-constrained NMPC via Koopman operator," in *8th IFAC Conference on Nonlinear Model Predictive Control*, 2024, in press, arXiv:2401.04653.

[17] M. Korda and I. Mezić, "Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control," *Automatica*, vol. 93, pp. 149–160, 2018.

[18] S. J. Wright, *Primal-Dual Interior-Point Methods*. Philadelphia: SIAM, 1997.

[19] G. Frison and J. B. Jørgensen, "Efficient implementation of the Riccati recursion for solving linear-quadratic control problems," in *IEEE International Conference on Control Applications*, 2013, pp. 1117–1122.

[20] E. N. Lorenz, "Deterministic nonperiodic flow," *Journal of Atmospheric Sciences*, vol. 20, no. 2, pp. 130–141, 1963.