Constraint-Adaptive MPC for linear systems

Constraint-Adaptive MPC for linear systems: A system-theoretic framework for speeding up MPC through online constraint removal

1st S.A.N. Nouwens
Department of Mechanical Engineering
Eindhoven University of Technology
s.a.n.nouwens@tue.nl

2nd M.M. Paulides
Department of Electrical Engineering
Eindhoven University of Technology
Department of Radiotherapy
Erasmus UMC Cancer Institute

3rd W.P.M.H. Heemels
Department of Mechanical Engineering
Eindhoven University of Technology

***Abstract*—Reducing the computation time of model predictive control (MPC) is important, especially for systems constrained by many state constraints. In this paper, we propose a new online constraint removal framework for linear systems, for which we coin the term constraint-adaptive MPC (ca-MPC). In so-called *exact* ca-MPC, we adapt the imposed constraints by removing, at each time-step, a subset of the state constraints in order to reduce the computational complexity of the receding-horizon optimal control problem, while ensuring that the closed-loop behavior is *identical* to that of the original MPC law. We also propose an *approximate* ca-MPC scheme in which a further reduction of computation time can be accomplished by a tradeoff with closed-loop performance, while still preserving recursive feasibility, stability, and constraint satisfaction properties. The online constraint removal exploits fast backward and forward reachability computations combined with optimality properties.**

***Index Terms*—Model predictive control, linear systems, large-scale optimization problems, online constraint removal.**

I. Introduction

Model predictive control (MPC) is a successful control technology adopted in many application fields [13], [14], and is based on recursively solving a finite-horizon optimization problem online. Solving an optimization problem at each time-step can prohibit the real-time feasibility of the controller for computationally complex scenarios. This is particularly the case in applications requiring the control of systems with many state constraints, which is the setting studied in this paper.

Efforts to improve the computational aspects of MPC are commonplace in the literature with, amongst others, explicit MPC, model reduction, and tailored numerical solvers as prominent examples, see, e.g., [2]–[6], [9], [10], [19]. In particular, constraint removal techniques were developed to accelerate MPC for systems subject to many constraints.

This research is supported by KWF Kankerbestrijding and NWO Domain AES, as part of their joint strategic research programme: Technology for Oncology II. The collaboration project is co-funded by the PPP Allowance made available by Health∼Holland, Top Sector Life Sciences & Health, to stimulate public-private partnerships.

Constraint removal techniques can be roughly separated into offline [1], [18], [20], and online methods [11], [12]. Although of interest, offline methods can be prohibitively complex to compute and do not always enable real-time MPC, as constraints can only be removed if they are redundant for *all* feasible states. The fact that offline methods do not depend on and thus can not exploit the current state information might make them less effective than online constraint removal techniques.

In contrast, online techniques *can* exploit knowledge of the current state and possibly even more. As a result, online techniques have the potential to remove considerably more constraints compared to offline methods. For example, in [11] so-called regions of activity for each constraint are (approximately) computed that are based on the initial state for the MPC problem. Loosely speaking, the region of activity represents the set of initial states for which the particular constraint in consideration is active at the minimizer of the MPC problem. The online complexity of this method is reported to scale linearly in the number of constraints. However, approximating the region of activity can become intractable in scenarios with many constraints, as for each constraint either an ellipsoidal or hypercube outer approximation must be computed that depends on all other constraints. Alternatively in [12], a Lyapunov-based approach is proposed, assuming the cost function of the MPC problem is a Lyapunov function. Here, for each inequality constraint, the cost function is minimized assuming the particular inequality is active in the sense of equality. This leads to the minimum cost function value for which the particular constraint can be active. Next, when the MPC control law is running, the value of the Lyapunov function for the current state is compared to the pre-computed values for all constraints. If this value for the current state is lower than the corresponding value for an inequality constraint, it can be removed permanently from the MPC problem.

In this work, we will present a new online constraint removal framework for linear systems, called *constraint-adaptive* MPC (ca-MPC). We present both *exact* and *ap-*

proximate ca-MPC strategies. Crucially, in exact ca-MPC, the closed-loop behavior of the resulting accelerated MPC feedback law is *identical* to that of the original MPC feedback law. In approximate ca-MPC a further reduction of computation time can be accomplished compared to exact ca-MPC due to a tradeoff with closed-loop performance (as the closed-loop behavior is no longer identical in approximate ca-MPC). However, in approximate ca-MPC crucial properties such as recursive feasibility, stability, and constraint satisfaction can be preserved by design. Both strategies exploit system-theoretic properties, such as reachability and optimality, in a computationally effective manner. The method presented in this paper extends our preliminary work in [16] and [17] in which only initial ideas were presented (without any technical proofs). The current work formalizes these initial ideas in a complete framework and specifies also the technical underlying results and their rigorous proofs. A new numerical case study, extending the earlier one, is provided as well, which shows a two-order reduction in computational time of the ca-MPC scheme compared to the original MPC scheme, while still having identical closed-loop behavior. We also show how the preliminary results of [16] can be seen as a special case, *approximate* ca-MPC framework presented in this paper (see Section V).

II. System and MPC setup

In this paper, we consider plants that can be described by a discrete-time linear time-invariant (LTI) system

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k, \tag{1}$$

although several ideas also apply to nonlinear and time-varying plants, see, e.g., [16]. In (1), $\boldsymbol{x}_k \in \mathbb{R}^n$ and $\boldsymbol{u}_k \in \mathbb{R}^m$ denote the plant states and the inputs, respectively, at discrete time $k \in \mathbb{N}$. Furthermore, $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{B} \in \mathbb{R}^{n\times m}$. The system (1) is subject to polyhedral state and input constraints given for $k \in \mathbb{N}$ by

$$\boldsymbol{x}_k \in \mathbb{X} := \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{c}_j\boldsymbol{x} \le b_j, \text{ for } j \in \mathbb{N}_{[1,n_x]}\}, \tag{2a}$$
$$\boldsymbol{u}_k \in \mathbb{U} := \{\boldsymbol{u} \in \mathbb{R}^m \mid \boldsymbol{g}_j\boldsymbol{u} \le h_j, \text{ for } j \in \mathbb{N}_{[1,n_u]}\}. \tag{2b}$$

Here, $\mathbb{X}$ and $\mathbb{U}$ are assumed to be non-empty polyhedral sets with $\boldsymbol{c}_j \in \mathbb{R}^{1\times n}$, $\boldsymbol{g}_j \in \mathbb{R}^{1\times m}$, $b_j \in \mathbb{R}$, and $h_j \in \mathbb{R}$. In this paper, we study systems that are constrained by many state constraints, i.e., $n_x \gg 1$.

A. MPC setup

Based on the system dynamics (1) and constraints (2), a common MPC setup, given state $\boldsymbol{x}_k$ at time $k \in \mathbb{N}$, is

$$\begin{aligned}
\underset{\boldsymbol{X}_k,\ \boldsymbol{U}_k}{\text{minimize}} \quad & J(\boldsymbol{X}_k, \boldsymbol{U}_k), && \text{(3a)}\\
\text{subject to} \quad & \boldsymbol{X}_k = \boldsymbol{\Phi}\boldsymbol{x}_k + \boldsymbol{\Gamma}\boldsymbol{U}_k, && \text{(3b)}\\
& \boldsymbol{X}_k \in \mathcal{X} := \textstyle\prod_{i=1}^N \mathbb{X}_i, && \text{(3c)}\\
& \boldsymbol{U}_k \in \mathcal{U} := \mathbb{U}^N, && \text{(3d)}
\end{aligned}$$

where

$$J(\boldsymbol{X}_k, \boldsymbol{U}_k) := \ell_T(\boldsymbol{x}_{N|k}) + \textstyle\sum_{i=0}^{N-1} \ell(\boldsymbol{x}_{i|k}, \boldsymbol{u}_{i|k}), \tag{3e}$$
$$\boldsymbol{X}_k := [\boldsymbol{x}_{1|k}^\top \cdots \boldsymbol{x}_{N|k}^\top]^\top, \boldsymbol{U}_k := [\boldsymbol{u}_{0|k}^\top \cdots \boldsymbol{u}_{N-1|k}^\top]^\top, \tag{3f}$$
$$\mathbb{X}_i := \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{c}_{i,j}\boldsymbol{x} \le b_{i,j}, \text{ for } j \in \mathbb{N}_{[1,n_{x_i}]}\}, \tag{3g}$$
$$\boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{A}^2 \\ \vdots \\ \boldsymbol{A}^N \end{bmatrix}, \qquad \boldsymbol{\Gamma} := \begin{bmatrix} \boldsymbol{B} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{A}\boldsymbol{B} & \boldsymbol{B} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}^{N-1}\boldsymbol{B} & \boldsymbol{A}^{N-2}\boldsymbol{B} & \cdots & \boldsymbol{B} \end{bmatrix}. \tag{3h}$$

Here, ℓ, ℓ_T, $\boldsymbol{x}_{i|k}$, $\boldsymbol{u}_{i|k}$, and $\mathbb{X}_i$ denote the stage cost, the terminal cost, the predicted state, the predicted input, and the state constraint set at predicted time $i \in \mathbb{N}_{[1,N]} := \{1, 2, \cdots, N\}$ made at time $k \in \mathbb{N}$, respectively. The state constraints depend on i for generality and to facilitate compact notation using the Cartesian product. Typically, $\mathbb{X}_i$ is chosen as $\mathbb{X}_i = \mathbb{X} \subset \mathbb{R}^n$ for $i \in \mathbb{N}_{[1,N-1]}$ and $\mathbb{X}_N = \mathbb{X}_T \subseteq \mathbb{X}$, where $\mathbb{X}_T$ denotes a suitable controlled invariant terminal set [14]. The $i|k$ subscript is used to denote the i-th prediction at time k.

For the optimization problem (3), we denote the set of feasible input sequences parameterized by $\boldsymbol{x}_k$ as

$$\mathcal{U}_f(\boldsymbol{x}_k) := \{\boldsymbol{U}_k \in \mathcal{U} \mid \text{(3b)}, \text{ (3c)}\}, \tag{4}$$

and the set of feasible states by $\mathbb{X}_f := \{\boldsymbol{x} \in \mathbb{X} \mid \mathcal{U}_f(\boldsymbol{x}) \neq \emptyset\}$. Under suitable assumptions on ℓ, ℓ_T, $\mathcal{U}$, and $\mathcal{X}$, e.g., ℓ and ℓ_T being continuous and $\mathcal{X}$ being closed and $\mathcal{U}$ being compact [14], a minimizer of (3) exists for all $\boldsymbol{x}_k \in \mathbb{X}_f$ and we denote by $\boldsymbol{U}_k^\star := [\boldsymbol{u}_{0|k}^{\star\top} \ \cdots \ \boldsymbol{u}_{N-1|k}^{\star\top}]^\top$ a particular one at time $k \in \mathbb{N}$ for state $\boldsymbol{x}_k$, i.e.,

$$\boldsymbol{U}_k^\star \in \mathcal{U}^\star(\boldsymbol{x}_k) := \underset{\boldsymbol{U}_k \in \mathcal{U}_f(\boldsymbol{x}_k)}{\arg\min}\ \bar{J}(\boldsymbol{x}_k, \boldsymbol{U}_k), \tag{5}$$

where $\bar{J}(\boldsymbol{x}_k, \boldsymbol{U}_k) := J(\boldsymbol{\Phi}\boldsymbol{x}_k + \boldsymbol{\Gamma}\boldsymbol{U}_k, \boldsymbol{U}_k)$. The set of all optimal predicted state sequences corresponding to $\mathcal{U}^\star(\boldsymbol{x}_k)$ is denoted by

$$\mathcal{X}^\star(\boldsymbol{x}_k) := \boldsymbol{\Phi}\boldsymbol{x}_k + \boldsymbol{\Gamma}\mathcal{U}^\star(\boldsymbol{x}_k), \tag{6}$$

where a particular one is given by $\boldsymbol{X}_k^\star = \boldsymbol{\Phi}\boldsymbol{x}_k + \boldsymbol{\Gamma}\boldsymbol{U}_k^\star \in \mathcal{X}^\star(\boldsymbol{x}_k)$. Using a receding horizon implementation, the MPC problem (3) is turned into a feedback law $K_{\text{MPC}} : \mathbb{X}_f \to \mathbb{U}$ by applying the first computed input in $\boldsymbol{U}_k^\star$ on the real plant (1), i.e., $\boldsymbol{u}_k := K_{\text{MPC}}(\boldsymbol{x}_k) := \boldsymbol{u}_{0|k}^\star$, $k \in \mathbb{N}$.

B. Reduced MPC problem

To address the problem of removing redundant state constraints from (3), we introduce the reduced MPC problem, where the original constraints set $\mathcal{X}$ is replaced by a (state-dependent) reduced constraint set denoted by $\mathcal{X}^{\text{red}}(\mathcal{A}(\boldsymbol{x}_k)) = \prod_{i=1}^N \mathbb{X}_i^{\text{red}}(\mathbb{A}_i(\boldsymbol{x}_k))$, leading to

$$\begin{aligned}
\underset{\boldsymbol{X}_k,\ \boldsymbol{U}_k}{\text{minimize}} \quad & J(\boldsymbol{X}_k, \boldsymbol{U}_k), && \text{(7a)}\\
\text{subject to} \quad & \text{(3b), (3d)}, && \text{(7b)}\\
& \boldsymbol{X}_k \in \mathcal{X}^{\text{red}}(\mathcal{A}(\boldsymbol{x}_k)). && \text{(7c)}
\end{aligned}$$

The reduced constraint set $\mathcal{X}^{\text{red}}(\mathcal{A}(\boldsymbol{x}_k))$ is described by an index-set $\mathcal{A}(\boldsymbol{x}_k) = \prod_{i=1}^N \mathbb{A}_i(\boldsymbol{x}_k)$, such that

$$\mathbb{X}_i^{\text{red}}(\mathbb{A}_i) := \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{c}_{i,j}\boldsymbol{x} \le b_{i,j}, \text{ for } j \in \mathbb{A}_i\}. \tag{8}$$

Clearly, $\mathcal{X}^{\text{red}}$ is defined by a subset of the constraints in $\mathcal{X}$. Moreover, observe that the index set $\mathcal{A}(\boldsymbol{x}_k)$ depends on $\boldsymbol{x}_k$. Hence, our ca-MPC scheme requires the specification of the set-valued mapping $\mathbb{A}_i : \mathbb{X}_f \rightrightarrows \mathbb{N}_{[1,n_{x_i}]}$ for $i \in \mathbb{N}_{[1,N]}$. A formal problem formulation will be given in Section III. We use the notation $\rightrightarrows$ to indicate the set-valuedness of the maps of $\mathbb{A}_i$ in the sense that $\mathbb{A}_i(\boldsymbol{x}_k) \subseteq \mathbb{N}_{[1,n_{x_i}]}$ for $i \in \mathbb{N}_{[1,N]}$.

The reduced MPC problem (7) with the reduced constraint sets (8) gives rise to the set of minimizers

$$\mathcal{U}^{\text{red}\star}(\boldsymbol{x}_k, \mathcal{A}(\boldsymbol{x}_k)) := \underset{\boldsymbol{U}_k \in \mathcal{U}_f^{\text{red}}(\boldsymbol{x}_k, \mathcal{A}(\boldsymbol{x}_k))}{\arg\min} \ \bar{J}(\boldsymbol{x}_k, \boldsymbol{U}_k), \tag{9a}$$

$$\mathcal{U}_f^{\text{red}}(\boldsymbol{x}_k, \mathcal{A}(\boldsymbol{x}_k)) := \{\boldsymbol{U}_k \in \mathcal{U} \mid \text{(7b)} - \text{(7c)}\}, \tag{9b}$$

$$\mathbb{X}_f^{\text{red}} := \{\boldsymbol{x} \in \mathbb{X} \mid \mathcal{U}_f^{\text{red}}(\boldsymbol{x}_k, \mathcal{A}(\boldsymbol{x}_k)) \neq \emptyset\}. \tag{9c}$$

Again, similar to the original MPC problem, we introduce the set of "optimal" state trajectories

$$\mathcal{X}^{\text{red}\star}(\boldsymbol{x}_k, \mathcal{A}(\boldsymbol{x}_k)) := \boldsymbol{\Phi}\boldsymbol{x}_k + \boldsymbol{\Gamma}\mathcal{U}^{\text{red}\star}(\boldsymbol{x}_k, \mathcal{A}(\boldsymbol{x}_k)). \tag{10}$$

C. Preliminaries and notation

Given a set $\mathbb{V} \subset \mathbb{R}^n$, we denote the affine transformation of $\mathbb{V}$ with matrix $\boldsymbol{M} \in \mathbb{R}^{m\times n}$ and vector $\boldsymbol{b} \in \mathbb{R}^m$ by $\boldsymbol{M}\mathbb{V}+\boldsymbol{b} := \{\boldsymbol{M}\boldsymbol{v} + \boldsymbol{b} \in \mathbb{R}^m \mid \boldsymbol{v} \in \mathbb{V}\}$. We define an ellipsoidal set $\mathcal{E}(\boldsymbol{L}, \boldsymbol{q}) := \{\boldsymbol{x} \in \mathbb{R}^n \mid \|\boldsymbol{L}(\boldsymbol{x} - \boldsymbol{q})\|_2 \leq 1\}$, where $\boldsymbol{L} \in \mathbb{R}^{n\times n}$, $\boldsymbol{q} \in \mathbb{R}^n$, with $\|\boldsymbol{v}\|_2^2 := \sum_{i=1}^n v_i^2$ for $\boldsymbol{v} \in \mathbb{R}^n$. To project from sets defined for state (or input) trajectories to individual state vectors (or input vectors), we introduce the projection P_i of a set $\mathcal{V} \subseteq \mathbb{R}^{Nn}$ as

$$\begin{aligned}\mathrm{P}_i(\mathcal{V}) := \{\boldsymbol{v}_i \in \mathbb{R}^n \mid &\exists \boldsymbol{v}_j \in \mathbb{R}^n,\ j \in \mathbb{N}_{[1,N]}\backslash\{i\} \\ &\text{s.t. } [\boldsymbol{v}_1^\top \ \cdots \ \boldsymbol{v}_N^\top]^\top \in \mathcal{V}\},\end{aligned} \tag{11}$$

where $i \in \mathbb{N}_{[1,N]}$, assuming N is clear from the context. We define the normal cone of a set $\mathcal{V} \subset \mathbb{R}^d$ at $\boldsymbol{v} \in \mathcal{V}$ as $N_{\mathcal{V}}(\boldsymbol{v}) := \{\boldsymbol{p} \in \mathbb{R}^d \mid \boldsymbol{p}^\top(\boldsymbol{y} - \boldsymbol{v}) \leq 0 \text{ for all } \boldsymbol{y} \in \mathcal{V}\}$.

III. Exact constraint-adaptive MPC

We introduce now *exact ca-MPC*, in which the term "exact" refers to the property that the closed-loop system will *not* be changed when replacing (3) by the reduced MPC problem (7).

Definition 1. *A ca-MPC scheme based on* (7) *for given reduced constraint mappings* $\mathbb{A}_i : \mathbb{X}_f \rightrightarrows \mathbb{N}_{[1,n_{x_i}]}$, $i \in \mathbb{N}_{[1,N]}$, *is called exact, if* $\mathbb{X}_f = \mathbb{X}_f^{red}$ *and the set of minimizers of the reduced MPC problem* (7) *is the same as for the original MPC problem* (3) *for all* $\boldsymbol{x}_k \in \mathbb{X}_f$, *i.e.,*

$$\mathcal{U}^\star(\boldsymbol{x}_k) = \mathcal{U}^{red\star}(\boldsymbol{x}_k, \mathcal{A}(\boldsymbol{x}_k)). \tag{12}$$

As the minimizers are identical, *exact* ca-MPC trivially inherits all performance, stability, and constraint satisfaction properties from the original MPC problem.

The main problem considered in this paper can now be formulated as follows: Construct computationally tractable set-valued mappings $\mathbb{A}_i : \mathbb{X}_f \rightrightarrows \mathbb{N}_{[1,n_{x_i}]}$, with the number of constraints $\text{card}(\mathbb{A}_i(\boldsymbol{x}_k)) \ll n_{x_i}$ (if possible) for $i \in \mathbb{N}_{[1,N]}$ for all $\boldsymbol{x}_k \in \mathbb{X}_f$, such that, the resulting ca-MPC scheme is exact in the sense of Definition 1.

By creating simpler MPC problems, ca-MPC can accelerate both interior-point and active-set solvers. For interior-point methods, ca-MPC straightforwardly reduces the complexity of each Newton step, thereby accelerating the optimization problem. Additionally, ca-MPC is also a natural extension to active-set solvers, as the working and inactive constraint sets are already dynamically updated. By removing constraints a priori, determining which constraint has to be added to the working set is easier, as there are fewer constraints to evaluate.

The key concept of exact ca-MPC is inspired by the following basic observation. A ca-MPC scheme based on (7) for given mappings $\mathcal{A}$, is exact if and only if

$$\mathcal{X}^{\text{red}\star}(\boldsymbol{x}_k, \mathcal{A}(\boldsymbol{x}_k)) \subseteq \mathcal{X}, \tag{13}$$

for all $\boldsymbol{x}_k \in \mathbb{X}_f$. Indeed, when the optimal set of state trajectories satisfy all state constraints, we obtain the same set of minimizers. However, note that (13) is not suitable as a direct tool to design appropriate set-valued mappings $\mathcal{A}$, as (13) cannot be used in a constructive manner. To overcome this, we build upon the following theorem using outer approximations of $\mathcal{X}^{\text{red}\star}(\boldsymbol{x}_k, \mathcal{A}(\boldsymbol{x}_k))$ by a set $\mathcal{M}(\boldsymbol{x}_k)$ (not depending on $\mathcal{A}$).

Theorem 1. *Consider the original and reduced MPC optimization problem* (3) *and* (7)*, respectively, with* $\mathbb{A}_i : \mathbb{X}_f \rightrightarrows \mathbb{N}_{[1,n_{x_i}]}$ *and* $\mathcal{M} : \mathbb{X}_f \rightrightarrows \mathbb{R}^{Nn}$. *If for all* $\boldsymbol{x}_k \in \mathbb{X}_f$,

$$\mathcal{X}^{red\star}(\boldsymbol{x}_k, \mathcal{A}(\boldsymbol{x}_k)) \subseteq \mathcal{M}(\boldsymbol{x}_k), \tag{C1}$$

$$\mathcal{M}(\boldsymbol{x}_k) \cap \mathcal{X}^{red}(\mathcal{A}(\boldsymbol{x}_k)) \subseteq \mathcal{X}, \tag{C2}$$

then $\mathbb{X}_f = \mathbb{X}_f^{red}$ *and the ca-MPC scheme* (7) *is exact.*

Proof. *Let* $\boldsymbol{x}_k \in \mathbb{X}_f$ *be given. First, observe*

$$\mathcal{U}_f(\boldsymbol{x}_k) \subseteq \mathcal{U}_f^{red}(\boldsymbol{x}_k, \mathcal{A}(\boldsymbol{x}_k)), \tag{14}$$

which is immediate as (7) *uses a subset of the constraints of* (3)*. Hence,* $\mathbb{X}_f \subseteq \mathbb{X}_f^{red}$ *and for all* $\boldsymbol{x}_k \in \mathbb{X}_f$

$$\min_{\boldsymbol{U}_k \in \mathcal{U}_f(\boldsymbol{x}_k)} \bar{J}(\boldsymbol{x}_k, \boldsymbol{U}_k) \geq \min_{\boldsymbol{U}_k \in \mathcal{U}_f^{red}(\boldsymbol{x}_k, \mathcal{A}(\boldsymbol{x}_k))} \bar{J}(\boldsymbol{x}_k, \boldsymbol{U}_k). \tag{15}$$

Next, to show (12)*, take* $\boldsymbol{x}_k \in \mathbb{X}_f^{red}$*, for which each*

$$\boldsymbol{U}_k^{red\star} \in \mathcal{U}^{red\star}(\boldsymbol{x}_k, \mathcal{A}(\boldsymbol{x}_k)) \tag{16}$$

satisfies $\boldsymbol{U}_k^{red\star} \in \mathcal{U}$ *by* (7c) *and* $\boldsymbol{X}_k^{red\star} \in \mathcal{X}$ *by combining* (7b)*,* (C1)*, and* (C2)*. Hence, we obtain* $\mathcal{U}^{red\star}(\boldsymbol{x}_k, \mathcal{A}(\boldsymbol{x}_k)) \in \mathcal{U}_f(\boldsymbol{x}_k)$*, and by extension,* $\mathbb{X}_f^{red} \subseteq \mathbb{X}_f$*. Therefore, we have,* $\mathbb{X}_f^{red} = \mathbb{X}_f$ *and*

$$\min_{\boldsymbol{U}_k \in \mathcal{U}_f(\boldsymbol{x}_k)} \bar{J}(\boldsymbol{x}_k, \boldsymbol{U}_k) = \min_{\boldsymbol{U}_k \in \mathcal{U}_f^{red}(\boldsymbol{x}_k, \mathcal{A}(\boldsymbol{x}_k))} \bar{J}(\boldsymbol{x}_k, \boldsymbol{U}_k). \tag{17}$$

Combining (15)*,* (16)*, and* (17) *leads now to* (12). □

The main takeaway from Theorem 1 is that we concentrate the $\mathcal{A}$-dependence in $\mathcal{X}^{\text{red}}(\mathcal{A}(\boldsymbol{x}_k))$. The way we apply Theorem 1 is to first construct an $\mathcal{A}$-independent set $\mathcal{M}(\boldsymbol{x}_k)$ satisfying (C1), for all possible choices of $\mathcal{A}$. Once, this $\mathcal{M}(\boldsymbol{x}_k)$ is available, we can select, in the second step, the indices in $\mathcal{A}(\boldsymbol{x}_k)$ such that (C2) is satisfied. Intuitively, when $\mathcal{M}$ satisfies

(C1), then, its elements indicate which state constraints can be violated. Therefore, by adding these state constraint indices to $\mathcal{A}$, we ensure that these "at-risk" constraints are not violated. As a result, the optimal state trajectory $\mathcal{X}^{\text{red}\star}(\boldsymbol{x}_k, \mathcal{A}(\boldsymbol{x}_k))$ will satisfy all state constraints, and, hence, we obtain an exact ca-MPC scheme.

An important observation is that a smaller $\mathcal{M}$ has the potential to remove more constraints, see (C2). However, $\mathcal{M}$ still has to satisfy (C1). To this end, a useful extension to Theorem 1, is to utilize additional information besides the initial state, see, e.g., Sections IV-B and IV-C. For example, we can exclude certain state constraints from the removal process in order to find a smaller $\mathcal{M}$. To this end, we introduce a set of *fixed constraints* given by $\mathcal{F}(\boldsymbol{x}_k) := \prod_{i=1}^{N} \mathbb{F}_i(\boldsymbol{x}_k)$, where the index set $\mathcal{A}$ must satisfy $\mathcal{F}(\boldsymbol{x}_k) \subseteq \mathcal{A}(\boldsymbol{x}_k)$. We will omit the dependence of $\mathcal{M}$ on $\mathcal{F}$, as it will be clear from context.

Besides introducing fixed constraints to exclude constraints from the removal process, it is useful to construct $\mathcal{A}$ in a subtractive manner. By doing so, we can be conservative in removing constraints without losing exactness, but this will provide many computational benefits. To make this concrete, we parameterize $\mathcal{A}$ as

$$\mathbb{A}_i(\boldsymbol{x}_k) = \mathbb{N}_{[1,n_{x_i}]} \backslash \mathbb{I}_i(\boldsymbol{x}_k), \quad i \in \mathbb{N}_{[1,N]}, \tag{18}$$

where $\mathbb{I}_i : \mathbb{X}_f \rightrightarrows \mathbb{N}_{[1,n_{x_i}]}$, with $\mathbb{I}(\boldsymbol{x}_k) \cap \mathbb{F}_i(\boldsymbol{x}_k) = \emptyset$ for all $\boldsymbol{x}_k \in \mathbb{X}_f$, denotes the set of removed constraints from the MPC problem. We also define the compact notation $\mathcal{I}(\boldsymbol{x}_k) := \prod_{i=1}^{N} \mathbb{I}_i(\boldsymbol{x}_k)$.

Lemma 1. *Consider the original and reduced MPC problems* (3) *and* (7)*, respectively, and let* $\mathbb{F}_i : \mathbb{X}_f \rightrightarrows \mathbb{N}_{[1,n_{x_i}]}$ *and* $\mathcal{M} : \mathbb{X}_f \rightrightarrows \mathbb{R}^{Nn}$ *satisfying* (C1) *be given. Let the set-valued mapping* $\mathbb{I}_i : \mathbb{X}_f \rightrightarrows \mathbb{N}_{[1,n_{x_i}]}$*, with* $\mathbb{I}_i(\boldsymbol{x}_k) \cap \mathbb{F}_i(\boldsymbol{x}_k) = \emptyset$*,* $i \in \mathbb{N}_{[1,N]}$*, capture the removed state constraints, i.e.,* $\mathbb{A}_i(\boldsymbol{x}_k) = \mathbb{N}_{[1,n_{x_i}]} \backslash \mathbb{I}_i(\boldsymbol{x}_k), i \in \mathbb{N}_{[1,N]}$*. If for all* $\boldsymbol{x}_k \in \mathbb{X}_f$

$$\mathcal{M}(\boldsymbol{x}_k) \cap \mathcal{X}^{red}(\mathcal{I}(\boldsymbol{x}_k)) = \mathcal{M}(\boldsymbol{x}_k). \tag{C3}$$

Then, the resulting ca-MPC scheme is exact.

Proof. *To show* (C2)*, we start with the inclusion* $\mathcal{X} \cap \mathcal{M}(\boldsymbol{x}_k) \subseteq \mathcal{X}$*. Since* $\mathbb{A}_i \cup \mathbb{I}_i = \mathbb{N}_{[1,n_{x_i}]}$*,* $i \in \mathbb{N}_{[1,N]}$*, we obtain* $\mathcal{X}^{red}(\mathcal{A}(\boldsymbol{x}_k)) \cap \mathcal{X}^{red}(\mathcal{I}(\boldsymbol{x}_k)) \cap \mathcal{M}(\boldsymbol{x}_k) \subseteq \mathcal{X}$*. Exploiting* (C3) *gives* $\mathcal{X}^{red}(\mathcal{A}(\boldsymbol{x}_k)) \cap \mathcal{M}(\boldsymbol{x}_k) \subseteq \mathcal{X}$*, which is equivalent to* (C2)*.* □

The result of Lemma 1 is instrumental as it provides a computationally efficient method to remove constraints from the reduced MPC problem. For example, (C3) can be evaluated for each constraint independently. As a result, we can also consider projections of $\mathcal{M}$, i.e.,

$$\mathrm{P}_i(\mathcal{M}(\boldsymbol{x}_k)) \cap \mathbb{X}_i^{\text{red}}(\mathbb{I}_i(\boldsymbol{x}_k)) = \mathrm{P}_i(\mathcal{M}(\boldsymbol{x}_k)), \tag{19}$$

for all $i \in \mathbb{N}_{[1,N]}$, is equivalent to (C3). This observation will lead to an efficient ca-MPC implementation, as we will see in the next section.

IV. Proposed exact ca-MPC implementation

In this section, we will make the exact ca-MPC scheme concrete by introducing three sets $\mathcal{M}^{(1)}$, $\mathcal{M}^{(2)}$, and $\mathcal{M}^{(3)}$ to construct $\mathcal{M}$ as $\mathcal{M}^{(1)} \cap \mathcal{M}^{(2)} \cap \mathcal{M}^{(3)}$. Hereafter, we will provide a concrete outline on how $\mathcal{M}$ is used to compute the mapping $\mathcal{A}$.

A. Forward reachable set

As all state trajectories start at $\boldsymbol{x}_k$ and satisfy the system dynamics and input constraints, we can use the input-constrained forward reachable set to construct $\mathcal{M}^{(1)}(\boldsymbol{x}_k)$. To this end, we introduce the *input-constrained* forward reachable set $\overrightarrow{\mathcal{H}}(\boldsymbol{x}_k, \mathbb{U}) := \prod_{i=1}^{N} \overrightarrow{\mathbb{H}}_i(\boldsymbol{x}_k, \mathbb{U}) \subset \mathbb{R}^{Nn}$, which can be computed by the recursion

$$\overrightarrow{\mathbb{H}}_{i+1}(\boldsymbol{x}_k, \mathbb{U}) := \boldsymbol{A} \overrightarrow{\mathbb{H}}_i(\boldsymbol{x}_k, \mathbb{U}) + \boldsymbol{B}\mathbb{U}, \tag{20a}$$

$$\overrightarrow{\mathbb{H}}_0(\boldsymbol{x}_k, \mathbb{U}) = \{\boldsymbol{x}_k\}. \tag{20b}$$

To compute the state-dependent forward reachable set in a real-time setting we define $\mathcal{M}^{(1)}(\boldsymbol{x}_k)$ as

$$\mathcal{M}^{(1)}(\boldsymbol{x}_k) = \prod_{i=1}^{N} \mathcal{E}(\boldsymbol{L}_{i,1}, \boldsymbol{q}_{i,1}) + \boldsymbol{A}^i \boldsymbol{x}_k, \tag{21}$$

where $\mathcal{E}(\boldsymbol{L}_{i,1}, \boldsymbol{q}_{i,1}) \supset \overrightarrow{\mathbb{H}}_i(\boldsymbol{0}, \mathbb{U})$, which can be computed using [7], [8]. The construction (21) solves two problems. First, note that the forward reachable set can be decomposed as $\overrightarrow{\mathbb{H}}_i(\boldsymbol{x}_k, \mathbb{U}) = \boldsymbol{A}^i \boldsymbol{x}_k + \overrightarrow{\mathbb{H}}_i(\boldsymbol{0}, \mathbb{U})$ for linear systems. Hence, we can compute the forward reachable set once offline for a zero initial state, and shift the result by the free state response $\boldsymbol{A}^i \boldsymbol{x}_k$. Second, we use ellipsoids that have a fixed complexity to outer approximate the true forward reachable set, which can be arbitrarily complex.

B. Backward reachable set

Given the forward reachable set, a natural extension is to exploit backward reachability too. Indeed, when we choose $\mathcal{F}(\boldsymbol{x}_k) = \prod_{i=1}^{N-1} \emptyset \times \mathbb{N}_{[1,n_{x_N}]}$, all optimal state trajectories that can result from (7) must end in the terminal set. To this end, we introduce the *input- and terminal-state-constrained* backward reachable set for (1) and (2b), i.e., $\overleftarrow{\mathcal{H}}(\mathbb{X}_N, \mathbb{U}) := \prod_{i=1}^{N} \overleftarrow{\mathbb{H}}_i(\mathbb{X}_N, \mathbb{U}) \subset \mathbb{R}^{Nn}$, which can be computed using the recursion

$$\overleftarrow{\mathbb{H}}_{i-1}(\mathbb{X}_N, \mathbb{U}) := \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \in \overleftarrow{\mathbb{H}}_i(\mathbb{X}_N, \mathbb{U}) \text{ for some } \boldsymbol{u} \in \mathbb{U}\}, \ i \in \mathbb{N}_{[2,N]} \tag{22a}$$

$$\overleftarrow{\mathbb{H}}_N(\mathbb{X}_N, \mathbb{U}) = \mathbb{X}_N. \tag{22b}$$

Note that the backwards reachable set is not state-dependent but can be prohibitively complex, similar to the forward reachable set. Hereto, we define $\mathcal{M}^{(2)}$ as

$$\mathcal{M}^{(2)}(\boldsymbol{x}_k) = \prod_{i=1}^{N} \mathcal{E}(\boldsymbol{L}_{i,2}, \boldsymbol{q}_{i,2}), \tag{23}$$

where $\mathcal{E}(\boldsymbol{L}_{i,2}, \boldsymbol{q}_{i,2}) \subseteq \overleftarrow{\mathbb{H}}_i(\mathbb{X}_N, \mathbb{U})$ to manage online complexity.

C. First-order optimality set

The third set, $\mathcal{M}^{(3)}$, exploits the optimization based nature of MPC. For example, a feasible solution to (3) upper bounds the cost function, thereby bounding the minimizer [17]. One method to obtain a feasible solution, is by extending the minimizer $\boldsymbol{U}^\star_{k-1}$ obtained at time $k-1$ (as is common when using terminal set and cost methods, see [14]). In this section, we introduce a more advanced notion of this concept by exploiting the property that optimal state trajectories satisfy first-order optimality conditions, which, in combination with a feasible solution to (3), yields a smaller set.

We start by formalizing the first-order optimality conditions that optimal input trajectories satisfy, under differentiability of $\bar{J}$ with respect to $\boldsymbol{U}$,

$$-\nabla_{\boldsymbol{U}} \bar{J}(\boldsymbol{x}_k, \boldsymbol{U}_k) \in N_{\mathcal{U}_f^{\text{red}}(\boldsymbol{x}_k, \mathcal{A}(\boldsymbol{x}_k))}(\boldsymbol{U}^\star), \tag{24}$$

where $N_{\mathcal{U}_f^{\text{red}}(\boldsymbol{x}_k, \mathcal{A}(\boldsymbol{x}_k))}(\boldsymbol{U}^\star)$ denotes the normal cone of $\mathcal{U}_f^{\text{red}}(\boldsymbol{x}_k, \mathcal{A}(\boldsymbol{x}_k))$ at $\boldsymbol{U}^\star$, as is commonly seen in stationary conditions for optimization problems [15, Sec. 12.7]. Observe that (24) is not a constructive condition to design $\mathcal{M}^{(3)}$, as it uses $\mathcal{A}$ and $\boldsymbol{U}^\star$ in its definition. To get around this issue, we introduce the set of input trajectories for which a convex constraint set exists, such that, first-order optimality conditions can be satisfied given a feasible input trajectory $\tilde{\boldsymbol{U}}_k \in \mathcal{U}_f(\boldsymbol{x}_k)$, see Figure 1. Note that the set of all convex sets that include the feasible input trajectory is guaranteed to include $\mathcal{U}_f^{\text{red}}(\boldsymbol{x}_k, \mathcal{A}(\boldsymbol{x}_k))$. Hereto, we define $\mathcal{J}(\boldsymbol{x}_k, \tilde{\boldsymbol{U}}_k) \subset \mathbb{R}^{Nm}$,

$$\begin{aligned}\mathcal{J}(\boldsymbol{x}_k, \tilde{\boldsymbol{U}}_k) := \{\boldsymbol{U} \in \mathbb{R}^{Nm} \mid \exists \text{convex } \mathcal{S} \subseteq \mathbb{R}^{Nm}, \\ \text{s.t. } \boldsymbol{U}, \tilde{\boldsymbol{U}}_k \in \mathcal{S}, \text{ and } -\nabla_{\boldsymbol{U}} \bar{J}(\boldsymbol{x}_k, \boldsymbol{U}_k) \in N_{\mathcal{S}}(\boldsymbol{U})\}.\end{aligned} \tag{25}$$

Given (25), we obtain $\mathcal{M}^{(3)}(\boldsymbol{x}_k) = \boldsymbol{\Phi}\boldsymbol{x}_k + \boldsymbol{\Gamma}\mathcal{J}(\boldsymbol{x}_k, \tilde{\boldsymbol{U}}_k)$.

While (25) seems complex at first sight, it has an elegant solution for quadratic cost functions. To show this, we specify a quadratic cost function J as in (3e) with $\ell(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^\top \boldsymbol{R}\boldsymbol{u}$, $\ell_T(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{P}\boldsymbol{x}$, where $\boldsymbol{Q}$ and $\boldsymbol{P}$ denote positive semi-definite matrices and $\boldsymbol{R}$ is positive definite. When substituting the quadratic cost and system dynamics into (3e), it is easy to see that an equivalent formulation for the cost is

$$\bar{J}(\boldsymbol{x}_k, \boldsymbol{U}_k) = \|\boldsymbol{G}(\boldsymbol{U}_k - \boldsymbol{q}(\boldsymbol{x}_k))\|_2^2 + r(\boldsymbol{x}_k) \tag{26}$$

with $\boldsymbol{G} \in \mathbb{R}^{Nm \times Nm}$ and linear maps $\boldsymbol{q} : \mathbb{X}_f \to \mathbb{R}^{Nm}$, $r : \mathbb{X}_f \to \mathbb{R}$ that depend on the system dynamics, the current state, and $\boldsymbol{Q}$, $\boldsymbol{P}$, $\boldsymbol{R}$. Observe that $\boldsymbol{q}(\boldsymbol{x}_k)$ denotes the unconstrained minimizer of the MPC problem, which is the unique solution to $\nabla_{\boldsymbol{U}} \bar{J}(\boldsymbol{x}_k, \boldsymbol{q}(\boldsymbol{x}_k)) = \boldsymbol{0}$. Using (26), $\mathcal{J}(\boldsymbol{x}_k, \tilde{\boldsymbol{U}}_k)$ is given by the ellipsoid

$$\begin{aligned}\mathcal{J}(\boldsymbol{x}_k, \tilde{\boldsymbol{U}}_k) = \{\boldsymbol{U} \in \mathbb{R}^{Nm} \mid \\ \|\boldsymbol{G}\big(\boldsymbol{U} - \frac{1}{2}(\tilde{\boldsymbol{U}}_k + \boldsymbol{q}(\boldsymbol{x}_k))\big)\|_2 \leq \frac{1}{2}\|\boldsymbol{G}(\tilde{\boldsymbol{U}}_k - \boldsymbol{q}(\boldsymbol{x}_k))\|_2\}.\end{aligned} \tag{27}$$

The proof of this result is provided in Appendix A, and a schematic illustration is in Figure 1.

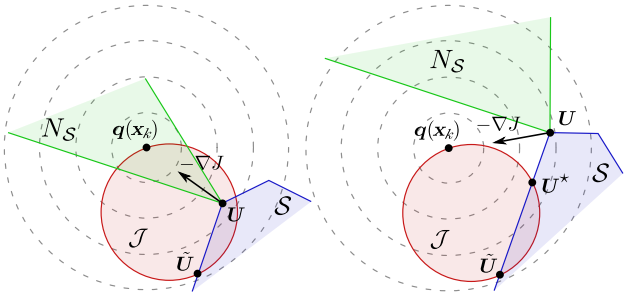

Fig. 1: Two illustrations of (25) for a quadratic cost function at different points $\boldsymbol{U}$. The dashed lines denote level sets of the cost function and $\boldsymbol{q}(\boldsymbol{x}_k)$ denotes the unconstrained minimizer, where $\nabla_{\boldsymbol{U}} \bar{J}(\boldsymbol{x}_k, \boldsymbol{q}(\boldsymbol{x}_k)) = \boldsymbol{0}$. Note that $\mathcal{J}$ is significantly smaller than the level set of $\bar{J}$ at $\tilde{\boldsymbol{U}}$. *Left*: $\boldsymbol{U} \in \mathcal{J}$, as there exists a set $\mathcal{S}$ such that $N_{\mathcal{S}}$ includes $-\nabla_{\boldsymbol{U}} \bar{J}$. In fact, $\boldsymbol{U}$ is the minimizer. *Right*: $\boldsymbol{U} \notin \mathcal{J}$, as there does not exist a set $\mathcal{S}$ such that $-\nabla_{\boldsymbol{U}} \bar{J} \in N_{\mathcal{S}}$. As expected, we observe that the minimizer for $\mathcal{S}$ satisfies $\boldsymbol{U}^\star \in \mathcal{J}$.

D. Integration into an exact ca-MPC scheme

Now we will present how the sets $\mathcal{M}^{(i)}$, $i = \{1, 2, 3\}$, are used to construct an exact ca-MPC scheme. Recall that $\mathcal{M}^{(2)}$ requires $\mathcal{F}(\boldsymbol{x}_k) = \prod_{i=1}^{N-1} \emptyset \times \mathbb{N}_{[1, n_{x_N}]}$ and $\mathcal{M}^{(3)}$ requires availability of an $\tilde{\boldsymbol{U}}_k \in \mathcal{U}_f(\boldsymbol{x}_k)$. As $\mathcal{M}$ is defined as the intersection of ellipsoids, we utilize (C3) to construct $\mathcal{I}(\boldsymbol{x}_k)$ as

$$\mathbb{I}_i(\boldsymbol{x}_k) = \mathbb{I}_i^{(1)}(\boldsymbol{x}_k) \cup \mathbb{I}_i^{(2)}(\boldsymbol{x}_k) \cup \mathbb{I}_i^{(3)}(\boldsymbol{x}_k), \tag{28a}$$

$$\begin{aligned}\mathbb{I}_i^{(l)}(\boldsymbol{x}_k) = \{j \in \mathbb{N}_{[1, n_{x_i}]} \backslash \mathbb{F}_i(\boldsymbol{x}_k) \mid \\ \|\boldsymbol{c}_{i,j} \boldsymbol{L}_{i,l}^{-1}\|_2 \leq |b_{i,j} - \boldsymbol{c}_{i,j} \boldsymbol{q}_{i,l}|\},\end{aligned} \tag{28b}$$

where $\mathcal{E}(\boldsymbol{L}_{i,l}, \boldsymbol{q}_{i,l}) = \mathrm{P}_i(\mathcal{M}^{(l)}(\boldsymbol{x}_k))$. Observe that for each $l \in \{1, 2, 3\}$, it holds that

$$\mathrm{P}_i(\mathcal{M}^{(l)}(\boldsymbol{x}_k)) \cap \mathbb{X}_i(\mathbb{I}_i^{(l)}(\boldsymbol{x}_k)) = \mathrm{P}_i(\mathcal{M}^{(l)}(\boldsymbol{x}_k)). \tag{29}$$

Hence, (28) implies (C3). The derivation of (28b) is given in Appendix A. Crucially, $\boldsymbol{L}_{i,l}$ is not state-dependent (up to a constant), therefore, $\|\boldsymbol{c}_{i,j} \boldsymbol{L}_{i,l}^{-1}\|_2$ can be pre-computed (up to a constant) for all i, j, l. As a result, the ellipsoidal description of $\mathcal{M}$ allows for the computationally efficient of $\mathcal{I}(\boldsymbol{x}_k)$ using (28). In fact, (28b) only requires one inner product, a scalar absolute value, and a comparison.

For the remainder of this section, we present an overview of the integrated ca-MPC scheme in Algorithm 1. In Algorithm 1, we use $\leftarrow$ to denote "compute using". First, in lines 1-5, in an offline setting, we pre-compute the forward and backward reachable sets and corresponding ellipsoidal outer approximations. Second, in an online setting, we start by computing the first-order optimality set in line 8 based on the measured state $\boldsymbol{x}_k$ and generated feasible input sequence $\tilde{\boldsymbol{U}}_k$. Next, in lines 9-13, we perform the constraint removal using the forward and backward reachable sets and first-order optimality set. Finally, we compute the minimizer of the resulting reduced MPC

problem (line 14) and apply $\boldsymbol{u}_k = \boldsymbol{u}_{0|k}^{\text{red}\star}$ to the plant (line 15).

Algorithm 1 Implementation of exact ca-MPC

1: $k = 0,\ N \in \mathbb{N}$
2: $\overrightarrow{\mathbb{H}}_i(\mathbf{0}, \mathbb{U}) \leftarrow$ (20), $i \in \mathbb{N}_{[1,N]}$
3: $\overleftarrow{\mathbb{H}}_i(\mathbb{X}_N, \mathbb{U}) \leftarrow$ (22), $i \in \mathbb{N}_{[1,N-1]}$
4: $\boldsymbol{L}_{i,1},\ \boldsymbol{q}_{i,1} \leftarrow \overrightarrow{\mathbb{H}}_i(\mathbf{0}, \mathbb{U})$, s.t. (20), $i \in \mathbb{N}_{[1,N]}$
5: $\boldsymbol{L}_{i,2},\ \boldsymbol{q}_{i,2} \leftarrow \overleftarrow{\mathbb{H}}_i(\mathbb{X}_N, \mathbb{U})$, s.t. (22), $i \in \mathbb{N}_{[1,N-1]}$
6: **while** TRUE **do**
7: MEASURE $\boldsymbol{x}_k \in \mathbb{X}_f$
8: $\mathcal{J}(\boldsymbol{x}_k, \tilde{\boldsymbol{U}}_k) \leftarrow$ (25) WITH $\tilde{\boldsymbol{U}}_k \in \mathcal{U}_f(\boldsymbol{x}_k)$
9: **for** $i = 1, 2, \cdots, N-1$ **do**
10: $\mathbb{I}_i^{(1)} \leftarrow$ (28) WITH $\mathcal{E}(\boldsymbol{L}_{i,1}, \boldsymbol{q}_{i,1}) + \boldsymbol{A}^i \boldsymbol{x}_k$
11: $\mathbb{I}_i^{(2)} \leftarrow$ (28) WITH $\mathcal{E}(\boldsymbol{L}_{i,2}, \boldsymbol{q}_{i,2})$
12: $\mathbb{I}_i^{(3)} \leftarrow$ (28) WITH $\mathrm{P}_i(\boldsymbol{\Gamma}\mathcal{J}(\boldsymbol{x}_k, \tilde{\boldsymbol{U}}_k)) + \boldsymbol{A}^i \boldsymbol{x}_k$
13: $\mathbb{A}_i \leftarrow \mathbb{N}_{[1,n_{x_i}]} \backslash (\mathbb{I}_i^{(1)} \cup \mathbb{I}_i^{(2)} \cup \mathbb{I}_i^{(3)})$
14: $\boldsymbol{U}_k^{\text{red}\star} \leftarrow$ (9a)
15: APPLY CONTROL INPUT $\boldsymbol{u}_k \leftarrow \boldsymbol{u}_{0|k}^{\text{red}\star}$
16: $k \leftarrow k + 1$

V. APPROXIMATE CA-MPC

The exact ca-MPC method introduced in Sections III and IV is highly effective in the sense that the online complexity of computing $\mathcal{A}(\boldsymbol{x}_k)$ is low, while significant reductions in the number of constraints can be achieved. A relaxation, *approximate ca-MPC*, aims to remove more constraints from the MPC problem. Approximate ca-MPC will maintain the constraint satisfaction (and stability), but will no longer be exact.

Based on (C2), we observe that shrinking $\mathcal{M}$ can result in the removal of more state constraints. One way to realize this, is by designing a smaller $\mathcal{M}$ that satisfies (C1). Hereto, we impose *additional* input-constraints that allow us to design a tighter $\mathcal{M}$. To this end, we augment the original MPC problem (3) with the input constraint $\boldsymbol{U}_k \in \tilde{\boldsymbol{U}}_k + \delta\mathbb{U}^N$, where $\tilde{\boldsymbol{U}}_k \in \mathcal{U}_f(\boldsymbol{x}_k)$ is a feasible input sequence, and $\delta\mathbb{U} \subset \mathbb{R}^m$ with $\mathbf{0} \in \delta\mathbb{U}$ is a set limiting the input sequence $\boldsymbol{U}_k$ to be "close" to $\tilde{\boldsymbol{U}}_k$. By constraining the minimizer to be close to a feasible input trajectory, we obtain "smaller" forward and backward reachable sets that can lead to the removal of more constraints from the MPC problem. We will consider a particular choice of $\tilde{\boldsymbol{U}}_k$, namely, $\tilde{\boldsymbol{U}}_k = [\boldsymbol{u}_{1|k-1}^{\star\top} \ \cdots \ \boldsymbol{u}_{N-1|k-1}^{\star\top} \ \tilde{\boldsymbol{u}}^\top(\boldsymbol{x}_{N|k-1}^\star)]^\top$, where $\tilde{\boldsymbol{u}} : \mathbb{X}_N \to \mathbb{U}$ denotes an auxiliary feedback law that renders $\mathbb{X}_N$ positively invariant, i.e., $\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\tilde{\boldsymbol{u}}(\boldsymbol{x}) \in \mathbb{X}_N$ for all $\boldsymbol{x} \in \mathbb{X}_N$. The MPC problem corresponding to this new setup is given by

$$\begin{aligned} \underset{\boldsymbol{X}_k,\ \boldsymbol{U}_k}{\text{minimize}} \quad & J(\boldsymbol{X}_k, \boldsymbol{U}_k), && \text{(30a)} \\ \text{subject to} \quad & \text{(3b), (3d)} && \text{(30b)} \\ & \boldsymbol{U}_k \in \mathcal{U} \cap (\tilde{\boldsymbol{U}}_k + \delta\mathbb{U}^N). && \text{(30c)} \end{aligned}$$

Note that depending on $\tilde{\boldsymbol{u}} : \mathbb{X}_N \to \mathbb{U}$, the closed-loop stability of (30) can be unchanged with respect to (3), as the standard stability proof based on a terminal set and cost [14], is still applicable. Interestingly, the approximate ca-MPC scheme is exact with respect to (30), but not (3), hence, its properties can be analyzed using (30). Last, note that the size of $\delta\mathbb{U}$ is a tuning knob (and can even depend on time k and prediction step i) that can trade-off potential performance loss to increased constraint removal, and thus computational benefits.

VI. NUMERICAL EXAMPLE

We will use a double integrator system

$$\boldsymbol{x}_{k+1} = \left[\begin{smallmatrix} 1 & 0.1 \\ 0 & 1 \end{smallmatrix}\right] \boldsymbol{x}_k + \left[\begin{smallmatrix} 0.005 \\ 0.1 \end{smallmatrix}\right] u_k, \tag{31}$$

as the two-dimensional state allows for convenient visualization. The input constraints are $\mathbb{U} = \{u \in \mathbb{R} \mid |u| \leq 1\}$. To obtain an example with a large number of linear state constraints as in (3), we approximate two quadratic constraints with linear inequalities, i.e.,

$$\begin{aligned} \mathbb{X}_i := & \{\boldsymbol{x} \mid (\boldsymbol{v}_{1,j} - \boldsymbol{d})^\top \boldsymbol{P}_1 (\boldsymbol{x} - \boldsymbol{d}) \leq 1,\ j \in \mathbb{N}_{[1,n_v]}\} \\ & \cap \{\boldsymbol{x} \mid (\boldsymbol{v}_{2,j} - \boldsymbol{d})^\top \boldsymbol{P}_2 (\boldsymbol{x} - \boldsymbol{d}) \leq 1,\ j \in \mathbb{N}_{[1,n_v]}\}, \end{aligned} \tag{32}$$

for $i \in \mathbb{N}_{[1,N-1]}$, $\boldsymbol{d} = [2.15\ 0]^\top$, and

$$\boldsymbol{P}_1 = \left[\begin{smallmatrix} 0.14 & 0.17 \\ 0.17 & 0.97 \end{smallmatrix}\right], \quad \boldsymbol{P}_2 = \left[\begin{smallmatrix} 0.20 & 0.05 \\ 0.05 & 0.21 \end{smallmatrix}\right], \tag{33a}$$

$$(\boldsymbol{v}_{1,j} - \boldsymbol{d})^\top \boldsymbol{P}_1 (\boldsymbol{v}_{1,j} - \boldsymbol{d}) = 1, \tag{33b}$$

$$(\boldsymbol{v}_{2,j} - \boldsymbol{d})^\top \boldsymbol{P}_2 (\boldsymbol{v}_{2,j} - \boldsymbol{d}) = 1,\ j \in \mathbb{N}_{[1,n_v]}. \tag{33c}$$

The points $\boldsymbol{v}_{1,j},\ \boldsymbol{v}_{2,j},\ j \in \mathbb{N}_{[1,n_v]}$, are chosen on the boundary of the respective ellipses, which will result in a tangent inequality constraint at $\boldsymbol{v}_{1,j}$ and $\boldsymbol{v}_{2,j}$, respectively. Note that all constraints are non-redundant, i.e., the removal of each constraint changes the set. Moreover, the number of state constraints will be $n_{x_i} = 2n_v$, $i \in \mathbb{N}_{[1,N-1]}$ (for the results in Figures 6 and 7 we picked $n_v = 330$, which results in 660 state constraints for $i \in \mathbb{N}_{[1,N-1]}$). The terminal set $\mathbb{X}_N \subseteq \mathbb{X}_1$ is chosen to be positively invariant for the control law $u_k = \boldsymbol{K}_T \boldsymbol{x}_k = -[0.01\ 0.01]\boldsymbol{x}_k$. An illustration of the state constraints, including the terminal set, is shown in Figure 2. When choosing $N = 12$, the total number of state constraints in this example is $\sum_{i=1}^{N} n_{x_i} = 2(N-1)n_v + n_{x_N} = 7468$.

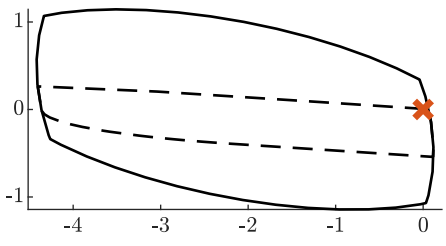

Fig. 2: Illustration of the state constraints $\mathbb{X}_i$, $i \in \mathbb{N}_{[1,N-1]}$ (——), the terminal set $\mathbb{X}_N$ (- - -), and the origin (✖).

We define the MPC cost function as $\ell(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^\top \boldsymbol{R}\boldsymbol{u}$, $\ell_T(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{P}\boldsymbol{x}$ using $\boldsymbol{Q} = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]$, $\boldsymbol{P} = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]$, $\boldsymbol{R} = 1$. For both the first-order optimality set and the approximate ca-MPC scheme, we use the feasible input sequence, $\tilde{\boldsymbol{U}}_k =$

$[\boldsymbol{u}_{1|k-1}^{\star\top} \cdots \boldsymbol{u}_{N-1|k-1}^{\star\top} (\boldsymbol{K}_T \boldsymbol{x}_{N|k-1}^{\star})^\top]^\top$. The last component needed for the *approximate* ca-MPC scheme is the additional input constraint, which we take as $\delta\mathbb{U} := \{u \in \mathbb{R} \mid |u| \leq 0.3\}$.

The two-dimensional state of (31) allows for the visualization of $\mathcal{M}^{(l)}(\boldsymbol{x}_k)$ for $l \in \{1, 2, 3\}$, see Figures 3-5. Note that forward and backward reachable set are approximated using the outer Löwner-John ellipsoid of (20) and (22), respectively. These offline ca-MPC computations required approximately 30 seconds for this example.

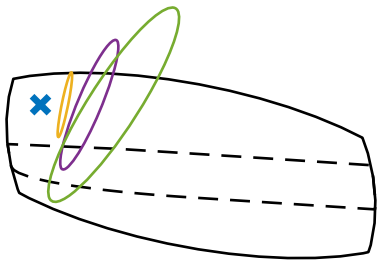

Fig. 3: Illustration of the outer ellipsoidal approximations of the forward reachable sets $\mathcal{E}(\boldsymbol{L}_{i,1}, \boldsymbol{q}_{i,1}) + \boldsymbol{A}^i \boldsymbol{x}_k$ for $i = 4$ (—), $i = 8$ (—), $i = 12$ (—), starting from $\boldsymbol{x}_k$ (✖).

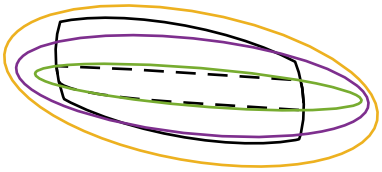

Fig. 4: Illustration of the outer ellipsoidal approximations of the backward reachable sets $\mathcal{E}(\boldsymbol{L}_{i,2}, \boldsymbol{q}_{i,2})$ for $i = 4$ (—), $i = 8$ (—), $i = 12$ (—), starting from the $\mathbb{X}_N$ (- - -).

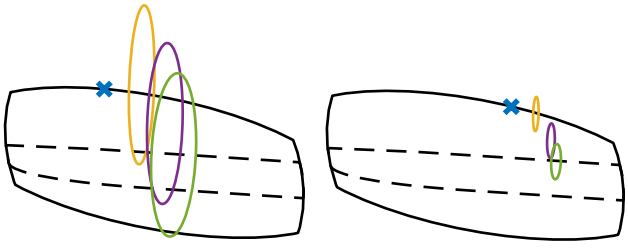

Fig. 5: Illustration of the first-order optimality sets $\mathrm{P}_i(\boldsymbol{\Gamma}\mathcal{J}(\boldsymbol{x}_k, \tilde{\boldsymbol{U}}_k)) + \boldsymbol{A}^i \boldsymbol{x}_k$ for $i = 4$ (—), $i = 8$ (—), $i = 12$ (—) for the current state $\boldsymbol{x}_k$ (✖).

Simulation results: we initialize (31) at $\boldsymbol{x}_0 = -[4\ 0.4]^\top$ and let the MPC, exact ca-MPC, and the approximate ca-MPC scheme regulate the state to the origin. The quadratic programs are solved using a primal-dual interior-point method from the MATLAB MPC toolbox. The resulting state trajectories are shown in Figure 6. First of all, none of the schemes violated the state constraints, as expected. Second, as also expected, the exact ca-MPC scheme has an indistinguishable closed-loop trajectory compared to the original MPC solution, while, the approximate ca-MPC scheme did result in a different closed-loop trajectory. In Figure 7, the computation time and the number of reduced constraints over time for both ca-MPC schemes are shown. Clearly, both the exact and approximate ca-MPC schemes are significantly faster to compute compared to the original MPC problem. Last, in Figure 8, we show the maximum computation time for the MPC and exact ca-MPC feedback law (and the breakdown of the ca-MPC scheme), for a range of state constraints by adjusting in (32) n_v ($1200 \leq \sum_{i=1}^{N} n_{x_i} \leq 45600$). From this figure, we observe a consistent two-order of magnitude computational time improvement using our exact ca-MPC scheme for this example.

Remark 1. *While the double-integrator is convenient for illustrating all aspects of the ca-MPC scheme due to its 2-dimensional visualization possibilities, the interested reader is referred for a higher-dimensional example to [17]. In [17] a similar ca-MPC scheme is applied to a discretized thermal PDE with a temperature upper bound on the (discrete) spatial domain. This leads to a discrete-time linear plant model with 2000 states. Here, a similar two-order of magnitude improvement in computational time was observed for an MPC setup with 20,000 constraints. Due to space limitations, we refer to [17] for further details on this thermal PDE example.*

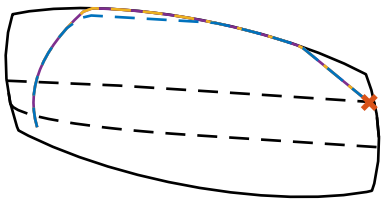

Fig. 6: Closed-loop trajectories for the original MPC (—), exact ca-MPC (- - -), and approximate ca-MPC (- - -).

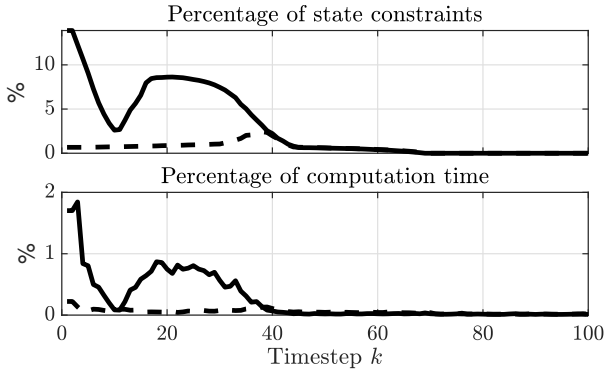

Fig. 7: The percentage of state constraints and computation time for exact ca-MPC (—) and approximate ca-MPC (- - -) relative to the original MPC feedback law.

VII. Conclusions

In this paper, we presented an efficient online constraint removal framework for accelerating MPC for linear sys-

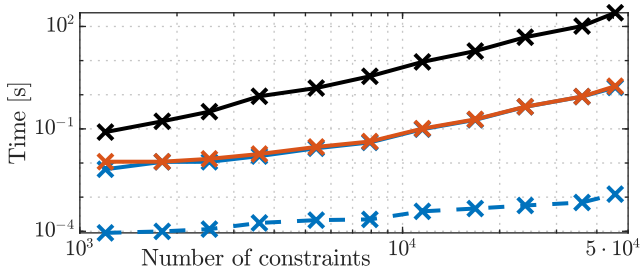

Fig. 8: Numerical scaling of the original MPC (—) and exact ca-MPC scheme (—) with an increasing number of state constraints. The computation time of exact ca-MPC is split in computing $\mathcal{A}$ (- - -) and the resulting QP (—).

tems using system-theoretic insights. A crucial aspect of our proposed method is that the closed-loop behavior and thus properties of the reduced MPC feedback law, such as stability, performance, and constraint satisfaction, remain unchanged when compared to the original MPC feedback law. We achieve this by exploiting computationally efficient bounds on the optimal state trajectory that indicate which constraints can be removed from the MPC problem. In particular, we showed that the forward and backward reachable sets and a first-order optimality set are computationally efficient and powerful tools to remove state constraints from the MPC problem. Additionally, we presented an extension, called approximate ca-MPC, that is able to trade-off closed-loop performance with the computational complexity of the resulting reduced MPC problem, while still maintaining constraint satisfaction and stability.

The results from a numerical example show that the resulting constraint removal scheme can achieve computational speed ups of two-orders of magnitude, without loss of closed-loop performance. An alternative example with a similar ca-MPC using a thermal system described by a PDE is presented in [17], where a comparable two-orders of magnitude improvement was observed as well. Moreover, many of the conceptual ideas that we exploited to get to our constraint-adaptive MPC framework can be extended to nonlinear and time-varying systems as well, see also [16] for first steps.

References

[1] A.J. Ardakani and F. Bouffard. Acceleration of Umbrella Constraint Discovery in Generation Scheduling Problems. *IEEE Trans. Power Syst.*, 30(4):2100–2109, 2015.

[2] D. Arnström, A. Bemporad, and D. Axehill. A Dual Active-Set Solver for Embedded Quadratic Programming Using Recursive LDL. *IEEE Trans. Automat. Contr.*, 67(8):4362–4369, 2022.

[3] A. Bemporad, F. Borrelli, and M. Morari. Model predictive control based on linear programming - the explicit solution. *IEEE Trans. on Automat. Contr.*, 47(12):1974–1985, 2002.

[4] A. Bemporad, A. Oliveri, T Poggi, and M. Storace. Ultra-fast stabilizing model predictive control via canonical piecewise affine approximations. *IEEE Trans. Automat. Contr.*, 56(12):2883–2897, 2011.

[5] G. Frison and M. Diehl. HPIPM: a high-performance quadratic programming framework for model predictive control. In *Proceedings of the IFAC World Congress*, 2020.

[6] Genuit, B.A.G. and Lu, L. and Heemels, W.P.M.H. Approximation of explicit model predictive control using regular piecewise affine functions: an input-to-state stability approach. *IET Control Theory & Applications*, 6(8):1015–1028, 2012.

[7] A. Halder. On the Parameterized Computation of Minimum Volume Outer Ellipsoid of Minkowski Sum of Ellipsoids. In *CDC*, pages 4040–4045. IEEE, 2018.

[8] M. Henk. Löwner-John ellipsoids. *Documenta Math*, 95:106, 2012.

[9] S. Hovland, K. Willcox, and J. T. Gravdahl. MPC for Large-Scale Systems via Model Reduction and Multiparametric Quadratic Programming. In *CDC*, pages 3418–3423. IEEE, 2006.

[10] J.L. Jerez, E.C. Kerrigan, and G.A. Constantinides. A condensed and sparse QP formulation for predictive control. In *CDC*, pages 5217–5222. IEEE, 2011.

[11] M. Jost and M. Mönnigmann. Accelerating model predictive control by online constraint removal. In *CDC*, pages 5764–5769. IEEE, 2013.

[12] M. Jost, G. Pannocchia, and M. Mönnigmann. Online constraint removal: Accelerating MPC with a Lyapunov function. *Automatica*, 57:164–169, 2015.

[13] D.Q. Mayne. Model predictive control: Recent developments and future promise. *Automatica*, 50(12):2967–2986, 2014.

[14] D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[15] J. Nocedal and S. Wright. *Numerical Optimization*, volume 83 of *Springer Series in Operations Research and Financial Engineering*. Springer New York, 2006.

[16] S.A.N. Nouwens, B. de Jager, M. Paulides, and W.P.M.H. Heemels. Constraint-adaptive MPC for large-scale systems: Satisfying state constraints without imposing them. In *NMPC*, pages 232–237. IFAC, 2021.

[17] S.A.N. Nouwens, B. de Jager, M. Paulides, and W.P.M.H. Heemels. Constraint Removal for MPC with Performance Preservation and a Hyperthermia Cancer Treatment Case Study. In *CDC*, pages 4103–4108. IEEE, 2021.

[18] S. Paulraj and P. Sumathi. A Comparative Study of Redundant Constraints Identification Methods in Linear Programming Problems. *Mathematical Problems in Engineering*, pages 1–16, 2010.

[19] J.B. Rawlings, D.Q. Mayne, and M.M. Diehl. *Model Predictive Control: Theory, Computation, and Design*Nob Hill, 2nd edition, 2019.

[20] L.A. Roald and D.K. Molzahn. Implied constraint satisfaction in power system optimization: The impacts of load variations. In *Allert. Conf. Commun. Control. Comput.*, pages 308–315, 2019.

Appendix

Lemma 2. *Given a point $\tilde{\boldsymbol{w}} \in \mathbb{W}$, where $\mathbb{W} \subseteq \mathbb{R}^n$ is compact and convex, then $\boldsymbol{w}^\star := \arg\min_{\boldsymbol{w}\in\mathbb{W}} \|\boldsymbol{L}(\boldsymbol{w}-\boldsymbol{q})\|_2^2$ satisfies $\boldsymbol{w}^\star \in \{\boldsymbol{w} \mid -(\boldsymbol{w}-\boldsymbol{q})^\top \boldsymbol{L}^\top \boldsymbol{L}(\tilde{\boldsymbol{w}}-\boldsymbol{w}) \leq 0\}$.*

Proof. *First-order optimality requires $-2\boldsymbol{L}^\top \boldsymbol{L}(\boldsymbol{w}^\star - \boldsymbol{q}) \in N_{\mathbb{W}}(\boldsymbol{w}^\star)$ [15, Sec 12.7]. Since $\tilde{\boldsymbol{w}}, \boldsymbol{w}^\star \in \mathbb{W}$, the normal cone is bounded by $N_{\mathbb{W}}(\boldsymbol{w}^\star) \subseteq \{\boldsymbol{p} \mid \boldsymbol{p}(\tilde{\boldsymbol{w}} - \boldsymbol{w}^\star) \leq 0\}$. Hence, the minimizer must satisfy $-2\boldsymbol{L}^\top \boldsymbol{L}(\boldsymbol{w}^\star - \boldsymbol{q}) \in \{\boldsymbol{p} \mid \boldsymbol{p}(\tilde{\boldsymbol{w}} - \boldsymbol{w}^\star) \leq 0\}$. Equivalently, we obtain $-(\boldsymbol{w}^\star - \boldsymbol{q})^\top \boldsymbol{L}^\top \boldsymbol{L}(\tilde{\boldsymbol{w}} - \boldsymbol{w}^\star) \leq 0$.* □

Rewriting the result using the center of the ellipse $\frac{1}{2}(\tilde{\boldsymbol{w}}+\boldsymbol{q})$, gives $\boldsymbol{w}^\star \in \{\boldsymbol{w} \mid \|\boldsymbol{L}(\boldsymbol{w} - \frac{1}{2}(\tilde{\boldsymbol{w}} + \boldsymbol{q}))\|_2 \leq \frac{1}{2}\|\boldsymbol{L}(\tilde{\boldsymbol{w}} - \boldsymbol{q})\|_2\}$, which is the same result as used in (27).

When considering individual constraints (29) becomes

$$\mathcal{E}(\boldsymbol{L}, \boldsymbol{q}) \cap \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{c}\boldsymbol{x} \leq b\} = \mathcal{E}(\boldsymbol{L}, \boldsymbol{q}), \tag{34}$$

Note that $\mathcal{E}(\boldsymbol{L}, \boldsymbol{q}) \cap \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{c}\boldsymbol{x} \leq b\} \neq \emptyset$, due to feasibility of the MPC. Hence, an empty intersection between $\{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{c}\boldsymbol{x} = b\}$ and $\mathcal{E}(\boldsymbol{L}, \boldsymbol{q})$ implies (34).

We map the ellipsoid to a unit ball on the origin using $\boldsymbol{x} = \boldsymbol{q} + \boldsymbol{L}^{-1}\boldsymbol{w}$. Substituting the mapping into the hyperplane $\boldsymbol{c}\boldsymbol{x} =$

b yields $\boldsymbol{c}\boldsymbol{L}^{-1}\boldsymbol{w} = b - \boldsymbol{c}\boldsymbol{q}$. The distance from the hyperplane to the origin is then $\frac{|b-\boldsymbol{c}\boldsymbol{q}|}{\|\boldsymbol{c}\boldsymbol{L}^{-1}\|_2}$. The hyperplane intersects the ellipse when $\frac{|b-\boldsymbol{c}\boldsymbol{q}|}{\|\boldsymbol{c}\boldsymbol{L}^{-1}\|_2} \leq 1$. Hence, $\|\boldsymbol{c}\boldsymbol{L}^{-1}\|_2 \leq |b-\boldsymbol{c}\boldsymbol{q}|$ implies (34), which is (28b).