# Scalable command governor via semi-ellipsoidal set for linear systems with time-varying soft constraints

Hoai Nam Nguyen

# Scalable Command Governor via Semi-Ellipsoidal Set for Linear Systems with Time-Varying Soft Constraints

Hoai-Nam Nguyen [a]

[a] *Samovar, Telecom SudParis, Institute Polytechnique de Paris, 91120 Palaiseau, France*

**Abstract**

This paper provides a solution for a command governor (CG) that employs a semi-ellipsoidal set. The motivation is driven by the need to address the shortcomings of polyhedral and ellipsoidal sets widely used in CG. In particular for many applications, polyhedral sets require a large number of linear inequality constraints while ellipsoidal sets are too conservative. Furthermore, both types of sets are generally designed for systems with time-invariant and hard constraints. The contributions of the paper are: i) we provide new convex conditions to construct an invariant and constraint-admissible semi-ellipsoidal set for discrete-time linear systems with time-varying soft constraints; ii) we propose a computationally efficient procedure to solve the online optimization problem associated with the newly introduced semi-ellipsoidal set in CG. Three numerical examples with comparison to earlier solutions from the literature illustrate the effectiveness of the proposed approach.

*Key words:* Command Governor, Linear System, State and Input Constraint, Invariant Set, ADMM

## 1 Introduction

This paper is concerned with the tracking problem of discrete-time linear time-invariant systems, subject to possibly time-varying soft constraints on both input and state. This is a problem that has been extensively studied over the last decades, and a number of different solutions are available as, for example, those based on model predictive control (MPC) [15], or command governor (CG) [5].

*Email address:* hoai-nam.nguyen@telecom-sudparis.eu (Hoai-Nam Nguyen).

In MPC [15], an optimization problem is solved at each time instant to find the optimal input that drives the predicted plant output to the desired reference. MPC provides a natural way to take the time-varying and/or soft constraints into account. However, it is not trivial to guarantee recursive feasibility and asymptotic stability for MPC with time-varying and/or soft constraints.

CG provides a useful alternative to MPC [5]. The main idea of CG is to use optimization to find the best *applied reference* or equivalently the *command* based on the current state and the current value of the *desired reference*. CG may be attractive to practitioner concerned with online computational effort and/or interested in preserving an existing well-designed nominal controller.

In CG, the state and the command are imposed to lie in the set $\Omega$, which is invariant and constraint-admissible. The set $\Omega$ is constructed offline, and is used to guarantee the constraint satisfaction. In CG, ellipsoidal [18] and polyhedral sets [6] are widely considered for $\Omega$. The popularity of ellipsoidal sets stems from their computational efficiency via Linear Matrix Inequality (LMI) formulation, their fixed complexity with respect to the state space dimension. However, it is well known [1], [17] that ellipsoids provide a conservative approximation of the domain of attraction (DoA). In contrast, polyhedral sets can provide a less conservative solution. However, their use comes with significant trade-offs. To describe a polyhedral invariant set, a large number of linear inequality constraints is typically required even for average-size systems [1].

The set $\Omega$ is computed using iterative procedures and/or convex optimization. In general, there is no closed-form relationship between $\Omega$ and the state/input constraint sets. Consequently, considering time-varying and/or soft constraints in CG is not straightforward. As a result, most of CG schemes in the literature assume that the constraints are time-invariant and hard. There are only two papers that we are aware of, and that consider time-varying or soft constraints: [11], [12]. In [12], it was shown that the use of strictly contractive sets instead of invariant ones can handle time-varying constraints, if the constraints vary slowly enough. In [11], slack variables were considered to relax the constraints. However, the introduction of slack variables along with the command in the state extension makes the problem of constructing $\Omega$ very complex.

In [14], [20] an interesting class of invariant and constraint-admissible sets is considered for nominal linear systems with input and state constraint. The set, which is called semi-ellipsoidal set. However, the construction of the set is based on a non-convex optimization problem.

In this paper, we follow the works in [14], [20]. The main aim is to propose new CG schemes that employ invariant and constraint-admissible semi-ellipsoidal sets. The main contributions of the paper are:

(1) To the best of the author's knowledge, this is the first time a semi-ellipsoidal set is used in the context of a CG.
(2) We provide new convex conditions to construct an invariant and constraint-admissible semi-ellipsoidal set for CGs with linear feedback. The new set is formed explicitly by the state and input constraint sets. Consequently, it can be easily extended to deal with time-varying soft constraints.
(3) As shown in [4] for CGs, using saturated feedback instead of linear one can enlarge the DoA, and hence improve the transient time. We provide convex conditions to construct invariant and constraint-admissible semi-ellipsoidal sets for CGs with saturated feedback. Our algorithm can cope with time-varying soft constraints.
(4) Moving from an ellipsoidal and polyhedral set to a semi-ellipsoidal set introduces new types of constraints. Hence, the new optimization problem is no longer a quadratically constrained quadratic program [18] or a quadratic progam [6]. We develop a tailored algorithm specifically for the new optimization problem.

The paper is organized as follows. Section 2 covers the problem formulation and earlier works on CG. Section 3, Section 4 are, respectively, dedicated to the construction of an invariant and constraint-admissible semi-ellipsoidal set with linear feedback, with saturated feedback for time-invariant hard constraints. Then in Section 5, we extend the results in Section 3 and Section 4 to cope with time-varying soft constraints. Section 6 is concerned with the online optimization problem. Three numerical examples with comparison to earlier solutions from the literature are evaluated in Section 7. Finally, Section 8 concludes the article.

**Notation:** We denote by $\mathbb{R}$ the set of real numbers, by $\mathbb{R}^n$ the set of real $n \times 1$ vectors, by $\mathbb{R}^{n\times m}$ the set of real $n \times m$ matrices. We use $\mathbf{0}, \mathbf{I}$, respectively, to denote the zero matrix, and the identity matrix of appropriate dimension. We denote a positive definite(semi-definite) matrix $P$ by $P \succ 0 (P \succeq 0)$. For a given matrix $P \succ 0$ of size $n$, $\mathcal{E}(P)$ is defined as

$$\mathcal{E}(P) = \{x \in \mathbb{R}^n | x^T P^{-1} x \leq 1\}. \tag{1}$$

For a given positive natural number $n$, we use $\overline{1, n}$ to denote the set $\{1, 2, \ldots, n\}$. For symmetric matrices, the symbol $(*)$ denotes each of its symmetric block. For a given vector $d$, we use $\text{diag}(d)$ to denote the square diagonal matrix with the elements of vector $d$ on the main diagonal.

## 2 Problem Formulation and Earlier Works on Command Governor

### *2.1 Problem Formulation*

We consider the following discrete-time linear time-invariant system

$$\begin{cases} x(k+1) = Ax(k) + Bu(k) \\ y(k) \quad\quad = Cx(k) \end{cases}, \tag{2}$$

where $x(k) \in \mathbb{R}^{n_x}$, $u(k) \in \mathbb{R}^{n_u}$, and $y(k) \in \mathbb{R}^{n_y}$ are, respectively, the state, the control input, and the output. $A \in \mathbb{R}^{n_x \times n_x}, B \in \mathbb{R}^{n_x \times n_u}, C \in \mathbb{R}^{n_y \times n_x}$.

The state $x(k)$ and the input $u(k)$ are subject to the constraints $x(k) \in \mathcal{X}$, $u(k) \in \mathcal{U}$, where $\mathcal{X}$,$\mathcal{U}$ are

$$\begin{cases} \mathcal{X} = \{x \in \mathbb{R}^{n_x} | \underline{g}_i \leq f_i x \leq \overline{g}_i, \forall i \in \overline{1, l_x}\} \\ \mathcal{U} = \{u \in \mathbb{R}^{n_u} | \underline{u}_j \leq u_j \leq \overline{u}_j, \forall j \in \overline{1, n_u}\} \end{cases}, \tag{3}$$

where $f_i^T \in \mathbb{R}^{n_x}, \underline{g}_i \in \mathbb{R}, \overline{g}_i \in \mathbb{R}$, $\underline{u}_j \in \mathbb{R}, \overline{u}_j \in \mathbb{R}$, $i \in \overline{1, l_x}, j \in \overline{1, n_u}$. At the moment for simplicity, we consider the case where the constraints (3) are time-invariant and hard.

**Control Objectives:** For a given reference signal $r(k) \in \mathbb{R}^{n_y}$, the objective of the paper is to calculate a CG based control action $u(k) = \mathbf{U}(x(k), r(k))$ such that: i) the output $y(k)$ follows as close as possible to $r(k)$; ii) the state and input constraints (3) are fulfilled.

We recall some earlier works on CG in the next section.

### *2.2 Earlier Works on Command Governor*

Any steady state $(x_s, u_s, y_s)$ of the system (2) satisfies the following equation

$$A_s \begin{bmatrix} x_s \\ u_s \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} y_s, \tag{4}$$

where

$$A_s = \begin{bmatrix} A - \mathbf{I} & B \\ C & \mathbf{0} \end{bmatrix}.$$

For simplicity, in the rest of the paper we assume that the matrix $A_s$ is invertible. In this case we can express the solution of (4) as

$$x_s = L_x v,\ u_s = L_u v, \tag{5}$$

where $v = y_s$ is the so-called command, and $L_x \in \mathbb{R}^{n_x \times n_y}$, $L_u \in \mathbb{R}^{n_u \times n_y}$ are

$$\begin{bmatrix} L_x \\ L_u \end{bmatrix} = A_s^{-1} \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix}.$$

The CG is an add-on scheme for enforcing the state and input constraints. It is based on an assumption that an asymptotically stabilizing control law $u(k) = Kx(k)$ is available. This control law is designed to satisfy some performance specifications for small signals near the origin. In the CG, the control input is of the form

$$u(k) = K(x(k) - x_s) + u_s = Kx(k) + Lv, \tag{6}$$

where $L = L_u - KL_x$.

By substituting (6) into (2), and by considering $v(k+1) = v(k)$, we can express the closed-loop system as

$$x_e(k+1) = A_e x_e(k), \tag{7}$$

where $x_e(k) = [x(k)^T\ v(k)^T]^T$, and

$$A_e = \begin{bmatrix} A + BK & BL \\ \mathbf{0} & \mathbf{I} \end{bmatrix}.$$

Using (6), the constraints (3) become $x_e \in \mathcal{X}_e$ with

$$\mathcal{X}_e = \left\{ x \in \mathbb{R}^{n_x}, v \in \mathbb{R}^{n_y} \;\middle|\; \begin{array}{l} x \in \mathcal{X}, \\ Kx + Lv \in \mathcal{U} \end{array} \right\}. \tag{8}$$

We say that a set $\Omega \subseteq \mathbb{R}^{n_x+n_y}$ is invariant for (7) if for any $x_e(k) \in \Omega$, one has $x_e(k+1) \in \Omega$. In addition, if $\Omega \subseteq \mathcal{X}_e$, then $\Omega$ is constraint-admissible with respect to (8).

For a given state $x(k)$ and a given reference signal $r(k)$ at time instant $k$, the CG applies the control law $u(k) = Kx(k) + Lv^*(k)$ to the system (2), where $v^*(k)$ is the solution of the following optimization problem

$$\begin{array}{l} \min\limits_{v}\ (v - r(k))^T Q (v - r(k)) \\ \text{s.t. } (x(k), v) \in \Omega, \end{array} \tag{9}$$

where $Q \succ 0$ is a weighting matrix, and $\Omega$ is any invariant and constraint-admissible set for (7), (8).

The basic idea of CG is the following. If there is no constraint violation, then $v^*(k) = r(k)$ is the solution of (9). Hence, the CG does not interfere with the operation of the system. If a potential for constraint violation exists, the CG seeks the closest admissible command $v^*(k)$ to $r(k)$. In the extreme case thanks to the robust invariance of $\Omega$, $v(k+1) = v^*(k)$ remains a feasible solution of (9). This implies that the CG temporarily isolates the system from further variations of the reference to assure the constraint satisfaction.

Assuming feasibility at the initial condition, the CG (6), (9) guarantees recursive feasibility, i.e., if (9) is feasible at time $k$, then it is feasible at time $k+1$. If $r(k) = r$ remains constant and $r$ is reachable, then $v^*(k)$ converges to $r$ in a *finite time*. If $r$ is not reachable, then $v^*(k)$ will converge to the closest feasible value to $r$ in a *finite time*. The finite-settling-time convergence is a desirable property. It shows that after transients caused by large changes in $r(k)$, the CG (6), (9) ensures the system reaches a stable steady-state condition.

As written in Introduction, in the CG literature, only ellipsoidal and polyhedral sets are used for $\Omega$. The aim of this paper is to propose new CG schemes that are based on semi-ellipsoidal sets. We will first consider a semi-ellipsoidal set with a linear feedback. We then extend our approach to the case of nonlinear saturated feedback.

**Remark 1:** For computational reasons [5], if polyhedral set is used for $\Omega$, we generally impose $x_e \in \tilde{X}_e$ instead of the constraint (8) where

$$\tilde{X}_e = X_e \cap X_e^{\mu}, \tag{10}$$

with

$$X_e^{\mu} = \{(x, v) | L_x v \in (1-\mu)\mathcal{X}, L_u v \in (1-\mu)\mathcal{U}\}. \tag{11}$$

The margin $\mu > 0$ is typically small.

## 3 Semi-Ellipsoidal Set with Linear Feedback

In this section, we will provide a convex procedure to obtain an invariant and constraint-admissible semi-ellipsoidal set for (7), (8). The main idea is to exploit the particular structure of $A_e, E_e$. Starting from the original $x-$space, we show how to obtain invariance and constraint-admissibility conditions for semi-ellipsoidal set in the extended $x_e-$space via the translation and the scaling operators.

*3.1 Translation*

For a given $P \succ 0$, consider the sets $\Omega_x(P)$ and $\Omega_v(P)$ with

$$\Omega_x(P) = \left\{ \begin{bmatrix} x \\ v \end{bmatrix} \,\middle|\, (x - L_x v)^T P^{-1} (x - L_x v) \leq 1 \right\}, \tag{12}$$

and, $i \in \overline{1, l_x}, j \in \overline{1, n_u}$

$$\Omega_v(P) = \left\{ v \,\middle|\, \begin{array}{c} -f_i L_x v \leq -\underline{g}_i - \sqrt{f_i P f_i^T} \\ f_i L_x v \leq \overline{g}_i - \sqrt{f_i P f_i^T} \\ -L_{u,j} v \leq -\underline{u}_j - \sqrt{K_j P K_j^T} \\ L_{u,j} v \leq \overline{u}_j - \sqrt{K_j P K_j^T} \end{array} \right\}, \tag{13}$$

where $L_{u,j}$ and $K_j$ are, respectively, the $j-$th row of $L_u$ and $K$.

In the $(x, v)-$space, $\Omega_x(P)$ is an unbounded ellipsoid. In the $x-$space, $\Omega_x(P)$ is the translation of $\mathcal{E}(P)$ from the origin to the point $L_x v$. In other words, $\Omega_x(P)$ is an ellipsoid centered at $L_x v$. Define

$$\Omega_e(P) := \Omega_x(P) \cap \Omega_v(P). \tag{14}$$

Clearly, $\Omega_e(P)$ is a semi-ellipsoidal set as it is the intersection of the ellipsoid $\Omega_x(P)$ and the polyhedral set $\Omega_v(P)$.

**Remark 2:** There is another useful geometric interpretation of $\Omega_e(P)$, namely, $\Omega_e(P)$ is a parameterized ellipsoid $\Omega_x(P)$ with its center constrained within the set $\Omega_v(P)$.

**Theorem 1:** The set $\Omega_e(P)$ is invariant and constraint-admissible for (7), (8) if and only if $P$ satisfies the following condition

$$\begin{bmatrix} P & A_c P \\ * & P \end{bmatrix} \succeq 0. \tag{15}$$

**Proof:** We divide the proof into two parts: invariance proof and constraint admissibility proof.

**Invariance Proof:** One needs to show that $x_e(k+1) \in \Omega_e(P)$ for any $x_e(k) \in \Omega_e(P)$. As $v(k+1) = v(k)$, it is clear that $x_e(k+1) \in \Omega_v(P)$ if $x_e(k) \in \Omega_v(P)$.

It remains to show that $x_e(k+1) \in \Omega_x(P)$. Using (6), (7), one gets

$$x(k+1) = A_c x(k) + B(u_s(k) - K x_s(k)), \tag{16}$$

where $x_s(k) = L_x v(k)$, $u_s(k) = L_u v(k)$. Using (4), (5) one gets

$$x_s(k+1) = Ax_s(k) + Bu_s(k),$$

where $x_s(k+1) = L_x v(k+1)$. Thus, with (16)

$$x(k+1) - x_s(k+1) = A_c(x(k) - x_s(k)),$$

or, equivalently

$$\zeta(k+1) = A_c\zeta(k). \tag{17}$$

where $\zeta(k) = x(k) - L_x v(k)$. It is well known [2] that condition (15) is necessary and sufficient for the invariance of $\mathcal{E}(P)$ with respect to (17). Hence, for any $x(k), v(k)$ such that

$$(x(k) - L_x v(k))^T P^{-1}(x(k) - L_x v(k)) \leq 1,$$

one has

$$(x(k+1) - L_x v(k+1))^T P^{-1}(x(k+1) - L_x v(k+1)) \leq 1.$$

It follows that for any $x_e(k) \in \Omega_x(P)$, one has $x_e(k+1) \in \Omega_x(P)$.

**Constraint Admissibility Proof:** One needs to show that $\Omega_e(P) \subseteq \mathcal{X}_e$, or equivalently

$$x \in \mathcal{X}, Kx + Lv \in \mathcal{U},$$

for any $x_e \in \Omega_e(P)$. First we will show that $x \in \mathcal{X}$. For a given $v \in \mathbb{R}^{n_y}$, consider the following optimization problem, $i \in \overline{1, l_x}$

$$\begin{aligned} &\min_x \ f_i x \\ &\text{s.t. } (x - L_x v)^T P^{-1}(x - L_x v) \leq 1. \end{aligned} \tag{18}$$

Note that the constraint of (18) is the set $\Omega_x(P)$ with a given $v$. We denote this set as $\Omega_x(P, v)$. As the cost is linear and the set $\Omega_x(P, v)$ is bounded, the constraint of (18) is always active. The Lagrangian is

$$\mathcal{L}(x, \lambda) = f_i x + \frac{\lambda}{2}\left((x - L_x v)^T P^{-1}(x - L_x v) - 1\right),$$

where $\lambda \geq 0$ is the Lagrange multiplier. Since the constraint (18) is always active, it follows that $\lambda > 0$. One has

$$\frac{\partial \mathcal{L}}{\partial x} = f_i^T + \lambda^* P^{-1}(x^* - L_x v) = \mathbf{0}.$$

Therefore

$$x^* = L_x v - \frac{1}{\lambda^*} P f_i^T$$

As the constraint in (18) is active, one obtains

$$(L_x v - \frac{1}{\lambda^*} P f_i^T - L_x v)^T P^{-1} (L_x v - \frac{1}{\lambda^*} P f_i^T - L_x v) = 1,$$

or equivalently, $\lambda^* = \sqrt{f_i P f_i^T}$. Hence

$$x^* = L_x v - \frac{1}{\sqrt{f_i P f_i^T}} P f_i^T.$$

It follows that

$$\min_{x \in \Omega_x(P,v)} f_i x = f_i L_x v - \sqrt{f_i P f_i^T}.$$

Using (13), for any $v \in \Omega_v(P)$ one has

$$f_i L_x v - \sqrt{f_i P f_i^T} \geq \underline{g}_i, i \in \overline{1, l_x}.$$

Therefore, for any $x_e \in \Omega_e(P), i \in \overline{1, l_x}$

$$f_i x \geq \underline{g}_i. \tag{19}$$

Analogously, it can be shown for any $x_e \in \Omega_e(P), i \in \overline{1, l_x}$ that

$$f_i x \leq \overline{g}_i. \tag{20}$$

Combining (19), (20), one gets $x \in \mathcal{X}$ for any $x_e \in \Omega_e(P)$.

Now we show for any $x_e \in \Omega_e(P)$ that $Kx + Lv \in \mathcal{U}$. Using similar arguments like those for the state constraints, it can be shown that for a given $v$ and for any $x_e \in \Omega_x(P)$

$$\min_{x \in \Omega_x(P,v)} (K_j x + L_j v) = L_{u,j} v - \sqrt{K_j P K_j^T},$$

where $L_j$ is the $j-$th row of $L$. Using (13), for any $v \in \Omega_v(P)$ one has

$$L_{u,j} v - \sqrt{K_j P K_j^T} \geq \underline{u}_j.$$

Hence, for any $j \in \overline{1, n_u}$, $x_e \in \Omega_e(P)$

$$K_j x + L_j v \geq \underline{u}_j. \tag{21}$$

Analogously, one obtains, for any $j \in \overline{1, n_u}$, $x_e \in \Omega_e(P)$

$$K_j x + L_j v \leq \overline{u}_j. \tag{22}$$

Using (21), (22), one gets $x_e \in \Omega_v$ for any $x_e \in \Omega_e(P)$. The proof is complete. □

Once invariance and constraint-admissibility conditions for $\Omega_e(P)$ are obtained, our next step is to maximize the size of $\Omega_e(P)$. Since $\Omega_e(P)$ can be considered as a parameterized ellipsoid, our idea is to optimize $\Omega_x(P)$ or equivalently $P$ for a particular fixed value of $v = v_f, v_f \in \Omega_v(P)$. Using (13), and since $f_i P f_i^T \geq 0, K_j P K_j^T \geq 0$, we obtain a necessary condition of $v_f$ for guaranteeing that $P$ exists

$$\begin{cases} \underline{g}_i \leq f_i L_x v_f \leq \overline{g}_i, \forall i \in \overline{1, l_x}, \\ \underline{u}_j \leq L_{u,j} v_f \leq \overline{u}_j, \forall j \in \overline{1, n_u} \end{cases} \tag{23}$$

Using (13), for a fixed $v_f$ satisfying (23), the constraints on $P$ are

$$\begin{cases} f_i P f_i^T \leq \left(f_i L_x v_f - \underline{g}_i\right)^2, \forall i \in \overline{1, l_x}, \\ f_i P f_i^T \leq \left(f_i L_x v_f - \overline{g}_i\right)^2, \forall i \in \overline{1, l_x}, \\ K_j P K_j^T \leq \left(L_{u,j} v_f - \underline{u}_j\right)^2, \forall j \in \overline{1, n_u}, \\ K_j P K_j^T \leq \left(L_{u,j} v_f - \overline{u}_j\right)^2, \forall j \in \overline{1, n_u}. \end{cases} \tag{24}$$

For a given and fixed $v_f$ satisfying (23), the problem of optimizing $P$ can be formulated as

$$\begin{aligned} &\min_P f(P) \\ &\text{s.t. } (15), (24), \end{aligned} \tag{25}$$

where $f(P)$ can be any convex function that maximizes the volume of the set $\Omega_x(P, v_f)$. For example, $f(P)$ can be $-\text{logdet}(P)$ or $-\text{tr}(P)$.

Problem (25) is a convex semi-definite program (SDP). We can solve it efficiently using free available LMI parser such as YALMIP [16] or CVX [8].

It is clear that one should select $v_f$ to optimize $P$. Using (24), one has if

$$\left|f_i L_x v_f - \underline{g}_i\right| \leq \left|f_i L_x v_f - \overline{g}_i\right|, i \in \overline{1, l_x}$$

then the constraint

$$f_i P f_i^T \leq \left(f_i L_x v_f - \overline{g}_i\right)^2, i \in \overline{1, l_x}$$

is redundant. Otherwise the constraint

$$f_i P f_i^T \leq \left(f_i L_x v_f - \underline{g}_i\right)^2, i \in \overline{1, l_x}$$

is redundant. We have the same conclusion for any $j \in \overline{1, n_u}$. Hence, we can

select $v_f$ as a solution of the following optimization problem

$$\begin{aligned}\max_{v_f} \; & \sum_{i=1}^{l_x} w_{x,i} \min\left\{\left|f_i L_x v_f - \underline{g}_i\right|, |f_i L_x v_f - \overline{g}_i|\right\} \\ & + \sum_{i=1}^{n_u} w_{u,j} \min\left\{\left|L_{u,j} v_f - \underline{u}_j\right|, |L_{u,j} v_f - \overline{u}_j|\right\} \\ \text{s.t.} \; & \text{Constraints (23)},\end{aligned} \tag{26}$$

where $w_{x,i} \geq 0, w_{u,j} \geq 0$ are weighting parameters.

Problem (26) can be converted into a mixed integer linear program. Note that if $\mathcal{X}$,$\mathcal{U}$ contain the origin in their interior, a simple but not necessarily optimal choice for $v_f$ is $v_f = \mathbf{0}$.

### *3.2 Scaling*

We assume that $\Omega_e(P)$ is available as a result from Section 3.1. Clearly, it is possible to employ $\Omega_e(P)$ in a CG framework, as $\Omega_e(P)$ is invariant and constraint-admissible for (7), (8). Using the scaling operator, our aim of this section is to provide an even larger set than $\Omega_e(P)$ that expands the feasible region for the CG to operate in. For a given scalar $\alpha \geq 0$, define

$$\Omega_x(\alpha, P) = \left\{ \begin{bmatrix} x \\ v \end{bmatrix} \;\middle|\; (x - L_x v)^T P^{-1} (x - L_x v) \leq \alpha^2 \right\}. \tag{27}$$

and, $\forall i \in \overline{1, l_x}, \forall j \in \overline{1, n_u}$

$$\Omega_v(\alpha, P) = \left\{ v \;\middle|\; \begin{array}{c} -f_i L_x v \leq -\underline{g}_i - \alpha\sqrt{f_i P f_i^T} \\ f_i L_x v \leq \overline{g}_i - \alpha\sqrt{f_i P f_i^T} \\ -L_{u,j} v \leq -\underline{u}_j - \alpha\sqrt{K_j P K_j^T} \\ L_{u,j} v \leq \overline{u}_j - \alpha\sqrt{K_j P K_j^T} \end{array} \right\}; \tag{28}$$

Define also

$$\Omega_e(\alpha, P) := \Omega_x(\alpha, P) \cap \Omega_v(\alpha, P). \tag{29}$$

Clearly, $\Omega_e(\alpha, P)$ is a semi-ellipsoidal set.

Note that $\Omega_x(\alpha, P)$ is an unbounded ellipsoid in the $(x, v)-$space for any $\alpha \geq 0$. Note also that $\Omega_x(\alpha, P)$ is the scaling of $\Omega_x(P)$ with the scaling factor $\alpha$, i.e., $\alpha[x^T \; v^T]^T \in \Omega_x(\alpha, P)$ for any $[x^T \; v^T]^T \in \Omega_x(P$. However, this is not the case for $\Omega_v(\alpha, P)$, i.e., $\Omega_v(\alpha, P)$ is not a scaling of $\Omega_v(P)$.

**Theorem 2:** Given any $\alpha \geq 0$ such that $\Omega_v(\alpha, P)$ is non-empty. If $\Omega_e(P)$ is invariant for (7), then $\Omega_e(\alpha, P)$ is invariant and constraint-admissible for (7), (8).

**Proof:** Similar to the proof of Theorem 1, we divide the proof of Theorem 2 into two parts: invariance proof and constraint-admissibility proof.

**Invariance Proof:** Clearly, for any $\alpha$ such that $\Omega_v(\alpha, P)$ is non-empty, one has

$$x_e(k+1) \in \Omega_v(\alpha, P),$$

for any $x_e(k) \in \Omega_v(\alpha, P)$ as $v(k+1) = v(k)$.

It remains to prove that $x_e(k+1) \in \Omega_x(\alpha, P)$ for any $x_e(k) \in \Omega_e(\alpha, P)$. Note that if $\alpha = 0$, then $x_e(k) \in \Omega_e(\alpha, P)$ if and only if $x_e(k) = \mathbf{0}$. Hence $x_e(k+1) = \mathbf{0} \in \Omega_e(\alpha, P)$. If $\alpha > 0$, one has

$$\frac{x_e(k+1)}{\alpha} = A_e \frac{x_e(k)}{\alpha}.$$

Using the two facts that

- one has $\frac{x_e(k)}{\alpha} \in \Omega_e(P)$ for any $x_e(k) \in \Omega_e(\alpha, P)$, and for any $\alpha > 0$;
- $\Omega_e(P)$ is invariant for (7);

one gets $\frac{x_e(k+1)}{\alpha} \in \Omega_e(P)$. It follows that $x_e(k+1) \in \Omega_x(\alpha, P)$.

**Constraint Admissibility Proof:** The main idea of the proof is to show for any $x_e \in \Omega_e(\alpha, P)$ that $x \in \mathcal{X}$ and $Kx + Lv \in \mathcal{U}$. The proof is omitted here, as it follows the same arguments as the one of Theorem 1. □

The following Algorithm can be used to construct an invariant and constraint-admissible semi-ellipsoidal set $\Omega_e(\alpha, P)$.

Algorithm 1: Construction of $\Omega_e(\alpha, P)$

1: Select $v_f$.
2: Obtain $P$ by solving (25).
3: Obtain $\Omega_x(\alpha, P)$ by using (27).
4: Obtain $\Omega_v(\alpha, P)$ by using (28).
5: $\Omega_e(\alpha, P) = \Omega_x(\alpha, P) \cap \Omega_v(\alpha, P)$.

**Remark 3:** It is stressed that $\alpha$ is not a decision variable in Algorithm 1. We use $\alpha$ as an auxiliary variable to enlarge the feasible set for the CG.

**Remark 4:** An interesting feature of $\Omega_e(\alpha, P)$ is that it has a fixed complexity of representation. Given $n_x, n_u, n_y, l_x$, the number of parameters required to describe $\Omega_e(\alpha, P)$ is the sum of $n_x \times n_y + \frac{n_x(n_x+1)}{2}$ and $2(l_x + n_u) \times (n_y + 2)$. These two terms are, respectively, for describing $\Omega_x(\alpha, P)$ and $\Omega_v(\alpha, P)$.

## 4 Semi-Ellipsoidal Set with Saturated Feedback

The aim of this section is to provide a way to construct an invariant and constraint-admissible semi-ellipsoidal set for CG using the following saturated control law

$$u(k) = \text{sat}(Kx(k) + Lv(k)), \tag{30}$$

where the saturation function $\text{sat}(\kappa)$ with $\kappa \in \mathbb{R}^{n_u}$ is defined as $\text{sat}(\kappa) = \min\{\tilde{\kappa}, \overline{u}\}$ with $\tilde{\kappa} = \max\{\kappa, \underline{u}\}$. The operators min and max are taken component-wise.

As noticed in [4], the motivation of using the nonlinear saturated control law (30) instead of the linear one (6) is that the associated DoA of (30) can be significantly larger than that of (6). As a result, the transient behavior of the closed-loop system can thus be improved.

It is worth noticing that in the linear feedback case (6), Theorem 1 provides a foundation to construct a semi-ellipsoidal invariant and constraint-admissible set. Theorem 2 is built upon Theorem 1. Hence, due to space limitations, the focus in this section is on extending Theorem 1 to handle the saturated control law (30).

In the following we recall the linear differential inclusion (LDI) modeling framework in [9]. We will use it to model the saturation nonlinearity. Define the following set of $n_u \times n_u$ diagonal matrices $\mathcal{S}$ as

$$\mathcal{S} = \left\{ S_m | S_m = \text{diag}(s_1, \ldots, s_{n_u}), s_i \in \{0, 1\}, i \in \overline{1, n_u} \right\}.$$

There are $2^{n_u}$ elements in $\mathcal{S}$. For a given matrix element $S_m \in \mathcal{S}$, define $S_m^- = \mathbf{I} - S_m, m \in \overline{1, m_u}$ with $m_u = 2^{n_u}$.

For given vectors $u \in \mathbb{R}^{n_u}, \xi \in \mathbb{R}^{n_u}$, define the set $\mathcal{D} \subseteq \mathbb{R}^{n_u}$ as

$$\mathcal{D} = \{ d_m \in \mathbb{R}^{n_u} | d_m = S_m u + S_m^- \xi, m \in \overline{1, n_u} \},$$

For example if $n_u = 2$, we have $m_u = 4$ and

$$d_1 = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, d_2 = \begin{bmatrix} u_1 \\ \xi_2 \end{bmatrix}, d_3 = \begin{bmatrix} \xi_1 \\ u_2 \end{bmatrix}, d_4 = \begin{bmatrix} \xi_1 \\ \xi_2 \end{bmatrix}.$$

Define also $\text{Co}(\mathcal{D})$ as the convex hull of $\mathcal{D}$, i.e.,

$$\text{Co}(\mathcal{D}) = \left\{ \sum_{m=1}^{m_u} \lambda_m d_m | d_m \in \mathcal{D}, \lambda_m \geq 0, \sum_{m=1}^{m_u} \lambda_m = 1 \right\}.$$

**Lemma 1:** [9] Given $u \in \mathbb{R}^{n_u}, \xi \in \mathbb{R}^{n_u}$ such that $\underline{u}_j \leq \xi_j \leq \overline{u}_j, j \in \overline{1, n_u}$, one has,

$$\text{sat}(u) \in \text{Co}(\mathcal{D}). \tag{31}$$

By substituting (30) into (2), and by considering $v(k+1) = v(k)$, one gets

$$\begin{bmatrix} x(k+1) \\ v(k+1) \end{bmatrix} = \begin{bmatrix} Ax(k) + B\text{sat}(Kx(k) + Lv(k)) \\ v(k) \end{bmatrix}. \tag{32}$$

Given matrices $P_s \in \mathbb{R}^{n_x \times n_x}$, $H \in \mathbb{R}^{n_u \times n_x}$, define $\Omega_v(P_s, H)$ as, $\forall i \in \overline{1, l_x}, \forall j \in \overline{1, n_u}$

$$\Omega_v(P_s, H) = \left\{ v \left| \begin{array}{c} -f_i L_x v \leq -\underline{g}_i - \sqrt{f_i P_s f_i^T} \\ f_i L_x v \leq \overline{g}_i - \sqrt{f_i P_s f_i^T} \\ -L_{u,j} v \leq -\underline{u}_j - \sqrt{H_j P_s H_j^T} \\ L_{u,j} v \leq \overline{u}_j - \sqrt{H_j P_s H_j^T} \end{array} \right. \right\}, \tag{33}$$

where $H_j$ is the $j-$th row of $H$. Define also

$$\Omega_e(P_s, H) = \Omega_x(P_s) \cap \Omega_v(P_s, H), \tag{34}$$

where $\Omega_x(P_s)$ is the parameterized ellipsoid (12) with $P = P_s$. Clearly, $\Omega_e(P_s, H)$ is a semi-ellipsoidal set.

We have the following result.

**Theorem 3:** If there exist $P_s \succ 0$, $H \in \mathbb{R}^{n_u \times n_x}$ such that

$$\begin{bmatrix} P_s & A_m P_s + B_m Y \\ * & P_s \end{bmatrix} \succeq 0, \forall m \in \overline{1, m_u}, \tag{35}$$

and that the set $\Omega_v(P_s, H)$ is non-empty, where $Y = HP_s$, and

$$A_m = A + BS_m K, B_m = BS_m^-, \forall m \in \overline{1, m_u},$$

then the set $\Omega_e(P_s, H)$ is invariant and constraint-admissible for (32), (8).

**Proof:** Following the same arguments as the ones of the proofs of Theorem 1, it is straightforward to prove that $\Omega_v(P_s, H)$ is invariant, and that $\Omega_e(P_s, H)$ is constraint-admissible. Hence we skip these proofs here.

It remains to show that $\Omega_x(P_s)$ is invariant. Define $N = L_u - HL_x$. Using Lemma 1, for system (2) under the saturated control law (30), one has, for any $x, v$ such that $\underline{u}_j \leq H_j x + N_j v \leq \overline{u}_j$, $j \in \overline{1, n_u}$,

$$x(k+1) \in \text{Co}\left(\mathbf{A}_m \left(x(k) - L_x v(k)\right) + B u_s(k)\right), \tag{36}$$

where $u_s(k) = L_u u(k)$ and

$$\mathbf{A}_m = A + BS_m K + BS_m^- H, \forall m \in \overline{1, m_u}.$$

Using (4), (5) one gets

$$x_s(k+1) = Ax_s(k) + Bu_s(k),$$

with $x_s(k+1) = L_x v(k+1), x_s(k) = L_x v(k)$. Thus, with (36)

$$x(k+1) - x_s(k+1) \in \mathrm{Co}\left(\mathbf{A}_m \left(x(k) - x_s(k)\right)\right). \tag{37}$$

Recall that $Y = HP_s$. Rewrite (35) as

$$\begin{bmatrix} P_s & \mathbf{A}_m P_s \\ * & P_s \end{bmatrix} \succeq 0, \forall m \in \overline{1, m_u}. \tag{38}$$

It is well known [2] that conditions (38) assure the invariance of $\mathcal{E}(P_s)$ for system (37), i.e., for any $x(k), v(k)$ such that

$$(x(k) - L_x v(k))^T P_s^{-1} (x(k) - L_x v(k)) \leq 1,$$

one has

$$(x(k+1) - L_x v(k+1))^T P_s^{-1} (x(k+1) - L_x v(k+1)) \leq 1.$$

Hence $\Omega_x(P_s)$ is invariant for (32). □

**Remark 5:** It is clear that if $H = K$, then $\Omega_e(P_s, H) = \Omega_e(P_s)$. In this case, one also has

$$A + BS_m K + BS_m^- K = A + BK, \forall m \in \overline{1, m_u}.$$

It follows that (35) becomes (15) with $P_s = P, Y = KP_s$. Consequently, compared to the linear control law (6), the use of the saturated one (30) in conjunction with the LDI framework introduces $H$ as an additional degree of freedom.

We use $H$ and $P_s$ as decision variables to maximize the size of $\Omega_e(P_s, H)$. For this purpose, we select a particular value of $v = v_f, v_f \in \Omega_v(P_s, H)$. Using (33) and as

$$f_i P_s f_i^T \geq 0, H_j P_s H_j^T \geq 0,$$

for any $i \in \overline{1, l_x}$, $j \in \overline{1, n_u}$, it follows that (23) provides a necessary condition of $v_f$ for the existence of $P_s$ and $H$.

For any $v_f$ satisfying (23), using the first two equation of (33), one gets

$$\begin{cases} f_i P_s f_i^T \leq (f_i L_x v_f - \underline{g}_i)^2, \\ f_i P_s f_i^T \leq (f_i L_x v_f - \overline{g}_i)^2, \end{cases} \forall i \in \overline{1, l_x}, \tag{39}$$

Using the last two equation of (33), one obtains

$$\begin{cases} H_j P_s H_j^T \leq (L_{u,j} v_f - \underline{u}_j)^2, \\ H_j P_s H_j^T \leq (L_{u,j} v_f - \overline{u}_j)^2, \end{cases} \forall j \in \overline{1, n_u},$$

thus, using the Schur complement and $Y = HP_s$

$$\begin{cases} \begin{bmatrix} (L_{u,j} v_f - \underline{u}_j)^2 & Y_j \\ Y_j^T & P_s \end{bmatrix} \succeq 0, \\ \begin{bmatrix} (L_{u,j} v_f - \overline{u}_j)^2 & Y_j \\ Y_j^T & P_s \end{bmatrix} \succeq 0 \end{cases} \forall j \in \overline{1, n_u}, \tag{40}$$

where $Y_j$ is the $j-$th row of $Y$, $\forall j \in \overline{1, n_u}$.

The problem of optimizing $P_s, H$ can be formulated as

$$\begin{cases} \min\limits_{P_s, Y} f(P_s) \\ \text{s.t. } (35), (39), (40), \end{cases} \tag{41}$$

where $f(\cdot)$ can be any convex function that maximizes the volume of the set $\Omega_x(P_s)$ for a given and fixed value of $v_f$. Note that (41) is a SDP program.

Concerning the scaling operator, using similar arguments as the ones in Section 3.2, it is possible to show that if $\Omega_e(P_s, H)$ is invariant and constraint-admissible for (32), (8), then so is the set

$$\Omega_e(\alpha, P_s, H) = \Omega_x(\alpha, P_s) \cap \Omega_v(\alpha, P_s, H), \tag{42}$$

where $\Omega_x(\alpha, P_s)$ is defined in (27) with $P = P_s$, $\alpha \geq 0$, and

$$\Omega_v(\cdot) = \left\{ v \,\middle|\, F_v v \leq g - F_\alpha \alpha \right\}, \tag{43}$$

with

$$F_v = \begin{bmatrix} -f_1 L_x \\ \vdots \\ -f_{l_x} L_x \\ f_1 L_x \\ \vdots \\ f_{l_x} L_x \\ -L_{u,1} \\ \vdots \\ -L_{u,n_u} \\ L_{u,1} \\ \vdots \\ L_{u,n_u} \end{bmatrix}, g = \begin{bmatrix} -\underline{g}_1 \\ \dots \\ -\underline{g}_{l_x} \\ \overline{g}_1 \\ \dots \\ \overline{g}_{l_x} \\ -\underline{u}_1 \\ \dots \\ -\underline{u}_{n_u} \\ \overline{u}_1 \\ \dots \\ \overline{u}_{n_u} \end{bmatrix}, F_\alpha = \begin{bmatrix} \sqrt{f_1 P_s f_1^T} \\ \vdots \\ \sqrt{f_{l_x} P_s f_{l_x}^T} \\ \sqrt{f_1 P_s f_1^T} \\ \vdots \\ \sqrt{f_{l_x} P_s f_{l_x}^T} \\ \sqrt{H_1 P_s H_1^T} \\ \vdots \\ \sqrt{H_{n_u} P_s H_{n_u}^T} \\ \sqrt{H_1 P_s H_1^T} \\ \vdots \\ \sqrt{H_{n_u} P_s H_{n_u}^T} \end{bmatrix}. \tag{44}$$

## 5 Semi-Ellipsoidal Set with Time-Varying Soft Constraints

Perhaps the most interesting aspect of the results in Section 3 and in Section 4 is that they can be easily extended to cope with time-varying soft constraints. This is because

- The semi-ellipsoidal set $\Omega_e(\alpha, P)/\Omega_e(\alpha, P_s, H)$ is the intersection of the polyhedral set $\Omega_v(\alpha, P)/\Omega_v(\alpha, P_s, H)$ and the unbounded ellipsoid $\Omega_x(\alpha, P)/\Omega_x(\alpha, P_s)$.
- The set $\Omega_v(\alpha, P)/\Omega_v(\alpha, P_s, H)$ is formed explicitly by the constraints of $X, U$.

From this point on we will focus only on semi-ellipsoidal set with saturated feedback. This is because the results in Section 3 are special cases of the results in Section 4. At time instant $k$, the state constraint set $\mathcal{X}(k)$ is

$$\mathcal{X}(k) = \{x | \underline{g}_i(k) \leq f_i(k)x \leq \overline{g}_i(k), i \in \overline{1, l_x(k)}\}. \tag{45}$$

Note that not only $f_i(k), \underline{g}_i(k), \overline{g}_i(k)$ but also the number of constraints $l_x(k)$ of $X(k)$ are time-varying.

Clearly, if $\mathcal{X}(k) \subseteq \mathcal{X}(k+1)$, then it is quite simple to guarantee the recursive feasibility in CG. Otherwise, if $\mathcal{X}(k)$ varies arbitrarily, it becomes difficult - if not impossible - to design a control strategy that guarantees the recursive feasibility. A way to cope with this problem is to relax the state constraints

(45) as

$$\mathcal{X}(k) = \{x | \underline{g}_i(k) - \epsilon_i(k) \leq f_i(k)x \leq \overline{g}_i(k) + \epsilon_i(k)\}, \tag{46}$$

where $\epsilon_i(k) \geq 0, i \in \overline{1, l_x(k)}$ are slack variables. Note that in addition to guaranteeing the recursive feasibility, the use of $\epsilon_i(k)$ also allows to trade-off constraint violation against the improvements in tracking performance.

Consider any matrices $P_s, H$ satisfying (35), $Y = HP_s$. Define

$$\epsilon(k) = [\epsilon_1(k)\ \epsilon_2(k)\ \ldots \epsilon_{l_x}(k)]^T. \tag{47}$$

Define also the set $\Omega_v(\epsilon(k), \alpha, P_s, H)$ as

$$\Omega_v(\cdot) = \left\{ v \,\middle|\, F_v(k)v \leq g(k) - F_\alpha(k)\alpha - F_\epsilon\epsilon(k) \right\}, \tag{48}$$

where $F_v(k), F_\alpha(k)$ and $g(k)$ are defined in (44) with time-varying $f_i(k), \underline{g}_i(k), \overline{g}_i(k), \forall i \in \overline{1, l_x(k)}$, and

$$F_\epsilon = [-\mathbf{I} \ldots -\mathbf{I}\ -\mathbf{I} \ldots -\mathbf{I}\ \mathbf{0}\ \ldots\ \mathbf{0}\ \mathbf{0}\ \ldots\ \mathbf{0}]^T.$$

Define

$$\Omega_e(\epsilon(k), \alpha, P_s, H) = \Omega_x(\alpha, P_s) \cap \Omega_v(\epsilon(k), \alpha, P_s, H). \tag{49}$$

The following corollary is a direct consequence of the results in Section 4.

**Corollary 1:** For any $\alpha \geq 0$, $\epsilon(k) \geq \mathbf{0}$ such that $\Omega_v(\epsilon(k), \alpha, P_s, H)$ is non-empty, the set $\Omega_e(\epsilon(k), \alpha, P_s, H)$ is invariant for (32) and constraint-admissible for the input constraints (3), and for the state constraints (46).

## 6 Online Optimization Problem

### *6.1 Semi-Ellipsoidal Set Based Command Governor*

The CG based on the set $\Omega_e(\epsilon, \alpha, P_s, H)$ for time-varying soft constraints (46) requires the online solution of the following optimization problem

$$\begin{aligned} &\min_{v,\alpha,\epsilon} \left( (v - r(k))^T Q_t (v - r(k)) + \epsilon^T Q_\epsilon \epsilon \right) \\ &\text{s.t.} \begin{cases} [x(k)^T\ v^T]^T \in \Omega_e(\epsilon(k), \alpha, P_s, H), \\ \alpha \geq 0, \\ \epsilon \geq \mathbf{0}, \end{cases} \end{aligned} \tag{50}$$

where $x(k)$ is the current state; $Q_t \succ 0, Q_\epsilon \succ 0$ are weighting matrices. The first term in the cost function penalizes the deviation of the command from the

desired reference. The second term penalizes the violation of the constraints. Note that we consider quadratic cost for $\epsilon$ in (50). However, linear cost can also be applied, depending on the specific requirements of the problem.

Once the optimal solution $(v^*(k), \alpha^*(k), \epsilon^*(k))$ of (50) is found, the control action is computed as

$$u(k) = \text{sat}(Kx(k) + Lv^*(k)). \tag{51}$$

Using $\epsilon$, it is clear that the optimization problem (50) is always feasible. Concerning the finite time convergence for the command $v$, consider the case when there exists $k_s$ such that, $\forall k \geq k_s$

$$\begin{cases} f_i(k) = f_i(k_s), \underline{g}_i(k) = \underline{g}_i(k_s), \\ \overline{g}_i(k) = \overline{g}_i(k_s), r(k) = r(k_s). \end{cases} \tag{52}$$

Denote $(v^*(k_s), \alpha^*(k_s), \epsilon^*(k_s)))$ as the optimal solution of the following problem

$$\begin{aligned} &\min_{v,\alpha,\epsilon} \Big( (v - r(k_s))^T Q_t (v - r(k_s)) + \epsilon^T Q_\epsilon \epsilon \Big) \\ &\text{s.t.} \begin{cases} v \in \Omega_v(\epsilon(k), \alpha, P_s, H), \\ \alpha \geq 0, \\ \epsilon \geq \mathbf{0}. \end{cases} \end{aligned} \tag{53}$$

Note that the control scheme (50), (51) to satisfy the state constraints in a soft way is an extension of the regular CG theory. Hence, if (52) holds for any $k \geq k_s$, then there exists a time index $k_f$ such that, $\forall k \geq k_f$

$$v^*(k) = v^*(k_s), \alpha^*(k) = \alpha^*(k_s), \epsilon^*(k) = \epsilon^*(k_s).$$

If $\exists \alpha \geq 0$ such that $r(k_s) \in \Omega_v(\mathbf{0}, \alpha, P_s, H)$, then $v^*(k_s) = r(k_s), \epsilon^*(k_s) = \mathbf{0}$ is the trivial solution of (53). This means that the command converges to the desired reference in finite time. If $\nexists \alpha \geq 0$ such that $r(k_s) \in \Omega_v(\mathbf{0}, \alpha, P_s, H)$, then $v^*(k_s)$ may not coincide with $r(k_s)$. The optimal solution $(v^*(k_s), \epsilon^*(k_s)))$ of (53) is the trade-off between tracking performance and constraint violation.

Concerning the online optimization problem (50), clearly, it is a convex program. In the following we provide a way to solve efficiently (50). Our solver, which is based on the alternating direction method of multipliers (ADMM), can exploit the particular structure of $\Omega_e(\cdot)$. Hence, our solver can solve (50) extremely fast via very simple mathematical operations. In the last decade, ADMM has emerged as an effective algorithm for solving structured convex optimization problems [3]. Our main contribution in this section is to show how to convert (50) into a form that the sub-optimization problems associated

with the ADMM can be solved efficiently. To make the paper self-contained, we recall the ADMM theory in the next section.

### *6.2 Alternating Direction Method of Multipliers*

Consider the following optimization problem

$$\begin{aligned} &\min_{s,z} h(s) \\ &\text{s.t.} \begin{cases} Ms - z = b, \\ z \in \mathcal{Z}, \end{cases} \end{aligned} \tag{54}$$

where $h(s)$ is a convex function, the matrix $M$ and the vector $b$ are of appropriate dimension, $\mathcal{Z}$ is a convex set. One way to solve (54) is to form the augmented Lagrangian

$$\begin{aligned} \mathcal{L}_\rho(s,z,y) = h(s) + \eta^T(Ms - z - b) \\ +\tfrac{\rho}{2}(Ms - z - b)^T(Ms - z - b), \end{aligned} \tag{55}$$

where $\eta$ is the Lagrange multiplier, $\rho > 0$ is a tunning parameter that presents the trade-off between the cost function and the equality constraints.

ADMM works by solving iteratively two sub-problems and then updating the Lagrange multiplier. At iteration $q$ we carry out the following steps.

(1) **Step 1**: minimize $\mathcal{L}_\rho(s,z,\eta)$ with respect to $s$

$$s^{(q+1)} := \arg\min_s \left\{ \mathcal{L}_\rho(s, z^{(q)}, \eta^{(q)}) \right\}. \tag{56}$$

(2) **Step 2**: minimize $\mathcal{L}_\rho(s,z,\eta)$ with respect to $z$

$$z^{(q+1)} := \arg\min_{z\in\mathcal{Z}} \left\{ \mathcal{L}_\rho(s^{(q+1)}, z, \eta^{(q)}) \right\}. \tag{57}$$

(3) **Step 3**: update the Lagrange multiplier

$$\eta^{(q+1)} = \eta^{(q)} + \rho(Ms^{(q+1)} - z^{(q+1)} - b). \tag{58}$$

We use the superscript $(q)$ in (56), (57), (58) to denote the values of variables calculated at iteration $q$.

The ADMM is particularly useful when it is possible to solve (56), (57) efficiently. The main contribution of this section is to show how to convert (50) into a form that (56), (57) admit closed-form expressions.

The primal and dual residuals at iteration $q$ are

$$\zeta_p^{(q)} = Ms^{(q)} - z^{(q)} - b,\ \zeta_d^{(q)} = z^{(q)} - z^{(q-1)}. \tag{59}$$

The algorithm is terminated when the primal and dual residuals satisfy a stopping criterion. A typical criterion is $||\zeta_p^{(q)}||_\infty \leq \zeta_p$, $||\zeta_d^{(q)}||_\infty \leq \zeta_d$, where $\zeta_p, \zeta_d$ are given tolerances.

### *6.3 Solving the Online Optimization Problem of the Semi-Ellipsoidal Set Based CG Using ADMM*

We reformulate (50) as the optimization problem (54) as follows. Let us take

$$s = \begin{bmatrix} v^T\ \alpha\ \epsilon^T \end{bmatrix}^T, \tag{60}$$

thus, using (50), the cost function $h(s)$ is

$$h(s) = \frac{1}{2} s^T \mathcal{Q} s - r(k)^T Q_0^T s, \tag{61}$$

with

$$\mathcal{Q} = \begin{bmatrix} Q_t & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & Q_\epsilon \end{bmatrix}, Q_0 = \begin{bmatrix} Q \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}.$$

Define the following auxiliary variables

$$\begin{cases} z_1 = D_s \left( x(k) - L_x v \right),\ z_2 = \alpha, \\ z_3 = F_v v + F_\alpha \alpha + F_\epsilon \epsilon,\ z_4 = \epsilon, \\ z = [z_1^T\ z_2\ z_3^T\ z_4]^T, \end{cases} \tag{62}$$

where $D_s \in \mathbb{R}^{n_x \times n_x}$is a square root of $P_s^{-1}$, i.e., $D_s^T D_s = P_s^{-1}$. Using (60), (62) one has

$$Ms - z = b_0 x(k), \tag{63}$$

with

$$M = \begin{bmatrix} -D_s L_x & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 1 & \mathbf{0} \\ F_v & F_\alpha & F_\epsilon \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}, b_0 = \begin{bmatrix} -D_s \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}.$$

Using (49), (50), one obtains the set $\mathcal{Z}$

$$\mathcal{Z} = \left\{ z \left| \begin{array}{l} z_1^T z_1 \leq z_2^2,\ z_2 \geq 0, \\ z_3 \leq g(k),\ z_4 \geq \mathbf{0}. \end{array} \right. \right\} \tag{64}$$

Hence problem (50) is reformulated as (54) with $h(s), M, b = b_0 x(k)$, and $\mathcal{Z}$ being given in (61), (63), (64), respectively.

Using (56), (57), (58), we carry out the following steps in each iteration.

**Step 1:** The optimization problem is

$$\begin{array}{l} \min\limits_{s} \Big( \frac{1}{2} s^T (\mathcal{Q} + \rho M^T M) s \\ \qquad - \Big( M^T (\rho z^{(q)} - \eta^{(q)} + \rho b_0 x(k)) + Q_0 r(k) \Big)^T s \Big). \end{array} \tag{65}$$

The optimization problem (65) is an unconstrained quadratic program. The solution to (65) is given explicitly as

$$s^{(q+1)} = \mathcal{M} \left( M^T (\rho z^{(q)} - \eta^{(q)} + \rho b_0 x(k)) + Q_0 r(k) \right), \tag{66}$$

where $\mathcal{M} = (\mathcal{Q} + \rho M^T M)^{-1}$.

**Step 2:** The optimization problem is

$$\begin{cases} \min\limits_{z} \left( z^T z - 2(M s^{(q+1)} + \frac{\eta^{(q)}}{\rho} - b_0 x(k))^T z \right) \\ \text{s.t. } z \in \mathcal{Z}. \end{cases} \tag{67}$$

Define

$$\nu = M s^{(q+1)} + \frac{\eta^{(q)}}{\rho} - b_0 x(k). \tag{68}$$

Note that $\nu$ and $z$ are of the same dimension. We rewrite $\nu$ as

$$\nu = [\nu_1^T \ \nu_2 \ \nu_3^T \ \nu_4^T]^T,$$

where $\nu_i$ has the same dimension of $z_i$, $i \in \overline{1,4}$.

The cost function and the constraints of (67) are separable in $z_i, i \in \overline{1,4}$. We can carry our their update in parallel. The update of $(z_1, z_2)$ is the solution of the following problem

$$\begin{cases} \min\limits_{z_1, z_2} (z_1^T z_1 + z_2^2 - 2\nu_1^T z_1 - 2\nu_2 z_2) \\ \text{s.t. } z_1^T z_1 \leq z_2^2, z_2 \geq 0. \end{cases} \tag{69}$$

By using the method of Lagrange multipliers, the solution of (69) can be calculated analytically as [19]

$$\begin{cases} \text{If } \nu_1^T\nu_1 \leq \nu_2^2, \nu_2 < 0 \\ \quad z_1^{(q+1)} = \mathbf{0}, z_2^{(q+1)} = 0, \\ \text{If } \nu_1^T\nu_1 \leq \nu_2^2, \nu_2 \geq 0 \\ \quad z_1^{(q+1)} = \nu_1, z_2^{(q+1)} = \nu_2, \\ \text{Otherwise} \\ \quad z_1^{(q+1)} = \frac{\nu_2+\sqrt{\nu_1^T\nu_1}}{2\sqrt{\nu_1^T\nu_1}}\nu_1, z_2^{(q+1)} = \frac{\nu_2+\sqrt{\nu_1^T\nu_1}}{2}. \end{cases} \tag{70}$$

The update of $z_3$ is the solution of the optimization problem

$$\begin{cases} \min\limits_{z_3}(z_3^T z_3 - 2\nu_3^T z_3) \\ \text{s.t. } z_3 \leq g(k). \end{cases} \tag{71}$$

The solution of (71) can be given in the closed-form as

$$z_3^{(q+1)} = \min(\nu_3, g(k)), \tag{72}$$

where the min operator is taken component-wise.

Analogously, the update of $z_4$ is given as

$$z_4^{(q+1)} = \max(\nu_4, \mathbf{0}), \tag{73}$$

For the given stopping tolerances $\epsilon_d, \epsilon_p$ and initial point $(s^{(0)}, z^{(0)}, \eta^{(0)})$, Algorithm 2 shows the ADMM algorithm applied to the problem (50)

Algorithm 2: ADMM Based Solver for (50)

1: Update $s^{(q+1)}$ using (66).
2: Update $z^{(q+1)}$ using (70), (72), (73).
3: Update $\eta^{(q+1)}$ as

$$\eta^{(q+1)} \leftarrow \eta^{(q)} + \rho(Ms^{(q+1)} - z^{(q+1)} - b_0 x(k))$$

**Remark 6:** Using Algorithm 2, the updates of $s^{(q)}$ and $z^{(q)}$ are given explicitly with the use of the auxiliary variables (62). The use of other types of auxiliary variables, e.g., $z_3 = d_2 - L_x\theta_2$ instead of (62), would lead to a sub-optimization problem of the ADMM, which does not have an explicit solution.

## 7 Examples

This section illustrates the potential benefit of the new methods by simulations of three examples system. The CVX toolbox [8] was used to solve SDP optimization problems. For comparison purpose, we denote the standard polyhedral based CG, the semi-ellipsoidal set based CG with linear feedback, and with saturated feedback as StdCG, LiCG, and SatCG, respectively. We use an Intel Core i7-10610U 1.8GHz to evaluate the algorithms.

### *7.1 Example 1: Time-Invariant Hard Constraints*

This example is taken from [4]. Consider system (2) with

$$A = \begin{bmatrix} 1 & 0.1 \\ 0 & 1 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0.1 \end{bmatrix}, C = \begin{bmatrix} 1 & 0 \end{bmatrix}. \tag{74}$$

The gain $K$ is given as

$$K = \begin{bmatrix} -4.2674 & -3.1426 \end{bmatrix}. \tag{75}$$

The input and state constraints are $-0.2 \leq u \leq 0.2$, $-1 \leq x_1 \leq 2$.

By solving the SDP problem (25) with $v_f = \mathbf{0}$, one gets

$$P = \begin{bmatrix} 0.0118 & -0.0184 \\ -0.0184 & 0.0322 \end{bmatrix}.$$

By solving the SDP problem (41) with $v_f = \mathbf{0}$, one obtains

$$\begin{aligned} P_s &= \begin{bmatrix} 0.0947 & -0.0708 \\ -0.0708 & 0.1180 \end{bmatrix}, \\ H &= [-0.6141 \quad -0.7835]. \end{aligned} \tag{76}$$

For this example, we were able to calculate the maximal invariant and constraint-admissible set $\Omega_p$ with $\mu = 10^{-4}$. Fig. 1 presents the projection of $\Omega_p$, of $\Omega_e(\alpha, P)$ and of $\Omega_e(\alpha, P_s, H)$ onto the $x-$space. The projection of these sets are the feasible sets of StdCG, LiCG and SatGG. We can observe that: i) the feasible set of LiCG is slightly smaller than that of StdCG; ii) both of them are much smaller than that of SatCG.

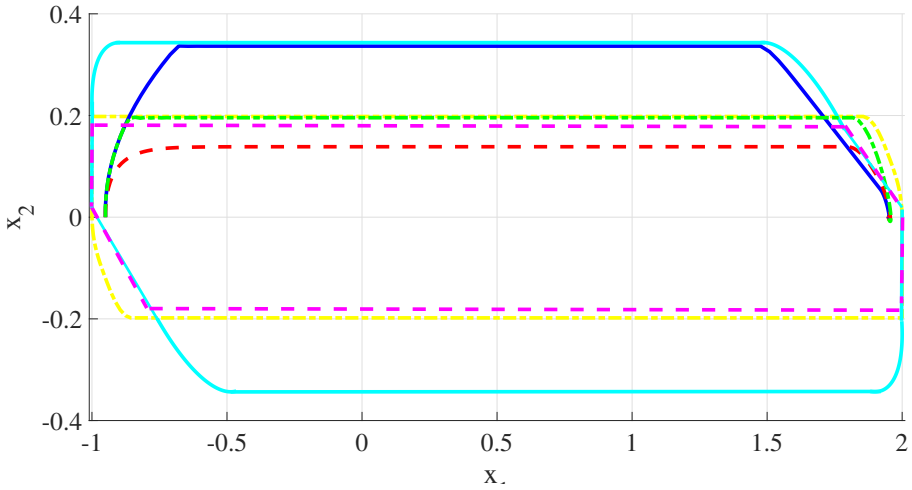



Fig. 1. Feasible sets and phase-space trajectories for example 1. Projection of $\Omega_e(\alpha, P)$ onto the $x-$space (dashed magenta), projection of $\Omega_e(\alpha, P_s, H)$ onto the $x-$space (solid cyan), projection of the maximal invariant and constraint-admissible polyhedral set onto the $x-$space (dash-dot yellow). Phase-space trajectory of StdCG (dash-dot green), of LiCG 2 (dashed red), and of SatCG 3 (solid blue).

For the initial condition $x(0) = [-0.95\ 0]^T$ and for the reference $r(k) = 1.95$, Fig 1 shows the phase-space trajectory of the three algorithms.

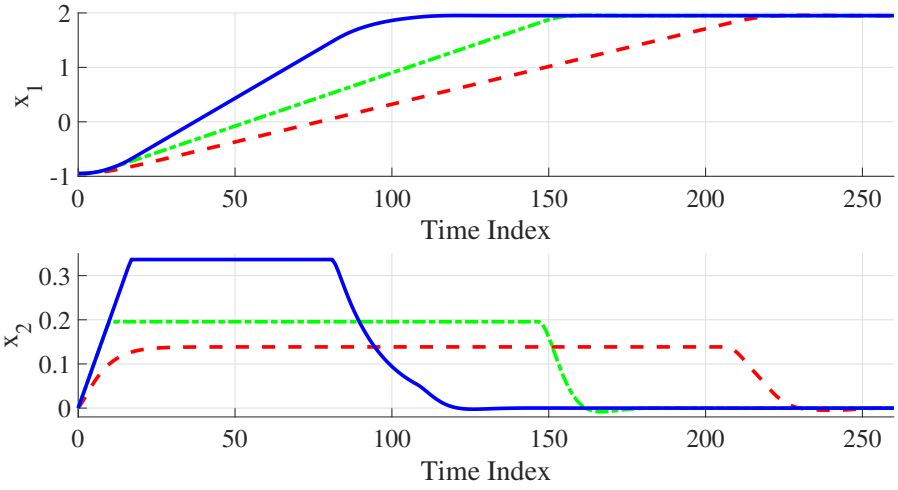
Fig. 2 presents the state, the command and the input trajectories as functions of time for the three algorithms. We observe that SatCG has the best performance in terms of transient time, followed by StdCG, then by LiCG.

Finally, using the function TIC/TOC of MATLAB 2023b, we found that the online computation time for one sampling interval was $3.1312 \times 10^{-3}$[sec], $1.7415 \times 10^{-4}$[sec], and $1.5775 \times 10^{-4}$[sec] for StdCG, LiCG, and SatCG, respectively. We use *quadprog* solver in Matlab to solve the online quadratic program of StdCG.

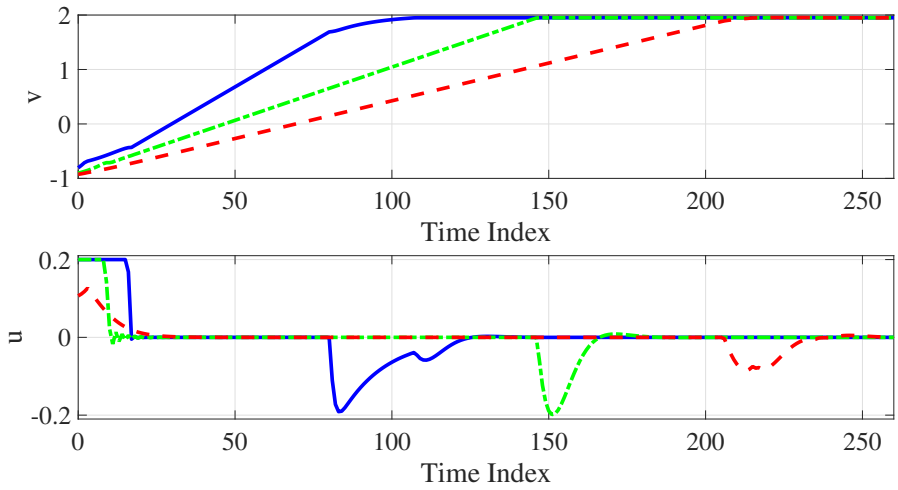
### *7.2 Example 2: Time-Varying Soft Constraints*

We consider system (2) with $A, B, C, K$ being given in (74), (75). The input constraints are $-0.2 \leq u \leq 0.2$. The objective is to design a CG based control law that tracks a time-varying piecewise constant reference $r(k)$ while managing overshoot of the output. The reference $r(k)$ is

$$r(k) = \begin{cases} 1, \text{ if } k \leq 120, \\ 2, \text{ otherwise} \end{cases} \tag{77}$$




(a) State trajectories.




(b) Command and input trajectories

Fig. 2. (a) State trajectories; (b) Command and input trajectories for StdCG (dash–dot green), for LiCG (dashed red), for SatCG (solid blue) for example 1.

For managing overshoot of the output, we impose the following constraint on $x_1$ at time $k$,

$$\begin{cases} x_1(k) \leq r(k) + \epsilon_1(k), \text{ if } x_1(0) \leq r(0), \\ x_1(k) \geq r(k) - \epsilon_1(k), \text{ if } x_1(0) \geq r(0), \end{cases} \tag{78}$$

where $\epsilon_1 \geq 0$.

To demonstrate the ability of the proposed algorithms to handle arbitrarily time-varying constraints, we consider the following constraint on $x_2$ in addition to (78)

$$x_2(k) \leq 0.3, \text{if } k \geq 120, \tag{79}$$

To guarantee that the optimization problem is always feasible, we relax the constraint (79) as

$$x_2(k) \leq 0.3 + \epsilon_2(k), \text{if } k \geq 120, \tag{80}$$

where $\epsilon_2 \geq 0$.

The initial condition is $x(k) = [0\ 0]^T$. For simplicity of discussion, we apply only SatCG in this example. We use $P_s, H$ in (76) for the set $\Omega_e(\epsilon(k), \alpha, P_s, H)$. Three difference simulations consist of three different weights for the optimization problem (50), $Q_t = 1$ and $Q_\epsilon = \text{diag}([10\ 400]), Q_\epsilon = \text{diag}([1\ 400]), Q_\epsilon = \text{diag}([0.1\ 400])$. Note that the weight of $\epsilon_2$ is significantly larger than that of the other two terms. his implies that it is desirable for $x_2(k)$ not to violate the constraint (79).

Fig. 3 presents the reference, the state, the command, and the input trajectories for SatCG with different $Q_\epsilon$. Fig. 3 also presents the simulation results when $v(k) = r(k)$ is directly applied to the system, i.e., without using CG.

For a given constant reference $r$, define the percentage overshoot as

$$\text{PO} = 100 \max_k \frac{y(k) - r}{r}$$

We can observe that

- Without using CG, the PO is about 36%.
- The PO is 6% for the CG with $Q_\epsilon = \text{diag}([0.1\ 400])$.
- The PO is 0% for the CG with $Q_\epsilon = \text{diag}([1\ 400])$, and with $Q_\epsilon = \text{diag}([10\ 400])$.
- The transient time of the CG with $Q_\epsilon = \text{diag}([1\ 400])$ is shorter than that of the CG with $Q_\epsilon = \text{diag}([10\ 400])$.
- The time-varying constraint (79) is satisfied for all three CG schemes.

### *7.3 Example 3: High-Order System*

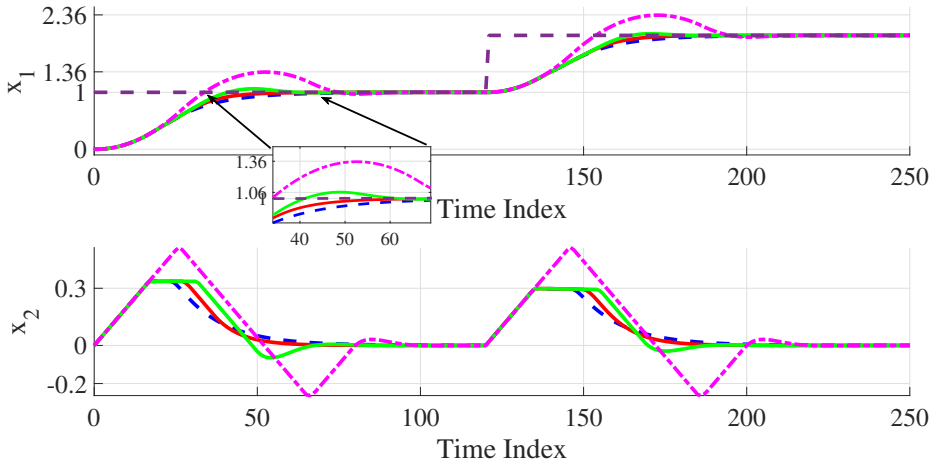
This example is inspired by the case study from [13]. We consider a system of $p$ objects connected by springs, illustrated in Fig. 4.

The mass are all taken as $m = 2$[kg]. The spring constants are $k_s = 1$[N/m]. There are two external forces acting on the system: a force $u_1$[N] acting on the first object and a force $u_2$[N] acting on the last object. The system can be described using a set of differential equations. For the first object,

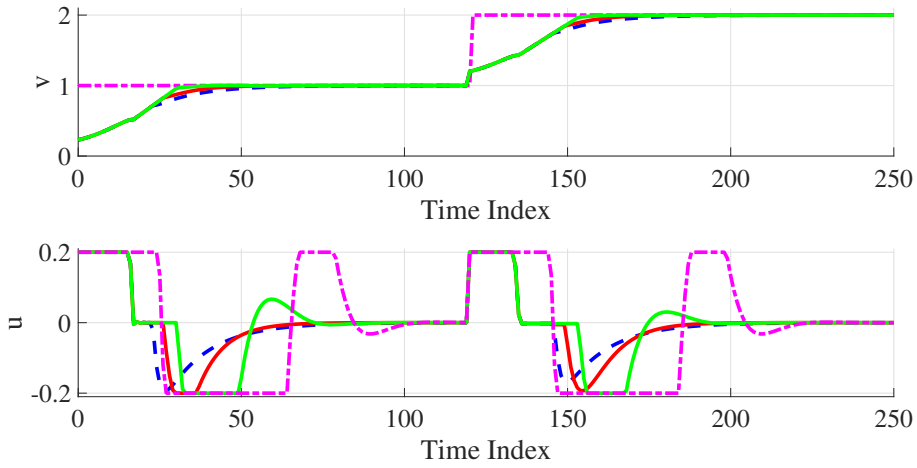
$$m\ddot{p}_1 = -2k_s p_1 + k_s p_2 + u_1. \tag{81}$$

For object $i$, $i \in \overline{2, p-1}$

$$m\ddot{p}_i = k_s p_{i-1} - 2k_s p_i + k_s p_{i+1}. \tag{82}$$




(a) Reference and state trajectories.




(b) Command and input trajectories

Fig. 3. (a) Reference (dash-dot violet) and state trajectories; (b) Command and input trajectories for SatCG with $Q_\epsilon = \text{diag}([10\ \ 400])$ (dashed blue), $Q_\epsilon = \text{diag}([1\ \ 400])$ (solid red), $Q_\epsilon = \text{diag}([0.1\ \ 400])$ (solid green), and without using CG (dash-dot magenta) for example 2.


Fig. 4. Chain of $p$ masses for example 3.

For the last object,

$$m\ddot{p}_p = k_s p_{p-1} - 2k_s p_p + u_2. \tag{83}$$

The state of the system is given by the position $p_i$[m], and velocity $v_i$[m/s], of each object, $i = \overline{1,p}$, i.e.,

$$x = [p_1\ p_2\ \ldots\ p_p\ v_1\ v_2\ \ldots\ v_p]^T \tag{84}$$

The input is $u = [u_1\ u_2]^T$, and the output is $y = [p_1\ p_p]^T$.

We compute a model (2) by using a sampling time of $T_s = 0.01$[sec], and a zero-order hold. We consider the following state and input constraints: $|x_i| \leq 1, i \in \overline{1,p}$, $|u_1| \leq 1$, $|u_2| \leq 1$. For this example, we add slack variables $\epsilon_i \geq 0$ to relax the state constraints $|x_i| \leq 1, i \in \overline{1,p}$. As a result $|x_i| \leq 1+\epsilon_i, i \in \overline{1,p}$.

The feedback gain $K$ is chosen as an LQ gain with the following state and control weighting matrices

$$Q_x = \begin{bmatrix} \mathbf{I}\ \mathbf{0} \\ \mathbf{0}\ \mathbf{0} \end{bmatrix},\ Q_u = 10^{-2}\mathbf{I}.$$

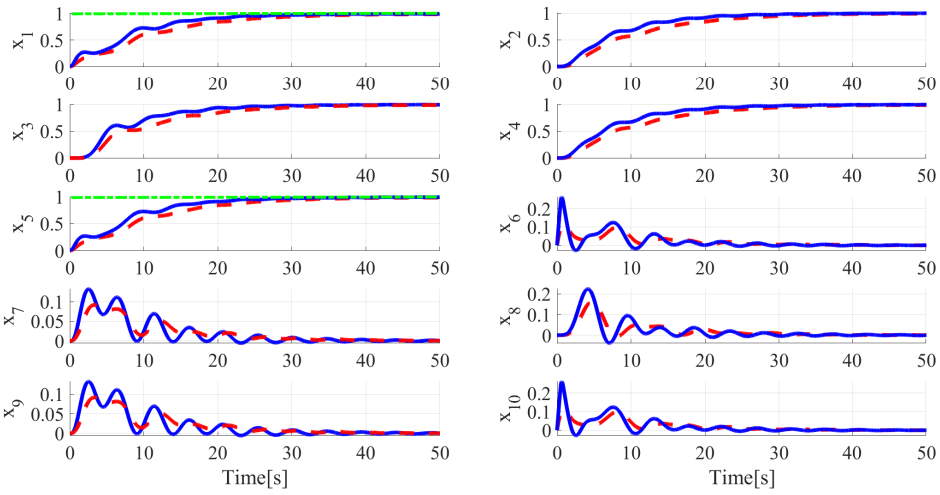
The matrices $P, P_s, H$ are optimized, respectively, by using (25), (41) with $f(P) = -\text{logdet}(P), f(P_s) = -\text{logdet}(P_s)$. Numerical values for $K, P, P_s, H$ are reported in the Appendix for $p = 5$.

Table 1 shows the number of ellipsoids and the number of linear inequalities, i.e., half-spaces required to describe $\Omega_e(\epsilon(k), \alpha, P_s, H)$ for $p = 3, 4, 5$. For comparison purpose, Table 1 also shows the number of half-spaces required to describe the maximal invariant and constraint admissible set $\Omega_p$ with $\mu = 0.01$. We used standard procedures in [7] to construct $\Omega_p$ where the state constraints are hard, i.e., $|x_i| \leq 1, i \in \overline{1,p}$. Note that we were not be able to find $\Omega_p$ for $p = 5$.

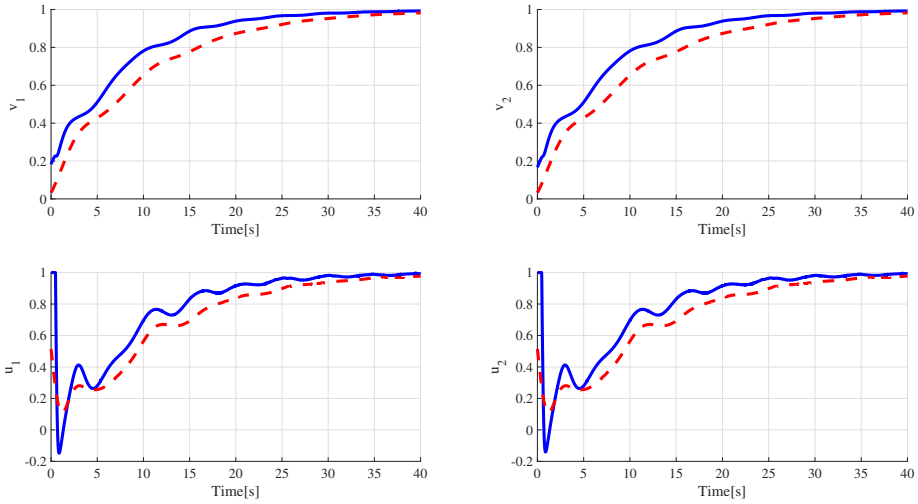
| p | 3 | 4 | 5 |
|---|---|---|---|
| Poly-hedral | 3912 half-spaces | 8968 half-spaces | ? |
| Semi -ellipsoid | 1 ellipsoid + 14 half-spaces | 1 ellipsoid + 17 half-spaces | 1 ellipsoid+ 20 half-spaces |

Table 1
Complexity of the set representation for example 3.

The weighting matrices for the optimization (50) are $Q_t = \mathbf{I}$, $Q_\epsilon = 10\mathbf{I}$. The reference is $r(k) = [1\ 1]^T$. The initial condition is zero. Using $p = 5$, Fig 5 presents the reference (dash-dot green), the state, the command and the input trajectories as functions of time for LiCG, and for SatCG.




(a) Reference and state trajectories.




(b) Command and input trajectories

Fig. 5. (a) Reference (dash-dot green) and state trajectories; (b) Command and input trajectories for LiCG (dashed red) and for SatCG (solid blue) for example 3.

Finally, using the function TIC/TOC of MATLAB 2023b, we found that the online computation time for one sampling interval was $9.7564 \times 10^{-4}$[sec], and $9.6241 \times 10^{-4}$[sec] for LiCG and for SatCG, respectively.

## 8 Conclusion

In this paper we provide new command governor schemes that employ a semi-ellipsoidal set. We address both linear and saturated feedback cases, for which we propose new linear matrix inequality conditions to construct an invariant and constraint-admissible semi-ellipsoidal set. Furthermore, we show that these conditions can be extended to handle time-varying soft constraints, providing great flexibility and adaptability. To solve the online optimization problem associated with the introduced semi-ellipsoidal set, we propose a tailored alternating direction method of multipliers based technique. The main feature of the new technique is that it requires very simple mathematical operations at each iteration, thus ensuring numerical efficiency for real-time applications. Given the fixed complexity in representing the considered semi-ellipsoidal sets, the proposed approaches are suitable for medium to large-sized systems. Three numerical examples with comparison to earlier solutions in the literature demonstrate the effectiveness of the new methods.

## References


[1] Franco Blanchini, Stefano Miani, et al. *Set-theoretic methods in control*, volume 78. Springer, 2008.

[2] Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. *Linear matrix inequalities in system and control theory*. SIAM, 1994.

[3] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[4] Andres Cotorruelo, Daniel Limon, and Emanuele Garone. Output admissible sets and reference governors: Saturations are not constraints! *IEEE Transactions on Automatic Control*, 65(3):1192–1196, 2019.

[5] Emanuele Garone, Stefano Di Cairano, and Ilya Kolmanovsky. Reference and command governors for systems with constraints: A survey on theory and applications. *Automatica*, 75:306–328, 2017.

[6] Elmer G Gilbert and Ilya Kolmanovsky. Fast reference governors for systems with state and control constraints and disturbance inputs. *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, 9(15):1117–1141, 1999.

[7] Elmer G Gilbert and K Tin Tan. Linear systems with state and control constraints: The theory and application of maximal output admissible sets. *IEEE Transactions on Automatic control*, 36(9):1008–1020, 1991.

[8] Michael Grant, Stephen Boyd, and Yinyu Ye. Cvx: Matlab software for disciplined convex programming, 2008.

[9] Tingshu Hu, Zongli Lin, and Ben M Chen. Analysis and design for discrete-time linear systems subject to actuator saturation. *Systems & control letters*, 45(2):97–112, 2002.

[10] Lars Imsland, Nadav Bar, and Bjarne A Foss. More efficient predictive control. *Automatica*, 41(8):1395–1403, 2005.

[11] Uroš Kalabić, Yash Chitalia, Julia Buckland, and Ilya Kolmanovsky. Prioritization schemes for reference and command governors. In *2013 European Control Conference (ECC)*, pages 2734–2739. IEEE, 2013.

[12] Uroš Kalabić and Ilya Kolmanovsky. Reference and command governors for systems with slowly time-varying references and time-dependent constraints. In *53rd IEEE conference on decision and control*, pages 6701–6706. IEEE, 2014.

[13] Markus Kögel and Rolf Findeisen. A fast gradient method for embedded linear predictive control. *IFAC Proceedings Volumes*, 44(1):1362–1367, 2011.

[14] Benoît Legat, Saša V Raković, and Raphaël M Jungers. Piecewise semi-ellipsoidal control invariant sets. *IEEE Control Systems Letters*, 5(3):755–760, 2020.

[15] Daniel Limón, Ignacio Alvarado, Teodoro Alamo, and Eduardo F Camacho. Mpc for tracking piecewise constant references for constrained linear systems. *Automatica*, 44(9):2382–2387, 2008.

[16] Johan Lofberg. Yalmip: A toolbox for modeling and optimization in matlab. In *2004 IEEE international conference on robotics and automation (IEEE Cat. No. 04CH37508)*, pages 284–289. IEEE, 2004.

[17] Hoai-Nam Nguyen. Constrained control of uncertain, time-varying, discrete-time systems. *Lecture Notes in Control and Information Sciences*, 451:17, 2014.

[18] Hoai-Nam Nguyen. Ellipsoidal set based extended command governor for constrained linear systems with bounded disturbances. *IEEE Transactions on Automatic Control*, 2022.

[19] Hoai-Nam Nguyen. Improved prediction dynamics for robust mpc. *IEEE Transactions on Automatic Control*, 2022.

[20] Brian D O'Dell and Eduardo A Misawa. Semi-ellipsoidal controlled invariant sets for constrained linear systems. *J. Dyn. Sys., Meas., Control*, 124(1):98–103, 2002.

**APPENDIX:** Numerical values for $K, P, P_s, H$ for $p = 5$ in example 3.

$$K = -\begin{bmatrix} 12.063 & -1.316 & 1.339 & 1.316 & 1.153 & 6.939 & 13.623 & 5.390 & 4.426 & 0.330 \\ 1.153 & 1.316 & 1.339 & -1.316 & 12.063 & 0.330 & 4.426 & 5.390 & 13.623 & 6.939 \end{bmatrix},$$

$$P = \begin{bmatrix} 0.38 & 0.09 & -0.09 & -0.02 & 0.06 & -0.01 & -0.35 & 0.10 & -0.01 & -0.01 \\ 0.09 & 0.55 & 0.23 & -0.00 & -0.02 & 0.29 & -0.13 & -0.13 & 0.02 & -0.01 \\ -0.09 & 0.23 & 1.00 & 0.23 & -0.09 & -0.13 & 0.05 & -0.03 & 0.05 & -0.13 \\ -0.02 & -0.00 & 0.23 & 0.55 & 0.09 & -0.01 & 0.02 & -0.13 & -0.13 & 0.29 \\ 0.06 & -0.02 & -0.09 & 0.09 & 0.38 & -0.01 & -0.01 & 0.10 & -0.35 & -0.01 \\ -0.01 & 0.29 & -0.13 & -0.01 & -0.01 & 0.36 & -0.05 & -0.18 & 0.02 & 0.10 \\ -0.35 & -0.13 & 0.05 & 0.02 & -0.01 & -0.05 & 0.39 & -0.20 & -0.01 & 0.02 \\ 0.10 & -0.13 & -0.03 & -0.13 & 0.10 & -0.18 & -0.20 & 0.67 & -0.20 & -0.18 \\ -0.01 & 0.02 & 0.05 & -0.13 & -0.35 & 0.02 & -0.01 & -0.20 & 0.39 & -0.05 \\ -0.01 & -0.01 & -0.13 & 0.29 & -0.01 & 0.10 & 0.02 & -0.18 & -0.05 & 0.36 \end{bmatrix},$$

$$H = -\begin{bmatrix} 1.835 & -0.743 & 0.554 & -0.051 & 0.398 & 1.800 & 1.648 & 0.864 & 0.719 & 0.202 \\ 0.371 & -0.056 & 0.535 & -0.744 & 1.807 & 0.185 & 0.695 & 0.875 & 1.634 & 1.814 \end{bmatrix},$$

$$P_s = \begin{bmatrix} 0.69 & 0.29 & -0.14 & -0.08 & 0.02 & -0.04 & -0.56 & 0.16 & 0.04 & -0.01 \\ 0.29 & 1.00 & 0.15 & -0.24 & -0.07 & 0.38 & -0.18 & -0.12 & 0.02 & -0.06 \\ -0.14 & 0.16 & 1.00 & 0.23 & -0.10 & -0.15 & 0.09 & -0.07 & 0.07 & -0.14 \\ -0.08 & -0.24 & 0.23 & 1.10 & 0.32 & -0.06 & 0.01 & -0.12 & -0.21 & 0.40 \\ 0.01 & -0.07 & -0.10 & 0.32 & 0.71 & -0.01 & 0.05 & 0.15 & -0.57 & -0.04 \\ -0.04 & 0.38 & -0.15 & -0.06 & -0.01 & 0.70 & -0.04 & -0.28 & 0.03 & 0.10 \\ -0.56 & -0.18 & 0.09 & 0.01 & 0.05 & -0.04 & 0.66 & -0.25 & -0.16 & 0.02 \\ 0.16 & -0.12 & -0.07 & -0.12 & 0.15 & -0.28 & -0.25 & 0.92 & -0.24 & -0.27 \\ 0.04 & 0.02 & 0.07 & -0.21 & -0.57 & 0.03 & -0.16 & -0.24 & 0.67 & -0.04 \\ -0.01 & -0.06 & -0.14 & 0.40 & -0.04 & 0.10 & 0.02 & -0.27 & -0.04 & 0.69 \end{bmatrix}$$