Linear Model Predictive Control under Continuous Path Constraints via Parallelized Primal-Dual Hybrid Gradient Algorithm

Zishuo Li[1], Bo Yang[1], Jiayun Li[1], Jiaqi Yan[2], Yilin Mo[1]

***Abstract*— In this paper, we consider a Model Predictive Control (MPC) problem of a continuous-time linear time-invariant system subject to continuous-time path constraints on the states and the inputs. By leveraging the concept of differential flatness, we can replace the differential equations governing the system with linear mapping between the states, inputs, and flat outputs (including their derivatives). The flat outputs are then parameterized by piecewise polynomials, and the model predictive control problem can be equivalently transformed into a Semi-Definite Programming (SDP) problem via Sum-of-Squares (SOS), ensuring constraint satisfaction at every continuous-time interval. We further note that the SDP problem contains a large number of small-size semi-definite matrices as optimization variables. To address this, we develop a Primal-Dual Hybrid Gradient (PDHG) algorithm that can be efficiently parallelized to speed up the optimization procedure. Simulation results on a quadruple-tank process demonstrate that our formulation can guarantee strict constraint satisfaction, while the standard MPC controller based on the discretized system may violate the constraint inside a sampling period. Moreover, the computational speed superiority of our proposed algorithm is collaborated by the simulation.**

I. Introduction

The optimal control theory aims to find control laws for a dynamical system in order to optimize a given objective function, which finds numerous applications in fields of engineering [1]–[3] and economics [4]–[6] etc. Closed-form optimal control law can be found for certain unconstrained problems, such as linear-quadratic control problem [7], or brachistochrone problem [8]. However, analytically solving the optimal control problem of continuous-time systems remains a challenging task. Furthermore, a vast majority of real-world dynamical systems operate under various constraints, such as input saturation or safety constraint on the state. For constrained optimal control problem, Pontryagin's maximum principle [9] can be used to derive necessary condition for optimality. However, in practice, only a small number of problems can be solved analytically. Therefore, algorithms, such as model predictive control, discretize the system and thus reducing the search space of the control input from the infinite dimensional function space into a finite dimensional space, where numerical optimization can be used.

Dynamic Matrix Control (DMC) [10] and Model Algorithmic Control (MAC) [11] are two formulations of MPC algorithm for discretized optimal control problems with constraints [12]. Both formulations employ a zero-order hold for the control inputs, which implies that the control inputs are step functions and hence reside in a finite dimensional space. However, the discretization of a continuous time system means that one can only guarantee constraint satisfaction at all discrete-time instant, where constraint violation can occur in between.

For control applications with high safety requirements, constraints violations can be intolerable. In order to meet the constraints at all time, Semi-Infinite Programming (SIP) [13] has been used to deal with infinite number of constraints. Several approaches for solving SIP have been proposed, and a common framework is to check constraint violations in intervals, and adaptively add additional discrete-time points until the tolerance level is guaranteed or no constraint violations occur. Chen et al [14] introduce ϵ-tolerance on inequality constraints, which means that the constraints may still be violated up to ϵ. Fu et al. [15] tighten the inequality constraints at discrete-time instant, hence guarantee the satisfactory of constraints over the whole interval. However, tighter constraints may lead to relatively conservative solution.

To address these issues, we parameterize the flat output of the continuous-time linear system by piecewise polynomials. The differential equation of the dynamic system is eliminated and replaced by flatness map between flat output y and system state x, input u [16]. In this way, the decision variables become finite-dimensional polynomial coefficients. On the other hand, the inequality constraints become non-negative polynomials over intervals, which are still infinite-dimensional. Fortunately, we can leverage Markov-Lukács theorem [17], [18] to transcribe a polynomial inequality constraint on an interval into an equivalent matrix Positive Semi-Definite (PSD) constraint, thus ensures the path constraints hold at every time interval. With this procedure, the continuous-time MPC problem can be transcribed into a Semi-Definite Programming (SDP) problem.

It is worth noticing that the SDP problem we formulate contains a large amount of small symmetric matrices. As a result, we propose to use parallel computing to speed up the calculation. To this end, we use a customized Primal-Dual Hybrid Gradient (PDHG) algorithm to solve the SDP problem. PDHG, also known as ChambollePock [19], is a well-known first-order algorithm dealing with convex optimization problems with equality constraints. For large scale problems, it has been one of the preferred first-order algorithm [20] due to the fact that it can be easily parallelized.

[1]Zishuo Li, Bo Yang, Jiayun Li, and Yilin Mo are with the Department of Automation, Tsinghua University, Beijing, 100084, China. {lizs19,yang-b21,lijiayun22}@mails.tsinghua.edu.cn, ylmo@tsinghua.edu.cn

[2]Jiaqi Yan is with Department of Computer Science, Tokyo Institute of Technology, Tokyo, Japan. jyan@sc.dis.titech.ac.jp

The main contributions of this article are as follows.

- An equivalent formulation of continuous inequality constrained linear MPC problem is derived, in which the system dynamics are eliminated by using differential flatness. The problem is then converted into a polynomial optimization problem by parameterizing the flat output with piece-wise polynomials.
- Path constraints are rigorously guaranteed by using sum of squares theory to transcribe the non-negative constraints of polynomials into the equivalent positive semi-definite constraints of matrices, and an equivalent SDP programming problem is formulated.
- The SDP problem is solved by using the customized primal-dual splitting-based iterations and accelerated by parallel computing.

It is worth noting that, although the derivations in this paper are carried out for linear MPC problems, it can also be extended to nonlinear MPC problems if the constraints remain linear and the objective remains quadratic after the differentially flat transformation.

The paper is organized as follows. Differential flatness theory is stated and the form of flatness map for linear systems is described in Section II. The transformation of linear MPC problem with continuous-time path constraints into SDP problem is discussed in Section III. In Section IV, we present the PDHG algorithm for SDP solving and explain that it can be accelerated by parallel computing. The simulation validation of our proposed MPC solver on quadruple-tank process is provided in Section V. Finally, concluding remarks are made in Section VI

II. Preliminary: Differential Flatness of Linear System

Differential flatness is an important concept for a class of linear and nonlinear systems [16]. A system is differentially flat if and only if there exists a flat output, such that all states and inputs subject to system dynamical constraints can be explicitly expressed as functions of the flat output (which is free of dynamical constraints) and a finite number of its derivatives.

In this paper, we restrict our discussion to linear system. Consider an LTI system governed by the following ordinary differential equation:

$$\dot{x}(t) = Ax(t) + Bu(t), \tag{1}$$

where $x \in \mathbb{R}^n, u \in \mathbb{R}^m$. Without loss of generality, we assume that (A, B) is controllable. Otherwise, we can always perform a Kalman decomposition and only consider the controllable part of the system.

For such a system, Filess et al. [16] proved the following theorem:

Theorem 1 (Linear flatness [16])**.** *A linear system is differentially flat, if and only if, it is controllable.*

In general, the choice of the flat output may not be unique. In this paper, we adopt the procedure proposed by Yong et al. [21] to derive the flat output as well as the flatness map:

Theorem 2. *If (A, B) is controllable, then there exists a matrix $\mathcal{T} \in \mathbb{R}^{m\times n}$, such that the following y is the flat output of the system:*

$$y = \begin{bmatrix} y_1 & \cdots & y_m \end{bmatrix}^\top \triangleq \mathcal{T}x \in \mathbb{R}^m. \tag{2}$$

Moreover, there exists matrices $S \in \mathbb{R}^{n\times(n+m)}$, and $H \in \mathbb{R}^{m\times(n+m)}$, such that the state and the input of the system can be represented by the following flatness map:

$$x = S\boldsymbol{y}, u = H\boldsymbol{y},$$

where $\boldsymbol{y}$ is the extended flat output vector consisting of y_is and their derivatives, i.e.,[1]

$$\boldsymbol{y} \triangleq \begin{bmatrix} y_1 & \cdots & y_1^{(\kappa_1+1)} & \cdots & y_m & \cdots & y_m^{(\kappa_m+1)} \end{bmatrix}^\top \in \mathbb{R}^{n+m}. \tag{3}$$

The procedure to construct the $\mathcal{T}$, S, H matrices and extended flat output $\boldsymbol{y}$ (including the calculation of κ_i) is omitted due to space limit and the readers can refer to [21] for more details.

III. SDP Formulation of MPC

This section is devoted to transcribing the MPC problem of a continuous-time path constrained linear system (1) to an SDP problem, the procedure of which is depicted in Fig 1. In the next subsection, we first remove the differential equality constraints in the MPC problem by differential flatness and then convert the problem into a polynomial optimization problem by parameterizing the flat output with piecewise polynomials. The polynomial optimization problem is then transformed into an equivalent SDP problem via Markov-Lukács Theorem [17], [18] and Sum-of-Squares (SOS) in Section III-B.

A. Polynomial Optimization Formulation of MPC

We consider an optimal control problem of the continuous-time linear system (1) under state and input constraints, which can be formulated in a receding horizon fashion as follows:

Problem 1 (Continuous-time Linear MPC Problem)**.**

$$\begin{aligned} \min_{x(t),u(t)} \quad & \int_0^T x(t)^\top Q x(t) + u(t)^\top R u(t) dt \\ \textit{s.t.} \quad & \dot{x}(t) = Ax(t) + Bu(t), \quad \forall t \in [0, T] \\ & \Xi x(t) + \Upsilon u(t) \le b, \quad \forall t \in [0, T] \\ & x(0) = x_0, \end{aligned}$$

where T is the horizon length and $\Xi \in \mathbb{R}^{p\times n}$, $\Upsilon \in \mathbb{R}^{p\times m}$ are matrices and $b \in \mathbb{R}^p$ is a vector of proper dimensions.

Adopting the flatness map in Section II, we can express the state $x(t)$ and control input $u(t)$ using the extended flat output $\boldsymbol{y}(t)$ and hence removing the differential equation constraint in Problem 1, which results in the following problem:

[1] $\kappa_i, i \in \{1, \cdots, m\}$ is determined by A, B and satisfies $\sum_{i=1}^m \kappa_i = n$.

Problem 2 (MPC using Flat Output)**.**

$$\begin{aligned}\min_{y(t)} \quad & \int_0^T \boldsymbol{y}(t)^\top (S^\top Q S + H^\top R H)\boldsymbol{y}(t)\ \mathrm{d}t \\ \textit{s.t.} \quad & (\Xi S + \Upsilon H)\boldsymbol{y}(t) \le b, \quad \forall t \in [0,T] \\ & S\boldsymbol{y}(0) = x_0.\end{aligned}$$

Notice that Problem 1 and Problem 2 are equivalent, in the sense that we can use the definition of the flat output $y = \mathcal{T}x$ and the flatness map $x = S\boldsymbol{y}$, $u = H\boldsymbol{y}$ to map the solution of one problem to the other.

Further notice that the path constraint $\Xi x(t) + \Upsilon u(t) \le b$ (or $(\Xi S + \Upsilon H)\boldsymbol{y}(t) \le b$), which consists of p linear inequalities, requires that the state and the control input (or the flat output) to be inside a polytope at all time interval $[0, T]$. Aside from very special cases, Problem 1 (or Problem 2) cannot be solved in the infinite dimensional function space, due to the difficulty to determine when the path constraints are active [12].

To facilitate optimization-based method to solve Problem 2, we propose to parameterize the flat output $y(t)$ by piecewise polynomials, which effectively reduce the domain of the optimization problem from infinite dimensional function space to a finite dimensional space. To this end, first define the polynomial basis of degree d as

$$\gamma(t) = \begin{bmatrix} t^d & \cdots & t & 1 \end{bmatrix}^\top. \tag{4}$$

Suppose each entry of flat output y is represented by N segments of polynomials in the horizon $[0, T]$. Denote row vector $c_{l,i} \in \mathbb{R}^{(d+1)\times 1}$ as the coefficient of segment l of flat output y_i, i.e.,[2]

$$y_i(t) = \begin{cases} c_{1,i}^\top \gamma\left(\frac{tN}{T}\right), 0 \le t < \frac{T}{N} \\ c_{2,i}^\top \gamma\left(\frac{tN}{T} - 1\right), \frac{T}{N} \le t < \frac{2T}{N} \\ \vdots \\ c_{N,i}^\top \gamma\left(\frac{tN}{T} - (N-1)\right), \frac{(N-1)T}{N} \le t < T \end{cases}. \tag{5}$$

Each segment of polynomial $c_{l,i}^\top \gamma(\cdot), l \in \{1, \cdots, N\}$ has been normalized such that the time variable is on interval $[0, 1]$.

By stacking the coefficients of the l-th segment $c_{l,i}$ vertically, we have the overall coefficient vector

$$c_l \triangleq \begin{bmatrix} c_{l,1} \\ \vdots \\ c_{l,m} \end{bmatrix} \in \mathbb{R}^{m(d+1)\times 1},\ \boldsymbol{c} \triangleq \begin{bmatrix} c_1 \\ \vdots \\ c_N \end{bmatrix} \in \mathbb{R}^{m(d+1)N\times 1}. \tag{6}$$

As a result, instead of optimizing y in the infinite-dimensional function space, we can restrict ourselves to the following polynomial optimization problem:

[2]Since we need smoothness constraints on the conjecture points of segments, $c_{l,i}, \cdots, c_{l+1,i}$ are not fully free and coupled by equality constraints.

Problem 3 (Polynomial Optimization)**.**

$$\begin{aligned}\min_{\boldsymbol{c}} \ & J(\boldsymbol{c}) = \boldsymbol{c}^\top P \boldsymbol{c} \\ \textit{s.t.} \ & (L_j c_l - g_j)^\top \gamma(t) \ge 0, \forall t \in [0,1], \\ & \qquad j \in \{1, \cdots, p\}, l \in \{1, \cdots, N\} \\ & h_j^\top \boldsymbol{c} = r_j \ , j \in \{1, \cdots, 2mN\}\end{aligned}$$

The calculation of parameters in Problem 3 is as follows. Define $\mathbf{e}_j$ as the canonical basis vector of length $d+1$, where 1 in on the j-th entry and 0 on other entries. Define a matrix to represent the derivative of d degree polynomial:

$$D \triangleq \begin{bmatrix} 0 & & & & \\ d & 0 & & & \\ & d-1 & 0 & & \\ & & \ddots & \ddots & \\ & & & 1 & 0 \end{bmatrix} \in \mathbb{R}^{(d+1)\times(d+1)}. \tag{7}$$

Thus, for any coefficient $c \in \mathbb{R}^{d+1}$, we have polynomial derivative equation $\frac{\mathrm{d}\left(c^\top \gamma(t)\right)}{\mathrm{d}t} = (Dc)^\top \gamma(t)$. Define $\boldsymbol{D}_k = \begin{bmatrix} I & D^\top & (D^2)^\top & \cdots & (D^k)^\top \end{bmatrix}^\top$. Then based on (3), one can verify that after polynomial parameterizing, the relationship between flat output y and extend flat output $\boldsymbol{y}$ are $\boldsymbol{y} = \boldsymbol{\Pi} \cdot y$ with $\boldsymbol{\Pi}$ defined as

$$\boldsymbol{\Pi} \triangleq \begin{bmatrix} \boldsymbol{D}_{\kappa_1+1} & & & \\ & \boldsymbol{D}_{\kappa_2+1} & & \\ & & \ddots & \\ & & & \boldsymbol{D}_{\kappa_m+1} \end{bmatrix}.$$

Denote the j-th row of Ξ, Υ as $[\Xi]_j, [\Upsilon]_j$ respectively. Denote the j-th entry of vector b as b_j. Then the parameters in Problem 3 is defined as:

$$\begin{aligned} L_j &\triangleq (([\Xi]_j S + [\Upsilon]_j H) \otimes I_{d+1}) \times \boldsymbol{\Pi}, \\ g_j &\triangleq b_j\ \mathbf{e}_{d+1}, \end{aligned}$$

where $\otimes$ is the Kronecker product and $\mathbf{0}_d$ is the all-zero vector of length d. I_{d+1} is the identity matrix of size $(d+1)\times(d+1)$.

The equality constraints are composed of segment smooth conditions and initial conditions, that is, for neighboring polynomial segments, the value of the polynomial and the value of its first-order derivative at conjecture points or at initial time are the same. The equality constraint parameters are defined by

$$h_j = \begin{cases} \mathbf{e}_1^N \otimes \mathbf{e}_j \otimes \mathbf{e}_{d+1}, \text{ if } 1 \le j \le m. \\ \mathbf{e}_l^N \otimes \mathbf{e}_{j-m} \otimes \mathbf{e}_{d+1} - \mathbf{e}_{l+1}^N \otimes \mathbf{e}_{j-m} \otimes \mathbf{1}_{d+1}, \\ \qquad \text{if } m+1 \le j \le mN. \\ \mathbf{e}_1^N \otimes \mathbf{e}_j \otimes \mathbf{e}_d, \text{ if } mN+1 \le j \le mN+m. \\ \mathbf{e}_l^N \otimes \mathbf{e}_{j-mN-m} \otimes \mathbf{e}_{d+1} - \mathbf{e}_{l+1}^N \otimes \mathbf{e}_{j-Nm-m} \otimes \mathbf{1}_{d+1}, \\ \qquad \text{if } mN+1 \le j \le mN+m. \end{cases}$$

where $\mathbf{e}_j^N$ is the canonical basis vector of size N, with 1 one j-th entry and 0 on other entries.

$$r_j = \begin{cases} [\mathcal{T}x_0]_j, \text{ if } 1 \le j \le m. \\ 0, \text{ if } m+1 \le j \le mN. \\ [\mathcal{T}Ax_0]_{j-mN}, \text{ if } mN+1 \le j \le mN+m. \\ 0, \text{ if } mN+1 \le j \le mN+m. \end{cases}$$

Define matrix $P_{\text{int}} \in \mathbb{R}^{(d+1)\times(d+1)}$ associating objective function integration on $[0,1]$ as

$$[P_{\text{int}}]_{u,v} = \frac{1}{2d+3-u-v}. \tag{8}$$

The parameter $P \in \mathbb{R}^{mN(d+1)\times mN(d+1)}$ in the objective of Problem 3 is calculated by

$$P = \left((S\mathbf{\Pi})^\top Q S\mathbf{\Pi} + (H\mathbf{\Pi})^\top R H\mathbf{\Pi}\right) \otimes I_N \otimes P_{\text{int}}. \tag{9}$$

Remark 1. *Piecewise polynomials are chosen to represent the flat output for the following reasons:*

- *The set of polynomials are closed under derivative operation and is dense in the function space, as is shown by the Stone-Weierstrass theorem. Hence, we can approximate any continuous functions to arbitrary precision. In fact, one can also use polynomials to approximate the derivatives and high order derivatives of a smooth enough function [22].*
- *The continuous-time path constraints are transformed into non-negativity of a univariate polynomial inside an interval, which can be transformed* **exactly** *into Positive Semi-Definite (PSD) cone constraint using Markov-Lukács theorem and SOS [17]. The detailed discussion is reported in the subsequent subsection.*

B. SDP Formulation via SOS

This subsection is devoted to the *exact* SDP formulation of the polynomial optimization Problem 3. To this end, the following theorem is needed:

Theorem 3 (Markov-Lukács theorem [17])**.** *Let $a < b$. Then, a polynomial $p(t)$ is non-negative for $t \in [a,b]$, if and only if it can be written as*

$$p(t) = \begin{cases} f(t) + (t-a)(b-t)g(t), & \text{if } \deg(p) \text{ is even} \\ (t-a)f(t) + (b-t)g(t), & \text{if } \deg(p) \text{ is odd} \end{cases},$$

where $f(t), g(t)$ are SOS polynomials, with degree $\deg(f) \le \deg(p)$, $\deg(g) \le \deg(p) - 2$ when $\deg(p)$ is even, or $\deg(f) \le \deg(p) - 1$, $\deg(g) \le \deg(p) - 1$ when $\deg(p)$ is odd.

For simplicity, we shall only consider the case where the flat output y is an odd degree polynomial, i.e., d is an odd number. The case where d is even can be treated similarly. Let us denote $\delta \triangleq \frac{d-1}{2}$. Notice that a degree $d-1$ SOS polynomial f can be represented as

$$f(t) = \tilde{\gamma}(t)^\top X \tilde{\gamma}(t),$$

with $\tilde{\gamma}(t) \triangleq \begin{bmatrix} t^\delta & \cdots & t & 1 \end{bmatrix}^\top$ and positive semi-definite matrix $X \in \mathbb{R}^{(\delta+1)\times(\delta+1)}$.

As a result, each inequality constraint $(L_j c_l - g_j)^\top \gamma(t) \ge 0$ in Problem 3 can be equivalently represented as

$$\begin{aligned}&(L_j c_l - g_j)^\top \gamma(t) = \\ &\quad t\tilde{\gamma}(t)^\top X^f_{j,l} \tilde{\gamma}(t) + (1-t)\,\tilde{\gamma}(t)^\top X^g_{j,l} \tilde{\gamma}(t),\end{aligned} \tag{10}$$

with positive semi-define matrices $X^f_j, X^g_j \in \mathbb{R}^{(\delta+1)\times(\delta+1)}$. By comparing the coefficients of the polynomials on the LHS and RHS of (10), we know that (10) is equivalent to:

$$L_j c_l - g_j = \mathcal{M}(X^f_{j,l}, X^g_{j,l}) \tag{11}$$

where

$$\mathcal{M}(X^f_{j,l}, X^g_{j,l}) = \begin{bmatrix} \operatorname{tr}(F_0 X^f_{j,l}) + \operatorname{tr}(G_0 X^g_{j,l}) \\ \operatorname{tr}(F_1 X^f_{j,l}) + \operatorname{tr}(G_1 X^g_{j,l}) \\ \vdots \\ \operatorname{tr}(F_d X^f_{j,l}) + \operatorname{tr}(G_d X^g_{j,l}) \end{bmatrix}, \tag{12}$$

and $\{F_i, G_i | i = 0, 1, \cdots, d\}$ is a sequence of constant matrices defined as

$$[F_i]_{u,v} = \begin{cases} 1, & \text{if } u+v = i+2 \\ 0, & \text{otherwise} \end{cases},$$

$$[G_i]_{u,v} = \begin{cases} -1, & \text{if } u+v = i+2 \\ 1, & \text{if } u+v = i+1 \\ 0, & \text{otherwise} \end{cases},$$

where $[\cdot]_{u,v}$ represents the entry at row u, column v in a matrix. Notice that linear function $\mathcal{M}(\cdot,\cdot)$ is independent of index j, l.

Now we handle the second order objective function $J(c) = \boldsymbol{c}^\top P \boldsymbol{c}$ by linear matrix inequality techniques. Notice that P is a positive semi-definite matrix, define $\tilde{P} \in \mathbb{R}^{\operatorname{rank}(P)\times \operatorname{size}(P)}$, such that $\tilde{P}^\top \tilde{P} = P$. Notice that the following three optimization problems are equivalent where s is a scalar:

$$\begin{aligned}&\min_{\boldsymbol{c}\in\mathcal{C}} \quad \boldsymbol{c}^\top P \boldsymbol{c} \Leftrightarrow \min_{\boldsymbol{c}\in\mathcal{C}, s} \; s, \text{ s.t. } \|\tilde{P}\boldsymbol{c}\|_2 \le s \\ &\Leftrightarrow \min_{\boldsymbol{c}\in\mathcal{C}, s\ge 0} s, \text{ s.t. } \begin{bmatrix} \tilde{P}\boldsymbol{c} \\ s \end{bmatrix} \in \text{second order cone}.\end{aligned}$$

We arrive at the following SDP problem which is equivalent to Problem 3 and computationally tractable.

Problem 4 (SDP Problem)**.**
Original form

$$\begin{aligned}&\min_{\boldsymbol{c}, s, \{X^f_{j,l}, X^g_{j,l}\}} \quad s \\ &\text{s.t. } L_j c_l - g_j = \mathcal{M}(X^f_{j,l}, X^g_{j,l}), X^f_{j,l},\, X^g_{j,l} \in \mathbb{S}_+, \\ &\qquad\qquad j \in \{1, \cdots, p\}, l \in \{1, \cdots, N\} \qquad (13) \\ &\qquad h_j^\top \boldsymbol{c} = r_j, j \in \{1, \cdots, 2mN\} \qquad (14) \\ &\qquad \begin{bmatrix} \tilde{P}\boldsymbol{c} \\ s \end{bmatrix} \in \text{SOC}\end{aligned}$$

where SOC *denotes the second order cone and $\mathbb{S}_+$ is the positive semi-definite cone.*

For notation conciseness, we define:

$$\boldsymbol{X} \triangleq \begin{bmatrix} X^f_{1,1} & & & & \\ & X^g_{1,1} & & & \\ & & \ddots & & \\ & & & X^f_{p,N} & \\ & & & & X^g_{p,N} \end{bmatrix}. \tag{15}$$

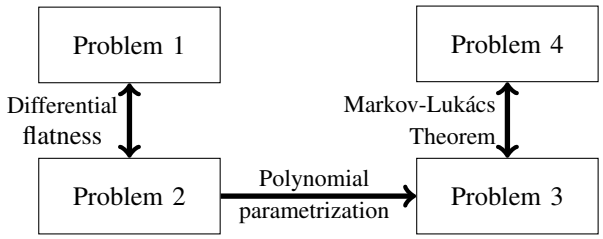


Fig. 1: The relationships between optimization problems in this paper. Double tail arrow represents that the two problems are equivalent. Single tail arrow means that Problem 3 is obtained by parameterizing Problem 2 using polynomials.

Moreover, define function

$$\boldsymbol{M}(\boldsymbol{X}) = \begin{bmatrix} \mathcal{M}(X^f_{1,1}, X^g_{1,1}) \\ \vdots \\ \mathcal{M}(X^f_{p,N}, X^g_{p,N}) \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} \mathrm{tr}(M^0_{1,1}\boldsymbol{X}) \\ \vdots \\ \mathrm{tr}(M^d_{1,1}\boldsymbol{X}) \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \mathrm{tr}(M^0_{p,N}\boldsymbol{X}) \\ \vdots \\ \mathrm{tr}(M^d_{p,N}\boldsymbol{X}) \end{bmatrix} \end{bmatrix}, \tag{16}$$

where $M^i_{j,l}$ is the corresponding matrix composed of F_i, G_i at compatible position that generates $\mathcal{M}(X^f_{j,l}, X^g_{j,l})$ in (12). Define

$$\boldsymbol{L} = I_N \otimes \begin{bmatrix} L_1^\top & L_2^\top & \cdots & L_p^\top \end{bmatrix}^\top, \tag{17}$$

$$\boldsymbol{g} = I_N \otimes \begin{bmatrix} g_1^\top & g_2^\top & \cdots & g_p^\top \end{bmatrix}^\top, \tag{18}$$

$$\boldsymbol{h} = \begin{bmatrix} h_1^\top & h_2^\top & \cdots & h_{2mN}^\top \end{bmatrix}^\top, \tag{19}$$

$$\boldsymbol{r} = \begin{bmatrix} r_1^\top & r_2^\top & \cdots & r_{2mN}^\top \end{bmatrix}^\top. \tag{20}$$

We can rewrite Problem 4 as

Compact form

$$\begin{aligned} \min_{s,\boldsymbol{c},\boldsymbol{X}} \quad & s \\ \text{s.t. } & \boldsymbol{L}\boldsymbol{c} - \boldsymbol{M}(\boldsymbol{X}) = \boldsymbol{g} && (21) \\ & \boldsymbol{h}\boldsymbol{c} = \boldsymbol{r} && (22) \\ & \boldsymbol{X} \in \mathbb{S}_+, s \geq 0 \\ & \begin{bmatrix} \tilde{P}\boldsymbol{c} \\ s \end{bmatrix} \in \mathrm{SOC} \end{aligned}$$

Since there are p inequality constraints in the original Problem 1, $\boldsymbol{X}$ is a block diagonal matrix with $2pN$ positive semi-definite matrices of size $\delta + 1$. As a result, in the following section, we introduce a customized algorithm that solves Problem 4 by primal-dual hybrid gradient methods which can handle $\boldsymbol{X}$ in a parallel fashion. However, before continuing on, we would like to give a comparison between the conventional quadratic programming-based linear MPC and our approach.

C. Discussions

A conventional way to solve the continuous-time optimal control problem is to discretize it into the following discrete-time linear MPC problem [12]:

Problem 5 (Discrete-time Linear MPC Problem)**.**

$$\begin{aligned} \min_{\{x[k],u[k]\}_{k=1}^{T_d}} \quad & \sum_{k=1}^{T_d} x[k]^\top Q x[k] + u[k]^\top R u[k] \\ \textit{s.t.} \quad & x_{k+1} = A_d x_k + B_d u_k, \quad k = 1, \cdots, T_d \\ & \Xi x[k] + \Upsilon u[k] \leq b, \quad k = 1, \cdots, T_d \end{aligned}$$

where A_d, B_d are the discretized system matrix assuming zero-order hold for the control input is used and $T_d \in \mathbb{Z}_+$ is the discrete horizon length.

One of the main differences between Problem 5 and Problem 3 is that the control input is parameterized as step functions in Problem 5 (assuming zero-order hold is used), while for our case, the flat output (and hence the control input as it is a linear function of the flat input and its derivatives) is parameterized as polynomials.

Another difference is that Problem 5 is a Quadratic Programming (QP) problem and hence can be solved more efficiently than SDP. However, this is due to the fact that in Problem 5, constraints are only required to hold at discrete sampling time instants and therefore they may be violated in between sampling times.

On the other hand, the reason for our SDP formulation is that we want to have an exact representation of the continuous time path constraints. If we only require the constraint to hold at discrete time instant, since the value of a polynomial at a time instant is a linear function of its coefficients, we can express such constraints as linear inequalities on the coefficients of the polynomial, which effectively relaxed the polynomial optimization Problem 3 into a QP problem that only guarantees constraint satisfaction at a discrete time instant. As an alternative, one could also leverage the following theorem to generate a QP problem, which has a smaller feasible set than that of Problem 3, but is guaranteed to satisfy the path constraints at every time instant.

Theorem 4 ([23])**.** *Let* $\mathrm{Pd}([a,b])$ *denote the set of polynomials* $p(t) > 0, \forall t \in [a,b]$*. Define*

$$\mathcal{P}_q := \left\{ \sum_{i+j \leq q} c_{ij}(b-x)^i (x-a)^j \,\middle|\, c_{ij} \geq 0 \right\}.$$

If polynomial $p \in \mathrm{Pd}([a,b])$*, then* $p \in \mathcal{P}_q$ *for sufficiently large integer* q*.*

IV. Accelerated SDP Solving with Parallel Computing

A. Primal dual hybrid gradient for SDP solving

In this subsection, we present the primal-dual hybrid gradient algorithm that solves Problem 4. Encode the constraints

into the objective function as

$$\min_{s,\boldsymbol{c},\boldsymbol{X}} s + \mathbb{I}_{\mathbb{S}_+,\mathrm{SOC}}(\boldsymbol{X},\tilde{\boldsymbol{c}},s) + \mathbb{I}_{=}(\boldsymbol{X},\boldsymbol{c},\tilde{\boldsymbol{c}}), \tag{23}$$

where $\tilde{\boldsymbol{c}} = \tilde{P}\boldsymbol{c}$ is the slack variable, and the indicator functions are defined as

$$\mathbb{I}_{\mathbb{S}_+,\mathrm{SOC}}(\boldsymbol{X},\tilde{\boldsymbol{c}},s) = \begin{cases} 0, \text{ if } \boldsymbol{X} \in \mathbb{S}_+ \text{ and } \begin{bmatrix}\tilde{\boldsymbol{c}} \\ s\end{bmatrix} \in \mathrm{SOC} \\ +\infty, \text{otherwise} \end{cases}. \tag{24}$$

$$\mathbb{I}_{=}(\boldsymbol{X},\boldsymbol{c},\tilde{\boldsymbol{c}}) = \begin{cases} 0, \text{ if } \begin{cases} \boldsymbol{L}\boldsymbol{c} - \boldsymbol{M}(\boldsymbol{X}) = \boldsymbol{g} \\ \boldsymbol{h}\boldsymbol{c} = \boldsymbol{r} \\ \tilde{P}\boldsymbol{c} - \tilde{\boldsymbol{c}} = 0 \end{cases} \\ +\infty, \text{otherwise} \end{cases}. \tag{25}$$

Using primal-dual operator splitting [24], the iterations can be derived as the following where α is the primal step-size and β is the dual step-size. $\mathcal{D}^*_{\boldsymbol{X}}(\cdot)$ is the conjugate operator of the linear mapping $\boldsymbol{M}(\boldsymbol{X})$, and $\mathcal{D}^*_{\boldsymbol{c}}(\cdot)$ is the conjugate operator of the linear mapping $\begin{bmatrix}\boldsymbol{L}^\top & \boldsymbol{h}^\top & \tilde{P}^\top\end{bmatrix}^\top \boldsymbol{c}$, the definition of which are given in (35) and (36) respectively.

1. Primal step:

$$\boldsymbol{X}^{k+1} \leftarrow \mathrm{proj}_{\mathbb{S}_+}(\boldsymbol{X}^k - \alpha\mathcal{D}^*_{\boldsymbol{X}}(\lambda_1^k)) \tag{26}$$

$$\boldsymbol{c}^{k+1} \leftarrow \boldsymbol{c}^k - \alpha\mathcal{D}^*_{\boldsymbol{c}}(\lambda_1^k,\lambda_2^k,\lambda_3^k) \tag{27}$$

$$\begin{bmatrix}\tilde{\boldsymbol{c}}^{k+1} \\ s^{k+1}\end{bmatrix} \leftarrow \mathrm{proj}_{\mathrm{SOC}} \begin{bmatrix}\tilde{\boldsymbol{c}}^k - \alpha(-\lambda_3^k) \\ (s^k - \alpha)^+\end{bmatrix} \tag{28}$$

where $(s^k-\alpha)^+ = \max(s^k-\alpha, 0)$.

2. Calculating difference:

$$\Delta\boldsymbol{X}^{k+1} \leftarrow 2\boldsymbol{X}^{k+1} - \boldsymbol{X}^k \tag{29}$$

$$\Delta\boldsymbol{c}^{k+1} \leftarrow 2\boldsymbol{c}^{k+1} - \boldsymbol{c}^k \tag{30}$$

$$\Delta\tilde{\boldsymbol{c}}^{k+1} \leftarrow 2\tilde{\boldsymbol{c}}^{k+1} - \tilde{\boldsymbol{c}}^k \tag{31}$$

3. Dual step:

$$\lambda_1^{k+1} \leftarrow \lambda_1^k + \beta\boldsymbol{L}\Delta\boldsymbol{c}^{k+1} - \beta\boldsymbol{M}(\Delta\boldsymbol{X}^{k+1}) - \beta\boldsymbol{g} \tag{32}$$

$$\lambda_2^{k+1} \leftarrow \lambda_2^k + \beta\boldsymbol{h}\Delta\boldsymbol{c}^{k+1} - \beta\boldsymbol{r} \tag{33}$$

$$\lambda_3^{k+1} \leftarrow \lambda_3^k + \beta\tilde{P}\Delta\boldsymbol{c}^{k+1} - \beta\Delta\tilde{\boldsymbol{c}}^{k+1} \tag{34}$$

Here λ_1 is the dual variable corresponding to the equality constraint (21) and equivalently (13). λ_2 is the dual variable corresponding to the equality constraint (22) and equivalently (14). λ_3 is the dual variable corresponding to the equality constraint $\tilde{P}\boldsymbol{c} - \tilde{\boldsymbol{c}} = 0$. Define $\lambda = \begin{bmatrix}\lambda_1^\top & \lambda_2^\top & \lambda_3^\top\end{bmatrix}^\top$.

We denote the entry of λ_1 corresponding to segment l, inequality index j, order i as $\lambda_1[i,j,l], i \in \{0,\cdots,d\}, j \in \{1,\cdots,p\}, l \in \{1,2,\cdots,N\}$. Recall the definition of $M^i_{j,l}$ in (16). The conjugate operators $\mathcal{D}^*$ are defined as[3]

[3]We use $[A]_j$ to denote the j-th row of matrix A, or in case of A is column vector, j-th entry of the vector A.

$$\mathcal{D}^*_{\boldsymbol{X}}(\lambda_1) \triangleq -\sum_{l=1}^{N}\sum_{j=1}^{p}\sum_{i=1}^{d+1}\lambda_1[i,j,l]M^i_{j,l} \tag{35}$$

$$\mathcal{D}^*_{\boldsymbol{c}}(\lambda_1,\lambda_2,\lambda_3) \triangleq \sum_{l=1}^{N}\sum_{j=1}^{p}\sum_{i=1}^{d+1}\lambda_1[i,j,l]\left(\mathbf{e}_l^N \otimes [L_j]_i^\top\right) + \sum_{j=1}^{2mN}[\lambda_2]_j h_j + \tilde{P}^\top\lambda_3 \tag{36}$$

where $\mathbf{e}_l^N$ is the canonical basis vector of size N, with 1 on l-th entry and 0 on other entries.

The projection to the semi-definite cone is

$$\mathrm{proj}_{\mathbb{S}_+}(X) = \sum_{i=1}^{\mathrm{size}(X)} \max\{0,\nu_i\}\,\mu_i\mu_i^\top \tag{37}$$

where ν_i, μ_i are the eigenvalue and the corresponding eigenvector of X. The projection to the second order cone is

$$\mathrm{proj}_{\mathrm{SOC}}\begin{bmatrix}\boldsymbol{c} \\ s\end{bmatrix} = \begin{cases} \frac{s+\|\boldsymbol{c}\|_2}{2\|\boldsymbol{c}\|_2}\begin{bmatrix}\boldsymbol{c} \\ \|\boldsymbol{c}\|_2\end{bmatrix} & \text{if } \|\boldsymbol{c}\|_2 > s. \\ \begin{bmatrix}\boldsymbol{c} \\ s\end{bmatrix} & \text{if } \|\boldsymbol{c}\|_2 \le s. \end{cases}. \tag{38}$$

The convergence of algorithm (26)-(34) is provided in the following.

Theorem 5 ([25]). *Assume the solution to KKT conditions of Problem 4 exists (denoted by $\boldsymbol{c}^\star, \boldsymbol{X}^*, s^*, \lambda^*$), and strong duality holds. If the linear projection defined by*

$$\mathcal{L}(\boldsymbol{c},\boldsymbol{X}) = \begin{bmatrix}\boldsymbol{L}\boldsymbol{c} - \boldsymbol{M}(\boldsymbol{X}) \\ \boldsymbol{h}\boldsymbol{c}\end{bmatrix}$$

and step sizes α, β satisfy $0 < \alpha\beta < 1/\left\|\mathcal{L}(\boldsymbol{c},\boldsymbol{X})\right\|_2$, then the primal dual hybrid gradient descent algorithm (26)-(34) *converges to the solution to KKT conditions, i.e., $\boldsymbol{c}^k \to \boldsymbol{c}^\star, \boldsymbol{X}^k \to \boldsymbol{X}^\star, s^k \to s^\star, \lambda^k \to \lambda^\star$.*

B. GPU parallel computing

It is worth noticing that for our proposed iterations, a significant proportion of the time will be spent on the projection $\mathrm{proj}_{\mathbb{S}_+}(\cdot)$. However, since $\boldsymbol{X}$ is a block diagonal matrix with $2pN$ matrices of size $\delta+1$ on its diagonal, the projection of $\boldsymbol{X}$ can be parallelized by projecting each small matrices onto the PSD cone. Furthermore, the calculation of $\mathcal{D}^*_{\boldsymbol{X}}, \mathcal{D}^*_{\boldsymbol{c}}$ in (35) and (36), and the difference calculation in (29) are essentially tensor operations and hence can be accelerated by parallel computation.

To speed up the computation of the proposed PDHG solver, we implement it in a parallelized manner on the GPU. Specifically, the projection step (26) is wrapped as a kernel to be computed in parallel on the GPU. Additionally, we implement the calculation of $\mathcal{D}^*_{\boldsymbol{X}}$ and $\mathcal{D}^*_{\boldsymbol{c}}$ in (35) and (36), as well as the update steps from (27) to (34) as tensor operations, which can also be accelerated by GPU parallelization.

We test the implemented our proposed solver on a desktop computer equipped with an AMD Ryzen Threadripper 3970X 32-Core Processor and an NVIDIA GeForce

RTX 3080 GPU. We report the computation time for a single iteration of the parallelized solver running on GPU in Table I, and compare it to that of a serialized version running on the CPU. Our results show that the iteration time of the accelerated solver is significantly shorter than that of the CPU version. Moreover, the computation time of the accelerated solver increases slowly as the problem size (the value d and N) grows. Even for the largest problem instances considered, the iteration time remains within a few milliseconds, demonstrating the effectiveness of the GPU acceleration and the efficiency of the implementation.

N	d		
	3	5	7
2	0.318 (1.874)	0.345 (2.773)	0.390 (3.778)
6	0.408 (5.159)	0.443 (8.012)	0.435 (11.240)
10	0.431 (8.811)	0.506 (13.758)	0.547 (19.254)
20	0.484 (19.449)	0.649 (30.721)	0.821 (42.886)
30	0.561 (32.330)	0.767 (50.046)	1.187 (63.170)
40	0.776 (46.167)	1.394 (65.989)	2.593 (93.495)

TABLE I: Time consumption (in milliseconds) of a single iteration of the PDHG solver (calculating updates (26)-(34)) for different problem sizes. The runtime on CPU is in parentheses.

C. Warm Start and Termination Rule

Denote Δt as the control apply time length of each solution. To facilitate the simple warm start strategy, the horizon T satisfies $T = N \cdot \Delta t$, i.e., the first segment of control input $u(t)$ is applied before receding to a new horizon. We evaluate our algorithm's performance in two different strategies: cold start, and warm start. The cold start strategy initializes the optimization variable $\boldsymbol{c}(t+\Delta t), \boldsymbol{X}(t+\Delta t), \lambda(t+\Delta t)$ as random vectors/matrices with each entry uniformly distributed on $[-0.5, 0.5]$. The warm start strategy initializes the optimization variable in a shifting manner, i.e., $\forall\ 2 \le l \le N, 1 \le j \le p$:

$$\left(X^f_{j,l}(t+\Delta t)\right)^0 := \left(X^f_{j,l-1}(t)\right)^{\tau(t)}$$
$$\left(X^g_{j,l}(t+\Delta t)\right)^0 := \left(X^g_{j,l-1}(t)\right)^{\tau(t)}$$
$$\left(c_l(t+\Delta t)\right)^0 := \left(c_{l-1}(t)\right)^{\tau(t)}$$

where the super script $\tau(t)$ is the number of iterations applied to solve the SDP problem at time t. For the first segment, $\left(X^f_{j,1}(t+\Delta t)\right)^0, \left(X^g_{j,1}(t+\Delta t)\right)^0, \left(c_1(t+\Delta t)\right)^0$ are initialized randomly. The dual variable λ is also initialized in a shifting manner according to its correspondence with $\boldsymbol{X}, \boldsymbol{c}$ in (32)-(34).

The algorithm termination is determined by the residue $\epsilon^k = \epsilon^k_{\text{primal}} + \epsilon^k_{\text{dual}}$ where the primal and dual residue are defined as:

$$\epsilon^k_{\text{primal}} = \left\| \frac{1}{\alpha}\left(\boldsymbol{X}^k - \boldsymbol{X}^{k-1}\right) - \mathcal{D}^*_{\boldsymbol{X}}\left(\lambda^k - \lambda^{k-1}\right) \right\|_F + \left\| \frac{1}{\alpha}\left(\boldsymbol{c}^k - \boldsymbol{c}^{k-1}\right) - \mathcal{D}^*_{\boldsymbol{c}}\left(\lambda^k - \lambda^{k-1}\right) \right\|_2$$

$$\epsilon^k_{\text{dual}} = \left\| \frac{1}{\beta}\left(\lambda^k - \lambda^{k-1}\right) - \begin{bmatrix} \boldsymbol{L}(\boldsymbol{c}^k - \boldsymbol{c}^{k-1}) - \boldsymbol{M}(\boldsymbol{X}^k - \boldsymbol{X}^{k-1}) - \boldsymbol{g} \\ \boldsymbol{h}(\boldsymbol{c}^k - \boldsymbol{c}^{k-1}) - \boldsymbol{r} \\ \tilde{P}(\boldsymbol{c}^k - \boldsymbol{c}^{k-1}) - (\tilde{\boldsymbol{c}}^k - \tilde{\boldsymbol{c}}^{k-1}) \end{bmatrix} \right\|_2.$$

$\|\cdot\|_F$ is the Frobenius norm. Our proposed algorithm terminates when ϵ^k corresponding to segment 1 is below 2×10^{-2}, which is accurate enough for control performance. The control and computational speed performance is demonstrated in the next section.

V. Simulation

The performance of the proposed MPC solver is validated on the quadruple-tank process [26], whose schematic diagram is visualized in Fig. 2. The system has 4 states, which represent the liquid levels (in centimeter) of each tank. There are two control inputs in the system, namely the voltage (in volt) of Pump 1 and Pump 2 in Fig. 2. The simulation employs the same linearized system equations and system parameters as [26], which are omitted due to space limitations.

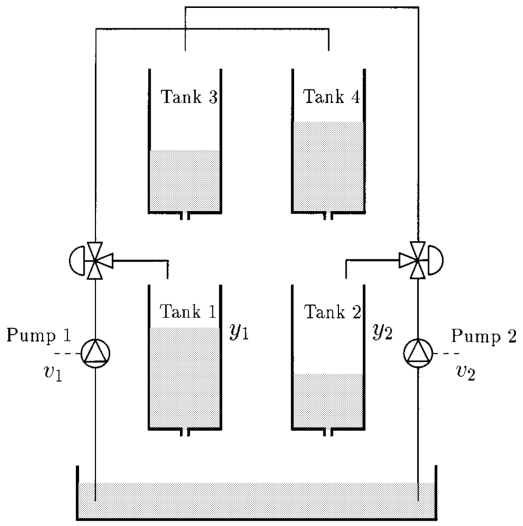


Fig. 2: The schematic diagram of the quadruple-tank process.

The initial conditions of the tanks are

$$x_0 = \begin{bmatrix} 10 & 19 & 19 & 1 \end{bmatrix}^\top,$$

and the control objective of the MPC is to track a reference trajectory $r(t)$ of the liquid levels. For simplicity, we set the constant reference trajectory as

$$r(t) = \begin{bmatrix} 19.9 & 19.9 & 2.4 & 2.4 \end{bmatrix}^\top.$$

In addition to tracking the reference signal, the MPC must ensure that the liquid levels in all tanks remain between 0 to 20 cm and that the control inputs stay in the voltage limit between 0 to 8 V during the control process. The objective weighting matrices in MPC Problem 1 are defined as $Q = I, R = 0.1I$ with appropriate dimensions. Our code is available on https://github.com/zs-li/MPC_PDHG.

A. Control Performance

For comparison, we employ the Quadratic Programming (QP) formulation (Problem 5), where the MPC problem is discretized with a sampling interval of $T_s = 1$s and horizon length $T_d = 20$. On the other hand, for the proposed method, we set the degree of polynomial $d = 3$ and the segments of polynomials $N = 20$, horizon length $T = 20$. For each iteration, the resulting control input applies to the system for $\Delta t = 1$ second. Thus, the two methods are comparable in terms of horizon length and update frequency. We simulate the control process for 120 seconds and visualize the resulting system states and control inputs in Fig. 3-Fig. 5.

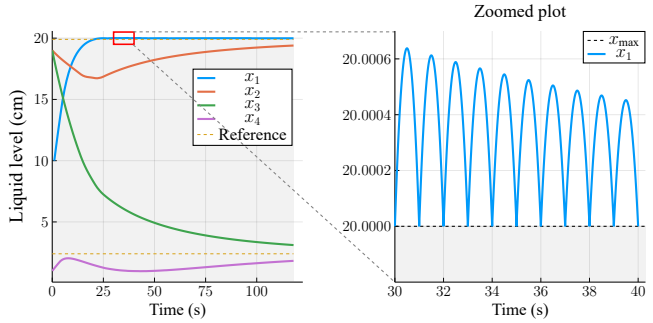


Fig. 3: The states of discrete time linear MPC using the QP solver. The gray area in the figures denotes the feasible region of states. The states between sampling times violate the constraints.

As shown in Fig. 5, at the first glance, the state trajectories obtained from both solvers are nearly identical. However, upon close inspection, it can be seen that even though the QP-based controller satisfies the constraints at discrete-time instants, the constraints are violated in between sampling instants. In contrast, the proposed algorithm ensures constraint satisfaction on the whole time interval. Moreover, our proposed solver satisfies trajectory smooth conditions depicted by constraint $\boldsymbol{hc} = \boldsymbol{r}$.

B. Computational Speed Performance

In the following, we compare the computational speed performance of our proposed algorithm and several off-the-shelf solvers (on Problem 4) under different numbers

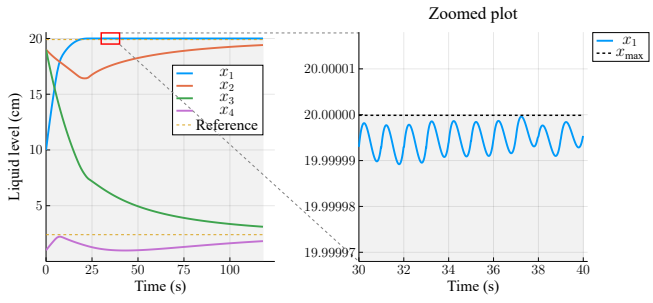
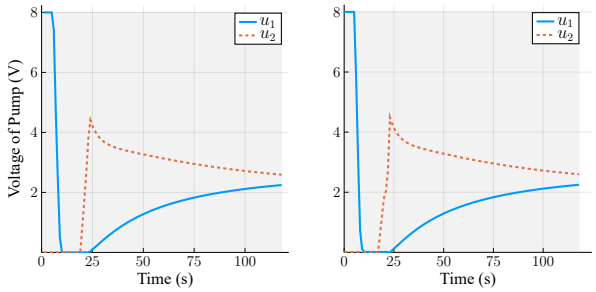


Fig. 4: The states of continuous time linear MPC using our proposed solver. The gray area in the figures denotes the feasible region of states. The states using our proposed MPC input stays in the feasible region for whole time interval.



(a) Control input using the QP solver. (b) Control input using the proposed solver.

Fig. 5: Comparison on the input trajectory using the QP and the proposed SDP strategy respectively. The gray area in the figures denotes the feasible regions of control input.

of polynomial degrees d and polynomial segments N. The apply time Δt is set as 1 second. The block number for GPU acceleration is set as 128. The number of threads on every block is $\lceil \frac{pN}{128} \rceil$. The computational time in Figure 6 is the average solving time of the first 100 apply steps. The step sizes are $\alpha = 0.2, \beta = 0.4$. The computation platform is the same as in Subsection IV-B, i.e., a desktop computer equipped with an AMD Ryzen Threadripper 3970X 32-Core Processor and an NVIDIA GeForce RTX 3080 GPU. The real number calculations on GPU are floating point number with hybrid precision 32-bit and 16-bit, which is computationally efficient and accurate enough for control applications. As for comparison, the other solvers are of default precision 64-bit. Thus, the time comparison may not be equal but represents our computation speed superiority to some extent.

As shown in Fig. 6, our proposed algorithm has better scalability for large problems (especially lagre N), and has low computational time promising for real-time control applications. The warm-start technique introduced in Subsection IV-B can effectively reduce the computation time by reducing iterations. For off-the-shelf solvers, COSMO and COPT perform well on large-scale problems compared to other solvers. However, their computation is still slow and incompatible with real-time control scenarios.

We demonstrate the number of iterations required to reach $(\epsilon^k)_1$ for different problem sizes in Fig 7. The iteration number required grows gently as the problem size grows, which also corroborates the scalability of our proposed solver. The warm-start technique introduced in Subsection IV-B can effectively reduce the iteration number.

Define Lagrange function of Problem 4 as $\mathscr{L}(\boldsymbol{c}, s, \boldsymbol{X}; \lambda)$, then the relative duality gap is defined as

$$\frac{1}{J(\boldsymbol{c})}\left[\inf_{\boldsymbol{c},s,\boldsymbol{X}} \mathscr{L}(\boldsymbol{c}, s, \boldsymbol{X}; \lambda) - \sup_{\lambda} \mathscr{L}(\boldsymbol{c}, s, \boldsymbol{X}; \lambda)\right],$$

where $J(c)$ is the objective value of Problem 3 and equivalently Problem 4. We demonstrate the convergence of relative duality gap with respect to iteration number in Fig. 8. The relative duality gap converged below 10^{-5} within approximately 500 iterations. The problem sizes scarcely influence the convergence speed of the relative duality gap, which also indicates good scalability of our proposed algorithm.

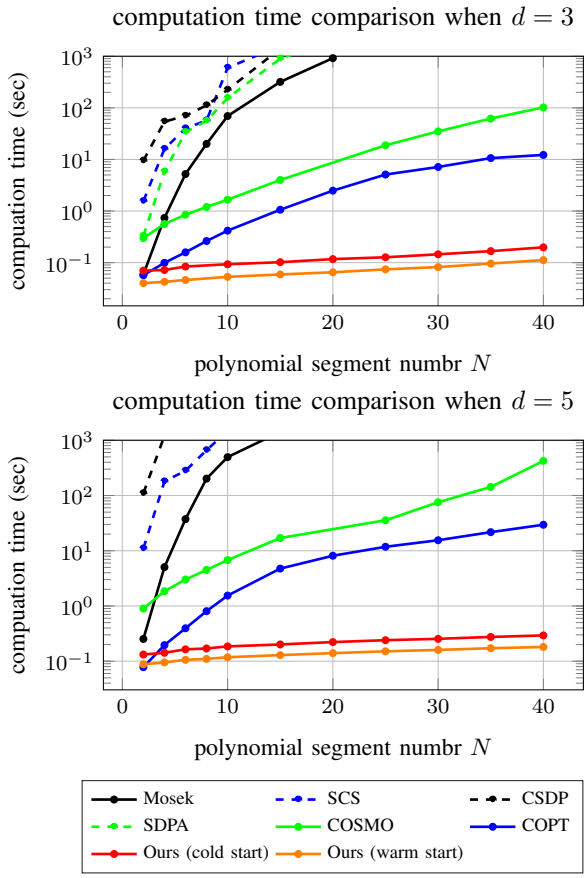


Fig. 6: The states of discrete time linear MPC using the QP solver.

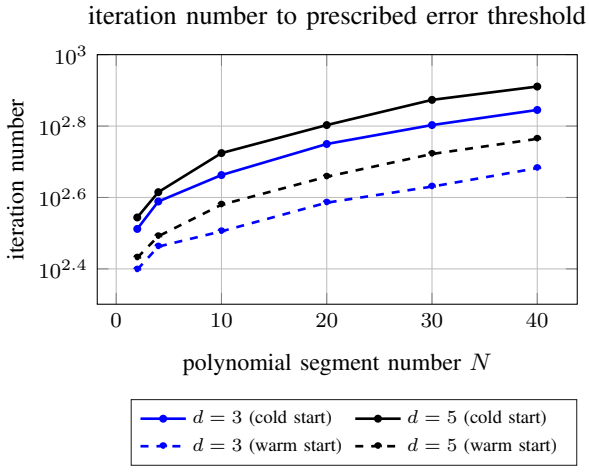


Fig. 7: The number of iterations required to reach residue $(\epsilon^k)_1 < 1 \times 10^{-2}$, where $(\epsilon^k)_1$ means residue of segment 1. Warm start can effectively reduce the number of iterations.

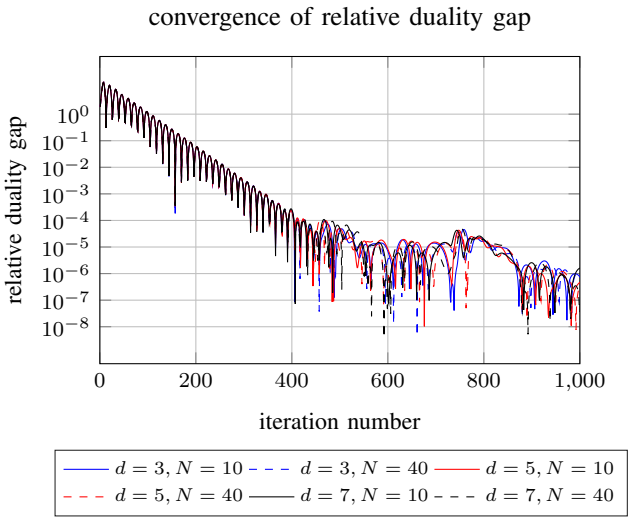


Fig. 8: The relative duality gap of our proposed algorithm (cold start). Problem sizes scarcely influence the convergence speed of the relative duality gap.

VI. Conclusion

In this paper, we aim to address continuous-time path-constrained linear MPC problems while ensuring that path constraints are satisfied at every time interval. To achieve this, we propose an algorithm that utilizes differential flatness to eliminate dynamic constraints. Furthermore, by parameterizing the flat output with piecewise polynomials, we formulate a polynomial optimization problem where the decision variables are finite-dimensional polynomial coefficients, and the inequality path constraints are polynomial non-negativity constraints on intervals, which remain infinite-dimensional. Taking advantage of the Markov-Lukács theorem from SOS theory, we transform the polynomial optimization problem into an equivalent SDP problem that is computationally tractable. To accelerate the solving process of the SDP problem, we use a customized PDHG algorithm, which exploits the block-diagonal structure of the PSD matrix to perform paralleled computation. The numerical simulation of a quadruple-tank process validates that our proposed algorithm can ensure that the path constraints are satisfied at every time interval. Moreover, the parallel accelerated design of our algorithm results in superior computational speed performance.

References

[1] J. Z. Ben-Asher, *Optimal control theory with aerospace applications*. American institute of aeronautics and astronautics, 2010.

[2] R. Sharp and H. Peng, "Vehicle dynamics applications of optimal control theory," *Vehicle System Dynamics*, vol. 49, no. 7, pp. 1073–1111, 2011.

[3] A. C. Satici, H. Poonawala, and M. W. Spong, "Robust optimal control of quadrotor uavs," *IEEE Access*, vol. 1, pp. 79–93, 2013.

[4] T. A. Weber, *Optimal control theory with applications in economics*. MIT press, 2011.

[5] S. M. Aseev, K. O. Besov, and A. V. Kryazhimskii, "Infinite-horizon optimal control problems in economics," *Russian Mathematical Surveys*, vol. 67, no. 2, p. 195, 2012.

[6] C. M. Kellett, S. R. Weller, T. Faulwasser, L. Grüne, and W. Semmler, "Feedback, dynamics, and optimal control in climate economics," *Annual Reviews in Control*, vol. 47, pp. 7–20, 2019.

[7] D. Bertsekas, *Dynamic programming and optimal control: Volume I*. Athena scientific, 2012, vol. 1.

[8] F. Clarke, *Functional analysis, calculus of variations and optimal control*. Springer, 2013, vol. 264.

[9] R. F. Hartl, S. P. Sethi, and R. G. Vickson, "A survey of the maximum principles for optimal control problems with state constraints," *SIAM review*, vol. 37, no. 2, pp. 181–218, 1995.

[10] P. Lundström, J. Lee, M. Morari, and S. Skogestad, "Limitations of dynamic matrix control," *Computers & Chemical Engineering*, vol. 19, no. 4, pp. 409–421, 1995.

[11] R. Rouhani and R. K. Mehra, "Model algorithmic control (MAC); basic theoretical properties," *Automatica*, vol. 18, no. 4, pp. 401–414, 1982.

[12] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: Theory and practicea survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.

[13] H. Djelassi, A. Mitsos, and O. Stein, "Recent advances in nonconvex semi-infinite programming: Applications and algorithms," *EURO Journal on Computational Optimization*, vol. 9, p. 100006, 2021.

[14] T. W. Chen and V. S. Vassiliadis, "Inequality path constraints in optimal control: a finite iteration ε-convergent scheme based on pointwise discretization," *Journal of Process Control*, vol. 15, no. 3, pp. 353–362, 2005.

[15] J. Fu, J. M. Faust, B. Chachuat, and A. Mitsos, "Local optimization of dynamic programs with guaranteed satisfaction of path constraints," *Automatica*, vol. 62, pp. 184–192, 2015.

[16] M. Fliess, J. Levine, P. Martin, and P. Rouchon, "A lie-backlund approach to equivalence and flatness of nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 44, no. 5, pp. 922–937, 1999.

[17] T. Roh and L. Vandenberghe, "Discrete transforms, semidefinite programming, and sum-of-squares representations of nonnegative polynomials," *SIAM Journal on Optimization*, vol. 16, no. 4, pp. 939–964, 2006.

[18] G. Szeg, "Orthogonal polynomials," in *American matemetical society colloquium publications*, vol. 23. American matemetical society, 2003.

[19] A. Chambolle and T. Pock, "A first-order primal-dual algorithm for convex problems withăapplications to imaging," *Journal of Mathematical Imaging and Vision*, vol. 40, no. 1, pp. 120–145, May 2011.

[20] D. Applegate, M. Diaz, O. Hinder, H. Lu, M. Lubin, B. Osingle Donoghue, and W. Schudy, "Practical large-scale linear programming using primal-dual hybrid gradient," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 20 243–20 257.

[21] S. Z. Yong, B. Paden, and E. Frazzoli, "Computational methods for mimo flat linear systems: Flat output characterization, test and tracking control," in *2015 American Control Conference (ACC)*, 2015, pp. 3898–3904.

[22] M. M. Peet, "Exponentially stable nonlinear systems have polynomial lyapunov functions on bounded regions," *IEEE Transactions on Automatic Control*, vol. 54, no. 5, pp. 979–987, 2009.

[23] V. Powers and B. Reznick, "Polynomials that are positive on an interval," *Transactions of the American Mathematical Society*, vol. 352, no. 10, pp. 4677–4692, 2000.

[24] M. Souto, J. D. Garcia, and Álvaro Veiga, "Exploiting low-rank structure in semidefinite programming by approximate operator splitting," *Optimization*, vol. 71, no. 1, pp. 117–144, 2022.

[25] E. K. Ryu and W. Yin, *Large-Scale Convex Optimization: Algorithms & Analyses via Monotone Operators*. Cambridge University Press, 2022.

[26] K. H. Johansson, "The quadruple-tank process: A multivariable laboratory process with an adjustable zero," *IEEE Transactions on control systems technology*, vol. 8, no. 3, pp. 456–465, 2000.