2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

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 numerical 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], [2] and economics [3], [4] etc. Closed-form optimal control law can be found for certain unconstrained problems, such as linear-quadratic control problem [5], or brachistochrone problem [6]. 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 [7] 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.

This work is supported by National Natural Science Foundation of China under grant no. 62192752.

[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 the Automatic Control Laboratory, ETH Zurich, Switzerland. (jiayan@ethz.ch)

Dynamic Matrix Control (DMC) [8] and Model Algorithmic Control (MAC) [9] are two formulations of MPC algorithm for discretized optimal control problems with constraints [10]. 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) [11] 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 [12] introduce ϵ-tolerance on inequality constraints, which means that the constraints may still be violated up to ϵ. Fu et al. [13] 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 [14]. In this way, the decision variables become finite-dimensional polynomial coefficients. On the other hand, the inequality constraints become polynomial non-negative constraints over intervals, which are still infinite-dimensional. Fortunately, we can leverage Markov-Lukács theorem [15] 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 Chambolle-Pock [16], 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 [17] due to the fact that it can be easily parallelized.

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 [14]. 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. [14] proved the following theorem:

Theorem 1 (Linear flatness [14])**.** *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. [18] 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 [18] for more details.

III. SDP Formulation of continuous MPC

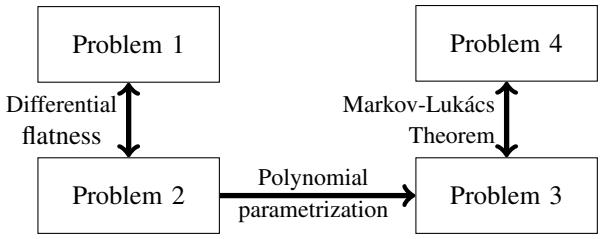


Fig. 1: The relationships between optimization problems in this paper. Double tail arrow represents that the two problems are equivalent. Single tail arrow means that Problem 3 is obtained by parameterizing Problem 2 using polynomials.

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 piece-wise polynomials. The polynomial optimization problem is then transformed into an equivalent SDP problem via Markov-Lukács Theorem [15] and Sum-of-Squares (SOS) in Subsection 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:

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

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:

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 [10].

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]

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

[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.

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

$$c_l \triangleq \begin{bmatrix} c_{l,1} \\ \vdots \\ c_{l,m} \end{bmatrix} \in \mathbb{R}^{m(d+1)\times 1},\ \boldsymbol{c} \triangleq \begin{bmatrix} c_1 \\ \vdots \\ c_N \end{bmatrix} \in \mathbb{R}^{m(d+1)N\times 1}. \tag{5}$$

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

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 P, L_j, g_j, h_j, r_j in Problem 3 is shown in our full version [19] and omitted here because of space limit.

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 [20].*
- *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 [15]. 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 [15])**.** *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*

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

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

$$f(t) = \sigma(t)^\top X \sigma(t),$$

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

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

$$\begin{aligned}(L_j c_l - g_j)^\top \gamma(t) = \\ t\sigma(t)^\top X_{j,l}^f \sigma(t) + (1-t)\,\sigma(t)^\top X_{j,l}^g \sigma(t),\end{aligned} \tag{6}$$

with positive semi-define matrices $X_j^f, X_j^g \in \mathbb{R}^{(\delta+1)\times(\delta+1)}$. By comparing the coefficients of the polynomials on the LHS and RHS of (6), we know that (6) is equivalent to:

$$L_j c_l - g_j = \mathcal{M}(X_{j,l}^f, X_{j,l}^g) \tag{7}$$

where $\mathcal{M}(\cdot,\cdot)$ is a linear function whose definition is seen in full version paper [19].

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 $\tilde{P} \in \mathbb{R}^{\mathrm{rank}(P)\times \mathrm{size}(P)}$, such that $\tilde{P}^\top \tilde{P} = P$. Notice that the following three optimization problems are equivalent where s is a scalar:

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

We arrive at the following SDP problem which is equivalent to Problem 3 and computationally tractable.The definition correspondence between $\boldsymbol{L}, \boldsymbol{M}, \boldsymbol{X}, \boldsymbol{g}, \boldsymbol{h}, \boldsymbol{r}$ and the original notations is shown in [19].

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

$$\begin{aligned}&\min_{s,\boldsymbol{c},\boldsymbol{X}} \quad s \\ &s.t.\ \boldsymbol{L}\boldsymbol{c} - \boldsymbol{M}(\boldsymbol{X}) = \boldsymbol{g},\ \boldsymbol{h}\boldsymbol{c} = \boldsymbol{r} && \text{(8a)} \\ &\boldsymbol{X} \in \mathbb{S}_+, s \geq 0,\ \begin{bmatrix}\tilde{P}\boldsymbol{c} \\ s\end{bmatrix} \in \mathrm{SOC} && \text{(8b)}\end{aligned}$$

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

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.

IV. Accelerated SDP Solving with Parallel Computing

A. Primal dual hybrid gradient algorithm for SDP solving

In this subsection, we present the primal-dual hybrid gradient algorithm that solves Problem 4. Using primal-dual operator splitting [21], 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 $\begin{bmatrix}\boldsymbol{L}^\top & \boldsymbol{h}^\top & \tilde{P}^\top\end{bmatrix}^\top \boldsymbol{c}$.

1. Primal step:

$$\boldsymbol{X}^{k+1} \leftarrow \mathrm{proj}_{\mathbb{S}_+}(\boldsymbol{X}^k - \alpha \mathcal{D}_{\boldsymbol{X}}^*(\lambda_1^k)) \tag{9}$$

$$\boldsymbol{c}^{k+1} \leftarrow \boldsymbol{c}^k - \alpha \mathcal{D}_{\boldsymbol{c}}^*(\lambda_1^k, \lambda_2^k, \lambda_3^k) \tag{10}$$

$$\begin{bmatrix}\tilde{\boldsymbol{c}}^{k+1} \\ s^{k+1}\end{bmatrix} \leftarrow \mathrm{proj}_{\mathrm{SOC}} \begin{bmatrix}\tilde{\boldsymbol{c}}^k - \alpha(-\lambda_3^k) \\ (s^k - \alpha)^+\end{bmatrix} \tag{11}$$

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

2. Calculating difference:

$$\Delta \boldsymbol{X}^{k+1} \leftarrow 2\boldsymbol{X}^{k+1} - \boldsymbol{X}^k \tag{12}$$

$$\Delta \boldsymbol{c}^{k+1} \leftarrow 2\boldsymbol{c}^{k+1} - \boldsymbol{c}^k \tag{13}$$

$$\Delta \tilde{\boldsymbol{c}}^{k+1} \leftarrow 2\tilde{\boldsymbol{c}}^{k+1} - \tilde{\boldsymbol{c}}^k \tag{14}$$

3. Dual step:

$$\lambda_1^{k+1} \leftarrow \lambda_1^k + \beta \boldsymbol{L}\Delta \boldsymbol{c}^{k+1} - \beta \boldsymbol{M}(\Delta \boldsymbol{X}^{k+1}) - \beta \boldsymbol{g} \tag{15}$$

$$\lambda_2^{k+1} \leftarrow \lambda_2^k + \beta \boldsymbol{h}\Delta \boldsymbol{c}^{k+1} - \beta \boldsymbol{r} \tag{16}$$

$$\lambda_3^{k+1} \leftarrow \lambda_3^k + \beta \tilde{P}\Delta \boldsymbol{c}^{k+1} - \beta \Delta \tilde{\boldsymbol{c}}^{k+1} \tag{17}$$

The calculation of $\mathcal{D}_{\boldsymbol{X}}^*(\lambda_1)$ and $\mathcal{D}_{\boldsymbol{c}}^*(\lambda_1, \lambda_2, \lambda_3)$ are seen in [19]. The projection to the semi-definite cone is

$$\mathrm{proj}_{\mathbb{S}_+}(X) = \sum_{i=1}^{\mathrm{size}(X)} \max\{0, \nu_i\}\, \mu_i \mu_i^\top \tag{18}$$

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

$$\mathrm{proj}_{\mathrm{SOC}} \begin{bmatrix}\boldsymbol{c} \\ s\end{bmatrix} = \begin{cases} \frac{s+\|\boldsymbol{c}\|_2}{2\|\boldsymbol{c}\|_2} \begin{bmatrix}\boldsymbol{c} \\ \|\boldsymbol{c}\|_2\end{bmatrix} & \text{if } \|\boldsymbol{c}\|_2 > s. \\ \begin{bmatrix}\boldsymbol{c} \\ s\end{bmatrix} & \text{if } \|\boldsymbol{c}\|_2 \leq s. \end{cases} \tag{19}$$

Define $\lambda = \begin{bmatrix}\lambda_1^\top & \lambda_2^\top & \lambda_3^\top\end{bmatrix}^\top$. The convergence of algorithm (9)-(17) is provided in the following.

Theorem 4 ([22])**.** *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* (9)-(17) *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 and warm start

It is worth noticing that for our proposed iterations, a significant proportion of the time will be spent on the projection $\text{proj}_{\mathbb{S}_+}(\cdot)$. 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 $\mathcal{D}^*_{\boldsymbol{X}}, \mathcal{D}^*_{\boldsymbol{c}}$, and the difference calculation in (12) are essentially tensor operations and hence can be accelerated by parallel computation. One can easily implement the computation of the proposed PDHG solver in a parallelized manner on the GPU and the computational would be significantly sped up so that real-time implementation is feasible (see Section V).

Moreover, we can also adopt a warm-start mechanism to reduce the number of iterations. One can design the control apply time to match the time interval length of each polynomial segment. Thus, after applying control for a segment, the used segment is discarded and a new segment is added at the end of the original horizon. By a "shifting" warm start mechanism, only the last segment is need to be updated by the PDHG iterations. The computational time after implementing GPU parallel and warm start is shown later in Fig. 5.

V. Simulation

The performance of the proposed MPC solver is validated on the quadruple-tank process [23]. The system has 4 states, which represent the liquid levels (in centimeter) of 4 tanks. There are two control inputs in the system: the voltage (in volt) of two pumps. The simulation employs the same linearized system equations and system parameters as [23], which are omitted due to space limitations. We first demonstrate control performance of our proposed algorithm in subsection V-A and then computation time in V-B. Our code is available on https://github.com/zs-li/MPC_PDHG.

A. Control Performance

The MPC is assigned a tracking task subject to box constraints on states and inputs, that is, the liquid levels in all tanks remain between 0 to 20 cm and the control inputs stay in between 0 to 8 V. For comparison, we employ the Quadratic Programming (QP) formulation 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 $T = 20$. Thus, the two methods are comparable in terms of horizon length and update frequency.

As shown in Fig. 4, 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.

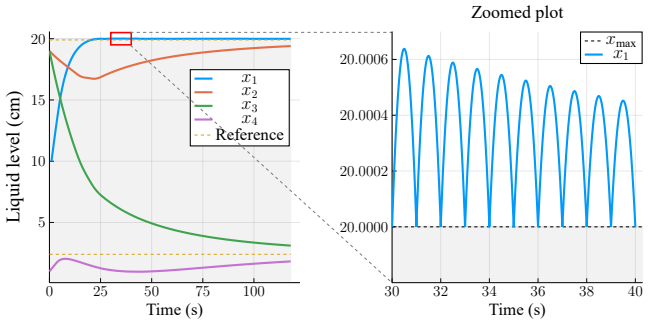


Fig. 2: The states of discrete time linear MPC using the QP solver. The gray area in the figures denotes the feasible region of states. The states between sampling times violate the constraints.

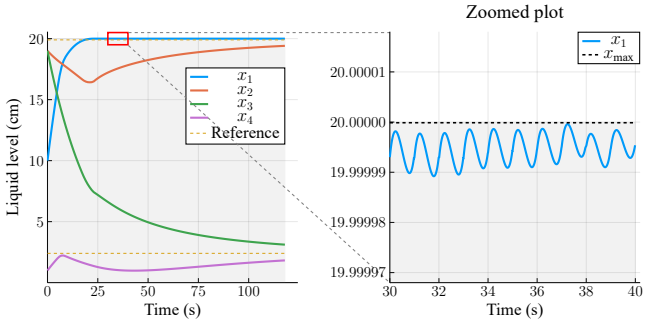
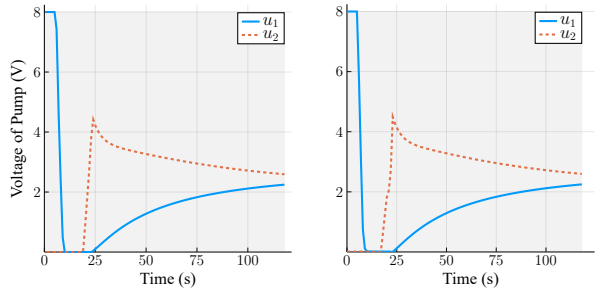


Fig. 3: The states of continuous time linear MPC using our proposed solver. The gray area in the figures denotes the feasible region of states. The states using our proposed MPC input stays in the feasible region for whole time interval.

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 of polynomial degrees d and polynomial segments N. 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 5 is the average solving time of the first 100 apply steps. The computation platform is 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. 5, 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.



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

Fig. 4: 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.

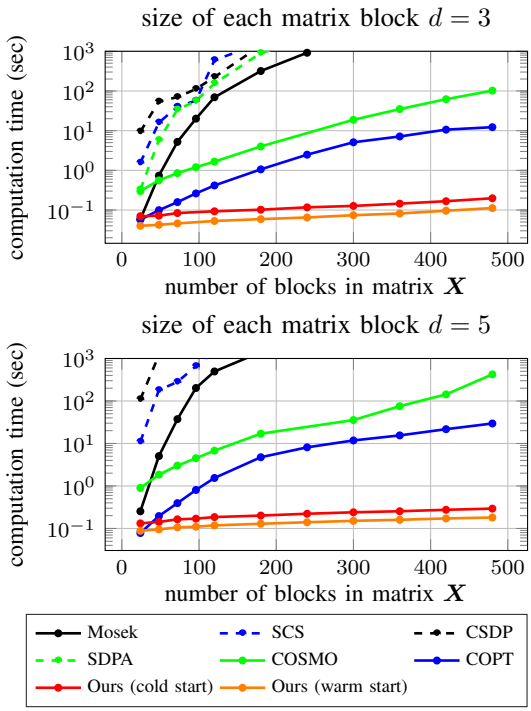


Fig. 5: The states of discrete time linear MPC using the QP solver.

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 piece-wise polynomials and utilizing the Markov-Lukács theorem from SOS theory, we transform the problem into an 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 validates the continuous-time constraint satisfaction and computational efficiency our proposed algorithm.

REFERENCES

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

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

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

[4] 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.

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

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

[7] 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.

[8] 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.

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

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

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

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

[17] 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.

[18] 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.

[19] Z. Li, B. Yang, J. Li, J. Yan, and Y. Mo, "Linear model predictive control under continuous path constraints via parallelized primal-dual hybrid gradient algorithm," 2023.

[20] 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.

[21] 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.

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

[23] 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.