Automatica 131 (2021) 109729

Contents lists available at ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

Brief paper

DMPC: A data-and model-driven approach to predictive control ☆

Hassan Jafarzadeh [a,*], Cody Fleming [b]

[a] Department of Systems Engineering, University of Virginia, 151 Engineer's Way Charlottesville, VA 22904, USA
[b] Department of Mechanical Engineering, Iowa State University, Ames, IA 50011, USA

ARTICLE INFO

Article history:
Received 4 August 2020
Received in revised form 28 February 2021
Accepted 20 April 2021
Available online 9 June 2021

Keywords:
Learning controller
Model predictive control
Data-and model-driven predictive control
Optimal control

ABSTRACT

This work presents DMPC (Data-and Model-Driven Predictive Control) to solve control problems in which some of the constraints or parts of the objective function are known, while others are entirely unknown to the controller. It is assumed that there is an exogenous "black box" system, e.g. a machine learning technique, that predicts the value of the unknown functions for a given trajectory. DMPC (1) provides an approach to merge both the model-based and black-box systems; (2) can cope with very little data and is sample efficient, building its solutions based on recently generated trajectories; and (3) improves its cost in each iteration until converging to an optimal trajectory, typically needing only a few trials even for nonlinear dynamics and objectives. Theoretical analysis of the algorithm is presented, proving that the quality of the trajectory does not worsen with each new iteration. We apply the DMPC algorithm to the motion planning of an autonomous vehicle with nonlinear dynamics.

1. Introduction

Traditional techniques for analyzing and developing control laws in safety-critical applications usually require a precise mathematical model of the system (Ames, Xu, Grizzle, & Tabuada, 2016; Taylor, Singletary, Yue, & Ames, 2020). However, there are many control applications where such precise, analytical models cannot be derived or are not readily available. System identification is a parametric model approach to such problems, mostly focusing on asymptotic error characterization or consistency guarantees, often assuming that the structure of the underlying system is known or that states are directly measurable (Matni, Proutiere, Rantzer, & Tu, 2019; Sarkar, Rakhlin, & Dahleh, 2019). On the other hand, data-driven approaches from machine learning are used in order to address these cases in a non-parametric way and often can be successful even with no assumptions about the structure of the underlying system. Such approaches can be used to identify unmodeled dynamics in a scalable way, and with high accuracy. However, an objective that is increasingly prevalent in the literature involves merging or complementing the analytical approaches from control theory with techniques from machine learning.

Recently, techniques based on model-predictive control (MPC) have addressed this problem by first using a statistical method to estimate a mathematical model that is compatible with the data, and then using this estimated model within a nominal MPC framework to find optimal trajectories and control actions. In addition to the aforementioned system identification techniques, a popular choice is to build statistical models using Gaussian Processes (GPs) (Deisenroth, Fox, & Rasmussen, 2013; Kamthe & Deisenroth, 2018), while Regression Trees and other machine learning techniques have been used in other cases (Behl, Smarra, & Mangharam, 2016). The use of GPs in the context of model-predictive control often creates highly nonlinear models, resulting in non-convex problems that are difficult to solve efficiently or online. Alternatively, approaches based on Reinforcement Learning have been applied in this setting. Model-based techniques again require a statistical method, for example, GPs or deep neural networks, to estimate transition probability distributions (Moerland, Broekens, & Jonker, 2020). Model-free methods represent, informally, a trial-and-error method for identifying control policies (Deisenroth & Rasmussen, 2011; Kamthe & Deisenroth, 2017). An open question in reinforcement learning (and indeed much of the literature that uses both control theory and machine learning) involves how to guarantee that the learned policy will not violate safety or other constraints (Bacci & Parker, 2020; Isele, Nakhaei, & Fujimura, 2018). In addition, sample complexity represents a challenge for all the aforementioned techniques and is a general problem in machine learning.

This paper seeks to leverage the notion that in many applications, some aspects of the system (and environment) may be

☆ The material in this paper was not presented at any conference. This paper was recommended for publication in revised form by Associate Editor Angelo Alessandri under the direction of Editor Thomas Parisini.

* Corresponding author.
E-mail addresses: hj2bh@virginia.edu (H. Jafarzadeh), flemingc@iastate.edu (C. Fleming).

https://doi.org/10.1016/j.automatica.2021.109729
0005-1098/

known mathematically while other aspects are unknown or represented by a so-called "black box". Our method attempts to utilize the capabilities of model-based (MPC) and data-driven (machine learning algorithm) approaches, and bring them together in a single framework in planning and control problems.

The paper addresses both sample complexity and online computational efficiency by dividing the state space, such that the dimensionality of the mathematical models and the data needed for statistical estimation and prediction are both reduced, while also accounting for the interconnection between these two classes of variables. Furthermore, we develop an algorithm that leverages this decoupling of variables, and efficiently focuses on a specific part of the state space that likely contains the optimal, feasible trajectory without sampling from the rest of the state space. Specifically, we assume that the dynamics of the system are available in the form of a known mathematical model, but there is an unknown function of the states and control inputs of the system that affects the performance index or feasible solution space. It is also assumed that the unknown aspects of the system or environment can be predicted/measured for a given system trajectory, e.g. by a "black box".

Our technique is based on notions from Iterative Learning Control (ILC). ILC is attractive because it can "learn" through repeated trials to converge to better solutions (Wang, Gao, & Doyle III, 2009). The concept of ILC has recently been extended to a framework that does not require a reference signal (Rosolia & Borrelli, 2018; Rosolia, Zhang, & Borrelli, 2017), although this approach still assumes that initial conditions, constraints, and costs remain consistent at each iteration. Although the aforementioned techniques have several nice qualities, e.g. no need for a reference signal or known cost function, they (a) assume a repetitive setting and (b) generally do not apply to so-called "black box" variables. We borrow from ILC concepts but generalize to non-repetitive or non-iterative tasks, where a controller needs to make real-time decisions in novel environments. Furthermore, our approach works when the dynamics are unknown for at least some aspects of the system or environment. The approach leverages machine learning and MPC to predict the behavior of the black-box and mathematically modeled components of the system, respectively, incorporating both into a technique called Data- and Model-driven Predictive Control (DMPC). DMPC works without a reference signal and – for a subset of the state or cost variables – completely unknown dynamics; furthermore, DMPC can work with an unknown cost function. We prove that DMPC is recursively feasible at each iteration of the algorithm, and the generated trajectories will not worsen at each iteration. This algorithm needs only a few iterations to converge to a locally optimal solution and is computationally efficient, even for nonlinear system dynamics. We also demonstrate the performance of the algorithm with an application to a motion planning problem with nonlinear dynamics in a totally unknown environment.

2. Problem statement

In this section, a formal definition of the problem is presented. Consider the dynamical system:

$$x_{t+1} = f\left(x_t, u_t\right), \tag{1}$$

where $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^m$ are the system states and control inputs, respectively, and $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is a known and in general nonlinear map which assigns the successor state x_{t+1} to state x_t and control input u_t. In this paper we address the following infinite time optimal control problem to find an optimal trajectory from an initial state x_S to final state x_F within the feasible state vector space $\mathcal{X}$ and control vector space $\mathcal{U}$:

$$\mathcal{J}_{0\to\infty}(x_S) = \min_{u_0,u_1,\dots} \sum_{t=0}^{\infty} \left[h\left(x_t, u_t\right) + \hat{z}\left(x_t, u_t\right)\right] \tag{2a}$$

$$\text{s.t. } x_{t+1} = f\left(x_t, u_t\right) \quad \forall t \geq 0 \tag{2b}$$

$$x_0 = x_S \tag{2c}$$

$$x_t \in \mathcal{X}, \quad u_t \in \mathcal{U} \quad \forall t \geq 0, \tag{2d}$$

where (2b) and (2c) are the system dynamics and the initial conditions, and (2d) are the state and input constraints. The cost function involves two different stage costs. (i) $h()$: a **known function** that can be defined by a precise mathematical model, often based on first principles from physics. We call this a "*model-driven*" function. The traditional cost function of MPC, containing quadratic terms to drive the state of the system to an equilibrium point and to penalize the applied control input, consists of model-driven functions. (ii) $\hat{z}()$: an **unknown function** to the controller. A mathematical model cannot be defined for this type of stage cost (or at least it is too expensive to derive such a function and solve the resulting optimization model), but it affects the overall cost function. It is assumed that, given the inputs, the controller has access to the output of this function. Improving an aircraft's flight safety under the presence of turbulence can be modeled as (2), where the behavior, location, and prediction of turbulent air comes from an unknown function (unknown to the controller). Another example involves connected autonomous vehicles (CAVs) (Jafarzadeh & Fleming, 2019; Soltanaghaei, Elnaggar, Kleeman, Whitehouse, & Fleming, 2019), in which the unknown function is a model of the wireless channel and can be predicted by e.g. recurrent neural networks (Liu & Shoji, 2019; Manh & Alaghband, 2018).

It is assumed that the model-driven stage cost $h(\cdot,\cdot)$ in Eq. (2a) is continuous and satisfies $h\left(x_F, 0\right) = 0$,

$$h\left(x_t, u_t\right) \succ 0 \ \forall x_t \in \mathbb{R}^n \backslash \{x_F\}, u_t \in \mathbb{R}^m \backslash \{0\}$$

where the final state x_F is a feasible equilibrium for the unforced system (1), $f(x_F, 0) = x_F$. In the second term of the cost function, $\hat{z}()$ is considered to be positive definite and unknown for the controller, $\hat{z} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^+$. There is an exogenous data-driven system acting as a black box, such as Long short-term memory (LSTM) that calculates $\hat{z}$, given x_t and u_t. Also, we assume that the condition $\hat{z}\left(x_F, 0\right) = 0$ is held in the equilibrium point x_F.

In the case that an unknown inequality is imposed as a constraint to the model rather than a penalty in the cost function, we can use a barrier function to transform it to model (2). If we write these constraints as $\hat{y}\left(x_t, u_t\right) \leqslant 0, \quad \forall t \geqslant 0$, the barrier function can be defined as

$$\hat{z}\left(x_t, u_t\right) = \begin{cases} -\frac{1}{\hat{y}(x_t, u_t)} & \text{if } \hat{y}\left(x_t, u_t\right) < 0 \\ \infty & o.w. \end{cases} \tag{3}$$

in the exogenous data-driven system, where the controller will receive the value of $\hat{z}()$ calculated from Eq. (3) and then considers this value as a prediction for the unknown cost in the performance index shown in model (2). Therefore, the problem involves generating an optimal sequence of control inputs that steers the system (1) from the initial state x_S to the equilibrium point x_F such that the cost function of optimal control problem (2), $\mathcal{J}_{0\to\infty}(x_S)$ – which is a combination of a known stage cost $h()$, and unknown stage cost $\hat{z}()$ functionals – achieves the minimum value.

At each time step of a (perhaps previously unseen) control task, the approach uses an iterative scheme, where it learns from each iteration and optimizes model (2) without explicitly determining the unknown function $\hat{z}()$. At iteration j, the following vectors collect the inputs applied to the system (1) and the corresponding state evolution from initial state x_S to the equilibrium point x_F:

$$\mathbf{x}^{*,j} = [x_0^j, x_1^{*,j}, \dots, x_t^{*,j}, \dots, x_F] \tag{4a}$$

$$\mathbf{u}^{*,j} = [u_0^{*,j}, u_1^{*,j}, \dots, u_t^{*,j}, \dots]. \tag{4b}$$

In (4), the optimal values of system state and the control input obtained at time t and iteration j are denoted by $x_t^{*,j}$ and $u_t^{*,j}$, respectively. Also, we assume that at each jth iteration, the trajectories start from the same initial condition $x_0^j = x_S, \quad \forall j \geq 0$.

3. DMPC approach

This section describes the DMPC approach to obtain vectors (4) as a sub-optimal solution for the infinite time optimal control problem (2). We begin with the following assumption, as the DMPC algorithm is designed such that, starting from a given initial trajectory, it converges to the optimal solution (trajectory) repetitively.

Assumption 1. Similar to the iterative learning control methods (Rosolia & Borrelli, 2018; Rosolia et al., 2017), it is assumed that there exists an initial feasible trajectory $\mathbf{x}^0$ for the infinite time optimal control problem (2) from the initial state, x_S, to the equilibrium point, x_F, at the first iteration but with no assumptions on optimality.

In addition, the concept of cost-to-go is defined for each state in a complete trajectory as the minimum cost of reaching the equilibrium point x_F from the current state. The algorithm records the last successful complete trajectory (i.e. from initial state x_s to the equilibrium point x_F), $\mathbf{x}^{*,j-1}$, and assigns to every state in this set a cost-to-go value obtained at iteration $j-1$,

$$\mathbf{q}^{j-1} = [q^{j-1}(x_S), \dots, q^{j-1}(x_t^{*,j-1}), \dots, q^{j-1}(x_F)].$$

The cost of following the trajectory obtained at iteration $j-1$ from state $x_t^{*,j-1}$ to final state x_F can be defined as:

$$q^{j-1}(x_t^{*,j-1}) = \mathcal{J}_{t\to\infty}^{*,j-1}(x_t^{*,j-1}), \quad \forall t \geq 0.$$

The main approach of DMPC is generating a full trajectory from x_S to x_F at iteration j, $\mathbf{x}^{*,j}$, based on the full trajectory generated at iteration $j-1$, $\mathbf{x}^{*,j-1}$. The full trajectory $\mathbf{x}^{*,j}$ is built iteratively from the initial state x_S to the final state x_F. At each time step t of iteration j, DMPC finds the optimal control input, $\mathbf{u}_{t:t+N|t}^j$, and associated trajectory, $\mathbf{x}_{t:t+N|t}^j$

$$\mathbf{x}_{t:t+N|t}^j = [x_t^{*,j}, \dots, x_{t+N|t}^j] \tag{5a}$$

$$\mathbf{u}_{t:t+N|t}^j = [u_{t|t}^j, \dots, u_{t+N-1|t}^j]. \tag{5b}$$

where $x_t^{*,j} = x_{t|t}^j$ is the current state of the system, which is considered as the optimal state of the trajectory at iteration j at time t. DMPC selects the last state in (5a), $x_{t+N|t}^j$, from a special set that results in a recursive feasibility guarantee.

At iteration j, DMPC is designed by repeatedly solving a finite time optimal control problem in a receding horizon fashion to obtain state and control input vectors (5). In the state vector (5a), the last state, $x_{t+N|t}^j$, is enforced to be selected from set $\mathcal{S}_t^j$, that is

$$\mathcal{S}_t^j = \left(\bigcup_{t=0}^{\infty} x_t^{*,j-1}\right) \cap \mathcal{R}_N(x_t^{*,j}). \tag{6}$$

The first term in Eq. (6) is the set of all the states in the most recently generated full trajectory (iteration $j-1$), $\mathbf{x}^{*,j-1}$, and the second term is *N-step reachable set* from state $x_t^{*,j}$. All the states in trajectory $\mathbf{x}^{*,j-1}$ are members of *control invariant set* $\mathcal{C} \subseteq \mathcal{X}$, because, for every point in the set, there exists a feasible control action in input vector $\mathbf{u}^{*,j-1}$, that satisfies the state and control constraints and steers the state of the system (1) toward the equilibrium point x_F. Therefore, forcing the controller to select the terminal state $x_{t+N|t}^j$ from the set $\mathcal{S}_t^j$ keeps the state of the system in set $\mathcal{C}$ for time steps beyond the time horizon N (Firoozi, Nazari, Guanetti, O'Gorman, & Borrelli, 2018), i.e.

$$\text{if} \quad x_{t+N}^j \in \mathcal{C} \Rightarrow x_{t+N+k}^j \in \mathcal{C} \quad \forall k > 0, \tag{7}$$

On the other hand, trajectory $\mathbf{x}_{t:t+N|t}^j$ drives the system (1) from state $x_t^{*,j}$ to one of the states in set $\mathcal{S}_t^j$ in N time steps (see Fig. 1). Therefore, $\mathcal{S}_t^j$ is a subset of the control invariant set and N-step reachable set, making the state $x_t^{*,j}$ a subset of the maximal stabilizable set. Intuitively, this guarantees the constraint satisfaction and feasibility for all time steps ($t \geqslant 0$) (the feasibility will be proven in *Theorem* 1). This means that constraint satisfaction at time steps beyond the time horizon does not depend on the length of the time horizon, and N can be picked freely; in this work we will select N to be small to speed up the algorithm. We denote each state in set $\mathcal{S}_t^j$ by $s_t^{i,j}, \forall i \in \{1, \dots, |\mathcal{S}_t^j|\}$.

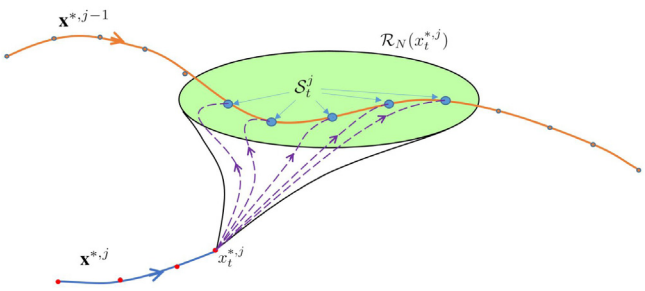

Fig. 1. The green area shows the N-step reachable set, $\mathcal{R}_N(x_t^{*,j})$, from current state, $x_t^{*,j}$. Controllable set $\mathcal{S}_t^j$ is illustrated by large blue dots and dashed purple line segments are the optimal trajectories from current state to available states in controllable set $\mathcal{S}_t^j$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3.1. Algorithmic details

To find the (local) optimal trajectory $\mathbf{x}_{t:t+N|t}^j$ in (5), DMPC generates two trajectories $\bar{\mathbf{x}}_{t:t+N|t}^j$ and $\mathbf{x}_{t:t+N|t-1}^j$, and selects the best of them based on their cost as $\mathbf{x}_{t:t+N|t}^j$. We now explain how these two trajectories are built.

(i) The *first* trajectory generated by DMPC is $\bar{\mathbf{x}}_{t:t+N|t}^j$, illustrated by a solid black trajectory in Fig. 2. This trajectory is the state vector associated with the optimal control input $\bar{\mathbf{u}}_{t:t+N|t}^j$ obtained from the following optimization model over all the candidate terminal states that are reachable in N time steps from the current state $x_t^{*,j}$, see Eq. (6). This set of terminal states is depicted by big blue points in Fig. 1 and indexed by $i \in \{1, \dots, |\mathcal{S}_t^j|\}$ in the following term

$$\bar{\mathbf{u}}_{t:t+N|t}^j = \underset{\mathbf{u}_{t:t+N|t}^{i,j}}{\operatorname{argmin}} \left\{ \mathcal{J}_{t\to t+N}^{i,j}(x_t^{*,j}), \ \forall i \right\} \tag{8}$$

where $\mathcal{J}_{t\to t+N}^{i,j}(x_t^{*,j})$ is the predicted overall cost (i.e. summation of both the model-based $\sum h(.)$ and black-box $\sum \hat{z}(.)$ costs) due to the system following the control input $\mathbf{u}_{t:t+N}^{i,j}$ to reach the terminal state $x_{t+N|t}^{i,j} = s_t^{i,j}$. To simplify the mathematical notations, we will use $\hat{z}_{k|t}^{i,j}$ to show the predicted value of the unknown function following the control input $\mathbf{u}_{t:t+N}^{i,j}$, instead of $\hat{z}(x_{k|t}^{i,j}, u_{k|t}^{i,j})$. Then the value of $\mathcal{J}_{t\to t+N}^{i,j}(x_t^{*,j})$ can be defined as:

$$\mathcal{J}_{t\to t+N}^{i,j}(x_t^{*,j}) = J_{t\to t+N}^{i,j}(x_t^{*,j}) + \sum_{k=t}^{t+N-1} \hat{z}_{k|t}^{i,j}. \tag{9}$$

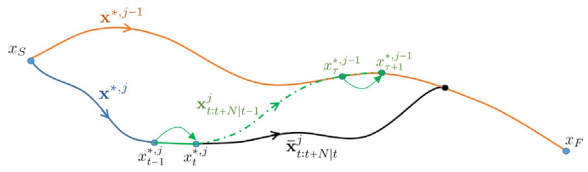

Fig. 2. DMPC generates two trajectories $\bar{\mathbf{x}}^j_{t:t+N|t}$ (solid black) and $\mathbf{x}^j_{t:t+N|t-1}$ (dashed green) at time step t of iteration j, and selects best of them. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

To find the optimal control input $\bar{\mathbf{u}}^j_{t:t+N|t}$ in Eq. (8), we first use the following formulation to generate $\mathbf{u}^{i,j}_{t:t+N|t}$ and $\mathbf{x}^{i,j}_{t:t+N|t}$ from state $x^{*,j}_t$ toward terminal state $s^{i,j}_t \in \mathcal{S}^j_t$, $\forall i \in \{1,\dots,|\mathcal{S}^j_t|\}$, and calculate the cost associated with the model-based term, which is denoted by $J^{i,j}_{t\to t+N}(x^{*,j}_t)$ in Eq. (9):

$$J^{i,j}_{t\to t+N}(x^{*,j}_t) = \min_{\mathbf{u}^{i,j}_{t:t+N}} \sum_{k=t}^{t+N-1} \ell(x^{i,j}_{k|t}, u^{i,j}_{k|t}) + (N+1)q^{j-1}(x^{i,j}_{t+N|t}) \tag{10a}$$

$$\text{s.t. } x^{i,j}_{k+1|t} = f(x^{i,j}_{k|t}, u^{i,j}_{k|t}) \quad \forall k \tag{10b}$$

$$x^{i,j}_{t|t} = x^{*,j}_t \tag{10c}$$

$$x^{i,j}_{t+N|t} = s^{i,j}_t \tag{10d}$$

$$x^{i,j}_{k|t} \in \mathcal{X},\ u^{i,j}_{k|t} \in \mathcal{U}, \quad \forall k. \tag{10e}$$

In this model, the predictive controller generates the best trajectory to reach state $s^{i,j}_t$ (i.e. enforced by constraint (10d)) and adds the cost to go $(N+1)q^{j-1}(x^{i,j}_{t+N|t})$ to compensate for the remaining cost from state $s^{i,j}_t$ to the final state x_F. We replace the stage cost $h(.,.)$ with a positive definite function $\ell(.,.)$ in the cost function

$$\ell(x^{i,j}_{k|t}, u^{i,j}_{k|t}) = \|x^{i,j}_{k|t} - x^{i,j}_{t+N|t}\|^2_P + \|u^{i,j}_{k|t}\|^2_R,$$

where P and R are positive (semi)definite tuning matrices. The function $h()$ in the general optimal control problem(2) penalizes the controller according to the difference between the generated state $x^{i,j}_{k|t}$ and the final state x_F, but $\ell(.,.)$ considers the selected terminal state $x^{i,j}_{t+N|t}$ instead of x_F. To compensate for the remaining trajectory cost from $x^{i,j}_{t+N|t}$ to x_F, we add a cost-to-go $q^{j-1}(x^{i,j}_{t+N|t})$ for each $N+1$ states in the trajectory. Constraint (10d) enforces that the controller steers the system to a specific terminal state, $s^{i,j}_t$.

The objective optimized by model (10) does not involve the cost value coming from the black-box variables, $\sum_{k=t}^{t+N-1} \hat{z}^{i,j}_{k|t}$. However, given the trajectory $\mathbf{x}^{i,j}_{t:t+N|t}$ generated by model (10), the value of this unknown function can be predicted by the external black-box system and added to $J^{i,j}_{t\to t+N}(x^{*,j}_t)$ to find $\mathcal{J}^{i,j}_{t\to t+N}(x^{*,j}_t)$ based on Eq. (9). Then, according to (8), between all of the trajectories that start from $x^{*,j}_t$ and reach the terminal states in set $\mathcal{S}^j_t$ which are counted by index i (dashed purple trajectories in Fig. 1), the trajectory that has the minimum cost value is selected and denoted $\bar{\mathbf{x}}^j_{t:t+N|t}$. This is the result for (8), where the input sequence $\bar{\mathbf{u}}^j_{t:t+N|t}$ produces the overall trajectory cost of $\bar{\mathcal{J}}^j_{t\to t+N}(x^{*,j}_t)$.

(ii) The *second* trajectory generated by DMPC is $\mathbf{x}^j_{t:t+N|t-1}$, that is illustrated by a dashed green trajectory in Fig. 2. In addition to $\bar{\mathbf{x}}^j_{t:t+N|t}$, another feasible available trajectory starting from x_S to x_F can be obtained from the solution of the previous time step $t-1$ at the current iteration j. This trajectory is generated by applying one more step of the control input, $\mathbf{u}^j_{t-1:t+N-1|t-1}$, to the trajectory of the previous time step $t-1$ and shifting its state one time step toward the final state x_F along the optimal trajectory of iteration $j-1$. This trajectory can be written as follows:

$$\mathbf{x}^j_{t:t+N|t-1} = \left[x^j_{t|t-1}, \dots, x^j_{t+N-2|t-1}, x^{*,j-1}_\tau, x^{*,j-1}_{\tau+1}\right] \tag{11a}$$

$$\mathbf{u}^j_{t:t+N|t-1} = \left[u^j_{t|t-1}, \dots, u^j_{t+N-2|t-1}, u^{*,j-1}_\tau\right]. \tag{11b}$$

$x^{*,j-1}_\tau$ denotes the optimal terminal state selected from the last iteration (i.e. the last generated complete trajectory) $j-1$, and τ is the time index of this state, $x^j_{t+N-1|t-1} = x^{*,j-1}_\tau$. The overall trajectory cost of $\mathbf{x}^j_{t:t+N|t-1}$ is given by $\mathcal{J}^j_{t\to t+N|t-1}(x^{*,j}_t)$ and is

$$\mathcal{J}^j_{t\to t+N|t-1}(x^{*,j}_t) = \sum_{k=t}^{t+N-2} \left[\ell(x^j_{k|t-1}, u^j_{k|t-1}) + \hat{z}^j_{k|t-1}\right] + Nq^{j-1}(x^j_{t+N-1|t-1}) + q^{j-1}(x^{*,j-1}_{\tau+1}). \tag{12}$$

Finally, the best trajectory of time step t and iteration j ($\mathbf{u}^j_{t:t+N|t}$ and $\mathbf{x}^j_{t:t+N|t}$) is selected between two obtained trajectories, $\bar{\mathbf{x}}^j_{t:t+N|t}$ and $\mathbf{x}^j_{t:t+N|t-1}$ based on their cost.

$$\mathcal{J}^j_{t\to t+N}(x^{*,j}_t) = \min\{\bar{\mathcal{J}}^j_{t\to t+N}(x^{*,j}_t), \mathcal{J}^j_{t\to t+N|t-1}(x^{*,j}_t)\}. \tag{13}$$

In other words, the algorithm selects between two trajectories: (a) the minimum-cost feasible trajectory from $t \to t+N$ at time step t of iteration j, and (b) the time-shifted trajectory from $t-1 \to t+N$ that leverages information from the prior time step $t-1$ of iteration j. After finding $\mathbf{x}^j_{t:t+N|t}$ and $\mathbf{u}^j_{t:t+N|t}$, the first step of its control input is applied to the system to push its state toward the equilibrium point, $u^{*,j}_t = u^j_{t|t}$, $\quad x^{*,j}_{t+1} = x^j_{t+1|t}$.

3.2. Theoretical analysis

In the remainder of this section, we provide theoretical analyses of the algorithm for the feasibility and optimality of the generated solutions.

Theorem 1. *In the DMPC scheme with given system (1), cost function (8), and constraints (10b)–(10e), if there is a feasible trajectory at iteration $j-1$, DMPC is feasible at the next iteration, j, as well.*

Proof. To prove this theorem, first, we need to show that, given a feasible solution at time step $t-1$ of iteration j, DMPC is feasible for the next time step, t, too. The solution of DMPC at iteration $j-1$ is

$$\mathbf{x}^{*,j-1} = \left[x_S, x^{*,j-1}_1, \dots, x^{*,j-1}_t, \dots, x_F\right],$$

and at iteration j and time step $t-1$ is:

$$\mathbf{x}^j_{t-1:t+N-1|t-1} = [x^{*,j}_{t-1}, x^j_{t|t-1}, \dots, x^j_{t+N-1|t-1}]$$
$$\mathbf{u}^j_{t-1:t+N-1|t-1} = [u^j_{t-1|t-1}, \quad u^j_{t|t-1}, \dots, u^j_{t+N-2|t-1}].$$

According to constraint (10d), DMPC selects terminal state $x^j_{t+N-1|t-1}$ from set $\mathcal{S}^j_{t-1}$ which is denoted by $s^{i,j}_{t-1}$. Because $s^{i,j}_{t-1} \in \mathbf{x}^{*,j-1}$, we know that $x^j_{t+N-1|t-1} \in \mathbf{x}^{*,j-1}$. Let us assume that $x^j_{t+N-1|t-1} = x^{*,j-1}_\tau$. Based on the assumption given in the theorem (existence of a feasible trajectory at iteration $j-1$), for every state in trajectory $\mathbf{x}^{*,j-1}$ there is a feasible sequence of control actions that satisfies the constraints and steers the system toward the final state x_F. This feasible trajectory for state $x^{*,j-1}_\tau$ can be shown as:

$$\mathbf{x}^{*,j-1}_{\tau:\infty} = [x^{*,j-1}_\tau, x^{*,j-1}_{\tau+1}, \dots, x_F]$$
$$\mathbf{u}^{*,j-1}_{\tau:\infty} = [u^{*,j-1}_\tau, u^{*,j-1}_{\tau+1}, \dots].$$

Then there is at least one feasible trajectory at time step t and iteration j that is constructed as:

$$\mathbf{x}^j_{t:\infty} = [x^j_{t|t-1}, \dots, x^j_{t+N-2|t-1}, x^{*,j-1}_{\tau}, x^{*,j-1}_{\tau+1}, \dots, x_F]$$
$$\mathbf{u}^j_{t:\infty} = [u^j_{t|t-1}, \dots, u^j_{t+N-2|t-1}, u^{*,j-1}_{\tau}, u^{*,j-1}_{\tau+1}, \dots].$$

This completes the proof of the statement that DMPC is feasible at time step t if it is feasible at $t-1$. Also, based on Assumption 1 and by induction we can conclude that DMPC is feasible for all iterations and time steps. ■

We showed that, given a feasible initial trajectory $\mathbf{x}^0$, the algorithm is feasible at every time step of different iterations. Theorem 2 proves that the algorithm will finally converge to the equilibrium point x_F, and Theorem 3 proves that the performance index is non-increasing at every DMPC iteration. The next two theorems follow a similar approach to Rosolia and Borrelli (2018).

Theorem 2. *In the DMPC scheme with given system (1), cost function (8), constraints (10b)–(10e), and an initial feasible trajectory $\mathbf{x}^0$, the equilibrium point x_F is asymptotically stable at every iteration $j \geqslant 1$.*

Proof. Let us start with writing the overall optimal trajectory cost of state $x^{*,j}_{t-1}$

$$\mathcal{J}^j_{t-1\to t+N-1}(x^{*,j}_{t-1}) = (N+1)q^{j-1}(x^j_{t+N-1|t-1}) + \sum_{k=t-1}^{t+N-2}\left[\ell(x^j_{k|t-1}, u^j_{k|t-1}) + \hat{z}^j_{k|t-1}\right]$$
$$= \ell(x^j_{t-1|t-1}, u^j_{t-1|t-1}) + \hat{z}^j_{t-1|t-1} + q^{j-1}(x^j_{t+N-1|t-1}) + \sum_{k=t}^{t+N-2}\ell(x^j_{k|t-1}, u^j_{k|t-1}) + Nq^{j-1}(x^j_{t+N-1|t-1}) + q^{j-1}(x^{*,j-1}_{\tau+1})$$

where $q^{j-1}(x^{*,j-1}_{\tau+1}) = \sum_{k=\tau+1}^{\infty}\left[h\left(x^{*,j-1}_k, u^{*,j-1}_k\right) + \hat{z}^{*,j-1}_k\right]$. Using Eq. (12),

$$\mathcal{J}^j_{t-1\to t+N-1}(x^{*,j}_{t-1}) = \mathcal{J}^j_{t\to t+N|t-1}(x^{*,j}_t) + \ell(x^j_{t-1|t-1}, u^j_{t-1|t-1}) + \hat{z}^j_{t-1|t-1} + q^{j-1}(x^j_{t+N-1|t-1}).$$

Also, according to Eq. (13),

$$\mathcal{J}^j_{t\to t+N}(x^{*,j}_t) \leqslant \mathcal{J}^j_{t\to t+N|t-1}(x^{*,j}_t).$$

From the last two inequalities we conclude that

$$\mathcal{J}^j_{t\to t+N}(x^{*,j}_t) - \mathcal{J}^j_{t-1\to t+N-1}(x^{*,j}_{t-1}) \leqslant -\ell(x^j_{t-1|t-1}, u^j_{t-1|t-1}) - \hat{z}^j_{t-1|t-1} - q^{j-1}(x^j_{t+N-1|t-1}) < 0, \quad \forall t \geqslant 1, \quad \text{and} \quad \forall j \geqslant 1. \tag{14}$$

This completes the proof of asymptotic stability of the equilibrium point x_F. ■

Theorem 3. *In the DMPC scheme with given system (1), cost function (8), and constraints (10b)–(10e), and a feasible trajectory $\mathbf{x}^{*,j-1}$ at iteration $j-1$,*

$$\mathcal{J}^{*,j}_{0\to\infty}(x_S) \leqslant \mathcal{J}^{*,j-1}_{0\to\infty}(x_S), \quad \forall j \geqslant 1 \tag{15}$$

the next trajectory $\mathbf{x}^{*,j}$ generated by DMPC has an overall trajectory cost, $\mathcal{J}^{*,j}_{t\to\infty}(x_S)$, not worse than $\mathcal{J}^{*,j-1}_{t\to\infty}(x_S)$

Proof. Assume that, at iteration j, the trajectory $\mathbf{x}^{*,j-1}$ is available for an overall cost of $\mathcal{J}^{*,j-1}_{0\to\infty}(x_S)$. It is desirable to show that, according to model (8) and Eq. (13), DMPC will generate trajectory $\mathbf{x}^j_{0:N}$ (trajectory blue) which is not worse than $\mathbf{x}^{*,j-1}$, $\mathcal{J}^j_{0\to\infty}(x_S) \leqslant \mathcal{J}^{*,j-1}_{0\to\infty}(x_S)$. Positive definiteness of z and h indicates that at different time steps, t, in iteration j

$$\mathcal{J}^j_{t\to t+N}(x^{*,j}_t) \leqslant \mathcal{J}^j_{t-1\to t+N-1}(x^{*,j}_{t-1}), \quad \forall t \geqslant 1. \tag{16}$$

Also, according to Eq. (14), for $t=1$

$$\mathcal{J}^j_{0\to N}(x_S) \geqslant \mathcal{J}^j_{1\to N+1}(x^{*,j}_1) + \ell(x_S, u^{*,j}_0) + \hat{z}^{*,j}_0 + q^{j-1}(x^j_{N|0})$$

for $t=2$,

$$\mathcal{J}^j_{1\to N+1}(x^{*,j}_1) \geqslant \mathcal{J}^j_{2\to N+2}(x^{*,j}_2) + \ell(x^{*,j}_1, u^{*,j}_1) + \hat{z}^{*,j}_1 + q^{j-1}(x^j_{N+1|1})$$

until $t \to \infty$, in which the system converges to x_F. Summing up these inequalities results in

$$\mathcal{J}^j_{0\to N}(x_S) \geqslant \sum_{k=0}^{\infty}\left[\ell(x^{*,j}_k, u^{*,j}_k) + \hat{z}^{*,j}_k + q^{j-1}(x^j_{k+N|k})\right].$$

The right-hand side of this inequality is the sum of all stage costs of optimal trajectory generated at iteration j $\mathcal{J}^{*,j}_{0\to\infty}(x_S) = \sum_{k=0}^{\infty}\left[\ell(x^{*,j}_k, u^{*,j}_k) + \hat{z}^{*,j}_k + q^{j-1}(x^j_{k+N|k})\right]$, which yields the following inequality

$$\mathcal{J}^j_{0\to N}(x_S) \geqslant \mathcal{J}^{*,j}_{0\to\infty}(x_S). \tag{17}$$

From the last two inequalities we can easily conclude that

$$\mathcal{J}^{*,j-1}_{0\to\infty}(x_S) \geqslant \mathcal{J}^j_{0\to N}(x_S) \geqslant \mathcal{J}^{*,j}_{0\to\infty}(x_S), \tag{18}$$

which shows, the overall cost of trajectories does not increase by the number of iterations

$$\mathcal{J}^{*,j}_{0\to\infty}(x_S) \leqslant \mathcal{J}^{*,j-1}_{0\to\infty}(x_S), \quad \forall j \geqslant 1, \tag{19}$$

and the proof is complete. ■

4. Example

We apply the proposed DMPC algorithm on the motion planning of an autonomous vehicle with a kinematic bicycle model in an inertial frame (Kong, Pfeiffer, Schildbach, & Borrelli, 2015). $\hat{z}$ is an unknown function and it is assumed that, given a trajectory, there is a black-box system that can predict its outputs and pass these to the controller. An example application of such a setting (see Fig. 3) involves motion planning in an environment with regions that have different cost values, where the associated cost of selected states can be predicted by a machine learning-based black box. In motion planning, such black-box variables could include predictions of other agents' states or simply a region with uneven terrain or a potentially dangerous zone for a robot. The infinite time optimal control problem is defined according to model (2), where $f(\mathrm{x}_t, \mathrm{u}_t)$ is defined as follows:

$$\dot{x}_t = v_t\ cos(\psi_t + \beta_t) \tag{20a}$$
$$\dot{y}_t = v_t\ sin(\psi_t + \beta_t) \tag{20b}$$
$$\dot{\psi}_t = \frac{v_t}{l_r}\ sin(\beta_t) \tag{20c}$$
$$\dot{v}_t = a_t. \tag{20d}$$

The state and control input vectors are $\mathrm{x}_t = [x_t\ y_t\ \psi_t\ v_t]^T$, $\mathrm{u}_t = [\delta_t\ a_t]^T$, respectively. x_t and y_t are the coordinates of the center of mass of the vehicle, ψ_t is the heading angle, and v_t is the velocity of the vehicle at time step t. l_f and l_r show the distance of the center of the mass from the front and rear axles, respectively. $\beta_t = tan^{-1}(\frac{l_r}{l_f+l_r}\ tan(\delta_t))$ is the angle between the current velocity vector of the center of mass and the longitudinal axis of the vehicle. The control input vector u_t is composed of

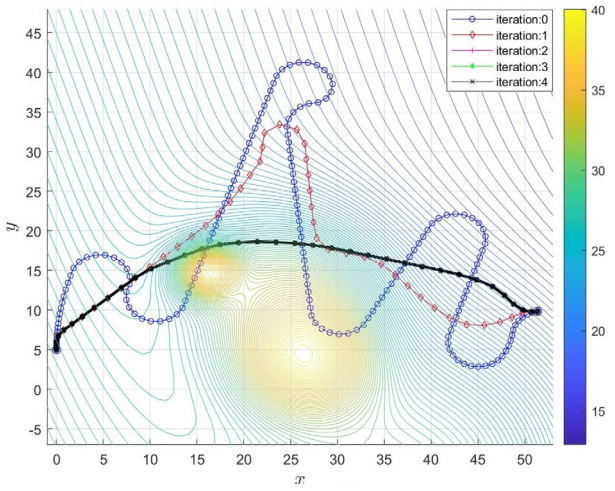

Fig. 3. The contour plot of unknown non-convex cost function, and local optimal trajectory generated by DMPC. The contours are totally unknown to the controller. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

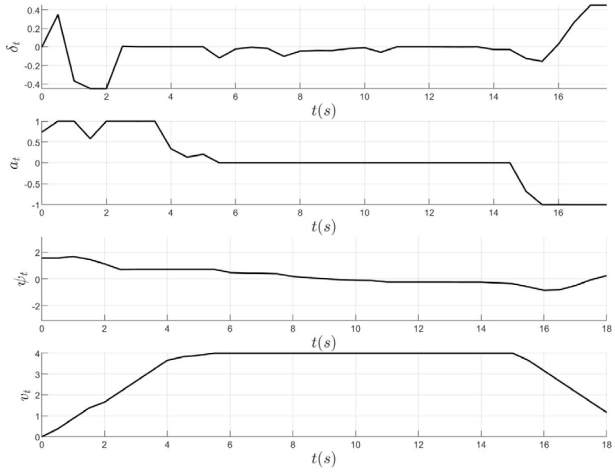

Fig. 4. Control inputs and states in the steady state.

the steering angle δ_t and the acceleration a_t that is defined for the center of mass in the same direction as v_t.

The upper and lower bounds of the state and control vectors are $\mathrm{x}_{min} = [-\infty \ -\infty \ 0 \ 0]^T$, $\mathrm{x}_{max} = [+\infty \ +\infty \ 2\pi \ 4]^T$, $\mathrm{u}_{min} = [-\frac{\pi}{7} \ -1]^T$ and $\mathrm{u}_{max} = [\frac{\pi}{7} \ 1]^T$. The equality constraint representing initial state x_0 is assumed to be $\mathrm{x}_S = [0 \ 5 \ \frac{\pi}{2} \ 0]$. Function $J_{0\to\infty}(\mathrm{x}_0)$ shows the overall cost imposed to the controller to steer the system from initial state x_0 to final state $\mathrm{x}_F = [51 \ 10 \ \frac{\pi}{10} \ 1.1]^T$. The stage cost $h(.,.)$ is defined as a quadratic function $h(\mathrm{x}_t, \mathrm{u}_t) = (\mathrm{x}_t - \mathrm{x}_F)^T P(\mathrm{x}_t - \mathrm{x}_F) + \mathrm{u}_t^T R\mathrm{u}_t$. The tuning matrices of the cost function are $P = diag[1 \ 1 \ 0.1 \ 0.1]$ and $R = diag[0.01 \ 0.01]$. In this example, the DMPC controller is expected to improve the given initial trajectory (blue circle trajectory in Fig. 3) in the presence of an unknown cost function. The controller will use the most recently generated trajectory to converge to an optimal trajectory. The algorithm will stop if $\sum_{t=0}^{\infty} |\mathrm{x}_t^j - \mathrm{x}_t^{j-1}| < 10^{-4}$. Also, the time step and time horizon are assumed to be 0.5 s and $N = 12$, respectively for this problem. We used ACADO Code Generation tool (Houska, Ferreau, & Diehl, 2011) with MATLAB to solve this problem, and DMPC converged after 4 iterations (trajectories 2 and 3 are very close to the optimal solution that makes them invisible in the figure). Figs. 3 and 4 depict the generated trajectories $\mathbf{x}^{*,j}$ $\forall j \geqslant 0$, and optimal steering angle and acceleration/deceleration as control inputs, velocity and heading angle at different time steps.

Reinforcement learning (RL) is a natural candidate for comparison, but these approaches typically require a large number of interactions with the unknown system/function to learn controllers, which is a practical limitation in real cases, such as robots, where these number of interactions can be impractical, unsafe, and time-consuming (Deisenroth et al., 2013). In this group of applications Gaussian Process-based MPC outperforms the RL approaches, so we compare the performance of the DMPC with state-of-the-art GP methods (Deisenroth & Rasmussen, 2011; Kamthe & Deisenroth, 2018). We consider a Gaussian Process setting where we seek deterministic control inputs $\mathbf{u}_t$ that minimize the cost function of the following finite time optimal control problems, which will be solved in a receding horizon fashion until reaching the terminal state

$$\min_{\mathbf{u}_t} \left\{ J_{t\to t+N}(x_t) + \sum_{k=t}^{t+N} \mathbb{E}_{x_{k|t}}[\hat{z}(x_{k|t})] \right\},$$

where $J_{t\to t+N}(x_t)$ denotes the conventional stage cost and $\mathbb{E}_{x_{k|t}}[\hat{z}(x_{k|t})]$ denotes the expected data-driven cost at time step k calculated at time t. To implement the GP we define the training input and target data to be $\tilde{\mathbf{x}} = [x \ y]^T$ and $\tilde{z}$ respectively. We refer the reader to Deisenroth and Rasmussen (2011), Kamthe and Deisenroth (2017) for details of the PILCO algorithm.

We use the same values of the parameters such as time horizon, time step, etc. However, without a decent reference trajectory this approach (PILCO) that is adopted from Deisenroth and Rasmussen (2011) cannot find the optimal trajectory that drives the system to the terminal state. The reason for this result is that the MPC uses a naive approach (quadratic Euclidean distance from the equilibrium point) at each iteration to estimate the cost of the terminal state. Therefore, a reference trajectory is necessary for this approach, but it may be hard to compute such a trajectory. Alternatively, DMPC does not need any reference trajectory, and like RL, calculates a cost-to-go value for available states in the terminal set but in fewer trials than RL.

After adding a reference trajectory (Jafarzadeh & Fleming, 2018) to the cost function and training the model with 5600 training samples, PILCO could solve the problem, whereas DMPC needs less than 2900 data samples, half the running time, and no reference trajectory. Another downside of using GP is that, even if the system has linear dynamics, adding such an estimation of $\hat{z}$ to the cost function will make the model non-convex. Such a result is not desirable in terms of running time and solution quality. Applying DMPC in this context results in a MILP model, which can be solved efficiently using off-the-shelf solvers such as CPLEX, Gurobi, etc.

5. Conclusions

In this work, a Data-and Model-driven Predictive Control (DMPC) algorithm is presented to solve a model predictive control problem in which there is a function in the performance index or constraints that (a) is unknown to the controller and (b) is interdependent with the decision variables (state and control vector) of the MPC. The controller is designed to exploit an existing, exogenous data-driven system such as a black-box deep learning model, along with model predictive control to find the optimal sequence of control inputs. To solve this problem, a controller is developed that conceptually borrows from iterative learning controller but is intended for non-iterative or nonrepetitive tasks. The algorithm starts from an initial arbitrary trajectory and it is proven that the algorithm will find a feasible trajectory in

each subsequent iteration, and the trajectory at each iteration is guaranteed to be no worse than the previous iteration. DMPC is effective with very little data and converges in only a few iterations. We provided an infinite time horizon optimal control example, in which the controller should drive a nonlinear system from an initial state to an equilibrium point where the environment is an uneven surface with an unknown non-convex shape.

Acknowledgments

This work was partially supported by the NSF, USA under grant CPS-1739333.

References

Ames, Aaron D, Xu, Xiangru, Grizzle, Jessy W, & Tabuada, Paulo (2016). Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, *62*(8), 3861–3876.

Bacci, Edoardo, & Parker, David (2020). Probabilistic guarantees for safe deep reinforcement learning. arXiv preprint arXiv:2005.07073.

Behl, Madhur, Smarra, Francesco, & Mangharam, Rahul (2016). DR-advisor: A data-driven demand response recommender system. *Applied Energy*, *170*, 30–46.

Deisenroth, Marc Peter, Fox, Dieter, & Rasmussen, Carl Edward (2013). Gaussian processes for data-efficient learning in robotics and control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(2), 408–423.

Deisenroth, Marc, & Rasmussen, Carl E. (2011). PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th international conference on machine learning* (pp. 465–472).

Firoozi, Roya, Nazari, Shima, Guanetti, Jacopo, O'Gorman, Ryan, & Borrelli, Francesco (2018). Safe adaptive cruise control with road grade preview and V2V communication. arXiv preprint arXiv:1810.09000.

Houska, B., Ferreau, H. J., & Diehl, M. (2011). An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range. *Automatica*, *47*(10), 2279–2285. http://dx.doi.org/10.1016/j.automatica.2011.08.020.

Isele, David, Nakhaei, Alireza, & Fujimura, Kikuo (2018). Safe reinforcement learning on autonomous vehicles. In *2018 IEEE/RSJ international conference on intelligent robots and systems* (pp. 1–6). IEEE.

Jafarzadeh, Hassan, & Fleming, Cody H. (2018). An exact geometry–based algorithm for path planning. *International Journal of Applied Mathematics and Computer Science*, *28*(3), 493–504.

Jafarzadeh, H., & Fleming, C. (2019). Learning model predictive control for connected autonomous vehicles. In *2019 IEEE 58th conference on decision and control* (pp. 2336–2343).

Kamthe, Sanket, & Deisenroth, Marc Peter (2017). Data-efficient reinforcement learning with probabilistic model predictive control. arXiv preprint arXiv:1706.06491.

Kamthe, Sanket, & Deisenroth, Marc (2018). Data-efficient reinforcement learning with probabilistic model predictive control. In *International conference on artificial intelligence and statistics* (pp. 1701–1710). PMLR.

Kong, Jason, Pfeiffer, Mark, Schildbach, Georg, & Borrelli, Francesco (2015). Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE intelligent vehicles symposium (IV)* (pp. 1094–1099). IEEE.

Liu, Wei, & Shoji, Yozo (2019). DeepVM: RNN-based vehicle mobility prediction to support intelligent vehicle applications. *IEEE Transactions on Industrial Informatics*, *16*(6), 3997–4006.

Manh, Huynh, & Alaghband, Gita (2018). Scene-lstm: A model for human trajectory prediction. arXiv preprint arXiv:1808.04018.

Matni, Nikolai, Proutiere, Alexandre, Rantzer, Anders, & Tu, Stephen (2019). From self-tuning regulators to reinforcement learning and back again. In *2019 IEEE 58th conference on decision and control* (pp. 3724–3740). IEEE.

Moerland, Thomas M., Broekens, Joost, & Jonker, Catholijn M. (2020). Model-based reinforcement learning: A survey. arXiv preprint arXiv:2006.16712.

Rosolia, Ugo, & Borrelli, Francesco (2018). Learning model predictive control for iterative tasks. A data-driven control framework. *IEEE Transactions on Automatic Control*, *63*(7), 1883–1896.

Rosolia, Ugo, Zhang, Xiaojing, & Borrelli, Francesco (2017). Robust learning model predictive control for iterative tasks: Learning from experience. In *2017 IEEE 56th annual conference on decision and control* (pp. 1157–1162). IEEE.

Sarkar, Tuhin, Rakhlin, Alexander, & Dahleh, Munther A. (2019). Finite-time system identification for partially observed lti systems of unknown order. arXiv preprint arXiv:1902.01848.

Soltanaghaei, Elahe, Elnaggar, Mahmoud, Kleeman, Katie, Whitehouse, Kamin, & Fleming, Cody (2019). Characterizing uncertainties of wireless channels in connected vehicles. In *The 25th annual international conference on mobile computing and networking* (pp. 1–3).

Taylor, Andrew, Singletary, Andrew, Yue, Yisong, & Ames, Aaron (2020). Learning for safety-critical control with control barrier functions. In *Learning for dynamics and control* (pp. 708–717). PMLR.

Wang, Youqing, Gao, Furong, & Doyle III, Francis J. (2009). Survey on iterative learning control, repetitive control, and run-to-run control. *Journal of Process Control*, *19*(10), 1589–1600.

Hassan Jafarzadeh received his BS and M.Sc., both in industrial engineering, from Tabriz University and K.N. Toosi University of Technology, respectively. He is currently a Ph.D. candidate with the Systems and Information Engineering Department at the University of Virginia. His research interests include model predictive control and its applications to advanced automotive control, Optimization and Machine Learning.

Cody Fleming is on the faculty of Mechanical Engineering at Iowa State University, and was previously an assistant professor of Systems Engineering and Aerospace Engineering at the University of Virginia. He received his Ph.D. in Aeronautics and Astronautics at MIT. He is interested in modeling and analysis of complex systems, particularly those with high levels of automation. He has investigated several next-generation air traffic management initiatives, as well as safety assurance and algorithm development for driverless vehicles. Related research interests lie in modern feedback control, dynamics, and modeling, as well as model-based systems engineering and system assurance.