COMPUTATIONALLY EFFICIENT NONLINEAR OPTIMAL CONTROL USING NEIGHBORING EXTREMAL ADAPTATIONS

By

Amin Vahidi-Moghaddam

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Mechanical Engineering – Doctor of Philosophy

2025

ABSTRACT

Nonlinear optimal control schemes have achieved remarkable performance in numerous engineering applications; however, they typically require high computational time, which has limited their use in real-world systems with fast dynamics and/or limited computation power. To address this challenge, neighboring extremal (NE) has been developed as an efficient optimal adaption strategy to adapt a pre-computed nominal control solution to perturbations from the nominal trajectory. The resulting control law is a time-varying feedback gain that can be pre-computed along with the original optimization problem, which makes negligible online computation. This thesis focuses on reducing the computational time of the nonlinear optimal control problems using the NE in two parts. In Part I, we tackle model-based nonlinear optimal control and propose an extended neighboring extremal (ENE) to handle model uncertainties and reduce computational time (Chapter 3). Nonlinear Model predictive control (NMPC), which explicitly deals with system constraints, is considered as the case study due to its popularity, but ENE can be easily extended to other model-based nonlinear optimal control schemes. In Part II, we address data-driven nonlinear optimal control and introduce a data-enabled neighboring extremal (DeeNE) to remove parametric model requirement and reduce the computational time (Chapter 4). Data-enabled predictive control (DeePC), which makes a transition from the model-based optimal control to a data-driven one using raw input/output (I/O) data, is considered as the case study due to the attention it has received, but DeeNE can be easily extended to other data-driven nonlinear optimal control approaches. We also compare the control performance of DeeNE and DeePC for KINOVA Gen3 (7-DoF Arm Robot). Moreover, we introduce an adaptive DeePC framework, which can be easily transformed into an adaptive DeeNE, to use real-time informative data and handle time-varying systems (Chapter 5). Finally, we conclude the thesis and discuss the future works in Conclusion (Chapter 6).

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Constraint-aware optimal control schemes can explicitly handle system constraints while achieving optimal closed-loop performance [1, 2]. However, such controllers typically involve solving an optimization problem at each time step and are thus computationally expensive, especially for nonlinear systems. This has hindered their wider adoption in applications with fast dynamics and/or limited computation resources [3]. As such, several frameworks have been developed to improve computational efficiency of nonlinear optimal controllers. One approach is to simplify the system dynamics with model-reduction techniques [4, 5]. However, these techniques require a trade-off between system performance and computational complexity, and it is often still computationally expensive after the model reduction. Another approach is to use function approximators, where functions such as neural networks [6, 7], Gaussian process regression [8, 9], and spatial temporal filters [10, 11] are exploited to learn the optimal control policy, after which the learned policy is employed online to achieve efficient onboard computations. However, extensive data collection is required to ensure a comprehensive coverage of operating conditions. Another sound approach is to use cloud computing for moving on-board computations from the plant to a cloud, which employs computer system resources to provide on-demand computing power and data storage services to users [12]. However, the cloud computing has given rise to a set of new challenges related to request-response communication delays between the plant and the cloud [13]. Neighboring extremal (NE) [14, 15] is another promising paradigm to attain efficient computations by proposing a time-varying feedback gain on the state deviations. Despite promising performance, the NE rely on accurate parametric representations of real systems, but this can be challenging for complex systems. With this challenge in mind, this thesis focuses on three main objectives:

(i) Developing an extended NE (ENE) for model-based optimal control under model uncertainties;

(ii) Developing a data-enabled NE (DeeNE) for data-driven optimal control;

(iii) Developing an adaptive data-enabled optimal control for time-varying systems.

1.1 Background

For decades, the design of autonomous systems has critically relied on mathematical models describing how these systems behave. Models allow scientists and engineers to make predictions about the system's behavior and plan future decisions. Model-based optimal control tries to attain peak performance while guaranteeing system safety, i.e., ensuring that control actions respect physical limits and safety considerations. As a promising model-based optimal control framework, model predictive control (MPC) arises to accomplish this task by solving a constrained optimization problem with future state predictions [16, 17, 18]. However, obtaining the required mathematical model is often very time consuming and expensive for nonlinear and complex systems [19]. This shortcoming of the model-based optimal control motivates researches on various data-driven methods that make this controller more practically viable for the nonlinear and complex systems; however, the existing approaches cause model uncertainties, which necessitates robust optimization techniques or adaptive strategies to maintain reliable control performance.

As systems and data become increasingly complex and more widely available, respectively, scientists and practitioners are turning to data-driven methods instead of model-based techniques [20]. While the model-based optimal controllers rely on plant modeling, data-driven optimal control involves synthesizing a controller from input/output (I/O) data collected on the real system [21, 22]. There are two paradigms of the data-driven control: i) indirect data-driven control that first identifies a model using the I/O data and then conducts control design based on the identified model [10], and ii) direct data-driven control that circumvents the step of system identification and obtains control policy directly from the I/O data [23]. A central promise is that the direct data-driven control may have higher flexibility and better performance than the indirect data-driven control thanks to the data-centric representation that avoids using a specific model from identification [24]. Moreover, it is generally difficult to map uncertainty specifications from system identification over to robust control in the indirect data-driven control, while, this may become easier in the direct data-driven control.

Recently, a result in the context of behavioral system theory [25], known as Fundamental Lemma [26], has received renewed attention in the direct data-driven control. Rather than attempting to learn a parametric system model, this result enables us to learn the system's behaviour such that the subspace of the I/O trajectories of a linear time invariant (LTI) system can be obtained from the column span of a raw data Hankel matrix of time series trajectories. A direct data-driven optimal control, called data-enabled predictive control (DeePC) [27], has recently been proposed in the spirit of the Fundamental Lemma. The DeePC algorithm relies only on the raw I/O data to develop a non-parametric predictive model, learn the behavior of the unknown system, and perform safe and optimal control policies to drive the system along a desired trajectory using real-time feedback. In comparison with the machine learning-based controllers, the DeePC is more computationally efficient, less data hungry, and more suitable to rigorous stability and robustness analysis [28]. The DeePC algorithm has been successfully applied in many scenarios, including quadcopters [29] and power systems [30].

The NE [14] is a promising paradigm to attain (sub-)optimal performance with efficient computations suitable for the systems with fast dynamics and limited onboard computations. Specifically, given a pre-computed nominal solution, the NE provides an optimal correction law (to the first order) to the deviations from the nominal trajectory. The nominal control sequence can be obtained from a remote powerful controller (e.g., a cloud) or can be computed ahead of time based on an approximated initial state. The resulting NE control law is a time-varying feedback gain on the state deviations which is pre-computed along with the original optimal control problem. Therefore, this adaptation makes negligible online computation and is used towards nonlinear optimal control problems that are computationally too expensive. The NE has been employed in several engineering systems, including ship maneuvering control [31], power management [32], full bridge DC/DC converter [33], and spacecraft relative motion maneuvers [34]. However, the NE framework does not deal with the model uncertainties and the data-driven controllers (e.g. DeePC).

1.2 Thesis Outline and Contributions

Chapter 2 reviews the model-based nonlinear optimal control, the data-driven nonlinear optimal control, the DeePC framework, and the NE framework. Chapter 3 studies a robustification of the NE, called ENE, against the model uncertainties for the model-based nonlinear optimal control such that the model uncertainties are approximated and incorporated into the optimization problem as preview information. Chapter 4 studies a data-driven NE, called DeeNE, for the data-driven nonlinear optimal control using the DeePC framework such that the required parametric model is removed using Fundamental Lemma. Chapter 5 studies an adaptive DeePC strategy, which can be easily transformed into an adaptive DeeNE, for time-varying systems such that data matrix is updated using real-time informative data. The conclusions and the future works are provided in Chapter 6. Below are the detailed contributions of the main chapters.

Chapter 3: We study the problem of optimal trajectory tracking for the nonlinear systems with the model uncertainties. In modern applications, the optimal controllers frequently incorporate the model uncertainties as the preview information (e.g., using a preview prediction model [35]) while the actual model uncertainties are measured or approximated online. For the NE's control law, a time-varying feedback gain is pre-computed along with the original model-based nonlinear optimal control problem. However, the NE framework only deal with the state perturbations; therefore, the ENE is developed to consider the preview deviations in the NE adaptations. The derived ENE law is two time-varying feedback gains on the state perturbations and the preview perturbations.

Chapter 4: We study the problem of data-driven optimal trajectory tracking for the nonlinear systems with a non-parametric model. Given an initial I/O trajectory and a desired reference trajectory, the DeePC predicts the behavior of the real system and provides an optimal control sequence using raw I/O data; however, this approach has shown high computational cost due to dimension of decision variable. Several approaches have been proposed to reduce the computational cost of the DeePC for linear time-invariant (LTI) systems. However, finding a computationally efficient method to implement the DeePC on the nonlinear systems is still an open challenge. We

propose the DeeNE to approximate the DeePC policy and reduce its computational cost for the constrained nonlinear systems. The DeeNE adapts a pre-computed nominal DeePC solution to the perturbations of the initial I/O trajectory and the reference trajectory from the nominal ones.

Chapter 5: We study the problem of data-driven optimal control for the time-varying systems. DeePC uses pre-collected input/output (I/O) data to construct a data matrix for online predictive control. However, in systems with evolving dynamics, incorporating real-time data into the DeePC framework becomes crucial to enhance control performance. We propose an adaptive DeePC framework for the time-varying systems, which enables the algorithm to update the data matrix online by using real-time informative data. By exploiting the minimum non-zero singular value of the data matrix, the developed adaptive DeePC selectively integrates informative data and effectively captures evolving system dynamics. Additionally, a numerical singular value decomposition technique is introduced to reduce the computational complexity for updating a reduced-order data matrix.

CHAPTER 2

PRELIMINARIES

In this chapter, we review preliminaries of model-based nonlinear optimal control, data-driven nonlinear optimal control, data-enabled predictive control (DeePC), and neighboring extremal (NE), to provide contexts for later chapters.

2.1 Model-based Nonlinear Optimal Control

Consider the following discrete-time nonlinear system as:

$$\begin{aligned} x(k+1) &= f(x(k), u(k)), \\ y(k) &= h(x(k), u(k)), \end{aligned} \tag{2.1}$$

where $k \in \mathbb{N}^+$ represents the time step, $x \in \mathbb{R}^n$ denotes the state vector of the system, $u \in \mathbb{R}^m$ is the control input, and $y \in \mathbb{R}^p$ denotes the output of the system. Moreover, $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is the system dynamics with $f(0,0) = 0$, and $h : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^p$ represents the output dynamics.

Now, consider the following safety constraints for the system (2.1):

$$C(y(k), u(k)) \leq 0, \tag{2.2}$$

where $C : \mathbb{R}^p \times \mathbb{R}^m \to \mathbb{R}^l$.

Definition 1 (Closed-Loop Performance) *Consider the nonlinear system* (2.1) *and a control problem of tracking a desired time-varying reference r by the output of the system y. Starting from the initial state x_0, the closed-loop system performance over N steps is characterized by the following cost function:*

$$J_N(\mathbf{y}, \mathbf{u}, \mathbf{r}) = \sum_{k=0}^{N} \phi(y(k), u(k), r(k)), \tag{2.3}$$

where $\mathbf{y} = [y(0),\ y(1),\ \cdots,\ y(N)]$, $\mathbf{u} = [u(0),\ u(1),\ \cdots,\ u(N)]$, and $\phi(y, u, r)$ denotes stage cost.

With the defined closed-loop performance metric, the control goal is to minimize the cost function (2.3) while adhering to the constraints in (2.1)-(2.2). The optimal control aims at optimizing

the system performance over N future steps for the system (2.1), which is expressed as the following constrained nonlinear optimization problem:

$$\begin{aligned}
(\mathbf{y}^*, \mathbf{u}^*) &= \arg\min_{\mathbf{y},\mathbf{u}} J_N(\mathbf{y}, \mathbf{u}, \mathbf{r}) \\
\text{s.t.} \quad & x(k+1) = f(x(k), u(k)), \\
& y(k) = h(x(k), u(k)), \\
& C(y(k), u(k)) \leq 0, \\
& x(0) = x_0.
\end{aligned} \tag{2.4}$$

where the optimal control sequence $(\mathbf{y}^*(0:N), \mathbf{u}^*(0:N))$ is the solution of the above model-based nonlinear optimal control.

2.2 Data-Driven Nonlinear Optimal Control

In practice, the real nonlinear system (2.1) may not be available; thus, system identification algorithms are typically used to identify the system model. We denote the identified model as:

$$\begin{aligned}
\hat{x}(k+1) &= \hat{f}(\hat{x}(k), u(k)), \\
\hat{y}(k) &= h(\hat{x}(k), u(k)),
\end{aligned} \tag{2.5}$$

where $\hat{x}$, $\hat{y}$, and $\hat{f}$ denote the states, the outputs, and the dynamics of the identified model, respectively. It is worth noting that $\hat{f}$ is identified using the system identification algorithms by collecting sufficient data samples from the real system (2.1).

Now, using the nominal model (2.5), the data-driven nonlinear optimal control is presented as follows:

$$\begin{aligned}
(\hat{\mathbf{y}}^*, \mathbf{u}^*) &= \arg\min_{\hat{\mathbf{y}},\mathbf{u}} J_N(\hat{\mathbf{y}}, \mathbf{u}, \mathbf{r}) \\
\text{s.t.} \quad & \hat{x}(k+1) = \hat{f}(\hat{x}(k), u(k)), \\
& \hat{y}(k) = h(\hat{x}(k), u(k)), \\
& C(\hat{y}(k), u(k)) \leq 0, \\
& \hat{x}(0) = x_0.
\end{aligned} \tag{2.6}$$

where this control framework represents the indirect data-driven optimal control and requires the system identification process. However, the direct data-driven optimal control circumvents the step of the system identification and obtains control policy directly from collected data samples.

2.3 Data-Enabled Predictive Control

As a direct data-driven optimal control, the DeePC [27] makes a transition from model-based optimal control strategies (e.g. model predictive control (MPC)) to a data-driven one such that it seeks an optimal control policy from raw input/output (I/O) data without encoding them into a parametric model and requiring system identification prior to control deployment. Inspired by Fundamental Lemma [26], the system model (2.1) is replaced by an algebraic constraint that enables us to predict the length-N future input-output (I/O) trajectory for a given length-T_{ini} past (I/O) trajectory.

The Hankel matrices $\mathbb{H}(u^d)$ and $\mathbb{H}(y^d)$ are built from the offline collected I/O samples u^d and y^d as:

$$\mathbb{H}(u^d) = \begin{bmatrix} u_1 & u_2 & \cdots & u_{T-T_{ini}-N+1} \\ u_2 & u_3 & \cdots & u_{T-T_{ini}-N+2} \\ \vdots & \vdots & \ddots & \vdots \\ u_{T_{ini}+N} & u_{T_{ini}+N+1} & \cdots & u_T \end{bmatrix}, \tag{2.7}$$

where $\mathbb{H}(u^d) \in \mathbb{R}^{m(T_{ini}+N)\times L}$ needs to have full row rank of order $m(T_{ini}+N)+l$, where $l \leq n$ represents the observability index, to satisfy the persistency of excitation requirement, and the number of its columns is denoted as $L = T - T_{ini} - N + 1$. The Hankel matrix of outputs $\mathbb{H}(y^d) \in \mathbb{R}^{p(T_{ini}+N)\times L}$ is built in an analogous way from the collected samples y^d. Then, the Hankel matrices are partitioned in Past and Future subblocks as:

$$\begin{bmatrix} U_P \\ U_F \end{bmatrix} =: \mathbb{H}(u^d), \quad \begin{bmatrix} Y_P \\ Y_F \end{bmatrix} =: \mathbb{H}(y^d), \tag{2.8}$$

where $U_P \in \mathbb{R}^{mT_{ini}\times L}$, $U_F \in \mathbb{R}^{mN\times L}$, $Y_P \in \mathbb{R}^{pT_{ini}\times L}$, and $Y_F \in \mathbb{R}^{pN\times L}$.

Lemma 1 (Fundamental Lemma [26]) *Consider a controllable linear time-invariant (LTI) system, there is a unique $g \in \mathbb{R}^L$ such that any length-$T_{ini} + N$ trajectory of the system satisfies the following linear equation under a full row rank $\mathbb{H}(u^d)$ as:*

$$\begin{bmatrix} U_P \\ Y_P \\ U_F \\ Y_F \end{bmatrix} g = \begin{bmatrix} u_{ini} \\ y_{ini} \\ u \\ y \end{bmatrix}, \tag{2.9}$$

where U_P, Y_P, U_F, and Y_F are fixed data matrices obtained from the offline collected I/O data, (u_{ini}, y_{ini}) is a given length-T_{ini} initial trajectory, and (u, y) is a length-N future trajectory which is predicted online. □

For a given initial trajectory (u_{ini}, y_{ini}) collected from the real system (2.1), one can replace the optimization problem (2.4) with the DeePC as [27, 36]:

$$\begin{aligned} (\mathbf{y}^*, \mathbf{u}^*, {\sigma_\mathbf{y}}^*, {\sigma_\mathbf{u}}^*, \mathbf{g}^*) = & \underset{\mathbf{y},\mathbf{u},\sigma_\mathbf{y},\sigma_\mathbf{u},\mathbf{g}}{\arg\min} \quad J_N(\mathbf{y}, \mathbf{u}, \sigma_\mathbf{y}, \sigma_\mathbf{u}, \mathbf{g}, \mathbf{r}) \\ s.t. \quad & \begin{bmatrix} U_P \\ Y_P \\ U_F \\ Y_F \end{bmatrix} g = \begin{bmatrix} u_{ini} \\ y_{ini} \\ u \\ y \end{bmatrix} + \begin{bmatrix} \sigma_u \\ \sigma_y \\ 0 \\ 0 \end{bmatrix}, \\ & C(y, u) \leq 0, \end{aligned} \tag{2.10}$$

where $J_N(\mathbf{y}, \mathbf{u}, \sigma_\mathbf{y}, \sigma_\mathbf{u}, \mathbf{g}, \mathbf{r})$ is the modified cost function for the DeePC with noisy data and nonlinearities, $\sigma_u \in \mathbb{R}^{mT_{ini}}$ is an auxiliary slack variable to cover process noises, and $\sigma_y \in \mathbb{R}^{pT_{ini}}$ is an auxiliary slack variable to cover measurement noises and nonlinearities.

2.4 Neighboring Extremal Optimal Control

Solving a nonlinear optimal control at each time step causes a high computational cost for the control framework. To address this challenge, the NE [15] adapts a pre-computed nominal control

solution to perturbations from the nominal states. The resulting control law is a time-varying feedback gain that can be pre-computed along with the original optimal control problem, which takes negligible online computation.

Consider the closed-loop performance (2.3) for the control objective of regulating the state x as follows:

$$J_N(\mathbf{x}, \mathbf{u}) = \sum_{k=0}^{N-1} \phi(x(k), u(k)) + \psi(x(N)), \tag{2.11}$$

where $\mathbf{x} = [x(0), x(1), \cdots, x(N)]$, $\mathbf{u} = [u(0), u(1), \cdots, u(N-1)]$, and $\psi(x)$ denotes terminal cost.

Now, the model-based nonlinear optimal control (2.4) is expressed in the following form:

$$\begin{aligned} (\mathbf{x}^*, \mathbf{u}^*) &= \arg\min_{\mathbf{x},\mathbf{u}} J_N(\mathbf{x}, \mathbf{u}) \\ \text{s.t.} \quad & x(k+1) = f(x(k), u(k)), \\ & C(x(k), u(k)) \leq 0, \\ & x(0) = x_0. \end{aligned} \tag{2.12}$$

where the Hamiltonian function and the augmented cost function are defined for the above nonlinear optimization problem as:

$$H(k) = \phi(x(k), u(k)) + \lambda^T(k+1) f(x(k), u(k)) + \mu^T(k) C^a(x(k), u(k)), \tag{2.13}$$

$$\bar{J}_N(k) = \sum_{k=0}^{N-1} (H(k) - \lambda^T(k+1) x(k+1)) + \psi(x(N)), \tag{2.14}$$

where $C^a(x(k), u(k))$ represents the active constraints at the time step k. $\mu(k) \in \mathbb{R}^{l^a}$ is the Lagrange multiplier for the active constraints, and $\lambda(k+1) \in \mathbb{R}^n$ represents the Lagrange multiplier for the system dynamics (2.1).

Using variational analysis, the NE minimizes the second order variation of the augmented cost

function (2.14). Consequently, the NE solves the following nonlinear optimization problem as:

$$
\begin{aligned}
(\delta\mathbf{x}^*, \delta\mathbf{u}^*) &= \arg\min_{\delta\mathbf{x},\delta\mathbf{u}} \delta^2 \bar{J}_N(k) \\
s.t. \quad & \delta x(k+1) = f_x(k)\delta x(k) + f_u(k)\delta u(k), \\
& C_x^a(k)\delta x(k) + C_u^a(k)\delta u(k) = 0, \\
& \delta x(0) = \delta x_0,
\end{aligned}
\tag{2.15}
$$

where the solution of the above optimization problem is:

$$
\begin{aligned}
&\delta u(k) = K^*(k)\delta x(k), \\
&K^*(k) = -\begin{bmatrix} I & 0 \end{bmatrix} K^o(k) \begin{bmatrix} Z_{ux}(k) \\ C_x^a(k) \end{bmatrix}, \\
&K^o(k) = \begin{cases} \begin{bmatrix} Z_{uu}(k) & C_u^{aT}(k) \\ C_u^a(k) & 0 \end{bmatrix}^{-1} & if \;\; k \in \mathbb{K}^a \\ \begin{bmatrix} Z_{uu}^{-1}(k) & 0 \\ 0 & 0 \end{bmatrix} & if \;\; k \in \mathbb{K}^i \end{cases},
\end{aligned}
\tag{2.16}
$$

with

$$
\begin{aligned}
Z_{ux}(k) &= H_{ux}(k) + f_u^T(k)S(k+1)f_x(k), \\
Z_{uu}(k) &= H_{uu}(k) + f_u^T(k)S(k+1)f_u(k), \\
Z_{xx}(k) &= H_{xx}(k) + f_x^T(k)S(k+1)f_x(k), \\
Z_{xu}(k) &= H_{xu}(k) + f_x^T(k)S(k+1)f_u(k),
\end{aligned}
\tag{2.17}
$$

$$
S(k) = Z_{xx}(k) - \begin{bmatrix} Z_{xu}(k) & C_x^{aT}(k) \end{bmatrix} K^o(k) \begin{bmatrix} Z_{ux}(k) \\ C_x^a(k) \end{bmatrix}. \tag{2.18}
$$

where $\mathbb{K}^a$ and $\mathbb{K}^i$ are the sets of time steps at which the constraints are active (i.e., $C(x(k), u(k)) = 0$) and inactive (i.e., $C(x(k), u(k)) < 0$), respectively.

CHAPTER 3

EXTENDED NEIGHBORING EXTREMAL OPTIMAL CONTROL

In this chapter, we focus on incorporating model uncertainties into Neighboring Extremal (NE). The NE framework only deal with state perturbations while we can incorporate the model uncertainties as preview information (e.g., using a preview prediction model), where the actual model uncertainties are measured or estimated online. We develop an extended NE (ENE) framework to consider the preview deviations in the NE adaptations such that the control policy is two time-varying feedback gains on the state and preview perturbations. We also develop a constraint activity-based criteria for the ENE framework to handle large perturbations. Promising simulation results on cart inverted pendulum problem demonstrate the efficacy of the ENE algorithm.[1]

3.1 Background

Due to the vast success in optimal control and advancement in sensing, modern control applications can incorporate the model uncertainties into the control design as preview information. For example, the road profile preview obtained from vehicle crowdsourcing is exploited for simultaneous suspension control and energy harvesting, demonstrating a significant performance enhancement using the preview information despite noises in the preview [39]. Another example is thermal management for cabin and battery of hybrid electric vehicles, where traffic preview is employed in hierarchical model predictive control to improve energy efficiency [40]. Moreover, light detection and ranging systems are used to provide wind disturbance preview to enhance the controls of turbine blades in [41]. We develop an extended neighboring extremal (ENE) framework that can adapt a nominal control law to state and preview perturbations simultaneously. This setup is applicable when a nominal preview is available, and the preview signal is estimated online.

[1] The material of this chapter is from "Extended Neighboring Extremal Optimal Control with State and Preview Perturbations," IEEE Transactions on Automation Science and Engineering, 2023 [37] and "Event-Triggered Cloud-based Nonlinear Model Predictive Control with Neighboring Extremal Adaptations," IEEE 61st Conference on Decision and Control, 2022 [38].

Neighboring extremal (NE) [42, 43, 44, 45] is a promising paradigm to attain efficient computations by proposing a time-varying feedback gain on the state deviations. Specifically, given a pre-computed nominal solution based on a nominal initial state, the NE yields a control policy (to the first order) that adapts the nominal control to deviations from the nominal state. The nominal solution can be computed offline and stored online, can be performed on a remote powerful controller (e.g., cloud), or computed ahead of time by utilizing the idling time of the processor. The NE framework has been employed in several engineering systems, including ship maneuvering control [31], power management [32], full bridge DC/DC converter [33], and spacecraft relative motion maneuvers [34]. Using a parameter estimation for the unknown systems, parameter perturbations are considered in the NE, where the estimated parameters are considered constant during the predictions of the optimal control problem [46]. In [14], disturbance perturbations have been considered for the NE in the nonlinear optimal control problems; however, the formulation derived is limited to a constant disturbance.

In this chapter, we develop the ENE framework for the nonlinear optimal control problems to adapt a pre-computed nominal solution to both state perturbation and preview perturbation. This is a generalization of the NE framework [15] where only considers the state perturbation. Moreover, we treat the ENE problem when nominal non-optimal solution and large perturbations are appeared, and a multi-segment strategy is employed to guarantee constraint satisfaction in the presence of large perturbations. Furthermore, promising results are demonstrated by applying the developed control strategy to the cart inverted pendulum problem. Compared to Chapter 3, we incorporate the model uncertainties into the NE framework so that we do not need to return the NMPC at several time steps to handle the model uncertainties, which significantly reduces the computational cost. This chapter is outlined as: Section II describes the problem formulation and the preliminaries of the nonlinear optimal control problems. The proposed ENE framework is presented in Section III. Simulation on the cart-inverted pendulum is presented in Section IV. Finally, Section V discusses conclusions.

3.2 Problem Formulation

In this section, preliminaries on nonlinear optimal control problems are reviewed, and perturbation analysis problem on the optimal solution is presented for the nonlinear systems with state and preview perturbations. Specifically, the following discrete-time nonlinear system, that incorporates a system preview, is considered as:

$$x(k+1) = f(x(k), u(k), w(k)), \tag{3.1}$$

where $k \in \mathbb{N}^+$ represents the time step, $x \in \mathbb{R}^n$ denotes the measurable/observable states, and $u \in \mathbb{R}^m$ is the control input. Here $w \in \mathbb{R}^n$ represents the preview information that can be road profile preview in suspension controls [39], wind preview for turbine controls [41], and traffic preview in vehicle power management [40]. Furthermore, $f : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^n$ represents the system dynamics with $f(0, 0, 0) = 0$. Moreover, we consider the following general nominal preview model:

$$w(k+1) = g(x(k), w(k)), \tag{3.2}$$

where $g : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ represents the nominal preview dynamics.

We consider the following safety constraints for the system:

$$C(x(k), u(k), w(k)) \leq 0, \tag{3.3}$$

where $C : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^l$.

Definition 2 (Closed-Loop Performance) *Consider the nonlinear system* (3.1) *and the control objective of regulating the state x. Starting from the initial conditions x_0 and w_0, the closed-loop system performance over N steps is characterized by the following cost function:*

$$J_N(\mathbf{x}, \mathbf{u}, \mathbf{w}) = \sum_{k=0}^{N-1} \phi(x(k), u(k), w(k)) + \psi(x(N), w(N)), \tag{3.4}$$

where $\mathbf{x} = [x(0),\, x(1),\, \cdots,\, x(N)]$, $\mathbf{u} = [u(0),\, u(1),\, \cdots,\, u(N-1)]$, $\mathbf{w} = [w(0),\, w(1),\, \cdots,\, w(N)]$, *and $\phi(x, u, w)$ and $\psi(x, w)$ denote the stage and terminal costs, respectively.*

Assumption 1 (Twice Differentiable Functions) *The functions f, g, C, ϕ, and ψ are twice continuously differentiable.*

With the defined closed-loop performance metric, the control goal is to minimize the cost function (3.4) while adhering to the constraints (3.1) and (3.3). The optimal control aims at optimizing the system performance over N future steps for the system (3.1) using the nominal preview model (3.2), which is expressed as the following constrained optimization problem:

$$\begin{aligned}
(\mathbf{x}^o, \mathbf{u}^o, \mathbf{w}^o) &= \arg\min_{\mathbf{x},\mathbf{u},\mathbf{w}} J_N(\mathbf{x}, \mathbf{u}, \mathbf{w}) \\
\text{s.t.} \quad & x(k+1) = f(x(k), u(k), w(k)), \\
& w(k+1) = g(x(k), w(k)), \\
& C(x(k), u(k), w(k)) \leq 0, \\
& x(0) = x_0, \;\; w(0) = w_0.
\end{aligned} \tag{3.5}$$

Consider a nominal trajectory $\mathbf{x}^o$, $\mathbf{u}^o$, and $\mathbf{w}^o$ obtained by solving (3.5) with $\mathbf{w}^o$ being the nominal preview. This computation can be performed on a remote powerful controller (e.g., cloud computing or edge computing) or can be computed ahead of time based on an approximated initial state. During implementation, the actual state $x(k)$ and the preview information $w(k)$ will likely deviate from the nominal trajectory. Let $\delta x(k) = x(k) - x^o(k)$ and $\delta w(k) = w(k) - w^o(k)$ denote the state perturbation and the preview perturbation, respectively. Now, to solve the nonlinear optimal control problem (3.5) for the actual values at each time step k, we seek a (sub-)optimal control policy $u^*(k) = u^o(k) + \delta u(k)$ to efficiently adapt to the perturbations of the nominal trajectory. As such, using the nominal trajectory and the perturbation analysis, we develop an extended neighboring extremal (ENE) framework to account for both state and preview perturbations through two time-varying feedback gains. Moreover, to handle large perturbations, we modify the ENE algorithm to preserve constraint satisfaction and retain optimal control performance. The details of each algorithm and their benefits for nonlinear optimal control will be presented in the next part.

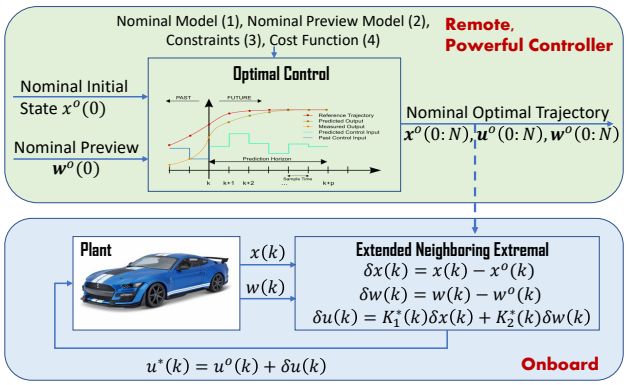

Figure 3.1: Schematic of Extended Neighboring Extremal Optimal Control with State and Preview Perturbations.

3.3 Main Result

In this section, we present the ENE framework for the optimal control problem (3.5) subject to state and preview perturbations. As shown in Fig. 3.1, a nominal trajectory is first computed based on system specifications (e.g., nominal model, nominal preview model, constraints, and cost function) along with a nominal initial state and preview. Then, the ENE exploits time-varying feedback gains to adapt to state and preview perturbations to retain optimal control performance.

3.3.1 Nominal Optimal Solution

In this subsection, we analyze the nominal optimal solution using the Karush-Kuhn-Tucker (KKT) conditions. Specifically, define $\mathbb{K}^a$ and $\mathbb{K}^i$ as the sets of time steps at which the constrains are active (i.e., $C(x(k), u(k), w(k)) = 0$ in (3.3)) and inactive (i.e., $C(x(k), u(k), w(k)) < 0$), respectively. From (3.5), the Hamiltonian function and the augmented cost function are defined as:

$$\begin{aligned} H(k) = \phi(x(k), u(k), w(k)) + \lambda^T(k+1) f(x(k), u(k), w(k)) + \bar{\lambda}^T(k+1) g(x(k), w(k)) \\ + \mu^T(k) C^a(x(k), u(k), w(k)), \end{aligned} \tag{3.6}$$

$$\bar{J}_N(k) = \sum_{k=0}^{N-1} (H(k) - \lambda^T(k+1)x(k+1) - \bar{\lambda}^T(k+1)w(k+1)) + \psi(x(N), w(N)), \tag{3.7}$$

where $C^a(x(k), u(k), w(k))$ represents the active constraints at the time step k. It is worth noting that $C^a(x(k), u(k), w(k))$ is an empty vector for inactive constraints, and $C^a(x(k), u(k), w(k)) \in$

$\mathbb{R}^{l^a}$ if we have l^a (out of l) active constraints. Furthermore, $\mu(k) \in \mathbb{R}^{l^a}$ is the Lagrange multiplier for the active constraints, and $\lambda(k+1) \in \mathbb{R}^n$ and $\bar{\lambda}(k+1) \in \mathbb{R}^n$ represent the Lagrange multipliers for the system dynamics (3.1) and the nominal preview model (3.2), respectively. It is worth noting that the Lagrange multipliers $\mu(k)$, $\lambda(k+1)$, and $\bar{\lambda}(k+1)$ are also referred as the co-states.

Assumption 2 (Active Constraints) *At each time step k, the number of active constraints is not greater than m, i.e., $C_u^a(k)$ is full row rank.*

Since $x^o(k)$, $u^o(k)$, and $w^o(k)$ ($k \in [0, N]$) represent the nominal optimal solution for the nonlinear optimal control problem (3.5), they satisfy the following KKT conditions for the augmented cost function (3.7):

$$\begin{aligned}
&H_u(k) = 0,\ k = 0, 1, ..., N-1,\\
&\lambda(k) = H_x(k),\ k = 0, 1, ..., N-1,\\
&\lambda(N) = \psi_x(x(N), w(N)),\\
&\bar{\lambda}(k) = H_w(k),\ k = 0, 1, ..., N-1,\\
&\bar{\lambda}(N) = \psi_w(x(N), w(N)),\\
&\mu(k) \geq 0,\ k = 0, 1, ..., N-1,
\end{aligned} \tag{3.8}$$

where the subscripts u, x, and w represent the partial derivatives of a function.

Now, using the KKT conditions (3.8) and the nominal solution $x^o(k)$, $u^o(k)$, and $w^o(k)$, one can calculate the Lagrange multipliers $\mu(k)$, $\lambda(k+1)$, and $\bar{\lambda}(k+1)$ online as:

$$\begin{aligned}
0 \quad &= \phi_u(x^o, u^o, w^o) + \lambda^T(k+1) f_u(x^o, u^o, w^o) + \mu^T(k) C_u^a(x^o, u^o, w^o),\\
\lambda(k) \ &= \phi_x(x^o, u^o, w^o) + \lambda^T(k+1) f_x(x^o, u^o, w^o) + \bar{\lambda}^T(k+1) g_x(x^o, w^o)\\
&\quad + \mu^T(k) C_x^a(x^o, u^o, w^o),\\
\lambda(N) &= \psi_x(x^o(N), w^o(N)),\\
\bar{\lambda}(k) \ &= \phi_w(x^o, u^o, w^o) + \lambda^T(k+1) f_w(x^o, u^o, w^o) + \bar{\lambda}^T(k+1) g_w(x^o, w^o)\\
&\quad + \mu^T(k) C_w^a(x^o, u^o, w^o),\\
\bar{\lambda}(N) &= \psi_w(x^o(N), w^o(N)).
\end{aligned} \tag{3.9}$$

Using the above equations, the Lagrange multipliers can be obtained as:

$$\begin{aligned}
\mu(k) &= -(C_u^a(k)C_u^{aT}(k))^{-1}C_u^a(k)\phi_u^T(k) - (C_u^a(k)C_u^{aT}(k))^{-1}C_u^a(k)f_u^T(k)\lambda(k+1),\\
\lambda(k) &= \phi_x(k) + \lambda^T(k+1)f_x(k) + \bar{\lambda}^T(k+1)g_x(k) + \mu^T(k)C_x^a(k),\\
\bar{\lambda}(k) &= \phi_w(k) + \lambda^T(k+1)f_w(k) + \bar{\lambda}^T(k+1)g_w(k) + \mu^T(k)C_w^a(k).
\end{aligned} \tag{3.10}$$

Note that $\delta\bar{J}_N(x^o, u^o, w^o, \mu^o, \lambda^o, \bar{\lambda}^o) = 0$, and Assumption 2 guarantees that $C_u^a(k)C_u^{aT}(k)$ is invertible.

3.3.2 Extended Neighboring Extremal

For this part, we assume that the state and preview perturbations are small enough such that they do not change the activity status of the constraint. To adapt to state and preview perturbations from the nominal values, the ENE seeks to minimize the second-order variation of (2.14)4 subject to linearized models and constraints. More specifically, the ENE algorithm solves the following optimization problem with the initial conditions $\delta x(0)$ and $\delta w(0)$ as:

$$\begin{aligned}
(\delta\mathbf{x^o}, \delta\mathbf{u^o}, \delta\mathbf{w^o}) &= \underset{\delta\mathbf{x},\delta\mathbf{u},\delta\mathbf{w}}{\arg\min}\ J_N^{ne}(k)\\
\text{s.t.}\quad & \delta x(k+1) = f_x(k)\delta x(k) + f_u(k)\delta u(k) + f_w(k)\delta w(k),\\
& \delta w(k+1) = g_x(k)\delta x(k) + g_w(k)\delta w(k),\\
& C_x^a(k)\delta x(k) + C_u^a(k)\delta u(k) + C_w^a(k)\delta w(k) = 0,\\
& \delta x(0) = \delta x_0,\ \ \delta w(0) = \delta w_0,
\end{aligned} \tag{3.11}$$

where

$$\begin{aligned}
J_N^{ne}(k) = \delta^2\bar{J}_N(k) = \frac{1}{2}\sum_{k=0}^{N-1}&\begin{bmatrix}\delta x(k)\\ \delta u(k)\\ \delta w(k)\end{bmatrix}^T\begin{bmatrix}H_{xx}(k) & H_{xu}(k) & H_{xw}(k)\\ H_{ux}(k) & H_{uu}(k) & H_{uw}(k)\\ H_{wx}(k) & H_{wu}(k) & H_{ww}(k)\end{bmatrix}\begin{bmatrix}\delta x(k)\\ \delta u(k)\\ \delta w(k)\end{bmatrix}\\
&+ \frac{1}{2}\delta x^T(N)\psi_{xx}(N)\delta x(N) + \frac{1}{2}\delta w^T(N)\psi_{ww}(N)\delta w(N)
\end{aligned} \tag{3.12}$$

For (3.11) and (3.12), the Hamiltonian function and the augmented cost function are obtained as:

$$\begin{aligned} H^{ne}(k) = & \frac{1}{2}\begin{bmatrix} \delta x(k) \\ \delta u(k) \\ \delta w(k) \end{bmatrix}^T \begin{bmatrix} H_{xx}(k) & H_{xu}(k) & H_{xw}(k) \\ H_{ux}(k) & H_{uu}(k) & H_{uw}(k) \\ H_{wx}(k) & H_{wu}(k) & H_{ww}(k) \end{bmatrix} \begin{bmatrix} \delta x(k) \\ \delta u(k) \\ \delta w(k) \end{bmatrix} \\ & + \delta\lambda^T(k+1)(f_x(k)\delta x(k) + f_u(k)\delta u(k) + f_w(k)\delta w(k)) \\ & + \delta\bar{\lambda}^T(k+1)(g_x(k)\delta x(k) + g_w(k)\delta w(k)) \\ & + \delta\mu^T(k)(C_x^a(k)\delta x(k) + C_u^a(k)\delta u(k) + C_w^a(k)\delta w(k)), \end{aligned} \tag{3.13}$$

$$\begin{aligned} \bar{J}_N^{ne}(k) = & \sum_{k=0}^{N-1}(H^{ne}(k) - \delta\lambda^T(k+1)\delta x(k+1) - \delta\bar{\lambda}^T(k+1)\delta w(k+1)) \\ & + \frac{1}{2}\delta x^T(N)\psi_{xx}(N)\delta x(N) + \frac{1}{2}\delta w^T(N)\psi_{ww}(N)\delta w(N), \end{aligned} \tag{3.14}$$

where $\delta\mu(k)$, $\delta\lambda(k)$, and $\delta\bar{\lambda}(k)$ are the Lagrange multipliers. By applying the KKT conditions to (3.14), one has

$$\begin{aligned} & H_{\delta u}^{ne}(k) = 0, && k = 0, 1, ..., N-1, \\ & \delta\lambda(k) = H_{\delta x}^{ne}(k), && k = 0, 1, ..., N-1, \\ & \delta\lambda(N) = \psi_{xx}(N)\delta x(N), \\ & \delta\bar{\lambda}(k) = H_{\delta w}^{ne}(k), && k = 0, 1, ..., N-1, \\ & \delta\bar{\lambda}(N) = \psi_{ww}(N)\delta w(N), \\ & \delta\mu(k) \geq 0, && k = 0, 1, ..., N-1. \end{aligned} \tag{3.15}$$

To facilitate the development of the ENE algorithm, several auxiliary variables $S(k)$, $W(k)$, $\bar{S}(k)$, and $\bar{W}(k)$, $k = 1, 2, \cdots, N-1$, are introduced as

$$\begin{aligned} S(k) &= Z_{xx}(k) - \left[Z_{xu}(k)\ C_x^{aT}(k)\right] K^o(k) \begin{bmatrix} Z_{ux}(k) \\ C_x^a(k) \end{bmatrix}, \\ W(k) &= Z_{xw}(k) - \left[Z_{xu}(k)\ C_x^{aT}(k)\right] K^o(k) \begin{bmatrix} Z_{uw}(k) \\ C_w^a(k) \end{bmatrix}, \end{aligned} \tag{3.16}$$

$$\bar{S}(k) = Z_{wx}(k) - \left[Z_{wu}(k)\ C_w^{aT}(k) \right] K^o(k) \begin{bmatrix} Z_{ux}(k) \\ C_x^a(k) \end{bmatrix},$$
$$\bar{W}(k) = Z_{ww}(k) - \left[Z_{wu}(k)\ C_w^{aT}(k) \right] K^o(k) \begin{bmatrix} Z_{uw}(k) \\ C_w^a(k) \end{bmatrix}, \tag{3.17}$$

where the terminal conditions for $S(k)$, $W(k)$, $\bar{S}(k)$, and $\bar{W}(k)$ are given by

$$S(N) = \psi_{xx}(N), \qquad W(N) = 0, \qquad \bar{S}(N) = 0, \qquad \bar{W}(N) = \psi_{ww}(N), \tag{3.18}$$

and

$$\begin{aligned}
Z_{ux}(k) &= H_{ux}(k) + f_u^T(k)S(k+1)f_x(k) + f_u^T(k)W(k+1)g_x(k),\\
Z_{uu}(k) &= H_{uu}(k) + f_u^T(k)S(k+1)f_u(k),\\
Z_{uw}(k) &= H_{uw}(k) + f_u^T(k)S(k+1)f_w(k) + f_u^T(k)W(k+1)g_w(k),\\
Z_{xx}(k) &= H_{xx}(k) + f_x^T(k)S(k+1)f_x(k) + f_x^T(k)W(k+1)g_x(k) + g_x^T(k)\bar{S}(k+1)f_x(k)\\
&\quad + g_x^T(k)\bar{W}(k+1)g_x(k),\\
Z_{xu}(k) &= H_{xu}(k) + f_x^T(k)S(k+1)f_u(k) + g_x^T(k)\bar{S}(k+1)f_u(k),\\
Z_{xw}(k) &= H_{xw}(k) + f_x^T(k)S(k+1)f_w(k) + f_x^T(k)W(k+1)g_w(k) + g_x^T(k)\bar{S}(k+1)f_w(k)\\
&\quad + g_x^T(k)\bar{W}(k+1)g_w(k),\\
Z_{wx}(k) &= H_{wx}(k) + f_w^T(k)S(k+1)f_x(k) + f_w^T(k)W(k+1)g_x(k) + g_w^T(k)\bar{S}(k+1)f_x(k)\\
&\quad + g_w^T(k)\bar{W}(k+1)g_x(k),\\
Z_{wu}(k) &= H_{wu}(k) + f_w^T(k)S(k+1)f_u(k) + g_w^T(k)\bar{S}(k+1)f_u(k),\\
Z_{ww}(k) &= H_{ww}(k) + f_w^T(k)S(k+1)f_w(k) + f_w^T(k)W(k+1)g_w(k) + g_w^T(k)\bar{S}(k+1)f_w(k)\\
&\quad + g_w^T(k)\bar{W}(k+1)g_w(k),
\end{aligned} \tag{3.19}$$

and

$$K^o(k) = \begin{cases} \begin{bmatrix} Z_{uu}(k) & C_u^{aT}(k) \\ C_u^a(k) & 0 \end{bmatrix}^{-1} & \text{if } k \in \mathbb{K}^a, \\ \begin{bmatrix} Z_{uu}^{-1}(k) & 0 \\ 0 & 0 \end{bmatrix} & \text{if } k \in \mathbb{K}^i. \end{cases} \tag{3.20}$$

Theorem 1 (Extended Neighboring Extremal) *Consider the optimization problem* (3.11) *and the Hamiltonian function* (3.13)*. If* $Z_{uu}(k) > 0$ *for* $k \in \Big[0, N-1\Big]$*, then the ENE policy*

$$\begin{aligned} \delta u(k) &= K_1^*(k)\delta x(k) + K_2^*(k)\delta w(k), \\ K_1^*(k) &= -\begin{bmatrix} I & 0 \end{bmatrix} K^o(k) \begin{bmatrix} Z_{ux}(k) \\ C_x^a(k) \end{bmatrix}, \\ K_2^*(k) &= -\begin{bmatrix} I & 0 \end{bmatrix} K^o(k) \begin{bmatrix} Z_{uw}(k) \\ C_w^a(k) \end{bmatrix}, \end{aligned} \tag{3.21}$$

approximates the perturbed solution for the nonlinear optimal control problem (3.5) *in the presence of state perturbation* $\delta x(k)$ *and preview perturbation* $\delta w(k)$*.*

Proof 1 *Using* (3.13)*,* (3.14)*, and the KKT conditions* (3.15)*, one has*

$$0 = H_{ux}(k)\delta x(k) + H_{uu}(k)\delta u(k) + H_{uw}(k)\delta w(k) + f_u^T(k)\delta\lambda(k+1) + C_u^{aT}(k)\delta\mu(k), \tag{3.22}$$

$$\begin{aligned} \delta\lambda(k) = {} & H_{xx}(k)\delta x(k) + H_{xu}(k)\delta u(k) + H_{xw}(k)\delta w(k) + f_x^T(k)\delta\lambda(k+1) + g_x^T(k)\delta\bar{\lambda}(k+1) \\ & + C_x^{aT}(k)\delta\mu(k), \end{aligned} \tag{3.23}$$

$$\begin{aligned} \delta\bar{\lambda}(k) = {} & H_{wx}(k)\delta x(k) + H_{wu}(k)\delta u(k) + H_{ww}(k)\delta w(k) + f_w^T(k)\delta\lambda(k+1) + g_w^T(k)\delta\bar{\lambda}(k+1) \\ & + C_w^{aT}(k)\delta\mu(k), \end{aligned} \tag{3.24}$$

where $\delta\lambda(N) = \psi_{xx}(N)\delta x(N)$ *and* $\delta\bar{\lambda}(N) = \psi_{ww}(N)\delta w(N)$. *Now, define the following general relation:*

$$\delta\lambda(k) = S(k)\delta x(k) + W(k)\delta w(k) + T(k), \tag{3.25}$$

$$\delta\bar{\lambda}(k) = \bar{S}(k)\delta x(k) + \bar{W}(k)\delta w(k) + \bar{T}(k). \tag{3.26}$$

Using (3.15), (3.25), *and* (3.26), *one has* $T(N) = 0$ *and* $\bar{T}(N) = 0$. *Substituting the linearized model* (3.11) *and* (3.25) *into* (3.22) *yields*

$$Z_{ux}(k)\delta x(k) + Z_{uu}(k)\delta u(k) + Z_{uw}(k)\delta w(k) + C_u^{aT}(k)\delta\mu(k) + f_u^T(k)T(k+1) = 0. \tag{3.27}$$

Using the linearized safety constraints (3.11) *and* (3.27), *one has*

$$\begin{aligned}\begin{bmatrix}\delta u(k)\\ \delta\mu(k)\end{bmatrix} = &-K^o(k)\begin{bmatrix}Z_{ux}(k)\\ C_x^a(k)\end{bmatrix}\delta x(k)\\ &- K^o(k)\begin{bmatrix}Z_{uw}(k)\\ C_w^a(k)\end{bmatrix}\delta w(k)\\ &- K^o(k)\begin{bmatrix}f_u^T(k)T(k+1)\\ 0\end{bmatrix}.\end{aligned} \tag{3.28}$$

Now, substituting the model (3.11), (3.25) *and* (3.26) *into* (3.23) *yields*

$$\begin{aligned}\delta\lambda(k) = &Z_{xx}(k)\delta x(k) + Z_{xu}(k)\delta u(k) + Z_{xw}(k)\delta w(k) + C_x^{aT}(k)\delta\mu(k) + f_x^T(k)T(k+1)\\ &+ g_x^T(k)\bar{T}(k+1).\end{aligned} \tag{3.29}$$

Furthermore, substituting (3.28) *into* (3.29) *yields*

$$\begin{aligned}\delta\lambda(k) = &\left(Z_{xx}(k) - \left[Z_{xu}(k)\ C_x^{aT}(k)\right]K^o(k)\begin{bmatrix}Z_{ux}(k)\\ C_x^a(k)\end{bmatrix}\right)\delta x(k)\\ &+\left(Z_{xw}(k) - \left[Z_{xu}(k)\ C_x^{aT}(k)\right]K^o(k)\begin{bmatrix}Z_{uw}(k)\\ C_w^a(k)\end{bmatrix}\right)\delta w(k)\\ &+ f_x^T(k)T(k+1) - \left[Z_{xu}(k)\ C_x^{aT}(k)\right]K^o(k)\begin{bmatrix}f_u^T(k)T(k+1)\\ 0\end{bmatrix} + g_x^T(k)\bar{T}(k+1).\end{aligned} \tag{3.30}$$

From (3.16)*,* (3.25) *and* (3.30)*, it can be concluded that*

$$T(k) = g_x^T(k)\bar{T}(k+1) + f_x^T(k)T(k+1) - \left[Z_{xu}(k)\ C_x^{aT}(k)\right] K^o(k) \begin{bmatrix} f_u^T(k)T(k+1) \\ 0 \end{bmatrix}. \quad (3.31)$$

Now, substituting the model (3.11)*,* (3.25) *and* (3.26) *into* (3.24) *yields*

$$\begin{aligned} \delta\bar{\lambda}(k) = {} & Z_{wx}(k)\delta x(k) + Z_{wu}(k)\delta u(k) + Z_{ww}(k)\delta w(k) + C_w^{aT}(k)\delta\mu(k) \\ & + f_w^T(k)T(k+1) + g_w^T(k)\bar{T}(k+1). \end{aligned} \quad (3.32)$$

Furthermore, plugging (3.28) *into* (3.32) *yields*

$$\begin{aligned} \delta\bar{\lambda}(k) = {} & \left(Z_{wx}(k) - \left[Z_{wu}(k)\ C_w^{aT}(k)\right] K^o(k) \begin{bmatrix} Z_{ux}(k) \\ C_x^a(k) \end{bmatrix} \right) \delta x(k) \\ & + \left(Z_{ww}(k) - \left[Z_{wu}(k)\ C_w^{aT}(k)\right] K^o(k) \begin{bmatrix} Z_{uw}(k) \\ C_w^a(k) \end{bmatrix} \right) \delta w(k) \\ & + f_w^T(k)T(k+1) - \left[Z_{wu}(k)\ C_w^{aT}(k)\right] K^o(k) \begin{bmatrix} f_u^T(k)T(k+1) \\ 0 \end{bmatrix} + g_w^T(k)\bar{T}(k+1). \end{aligned} \quad (3.33)$$

Using (3.17)*,* (3.26) *and* (3.33)*, one has*

$$\bar{T}(k) = g_w^T(k)\bar{T}(k+1) + f_w^T(k)T(k+1) - \left[Z_{wu}(k)\ C_w^{aT}(k)\right] K^o(k) \begin{bmatrix} f_u^T(k)T(k+1) \\ 0 \end{bmatrix}. \quad (3.34)$$

Based on (3.31)*,* (3.34)*, and the fact that* $T(N) = 0$*,* $\bar{T}(N) = 0$*, one can conclude that for* $k \in \left[1, N-1\right]$*,* $T(k) = 0$*,* $\bar{T}(k) = 0$*. Thus, by using* (3.28)*, the ENE policy* (2.16) *can be obtained. This completes the proof.* □

Remark 1 (Singularity) *It is worth noting that the assumption of* Z_{uu} *being positive definite (i.e.,* $Z_{uu}(k) > 0, k \in \left[0, N-1\right]$*) is essential for the ENE.* $Z_{uu}(k) > 0$ *is performed to calculate the ENE such that it guarantees the convexity of* (3.11)*. Considering* $Z_{uu}(k) > 0$ *and Assumption 2, it is clear that* $K^o(k)$ (3.20) *is well defined. However, when the constraints involve only state and*

preview (i.e., $C_u^a(k) = 0$), or when l^a is greater than m (i.e., $C_u^a(k)$ is not full row rank), the matrix K^o is singular, leading to the failure of the proposed algorithm. This issue can be solved using the constraint back-propagation algorithm presented in [15].

Remark 2 (Nominal Preview Model) *If we do not have any idea about the nominal preview model* (3.2) *for the existing preview information in the real system, we can simply use $w(k+1) = w(k)$ as the nominal preview model for the nonlinear optimal control problem* (3.5) *and the ENE algorithm. However, it is clear that we achieve the best performance using the ENE when the nominal preview model describes the preview information perfectly.*

Algorithm 1 summarizes the ENE procedure for adaptation the pre-computed nominal control solution $u^o(k)$ to the small state perturbation $\delta x(k)$ and the small preview perturbation $\delta w(k)$ such that it achieves the optimal control as $u^*(k) = u^o(k) + \delta u(k)$ using Theorem 1.

Algorithm 1 Extended Neighboring Extremal.

Input: The functions f, g, C, ϕ, and ψ, and the nominal trajectory $\mathbf{x}^o(0:N)$, $\mathbf{u}^o(0:N)$, and $\mathbf{w}^o(0:N)$.
1: Initialize the matrices $\lambda^o(N)$, $\bar{\lambda}^o(N)$, $S(N)$, $W(N)$, $\bar{S}(N)$, and $\bar{W}(N)$ using (3.9) and (3.18).
2: Calculate, in a backward run, the Lagrange multipliers $\mu^o(k)$, $\lambda^o(k)$, and $\bar{\lambda}^o(k)$ using (3.10).
3: Calculate, in a backward run, the matrices $Z(k)$, the gains $K_1^*(k)$ and $K_2^*(k)$, and the matrices $S(k)$, $W(k)$, $\bar{S}(k)$, and $\bar{W}(k)$ using (3.19), (3.21), (3.16), and (3.17), respectively.
4: Given $x^o(0)$, $w^o(0)$, $\delta x(0)$, and $\delta w(0)$, in a forward run, calculate $\delta u(k)$, $u^*(k)$, $x(k+1)$, and $w(k+1)$ using (3.21) and (3.1).

3.3.3 Nominal Non-Optimal Solution and Large Perturbations

The ENE is derived under the assumption that a nominal optimal solution is available, and the state and preview perturbations are small such that they do not change the activity status of the constraints. In this subsection, we modify the ENE policy for a nominal non-optimal solution and accordingly improve the algorithm to handle large state and preview perturbations which may change the sets of inactive and active constraints.

For the nominal non-optimal sequences $x^o(k)$, $u^o(k)$, $w^o(k)$, $\mu^o(k)$, $\lambda^o(k)$, and $\bar{\lambda}^o(k)$, we assume that they satisfy the constraints described in (3.5) and (3.8) but may not satisfy the

optimality condition $H_u(x^o, u^o, w^o, \mu^o, \lambda^o, \bar{\lambda}^o) = 0$. Under this circumstance, the cost function (3.12) is modified as

$$\begin{aligned} J_N^{ne}(k) = \delta^2 \bar{J}_N(k) + \sum_{k=0}^{N-1} H_u^T(k)\delta u(k) = \frac{1}{2}\sum_{k=0}^{N-1} \begin{bmatrix} \delta x(k) \\ \delta u(k) \\ \delta w(k) \end{bmatrix}^T \begin{bmatrix} H_{xx}(k) & H_{xu}(k) & H_{xw}(k) \\ H_{ux}(k) & H_{uu}(k) & H_{uw}(k) \\ H_{wx}(k) & H_{wu}(k) & H_{ww}(k) \end{bmatrix} \begin{bmatrix} \delta x(k) \\ \delta u(k) \\ \delta w(k) \end{bmatrix} \\ + \frac{1}{2}\delta x^T(N)\psi_{xx}(N)\delta x(N) + \frac{1}{2}\delta w^T(N)\psi_{ww}(N)\delta w(N) \\ + \sum_{k=0}^{N-1} H_u^T(k)\delta u(k). \end{aligned} \tag{3.35}$$

Considering the optimal control problem (3.11) and the cost function (3.35), the Hamiltonian function is modified as

$$\begin{aligned} H^{ne}(k) = \frac{1}{2} \begin{bmatrix} \delta x(k) \\ \delta u(k) \\ \delta w(k) \end{bmatrix}^T \begin{bmatrix} H_{xx}(k) & H_{xu}(k) & H_{xw}(k) \\ H_{ux}(k) & H_{uu}(k) & H_{uw}(k) \\ H_{wx}(k) & H_{wu}(k) & H_{ww}(k) \end{bmatrix} \begin{bmatrix} \delta x(k) \\ \delta u(k) \\ \delta w(k) \end{bmatrix} \\ + \delta\lambda^T(k+1)(f_x(k)\delta x(k) + f_u(k)\delta u(k) + f_w(k)\delta w(k)) \\ + \delta\bar{\lambda}^T(k+1)(g_x(k)\delta x(k) + g_w(k)\delta w(k)) \\ + \delta\mu^T(k)(C_x^a(k)\delta x(k) + C_u^a(k)\delta u(k) + C_w^a(k)\delta w(k)) + H_u^T(k)\delta u(k). \end{aligned} \tag{3.36}$$

Now, the following theorem is presented to modify the ENE policy for the nominal non-optimal solutions to the nonlinear optimal control problem (3.5).

Theorem 2 (Modified Extended Neighboring Extremal) *Consider the optimization problem* (3.11)*, the KKT conditions* (3.15)*, and the Hamiltonian function* (3.36)*. If* $Z_{uu}(k) > 0$ *for* $k \in \Big[0, N-1\Big]$*, then the ENE policy for a nominal non-optimal solution is modified as*

$$\begin{aligned} \delta u(k) &= K_1^*(k)\delta x(k) + K_2^*(k)\delta w(k) + K_3^*(k) \begin{bmatrix} f_u^T(k)T(k+1) + H_u(k) \\ 0 \end{bmatrix}, \\ K_3^*(k) &= -\Big[I \quad 0\Big] K^o(k), \end{aligned} \tag{3.37}$$

where the gain matrices K_1^*, K_2^*, and K^o are defined in (3.20) *and* (3.21)*, and $T(k)$ is a non-zero variable defined in* (3.42)*.*

Proof 2 *Using* (3.15) *and* (3.36)*,* (3.22) *is modified as*

$$H_{ux}(k)\delta x(k) + H_{uu}(k)\delta u(k) + H_{uw}(k)\delta w(k) + f_u^T(k)\delta\lambda(k+1) + C_u^{aT}(k)\delta\mu(k) + H_u(k) = 0. \tag{3.38}$$

Substituting the linearized model (3.11) *and* (3.25) *into* (3.38) *yields*

$$Z_{ux}(k)\delta x(k) + Z_{uu}(k)\delta u(k) + Z_{uw}(k)\delta w(k) + C_u^{aT}(k)\delta\mu(k) + f_u^T(k)T(k+1) + H_u(k) = 0. \tag{3.39}$$

Using the linearized safety constraints (3.11) *and* (3.39)*, one can obtain*

$$\begin{aligned}\begin{bmatrix}\delta u(k)\\ \delta\mu(k)\end{bmatrix} = &-K^o(k)\begin{bmatrix}Z_{ux}(k)\\ C_x^a(k)\end{bmatrix}\delta x(k) - K^o(k)\begin{bmatrix}Z_{uw}(k)\\ C_w^a(k)\end{bmatrix}\delta w(k)\\ &- K^o(k)\begin{bmatrix}f_u^T(k)T(k+1)\\ 0\end{bmatrix} - K^o(k)\begin{bmatrix}H_u(k)\\ 0\end{bmatrix}.\end{aligned} \tag{3.40}$$

Substituting (3.40) *into* (3.29) *yields*

$$\begin{aligned}\delta\lambda(k) = &\left(Z_{xx}(k) - \begin{bmatrix}Z_{xu}(k)\ C_x^{aT}(k)\end{bmatrix}K^o(k)\begin{bmatrix}Z_{ux}(k)\\ C_x^a(k)\end{bmatrix}\right)\delta x(k)\\ &+\left(Z_{xw}(k) - \begin{bmatrix}Z_{xu}(k)\ C_x^{aT}(k)\end{bmatrix}K^o(k)\begin{bmatrix}Z_{uw}(k)\\ C_w^a(k)\end{bmatrix}\right)\delta w(k)\\ &+ f_x^T(k)T(k+1) - \begin{bmatrix}Z_{xu}(k)\ C_x^{aT}(k)\end{bmatrix}K^o(k)\begin{bmatrix}f_u^T(k)T(k+1)\\ 0\end{bmatrix}\\ &+ g_x^T(k)\bar{T}(k+1) - \begin{bmatrix}Z_{xu}(k)\ C_x^{aT}(k)\end{bmatrix}K^o(k)\begin{bmatrix}H_u(k)\\ 0\end{bmatrix}.\end{aligned} \tag{3.41}$$

From (3.16)*,* (3.25)*, and* (3.41)*, it follows that*

$$T(k) = g_x^T(k)\bar{T}(k+1) + f_x^T(k)T(k+1) - \left[Z_{xu}(k)\ C_x^{aT}(k)\right] K^o(k) \begin{bmatrix} f_u^T(k)T(k+1) + H_u(k) \\ 0 \end{bmatrix}. \tag{3.42}$$

Now, plugging (3.40) *into* (3.32) *yields*

$$\begin{aligned} \delta\bar{\lambda}(k) = &\left(Z_{wx}(k) - \left[Z_{wu}(k)\ C_w^{aT}(k)\right] K^o(k) \begin{bmatrix} Z_{ux}(k) \\ C_x^a(k) \end{bmatrix} \right) \delta x(k) \\ &+ \left(Z_{ww}(k) - \left[Z_{wu}(k)\ C_w^{aT}(k)\right] K^o(k) \begin{bmatrix} Z_{uw}(k) \\ C_w^a(k) \end{bmatrix} \right) \delta w(k) \\ &+ f_w^T(k)T(k+1) - \left[Z_{wu}(k)\ C_w^{aT}(k)\right] K^o(k) \begin{bmatrix} f_u^T(k)T(k+1) \\ 0 \end{bmatrix} \\ &+ g_w^T(k)\bar{T}(k+1) - \left[Z_{wu}(k)\ C_w^{aT}(k)\right] K^o(k) \begin{bmatrix} H_u(k) \\ 0 \end{bmatrix}. \end{aligned} \tag{3.43}$$

Using (3.17)*,* (3.26)*, and* (3.43)*, one has*

$$\bar{T}(k) = g_w^T(k)\bar{T}(k+1) + f_w^T(k)T(k+1) - \left[Z_{wu}(k)\ C_w^{aT}(k)\right] K^o(k) \begin{bmatrix} f_u^T(k)T(k+1) + H_u(k) \\ 0 \end{bmatrix}. \tag{3.44}$$

Based on (3.40)*,* (3.42) *and* (3.44)*, the modified ENE policy* (3.37) *is obtained. This completes the proof.* □

Now, using (3.40), the relation between the state and preview perturbations and the Lagrange multiplier perturbation is expressed as:

$$\begin{aligned} &\delta\mu(k) = K_4^*(k)\delta x(k) + K_5^*(k)\delta w(k), \\ &K_4^*(k) = -\begin{bmatrix} 0 & I \end{bmatrix} K^o(k) \begin{bmatrix} Z_{ux}(k) \\ C_x^a(k) \end{bmatrix}, \\ &K_5^*(k) = -\begin{bmatrix} 0 & I \end{bmatrix} K^o(k) \begin{bmatrix} Z_{uw}(k) \\ C_w^a(k) \end{bmatrix}. \end{aligned} \tag{3.45}$$

Moreover, using (3.37), the constraint perturbation is represented as

$$\begin{aligned}\delta C(k) &= C_x(k)\delta x(k) + C_u(k)\delta u(k) + C_w(k)\delta w(k)\\ &= (C_x(k) + C_u(k)K_1^*(k))\delta x(k) + (C_w(k) + C_u(k)K_2^*(k))\delta w(k)\\ &\quad + C_u(k)K_3^*(k)(f_u^T(k)T(k+1) + H_u(k)).\end{aligned} \tag{3.46}$$

The perturbed Lagrange multiplier and the perturbed constraint are given by

$$\mu(k) = \mu^o(k) + \delta\mu(k), \tag{3.47}$$

$$C(k) = C^o(k) + \delta C(k). \tag{3.48}$$

Different activity statuses of the constraints may occur due to large perturbations. To address this issue, we consider a line that connects the nominal variables $x^o(0)$ and $w^o(0)$ to the perturbed variables $x(0)$ and $w(0)$. For the connecting line, we identify several intermediate points such that the status of the constraint remains the same between two consecutive points. Since we respectively have $\mu(k) = 0$ and $C(k) = 0$ for the inactive and active constraints, we use (3.47) for the active constraints to find the intermediate points which make the constraints inactive. Specifically, for the active constraints, an $\alpha(k)$ ($0 \leq \alpha(k) \leq 1$) is computed to have $\mu^o(k) + \alpha(k)\delta\mu(k) = 0$. Moreover, we employ (3.48) for the inactive constraints to find the intermediate points which make the constraints active. For the inactive constraints, the $\alpha(k)$ is computed to have $C^o(k) + \alpha(k)\delta C(k) = 0$. Thus, for $k \in [0, N-1]$, the intermediate points are achieved using the following equation:

$$\alpha(k) = \begin{cases} -\dfrac{\mu^o(k)}{\delta\mu(k)} & \text{if } \ k \in \mathbb{K}^a, \\ -\dfrac{C^o(k)}{\delta C(k)} & \text{if } \ k \in \mathbb{K}^i. \end{cases} \tag{3.49}$$

The smallest $\alpha(k)$ is found such that the obtained perturbation changes the activity statuses of the constraints at least at one time step k.

Algorithm 2 summarizes the modified ENE procedure for adaptation the pre-computed nominal non-optimal control solution to the large state and preview perturbations such that it achieves

the optimal control as $u^*(k) = u^o(k) + \delta u(k)$ using Theorem 2. The algorithm identifies the intermediate points and determines the modified ENE adaptation policy.

Algorithm 2 Modified Extended Neighboring Extremal.

Input: The functions f, g, C, ϕ, and ψ, and the nominal trajectory $\mathbf{x}^o(0:N)$, $\mathbf{u}^o(0:N)$, and $\mathbf{w}^o(0:N)$.
1: Set $j = 0$.
2: Initialize the matrices $\lambda^o(N)$, $\bar{\lambda}^o(N)$, $S(N)$, $W(N)$, $\bar{S}(N)$, and $\bar{W}(N)$ using (3.9) and (3.18).
3: Calculate, in a backward run, the Lagrange multipliers $\mu^o(k)$, $\lambda^o(k)$, and $\bar{\lambda}^o(k)$ using (3.10).
4: Calculate, in a backward run, the matrices $Z(k)$, the gains $K_1^*(k)$, $K_2^*(k)$, $K_3^*(k)$, $K_4^*(k)$, and $K_5^*(k)$, and the matrices $S(k)$, $W(k)$, $T(k)$, $\bar{S}(k)$, $\bar{W}(k)$, and $\bar{T}(k)$ using (3.19), (3.37), (3.45), (3.16), (3.17), (3.42), and (3.44), respectively.
5: Given initial state variation $\delta x(0)$ and initial preview variation $\delta w(0)$, in a forward run, calculate $\delta\mu(k)$, $\delta C(k)$, $\alpha(k)$, $\delta x(k+1)$, and $\delta w(k+1)$ using (3.45), (3.46), (3.49), (3.37), and the variations of the system (3.11), respectively.
6: Set, in a forward run, $\alpha(k) = 1$ if $\alpha(k) < 0$ or $\alpha(k) > 1$. Then, find $\lambda = \min(\alpha(k))$. If $\lambda = 0$, change the activity status of the constraint for the corresponding time step k and go to Step 2.
7: Given $x^o(0)$, $w^o(0)$, $\lambda\delta x(0)$, and $\lambda\delta w(0)$, in a forward run, calculate $\delta u(k)$, $u(k)$, $\delta x(k+1)$, $\delta w(k+1)$, $x(k+1)$, and $w(k+1)$ using (3.37) and the variations of the system (3.11).
8: If $0 < \lambda < 1$, set $x^o(0) = x^o(0) + \alpha\delta x(0)$, $w^o(0) = w^o(0) + \alpha\delta w(0)$, $\delta x(0) = (1-\alpha)\delta x(0)$, $\delta w(0) = (1-\alpha)\delta w(0)$, and $j = j+1$. Then, go to Step 2.
9: If $\lambda = 1$, in a forward run, calculate $u^*(k) = u^o(k) + \sum_j \delta u_j(k)$, $x(k+1)$, and $w(k+1)$ using (3.37) and (3.1).

Remark 3 (Designing Parameters) *Considering suitable nominal models, the main design parameters of the proposed approach come from the original optimization problem* (3.5)*, which are the prediction number N and the designing weights in the stage cost $\phi(x,u,w)$ and the terminal cost $\psi(x,w)$. The prediction number N must be high enough so that the obtained optimal controller stabilizes the system; however, higher N causes higher computational cost to solve the optimization problem. Moreover, the designing weights in the costs must be selected such that both minimum tracking error and minimum control input are achieved.*

Remark 4 (Implementation) *The proposed ENE framework is easy to implement and light in computation. Specifically, given a nominal initial state $x^o(0)$, a nominal preview $w^o(0:N)$, a control objective function to minimize, system and control constraints, a nominal optimal state*

and control trajectory (x^o, u^o) *will be computed using an optimal control strategy. Note that this nominal solution can be computed offline and stored online, can be performed on a remote powerful controller (e..g, cloud), or computed ahead of time by utilizing the idling time of the processor. In the same time, the ENE adaptation gains* $K_1^*(k)$, $K_2^*(k), k = 0, 1, \cdots, N-1$ *in* (3.21) *can also be computed along with the nominal control law. During the online implementation, the actual initial state* $x(0)$ *and the actual preview* w *are likely different from the nominal values used for the optimal control computations. Instead of recomputing the optimal control sequence, the control correction* (3.21) *is computed, where* $\delta x(k) = x(k) - x^o(k)$ *and* $\delta w(k) = w(k) - w^o(k)$ *denote the state perturbation and the preview perturbation, respectively. Then, the final control is used as* $u^*(k) = u^o(k) + \delta u(k)$*, and this implementation is easily extended for the modified ENE. As seen from the steps discussed above, the proposed approach is easy to implement and involves negligible online computational cost.*

Remark 5 (Nonlinear Model Predictive Control) *One can employ the nonlinear optimal control problem* (3.5) *as the open-loop nonlinear model predictive control (NMPC) or the closed-loop NMPC. For the open-loop version, providing the* N*-length nominal trajectory from the NMPC, the ENE algorithm approximates the NMPC policy such that it calculates two time-varying* N*-length feedback gains on the state and preview perturbations. Although the feedback gains are pre-computed, the ENE is able to take feedback from the real system for the* N *predictions in contrast to the open-loop NMPC. On the other hand, for the colsed-loop NMPC, we save the ENE solution but we only apply the first control input to the plant at each time step. Taking the feedback from the real system, the ENE solution from the previous step is considered as the nominal non-optimal solution, and the ENE algorithm is applied again to adapt the recent solution for the current time step.*

Remark 6 (Comparison) *In comparison with the existing NE frameworks [43, 42, 44, 45], we extend the regular NE approaches that only consider state deviations to a general setting that both state and preview deviations are considered. This is a significant extension as many modern control*

applications are employing preview information due to the increased availability of connectivity [39, 40, 41]. The necessity of adapting to preview perturbations is also demonstrated in our simulation studies, where we show that the proposed ENE can significantly outperform the regular NE when the preview information has certain variations.

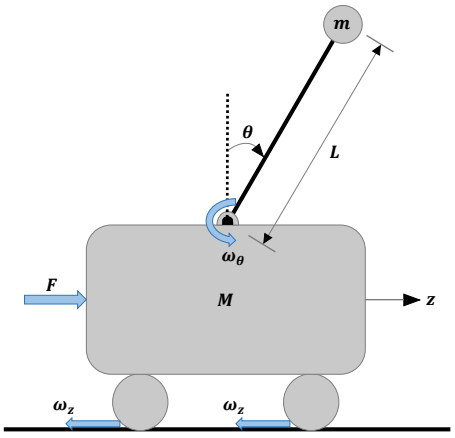

Figure 3.2: Cart-Inverted Pendulum.

3.4 Simulation Results

In this part, we demonstrate the performance of the proposed ENE framework for both small and large perturbations via a simulation example. The simulation example is adopted from the cart-inverted pendulum (see Fig. 3.2) whose system dynamics is described by:

$$\begin{aligned} \ddot{z} &= \frac{F - K_d\dot{z} - m(L\dot{\theta}^2\sin(\theta) - g\sin(\theta)\cos(\theta)) - 2w_z}{M + m\sin^2(\theta)}, \\ \ddot{\theta} &= \frac{\ddot{z}\cos(\theta) + g\sin(\theta)}{L} - \frac{w_\theta}{mL^2}, \end{aligned} \tag{3.50}$$

where z and θ denote the position of the cart and the pendulum angle. $m = 1$kg, $M = 5$kg, and $L = 2$m represent the mass of the pendulum, the mass of the cart, and the length of the pendulum, respectively. $g = 9.81\text{m/s}^2$ and $K_d = 10$Ns/m are respectively the gravity acceleration and the damping parameter. The variable force F controls the system under a friction force w_z and a friction torque w_θ. $T = 0.1$s is considered as the sampling time for discretization of the model

(3.50), and we assume that we have certain preview of w_z and w_θ. The states, the outputs, the preview information, and the control input constraint are respectively expressed as

$$x = [x_1, x_2, x_3, x_4]^T = [z, \dot{z}, \theta, \dot{\theta}]^T,$$

$$y = [x_1, x_3]^T = [z, \theta]^T,$$

$$w = [w_1, w_2, w_3, w_4]^T = [0, w_z, 0, w_\theta]^T,$$

$$-300 \leq F \leq 300.$$

The following values are used for the simulation: $N = 35$, $x^o(0) = [0, 0, -\pi, 0]^T$, $w^o(0) = [0, 0.1, 0, 0.1]^T$. Moreover, the nominal preview model is represented as $w^o(k+1) = -0.008x^o(k) - 0.1w^o(k)$. For the small perturbation setting, the initial state perturbation and the actual friction profile are set as $\delta x(0) = [0.01, 0.01, 0.01, 0.01]^T$ and $w(k) = 0.004\sin(k) + 0.004\text{rand}(k) + 0.002$, respectively. For the large perturbation setting, the initial state perturbation and the actual friction profile are chosen as $\delta x(0) = [0.2, 0.2, 0.2, 0.2]^T$ and $w(k) = 0.015\sin(k) + 0.015\text{rand}(k) + 0.01$, respectively.

Figs. 3.3-3.5 show the control performance of the open-loop NMPC, the standard NE, the ENE, and the closed-loop NMPC subject to the small perturbations. For the open-loop NMPC, under the nominal initial state $x^o(0)$ and preview $w^o(0)$, we obtain the N-length open-loop trajectory (x^o, u^o, w^o) and apply the open-loop control u^o to the system as shown in Fig. 3.3. It is worth noting that the state and preview information are updated during the optimization problem based on the considered nominal model (3.50) and the nominal preview model $w^o(k+1) = -0.008x^o(k) - 0.1w^o(k)$, respectively. However, since it is the open-loop version of the NMPC, the controller does not take the feedback from the real states and preview, makes the least control force, and leads to degraded performance due to the state and preview deviations as shown in Fig. 3.4. The NE is capable of taking the state feedback from the real system and adjusting the nominal optimal control, the open-loop control trajectory obtained by the NMPC, for the state perturbations. From Fig. 3.4, one can see that the NE does show an improved performance as compared to the

open-loop NMPC but it falls short against the ENE since it only handles the state perturbations without adapting to the preview perturbations. In comparison with the open-loop NMPC and the NE, the proposed ENE takes the state and preview feedback from the real system and achieves better performance, where it promptly stabilizes the system with the minimum cost in the presence of state and preview perturbations as shown in Fig. 3.5. Although we employ the ENE for the open-loop NMPC, due to the feedback from the real system, the ENE shows a similar control performance as the closed-loop NMPC for this case as shown in Figs. 3.4 and 3.5. However, the closed-loop NMPC has high computational cost since it solves the optimization problem (3.5) at each step.

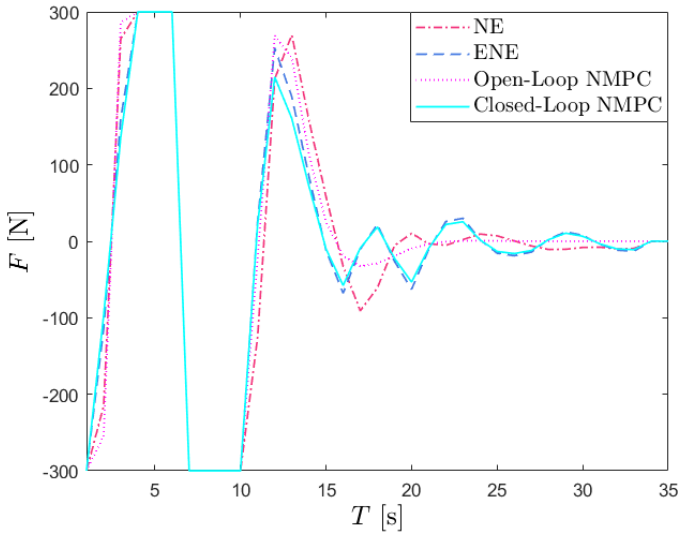

Figure 3.3: Control Input for Small Perturbation.

Figs. 3.6 and 3.7 illustrate the control performance of the open-loop NMPC, the NE, the ENE, the modified NE, the modified ENE, and the closed-loop NMPC subject to large perturbations. As shown in Fig. 3.6, one can see that the considered large perturbations change the activity status of the input constraint, and it causes that the NE and the ENE violates the constraint due to the absence of the intermediate points between the nominal initial state and preview and the perturbed ones. However, the modified NE and the modified ENE satisfies the constraint, and the modified ENE indicates a similar performance as the closed-loop NMPC as shown in Fig.

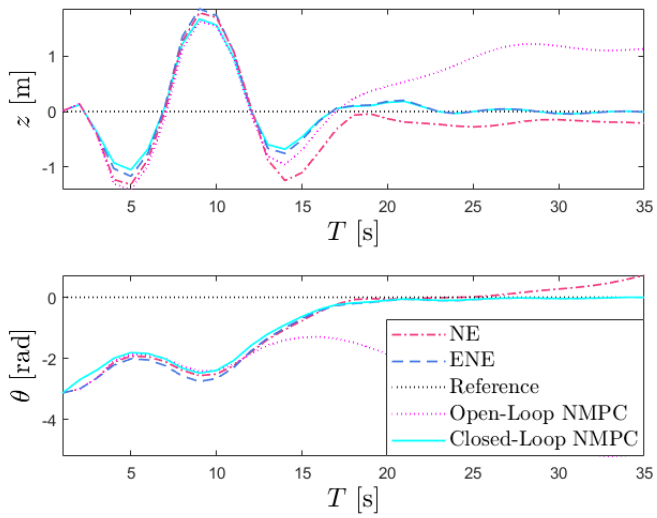

Figure 3.4: System Outputs for Small Perturbation.

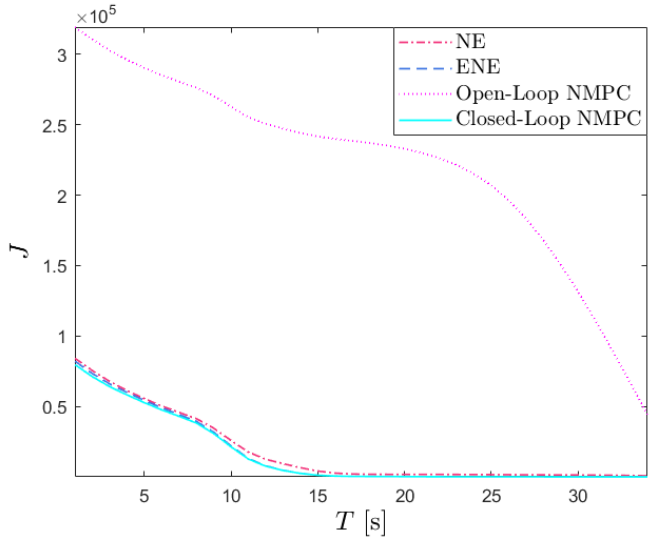

Figure 3.5: Cost for Small Perturbation.

3.7. Moreover, to see the role of the nominal preview model on the proposed control scheme, Figs. 3.8 and 3.9 compare the results of the ENE and the modified ENE for two nominal preview models $w^o(k+1) = w^o(k)$ and $w^o(k+1) = -0.008x^o(k) - 0.1w^o(k)$ with the actual friction profile $w(k) = 0.008\sin(k) + 0.008\text{rand}(k) + 0.004$. One can see that the activity status of the constraint is changed under the considered perturbation; however, it is not high enough to cause

the constraint violation for the ENE. Furthermore, it can be seen that both the modified ENE and the ENE accomplish better control performance when the preview model $w^o(k+1) = w^o(k)$ is applied. Providing a suitable nominal preview model leads to well control performance by the proposed ENE and modified ENE.

Table I compares the performances (i.e. $\|y - r\|$) and the computational times of the proposed controllers for the small perturbation. Based on the formulations, it is obvious that the ENE and the modified ENE (MENE) show the same performance and computational time for the small perturbations. We also have same result for the NE and the modified NE (MNE) for the small perturbations. Table II compares the performances and the computational times of the proposed controllers for the large perturbations. In Tables I and II, the closed-loop NMPC (CLNMPC) and the open-loop NMPC (OLNMPC) show the best and the worst performance, respectively; however, considering both performance and computational time, the modified ENE presents the best results.

The simulation setup is widely applicable as in many modern applications, a nominal preview model is available while the actual corresponding signal can also be measured or estimated online. For example, a wind energy forecast model is obtained using a deep federated learning approach [35], which can be served as a nominal preview model, and the wind disturbance can also be measured using light detection and ranging systems in real time [41]. For the considered cart-inverted pendulum simulations, the nominal preview information is obtained using a nominal model, i.e. $w^o(k+1) = -0.008x^o(k) - 0.1w^o(k)$; however, for each time step k, we generate the real preview information as $w(k) = 0.004\sin(k) + 0.004\text{rand}(k) + 0.002$, which leads to a perturbation from the nominal one. Providing a nominal solution based on the nominal state and preview, the proposed ENE framework adapts the nominal control to the perturbations generated by the measured/estimated real state and preview information. Furthermore, to simulate the large perturbation case, we follow the same process but change the real preview information as $w(k) = 0.015\sin(k) + 0.015\text{rand}(k) + 0.01$ for Figs. 3.6 and 3.7 and $w(k) = 0.008\sin(k) + 0.008\text{rand}(k) + 0.004$ for Figs. 3.8 and 3.9.

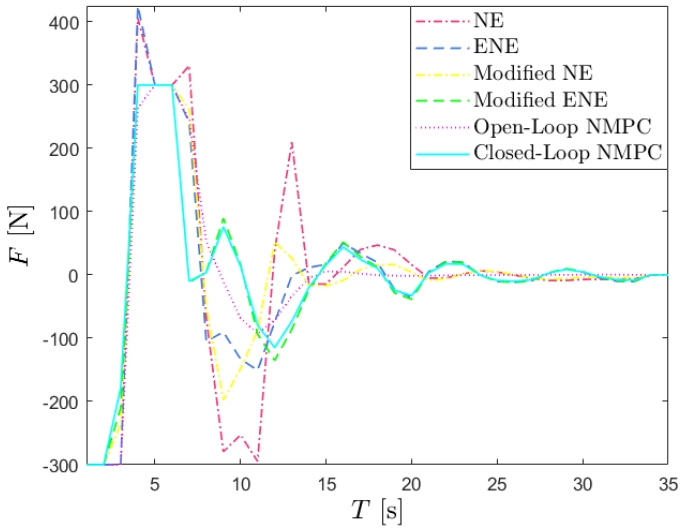

Figure 3.6: Control Input for Large Perturbation.

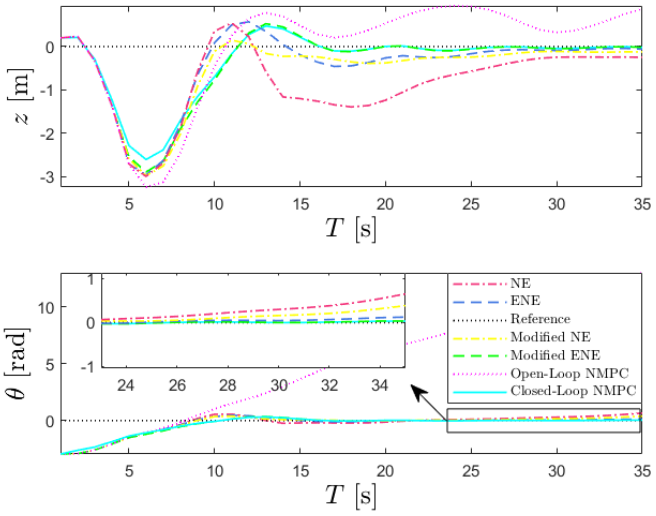

Figure 3.7: System Outputs for Large Perturbation.

3.5 Chapter Summary

The ENE algorithm was developed to adapt a nominal trajectory to the state and preview perturbations, and a multi-segment strategy was employed to handle the large perturbations. Simulations demonstrated the ENE's technological advances over the NE and the NMPC, and the nominal preview model is crucial to the effectiveness of the ENE. The proposed ENE framework is

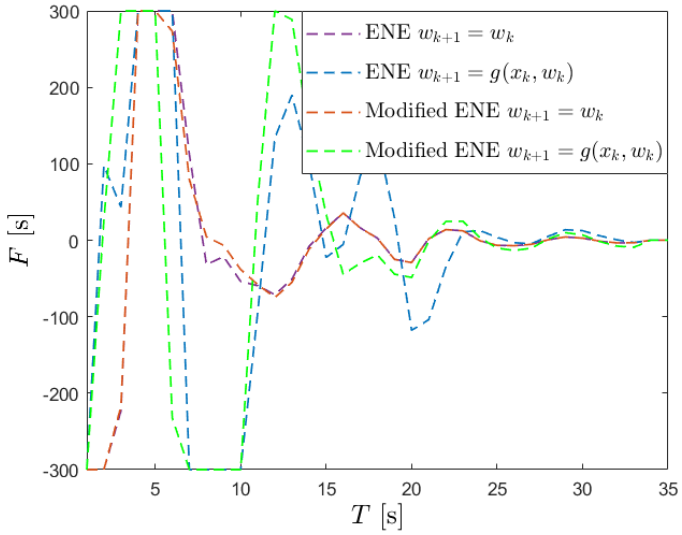

Figure 3.8: Control Input for Different Nominal Preview Models.

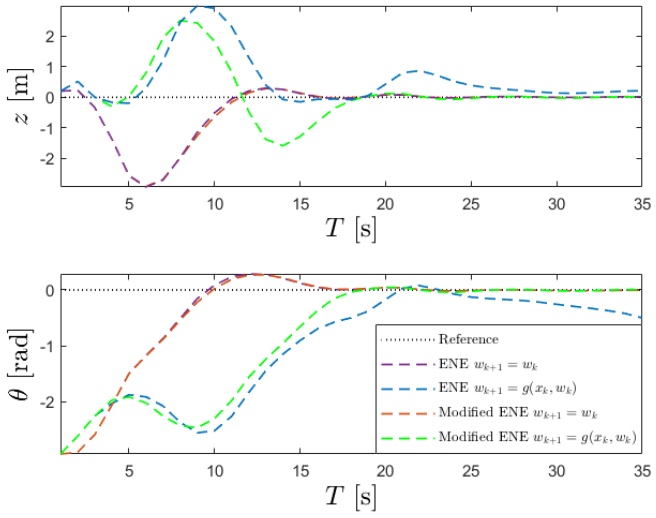

Figure 3.9: System Outputs for Different Nominal Preview Models.

applicable to general optimal control problem setting as there is no assumption on the under/over-actuation of the system. If a regular optimal control implementation can yield good performance, the ENE is expected to yield comparable performance with less computation complexity, where the computational load of the ENE grows linearly for the optimization horizon.

Table 3.1: Comparison of Controllers for Small Perturbations.

Control	Performance	Time (per loop)
CLNMPC	5.5735	5.7179 ms
MENE	5.6429	0.0659 ms
ENE	5.6429	0.0659 ms
MNE	5.9846	0.0658 ms
NE	5.9846	0.0658 ms
OLNMPC	19.2123	0.1770 ms

Table 3.2: Comparison of Controllers for Large Perturbations.

Control	Performance	Time (per loop)
CLNMPC	6.1609	5.7179 ms
MENE	6.2704	0.1225 ms
ENE	6.6838	0.0659 ms
MNE	6.7045	0.1224 ms
NE	7.4038	0.0658 ms
OLNMPC	41.8378	0.1770 ms

CHAPTER 4

DATA-ENABLED NEIGHBORING EXTREMAL CONTROL

In this chapter, we study the problem of data-driven optimal trajectory tracking for the nonlinear systems with a non-parametric model. Given an initial I/O trajectory and a desired reference trajectory, the data-enabled predictive control (DeePC) provides an optimal control sequence using a data matrix on raw I/O data; however, this approach has shown high computational cost due to the dimension of the decision variable. We propose a data-enabled neighboring extremal (DeeNE) to approximate the DeePC policy and reduce its computational cost for the constrained nonlinear systems. The DeeNE adapts a pre-computed nominal DeePC solution to the perturbations of the initial I/O trajectory and the reference trajectory from the nominal ones. Simulation-based analysis is used to gain insights into the effects of the DeeNE, and experimental results validate that these insights carry over to the real-world systems. The results are demonstrated with a video of successful trajectory tracking of KINOVA Gen3 (7-DoF Arm Robot). [1]

4.1 Background

Optimization-based control strategies typically rely on accurate parametric representations of real systems, but this can be challenging for complex systems. Therefore, data-driven optimal controllers have become increasingly attractive to both academics and industry practitioners [20]. There are two paradigms of the data-driven optimal control: i) indirect data-driven optimal control first identifies a model using the I/O data and then conducts control design based on the identified model [10], and ii) direct data-driven optimal control circumvents the step of system identification and obtains control policy directly from the I/O data [23]. The direct data-driven optimal control may have higher flexibility and better performance compared to the indirect one [24].

[1] The material of this chapter is from "Computationally Efficient Data-Enabled Predictive Control for Arm Robots," 2024 [47] and "Data-Enabled Neighboring Extremal Optimal Control: A Computationally Efficient DeePC," IEEE 62nd Conference on Decision and Control (CDC), 2023 [48].

Recently, a result in the context of behavioral system theory [25], known as Fundamental Lemma [26], has received renewed attention in the direct data-driven optimal control. In the spirit of the Fundamental Lemma, a direct data-driven optimal control, called data-enabled predictive control (DeePC) [27], makes a transition from model-based optimal control strategies (e.g. model predictive control (MPC)) to a data-driven one. When perfect (noiseless and uncorrupted) I/O data is accessible, the DeePC can accurately predict the future behaviors of the LTI systems thanks to the Fundamental Lemma. In this case, the DeePC has equivalent closed-loop behavior to conventional MPC with a model and perfect state estimation [27]. However, in practice, perfect data is in general not accessible to the controller due to measurement and process noises, which leads to inaccurate estimations and predictions and may degrade the quality of the obtained optimal control sequence. Moreover, the Fundamental Lemma has been proposed for the LTI systems and is not perfect to learn the behaviors of the nonlinear systems. Therefore, the DeePC is robustified through suitable regularizations to ensure good performance under noisy data and nonlinearities [49, 50]. Furthermore, it has been shown that a quadratic regularization is essential for stability [23].

Although the DeePC plays an inevitable role in optimal control strategies, it is computationally expensive because of the dimension of the decision variable and solving an online optimization problem at each time step. Several approaches have been proposed to optimize a lower dimension decision variable and reduce the computational cost of the DeePC for the LTI systems. Subspace predictive control (SPC) [51, 30] identifies a reduced model for the linear DeePC using the singular value decomposition of the raw data; however, it is not a pure data-driven controller due to the identification part. Null-space predictive control (NPC) [28] introduces a lower dimension decision variable to reduce the computational cost of the DeePC, but it only works for the unconstrained linear DeePC. Minimum-dimension DeePC [5] uses the singular value decomposition to make more efficient numerical computation for the constrained linear DeePC. However, for the nonlinear systems, the computational cost of the DeePC is still a challenging problem and needs to be solved. It is worth noting that for the SPC and the minimum-dimension DeePC, the choice of the number

of the singular values to retain/cut is very critical and often not automatized.

In this chapter, we develop a data-enabled NE (DeeNE) framework for the nonlinear DeePC problem with initial I/O and reference perturbations. Moreover, we treat the DeeNE problem when nominal non-optimal solutions are present, and a modified control policy is developed to guarantee the control performance. Promising results are demonstrated by applying the developed controller to the cart inverted pendulum and the arm robot. The outline of this chapter is as follows. The problem formulation and the preliminaries of the DeePC are provided in Section II. Section III presents the proposed data-enabled neighboring extremal for the unknown nonlinear systems. Section IV presents the simulation results and experimental verifications. Finally, the conclusions are provided in Section V.

4.2 Problem Formulation

Consider a discrete-time nonlinear system in the following form:

$$\begin{aligned} x(k+1) &= f(x(k), u(k)), \\ y(k) &= h(x(k), u(k)), \end{aligned} \tag{4.1}$$

where $k \in \mathbb{N}^+$ denotes the time step, $x \in \mathbb{R}^n$ represents the state vector of the system, $u \in \mathbb{R}^m$ is the control input, and $y \in \mathbb{R}^p$ denotes the outputs of the system. Moreover, $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is the system dynamics with $f(0,0) = 0$, and $h : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^p$ represents the output dynamics.

Consider a safety constraint as

$$C(y(k), u(k)) \leq 0, \tag{4.2}$$

where $C : \mathbb{R}^p \times \mathbb{R}^m \to \mathbb{R}^l$.

Definition 3 (Closed-Loop Performance) *Consider the nonlinear system* (4.1) *and a tracking control problem with the desired trajectory* $r(k)$*. Starting from an initial state* x_0*, the closed-loop system performance over N steps is characterized by the following cost term:*

$$J_N(\mathbf{y}, \mathbf{u}) = \sum_{k=0}^{N-1} \phi(y(k), u(k)), \tag{4.3}$$

where $\mathbf{u} = [u(0),\ u(1),\ \cdots,\ u(N-1)]$, $\mathbf{y} = [y(0),\ y(1),\ \cdots,\ y(N-1)]$, *and* $\phi(y, u)$ *is the stage cost.*

The model-based optimal control aims at optimizing the system performance over N future steps for the real system (4.1), which is reduced to the following constrained optimization problem:

$$\begin{aligned} (\mathbf{y}^*, \mathbf{u}^*) &= \arg\min_{\mathbf{y},\mathbf{u}} J_N(\mathbf{y}, \mathbf{u}) \\ s.t. \quad & x(k+1) = f(x(k), u(k)) \\ & y(k) = h(x(k), u(k)) \\ & C(y(k), u(k)) \leq 0. \end{aligned} \tag{4.4}$$

The key ingredient for the optimal control (4.4) is an accurate parametric model of the system, but obtaining such a model, using plant modeling or identification procedures, is often the most time consuming and expensive part of control design.

4.2.1 Non-Parametric Representation of Unknown Systems

Inspired by Fundamental Lemma [26], the system model (4.1) is replaced by an algebraic constraint that enables us to predict the length-N future input-output (I/O) trajectory for a given length-T_{ini} past (I/O) trajectory.

The Hankel matrices $\mathbb{H}(u^d)$ and $\mathbb{H}(y^d)$ are built from the offline collected I/O samples u^d and y^d as:

$$\mathbb{H}(u^d) = \begin{bmatrix} u_1 & u_2 & \cdots & u_{T-T_{ini}-N+1} \\ u_2 & u_3 & \cdots & u_{T-T_{ini}-N+2} \\ \vdots & \vdots & \ddots & \vdots \\ u_{T_{ini}+N} & u_{T_{ini}+N+1} & \cdots & u_T \end{bmatrix}, \tag{4.5}$$

where $\mathbb{H}(u^d) \in \mathbb{R}^{m(T_{ini}+N)\times L}$ needs to have full row rank to satisfy the persistency of excitation requirement, and the number of its columns is denoted as $L = T - T_{ini} - N + 1$. The Hankel matrix of outputs $\mathbb{H}(y^d) \in \mathbb{R}^{p(T_{ini}+N)\times L}$ is built in an analogous way from the collected samples y^d.

Then, the Hankel matrices are partitioned in Past and Future subblocks as

$$\begin{bmatrix} U_P \\ U_F \end{bmatrix} =: \mathbb{H}(u^d), \quad \begin{bmatrix} Y_P \\ Y_F \end{bmatrix} =: \mathbb{H}(y^d), \tag{4.6}$$

where $U_P \in \mathbb{R}^{mT_{ini} \times L}$, $U_F \in \mathbb{R}^{mN \times L}$, $Y_P \in \mathbb{R}^{pT_{ini} \times L}$, and $Y_F \in \mathbb{R}^{pN \times L}$.

Lemma 2 (Fundamental Lemma [26]) *Consider a controllable linear time-invariant (LTI) system, there is a unique $g \in \mathbb{R}^L$ such that any length-$T_{ini} + N$ trajectory of the system satisfies the following linear equation under a full row rank $\mathbb{H}(u^d)$ as*

$$\begin{bmatrix} U_P \\ Y_P \\ U_F \\ Y_F \end{bmatrix} g = \begin{bmatrix} u_{ini} \\ y_{ini} \\ u \\ y \end{bmatrix}, \tag{4.7}$$

where U_P, Y_P, U_F, and Y_F are fixed data matrices obtained from the offline collected I/O data, (u_{ini}, y_{ini}) is a given length-T_{ini} initial trajectory, and (u, y) is a length-N future trajectory which is predicted online. □

4.2.2 Data-Enabled Predictive Control

For a given initial trajectory (u_{ini}, y_{ini}) collected from the real system (4.1), one can replace the optimization problem (4.4) with data-enabled predictive control (DeePC) as [27, 36]

$$\begin{aligned} (\mathbf{y}^*, \mathbf{u}^*, {\sigma_\mathbf{y}}^*, {\sigma_\mathbf{u}}^*, \mathbf{g}^*) = & \underset{\mathbf{y},\mathbf{u},\sigma_\mathbf{y},\sigma_\mathbf{u},\mathbf{g}}{\arg\min} \quad J_N(\mathbf{y}, \mathbf{u}, \sigma_\mathbf{y}, \sigma_\mathbf{u}, \mathbf{g}) \\ s.t. \quad & \begin{bmatrix} U_P \\ Y_P \\ U_F \\ Y_F \end{bmatrix} g = \begin{bmatrix} u_{ini} \\ y_{ini} \\ u \\ y \end{bmatrix} + \begin{bmatrix} \sigma_u \\ \sigma_y \\ 0 \\ 0 \end{bmatrix} \\ & C(y, u) \leq 0, \end{aligned} \tag{4.8}$$

where $\sigma_u \in \mathbb{R}^{mT_{ini}}$ is an auxiliary slack variable to cover process noises, $\sigma_y \in \mathbb{R}^{pT_{ini}}$ is an auxiliary slack variable to cover measurement noises and nonlinearities, and $J_N(\mathbf{y}, \mathbf{u}, \sigma_\mathbf{y}, \sigma_\mathbf{u}, \mathbf{g})$ is the modified cost function for data-driven controllers with noisy data and nonlinearities.

Now, using $y = Y_F g$, $u = U_F g$, $\sigma_y = Y_P g - y_{ini}$, and $\sigma_u = U_P g - u_{ini}$ as free optimization variables, one can rewrite (4.8) as

$$\begin{aligned} \mathbf{g}^* = \arg\min_{\mathbf{g}} \; & J_N(Y_F\mathbf{g}, U_F\mathbf{g}, Y_P g - y_{ini}, U_P g - u_{ini}, g) \\ s.t. \quad & C(Y_F g, U_F g) \leq 0. \end{aligned} \tag{4.9}$$

If the constraint $C(y, u)$ was absent in (4.8), the problem is referred to the unconstrained DeePC, and the solution is available in closed form with reduced computational burden. For this case, one has $u = U_F g = K_d^r r + K_d^{ini} w_{ini}$ as the DeePC policy, where $K_d^r \in \mathbb{R}^{mN \times pN}$ and $K_d^{ini} \in \mathbb{R}^{mN \times (m+p)T_{ini}}$ are control gains, r is the desired reference trajectory, and $w_{ini} = \left[u_{ini}^T, y_{ini}^T\right]^T$ is the given initial trajectory. However, the constrained DeePC (4.8) requires an iterative solver; therefore, the DeePC may suffer from high computational cost since the dimension of the decision variable g depends on the length of the collected data T in the Hankel matrix.

4.3 Main Result

In this section, given a nominal solution (g^o, u^o, y^o), we propose a data-enabled neighboring extremal (DeeNE) to approximate the DeePC policy in the presence of initial (I/O) and reference trajectories perturbations. We assume that the initial I/O and reference trajectories perturbations are small enough such that they do not change the activity status of the constraint. The resulting equation helps us to reduce the time and effort in computing the data-driven optimal control for the system (4.1).

4.3.1 Nominal Lagrange Multipliers

Considering (4.9), the augmented cost function are defined as:

$$\bar{J}_N(w_{ini}, g, r, \mu) = J_N(w_{ini}, g, r) + \mu^T C^a(w_{ini}, g). \tag{4.10}$$

where $C^a(w_{ini}, g)$ represents the active constraints, and μ is the Lagrange multiplier associated with the active constraints.

Let (w^o_{ini}, g^o, r^o) represents the nominal solution for the DeePC (4.8). The nominal solution satisfies the following KKT conditions for the augmented cost function (4.10) as:

$$\begin{aligned} &\bar{J}_g(w_{ini}, g, r, \mu) = 0, \\ &\mu \geq 0, \end{aligned} \tag{4.11}$$

where $\bar{J}_g$ indicates $\nabla \bar{J}_N / \nabla g$.

Assumption 3 (Active Constraints) *$C^a_g(w_{ini}, g)$ is full row rank.*

Now, substituting the nominal solution (w^o_{ini}, g^o, r^o) into the above KKT condition, one can calculate the Lagrange multiplier μ online. From (4.11), it follows that

$$J_g(w^o_{ini}, g^o, r^o) + \mu^T C^a_g(w^o_{ini}, g^o) = 0. \tag{4.12}$$

Using the above equation, the Lagrange multiplier can be obtained online as:

$$\mu = -(C^a_g C^{aT}_g)^{-1} C^a_g J^T_g. \tag{4.13}$$

Note that Assumption 3 guarantees that $C^a_g C^{aT}_g$ is invertible. Moreover, it is worth noting that $\mu = 0$ if the constraint $C(w^o_{ini}, g^o)$ is not active. The Lagrange multiplier (4.13) is considered as the nominal optimal Lagrange multiplier μ^o.

4.3.2 Data-Enabled Neighboring Extremal

For this part, we consider the nominal solution (g^o, u^o, y^o) as an optimal solution obtained by the DeePC. To adapt to initial I/O and reference perturbations from the nominal values, the DeeNE seeks to minimize the second-order variation of (4.10) subject to linearized constraints. More specifically, the DeeNE algorithm solves the following optimization problem with the given information δw_{ini} and δr as:

$$\begin{aligned} \delta \mathbf{g}^* &= \arg\min_{\delta \mathbf{g}} J^{ne}_N \\ s.t. \quad & C^a_g \delta g = 0, \end{aligned} \tag{4.14}$$

where

$$J_N^{ne} = \delta^2 \bar{J}_N = \frac{1}{2} \begin{bmatrix} \delta w_{ini} \\ \delta g \\ \delta r \end{bmatrix}^T \begin{bmatrix} \bar{J}_{w_{ini}w_{ini}} & \bar{J}_{w_{ini}g} & \bar{J}_{w_{ini}r} \\ \bar{J}_{gw_{ini}} & \bar{J}_{gg} & \bar{J}_{gr} \\ \bar{J}_{rw_{ini}} & \bar{J}_{rg} & \bar{J}_{rr} \end{bmatrix} \begin{bmatrix} \delta w_{ini} \\ \delta g \\ \delta r \end{bmatrix}. \tag{4.15}$$

For (4.14), the augmented cost function are obtained as:

$$\bar{J}_N^{ne} = \frac{1}{2} \begin{bmatrix} \delta w_{ini} \\ \delta g \\ \delta r \end{bmatrix}^T \begin{bmatrix} \bar{J}_{w_{ini}w_{ini}} & \bar{J}_{w_{ini}g} & \bar{J}_{w_{ini}r} \\ \bar{J}_{gw_{ini}} & \bar{J}_{gg} & \bar{J}_{gr} \\ \bar{J}_{rw_{ini}} & \bar{J}_{rg} & \bar{J}_{rr} \end{bmatrix} \begin{bmatrix} \delta w_{ini} \\ \delta g \\ \delta r \end{bmatrix} + \delta\mu^T C_g^a \delta g, \tag{4.16}$$

where $\delta\mu$ is the Lagrange multiplier.

By applying the KKT conditions to (4.16), one has

$$\begin{aligned} \bar{J}_{\delta g}^{ne} &= 0, \\ \delta\mu &\geq 0. \end{aligned} \tag{4.17}$$

where $\bar{J}_{\delta g}^{ne}$ indicates $\nabla \bar{J}_N^{ne} / \nabla \delta g$.

Considering the DeePC (4.8), we have a nominal solution (g^o, u^o, y^o) for an initial I/O trajectory (u_{ini}^o, y_{ini}^o) and reference trajectory r^o. For a new initial I/O trajectory (u_{ini}, y_{ini}) and reference trajectory r, the optimal solution is approximated by $u^* = u^o + \delta u$ using the DeeNE adaptation. The objective is now to develop a DeeNE framework for the data-driven optimal trajectory tracking problem. The following theorem presents the proposed DeeNE to approximate the DeepC policy in the presence of initial I/O and reference perturbations.

Theorem 3 (Data-Enabled Neighboring Extremal) *Consider the optimization problem* (4.14)*, the augmented cost function* (4.16)*, and the KKT conditions* (4.17)*. If* $\bar{J}_{gg} > 0$*, then the DeeNE*

policy

$$
\begin{aligned}
\delta g &= K_1^* \delta w_{ini} + K_2^* \delta r, \\
K_1^* &= -\begin{bmatrix} I & 0 \end{bmatrix} K^o \begin{bmatrix} \bar{J}_{gw_{ini}} \\ 0 \end{bmatrix}, \\
K_2^* &= -\begin{bmatrix} I & 0 \end{bmatrix} K^o \begin{bmatrix} \bar{J}_{gr} \\ 0 \end{bmatrix}, \\
K^o &= \begin{bmatrix} \bar{J}_{gg} & C_g^{aT} \\ C_g^a & 0 \end{bmatrix}^{-1}
\end{aligned}
\tag{4.18}
$$

approximates the perturbed solution for the DeePC (4.8) *in the presence of initial I/O perturbation* δw_{ini} *and reference perturbation* δr.

Proof 3 *Using* (4.16) *and the KKT conditions* (4.17)*, one has*

$$
\bar{J}_{gw_{ini}} \delta w_{ini} + \bar{J}_{gg} \delta g + \bar{J}_{gr} \delta r + C_g^{aT} \delta \mu = 0. \tag{4.19}
$$

Now, using (4.19) *and the linearized safety constraints* (4.14)*, one has*

$$
\begin{bmatrix} \bar{J}_{gg} & C_g^{aT} \\ C_g^a & 0 \end{bmatrix} \begin{bmatrix} \delta g \\ \delta \mu \end{bmatrix} = -\begin{bmatrix} \bar{J}_{gw_{ini}} \\ 0 \end{bmatrix} \delta w_{ini} - \begin{bmatrix} \bar{J}_{gr} \\ 0 \end{bmatrix} \delta r, \tag{4.20}
$$

which yields

$$
\begin{bmatrix} \delta g \\ \delta \mu \end{bmatrix} = -K^o \begin{bmatrix} \bar{J}_{gw_{ini}} \\ 0 \end{bmatrix} \delta w_{ini} - K^o \begin{bmatrix} \bar{J}_{gr} \\ 0 \end{bmatrix} \delta r. \tag{4.21}
$$

Thus, the DeeNE policy (4.18) *is obtained, and the proof is completed.* □

Remark 7 (Singularity) *It is worth noting that the assumption of* $\bar{J}_{gg}$ *being positive definite (i.e.,* $\bar{J}_{gg} > 0$*) is essential for the DeeNE.* $\bar{J}_{gg} > 0$ *is performed to calculate the DeeNE such that it guarantees the convexity of* (4.14)*. Considering* $\bar{J}_{gg} > 0$ *and Assumption 3, it is clear that* K^o *in* (4.18) *is well defined. If* C_g^a *is not full row rank, the matrix* K^o *is singular, leading to the failure of the proposed algorithm. This issue can be solved using the constraint back-propagation algorithm presented in [15].*

Remark 8 (Control Input) *Using the control policy* (4.18)*, one can obtain* $g^* = g^o + \delta g$*, then* $u^* = u^o + \delta u$ *is obtained using* $u = U_F g$*. Therefore, one can conclude that* $\delta u = K_{ne}^r \delta r + K_{ne}^{ini} \delta w_{ini}$*.*

4.3.3 Nominal Non-Optimal Solution

The DeeNE is derived under the assumption that a nominal DeePC solution is available. In this subsection, we modify the DeeNE policy for a nominal non-optimal solution so that we can use the DeeNE solution as the nominal solution during the control process. For the nominal non-optimal sequences (w_{ini}^o, g^o, r^o), we assume that they satisfy the constraints described in (4.8) but may not satisfy the optimality condition $\bar{J}_g(w_{ini}^o, g^o, r^o, \mu^o) = 0$. Under this circumstance, the cost function (4.15) is modified as:

$$J_N^{ne} = \delta^2 \bar{J}_N + \bar{J}_g^T \delta g = \frac{1}{2} \begin{bmatrix} \delta w_{ini} \\ \delta g \\ \delta r \end{bmatrix}^T \begin{bmatrix} \bar{J}_{w_{ini} w_{ini}} & \bar{J}_{w_{ini} g} & \bar{J}_{w_{ini} r} \\ \bar{J}_{g w_{ini}} & \bar{J}_{gg} & \bar{J}_{gr} \\ \bar{J}_{r w_{ini}} & \bar{J}_{rg} & \bar{J}_{rr} \end{bmatrix} \begin{bmatrix} \delta w_{ini} \\ \delta g \\ \delta r \end{bmatrix} + \bar{J}_g^T \delta g. \tag{4.22}$$

Considering the optimal control problem (4.14) and the cost function (4.22), the augmented cost function is modified as:

$$\bar{J}_N^{ne} = \frac{1}{2} \begin{bmatrix} \delta w_{ini} \\ \delta g \\ \delta r \end{bmatrix}^T \begin{bmatrix} \bar{J}_{w_{ini} w_{ini}} & \bar{J}_{w_{ini} g} & \bar{J}_{w_{ini} r} \\ \bar{J}_{g w_{ini}} & \bar{J}_{gg} & \bar{J}_{gr} \\ \bar{J}_{r w_{ini}} & \bar{J}_{rg} & \bar{J}_{rr} \end{bmatrix} \begin{bmatrix} \delta w_{ini} \\ \delta g \\ \delta r \end{bmatrix} + \bar{J}_g^T \delta g + \delta \mu^T C_g^a \delta g. \tag{4.23}$$

Now, the following theorem is presented to modify the DeeNE policy for the nominal non-optimal solutions to the data-driven nonlinear optimal control problem.

Theorem 4 (Modified Data-Enabled Neighboring Extremal) *Consider the optimization problem* (4.14)*, the KKT conditions* (4.17)*, and the augmented cost function* (4.23)*. If* $\bar{J}_{gg} > 0$*, then the DeeNE policy is modified for a nominal non-optimal solution as:*

$$\begin{aligned} \delta g &= K_1^* \delta w_{ini} + K_2^* \delta r + K_3^* \begin{bmatrix} \bar{J}_g \\ 0 \end{bmatrix}, \\ K_3^* &= -\begin{bmatrix} I & 0 \end{bmatrix} K^o, \end{aligned} \tag{4.24}$$

where the gain matrices K_1^*, K_2^*, and K^o are defined in (4.18)*.*

Proof 4 *Using the KKT conditions* (4.17) *and the modified augmented cost function* (4.23)*, one has*

$$\bar{J}_{gw_{ini}}\delta w_{ini} + \bar{J}_{gg}\delta g + \bar{J}_{gr}\delta r + C_g^{aT}\delta\mu + \bar{J}_g = 0. \tag{4.25}$$

Now, using (4.25) *and the linearized safety constraints* (4.14)*, one has*

$$\begin{bmatrix} \bar{J}_{gg} & C_g^{aT} \\ C_g^a & 0 \end{bmatrix} \begin{bmatrix} \delta g \\ \delta\mu \end{bmatrix} = - \begin{bmatrix} \bar{J}_{gw_{ini}} \\ 0 \end{bmatrix} \delta w_{ini} - \begin{bmatrix} \bar{J}_{gr} \\ 0 \end{bmatrix} \delta r - \begin{bmatrix} \bar{J}_g \\ 0 \end{bmatrix}, \tag{4.26}$$

which yields

$$\begin{bmatrix} \delta g \\ \delta\mu \end{bmatrix} = -K^o \begin{bmatrix} \bar{J}_{gw_{ini}} \\ 0 \end{bmatrix} \delta w_{ini} - K^o \begin{bmatrix} \bar{J}_{gr} \\ 0 \end{bmatrix} \delta r - K^o \begin{bmatrix} \bar{J}_g \\ 0 \end{bmatrix}. \tag{4.27}$$

Thus, the modified DeeNE policy (4.24) *is obtained, and the proof is completed.* □

Remark 9 (Quadratic Cost) *One can consider a quadratic cost function $J_N(\mathbf{y}, \mathbf{u}, \sigma_\mathbf{y}, \sigma_\mathbf{u}, \mathbf{g})$ as:*

$$J_N(\mathbf{y}, \mathbf{u}, \sigma_\mathbf{y}, \sigma_\mathbf{u}, \mathbf{g}) = \|y - r\|_Q^2 + \|u\|_R^2 + \lambda_y\|\sigma_y\|_2^2 + \lambda_u\|\sigma_u\|_2^2 + \lambda_g\|g\|_2^2, \tag{4.28}$$

where the positive semi-definite matrix $Q \in \mathbb{R}^{pN\times pN}$ and the positive definite matrix $R \in \mathbb{R}^{mN\times mN}$ are weighting matrices, and the positive parameters $\lambda_y, \lambda_u, \lambda_g \in \mathbb{R}$ are regularization weights. For the quadratic cost function (4.28)*, the DeePC is a quadratic program (QP) problem on the decision variable g, which requires an iterative solver, i.e. an online QP solver such as qpOASES [52]. To use the DeeNE, we have*

$$\begin{aligned}
\bar{J}_g &= 2((Y_F g - r)^T Q Y_F + (U_F g)^T R U_F + \lambda_y (Y_P g - y_{ini})^T Y_P + \lambda_u (U_P g - u_{ini})^T U_P + \lambda_g g^T), \\
\bar{J}_{gg} &= 2(Y_F^T Q Y_F + U_F^T R U_F + \lambda_y Y_P^T Y_P + \lambda_u U_P^T U_P + \lambda_g), \\
\bar{J}_{gw_{ini}} &= -2(\lambda_y Y_P^T + \lambda_u U_P^T), \\
\bar{J}_{gr} &= -2Y_F^T Q,
\end{aligned} \tag{4.29}$$

where one can see that $\bar{J}_{gg} > 0$.

Remark 10 (Robustness) *The DeePC* (4.8)*, the DeeNE* (4.18)*, and the modified DeeNE* (4.24) *show that even though an accurate prediction of the future behavior is unattainable in practice due to the noisy I/O data and the nonlinearities, the obtained optimal control sequence provides performance guarantees for the actually realized I/O cost.*

Algorithm 1 summarizes the modified DeeNE procedure for adaptation the pre-computed nominal control solution u^o to the small initial I/O trajectory perturbation δw_{ini} and reference trajectory perturbation δr such that it achieves the optimal control as $u^* = u^o + \delta u$ using Theorem 4. One knows that at the first step, we have the nominal optimal solution from the DeePC as the input of the algorithm; thus, Theorem 1 and Theorem 2 represent same control policy since $\bar{J}_g(w^o_{ini}, g^o, r^o, \mu^o) = 0$ for nominal optimal solution. However, we use Theorem 4 for all time steps since we use the DeeNE solution as the nominal solution for the future time steps. Moreover, it is worth noting that choosing $s > 0$ reduces the computational cost, and in some cases may improve the control performance [27, 53].

Algorithm 3 Data-Enabled Neighboring Extremal.

Input: The data matrices U_P, Y_P, U_F, and Y_F; the function C; the weighting matrices Q and R; the regularization weights λ_y, λ_u, and λ_g; the nominal initial I/O w^o_{ini} and reference r^o trajectories; and the nominal optimal solution (g^o, u^o, y^o) from the DeePC.
1: Calculate the Lagrange multiplier using the nominal optimal solution g^o and the nominal initial I/O w^o_{ini} and reference r^o trajectories in (4.13).
2: Calculate the gains K^*_1 and K^*_2 using (4.18).
3: Given the real initial I/O w_{ini} and reference r trajectories, calculate δw_{ini} and δr, respectively, and then δg and g^* using (4.18).
4: Compute the optimal I/O sequences $u^* = U_F g^*$ and $y^* = Y_F g^*$.
5: Apply optimal control input $(u(k), u(k+1), \cdots, u(k+s)) = (u^*_0, u^*_1, \cdots, u^*_s)$ to the plant for some $s \leq N - 1$.
6: Update the nominal initial I/O trajectory, reference trajectory, and optimal solution as $w^o_{ini} = w_{ini}$, $r^o = r$, $g^o = g^*$, $u^o = u^*$ and $y^o = y^*$.
7: Set k to k + s and update the initial I/O trajectory w_{ini} and the reference trajectory r to the T_{ini} most recent I/O measurements.
8: Return to (1).

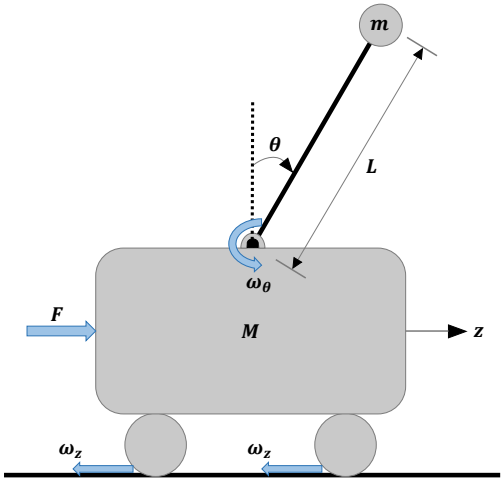

Figure 4.1: Cart-Inverted Pendulum.

4.4 Simulation Results

In this section, we demonstrate the performance of the proposed DeeNE framework via two simulation examples, i.e., a cart-inverted pendulum and an arm robot.

4.4.1 Cart-Inverted Pendulum

For a cart-inverted pendulum (see Fig. 4.1), one has

$$\begin{aligned} \ddot{z} &= \frac{F - K_d\dot{z} - m(L\dot{\theta}^2\sin(\theta) - g\sin(\theta)\cos(\theta)) - 2d_z}{M + m\sin^2(\theta)}, \\ \ddot{\theta} &= \frac{\ddot{z}\cos(\theta) + g\sin(\theta)}{L} - \frac{d_\theta}{mL^2}, \end{aligned} \tag{4.30}$$

where z and θ denote the position of the cart and the pendulum angle. $m = 1kg$, $M = 5kg$, and $L = 2m$ represent the mass of the pendulum, the mass of the cart, and the length of the pendulum. $g = 9.81m/s^2$ and $K_d = 10Ns/m$ are the gravity acceleration and the damping parameter. The variable force F controls the system under a friction force d_z and a friction torque d_θ. $T_s = 0.02s$ is considered as the sampling time for discretization of the model (4.30), and we assume d_z and d_θ as the process noises. The states, the outputs, the process noise, the measurement noise, and the control input constraint are expressed as

$$x = [x_1, x_2, x_3, x_4]^T = [z, \dot{z}, \theta, \dot{\theta}]^T,$$

$$y = [x_1, x_3]^T + v = [z, \theta]^T + v,$$

$$d = [d_1, d_2, d_3, d_4]^T = [0, d_z, 0, d_\theta]^T,$$

$$-50 \leq F \leq 50.$$

where d and v represent the process noise and measurement noise, respectively.

The following values are used for the simulation: $T_{ini} = 30$, $N = 45$, the simulation time $T = 200$, $x(0) = [0, 0, \pi/270, 0]^T$, and $d_z, d_\theta = 0.002(2rand(1, T) - 1)$. We generate the first initial trajectory (u_{ini}, y_{ini}) using zero control input, i.e. $u_{ini} = u(0 : 29) = 0$, which leads to the state $x(30) = [0.0151, 0.0783, 0.1225, 0.6200]^T$. Figs. 4.2 and 4.3 show the control performances of the DeeNE and the DeePC. For the DeePC, we use the DeePC policy (4.8), apply the length-s optimal control sequence to the plant, and update the initial trajectory w_{ini} for the next step (see Algorithm 2 in [27]). As we discussed in Algorithm 1, we use DeeNE policy (4.24) to avoid solving the DeePC problem at each step and reduce the computational cost. As is obvious from the Fig. 4.2, one can see that the DeeNE policy approximates the DeePC policy very well and is capable of adjusting the nominal DeePC by fully considering the initial I/O trajectory perturbations. From Fig. 4.2, one can see that the DeeNE provides similar performance for trajectory tracking problem as compared to the DeePC; however, the DeeNE reduces the computational cost of the DeePC very well as shown in Table II. Table I compares the cost-based performance and the computational time for the DeePC under various values of s, where demonstrates that we have the best performance for the considered system with $s = 5$. Table II illustrates the cost-based performance and the computational time for both DeePC and DeeNE under two cases $s = 0$ and $s = 5$, and one can see that the DeeNE with $s = 5$ shows the best performance for the regulation of the cart-inverted pendulum.

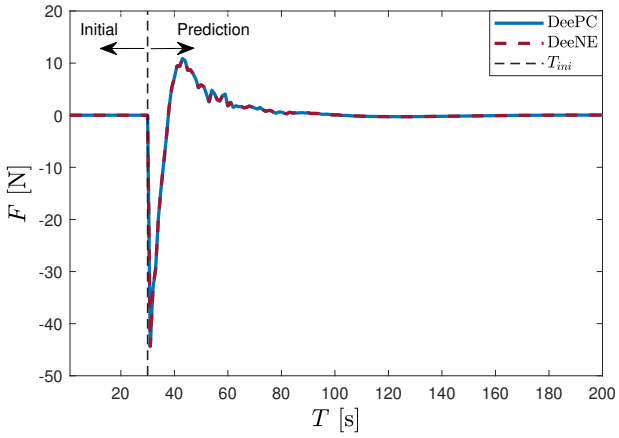

Figure 4.2: Control Input for Cart-Inverted Pendulum.

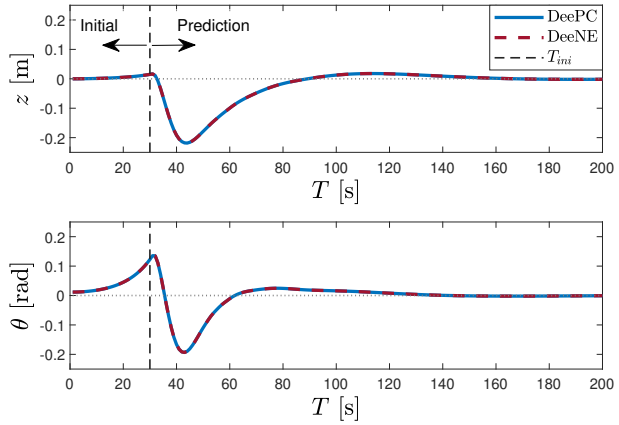

Figure 4.3: System Outputs for Cart-Inverted Pendulum.

Table 4.1: Comparison of Performance and Computational Cost for the DeePC with Various s.

DeePC	Performance	Time (per loop)
s = 0	22.3650	0.1419 ms
s = 5	22.1485	0.0284 ms
s = 10	22.7375	0.0159 ms

4.4.2 KINOVA Gen3

In order to evaluate the performance of the developed DeeNE, we prove its efficacy on a 7-DoF robotic manipulator. We learn and control KINOVA Gen3, which is a light weight 7-DoF arm robot. According to its specifications and for the sake of safety, we consider the minimum and

Table 4.2: Comparison of Performance and Computational Cost for both DeePC and DeeNE.

Controller	Performance	Time (per loop)
DeePC (s = 0)	22.3650	0.1419 ms
DeePC (s = 5)	22.1485	0.0284 ms
DeeNE (s = 0)	24.0375	0.0551 ms
DeeNE (s = 5)	23.5335	0.0102 ms

maximum values for the joint angular velocities (control inputs) as $[-\pi/6, \pi/6]rad/s$. Limitations on the Cartesian position coverage are all $[-0.9, 0.9]m$. We give the joint angular velocities $u \in \mathbb{R}^7$ to the arm robot and measure the pose of the robot $y \in \mathbb{R}^6$ which indicates the 3D position and the 3D orientation of the end-effector. However, to avoid a discontinuous behavior in the orientation part, we transfer the 3D orientation to 4D orientation using Quaternions. The protocol of data collection is as follows. We have collected data from the 7-DoF arm robot for 50 trajectories with $T_i = 100$ data points on each trajectory and the sampling time $T_s = 0.1s$. It is worth noting that since we are generating the Hankel matrix using multiple signal trajectories, called mosaic-Hankel matrix (a Hankel matrix with discontinuous signal trajectories), the number of data points on each trajectory must be greater than the depth of the Hankel matrix, i.e. $T_i > T_{ini} + N$ [54]. For each trajectory, the initial joint angles and the inputs are chosen randomly according to a uniform probability distribution. Due to the set up condition in the lab (desk structure, wall position, etc.), we had to stop the robot if it was close to hit an object, ignore that trajectory, and continue the data collection with another initial position and/or input values.

Details of DeePC are as follows. The reference trajectory $r(k) \in \mathbb{R}^7$ represents the desired values for the pose of the robot. According to the quadratic cost function, the matrices $Q = 5 \times 10^4 \times I_{pN}$ and $R = 1 \times 10^2 \times I_{mN}$ are considered to penalize the tracking error and control input amplitude, respectively. The slack variables $\lambda_y, \lambda_u = 5 \times 10^5$ are used to make sure the feasibility of the optimal control problem. The regularization parameter $\lambda_g = 5 \times 10^2$ avoids the overfitting issue due to the collected noisy data. Finally, the initial trajectory and the prediction lengths are $T_{ini} = 35$ and $N = 20$, respectively. Since we have $u \in \mathbb{R}^7$ and $y \in \mathbb{R}^7$, the dimension of the

mosaic-Hankel matrix is $\mathbb{H}(u^d, y^d) \in \mathbb{R}^{770\times2300}$ causing high computational cost for applying a real-time DeePC on the 7-DoF arm robot. For DeePC, we use the DeePC policy, apply the first s optimal control input $u(k : k + s)$ to the 7-DoF arm robot, measure the pose of the robot, and update the initial trajectory w_{ini} and the reference trajectory r for the next step (See Algorithm 2 in [27]). For an initial pose of the arm robot, we generate the first initial trajectory (u_{ini}, y_{ini}) using random control inputs, i.e. $(u(0 : 34), y(0 : 34))$. For a tracking performance index, we use Root Mean Square Error (RMSE) between the desired reference trajectory and the pose of the arm robot over the entire trajectory.

In this part, we compare the performance of the proposed DeeNE framework with DeePC such that we evaluate the tracking performance and computational time for different open-loop control scenarios s. For this part, we use the forward kinematics model of the 7-DoF arm robot to evaluate the performance of the control schemes. The reference trajectory $r(k)$ is consider as a sinusoidal trajectory with 300 data points for the pose of the end-effector. For the desired reference trajectory $r(k)$, DeePC and DeeNE must accomplish tracking control task. For this case, we use DeeNE policy to correct the open-loop DeePC solution at each step while we reduce the computational time. The tracking performance and the computational time are studied for DeePC and DeeNE frameworks under different open-loop control s. Fig. 4.4 compares the control input for the open-loop control scenario $s = 20$, which illustrates taht how DeeNE corrects the DeePC policy. Figs. 4.5 and 4.6 indicate the position and orientation tracking performance, respectively, where one can see that DeePC does not track the reference trajectory for $s = 20$; however, DeeNE tracks the reference very well. From Table 4.3, we can compare the tracking performance and the computational time indices for both control algorithms under $s = 0$, $s = 10$, and $s = 20$. One can see that both controllers perform similar on the tracking performance for $s = 0$, but DeeNE provides lower computational time. However, as we increase the open-loop part of the controller, i.e., s, the performance of DeePC goes down since it predicts the behavior of the system using the last available initial and reference trajectories. On the other hand, DeeNE takes a feedback from

the system and update the initial and reference trajectories at each time step, which corrects the DeePC predictions. Consequently, one can see that DeeNE enables us to have both high-precision tracking performance and faster motion speed for the 7-DoF arm robot.

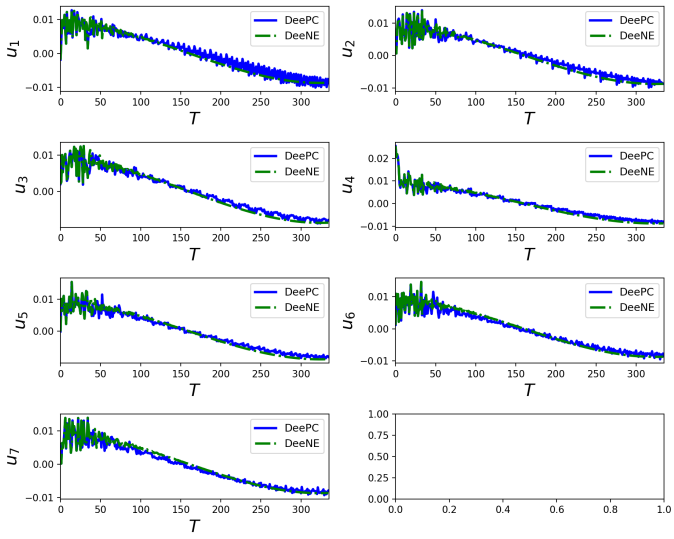

Figure 4.4: Control Input for 7-DoF Arm Robot (Simulation).

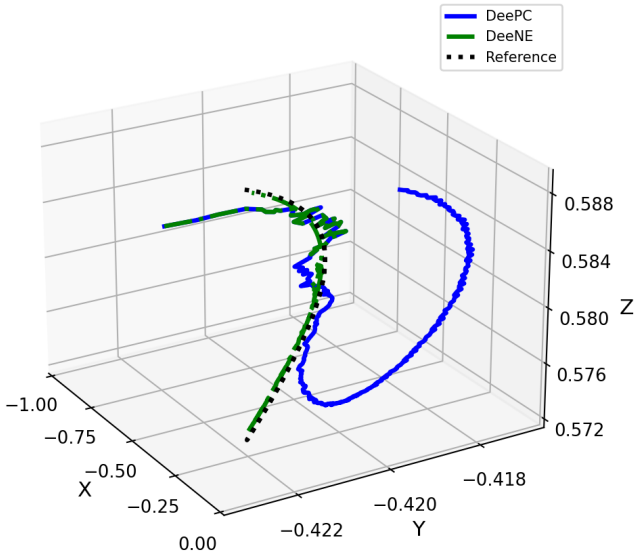

Figure 4.5: Position Tracking for 7-DoF Arm Robot (Simulation).

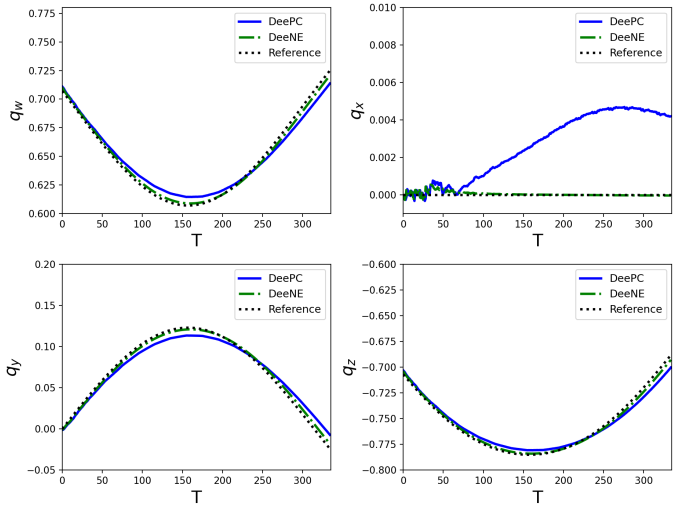

Figure 4.6: Orientation Tracking for 7-DoF Arm Robot (Simulation).

Table 4.3: Comparison of Performance and Computational Time for DeePC and DeeNE with Different Open-Loop Control Scenarios.

Controller	RMSE	Time (per loop)
DeePC (s = 0)	0.0023	0.2002 s
DeePC (s = 10)	0.0048	0.0204 s
DeePC (s = 20)	0.0051	0.0114 s
DeeNE (s = 0)	0.0024	0.0303 s
DeeNE (s = 10)	0.0027	0.0039 s
DeeNE (s = 20)	0.0031	0.0025 s

4.5 Experimental Verifications

In this part, we apply both controllers on the real 7-DoF arm robot for a closed-loop control scenario (i.e., $s = 0$) to make sure the safety and stability of the robot under DeePC. For the position of the end-effector, we consider the first part of the reference trajectory as the abbreviation of Michigan State University as MSU including 1000 data points. For the orientation of the end-effector, we consider the second part of the reference trajectory as 0.5 degree with 1000 data points. For the desired reference trajectory $r(k)$, DeePC must simultaneously accomplish tracking control and setpoint control tasks for the position and orientation of the end-effector, respectively. For this case, we use DeeNE policy to avoid solving the DeePC problem at each time step and reduce

the computational time, which provides a fast motion speed for the robot. Fig. 4.7 compares the control inputs generated by both control algorithms, which illustrates the well-performance of DeeNE on the approximation of the DeePC policy. Figs. 4.8 and 4.9 indicate the position and orientation tracking performance, respectively, where one can see that both controllers tracks the reference trajectory very well. From Table 4.4, we can compare the tracking performance and the computational time indices for both control algorithms, which perform similar on the tracking performance, but DeeNE provides lower computational time. It is worth noting that since the computational time of DeePC, i.e. $0.2s$, is higher than the sampling time, i.e. $0.1s$, we apply DeePC on the arm robot for the first $0.1s$, stop the robot for the second $0.1s$ to receive the updated response of DeePC, and then repeat the process using the updated control input. Consequently, it is obvious that DeeNE enables us to have both high-precision tracking performance and faster motion speed for the 7-DoF arm robot. Moreover, we cannot stop the system until receiving the updated response of DeePC for safety-critical scenarios since it may cause an accident for the robots/autonomous vehicles.

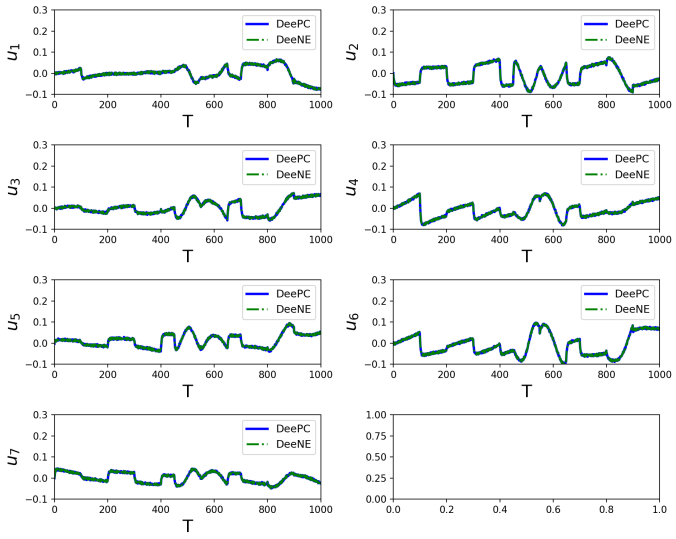

Figure 4.7: Control Input for 7-DoF Arm Robot (Experiment).

We next verify the performance of the control algorithms under the safety constraints, which the arm robot must avoid unsafe regions and dynamic obstacles. In the second experiment, the

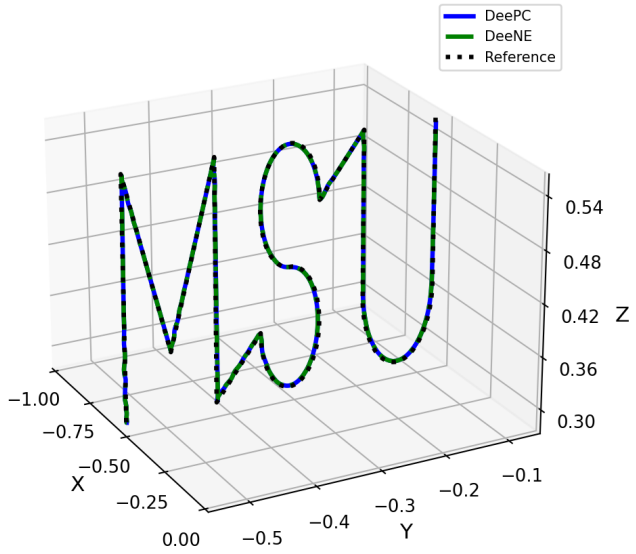

Figure 4.8: Position Tracking for 7-DoF Arm Robot (Experiment).

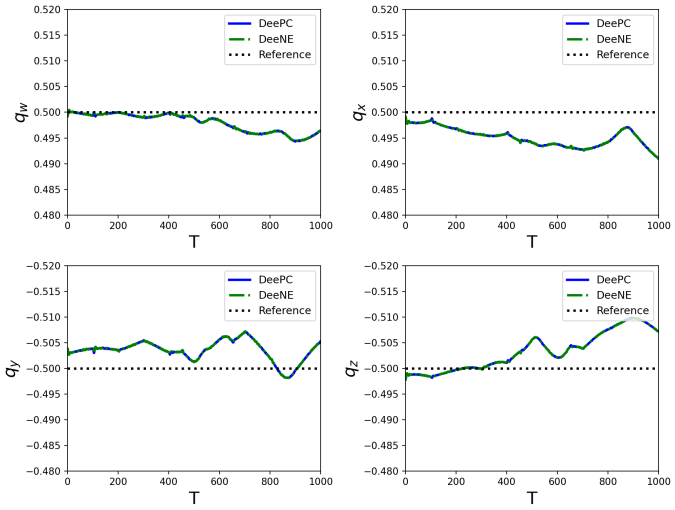

Figure 4.9: Orientation Tracking for 7-DoF Arm Robot (Experiment).

Table 4.4: Comparison of Performance and Computational Time for DeePC and DeeNE.

Controller	RMSE	Time (per loop)
DeePC	0.0142	0.2005 s
DeeNE	0.0143	0.0307 s

robot must track the same reference trajectory MSU; however, we consider an unsafe region on the S part. This case addresses the control tasks that the reference trajectory is obtained offline using path planning or by the operator, but the controller must avoid unsafe regions through the given reference trajectory due to the dynamic obstacles. For simplicity, we have considered a fixed unsafe region on the S part of MSU; thus, both DeePC and DeeNE can satisfy the safety constraints and track the reference trajectory well as shown in Figs. 4.10-4.12. However, like the previous task, Table 4.5 shows that the computational time of DeePC is higher than the sampling time, which may cause an accident for the dynamic obstacles since the robot updated the control input after receiving the response of DeePC. On the other hand, DeeNE quickly computes the optimal control input for the updated initial and reference trajectories; therefore, it can avoid the dynamic obstacles well. A video of the experiments performed can be found at the following link https://www.youtube.com/watch?v=BlKTUgkAMVo.

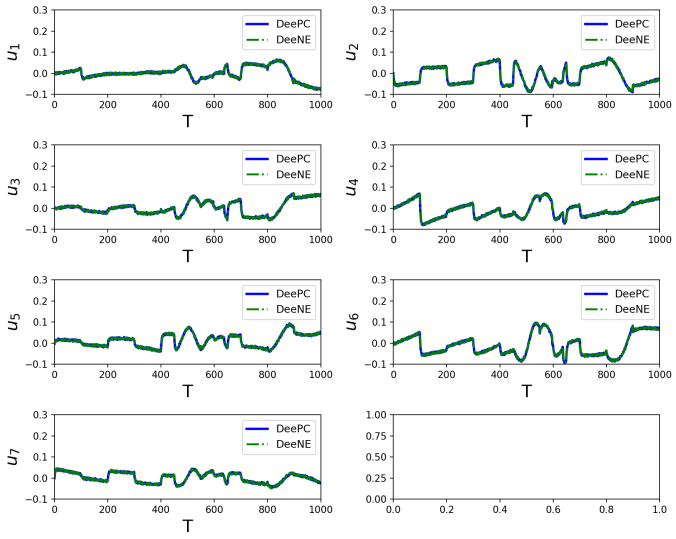

Figure 4.10: Safe Control Input for 7-DoF Arm Robot (Experiment).

4.6 Chapter Summary

In this chapter, we proposed a computationally efficient method to implement the data-driven optimal controllers (e.g. DeePC) that include nonlinearities in real time. The DeeNE algorithm

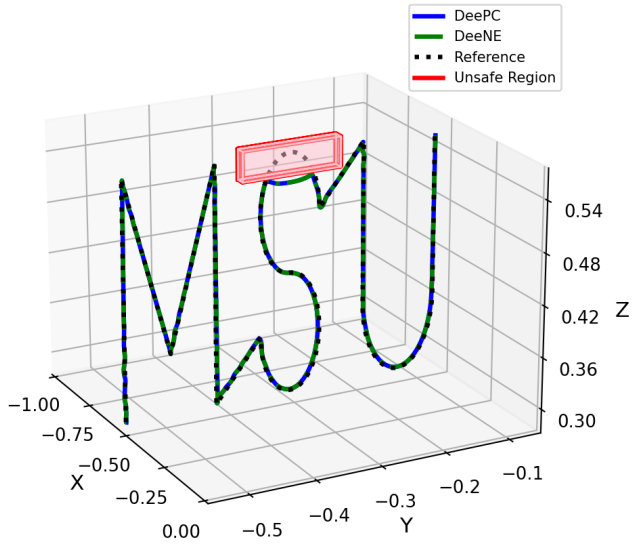

Figure 4.11: Safe Position Tracking for 7-DoF Arm Robot (Experiment).

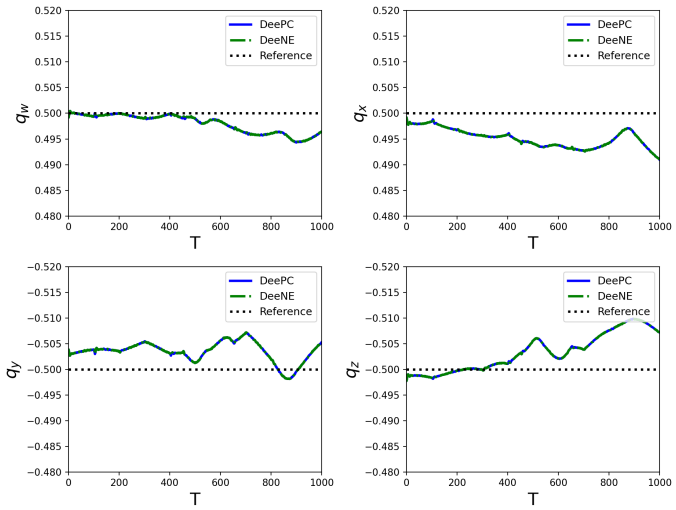

Figure 4.12: Safe Orientation Tracking for 7-DoF Arm Robot (Experiment).

Table 4.5: Comparison of Performance and Computational Cost for DeePC and DeeNE with Safety Guarantees.

Controller	RMSE	Time (per loop)
DeePC	0.0148	0.2008 s
DeeNE	0.0149	0.0309 s

was developed to approximate the DeePC policy in the presence of input/output and reference trajectories perturbations. The developed DeeNE was based on the second-order variation of the original DeePC problem such that the computational load of the DeeNE grows linearly for the optimization horizon. This control approach alleviates the online computational burden and extend the applicability of data-driven optimal controllers. Simulations of the cart inverted pendulum system demonstrated the DeeNE's technological advances over the DeePC. Moreover, simulation and experimental verifications on the 7-DoF arm robot demonstrated the performance of the DeeNE compared to the DeePC for a more complex system.

CHAPTER 5

ADAPTIVE DATA-ENABLED PREDICTIVE CONTROL

In this chapter, we focus on developing an adaptive data-driven optimal control for time-varying systems. DeePC uses pre-collected input/output (I/O) data to construct Hankel matrices for online predictive control. However, in systems with evolving dynamics, incorporating real-time data into the DeePC framework becomes crucial to enhance control performance. We propose an adaptive DeePC framework for time-varying systems, enabling the algorithm to update the Hankel matrix online by adding real-time informative signals. By exploiting the minimum non-zero singular value of the Hankel matrix, the developed online DeePC selectively integrates informative data and effectively captures evolving system dynamics. Additionally, a numerical singular value decomposition technique is introduced to reduce the computational complexity for updating a reduced-order Hankel matrix. Simulation results on two cases, a linear time-varying system and the vehicle anti-rollover control, demonstrate the effectiveness of the online reduced-order DeePC.[1]

5.1 Background

The Fundamental Lemma only holds for deterministic linear time-invariant (LTI) systems [55, 56]. For other systems such as nonlinear systems, stochastic systems, and time-varying systems, the rank condition on the Hankel matrix is not sufficient to accurately determine the trajectory subspace, which may lead to poor performance in the data-driven control. Several techniques employing slack variables and regularization methods have been proposed to address these limitations and enhance performance for the nonlinear systems and the stochastic systems [27, 29, 57]. Moreover, a robust Fundamental Lemma has been proposed to ensure the persistently exciting (PE) input with sufficient order for the stochastic LTI systems [55]. However, these techniques are only effective in local regions captured by the pre-collected I/O data and cannot handle the new system dynamics that emerge online. Therefore, it is necessary to update the Hankel matrix online using real-time I/O

[1] The material of this chapter is from "Online Reduced-Order Data-Enabled Predictive Control," arXiv preprint arXiv:2407.16066, 2024 [54].

data to predict system behavior accurately.

For time-varying systems, i.e., systems with evolving dynamics, online DeePC [58, 59] is developed to continuously update the Hankel matrix using real-time data. In [58], old data is replaced with new data: the first column (the oldest data point) is discarded from the Hankel matrix, all columns are shifted back by one step, and the most recent real-time data is added as the last column. However, this method requires a PE control input in real-time, which is achieved by adding a suitable excitation, such as injecting noise into the control input during closed-loop operation. Injecting noise can deteriorate control performance and put unnecessary stress on actuators. To address this, [56] presents a discontinuous online DeePC method that replaces the PE requirement with a rank condition on mosaic-Hankel matrix (a Hankel matrix with discontinuous I/O trajectories) proposed by [60]. This approach requires only an offline PE input trajectory, which is produced based on the robust Fundamental Lemma [55], and updates the mosaic-Hankel matrix if the rank condition is satisfied. However, as mentioned before, a rank condition (or PE condition) is not sufficient to indicate informative data for non-deterministic LTI systems.

[61] proposes a continuous online DeePC by adding real-time I/O trajectories to the Hankel matrix, which removes the PE requirement for the real-time control input as the rank condition is always satisfied. However, continuously increasing the columns of the Hankel matrix leads to high memory and computational costs. To mitigate this, two numerical singular value decomposition (SVD) algorithms are used alternatively based on the rank of the Hankel matrix: when the rank of the Hankel matrix is less than the number of rows, the numerical SVD algorithm [62], which is designed based on the eigen-decomposition algorithm [63], is applied; and when the rank of the Hankel matrix is equal to the number of rows, the numerical SVD algorithm [64] is employed. The online DeePC scheme [61] faces three main limitations: (i) it requires adding all real-time trajectories to the Hankel matrix, leading to high computational cost and the inclusion of data without considering its informativeness; (ii) the numerical SVD algorithm [64], as mentioned in [63], is not fast and can be unstable; and (iii) the dimension of the online SVD-based DeePC can be

reduced to a minimum possible dimension. These limitations motivate us to propose a new online reduced-order DeePC. Our approach measures the informativeness of real-time data by observing the minimum non-zero singular value of the Hankel matrix and uses the most informative data discontinuously, which addresses challenge (i). By setting a well-tuned threshold on the minimum non-zero singular value, we are able to capture the new system dynamics that emerge online and update the Hankel matrix only with the most informative data. Moreover, to overcome challenges (ii) and (iii), we modify the numerical SVD algorithm [62], as this algorithm only works for low-rank modifications, and develop an online reduced-order DeePC with adaptive order. These modifications result in lower computational complexity for the online DeePC and may improve the control performance for some cases.

In this chapter, we develop an adaptive DeePC framework for the time-varying systems such that the Hankel matrix is updated using real-time informative data. Promising results are demonstrated by applying the developed adaptive controller to the linear time-varying system and the vehicle anti-rollover control. The outline of this chapter is as follows. The problem formulation and the preliminaries of the DeePC are provided in Section II. Section III presents the proposed adaptive DeePC for the time-varying systems. Section IV presents the simulation results. Finally, the conclusions are provided in Section V.

5.2 Problem Formulation

Consider the following unknown discrete-time system:

$$\begin{aligned} x(k+1) &= f(x(k), u(k)), \\ y(k) &= h(x(k), u(k)), \end{aligned} \tag{5.1}$$

where $k \in \mathbb{N}^+$ represents the time step, $x \in \mathbb{R}^n$ denotes the system state, $u \in \mathbb{R}^m$ indicates the control input, and $y \in \mathbb{R}^p$ stands for the system output. Moreover, $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ indicates the system dynamics with $f(0,0) = 0$, and $h : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^p$ denotes the output dynamics.

Assuming the system (5.1) is a linear time-invariant (LTI) system, i.e., $f(x(k), u(k)) =$

$Ax(k) + Bu(k)$ and $h(x(k), u(k)) = Cx(k) + Du(k)$, the Fundamental Lemma [26] provides a non-parametric representation to describe the system behavior.

Definition 4 (Hankel Matrix) *Given a signal $w(k) \in \mathbb{R}^q$, we denote by $w_{1:T}$ the restriction of $w(k)$ to the interval $[1, T]$, namely $w_{1:T} = [w^\top(1), w^\top(2), \cdots, w^\top(T)]^\top$. The Hankel Matrix of depth $K \leq T$ is defined as:*

$$\mathcal{H}_K(w_{1:T}) := \begin{bmatrix} w(1) & w(2) & \cdots & w(T-K+1) \\ w(2) & w(3) & \cdots & w(T-K+2) \\ \vdots & \vdots & \ddots & \vdots \\ w(K) & w(K+1) & \cdots & w(T) \end{bmatrix}. \tag{5.2}$$

Let $L := T - K + 1$, then we have $\mathcal{H}_K(w_{1:T}) \in \mathbb{R}^{qK \times L}$.

Definition 5 (Persistently Exciting) *The sequence $w_{1:T}$ is persistently exciting (PE) of order K if $\mathcal{H}_K(w_{1:T})$ has full row rank, i.e., $rank(\mathcal{H}_K(w_{1:T})) = qK$.*

Lemma 3 (Fundamental Lemma [26]) *Consider the system* (5.1) *as a controllable LTI system with a pre-collected input/output (I/O) sequence $(u^{\mathrm{d}}_{1:T}, y^{\mathrm{d}}_{1:T})$ of length T. Providing a PE input sequence $u^{\mathrm{d}}_{1:T}$ of order $K + n$, any length-K sequence $(u_{1:K}, y_{1:K})$ is an I/O trajectory of the LTI system if and only if we have*

$$\begin{bmatrix} u_{1:K} \\ y_{1:K} \end{bmatrix} = \begin{bmatrix} \mathcal{H}_K(u^{\mathrm{d}}_{1:T}) \\ \mathcal{H}_K(y^{\mathrm{d}}_{1:T}) \end{bmatrix} g, \tag{5.3}$$

for some real vector $g \in \mathbb{R}^L$.

Remark 11 (Rank of Hankel Matrix) *Considering $r := rank\left(\begin{bmatrix} \mathcal{H}_K(u^{\mathrm{d}}_{1:T}) \\ \mathcal{H}_K(y^{\mathrm{d}}_{1:T}) \end{bmatrix}\right)$, a persistently exciting control input sequence of order $K + n$ ensures that $mK + 1 \leq r \leq mK + n$. Furthermore, $r = mK + n$ if $K \geq l$, where $l \leq n$ is the observability index. See [65] for more details.*

The Fundamental Lemma shows that the Hankel matrix in (5.3) spans the vector space of all length-K signal trajectories that an LTI system can produce, provided that the collected input

sequence is PE of order $K+n$ and the underlying system is controllable. However, the Fundamental Lemma requires a long continuous signal trajectory to construct the Hankel matrix. [60] extends the Fundamental Lemma to accommodate multiple short signal trajectories, which we refer to as discontinuous Fundamental Lemma in this paper. This extended Lemma is developed by using a general data structure called mosaic-Hankel matrix, which incorporates a dataset consisting of multiple short discontinuous signal trajectories.

Definition 6 (Mosaic-Hankel Matrix) *Let $W = \{w^1_{1:T_1}, \cdots, w^s_{1:T_s}\}$ be the set of s discontinuous sequences with length of $T_1, \cdots, T_s$. The mosaic-Hankel matrix of depth $K \leq \min(T_1, \cdots, T_s)$ is defined as:*

$$\mathcal{M}_K(W) = [\mathcal{H}_K(w^1_{1:T_1}), \mathcal{H}_K(w^2_{1:T_2}), \cdots, \mathcal{H}_K(w^s_{1:T_s})]. \tag{5.4}$$

Let $T = \sum_{i=1}^{s} T_i$ and $L = T - s(K-1)$, then we have $\mathcal{M}_K(W) \in \mathbb{R}^{qK \times L}$.

Lemma 4 (Discontinuous Fundamental Lemma [60]) *Consider the system* (5.1) *as a controllable LTI system with a pre-collected I/O sequence $U^{\mathrm{d}} = \{u^{\mathrm{d},1}_{1:T_1}, \cdots, u^{\mathrm{d},s}_{1:T_s}\}$, $Y^{\mathrm{d}} = \{y^{\mathrm{d},1}_{1:T_1}, \cdots, y^{\mathrm{d},s}_{1:T_s}\}$ of length T which consists of s I/0 sequences of length $T_1, \cdots, T_s$. Providing an input sequence U^{d} with*

$$r := \operatorname{rank}\left(\begin{bmatrix} \mathcal{M}_K(U^{\mathrm{d}}) \\ \mathcal{M}_K(Y^{\mathrm{d}}) \end{bmatrix}\right) = mK + n, \tag{5.5}$$

any length-K sequence $(u_{1:K}, y_{1:K})$ is an I/O trajectory of the LTI system if and only if we have

$$\begin{bmatrix} u_{1:K} \\ y_{1:K} \end{bmatrix} = \begin{bmatrix} \mathcal{M}_K(U^{\mathrm{d}}) \\ \mathcal{M}_K(Y^{\mathrm{d}}) \end{bmatrix} g, \tag{5.6}$$

for some real vector $g \in \mathbb{R}^L$.

It is worth noting that the Hankel matrix (5.2) represents a special case of the mosaic-Hankel matrix (5.4) with $s = 1$. The general form (5.4) also describes other special forms, such as Page matrix or Trajectory matrix [65]. The main advantages of Lemma 4 compared to Lemma 3 are: i) it uses multiple short discontinuous trajectories instead of one long continuous trajectory, and ii) it

replaces the PE condition on the input sequence with the rank condition (5.5) on the I/O matrix. (5.5) is referred as generalized PE condition [60].

Both (5.3) and (5.6) can be regarded as the non-parametric representation for system (5.1). Let T_{ini}, $N \in \mathbb{Z}$, and $K = T_{\text{ini}} + N$. The Hankel matrices $\mathcal{H}_K(u^{\text{d}}_{1:T})$, $\mathcal{H}_K(y^{\text{d}}_{1:T})$ (or mosaic-Hankel matrices $\mathcal{M}_K(U^{\text{d}})$, $\mathcal{M}_K(Y^{\text{d}})$) are divided into two parts (i.e., "past data" of length T_{ini} and "future data" of length N):

$$\begin{bmatrix} U_P \\ U_F \end{bmatrix} = \mathcal{H}_K(u^{\text{d}}_{1:T}), \quad \begin{bmatrix} Y_P \\ Y_F \end{bmatrix} = \mathcal{H}_K(y^{\text{d}}_{1:T}), \tag{5.7}$$

or

$$\begin{bmatrix} U_P \\ U_F \end{bmatrix} = \mathcal{M}_K(U^{\text{d}}), \quad \begin{bmatrix} Y_P \\ Y_F \end{bmatrix} = \mathcal{M}_K(Y^{\text{d}}), \tag{5.8}$$

where U_{p} and U_{f} denote the first T_{ini} block rows and the last N block rows of $\mathcal{H}_K(u^{\text{d}}_{1:T})$ (or $\mathcal{M}_K(U^{\text{d}})$), respectively (similarly for Y_{p} and Y_{f}). The data-enabled predictive control (DeePC) is formulated as [27, 36]

$$\begin{aligned} (y^*, u^*, \sigma_y^*, \sigma_u^*, g^*) = & \underset{y,u,\sigma_y,\sigma_u,g}{\arg\min} \; J(y, u, \sigma_y, \sigma_u, g) \\ \text{s.t.} \quad & \begin{bmatrix} U_P \\ U_F \\ Y_P \\ Y_F \end{bmatrix} g = \begin{bmatrix} u_{\text{ini}} \\ u \\ y_{\text{ini}} \\ y \end{bmatrix} + \begin{bmatrix} \sigma_u \\ 0 \\ \sigma_y \\ 0 \end{bmatrix}, \\ & u \in \mathcal{U}, y \in \mathcal{Y}. \end{aligned} \tag{5.9}$$

In (5.9), $J(y, u, \sigma_y, \sigma_u, g)$ represents the cost function. $u_{\text{ini}} = u_{k-T_{\text{ini}}:k-1}$ is the control input sequence within a past time horizon of length T_{ini}, and $u = u_{k:k+N-1}$ is the control input sequence within a prediction horizon of length N (similarly for y_{ini} and y). $\mathcal{U}$, $\mathcal{Y}$ represent the input and output constraints, respectively. $\sigma_u \in \mathbb{R}^{mT_{\text{ini}}}$, $\sigma_y \in \mathbb{R}^{pT_{\text{ini}}}$ stand for auxiliary variables.

Both Lemma 3 and Lemma 4 are only valid for deterministic LTI systems. [59] and [58] respectively show that Lemma 3 can be extended to the nonlinear systems and the time-varying

systems by continuously updating the Hankel matrix, which require a real-time PE input sequence. Moreover, [56] removes the real-time PE requirement for the time-varying systems by using Lemma 4. However, the proposed rank condition (or PE condition) is not sufficient to ensure the informativeness of the data for non-deterministic LTI systems. While the rank condition may hold for the non-deterministic LTI systems, it can still lead to large prediction errors and poor control performance. In this paper, we propose an online DeePC framework for the time-varying systems to improve the rank condition and present a valid indicator for evaluating data informativeness.

5.3 Main Result

In this section, based on the data informativeness of the mosaic-Hankel matrix, we propose an online DeePC framework for the time-varying systems. The data informativeness is evaluated with the minimum non-zero singular value of the mosaic-Hankel matrix and is enhanced by adding the most informative signals to the matrix. It should be mentioned that the rank condition is always satisfied in real time since we add signal trajectories as additional columns. Moreover, we develop an online reduced-order DeePC using a numerical singular value decomposition (SVD) to reduce the computational cost of the control scheme.

5.3.1 Adaptive Data-Enabled Predictive Control

For the deterministic LTI systems, Lemma 3 and Lemma 4 are valid only if the data is sufficiently informative. However, for the non-deterministic LTI systems, the collected data may result in a full-rank mosaic-Hankel matrix (or Hankel matrix) but cannot ensure accurate prediction of the system behavior. To address this issue, [55] proposes a quantitative measure of PE for the input trajectory based on the minimum non-zero singular value of the Hankel matrix, enhancing the robustness of the Fundamental Lemma against uncertainties. However, this approach is only effective locally around collected I/O data and leads to poor performance for the time-varying systems. Therefore, we focus on improving the performance of the discontinuous Fundamental Lemma for the time-varying systems. Singular values of a matrix are defined as $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq \cdots \geq \sigma_{\text{end}} \geq 0$,

where σ_1 and σ_{end} are also referred as $\sigma_{\max}$ and $\sigma_{\min}$, respectively. The singular value σ_r corresponds to the rank of the matrix and is the minimum non-zero singular value, i.e., all singular values from σ_{r+1} to $\sigma_{\min}$ are zero.

[55] shows that the prediction error of the Fundamental Lemma can be arbitrarily large, even if the rank condition on the Hankel matrix is met. The important factor is the data informativeness, which is represented by the minimum non-zero singular value σ_r. Therefore, we can update the mosaic-Hankel matrix with real-time signal trajectories if σ_r increases. If the real-time signal trajectory contains new information relevant to describing the system behavior (5.1), it increases σ_r; otherwise, σ_r decreases if the real-time signal trajectory lacks new information. Thus, we can set a threshold σ_{thr} for σ_r to ensure that only the most informative data is added provided $\sigma_r \geq \sigma_{\text{thr}}$. To achieve better prediction for the time-varying systems using the Fundamental Lemma, we formulate an online DeePC as follows:

$$
\begin{aligned}
(y^*, u^*, \sigma_y^*, \sigma_u^*, g^*) = & \underset{y,u,\sigma_y,\sigma_u,g}{\arg\min} \; J(y, u, \sigma_y, \sigma_u, g) \\
\text{s.t.} \quad & \begin{bmatrix} U_P^o \\ U_F^o \\ Y_P^o \\ Y_F^o \end{bmatrix} g = \begin{bmatrix} u_{\text{ini}} \\ u \\ y_{\text{ini}} \\ y \end{bmatrix} + \begin{bmatrix} \sigma_u \\ 0 \\ \sigma_y \\ 0 \end{bmatrix}, \\
& u \in \mathcal{U}, y \in \mathcal{Y},
\end{aligned}
\tag{5.10}
$$

where the matrices U_P^o, U_F^o, Y_P^o, and Y_F^o are updated online. Specifically, denote W^{d} as the combined data set of U^{d} and Y^{d}, i.e., $W^{\text{d}} = \{U^{\text{d}}, Y^{\text{d}}\}$. The corresponding mosaic-Hankel matrix $\mathcal{M}_K(W^{\text{d}})$ is defined as:

$$
\mathcal{M}_K(W^{\text{d}}) := \begin{bmatrix} \mathcal{M}_K(U^{\text{d}}) \\ \mathcal{M}_K(Y^{\text{d}}) \end{bmatrix}. \tag{5.11}
$$

When $k < K$, the matrices U_P^o, U_F^o, Y_P^o, and Y_F^o are initialized with $\mathcal{M}_K(W^{\text{d}})$ (i.e., U^{d} and Y^{d} as shown in (5.8)). When $k \geq K$, the most recent real-time I/O sequence $w_{k-K+1:k} :=$

$\left[u_{k-K+1:k}^\top, y_{k-K+1:k}^\top\right]^\top$ is first used to update the matrix

$$\mathcal{M}_K(W^{\mathrm{d}}) \leftarrow \left[\mathcal{M}_K(W^{\mathrm{d}}), w_{k-K+1:k}\right].$$

Then, the rank and the minimum non-zero singular value of $\mathcal{M}_K(W^{\mathrm{d}})$ are calculated. If the minimum non-zero singular value of $\mathcal{M}_K(W^{\mathrm{d}})$ is larger than the threshold σ_{thr}, i.e., $\sigma_r \geq \sigma_{\mathrm{thr}}$, then the matrices U_P^o, U_F^o, Y_P^o, and Y_F^o are updated with $\mathcal{M}_K(W^{\mathrm{d}})$; otherwise, the sequence $w_{k-K+1:k}$ is removed from $\mathcal{M}_K(W^{\mathrm{d}})$.

5.3.2 Adaptive Reduced-Order Data-Enabled Predictive Control

SVD techniques are effective in reducing computational complexity for data-driven control methods [66, 61]. In this section, we incorporate a new numerical SVD algorithm into the online DeePC framework such that the reduced-order mosaic-Hankel matrix and corresponding singular values can be updated efficiently.

Considering $r = \mathrm{rank}(\mathcal{M}_K(W^{\mathrm{d}}))$, one can formulate the SVD of the mosaic-Hankel matrix $\mathcal{M}_K(W^{\mathrm{d}})$ as follows:

$$\mathcal{M}_K(W^{\mathrm{d}}) = U\Sigma V^\top = [U_r \;\; U_{qK-r}] \begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix} [V_r \;\; V_{L-r}]^\top, \tag{5.12}$$

where $q = m + p$, $\Sigma \in \mathbb{R}^{qK \times L}$ is the singular matrix, and $U \in \mathbb{R}^{qK \times qK}$ and $V \in \mathbb{R}^{L \times L}$ are left and right singular vectors, respectively, such that $UU^\top = U^\top U = I_{qK}$ and $VV^\top = V^\top V = I_L$. Moreover, Σ_r contains the top r non-zero singular values, $U_r \in \mathbb{R}^{qK \times r}$, $U_{qK-r} \in \mathbb{R}^{qK \times (qK-r)}$, and $V_r \in \mathbb{R}^{L \times r}$, $V_{L-r} \in \mathbb{R}^{L \times (L-r)}$. Therefore, one can write

$$\mathcal{M}_K(W^{\mathrm{d}})g = U_r \Sigma_r V_r^\top g = \mathcal{M}'_K(W^{\mathrm{d}})g', \tag{5.13}$$

where $\mathcal{M}'_K(W^{\mathrm{d}}) = U_r\Sigma_r \in \mathbb{R}^{qK \times r}$ and $g' = V_r^\top g \in \mathbb{R}^r$. If the pre-collected data is sufficient rich, then we have $mK + n \leq r \leq \min(qK, L)$. Thus, one can approximate (5.13) using a rank order $mK + n \leq r_a \leq r$, as follows:

$$\mathcal{M}_K(W^{\mathrm{d}})g \approx U_{ra}\Sigma_{ra}V_{ra}^\top g = \mathcal{M}''_K(W^{\mathrm{d}})g'', \tag{5.14}$$

where $\mathcal{M}''_K(W^{\mathrm{d}}) = U_{r_a}\Sigma_{r_a} \in \mathbb{R}^{qK\times r_a}$ and $g'' = V_{r_a}^\top g \in \mathbb{R}^{r_a}$.

Now, one can formulate an online reduced-order DeePC as follows:

$$
\begin{aligned}
(y^*, u^*, \sigma_y^*, \sigma_u^*, g''^*) = & \underset{y,u,\sigma_y,\sigma_u,g''}{\arg\min} \; J(y, u, \sigma_y, \sigma_u, g'') \\
\text{s.t.} \quad & \begin{bmatrix} U_P^{o''} \\ U_F^{o''} \\ Y_P^{o''} \\ Y_F^{o''} \end{bmatrix} g'' = \begin{bmatrix} u_{\mathrm{ini}} \\ u \\ y_{\mathrm{ini}} \\ y \end{bmatrix} + \begin{bmatrix} \sigma_u \\ 0 \\ \sigma_y \\ 0 \end{bmatrix}, \\
& u \in \mathcal{U}, y \in \mathcal{Y},
\end{aligned}
\tag{5.15}
$$

where the matrices $U_P^{o''}$, $U_F^{o''}$, $Y_P^{o''}$, and $Y_F^{o''}$ are updated online based on $\mathcal{M}''_K(W^{\mathrm{d}})$ under an adaptive order r_a such that $\sigma_{r_a} \geq \sigma_{\mathrm{thr}}$. It should be mentioned that we use SVD to update $\mathcal{M}''_K(W^{\mathrm{d}})$ for $\mathcal{M}_K(W^{\mathrm{d}}) \leftarrow \left[\mathcal{M}_K(W^{\mathrm{d}}), w_{k-K+1:k}\right]$. Moreover, the adaptive order $mK+n \leq r_a \leq r$, which is based on the threshold singular value σ_{thr}, allows an adaptive dimension for the reduced-order mosaic-Hankel matrix $\mathcal{M}''_K(W^{\mathrm{d}})$ regarding the data informativity. Indeed, the dimension of $\mathcal{M}''_K(W^{\mathrm{d}})$, i.e., r_a, is changed adaptively based on $\sigma_{r_a} \geq \sigma_{\mathrm{thr}}$.

For both proposed online DeePC frameworks, calculating the SVD at each time step is not computationally efficient for real-time control. Therefore, inspired by [62], we propose a numerical algorithm to compute the SVD of $\left[\mathcal{M}_K(W^{\mathrm{d}}), w_{k-K+1:k}\right]$ by taking advantage of our knowledge of the SVD of $\mathcal{M}_K(W^{\mathrm{d}})$, which reduces the computational time of the proposed online DeePC.

When $\mathrm{rank}(\mathcal{M}_K(W^{\mathrm{d}})) < \mathrm{rows}(\mathcal{M}_K(W^{\mathrm{d}}))$, one can express $\left[\mathcal{M}_K(W^{\mathrm{d}}), w_{k-K+1:k}\right]$ as $X + AB^\top$, where $X = [\mathcal{M}_K(W^{\mathrm{d}}), 0]$, $A = w_{k-K+1:k}$, and $B = [0, \cdots, 0, 1]^\top$. Therefore, one has

$$
X + AB^\top = [U_r \;\; A] \begin{bmatrix} \Sigma_r & 0 \\ 0 & I \end{bmatrix} [V_x \;\; B]^\top. \tag{5.16}
$$

where $V_x = [V_r^\top \;\; 0]^\top$. Let P be an orthogonal basis of the column space of $(I - U_r U_r^\top)A$, i.e., the

component of A that is orthogonal to U_r, and set $R_A = P^\top (I - U_r U_r^\top) A$. Now, one can write

$$[U_r \;\; A] = [U_r \;\; P] \begin{bmatrix} I & U_r^\top A \\ 0 & R_A \end{bmatrix}, \tag{5.17}$$

where similar to a QR decomposition, R_A needs not be upper-triangular or square. Similarly, let $R_B = Q^\top (I - V_x V_x^\top) B$, where Q is the component of B that is orthogonal to V_x. Thus, one has

$$[V_x \;\; B] = [V_x \;\; Q] \begin{bmatrix} I & V_x^\top B \\ 0 & R_B \end{bmatrix}. \tag{5.18}$$

Substituting (5.17) and (5.18) into (5.16), we have

$$\begin{aligned} X + AB^\top &= [U_r \;\; P] S [V_x \;\; Q]^\top, \\ S &= \begin{bmatrix} I & U_r^\top A \\ 0 & R_A \end{bmatrix} \begin{bmatrix} \Sigma_r & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} I & V_x^\top B \\ 0 & R_B \end{bmatrix}^\top, \end{aligned} \tag{5.19}$$

where one can write S as:

$$S = \begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} U_r^\top A \\ R_A \end{bmatrix} \begin{bmatrix} V_x^\top B \\ R_B \end{bmatrix}^\top. \tag{5.20}$$

Using [63], diagonalizing $S = U_S \Sigma_S V_S^\top$ gives rotations U_S and V_S of the extended subspaces $[U_r \;\; P]$ and $[V_x \;\; Q]$ such that

$$X + AB^\top = ([U_r \;\; P] U_S) \Sigma_S ([V_x \;\; Q] V_S)^\top \tag{5.21}$$

is the desired SVD.

Remark 12 (Rank-1 Modifications) *For the proposed numerical SVD, we are not limited to only add one signal to the mosaic-Hankel matrix. The above formulations work for adding more signals at the same time by defining the correct matrices A and B. However, in our algorithm, we focus on adding one signal to the matrix at each time step and calculate the SVD of the new matrix based on*

the SVD of the original matrix, which is called Rank-1 modification in the numerical SVD. For Rank-1 modification, we define $P = \|(I-U_rU_r^T)A\|^{-1}(I-U_rU_r^T)A$ *and* $Q = \|(I-V_xV_x^T)B\|^{-1}(I-V_xV_x^T)B$, *which yield* $S = \begin{bmatrix} \Sigma_r & U_r^TA \\ 0 & R \end{bmatrix}$ *since* $V_x^TB = [V_r^T \;\; 0][0, ..., 0, 1]^T = 0$.

When $\text{rank}(\mathcal{M}_K(W^{\text{d}})) = \text{rows}(\mathcal{M}_K(W^{\text{d}}))$, one needs rewrite the SVD of the mosaic-Hankel matrix $\left[\mathcal{M}_K(W^{\text{d}}), w_{k-K+1:k}\right]$ as

$$X + AB^\top = U_r[\Sigma_r \;\; U_r^\top A][V_x \;\; B]^\top \tag{5.22}$$

Therefore, we only need to provide an orthogonal matrix for $[V_x \;\; B]$, which yields $S = \left[\Sigma_r \;\; 0\right] + U_r^\top A \begin{bmatrix} V_x^\top B \\ R_B \end{bmatrix}^\top$, and $X + AB^\top = (U_rU_S)\Sigma_S([V_x \;\; Q]V_S)^\top$ is the desired SVD.

Remark 13 (Comparison) *Compared to [61], the proposed online reduced-order DeePC measures data informativity of real-time signals by observing the minimum non-zero singular value of the mosaic-Hankel matrix, selectively using the most informative signals instead of adding all real-time trajectories. Moreover, the dimension of the online reduced-order DeePC starts from* $r_a = mK + n$ *instead of the rank, which leads to the minimum possible dimension for the mosaic-Hankel matrix. We also refine the numerical SVD algorithm, addressing its speed and stability issues as noted in [63]. These three contributions improve the control performance and reduce computational complexity.*

5.4 Simulation Results

5.4.1 Linear Time-Varying System

In this subsection, we demonstrate the performance of the proposed online reduced-order DeePC framework via a simulation example on a linear time-varying (LTV) system, described by:

$$\begin{aligned} x(k+1) &= A(k)x(k) + B(k)u(k) + d_p(k), \\ y(k) &= Cx(k) + d_m(k), \end{aligned} \tag{5.23}$$

where d_p and d_m represent process and measurement noises, respectively. The matrices $A(k)$, $B(k)$, and C are constructed as

$$A(k) = A^o + \lambda(k) \begin{bmatrix} 0.01 & 0 & 0.001 & 0 \\ 0 & 0.01 & 0 & 0.001 \\ 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix},$$

$$B(k) = B^o + \lambda(k) \begin{bmatrix} 0.001 & 0.0001 \\ 0.0001 & 0.001 \\ 0 & 0.001 \\ 0.001 & 0 \end{bmatrix},$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix},$$

where $\lambda(k)$ is a time-varying parameter, and A^o and B^o are set as follows:

$$A^o = \begin{bmatrix} 0.921 & 0 & 0.041 & 0 \\ 0 & 0.918 & 0 & 0.033 \\ 0 & 0 & 0.924 & 0 \\ 0 & 0 & 0 & 0.937 \end{bmatrix},$$

$$B^o = \begin{bmatrix} 0.017 & 0.001 \\ 0.001 & 0.023 \\ 0 & 0.061 \\ 0.072 & 0 \end{bmatrix}.$$

For the simulation, we consider the following values: $T_{\text{ini}} = 35$, $N = 45$, the simulation time $T_c = 2100$, $x(0) = [0.5, 0.5, 0.5, 0.5]^\top$, and d_p and d_m are considered random values such that $\|d_p\|, \|d_m\| \leq 0.002$. Moreover, the initial trajectory $(u_{\text{ini}}, y_{\text{ini}})$ is generated by applying zero control input to the system and measuring the corresponding output.

The evolution of $\lambda(k)$ is shown in Fig. 5.1, where one can see that the behavior of the LTV system is repeating in real time. Fig. 5.1 also depicts the order of the reduced-order mosaic-Hankel matrix for the proposed online DeePC (5.15), showing that the adaptive order r_a changes when the LTV system switches to different dynamics. For the online DeePC [61], the order is constant and equal to the rank of the mosaic-Hankel matrix, which is 320 for the collected data set.

Figs. 5.2 and 5.3 show the control performance of the proposed online reduced-order DeePC (5.15) in comparison with the traditional DeePC [27], the online DeePC [56] (replacing old data with new data in the mosaic-Hankel matrix), and the online DeePC [61] (adding new data to the mosaic-Hankel matrix). For the traditional DeePC [27], the DeePC policy (5.9) is employed such that the first optimal control input is applied to the system, and the initial trajectory $\{u_{\text{ini}}, y_{\text{ini}}\}$ is updated for the next step (see Algorithm 2 in [27]). From Figs. 5.2 and 5.3, one can see that the proposed online DeePC shows better tracking performance in comparison with other control schemes since it uses informative data to update the mosaic-Hankel matrix. The online DeePC [56] does not show a reasonable performance when all PE data is removed from the mosaic-Hankel matrix since the rank condition is not enough to evaluate suitable data. Table 5.1 demonstrates that the developed online DeePC significantly reduces the computational time compared to other controllers while keep the tracking performance well. Moreover, the average time for computing the numerical SVD is $0.0261ms$ for the online DeePC [61]; however, our proposed numerical SVD (5.22) takes $0.0085ms$, which leads to lower computational complexity for the the online DeePC (5.15). The dimension of the online reduced-order DeePC (5.15) is another reason for better computational time since it is the minimum possible dimension for the mosaic-Hankel matrix. It is worth noting that Track. Perf. represents tracking performance, which is root mean squared error as $\|reference - y\|$, for different control frameworks.

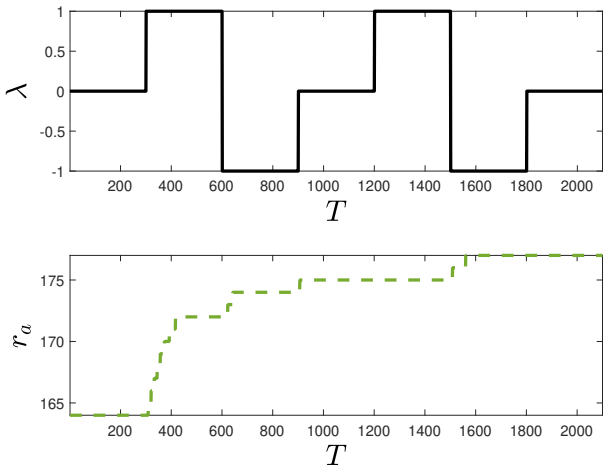

Figure 5.1: Order of Reduced-Order Mosaic-Hankel Matrix for LTV System.

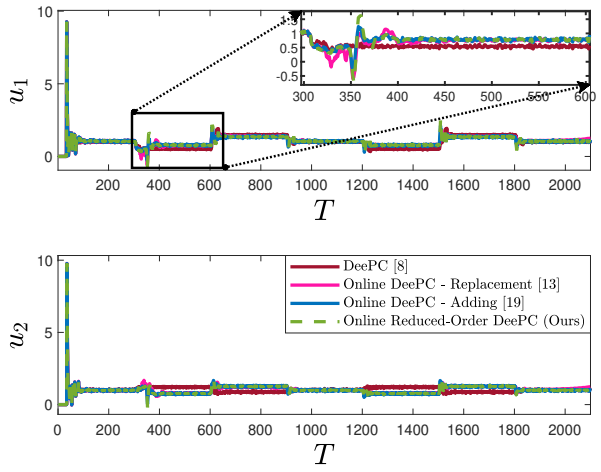

Figure 5.2: Comparison of Control Inputs for LTV System.

Table 5.1: Comparison of Tracking Performance and Computational Cost for LTV System.

Controller	Track. Perf.	Time (per loop)
DeePC [8]	4.83	0.21 *s*
Online DeePC [13]	3.39	0.18 *s*
Online DeePC [19]	3.05	0.09 *s*
Online R.O. DeePC (Ours)	3.05	0.05 *s*

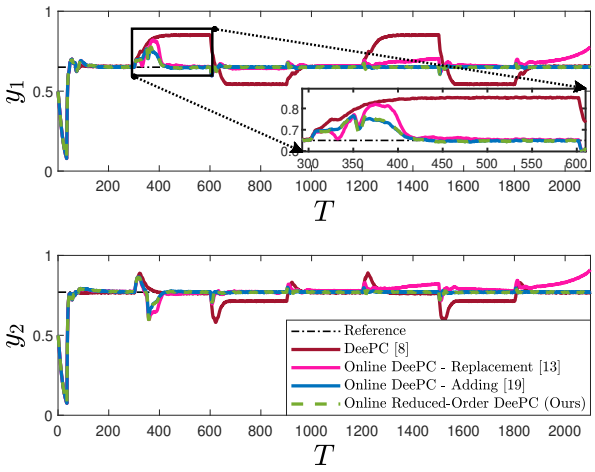

Figure 5.3: Comparison of System Outputs for LTV System.

5.4.2 Vehicle Rollover Avoidance

Rollover is a type of vehicle accident in which a vehicle tips over onto its side or roof. The rollover propensity of a vehicle is changed for different road surfaces or carried loads. Therefore, it is a big challenge to derive an accurate model for the vehicle dynamics which includes all operating conditions. We apply our online reduced-order DeePC to safeguard a vehicle against rollover. Considering a constant longitudinal speed for the vehicle, the steering wheel angle (SW) acts as the command, which is generated either by a human operator or a higher-level planning algorithm. For an arbitrary reference command SW, denoted as u_r, the rollover constraint may not be satisfied. Thus, DeePC is used as a safety filter for the reference command SW u_r to obtain an admissible input u. Indeed, DeePC ensures compliance with the load transfer ratio (LTR) constraint to prevent potential rollovers.

Following [67], a linear model is considered to represent the vehicle dynamics, which vehicle roll angle q, roll rate p, lateral velocity v, and yaw rate γ represent the states of the system. Considering the system in the format of (5.23), we have $A(k) = A^o + 0.01\lambda(k)A^o$, $B(k) =$

$B^o + 0.01\lambda(k)B^o$ as the time-varying matrices, and A^o, B^o, and C are considered as:

$$A^o = T_s \begin{bmatrix} 0.00499 & 0.997 & 0.0154 & -6.81 \times 10^{-5} \\ -78.3 & -12.2 & -65.3 & -3.89 \\ -0.932 & -0.799 & -6.20 & -1.57 \\ 1.52 & 3.32 & 8.27 & -1.49 \end{bmatrix} + I_4,$$

$$B^o = T_s \begin{bmatrix} -5.76 \times 10^{-5} & 2.80 & 0.278 & 0.655 \end{bmatrix}^T,$$

$$C = \begin{bmatrix} 0.1200 & 0.0124 & -0.0108 & 0.0109 \end{bmatrix},$$

where $x = \begin{bmatrix} q & p & v & \gamma \end{bmatrix}^T$, $u = SW$, $y = LTR$, and T_s is sampling time. It should be mentioned that the matrices A^o, B^o, and C are obtained based on a CarSim model for a standard utility truck under a constant longitudinal speed $80km/h$. More specifically, the vehicle tracks a constant reference longitudinal speed $80km/h$ using a feedback control on the gas pedal, which is not discussed here.

Through the LTR, the rollover constraint is defined as:

$$LTR = \frac{F_{z,R} - F_{z,L}}{mg},$$

where mg is the vehicle weight, and $F_{z,R}$ and $F_{z,L}$ stand for the total vertical force on the right-side tires and the left-side tires, respectively. Note that $|LTR| > 1$ means wheels lifting off; thus, the rollover constraint is imposed as:

$$-1 \leq LTR \leq 1.$$

For the simulation, we consider the following values: $T_{ini} = 10$, $N = 15$, the simulation time $T_c = 1200$, the smapling time $T_s = 0.1$, $x(0) = [0, 0, 0, 0]^T$, and random values $\|d_p\|, \|d_m\| \leq 0.002$. Moreover, the first initial trajectory (u_{ini}, y_{ini}) is generated by applying $u(k) = 55, k = 1 : T_{ini}$ to the system and measuring the corresponding output. For this case, $\lambda(k)$ and r_a are shown in Fig. 5.4, where one can see that the adaptive order r_a changes when the system switches to another dynamics. For the online DeePC [61], the rank of the mosaic-Hankel matrix is 50, which is the order of the reduced-order mosaic-Hankel matrix. Like the previous case, Fig. 5.4

illustrates that the online DeePC (5.15) has lower order than the online DeePC [61], which leads to better computational cost. Figs. 5.5 and 5.6 show the control performance of the proposed online reduced-order DeePC (5.15) in comparison with the traditional DeePC [27], the online DeePC [56] (replacing old data by new data in the mosaic-Hankel matrix), and the online DeePC [61] (adding new data to the mosaic-Hankel matrix). Fig. 5.5 shows the tracking performance and modification to the reference command SW for the control strategies. As shown in Fig. 5.6, one can see that the proposed online reduced-order DeePC only satisfies the LTR constraint, and other control schemes cannot satisfy the constraint. Table II demonstrates that the developed online DeePC significantly reduces the computational time compared to other controllers while keep the tracking performance and system safety well. It should be mentioned that Const. Viol. represents constraint violation number for different control frameworks. In this case, not only our online DeePC has lower computational cost compared to the online DeePC [61], but also it only satisfies the safety constraint due to employing useful information in the mosaic-Hankel matrix.

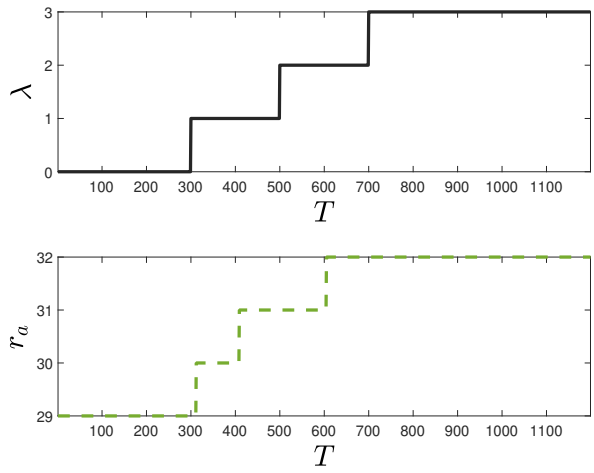

Figure 5.4: Order of Reduced-Order Mosaic-Hankel Matrix for Vehicle Rollover Avoidance.

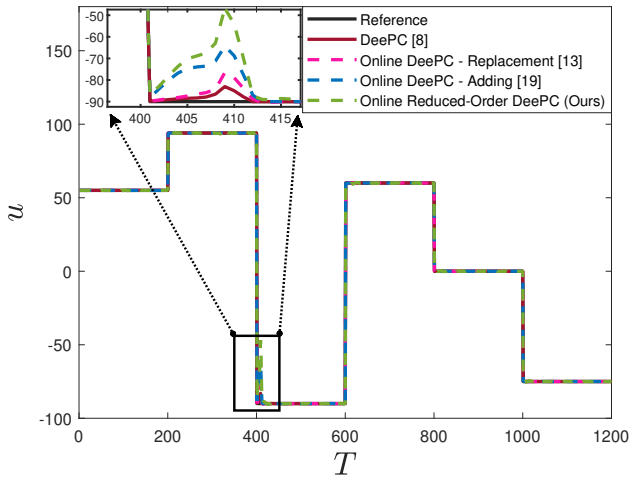

Figure 5.5: Control Inputs for Vehicle Rollover Avoidance.

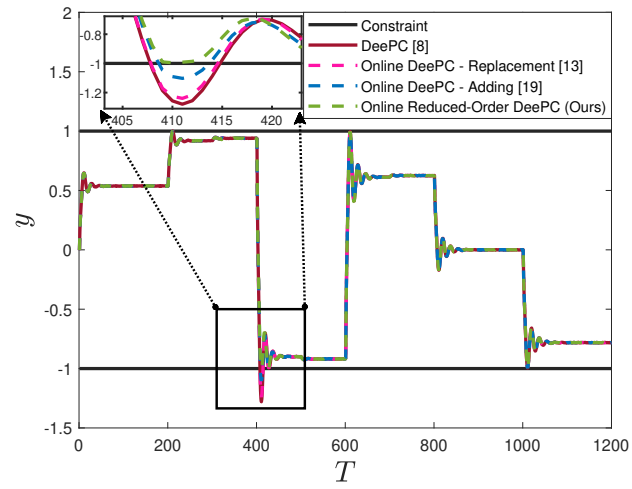

Figure 5.6: System Outputs for Vehicle Rollover Avoidance.

Table 5.2: Comparison of Safety Performance and Computational Cost for Vehicle Rollover Avoidance.

Controller	Const. Viol.	Time (per loop)
DeePC [8]	1	0.024 s
Online DeePC [13]	1	0.025 s
Online DeePC [19]	1	0.010 s
Online R.O. DeePC (Ours)	0	0.007 s

5.5 Chapter Summary

In this chapter, we proposed an online DeePC framework that incorporates real-time data updates into the Hankel matrix, leveraging the minimum non-zero singular value to selectively integrate informative signals. This approach effectively captures the dynamic nature of the system, ensuring improved control performance. Furthermore, we introduced a numerical SVD technique to mitigate the computational complexity associated with data integration. Simulation results validated the efficacy of the proposed online reduced-order DeePC framework, demonstrating its potential for achieving optimal control in evolving system environments.

CHAPTER 6

CONCLUSION

In this thesis, we addressed several existing challenges about the nonlinear optimal control, including parametric model requirement, model uncertainties, and computational cost. We approached this topic with three goals in mind: (i) Improving the neighboring extremal (NE) optimal control to handle the model uncertainties; (ii) Designing a data-driven neighboring extremal optimal control; and (iii) Developing an adaptive data-enabled predictive control (DeePC).

6.1 Contributions

In Chapter 3, we introduced an extended NE (ENE) to handle the model uncertainties for the NE, where effectively reduces the computational cost of the model-based nonlinear optimal control. The developed ENE was based on the second-order variation of the original optimization problem, which led to a set of Riccati-like backward recursive equations. The ENE adapted a nominal trajectory to the state and preview perturbations, and a multi-segment strategy was employed to guarantee closed-loop performance and constraint satisfaction for the large perturbations. In Chapter 4, we introduced a data-enabled neighboring extremal (DeeNE) to remove the parametric model requirement for the NE, where is very useful for high computational cost of DeePC. We also developed a scheme to handle nominal non-optimal solutions so that we can use the DeeNE solution as the nominal solution during the control process. The developed DeeNE was based on the second-order variation of the original DeePC problem such that the computational load of the DeeNE grows linearly for the optimization horizon. In Chapter 5, we introduced an adaptive DeePC framework for time-varying systems, enabling the algorithm to update the Hankel matrix online by adding real-time informative signals. By exploiting the minimum non-zero singular value of the Hankel matrix, the developed online DeePC selectively integrated informative data and effectively captured evolving system dynamics. Additionally, a numerical singular value decomposition technique was introduced to reduce the computational complexity for updating a reduced-order Hankel matrix.

6.2 Future Works

As the future works, two interesting research directions are Nonlinear Fundamental Lemma and Data-Enabled Control Barrier Function. Below are the detailed objectives of each topic.

6.2.1 Nonlinear Fundamental Lemma

The Fundamental Lemma accurately learns the system's behaviour such that the subspace of the I/O trajectories of a linear time invariant (LTI) system can be obtained from the column span of a data Hankel matrix. However, this lemma is not perfect to learn the behaviors of the nonlinear systems. Therefore, the DeePC is robustifies the Fundamental Lemma through a suitable regularization to ensure good performance for the nonlinear systems. However, deriving a rigorous mathematical theory to develop a Nonlinear Fundamental Lemma would highly show a better performance for the nonlinear optimal control compared to regularization.

6.2.2 Data-Enabled Control Barrier Function

To guarantee system safety, control barrier function (CBF) has recently emerged as a promising framework to efficiently handle system constraints [68]. The CBF implies forward invariance of a safety set based on system dynamics, and robust CBF and adaptive CBF maintains system safety in the presence of system uncertainties such as unknown disturbance, model mismatch, and state estimation error [69, 70, 71, 72, 73]. A data-enabled CBF can be proposed for the nonlinear optimal control to guarantee system safety using raw input/output data.

CHAPTER 7

PUBLICATIONS

The research conducted during the author's time as a PhD student has involved close collaboration with a number of colleagues, and this section lists all the articles submitted or published during this time.

Journal Publications:

- A. Vahidi-Moghaddam, K. Zhang, X. Yin, V. Srivastava, Z. Li, "Online Reduced-Order Data-Enabled Predictive Control," arXiv preprint arXiv:2407.16066, 2024.

- A. Vahidi-Moghaddam, K. Zhu, K. Zhang, and Z. Li, "Computationally Efficient Data-Enabled Predictive Control for Arm Robots," 2024.

- A. Vahidi-Moghaddam, K. Chen, K. Zhang, Z. Li, Y. Wang, and K. Wu. "A Unified Framework for Online Data-Driven Predictive Control with Robust Safety Guarantees," IEEE Transactions on Automation Science and Engineering, 2024.

- A. Vahidi-Moghaddam, K. Zhang, Z. Li, X. Yin, Z. Song, and Y. Wang, "Extended Neighboring Extremal Optimal Control with State and Preview Perturbations," IEEE Transactions on Automation Science and Engineering, 2023.

- Z. Li, A. Vahidi-Moghaddam, H. Modares, X. Wang, and J. Sun, "Finite-Time Disturbance Rejection for Nonlinear Systems using an Adaptive Disturbance Observer based on Experience-Replay," International Journal of Adaptive Control and Signal Processing, vol. 36, no. 8, pp. 2065-2082, 2022.

- A. Vahidi-Moghaddam, M. Mazouchi, and H. Modares, "Memory-Augmented System Identification with Finite-Time Convergence," IEEE Control Systems Letters, vol. 5, no. 2, pp. 571-576, 2020.

Conference Publications:

- A. Vahidi-Moghaddam, K. Zhang, Z. Li, and Y. Wang, "Data-Enabled Neighboring Extremal Optimal Control: A Computationally Efficient DeePC," in 2023 IEEE 62nd Conference on Decision and Control (CDC), IEEE, 2023.

- A. Vahidi-Moghaddam, Z. Li, N. Li, K. Zhang, and Y. Wang, "Event-Triggered Cloud-based Nonlinear Model Predictive Control with Neighboring Extremal Adaptations," in 2022 IEEE 61st Conference on Decision and Control (CDC), pp. 3724-3731. IEEE, 2022.

- A. Vahidi-Moghaddam, K. Chen, Z. Li, Y. Wang, and K. Wu, “Data-Driven Safe Predictive Control using Spatial Temporal Filter-based Function Approximators,” in 2022 American Control Conference (ACC), pp. 2803–2809, IEEE, 2022.

- A. Vahidi-Moghaddam, M. Mazouchi, and H. Modares, "Learning Dynamics System Models with Prescribed-Performance Guarantees using Experience-Replay," In 2021 American Control Conference (ACC), pp. 1941-1946, IEEE, 2021.

BIBLIOGRAPHY

[1] David A Copp, Kyriakos G Vamvoudakis, and João P Hespanha. "Distributed output-feedback model predictive control for multi-agent consensus". In: *Systems & Control Letters* 127 (2019), pp. 52–59.

[2] Robert AE Zidek, Ilya V Kolmanovsky, and Alberto Bemporad. "Model predictive control for drift counteraction of stochastic constrained linear systems". In: *Automatica* 123 (2021), p. 109304.

[3] Angelo Bratta et al. "Optimization-Based Reference Generator for Nonlinear Model Predictive Control of Legged Robots". In: *Robotics* 12.1 (2023), p. 6.

[4] Jeremy Lore et al. "Model predictive control of boundary plasmas using reduced models derived from SOLPS-ITER". In: *APS Division of Plasma Physics Meeting Abstracts*. Vol. 2021. 2021, NM09–002.

[5] Kaixiang Zhang, Yang Zheng, and Zhaojian Li. "Dimension Reduction for Efficient Data-Enabled Predictive Control". In: *arXiv preprint arXiv:2211.03697* (2022).

[6] Dinesh Krishnamoorthy, Ali Mesbah, and Joel A Paulson. "An adaptive correction scheme for offset-free asymptotic performance in deep learning-based economic MPC". In: *IFAC-PapersOnLine* 54.3 (2021), pp. 584–589.

[7] Yajie Bao et al. "Learning-based Adaptive-Scenario-Tree Model Predictive Control with Probabilistic Safety Guarantees Using Bayesian Neural Networks". In: *2022 American Control Conference (ACC)*. IEEE. 2022, pp. 3260–3265.

[8] Elena Arcari, Andrea Carron, and Melanie N Zeilinger. "Meta learning MPC using finite-dimensional Gaussian process approximations". In: *arXiv preprint arXiv:2008.05984* (2020).

[9] Elena Arcari et al. "Bayesian multi-task learning mpc for robotic mobile manipulation". In: *IEEE Robotics and Automation Letters* (2023).

[10] Amin Vahidi-Moghaddam et al. "Data-Driven Safe Predictive Control Using Spatial Temporal Filter-based Function Approximators". In: *2022 American Control Conference (ACC)* (2022).

[11] Amin Vahidi-Moghaddam et al. "A Unified Framework for Online Data-Driven Predictive Control with Robust Safety Guarantees". In: *arXiv preprint arXiv:2306.17270* (2023).

[12] Robert L Grossman. "The case for cloud computing". In: *IT professional* 11.2 (2009), pp. 23–27.

[13] Saad Mubeen et al. "Delay mitigation in offloaded cloud controllers in industrial IoT". In: *IEEE Access* 5 (2017), pp. 4418–4430.

[14] Reza Ghaemi, Jing Sun, and Ilya Kolmanovsky. "Computationally efficient model predictive control with explicit disturbance mitigation and constraint enforcement". In: *Proceedings of the 45th IEEE Conference on Decision and Control*. IEEE. 2006, pp. 4842–4847.

[15] Reza Ghaemi, Jing Sun, and Ilya Kolmanovsky. "Neighboring extremal solution for discrete-time optimal control problems with state inequality constraints". In: *2008 American Control Conference*. IEEE. 2008, pp. 3823–3828.

[16] Matt Wytock, Nicholas Moehle, and Stephen Boyd. "Dynamic energy management with scenario-based robust MPC". In: *2017 American Control Conference (ACC)*. IEEE. 2017, pp. 2042–2047.

[17] Brett T Lopez, Jean-Jacques E Slotine, and Jonathan P How. "Dynamic tube MPC for nonlinear systems". In: *2019 American Control Conference (ACC)*. IEEE. 2019, pp. 1655–1662.

[18] Carmen Amo Alonso et al. "Robust Distributed and Localized Model Predictive Control". In: *arXiv preprint arXiv:2103.14171* (2021).

[19] Mohammad R Hajidavalloo et al. "MPC-Based Vibration Control and Energy Harvesting using Stochastic Linearization for a New Energy Harvesting Shock Absorber". In: *2021 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE. 2021, pp. 38–43.

[20] Francoise Lamnabhi-Lagarrigue et al. "Systems & control for the future of humanity, research agenda: Current and future roles, impact and grand challenges". In: *Annual Reviews in Control* 43 (2017), pp. 1–64.

[21] Zhong-Sheng Hou and Zhuo Wang. "From model-based control to data-driven control: Survey, classification and perspective". In: *Information Sciences* 235 (2013), pp. 3–35.

[22] Valentina Breschi, Andrea Sassella, and Simone Formentin. "On the design of regularized explicit predictive controllers from input-output data". In: *IEEE Transactions on Automatic Control* (2022).

[23] Julian Berberich et al. "Data-driven model predictive control with stability and robustness guarantees". In: *IEEE Transactions on Automatic Control* 66.4 (2020), pp. 1702–1717.

[24] Claudio De Persis and Pietro Tesi. "Formulas for data-driven control: Stabilization, optimality, and robustness". In: *IEEE Transactions on Automatic Control* 65.3 (2019), pp. 909–924.

[25] Jan C Willems and Jan W Polderman. *Introduction to mathematical systems theory: a behavioral approach*. Vol. 26. Springer Science & Business Media, 1997.

[26] Jan C Willems et al. "A note on persistency of excitation". In: *Systems & Control Letters* 54.4 (2005), pp. 325–329.

[27] Jeremy Coulson, John Lygeros, and Florian Dörfler. "Data-enabled predictive control: In the shallows of the DeePC". In: *2019 18th European Control Conference (ECC)*. IEEE. 2019, pp. 307–312.

[28] Paolo Gherardo Carlet et al. "Data-driven continuous-set predictive current control for synchronous motor drives". In: *IEEE Transactions on Power Electronics* 37.6 (2022), pp. 6637–6646.

[29] Ezzat Elokda et al. "Data-enabled predictive control for quadcopters". In: *International Journal of Robust and Nonlinear Control* 31.18 (2021), pp. 8916–8936.

[30] Linbin Huang et al. "Data-enabled predictive control for grid-connected power converters". In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE. 2019, pp. 8130–8135.

[31] Reza Ghaemi, Soryeok Oh, and Jing Sun. "Path following of a model ship using model predictive control with experimental verification". In: *Proceedings of the 2010 American control conference*. IEEE. 2010, pp. 5236–5241.

[32] Hyeongjun Park et al. "Real-time model predictive control for shipboard power management using the IPA-SQP approach". In: *IEEE Transactions on Control Systems Technology* 23.6 (2015), pp. 2129–2143.

[33] Yanhui Xie et al. "Model predictive control for a full bridge DC/DC converter". In: *IEEE Transactions on Control Systems Technology* 20.1 (2011), pp. 164–172.

[34] Hyeongjun Park, Ilya Kolmanovsky, and Jing Sun. "Model predictive control of spacecraft relative motion maneuvers using the IPA-SQP approach". In: *Dynamic Systems and Control Conference*. Vol. 56123. American Society of Mechanical Engineers. 2013, V001T02A001.

[35] Amirhossein Ahmadi et al. "Deep Federated Learning-Based Privacy-Preserving Wind Power Forecasting". In: *IEEE Access* 11 (2022), pp. 39521–39530.

[36] Linbin Huang et al. "Robust data-enabled predictive control: Tractable formulations and performance guarantees". In: *IEEE Transactions on Automatic Control* 68.5 (2023), pp. 3163–3170.

[37] Amin Vahidi-Moghaddam et al. "Extended Neighboring Extremal Optimal Control with State and Preview Perturbations". In: *IEEE Transactions on Automation Science and Engineering* (2023).

[38] Amin Vahidi-Moghaddam et al. "Event-Triggered Cloud-based Nonlinear Model Predictive Control with Neighboring Extremal Adaptations". In: *2022 IEEE 61st Conference on Decision and Control (CDC)*. IEEE. 2022, pp. 3724–3731.

[39] Mohammad R. Hajidavalloo et al. “Simultaneous Suspension Control and Energy Harvesting Through Novel Design and Control of a New Nonlinear Energy Harvesting Shock Absorber”. In: *IEEE Transactions on Vehicular Technology* 71.6 (2022), pp. 6073–6087.

[40] Mohammad Reza Amini et al. “Cabin and battery thermal management of connected and automated HEVs for improved energy efficiency using hierarchical model predictive control”. In: *IEEE Transactions on Control Systems Technology* 28.5 (2019), pp. 1711–1726.

[41] Jason Laks et al. “Model predictive control using preview measurements from lidar”. In: *49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*. 2011, p. 813.

[42] Reza Ghaemi, Masoud Abbaszadeh, and Pierino G Bonanni. “Optimal flexibility control of large-scale distributed heterogeneous loads in the power grid”. In: *IEEE Transactions on Control of Network Systems* 6.3 (2019), pp. 1256–1268.

[43] Shiva Bagherzadeh, Hossein Karimpour, and Mehdi Keshmiri. “Neighboring extremal nonlinear model predictive control of a rigid body on SO (3)”. In: *Robotica* (2023), pp. 1–22.

[44] Rohit Gupta, Anthony M Bloch, and Ilya V Kolmanovsky. “Combined homotopy and neighboring extremal optimal control”. In: *Optimal Control Applications and Methods* 38.3 (2017), pp. 459–469.

[45] Anthony M Bloch, Rohit Gupta, and Ilya V Kolmanovsky. “Neighboring extremal optimal control for mechanical systems on Riemannian manifolds”. In: *J. Geom. Mech* 8.3 (2016), pp. 257–272.

[46] Hyeongjun Park, Jing Sun, and Ilya Kolmanovsky. “A tutorial overview of IPA-SQP approach for optimization of constrained nonlinear systems”. In: *Proceeding of the 11th World Congress on Intelligent Control and Automation*. IEEE. 2014, pp. 1735–1740.

[47] Amin Vahidi-Moghaddam et al. “Computationally Efficient Data-Enabled Predictive Control for Arm Robots”. In: *arXiv preprint* (2024).

[48] Amin Vahidi-Moghaddam et al. “Data-Enabled Neighboring Extremal Optimal Control: A Computationally Efficient DeePC”. In: *2023 IEEE 62st Conference on Decision and Control (CDC)*. IEEE. 2023.

[49] Florian Dörfler, Jeremy Coulson, and Ivan Markovsky. “Bridging direct and indirect data-driven control formulations via regularizations and relaxations”. In: *IEEE Transactions on Automatic Control* 68.2 (2022), pp. 883–897.

[50] Valentina Breschi, Alessandro Chiuso, and Simone Formentin. “Data-driven predictive control in a stochastic setting: A unified framework”. In: *Automatica* 152 (2023), p. 110961.

[51] Wouter Favoreel, Bart De Moor, and Michel Gevers. “SPC: Subspace predictive control”. In: *IFAC Proceedings Volumes* 32.2 (1999), pp. 4004–4009.

[52] Francesco Toso et al. “On-line continuous control set MPC for PMSM drives current loops at high sampling rate using qpOASES”. In: *2019 IEEE Energy Conversion Congress and Exposition (ECCE)*. IEEE. 2019, pp. 6615–6620.

[53] Lars Grüne et al. “Analysis of unconstrained nonlinear MPC schemes with time varying control horizon”. In: *SIAM Journal on Control and Optimization* 48.8 (2010), pp. 4938–4962.

[54] Amin Vahidi-Moghaddam et al. “Online Reduced-Order Data-Enabled Predictive Control”. In: *arXiv preprint arXiv:2407.16066* (2024).

[55] Jeremy Coulson et al. “A quantitative notion of persistency of excitation and the robust fundamental lemma”. In: *IEEE Control Systems Letters* 7 (2022), pp. 1243–1248.

[56] Johannes Teutsch et al. “An Online Adaptation Strategy for Direct Data-driven Control”. In: *arXiv preprint arXiv:2304.03386* (2023).

[57] Julian Berberich et al. “Data-driven model predictive control: closed-loop guarantees and experimental results”. In: *at-Automatisierungstechnik* 69.7 (2021), pp. 608–618.

[58] Stefanos Baros et al. “Online data-enabled predictive control”. In: *Automatica* 138 (2022), p. 109926.

[59] Julian Berberich et al. “Linear tracking MPC for nonlinear systems—Part II: The data-driven case”. In: *IEEE Transactions on Automatic Control* 67.9 (2022), pp. 4406–4421.

[60] Ivan Markovsky and Florian Dörfler. “Identifiability in the behavioral setting”. In: *IEEE Transactions on Automatic Control* 68.3 (2022), pp. 1667–1677.

[61] Jicheng Shi and Colin N Jones. “Efficient Recursive Data-enabled Predictive Control: an Application to Consistent Predictors”. In: *arXiv preprint arXiv:2309.13755* (2023).

[62] Matthew Brand. “Fast low-rank modifications of the thin singular value decomposition”. In: *Linear algebra and its applications* 415.1 (2006), pp. 20–30.

[63] Ming Gu and Stanley C Eisenstat. *A stable and fast algorithm for updating the singular value decomposition*. 1993.

[64] James R Bunch and Christopher P Nielsen. “Updating the singular value decomposition”. In: *Numerische Mathematik* 31.2 (1978), pp. 111–129.

[65] Ivan Markovsky and Florian Dörfler. “Behavioral systems theory in data-driven analysis, signal processing, and control”. In: *Annual Reviews in Control* 52 (2021), pp. 42–64.

[66] Kaixiang Zhang et al. “Dimension Reduction for Efficient Data-Enabled Predictive Control”. In: *IEEE Control Systems Letters* 7 (2023), pp. 3277–3282. DOI: 10.1109/LCSYS.2023.3322965.

[67] Ricardo Bencatel et al. “Reference governor strategies for vehicle rollover avoidance”. In: *IEEE Transactions on Control Systems Technology* 26.6 (2017), pp. 1954–1969.

[68] Aaron D Ames et al. “Control barrier functions: Theory and applications”. In: *2019 18th European control conference (ECC)*. IEEE. 2019, pp. 3420–3431.

[69] Kunal Garg and Dimitra Panagou. “Robust control barrier and control Lyapunov functions with fixed-time convergence guarantees”. In: *2021 American Control Conference (ACC)*. IEEE. 2021, pp. 2292–2297.

[70] Mitchell Black, Ehsan Arabi, and Dimitra Panagou. “A fixed-time stable adaptation law for safety-critical control under parametric uncertainty”. In: *2021 European Control Conference (ECC)*. IEEE. 2021, pp. 1328–1333.

[71] Axton Isaly et al. “Adaptive safety with multiple barrier functions using integral concurrent learning”. In: *2021 American Control Conference (ACC)*. IEEE. 2021, pp. 3719–3724.

[72] Brett T Lopez, Jean-Jacques E Slotine, and Jonathan P How. “Robust Adaptive Control Barrier Functions: An Adaptive & Data-Driven Approach to Safety (Extended Version)”. In: *arXiv preprint arXiv:2003.10028* (2020).

[73] Andrew J Taylor and Aaron D Ames. “Adaptive safety with control barrier functions”. In: *2020 American Control Conference (ACC)*. IEEE. 2020, pp. 1399–1405.