ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

IFA Automatic Control Laboratory

Semester Project

Data driven adaptive control: a geometric approach

Sujet Phodapol
February 24, 2023

Advisors
Alberto Padoan
Jeremy Coulson

Abstract

With the advent of enhanced computing and storage capabilities, data-driven methods have significantly impacted several fields of research. In control theory, the new wave of data-driven methods has generated a renewed appreciation of Willems' fundamental lemma, which allows one to represent behaviors of linear time-invariant systems using input/output data. The fundamental lemma allows one to reformulate classical MPC algorithms in a purely data-driven setting and to achieve surprising performance in several applications. However, the time-invariance assumption is restrictive. This thesis explores a new algorithm, referred to as Dynamic Data-Enable Predictive Control (DDeePC), to control linear time-varying behaviors. DDeePC uses a subspace tracking algorithm, Grassmanian Rank-One Update Subspace Estimation (GROUSE), to leverage online measurement data to update the estimated subspace in real-time. Our algorithm dynamically enhances the predictive capabilities of DeePC, resulting in improved tracking performance for slowly time-varying systems. We numerically demonstrate the benefits and limitations of the DDeePC algorithm in different scenarios.

Contents

Chapter 1

Introduction

Data-driven methods have had a substantial impact on various sectors of research, due to the development of more efficient computing and storage capabilities. There are many successful applications in machine learning and artificial intelligence, including computer vision and robotics. Also, in control theory, researchers are beginning to leverage the fundamental lemma [22], which allows them to implicitly represent a dynamical system from the sequence of input/output data instead of modelling from physical knowledge. With this lemma, we can change the approach to designing controllers for unknown systems. To illustrate, there are two main approaches to using data in designing a controller: a direct and indirect approach. Indirect data-driven control is the conventional way to use the collected data with the knowledge of the structure of the model to do system identification to model the unknown system. Then the model is used in order to design the model-based controller. However, this identification process is a very expensive and tedious task [10, 12, 17, 19]. Also, this approach will provide the best approximate model, not the best controller for the system. In other words, in the end, we would like to control the system, we do not want to know the system. On the other hand, direct data-driven control directly uses the collected input/output data to generate the control signal without explicitly identifying the model, which will decrease the time and cost to design the controller. With this approach, we also do not need physical knowledge of the model which can possibly be complicated in many applications.

There are several data-driven control approaches that build on the fundamental lemma [2, 3, 7, 8, 20] both with predictive and feedback controllers. In this project, we focus on a specific predictive data-driven control, namely Data-enabled Predictive Control (DeePC) [5]. This direct data-driven method has shown great success in many applications on real-world applications, including quadrotors [9] and excavators [21]. These examples show the advantages of direct data-driven control that, with only data collection, this method can functionally control the complex system. However, the theory developed for DeePC is still limited to a small range of problems, including linear systems and systems with a small nonlinearity. In addition, many systems in the real world evolve over time (i.e. time-variant systems). In order to extend this approach to a wider range of applications, this method has to be able to adapt in real-time to the changing system.

There are several approaches to modify DeePC to be able to handle non-linearity in the system. The work from [13] demonstrates the approach to robustify DeePC by solving a min-max optimization, resulting in stabilizing a noisy input/output and nonlinear system. Another work from [6] address how to handle output chance constraints for unknown stochastic linear time-invariant systems. However, these approaches are more suitable for the LTI system, and may not be suitable for a system that evolves over time.

To deal with systems that change over time, researchers have proposed several ways to detect the change and adapt the controller according to this change. Gain-scheduling [15] is one the most popular way to control a system that changes from one operating condition to another. However, this approach needs data on all operating conditions and cannot work with unknown systems. For predictive control, work from [14] uses Gaussian process regression to learn the uncertainty in the model to improve the performance of the racing car. However, this learning approach requires a parametric model, which may not directly apply to the direct data-driven control. Another approach is to track the change in the subspace and does online update on the subspace [1]. This work demonstrates an online tracking algorithm that builds on the sample observation vector. However, this algorithm is used to only identify the subspace of the system not to control the system.

In this project, we propose an adaptive data-driven control, namely Dynamic Data-enabled Predictive Control (DDeePC). This approach combines the state-of-the-art Data-enabled Predictive Control (DeePC) with the subspace tracking algorithm, which is Grassmanian Rank-One Update Subspace Estimation (GROUSE). DDeePC is able to adapt the data model based on the new data collection from evolving systems, resulting in stabilizing linear time-varying system.

The contributions of this project can be summarised as follows:

- We show the framework that combines DeePC and GROUSE as an adaptive data-driven controller framework.
- We illustrate the behaviour and performance of DeePC and DDeePC in different time-varying systems.
- We demonstrate the performance of the DeePC and DDeePC on noisy systems.

The outline of this report is organized as follows. In Chapter 2, we will go through theories on data-driven control, subspace tracking and an implementation technique. Chapter 3 will present the experimental setup and the numerical results of the experiment. Chapter 4 summarizes our main results and outlines future research directions.

Chapter 2

Background

This Chapter will explain the theory behind DDeePC algorithm. Before we can design the direct data-driven controller, we will introduce notations and theories that are needed in this approach. There are two main components used to build this algorithm: Data-enabled Predictive Control (DeePC) and subspace tracking algorithm (we use Grassmannian Rank-One Update Subspace Estimation (GROUSE) in this project).

2.1 Behavioral system theory

A dynamical system can be defined from its behaviour according to [16, 18]. From [5], we define behaviour and properties of a dynamical system as follows:

Definition 1. A *dynamical system* is a tuple with three elements $(\mathbb{Z}_{\geq 0}, \mathbb{W}, \mathcal{B})$, where $\mathbb{Z}_{\geq 0}$ is the discrete-time set, $\mathbb{W}$ is the signal space, and $\mathcal{B} \subseteq \mathbb{W}^{\mathbb{Z}_{\geq 0}}$ is the behavior. ⌟

Definition 2. A dynamical system $(\mathbb{Z}_{\geq 0}, \mathbb{W}, \mathcal{B})$ is *linear* if $\mathbb{W}$ is a vector space and $\mathcal{B}$ is a linear subspace of $(\mathbb{W})^{\mathbb{Z}_{\geq 0}}$, *time invariant* if $\sigma\mathcal{B} \subseteq \mathcal{B}$ where σ is the shift operator and *complete* if $\mathcal{B}$ is closed in the topology of pointwise convergence. ⌟

Definition 3. A dynamical system $\mathcal{B} \in \mathcal{L}^q$, where $\mathcal{L}^q$ is the linear, time-invariant and complete system and $q \in \mathbb{Z}_{\geq 0}$, is *controllable* if for every $T \in \mathbb{N}$, $w^1 \in \mathcal{B}|_T$, where $\mathcal{B}|_T = \{w \in (\mathbb{R}^q)^T | \exists v \in \mathcal{B} \text{ s.t. } w_t = v_t, 1 \leq t \leq T\}$ is a truncated trajectory of length T, and $w^2 \in \mathcal{B}$ there exists $T' \in \mathbb{N}$, and $w \in \mathcal{B}$ such that $w_t = w_t^1$ for $1 \leq t \leq T$ and $w_t = w_{t-T-T'}^2$ for $t > T + T'$. ⌟

In other words, a dynamical system is controllable if any two trajectories can be patched together in finite time.

Definition 4. Let $u = (u_1, u_2, ..., u_T) \in \mathbb{R}^{mT}$. Given $T, T_f \in \mathbb{N}$, the *Hankel matrix* of depth $T_f \leq T$ associated with u is defined as

$$\mathcal{H}_{T_f}(u) = \begin{bmatrix} u_1 & u_2 & \cdots & u_{T-T_f+1} \\ u_2 & u_3 & \cdots & u_{T-T_f+2} \\ \vdots & \vdots & \ddots & \vdots \\ u_{T_f} & u_{T_f+1} & \cdots & u_T \end{bmatrix}$$

⌟

Definition 5. A vector $u \in \mathbb{R}^{Tm}$ is *persistently exciting of order* T_f if the Hankel matrix $\mathcal{H}_{T_f}(u)$ is full row rank. ⌟

Persistency of excitation is important to reconstruct the system behaviour (i.e., system identification). It is necessary for the signal to be sufficiently rich and long. In other words, $H_{T_f}(u)$ has at least as many columns as rows:

$$T \geq (m+1)T_f - 1$$

Willems' fundamental lemma [22] is the key to the data-driven approach, DeePC. With all mentioned definitions, we can state the fundamental lemma here:

Lemma 1. Fundamental lemma : Consider a controllable discrete-time LTI system $\mathcal{B}$. Assume data trajectory $w^d = \text{col}(u^d, y^d) \in \mathcal{B}_T$ and input data u^d is persistently exciting of order $T_f + n$, where n is a number of states. Then, $\mathcal{B}_{T_f} = \text{colspan}(\mathcal{H}_{T_f}(w^d))$ ⌟

The fundamental lemma provides conditions for a constrained behavior B_{T_f} to be fully defined by the image of the Hankel matrix $\mathcal{H}_{T_f}(w^d)$, which is created using just input-output data, it is one of the most important lemmas in data-driven control.

2.2 Data-enabled Predictive Control (DeePC)

DeePC [5] is a direct data-driven control algorithm similar to model predictive control (MPC). In other words, it is the controller that solves an optimization problem in a receding horizon manner. The main difference is that, instead of using expert knowledge to build the predicted model in the optimization problem like MPC, DeePC relies on Willems' fundamental lemma to directly use data to build the predictor which can predict the trajectory within the finite horizon.

The algorithm starts by assuming we have collected data from an unknown controllable LTI system $\mathcal{B}$ with m inputs and p outputs. Let $u^{\text{d}} = \text{col}(u_1^{\text{d}}, \ldots, u_T^{\text{d}}) \in \mathbb{R}^{mT}$ be a sequence of T inputs applied to $\mathcal{B}$ and the sequence of outputs $y^{\text{d}} = \text{col}(y_1^{\text{d}}, \ldots, y_T^{\text{d}}) \in \mathbb{R}^{pT}$ are measured where $T \in \mathbb{N}$ and superscript d indicates that these data is the data collected from the offline process from the system. Also, assume u^{d} is persistently exciting of order $T_{\text{ini}} + T_{\text{f}} + n$ where n is the order of the system and $T_{\text{ini}}, T_{\text{f}} \in \mathbb{N}$. Then, these input/output data are divided into two parts: past data (p) and future data (f) and put into the Hankel matrices as follows:

$$\begin{pmatrix} U_{\text{p}} \\ U_{\text{f}} \end{pmatrix} = \mathcal{H}_{T_{\text{ini}}+T_{\text{f}}}(u^{\text{d}}), \quad \begin{pmatrix} Y_{\text{p}} \\ Y_{\text{f}} \end{pmatrix} = \mathcal{H}_{T_{\text{ini}}+T_{\text{f}}}(y^{\text{d}}),$$

where past data matrix (subscript p) includes the first T_{ini} block rows of $\mathcal{H}$ and future data matrix (subscript f) includes the last T_{f} block rows of $\mathcal{H}$.

From the Lemma 1, we can use past data and future data to construct the trajectory of length $\mathcal{B}|_{T_{\text{ini}}+T_{\text{f}}}$, where past data is used to estimate the initial state, whereas future data is used to predict the future trajectories. Then, a trajectory $\text{col}(u_{\text{ini}}, u, y_{\text{ini}}, y)$ belongs to $\mathcal{B}|_{T_{\text{ini}}+T_{\text{f}}}$ if and only if there exists $g \in \mathbb{R}^{T-T_{\text{ini}}-T_{\text{f}}+1}$ such that

$$\begin{pmatrix} U_{\text{p}} \\ Y_{\text{p}} \\ U_{\text{f}} \\ Y_{\text{f}} \end{pmatrix} g = \begin{pmatrix} u_{\text{ini}} \\ y_{\text{ini}} \\ u \\ y \end{pmatrix}$$

Given a reference trajectory $r = (r_0, r_1, r_2, ...) \in (\mathbb{R}^p)^{\mathbb{Z}_{>0}}$, a prediction horizon $N \in \mathbb{Z}_{>0}$, input constraints $\mathcal{U} \subseteq \mathbb{R}^m$, output constraints $\mathcal{Y} \subseteq \mathbb{R}^p$, output cost matrix $Q \in \mathbb{R}^{p \times p}$ and control cost matrix $R \in \mathbb{R}^{m \times m}$, the DeePC algorithm can be written as follows:

$$
\begin{aligned}
&\min_{g,u,y} \sum_{k=0}^{N-1} (||y_k - r_{t+k}||_Q^2 + ||u_k||_R^2) \\
&\text{subject to} \begin{pmatrix} U_p \\ Y_p \\ U_f \\ Y_f \end{pmatrix} g = \begin{pmatrix} u_{ini} \\ y_{ini} \\ u \\ y \end{pmatrix}, \\
&\qquad\qquad u_k \in \mathcal{U}, \forall k \in \{0, ..., N-1\}, \\
&\qquad\qquad y_k \in \mathcal{Y}, \forall k \in \{0, ..., N-1\}
\end{aligned} \tag{2.1}
$$

Algorithm 1 DeePC

1: Solve optimization problem for g, u, y.
2: Apply first optimal control input.
3: Update u_{ini} and y_{ini} in the optimization problem from the recent measurement.
4: Return to 1.

2.3 The GROUSE algorithm

Grassmannian Rank-One Update Subspace Estimation (GROUSE) [1] is an online subspace tracking algorithm that builds on a linear algebraic update rule in each updating iteration. This algorithm aims to track an evolving d-dimensional subspace ($S[t]$) of $\mathbb{R}^n$. At each time step t, we observe a data vector $v_t \in S[t]$ at location $\Omega_t \subset 1, ..., n$. Then, one can minimize a cost function which is the squared Euclidean distance between the current subspace estimate (U_{Ω_t}) and the observed data vector (v_{Ω_t}) on the position that we have access to data in Ω_t as:

$$F(S;t) = \min_a ||U_{\Omega_t} a - v_{\Omega_t}||^2 \tag{2.2}$$

where subscript Ω_t indicates the matrix with rows indexed by Ω_t. Since in this project, we have full access to the observation, this Ω_t can be ignored. We can rewrite

$$F(S;t) = \min_a ||Ua - v_t||^2 \tag{2.3}$$

In other words, the algorithm will be initialized with a subspace, then it will measure the observation from the subspace and solve the minimization problem to minimize the distance ($F(S;t)$) between the observation vector (v_t) and the subspace (U_t). Then this objective function will be used to compute the gradient descent on the Grassmannian manifold to update the subspace in that direction. Let η be a step size in the direction of the gradient. The algorithm can be summarized in pseudo-code as follows:

Algorithm 2 GROUSE

1: Estimate weights: $w = arg\min_a ||U_t a - v_t||^2$
2: Predict full vector: $p = U_t w$
3: Compute residual: $r = v_t - p$
4: Update subspace: $U_{t+1} = U_t + \left((cos(\sigma\eta) - 1)\frac{p}{||p||} + sin(\sigma\eta)\frac{r}{||r||}\right)\frac{w^T}{||w||}; \sigma = ||r||||p||$

2.4 Dynamic Data-enabled Predictive Control (DDeePC)

In order to incorporate adaptive behaviour into DeePC, we propose a novel adaptive data-driven controller, namely DDeePC, which integrates the subspace tracking mechanism into the data-driven control framework. The main idea of the algorithm is to adapt the data model according to the changing system. In DDeePC, GROUSE is used as a subspace tracking algorithm which receives a new observation vector from the system as an input. The received input and output data are stored in a buffer of the most recent measured input and output $v_{seq} = \text{col}(u_1, u_2, ..., u_G, y_1, y_2, ..., y_G) \in \mathbb{R}^{(m+p)(T_{\text{ini}}+T_f+G-1)}$, where G is the number of iterations in the GROUSE algorithm. The observation vector $v_{obs} = \text{col}(u_{p_t}, y_{p_t}, u_{f_t}, y_{f_t} f) \in \mathbb{R}^{(m+p)(Tini+T_f)}$ is then constructed from the data buffer in the same order as the optimization problem, where $u_{p_t}, y_{p_t}, u_{f_t}, y_{f_t}$ are the measured data vector at time t. The stored data are used to generate past observation vectors. The current and past observation vector is then normalized to a unit vector in order to make the algorithm converge. In each update step, these vectors are used to update the predictor according to the GROUSE update rule. As shown in Fig. 2.1, the main difference of the DDeePC to the standard DeePC is that the former updates its predictor every time step, while the latter uses the fixed predictor from the initialization. Let *dim* be the number of columns of U_1 in the initialization of the DDeePC. We now present the DDeePC algorithm.

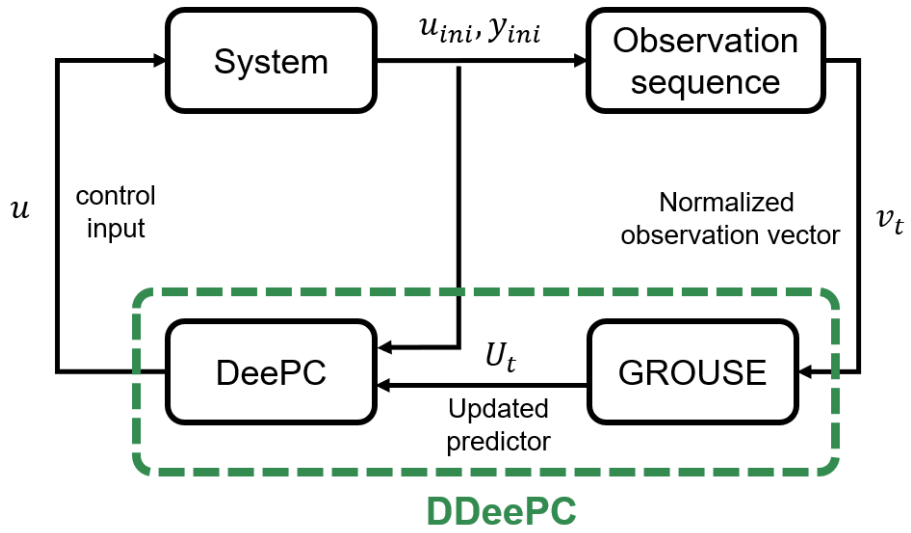


Figure 2.1: Block diagram of DDeePC illustrating the additional subspace tracking algorithm block.

Algorithm 3 DDeePC

Input: data trajectories $\text{col}(u^{\text{d}}, y^{\text{d}}) \in \mathbb{R}^{(m+p)T}$, most recent input/output measurements $\text{col}(u_{\text{ini}}, y_{\text{ini}}) \in \mathbb{R}^{(m+p)T_{\text{ini}}}$, most recent sequence of observation vector $v_{seq} \in \mathbb{R}^{(m+p)(T_{\text{ini}}+T_f+G-1)}$

Initialization

procedure INITIALIZATION($\mathcal{H}$,dim)
    Singular value decomposition: $\mathcal{H} = [U_1 U_2] \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} [V_1 V_2]^T$, $U_1 \in \mathbb{R}^{(m+p)(T_{ini}+T_f)\times dim}$
    Initialize a predictor: $U_t = U_1$
end procedure

Online adaptive control

procedure SUBSPACE UPDATE WITH GROUSE(U_t,v_{seq})
    for $i = 0$ to G **do**
        Construct observation vector: v_{obs} from v_{seq} at time $t-i$ back to $t-i-G$
        Normalized observation vector: $v_t = \frac{v_{obs}}{||v_{obs}||}$
        Estimate weights: $w = arg\min_a ||U_t a - v_t||^2$
        Predict full vector: $p = U_t w$
        Compute residual: $r = v_t - p$
        Update subspace: $U_t \leftarrow U_t + \left((cos(\sigma\eta) - 1)\frac{p}{||p||} + sin(\sigma\eta)\frac{r}{||r||}\right)\frac{w^T}{||w||}; \sigma = ||r||||p||$
    end for
    return U_t
end procedure
procedure DEEPC(U_t, u_{ini}, y_{ini})
    Similar to **Algorithm 1**
end procedure
Return to 1.

Chapter 3

Experiments and Results

This section illustrates the performance and limitations of the DDeePC algorithm by means of numerical simulations. First, we consider a slowly-varying system. The system is controlled by using both controllers, DeePC and DDeePC, in three different scenarios: small switching, large switching system, and continuously changing. Then hyperparameters which are the number of columns of the data matrix and regularization term (λ_g) for DeePC and the low-rank dimension from singular value decomposition in the initialization (dim) and regularization term (λ_g) for DDeePC are tuned using grid search. Second, both controllers will be tested with the systems that are corrupted with Gaussian measurement noise. To verify the stability of the controllers, two types of noise will be used. Similar to the first experiment, hyperparameter tuning will be done. Finally, the objective cost will be used to evaluate the results and limitations.

3.1 Experiments

To understand the benefit and drawbacks of the DDeePC. We set up three experiments to answer the following question:

- Q1: How is the performance of DeePC and DDeePC in switching systems?
- Q2: How is the performance of DeePC and DDeePC on a continuously evolving system?
- Q3: How noise has an effect on DeePC and DDeePC?

The data center model [4], which consists of a three-state system with the control u providing local cooling for each rack and the state x reflecting the internal temperature of the racks, is used in experiments. A linear model used to simulate this dynamic system is described as follows:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) + w(k) \end{aligned}$$

where each matrix is as follows:

$$A = \begin{bmatrix} 1.01 & 0.01 & 0 \\ 0.01 & 1.01 & 0.01 \\ 0 & 0.01 & 1.01 \end{bmatrix}, B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

w is the Gaussian measurement noise. To solve the regulation problem, the optimization problem 2.1 is modified to problem 3.2 with additional regularization terms as follows:

$$
\begin{aligned}
\underset{g,u,y}{\text{minimize}} \quad & \sum_{i=0}^{T_{\mathrm{f}}-1} \left(\|y_i\|_Q^2 + \|u_i\|_R^2 \right) \\
& + \lambda_g ||g||_2 + \lambda_{y_{ini}} ||\sigma_y||_2 \\
\text{subject to} \quad & Ug = \begin{pmatrix} u_{\mathrm{ini}} \\ y_{\mathrm{ini}} \\ u \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ \sigma_y \\ 0 \\ 0 \end{pmatrix}
\end{aligned} \tag{3.2}
$$

The weighting matrices of the controller (Q and R) are fixed in problem 3.2, as are the weighting parameters on the initial value ($\lambda_{y_{ini}}$) and the predicting horizon (T_f), but the hyperparameters on the regularizing term (λ_g) and the predictor's dimension (dim) are varied. We set the number of iterations to 20 for GROUSE in DDeePC (G). Other hyperparameters in the experiments are fixed as follows:

$$
Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, R = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{bmatrix}, \lambda_{y_{ini}} = 1000, T_f = 5
$$

By simulating system 3.1, experiments are conducted to assess the performance of controllers. In the initialization, generated data is then gathered to create a Hankel matrix (predictor). This predictor is then used to solve the optimization problem in the receding horizon manner for the regularization problem for both DeePC and DDeePC. The cost of the optimization problem is then evaluated.

3.1.1 Experiment I: Slow time-varying system

In the first experiment, we investigate the behaviour and performance of DeePC and DDeePC on the slow linear time-varying (LTV) noise free system. Small switching, large switching, and continuously changing systems are the three types of LTV systems, which we will examine. We select the first element in the A matrix to be a function that can change over time in order to imitate the variation in the system as follows:

$$
A(k) = \begin{bmatrix} a_1(k) & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix}
$$

To conduct experiments, We first collect data from a simulated time-invariant system and then build a data matrix. The optimization problem then uses this matrix as a predictor. Next, the system is simulated for 50 timesteps until it reaches a steady state, and then the system will be changed based on each experiment. Three objective costs in problem 3.2: total cost, pre-cost (i.e. total cost prior to the change) and post-cost (i.e. total cost after the change) is measured and evaluated. The experiment can be summarized in Fig. 3.1.

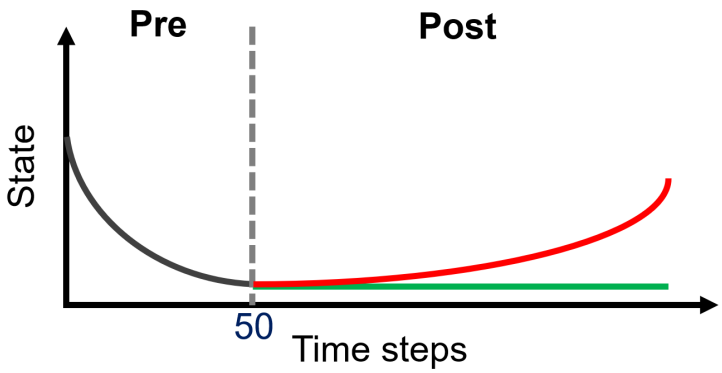


Figure 3.1: Time history of system controlled using DDeePC.The dashed line denotes the start of a system change. After the change, the system exhibits two different types of behaviors: the green line represents a stable behavior, whereas the red line represents an unstable behavior.

Switching system

In this experiment, our goal is to simulate a system that gradually transitions to another system and remains there indefinitely. For instance, a data center rack may have flaws that alter the material's heat capacity properties. In order to replicate this behaviour, we apply the Sigmoid function as a changing function since, with this function, the system will converge to another dynamic and stay there forever. We use the following equation to conduct two experiments with the small change and the large change (i.e., twice the size in amplitude of the small change) in the system:

$$a_1(k) = a_1(0) + 0.15(\frac{2}{1+e^{-k}} - 1)$$
$$a_1(k) = a_1(0) + 0.3(\frac{2}{1+e^{-k}} - 1)$$

The findings displayed in Fig. 3.2 demonstrate that DeePC is capable of handling this change in the system when given a sufficiently large data matrix and the right hyperparameter tuning. Also, as shown in Fig. 3.3, DDeePC can also stabilize this switching system. DeePC, however, has a smaller working region when the size of the change in the system is increased (as shown in Fig. 3.4), whereas DDeePC performs nearly equally as shown in Fig. 3.5.

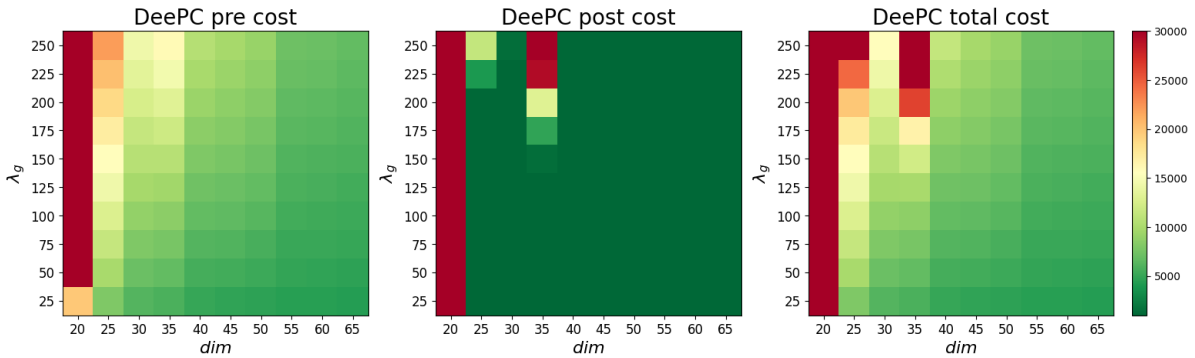


Figure 3.2: The objective cost of DeePC on a small switching system

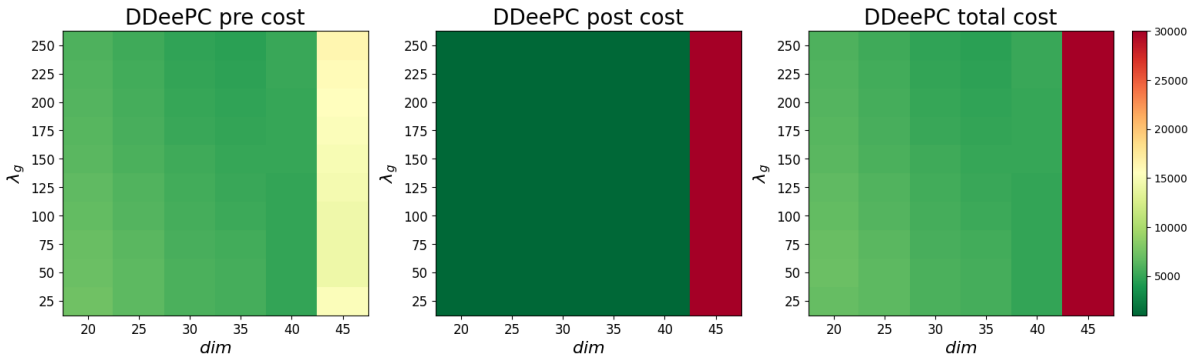


Figure 3.3: The objective cost of DDeePC on a small switching system

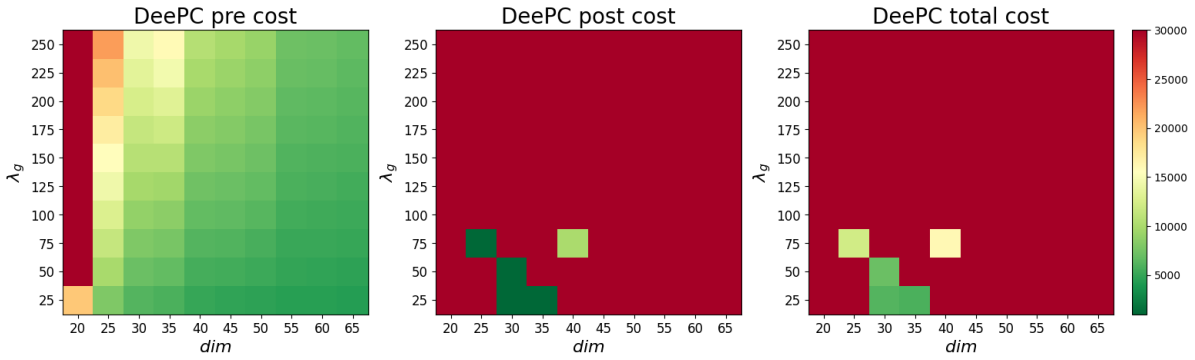


Figure 3.4: The objective cost of DeePC on a large switching system

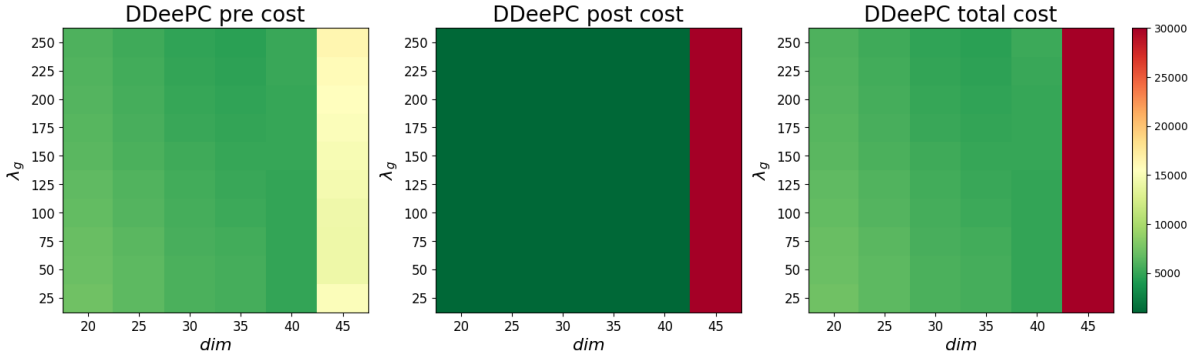


Figure 3.5: The objective cost of DDeePC on a large switching system

Continuously changing system

Instead of switching to a new dynamic as in the previous experiment, we gradually evolve the system over time to add more instability. This behaviour can represent the accumulation of dust on the rack, which might lead to an increase in temperature on the rack. In other words, this change will decrease the stability of the system. We utilize the logarithm function to replicate the slow change in order to model this phenomenon as follows:

$$a_1(k) = a_1(0) + 0.1\log(k+1)$$

The results in Fig. 3.6 demonstrate that DeePC cannot stabilize the system with any tuning of hyperparameters by illustrating all unstable systems in post cost. On the other hand, DDeePC

can still track this slow change, allowing it to regulate this system with the proper hyperparameters. For DDeePC, we can also observe that if the number of columns is too high (i.e. more than 40), the predictor matrix from SVD will become not slim enough (i.e. there are more rows than a column), which will lead to poor performance for the GROUSE algorithm to converge. As a result, DDeePC is unable to track and stabilize the system.

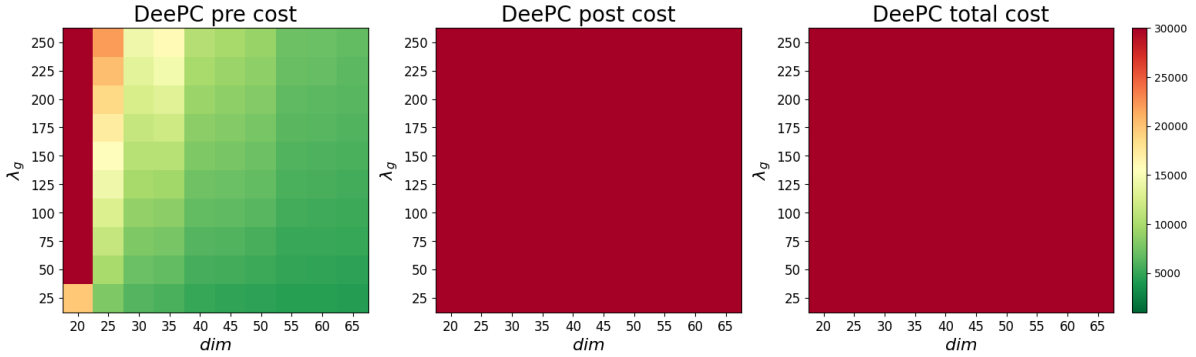


Figure 3.6: The objective cost of DeePC on a continuously changing system

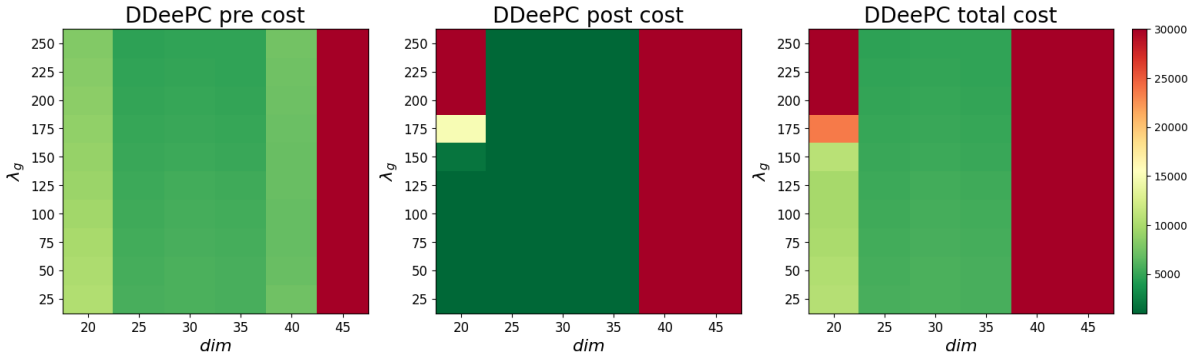
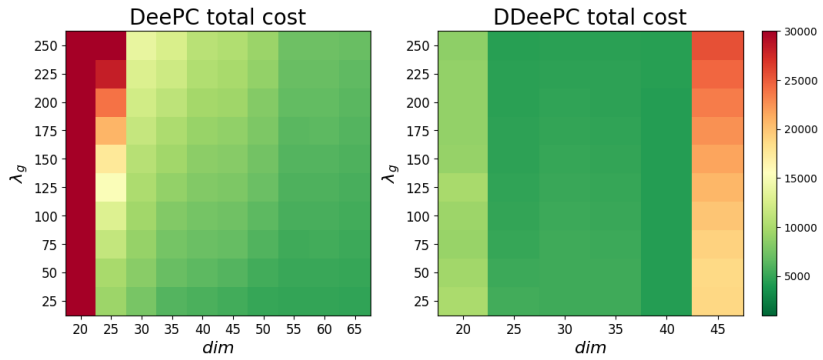


Figure 3.7: The objective cost of DDeePC on a continuously changing system

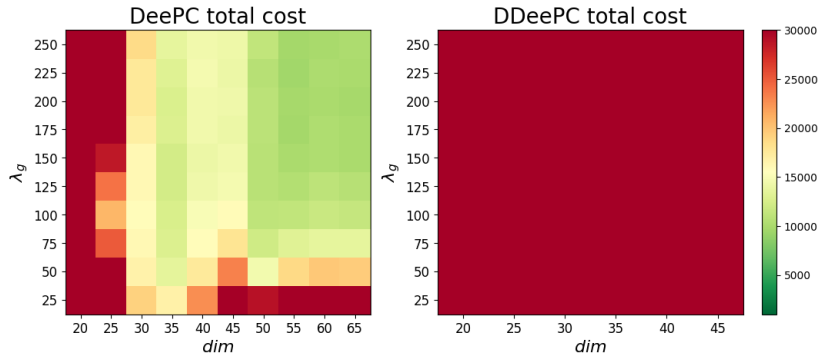
3.1.2 Experiment II: Noisy System

We now introduce measurement noise into the system in this experiment. Both the data collection procedure and the simulation process will incorporate this noise. We employ the Gaussian noiseÂ as measurement noise and then conduct the experiment with two sizes of noise in order to examine how noise affects both the DeePC and DDeePC algorithms. The zero mean ($\mu = 0$) and small standard deviation ($\sigma = 0.2$) noise is used in the first experiment. Both DeePC and DDeePC are capable of stabilizing the noisy system, as illustrated in Fig. 3.8. However, when we increase the size of the noise to $\sigma = 1.0$, DeePC has a smaller working region and DDeePC cannot stabilize the system at all tuning hyperparameters that were tested. This experiment demonstrates the DDeePC method's drawback, that it is sensitive to noise in this particular set of experiments. The outcome might be caused by the algorithm that updates the subspace using the observation vector's past data; as a result, DDeePC might converge to the incorrect subspace, leading to an unstable system.



(a) DeePC (b) DDeePC

Figure 3.8: The objective cost of DeePC and DDeePC in a low noise regime



(a) DeePC (b) DDeePC

Figure 3.9: The objective cost of DeePC and DDeePC in a large noise regime

Chapter 4

Conclusion

In this project, we have proposed an adaptive data-driven control, DDeePC, combining a subspace tracking algorithm, GROUSE, with DeePC. The algorithm works by using a series of measurement data to update the predictor model in the optimization problem. We also propose a way to modify GROUSE to incorporate it into this controller. The key advantage of this approach is the ability to adapt the optimal control rule based on the current data from the system, which allows it to stabilize the evolving system.

The standard DeePC can only stabilize the small change in the system, as demonstrated by the numerical experiments in the small amplitude switching system case, while it cannot stabilize the large change in the system (as shown in large amplitude switching and continuously changing system). DeePC can handle the tiny change system utilizing the regularization term. This regularization term, however, is insufficient to stabilize the system for the more significant change. The reasons are that the predictor matrix is fixed from the data-collecting process, thus the algorithm is unable to perceive the change in the system. On the other hand, DDeePC demonstrates the capability to track a variety of changes in slow time-varying systems utilizing the subspace tracking algorithm, allowing it to regulate systems to the origin.

The main drawback of DDeePC is that it is sensitive to noisy systems, despite performing better than DeePC in changing systems. DDeePC begins to malfunction when the noise-to-signal ratio is too high. The adaptive technique may converge to the incorrect subspace since the observation vector is built from both the past and the present data. Therefore, DDeePC is unable to stabilize a highly noisy system. Future work can address this problem by substituting different subspace tracking techniques for GROUSE. For instance, the work from GRASTA [11] provides a robust subspace tracking approach to handle outliers, which may solve the noise issue. Furthermore, further research is needed to provide formal theory and rigorous evidence of stability because this project is currently in its preliminary stages. The more obvious benefits of this strategy need to also be shown by applying this algorithm to a real-world system.

Bibliography

[1] Laura Balzano, Robert D. Nowak, and Benjamin Recht. "Online identification and tracking of subspaces from highly incomplete information". In: *2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)* (2010), pp. 704–711.

[2] Julian Berberich et al. "Data-Driven Model Predictive Control With Stability and Robustness Guarantees". In: *IEEE Transactions on Automatic Control* 66.4 (2021), pp. 1702–1717.

[3] Julian Berberich et al. "Robust data-driven state-feedback design". In: *2020 American Control Conference (ACC)*. IEEE, 2020.

[4] *Coarse-ID Control*. `https://www.argmin.net/2018/05/11/coarse-id-control/`.

[5] Jeremy Coulson, John Lygeros, and Florian Dörfler. "Data-Enabled Predictive Control: In the Shallows of the DeePC". In: *2019 18th European Control Conference (ECC)*. 2019, pp. 307–312.

[6] Jeremy Coulson, John Lygeros, and Florian DÃ¶rfler. *Distributionally Robust Chance Constrained Data-enabled Predictive Control*. 2020.

[7] Claudio De Persis and Pietro Tesi. "Formulas for Data-Driven Control: Stabilization, Optimality, and Robustness". In: *IEEE Transactions on Automatic Control* 65.3 (2020), pp. 909–924.

[8] Florian DÃ¶rfler, Pietro Tesi, and Claudio De Persis. "On the Role of Regularization in Direct Data-Driven LQR Control". In: *2022 IEEE 61st Conference on Decision and Control (CDC)*. 2022, pp. 1091–1098.

[9] Ezzat Elokda et al. "Data-enabled predictive control for quadcopters". In: *International Journal of Robust and Nonlinear Control* 31 (July 2021).

[10] Michel Gevers. "Identification for Control: From the Early Achievements to the Revival of Experiment Design*". In: *European Journal of Control* 11.4 (2005), pp. 335–352.

[11] Jun He, Laura Balzano, and John C. S. Lui. "Online Robust Subspace Tracking from Partial Information". In: *CoRR* abs/1109.3827 (2011).

[12] HÃ¥kan Hjalmarsson. "From experiment design to closed-loop control". In: *Automatica* 41.3 (2005). Data-Based Modelling and System Identification, pp. 393–438.

[13] Linbin Huang et al. "Robust Data-Enabled Predictive Control: Tractable Formulations and Performance Guarantees". In: (May 2021).

[14] Juraj Kabzan et al. "Learning-Based Model Predictive Control for Autonomous Racing". In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 3363–3370.

[15] Douglas Leith and W.E. Leithead. "Survey of gain-scheduling analysis and design". In: *Int. J. Control* 73 (Jan. 2000), pp. 1001–1025.

[16] Ivan Markovsky et al. "Exact and Approximate Modeling of Linear Systems: A Behavioral Approach". In: 2006.

[17] Babatunde A. Ogunnaike. “A contemporary industrial perspective on process control theory and practice”. In: *Annual Reviews in Control* 20 (1996), pp. 1–8.

[18] Jan Willem Polderman and Jan C. Willems. *Introduction to Mathematical Systems Theory : A Behavorial Approach*. Vol. 26. Texts in Applied Mathematics. New York: Springer, 1998.

[19] J. Richalet. “Industrial applications of model based predictive control”. In: *Automatica* 29.5 (1993), pp. 1251–1274.

[20] Henk J. van Waarde et al. *Data informativity: a new perspective on data-driven analysis and control*. 2019.

[21] Felix Wegner. “Data-enabled Predictive Control of Robotic Systems”. en. Master Thesis. Zurich: ETH Zurich, 2021-04.

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