ETH zürich

ETH Library

Data-enabled Predictive Control of Robotic Systems

Master Thesis

Author(s):
Wegner, Felix

Publication date:
2021-04

Permanent link:
https://doi.org/10.3929/ethz-b-000486029

Rights / license:
In Copyright - Non-Commercial Use Permitted

This page was generated automatically upon download from the ETH Zurich Research Collection.
For more information, please consult the Terms of use.

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

AUTOMATIC CONTROL LABORATORY

Felix Wegner
Matrikel Number: 346163

Data-enabled Predictive Control of Robotic Systems

Master Thesis

Automatic Control Laboratory
Swiss Federal Institute of Technology (ETH) Zurich

Supervision

Jeremy Coulson
Dr. Mathias Hudoba de Badyn
Prof. Dr. John Lygeros
Prof. Dr. Johann Sebastian Trimpe

April 2021

Contents

Abstract

Autonomous driving and highly automated manufacturing processes are only two examples for rapidly increasing system complexity. At the same time, almost unlimited connectivity produces never before seen amounts of data. As a consequence, the spotlight of control community turns more and more to data-driven approaches. Data-enabled Predictive Control (DeePC) is such an approach. Built on behavioural systems theory, it refers to Willems' Fundamental Lemma and replaces the parametric state-space model at the core of the Model Predictive Control framework by data trajectories. While benefiting from advantages of the MPC framework, like including safety constraints, DeePC is purely based on measured data and independent of any parametric system model. Unfortunately, the mathematical foundations of DeePC are strongly tied to linear time-invariant systems. Hence, challenges arise, when applying DeePC to nonlinear systems. Within this thesis, we explore nonlinear extensions of DeePC and derive heuristics for its hyperparameters by means of simulation. Using these insights, we step into the real world and use DeePC for the control of a Menzi Muck M545 12 t walking excavator. It is the first time, that DeePC is successfully applied to a strongly nonlinear real world mechanical system of such dimensions. After serving a proof of concept, we process an experimental study, and investigate DeePC's performance on the real world machine in terms of tracking accuracy and robustness.

Nomenclature

Symbols

φ	angle	$[rad]$
ϑ	angle	$[rad]$
$\mathcal{B}$	behaviour of a system	$[-]$
$\mathcal{H}$	Hankel matrix	$[-]$
$\mathcal{L}$	dynamical system	$[-]$
λ_g	penalty on norm of decision vector	$[-]$
λ_y	penalty on norm of slack variable	$[-]$
λ_u	penalty on norm of input slew rate	$[-]$
λ_a	penalty on norm of acceleration	$[-]$
a	acceleration	$[\frac{m}{s^2}]$
A	system matrix	$[-]$
B	input matrix	$[-]$
C	output matrix	$[-]$
d	damping	$[Nm \cdot s]$
D	feedthrough matrix	$[-]$
F	external force	$[N]$
g_z	gravitational constant	$[\mathrm{N}/\mathrm{kg}]$
g	decision vector	$[-]$
K	Kinetic Energy	$[J]$
L	Lagrange function	$[-]$
m	mass	$[kg]$
M	mass	$[kg]$
N	prediction horizon	$[-]$
l	length	$[m]$
Q	output cost matrix	$[-]$
R	input cost matrix	$[-]$
R_f	Rayleigh dissipation function	$[-]$
t	time	$[s]$
T	trajectory length	$[Nm]$
u	input vector	$[-]$
U	input matrix	$[-]$
V	Potential Energy	$[J]$
x	state vector	$[-]$
y	output vector	$[-]$
Y	output matrix	$[-]$

Indicies

a	accelerations
avg	average
cos	cosinus
d	discretized
delay	delay
f	future
g	decision vector
ini	initial
ISE	initial state estimation
max	maximum
obj	objective
p	past
prec	precision
sin	sinus
solve	solve
tr	tracking reference
u	inputs
y	outputs

Acronyms and Abbreviations

2DOF	two degrees of freedom
4DOF	four degrees of freedom
DeePC	Data-enabled predictive Control
ETH	Eidgenössische Technische Hochschule
HEAP	Hydraulic Excavator for an Autonomous Purpose
LQG	Linear Quadratic Gaussian
LQR	Linear Quadratic Regulation
LTI	Linear time-invariant
MPC	Model Predictive Control
RL	Reinforcement Learning

Chapter 1

Introduction

This chapter introduces the motivation for the research about Data-enabled Predictive Control (DeePC). Furthermore, the problem statement provides a collection of unsolved research questions related to DeePC and its application to nonlinear systems. Following up on the problem statement, the last sub-chapter highlights the contributions achieved within the scope of this thesis.

1.1 Motivation

In the recent decade, the topics of automation and big data are ubiquitous. Autonomous vehicles change the way we experience mobility [19], highly automated manufacturing processes reach new levels of productivity [54], and almost unlimited connectivity leads to never before seen amounts of data [18]. In the context of control engineering, this means that systems become more complex. At the same time, the increased availability and accessibility of data is an additional asset for the development of control algorithms [32]. As a consequence of these two trends, the control community gets more and more involved in the development of data-driven approaches. Data-driven approaches allow independence from parametric models, which is particularly beneficial when first-principle models are not obtainable or model derivation is too time-consuming and costly due to system complexity [20].
In 2019, a research group around F. Dörfler and J. Lygeros at the Automatic Control Laboratory of ETH Zürich presented the novel DeePC algorithm. As illustrated in [7], [8], and [21], DeePC appears to be a promising data-driven solution to several challenges, including the task of reference tracking. Reference tracking is a fundamental task in control engineering and spans a wide range of applications like unmanned aerial vehicle control [43], precision manufacturing [60], as well as intelligent construction sites [37] [45] [56].
Motivated by the environment of industry 4.0 and automated construction sites, this thesis investigates the performance of DeePC for the task of reference tracking on the use cases of a simulated 2-link robotic arm, as well as a real world Menzi Muck M545 12 t excavator.
Within this thesis, we first introduce the experimental setup in Chapter 2. Consecutively, in Chapter 3, we outline representative examples for model-based and data-driven control approaches in the context of robotic reference tracking. Referring to this literature review, we bring DeePC into line with the current state of the art. Focusing on DeePC, in Chapter 4, we explain the mathematical preliminaries, introduce the DeePC algorithm, and apply it to the simulation of a linearized system. In Chapter 5, we enter the nonlinear domain and refer to a simulation of a two-link robotic arm and a 12 t excavator, in order to derive heuristics for the hyperparameters of DeepC when applied to nonlinear systems. Using the insights from simulation, we then apply DeePC to a real world excavator. It is the first time that DeePC is used to control a mechanical system of such dimensions. Hence, in Chapter 6, we provide a proof of concept for the application of DeePC to strongly nonlinear mechanical systems. In the end, Chapter 7 summarizes the contents of this thesis, concludes on the key findings, and provides ideas for future research related to DeePC.

1.2 Problem Statement

The DeePC algorithm is based on mathematical preliminaries, which are valid for linear time-invariant (LTI) systems. Besides others, DeePC refers to the so-called Fundamental Lemma, which states that the behaviour of a LTI system can be fully described by data trajectories. Hence, DeePC sets up on this Lemma and is able to represent an algorithm, which is purely based on data and independent of any parametric system model. While providing certain guarantees for LTI systems, challenges emerge when being confronted with nonlinear systems. As it is not possible to refer back to the underlying concepts, there are several open points about DeePC and its behaviour when being applied to nonlinear systems. In the nonlinear domain, it is not finally clarified how data collection influences the behaviour of the DeePC algorithm and how to quantify the suitability of data for the purposes of the DeePC algorithm. Furthermore, DeePC introduces several hyperparameters used for, e.g., regularization. These hyperparameters need further investigation in order to outline their influence on the algorithm and their relation amongst each other, particularly for DeePC being applied to real world systems. For real world systems, the impact of nonlinearity becomes crucial. Additionally to the nonlinearity, it needs clarification on how well the DeePC algorithm copes with noise, time delays and other real world symptoms. Hence, a proof of concept for DeePC applied to a strongly nonlinear system is necessary.

1.3 Contribution

Within this thesis, we outline the evolution of the DeePC algorithm from a simulated LTI system to a real-world application on a strongly nonlinear system. On the journey to controlling a real world 12 t excavator by DeePC, the three main contributions within this thesis are:

- We confirm feasibility of applying DeePC to strongly nonlinear real world systems, by providing a proof of concept built on the use case of a Menzi Muck M545 excavator. It is the first time that DeePC is successfully applied to a mechanical real world machine of such dimensions and complexity.
- By incorporating a DeePC controller to the M545 excavator, we grant access to an ideal testing environment for future research. Being able to take recourse to a real world system might be a major advantage for the further development of the still young DeePC algorithm.
- We use simulations to derive heuristics for the hyperparameters of DeePC, and confirm these heuristics on the real machine by processing an experimental study. Furthermore, we provide a routine for efficient hyperparameter tuning.

By serving the proof of concept for a real nonlinear system, we path the way for testing DeePC on further real world systems. Additionally, we can refer to the experimental study in order to assess DeePC's capabilities when facing noise and time delays.

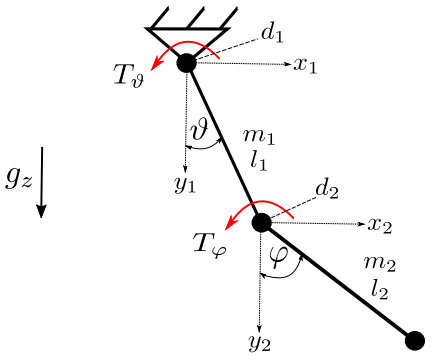
Chapter 2

Experimental Setup

This chapter introduces the experimental setup of this thesis consisting of a 2-link robotic arm and a Menzi Muck M545 excavator. We develop a python-embedded simulation of a 2-link robotic arm, which is then used in two versions. First, a linearized version of the 2-link helps to investigate DeePC when applied to a basic linear system. Second, since the 2-link can be interpreted as a simplification of the M545 excavator arm, a nonlinear version is used as an ideal starting point for investigations in the nonlinear domain.
Compared to the 2-link, the Menzi Muck M545 excavator is a drastically more complex system. For the M545, two high-fidelity simulations are provided, as well as the physical real world machine. The physical excavator ultimately serves as an exemplary proof of concept for the application of DeePC on a strongly nonlinear mechanical system.

2.1 2-Link Robotic Arm

For the purpose of initial investigations we refer to the example of a 2-link robotic arm. This setup allows us to derive a linearized version of it and is well-described in literature, which makes it an ideal starting point for our experiments. The simulation of the robotic arm developed within the scope of this thesis consists of two links and two damped joints. The user can apply a torque to each joint individually. Figure 2.1 illustrates the system schematically, while equations (2.1) - (2.4) describe the kinematics of the system.



ϑ, φ: joint angles
T_ϑ, T_φ: joint torques
d_1, d_2: joint damping
m_1, m_2: link masses
l_1, l_2: link lengths
g_z: gravitational constant

Figure 2.1: Schematic illustration of the two-link robotic arm.

$$x_1 = l_1 \cdot sin(\vartheta) \tag{2.1}$$

$$x_2 = x_1 + l_2 \cdot sin(\varphi) \tag{2.2}$$

$$y_1 = -l_1 \cdot cos(\vartheta) \tag{2.3}$$

$$y_2 = y_1 - l_2 \cdot cos(\varphi) \tag{2.4}$$

The trigonometric functions are linearized via Taylor series approximation under the assumption of small angles [48] and described by equations (2.5) - (2.9).

$$sin(\vartheta) = \vartheta \tag{2.5}$$

$$sin(\varphi) = \varphi \tag{2.6}$$

$$cos(\vartheta) = 1 - \vartheta^2/2 \tag{2.7}$$

$$cos(\varphi) = 1 - \varphi^2/2 \tag{2.8}$$

$$cos(\varphi - \vartheta) = 1 - (\varphi^2 - \vartheta^2)/2 \approx 1 \tag{2.9}$$

For the derivation of the equations describing the linearized 2-link we use the Euler-Lagrangian-Method extended by an additional term representing the Rayleigh dissipation function. The Lagrangian term L is defined as the difference of kinetic energy K and potential energy V as illustrated in (2.10). A distinction between the two degrees of freedom ϑ and φ leads to one differential equation each, defined in (2.11) and (2.12), where R_f represents the Rayleigh dissipation function and F is an external force.

$$L = K - V \tag{2.10}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\vartheta}}\right) - \frac{\partial L}{\partial \vartheta} + \frac{\partial R_f}{\partial \dot{\vartheta}} = F_\vartheta \tag{2.11}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\varphi}}\right) - \frac{\partial L}{\partial \varphi} + \frac{\partial R_f}{\partial \dot{\varphi}} = F_\varphi \tag{2.12}$$

The system's kinetic and potential energy as well as the Rayleigh dissipation function are defined in (2.13) - (2.16).

$$K = \frac{1}{2}(Ml_1^2\dot{\vartheta}^2 + m_2 l_1^2 \dot{\varphi}^2) + m_2 l_1 l_2 \dot{\varphi}\dot{\vartheta} \tag{2.13}$$

$$V = Mg_z l_1 \frac{\vartheta^2}{2} + m_2 g_z l_2 \frac{\varphi^2}{2} \tag{2.14}$$

$$R_f = \frac{1}{2}(d_1\dot{\vartheta}^2 + d_2\dot{\varphi}^2) \tag{2.15}$$

$$M = m_1 + m_2 \tag{2.16}$$

The previously introduced equations followed by mathematical operations allow the derivation of the system's second-order differential equations of motion, (2.17) and (2.18), describing the two degrees of freedom ϑ and φ.

$$\ddot{\vartheta} = \frac{m_2}{m_1}\frac{g_z}{l_1}\varphi + \frac{d_2}{m_1 l_1 l_2}\dot{\varphi} - \frac{1}{m_1 l_1 l_2}F_\varphi - \frac{Mg_z}{m_1 l_1}\vartheta - \frac{d_1}{m_1 l_1^2}\dot{\vartheta} + \frac{1}{m_1 l_1^2}F_\vartheta \tag{2.17}$$

$$\ddot{\varphi} = \frac{M}{m_1}\frac{g_z}{l_2}\vartheta + \frac{d_1}{m_1 l_1 l_2}\dot{\vartheta} - \frac{1}{m_1 l_1 l_2}F_\vartheta - \frac{Mg_z}{m_1 l_2}\varphi - \frac{Md_2}{m_1 m_2 l_2^2}\dot{\varphi} + \frac{M}{m_1 m_2 l_2^2}F_\varphi \tag{2.18}$$

Using equations (2.17) and (2.18), we can now represent the linearized 2-link (point of linearization: $\vartheta = 0, \varphi = 0$) as continuous time-invariant state-space representation of the form

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{2.19}$$

where

$$x(t) = \begin{pmatrix} \vartheta(t) \\ \dot{\vartheta}(t) \\ \varphi(t) \\ \dot{\varphi}(t) \end{pmatrix}, \quad u(t) = \begin{pmatrix} F_{\vartheta}(t) \\ F_{\varphi}(t) \end{pmatrix}, \tag{2.20}$$

$$\mathrm{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{Mg_z}{m_1 l_1} & -\frac{d_1}{m_1 l_1^2} & \frac{m_2 g_z}{m_1 l_1} & \frac{d_2}{m_1 l_1 l_2} \\ 0 & 0 & 0 & 1 \\ \frac{Mg_z}{m_1 l_2} & \frac{d_1}{m_1 l_1 l_2} & -\frac{Mg_z}{m_1 l_2} & -\frac{Md_2}{m_1 m_2 l_2^2} \end{bmatrix}, \tag{2.21}$$

$$\mathrm{B} = \begin{bmatrix} 0 & 0 \\ \frac{1}{m_1 l_1^2} & -\frac{1}{m_1 l_1 l_2} \\ 0 & 0 \\ -\frac{1}{m_1 l_1 l_2} & \frac{M}{m_1 m_2 l_2^2} \end{bmatrix}, \tag{2.22}$$

$$\mathrm{C} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \tag{2.23}$$

$$\mathrm{D} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}. \tag{2.24}$$

Finally, by discretizing the state-space representation and the related matrices, we derive the discrete time-invariant state-space representation of the form

$$\begin{aligned} x(k+1) &= A_d x(k) + B_d u(k) \\ y(t) &= C_d x(k) + D_d u(k) \end{aligned} \tag{2.25}$$

which is then implemented and used for the simulation of the linearized 2-link robotic arm.

At the same time, without applying a linearization, we use the introduced equations to derive the equations of motion for the nonlinear state-space representation of the 2-link robotic arm. For simulation, we use the basic nonlinear state-space representation

$$\begin{aligned} \dot{x} &= f(t, x(t), u(t)) \\ y(t) &= x(t) \end{aligned} \tag{2.26}$$

where

$$x(t) = \begin{pmatrix} \vartheta(t) \\ \dot{\vartheta}(t) \\ \varphi(t) \\ \dot{\varphi}(t) \end{pmatrix}, \quad u(t) = \begin{pmatrix} F_{\vartheta}(t) \\ F_{\varphi}(t) \end{pmatrix}, \tag{2.27}$$

$$f(t, x(t), u(t)) = \begin{pmatrix} \dot{\vartheta} \\ \\ (m_2 + m_1(1 - z_{cos}^2))^{-1} \cdot [-m_2 z_{sin} z_{cos} \dot{\vartheta}^2 \\ +m_2 g_z z_{cos} sin(\varphi) + \frac{d_2 z_{cos} \dot{\varphi}}{l_1 l_2} - \frac{z_{cos} F_{\varphi}}{l_1 l_2} \\ -\frac{m_2 l_2 z_{sin} \dot{\varphi}^2}{l_1} - \frac{g_z M sin(\vartheta)}{l_1} \frac{d_1 \dot{\vartheta}}{l_1^2} + \frac{F_{\vartheta}}{l_1^2}] \\ \\ \dot{\varphi} \\ \\ (m_1 + m_2(1 - z_{cos}^2))^{-1} \cdot [m_2 z_{sin} \dot{\varphi}^2 + \\ \frac{g_z M z_{cos} sin(\vartheta)}{l_2} + \frac{d_1 z_{cos} \dot{\vartheta}}{l_1 l_2} - \frac{z_{cos} F_{\vartheta}}{l_1 l_2} + \frac{M l_1 z_{sin} \dot{\vartheta}^2}{l_2} \\ -\frac{g_z M sin(\varphi)}{l_1} - \frac{d_2 M \dot{\varphi}}{m_2 l_2^2} + \frac{M F_{\varphi}}{m_2 l_2^2}] \end{pmatrix}, \tag{2.28}$$

$$\begin{aligned} z_{cos} &= \cos(\vartheta - \varphi), \\ z_{sin} &= \sin(\vartheta - \varphi). \end{aligned} \tag{2.29}$$

2.2 Menzi Muck M545 Excavator

The second experimental setup used within this thesis is a Menzi Muck M545 excavator provided by Robotic Systems Lab of ETH Zurich within the HEAP [41] project. Advancing from the two-link robotic arm, the M545 allows us to do research on a comparable system, while significantly increasing complexity. Controlling the arm of this 12 t walking excavator serves as use case for a proof-of-concept of the DeePC control algorithm on a strongly nonlinear mechanical system.

The arm of the M545 has four joints. Three of them (boom, dipper, shovel) are revolute joints, while the telescope joint is a linear one. For controlling the arm, we apply velocity inputs at each joint, which are then converted to valve set points by an additional low-level controller (see Section 6.1). We use a coordinate system oriented on the cabin for our research. Figure 2.2 shows the excavator and illustrates the links as well as the coordinate system.

Besides the non-linearity, a time delay between control command and actuation due to the hydraulics of the system represents a further challenge when controlling the M545. Depending on operating point and moving direction, the time delay can be up to 0.7 s [11]. Additionally, one further point to take into account when applying DeePC to the M545, is the increased dimension of the system. As a standard industrial robot is able to operate a load of roughly 150 kg, the M545 can handle a load of 3000 kg [25]. This highlights the scale of the occurring torques and moments of inertia when operating this system and outlines the importance of an accurate and smooth control.

The M545 allows us to test several different setups by freezing joints. For the experiments within this thesis, we mostly control boom and dipper joint for the task of reference tracking, and fix the shovel and telescope joints. Nevertheless, we use the shovel and telescope joints to derive different versions of the system by varying the fixed shovel angle and telescope stroke, respectively.

A major help on the way to applying DeePC on the M545 are two high-fidelity simulations of the excavator. The simulations are based on RaiSim/RaiSimPY (see Appendix A) and are a very accurate representation of the real-world system. Their main difference is, besides the programming language, the way of controlling the excavator arm. For the python-embedded simulation we control the joint torques. In contrast, we use the joint velocities as control inputs in the C++-embedded simulation. Both simulations provide an ideal testing environment, providing insights not only on tracking accuracy, but system oscillations as well as joint, contact and wheel forces. Despite their very advanced maturity level, the simulations do not cover the hydraulics-caused time-delay between command and actuation of the real machine. Therefore, the influence of the time-delay cannot be evaluated by means of simulation and needs to be investigated on the real

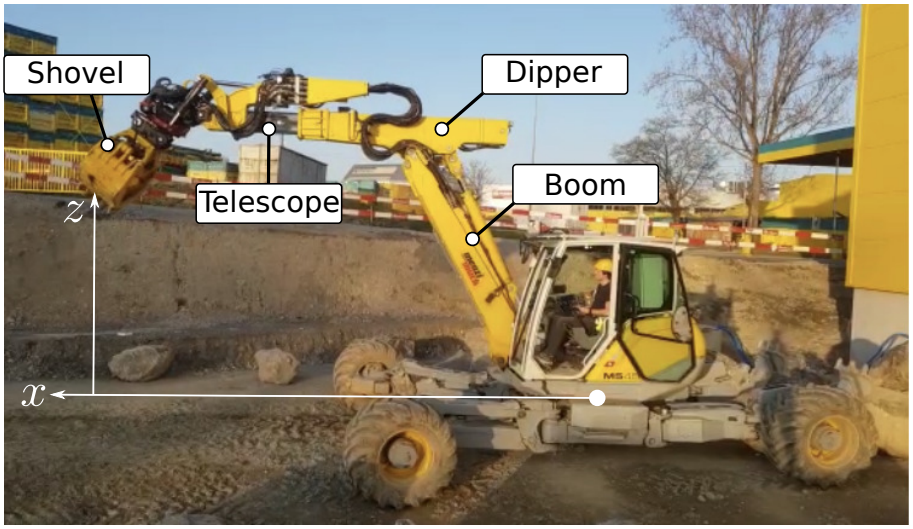


Figure 2.2: Menzi Muck M545 12 t walking excavator

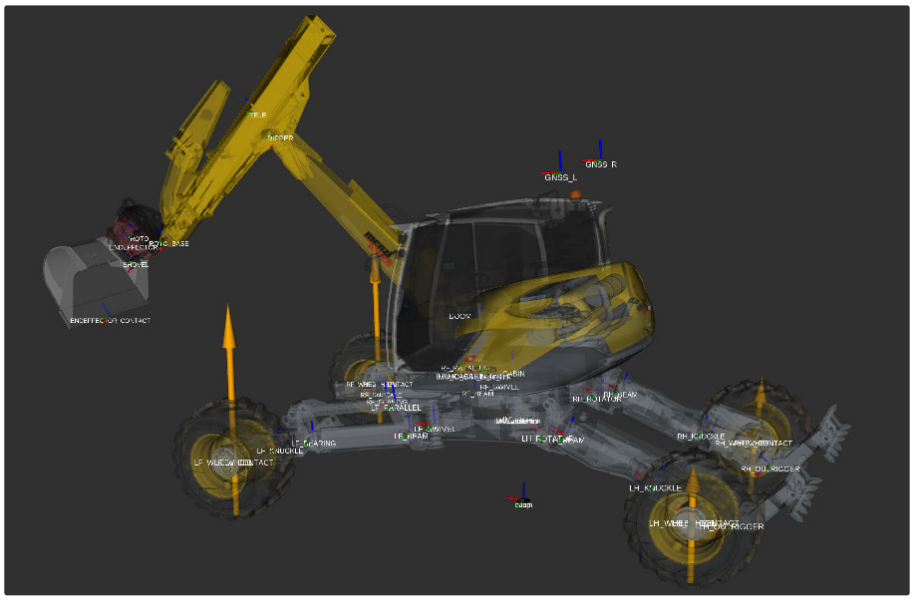


Figure 2.3: High-fidelity simulation of the M545. The arrows indicate the wheel forces and therefore provide a good indication for system oscillations due to to inappropriate sequences of control inputs.

machine. Figure 2.3 illustrates the simulated M545 excavator. For the the implementation of DeePC, we measure the following data of the excavator:

- Generalized angles of boom, dipper, telescope, and shovel joint
- x-, and z-coordinate of the shovel contact point
- Inputs at boom, dipper, telescope, and shovel joint in the form of torque or velocity commands

Chapter 3

State of the Art

This chapter provides insights on control approaches used for the task of reference tracking and robotic pose estimation. To structure the vast amount of literature, we distinguish between model-based and data-driven approaches. We illustrate both approaches by an exemplary control framework each. We highlight Model Predictive Control (MPC) for model-based approaches and choose Reinforcement Learning (RL) as an representative for data-driven approaches. We refer to these two methods, since literature outlines them as suitable methods for the task of reference tracking on the 2-link robotic arm as well as on the M545 excavator. Furthermore, we use the insights from the literature review to bring DeePC into line with the current state of the art.

3.1 From model-based to data-driven: Control Approaches for Reference Tracking and robotic Pose Estimation

The tasks of reference tracking and pose estimation are traditional challenges in control engineering. With the rise of robotics [47], these two tasks were moved to the spotlight of control engineering and gained massive attention. Hence, the control community proposes a large amount of solutions, covering rather traditional model-based approaches, as well as new data-driven methods. In the following section, we outline their application to robotics on the example of MPC and RL, and illustrate the most relevant advantages as well as disadvantages of the two frameworks.

Model-based Approaches

As an example for model-based approaches, we highlight Model Predictive Control. It is a widely celebrated algorithm for the task of reference tracking. The opportunity to include safety constraints in control design is a major reason for the popularity of MPC. Furthermore, MPC is a popular choice as it allows to be adjusted more easily for robustness and carries over most of its attractions to nonlinear systems [17]. An introduction to nonlinear MPC is provided in [15]. No matter if linear or nonlinear MPC, at the core of the MPC algorithm, there is an optimization problem (see Equation (4.2)).
As already indicated by the name *Model* Predictive Control, the MPC optimization problem relies on a parametric system model. This model is commonly expressed by a state-space representation based on the system matrix A, the input matrix B, the output matrix C, and the feedthrough matrix D. Introducing the receding horizon principle [31], we can formulate the standard MPC algorithm.

Algorithm 1: MPC (Algorithm 1 in [7])

Input: A, B, C, D, reference trajectory r, past input/output data (u, y), constraint sets $\mathcal{U}$ and $\mathcal{Y}$, weighting matrices Q and R.
1) Generate state estimate $x_0 = \tilde{\mathrm{x}}(t_0)$ based on past input/output data.
2) Solve optimization problem (4.2) for $(u_0^*, ..., u_{N-1}^*)$.
3) Apply input $(u(t), ..., u(t+s)) = (u_0^*, ..., u_s^*)$ for some $s \leq N-1$.
4) Set t to $t+s$ and update past input/output measurements.
5) Return to 1).

While the ability of including safety constraints is a major advantage, MPC's dependency on a precise model representing the system is a huge drawback. The model is needed to solve the optimization problem stated in Equation (4.2), as well as, if there is no exact measurement of the state, for the initial state estimate $x_0 = \tilde{\mathrm{x}}(t_0)$. Most commonly, the need for a model in the context of MPC is covered through system identification [50], which allows to synthesize a model based on offline observations of the system. In the context of more and more complex systems and applications, this can become a very challenging, time-intense and therefore costly procedure [16], which motivates research into the direction of less model-dependent control approaches.
Revisiting the idea of system identification, we direct the reader to [29] [39] [30] in order to illustrate that there are several variations of system identification and plenty of opportunities for combining it with control frameworks different than MPC. However, all of these approaches fall back to a parametric system model, which might be worth avoiding due to the above outlined reasons. Furthermore, besides the major drawback of dependency on a parametric model, non-parametric learning approaches like Gaussian Processes show the ability to outperform parametric models like illustrated in [44].

Data-driven Approaches

We present reinforcement learning as an example for data-driven approaches. Nowadays, RL is a popular non-parametric learning approach. In general, it is a framework for the problem of learning from interaction in order to achieve a goal [53]. In this framework, there is the so-called agent, which learns and decides about certain control actions. The agent itself interacts with the environment, which basically represents everything outside the agent. Together, their ongoing interaction is defined by the agent taking actions and the environment providing feedback. This means that the environment gives back a so-called reward to the agent based on a reward function at each discrete time step. While considering the state of the environment, the agent takes an action according to a control policy. The control policy represents a mapping from states to probabilities for selecting certain actions. It is updated with the purpose of serving the agent's ultimate goal, which is maximizing the reward given by the environment. Figure 3.1 illustrates a general representation of the basic idea behind reinforcement learning.
Note, that this is, in terms of the outcome, exactly the opposite of what we aimed for with MPC.

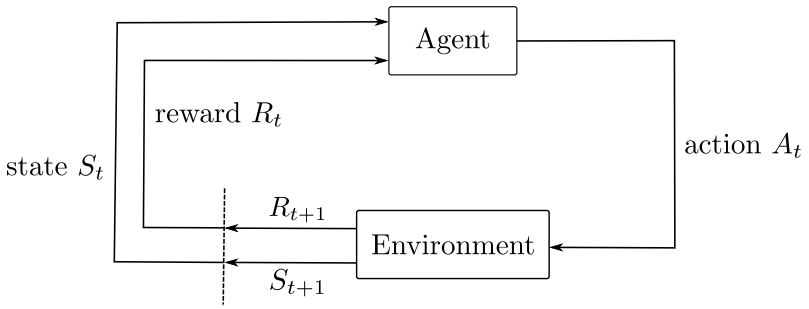


Figure 3.1: Agent-environment interaction in reinforcement learning [53]

While MPC minimizes costs defined by a cost function, reinforcement learning aims to maximize the reward as a result of a reward function. The essential underlying difference is, that MPC decides about its control actions based on a parametric system model, while RL chooses its control actions based on a learned, data-driven and model-free control policy. When setting up a reinforcement learning algorithm, major topics of interest include but are not limited to exploration-exploitation-balance, Markov-Decision-Processes, Temporal-Difference Learning, as well as on-/offline policy approximation. A comprehensive explanation is provided in [53].
Focusing on the application of reinforcement learning to robotics, [28] highlights that most of the challenges of reinforcement learning are particularly present when applied to a robotics setup. By processing an extensive survey, the authors of [28] give an excellent overview and state the most dominant robotics-related challenges of reinforcement learning as follows:

- Dimensionality: In 1957, in [4], Bellman established the *curse of dimensionality* when facing an exponential explosion of states and actions after exploring optimal control in discrete high-dimensional spaces. In relation to reinforcement learning this means, that the amount of data and computation grows exponentially when dimensions increase. This is quite a problem for robotics, since most of nowadays robotic applications deal with a high number of dimensions (consider the example of, e.g., human-like actuation [59]). As this is recognised as a major issue, there are several proposals for handling high dimensionality by means of, e.g., adaptive discretization [5], function approximation [53], as well as macro actions [3]. Nevertheless, it becomes clear, that when applying reinforcement learning to robotics it either needs consideration of some non-trivial extensions or a massive amount of data.

- Quality and quantity of real-world samples: When collecting data of the real-world robotic application, safe exploration is a major aspect [38]. Furthermore, reproducibility is a difficult task as experimental and environmental settings can change with the supervisor not having any influence (e.g. wind conditions). Besides that, most learning tasks need human supervision. Additionally, varying noise levels, delays in sensing and actuation, as well as a lack of full state information at all times lead to uncertainty and therefore decrease efficiency of the data collection. As a consequence, collecting real-world samples is complex, time-consuming, and costly, which makes sample efficiency, describing the ability of an algorithm to provide reasonable performance with a reduced amount of data, an essential characteristic of reinforcement learning algorithms [28].

- Under-modeling and model uncertainty: One way to decrease the dependency on real-world samples is to collect data from high-fidelity simulations. In order to build such a simulation, a model is needed. Besides the problem that such a model is not always easy to obtain or even not available at all, for reinforcement learning, there is also the problem of under-modelling and model uncertainty. Particularly for unstable tasks, small variations between simulation and real-world can lead to severe consequences [53].

- Goal specification: In robotics it is often intuitive to define the reward dependent on failure or success with the task. Anyhow, such a binary definition of a reward function most probably leads to very poor performance. As a solution, [33] introduces the concept of reward shaping, which gives the algorithm an idea of how close it was to a successful performance by introducing intermediate rewards [40]. Further solutions for an efficient design of reward functions are Inverse Reinforcement Learning proposed in [46], as well as a derivation of complex reward functions based on policy search techniques like illustrated in [61]. All in all, the proposed solutions highlight the relation between the complexity of the reward function and the complexity of the learning problem, and implicate that defining the reward function for reinforcement learning in robotics can be a tedious task [28].

Considering the applications covered within this thesis, a 2-link robotic arm and a Menzi Muck M545 excavator (see Sections 2.1, 2.2), we would like to direct the reader to [55], [10], and [11]. In [55], a method for robust model-free reinforcement learning based on multi-objective Bayesian Optimization is presented. The authors quantify the robustness of the policy by delay margin and

gain margin, and illustrate their findings in a sim-to-real and a pure hardware experiment on a Furuta pendulum. In [10], the authors introduce an algorithm called Locally Weighted Projection Regression, which allows learning the inverse kinematics of a humanoid robot. With their new approach they try to bypass the previously mentioned challenge of high dimensionality. Third, as we target an application of DeePC on a M545 excavator, we recommend [11]. In this paper, the author applies reinforcement learning to the M545 excavator in order to track a circular reference. The experiments performed in the paper serve as an orientation for the experiments on the M545 presented within this thesis.

3.2 Placing DeePC within the State of the Art

Based on the findings presented in Section 3.1, we can now summarize the positive and negative aspects of the investigated methods. For MPC, we learned that it is advantageous to have the opportunity to include safety constraints. At the same time, it became obvious that model-dependency should be avoided due to increasing system complexity and the related efforts and costs. Triggered by the rising accessibility of data [32], data-driven approaches are the logical reaction to that challenge. For the domain of data-driven approaches we highlighted reinforcement learning. It allows control algorithms independent of a parametric model, but reveals drawbacks related to the needed amount of data, the quality and quantity of real-world samples, under-modeling, as well as goal specification.
Combining the insights from our literature review, we identify an ideal controller for the task of reference tracking, which should be defined by the following characteristics:

- The controller should be able to operate the system with minimal knowledge of the system and not rely on a parametric system model.
- The controller should be able to respect safety constraints.
- The algorithm should have a high sample efficiency in order to reduce the needed amount of real-world samples.
- The goal specification of the algorithm should be possible in an intuitive and easy-to-realize manner.

As indicated in [7], the developments for DeePC start at exactly that description of an ideal controller for reference tracking. Since MPC is widely used and well-established, DeePC is set up on the basic framework of MPC. Hence, it is able to include safety constraints. Following up from that point, it becomes the next task, to replace the system model. Since reinforcement learning needs massive amounts of data, full state feedback, and is vulnerable to noise, the authors of [7] work in another direction on their way to the DeePC algorithm. Starting with linear systems, they base their algorithm on the groundbreaking work of Willems and his results on the representation of linear systems based on time series data presented in [57]. Aligned to the idea outlined in [36], the authors orient their perspective on behavioural systems theory. Therefore, they shift their focus from understanding a system in the form of an underlying model to rather understanding a system in the form of its behaviour. A next ingredient of the DeePC algorithm are the guarantees and characteristics coming with the idea of persistency of excitation addressed in [58]. In [35], the work of [57] and [58] is continued with a special focus on data-driven simulation and control. Related work also includes [42], presenting concepts for optimality, stability and robustness for data driven control, as well as [34] and [13], introducing the concepts for a data-driven Linear Quadratic Regulator (LQR) controller and a data-driven Linear Quadratic Gaussian (LQG) controller, respectively.
Evolving from the work presented in the above-cited papers, DeePC represents a control approach, which needs only minimal system knowledge, respects safety constraints, relies on drastically less data compared to reinforcement learning, and provides an intuitive goal definition in the form of a cost function [7]. The following Chapter 4 states the mathematical preliminaries for DeePC and consecutively introduces the algorithm for linear systems.

Chapter 4

Preliminaries

This chapter outlines the mathematical preliminaries of DeePC. Furthermore, it introduces the DeePC algorithm and illustrates the application of DeePC to the simulation of a linearized 2-link. Referring to the exemplary use case, it provides a direct comparison to MPC and confirms the theoretical guarantees provided by the mathematical preliminaries.

4.1 Mathematical Preliminaries

In this sub-chapter we provide the mathematical preliminaries for DeePC. We start with a definition of dynamical systems. Furthermore, we introduce the concept of persistency of excitation, which is of essential importance for DeePC. Third, we define controllability and observability of a dynamical system. Finally, the sub-chapter culminates with the introduction of the so-called Fundamental Lemma.

Dynamical Systems

Before introducing the definition of a dynamical system, it is important to note that all deliberations on the DeePC algorithm are made from the perspective of behavioural systems theory. In contrast to traditional control theory, which is based on a parametric system representation, this change in perspective leads to a non-parametric system representation. In other words, this means that one is not particularly interested in a system's explicit representation, but much more in its behaviour, meaning the subspace of the signal space in which the trajectories of the system live.
Setting up on the idea of behavioural systems theory, [7] defines a dynamical system as 3-tuple $(\mathbb{Z}_{\geq 0}, \mathbb{W}, \mathcal{B})$.

Definition 1. *A dynamical system is defined as 3-tuple $(\mathbb{Z}_{\geq 0}, \mathbb{W}, \mathcal{B})$. $\mathbb{Z}_{\geq 0}$ represents the discrete-time axis, $\mathbb{W}$ is a signal space, and $\mathcal{B} \subseteq \mathbb{W}^{\mathbb{Z}_{\geq 0}}$ is the behaviour of the system.*

As a consequence of this definition, we now define three properties of dynamical systems:

(i) $(\mathbb{Z}_{\geq 0}, \mathbb{W}, \mathcal{B})$ is *linear* if $\mathbb{W}$ is a vector space and $\mathcal{B}$ is a linear subspace of $\mathbb{W}^{\mathbb{Z}_{\geq 0}}$.

(ii) $(\mathbb{Z}_{\geq 0}, \mathbb{W}, \mathcal{B})$ is *time invariant* if $\mathcal{B} \subseteq \sigma\mathcal{B}$ where σ: $\mathbb{W}^{\mathbb{Z}_{\geq 0}} \to \mathbb{W}^{\mathbb{Z}_{\geq 0}}$ is the backward time shift defined by $(\sigma w)(t) = w(t+1)$ and $\sigma\mathcal{B} = \{\sigma w \mid w \in \mathcal{B}\}$.

(iii) $(\mathbb{Z}_{\geq 0}, \mathbb{W}, \mathcal{B})$ is *complete* if $\mathcal{B}$ is closed in the topology of pointwise convergence.

The class of systems satisfying (i) - (iii) is denoted by $\mathcal{L}^{m+p}$ with $m, p \in \mathbb{Z}_{\geq 0}$. Due to simplicity, this thesis denotes a dynamical system within the above defined class of systems only by its behaviour $\mathcal{B}$.
Having a look at the behaviour $\mathcal{B}$ of the system and remembering the importance of input/output samples, it becomes clear that the behaviour $\mathcal{B}$ of the system can be defined as the product space

of the two sub-behaviours $\mathcal{B}^u$ and $\mathcal{B}^y$. Here, $\mathcal{B}^u = (\mathbb{R}^m)^{\mathbb{Z}_{\geq 0}}$ represents the space of the inputs of dimension m and $\mathcal{B}^y \subseteq (\mathcal{R}^p)^{\mathbb{Z}_{\geq 0}}$ the space of the outputs of dimension p. Referencing [57], this also means for the truncated system behaviour $\mathcal{B}_T$ covering the trajectory of finite length T, that any trajectory $w \in \mathcal{B}_T$ is described by input/output samples written as $w = col(u, y)$ where $col(u, y) := (u^T, y^T)^T$.

Persistency of Excitation

Persistency of Excitation is an essential aspect for the development and application of DeePC. Basically speaking, if a signal is *persistently exciting*, the signal is long and rich enough to cause an excitation of the system, which leads to a output sequence that is representative for the system behaviour. This means that the signal spans the whole space of all trajectories the system is able to produce [58].
For the investigation of persistency of excitation it is necessary to introduce the concept of Hankel matrices. An introduction to Hankel matrices and their applications to system identification is presented in [14]. The structure of a Hankel matrix $\mathcal{H}_L(u)$ of order L for an input signal u of length T is given by

$$\mathcal{H}_L(u) = \begin{bmatrix} u_1 & u_2 & \dots & u_{T-L+1} \\ u_2 & u_3 & \dots & u_{T-L} \\ \vdots & \ddots & \ddots & \vdots \\ u_L & \dots & \dots & u_T \end{bmatrix}.$$

Based on the arrangement of the input signal u in the form of a Hankel matrix, [7] provides a mathematical definition for persistency of excitation.

Definition 2. *Let $L, T \in \mathbb{Z}_{\geq 0}$ such that $T \geq L$. The signal $u = col(u_1, ..., u_T) \in \mathbb{R}^{Tm}$ is persistently exciting of order L if the Hankel matrix $\mathcal{H}_L(u)$ is of full rank.*

Note already at this early point, that it is essential to ensure persistency of excitation of the used input signal in order to reach a sufficient performance of the DeePC algorithm.

Controllability, Observability and Lag of a System

Two further mathematical preliminaries needed for the DeePC algorithm are the terms controllability and observability.
The system characteristic of controllability is a precondition for the later introduced Fundamental Lemma. For an illustrative explanation as well as a mathematical definition, this text refers to [7]. From the perspective of behavioural systems theory, a controllable system is described as a system for which any two trajectories can be patched together in finite time.

Definition 3. *A behavioural system $\mathcal{B} \in \mathcal{L}^{m+p}$ is controllable if for every $T \in \mathbb{Z}_{\geq 0}$, $w^1 \in \mathcal{B}_T$, $w^2 \in \mathcal{B}$ there exists a $w \in \mathcal{B}$ and $T' \in \mathbb{Z}_{\geq 0}$ such that $w_t = w_t^1$ for $1 \leq t \leq T$ and $w_t = w_{t-T-T'}^2$ for $t \geq T + T'$.*

Furthermore, considering the classical input/output state representation captured in $\mathcal{B}(A, B, C, D)$, for our needs, the concept of observability is essential in order to identify the *lag* of the system. The lag is defined by the smallest integer $\ell \in \mathbb{Z}_{\geq 0}$ such that the observability matrix $\mathcal{O}_\ell(A, C) := col(C, CA, ..., CA^{l-1})$ has rank $n(\mathcal{B})$. Here, $n(\mathcal{B})$ represents the state dimension of a minimal representation.
Additionally to the lag, we introduce the lower triangular Toeplitz matrix $\mathcal{T}_N$ based on (A, B, C, D) as

$$\mathcal{T}_N(A, B, C, D) = \begin{bmatrix} D & 0 & \dots & 0 \\ CB & D & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ CA^{N-2}B & \dots & CB & D \end{bmatrix}.$$

Based on the lag and the lower triangular Toeplitz matrix, [7] formulates the following lemma:

Lemma 1 (Lemma 4.1 in [7]). *Let $\mathcal{B} \in \mathcal{L}^{m+p}$ and $\mathcal{B}(A, B, C, D)$ a minimal input/output/state representation. Let $T_{ini}, N \in \mathbb{Z}_{\geq 0}$ with $T_{ini} \geq \ell(\mathcal{B})$ and $col(u_{ini}, u, y_{ini}, y) \in \mathcal{B}_{T_{ini}+N}$. Then there exists a unique $x_{ini} \in \mathbb{R}^{n(\mathcal{B})}$ such that*

$$y = \mathcal{O}_N(A, C)x_{ini} + \mathcal{T}_N(A, B, C, D)u\,. \tag{4.1}$$

Equation (4.1) illustrates that the state y as a result of an input sequence u is unique, if there is a sufficiently long trajectory of system data $col(u_{ini}, y_{ini})$. Additionally, if A, B, C, D are given, the initial state x_{ini} can be computed.

The Fundamental Lemma

The *Fundamental Lemma* based on the work of J.C. Willems and his research group is the backbone of DeePC. Referring to the previously introduced mathematical preliminaries, [35] defines the Fundamental Lemma as follows:

Lemma 2 (Lemma 2 in [35]). *Let $\mathcal{B} \in \mathcal{L}^{m+p}$ be a controllable system. Let $T, t \in \mathbb{Z}_{\geq 0}$ and $w = col(u, y) \in \mathcal{B}$. Assume the input signal u to be persistently exciting of order $t + n(\mathcal{B})$. Then $colspan(\mathcal{H}_t(w)) = \mathcal{B}_t$.*

In other words, for linear time-invariant systems, the Fundamental Lemma proves that it is possible to exactly represent a system's behaviour by data trajectories without knowledge of any model.

4.2 Introduction of the DeePC Algorithm

The DeePC algorithm is an adaptation of the widely known Model Predictive Control algorithm. The essential difference is that DeePC is not based on a parametric model. In fact, DeePC is based on data trajectories and relies only on little system knowledge. This section outlines the evolution and structure of the basic DeePC algorithm conceptualized for linear systems.
We start with a review of MPC in order to introduce the DeePC algorithm beginning at its fundamentals. As illustrated in Section 3.1, for the task of reference tracking, MPC is a popular control framework as it allows to include safety constraints. We introduce the underlying MPC optimization problem as

$$\begin{aligned}
&\min_{u,x,y} \quad \sum_{k=0}^{N-1} \left(\|y_k - r_{t+k}\|_Q^2 + \|u_k\|_R^2\right) \\
\text{s.t.} \quad & x_{k+1} = Ax_k + Bu_k, \forall k \in \{0, ..., N-1\}, \\
& y_k = Cx_k + Du_k, \forall k \in \{0, ..., N-1\}, \\
& x_0 = \hat{x}(t), \\
& u_k \in \mathcal{U}, \forall k \in \{0, ..., N-1\}, \text{where } \mathcal{U} \subseteq \mathbb{R}^m \\
& y_k \in \mathcal{Y}, \forall k \in \{0, ..., N-1\}, \text{where } \mathcal{Y} \subseteq \mathbb{R}^p.
\end{aligned} \tag{4.2}$$

where

$$\|y_k - r_{t+k}\|_Q^2 = y_k^T Q y_k - y_k^T Q r_{t+k} - r_{t+k}^T Q y_k + r_{t+k}^T Q r_{t+k} \tag{4.3}$$

and

$$\|u_k\|_Q^2 = u_k^T R u_k. \tag{4.4}$$

In this optimization problem, there are the decision variables $u = (u_0, ..., u_{N-1})$, $x = (x_0, ..., x_N)$ and $y = (y_0, ..., y_{N-1})$. The estimated state $\hat{x}(t)$ at time step t equals $x(t)$ if the entire state is measured. The predicted state and output at time step $t + k$ are denoted by x_k and y_k with

$t \in \mathbb{Z}_{\geq 0}$ as the time when the optimization problem should be solved. Furthermore, $N \in Z_{>0}$ is the prediction horizon, $\mathcal{U}$ the input constraint set, $\mathcal{Y}$ the output constraint set, and $r_{t+k} \in \mathbb{R}^p$ is the reference at the time step $t+k$. Q and R represent the output cost matrix and control cost matrix, respectively [7]. Revising the structure of the MPC optimization problem in (4.2), it becomes clear, that it is based on a parametric model. In this case, the model is formulated as a state-space representation expressed via the matrices A,B,C and D.
Such a dependency on a parametric model is exactly what DeePC wants to avoid. Therefore it replaces the parametric state-space model by data trajectories, and in order to do so it refers to the previously introduced Fundamental Lemma (Lemma 2 in Section 4.1). Recalling the guarantees brought by the Fundamental Lemma, we know that the behaviour of a linear time-invariant system can be exactly represented via data trajectories if the Hankel matrix of the input signal is persistently exciting of sufficient order. Assuming an input sequence $u^d = \text{col}(u_1^d, ..., u_T^d)$ and a output sequence $y^d = \text{col}(y_1^d, ..., y_T^d)$ of length $T \geq (m+1)(T_{ini}N + n(\mathcal{B})) - 1$ generated by a controllable system and collected offline, the Hankel matrix $\mathcal{H}(u)$ needs to have rank of at least $T_{ini} + N + n(\mathcal{B})$, where $T_{ini} \geq \ell(\mathcal{B})$. After ensuring persistency of excitation of the input signal u^d by checking $\mathcal{H}_{T_{ini}+N}(u^d)$ for full row rank, we can split up the collected data in past and future data sets

$$\begin{pmatrix} U_p \\ U_f \end{pmatrix} = \mathcal{H}_{T_{ini}+N}(u^d), \quad \begin{pmatrix} Y_p \\ Y_f \end{pmatrix} = \mathcal{H}_{T_{ini}+N}(y^d) \tag{4.5}$$

where U_p and Y_p consist of the first T_{ini} block rows of $\mathcal{H}_{T_{ini}+N}(u^d)$ and $\mathcal{H}_{T_{ini}+N}(y^d)$, respectively, while U_f and Y_f consist of the last N block rows of $\mathcal{H}_{T_{ini}+N}(u^d)$ and $\mathcal{H}_{T_{ini}+N}(y^d)$, respectively. The p-denoted past data sets are used to estimate the initial state, and the f-denoted future data sets are used to predict the future trajectories. Recalling Lemma 2, the decision vector g is introduced for state estimation and trajectory prediction in Lemma 3.

Lemma 3 ([8]). *A trajectory $col(u_{ini}, u, y_{ini}, y)$ belongs to $\mathcal{B}_{T_{ini}+N}$ if and only if there exists $g \in \mathbb{R}^{T-T_{ini}-N-1}$ such that*

$$\begin{pmatrix} U_p \\ Y_p \\ U_f \\ Y_f \end{pmatrix} g = \begin{pmatrix} u_{ini} \\ y_{ini} \\ u \\ y \end{pmatrix}. \tag{4.6}$$

Furthermore, if $T_{ini} \geq \ell(\mathcal{B})$, Equation (4.1) defined in Lemma 1 provides a unique output y based on a unique state $x_{ini} \in \mathbb{R}^{n(\mathcal{B})}$. This means, that the initial state x_{ini} is implicitly predefined by $col(u_{ini}, y_{ini})$. Additionally, when considering the results presented in [35], it can be concluded that future trajectories departing from this initial state are predictable by including the system-generated data U_p, Y_p, U_f, Y_f. Since Lemma 1 allows to compute unique outputs y related to inputs u, this means for the task of reference tracking on LTI systems, that it is also possible to compute an input u for a targeted output y matching a reference r.
Given a prediction horizon $N \in Z_{>0}$, a reference $r = (r_0, r_1, ...) \in (\mathbb{R}^p)^{\mathbb{Z}\geq 0}$, past input/output data $\text{col}(u_{ini}, y_{ini}) \in \mathcal{B}_{T_{ini}}$, an output cost matrix $Q \in \mathbb{R}^{p\times p}$, a control cost matrix $R \in \mathbb{R}^{m\times m}$, an output constraint set $\mathcal{Y} \subseteq \mathbb{R}^p$, and an input constraint set $\mathcal{U} \subseteq \mathbb{R}^m$, we can now formulate the DeePC optimization problem as

$$\begin{aligned} \min_{g,u,y} \quad & \sum_{k=0}^{N-1} \left(\|y_k - r_{t+k}\|_Q^2 + \|u_k\|_R^2\right) \\ \text{s.t.} \quad & \begin{pmatrix} U_p \\ Y_p \\ U_f \\ Y_f \end{pmatrix} g = \begin{pmatrix} u_{ini} \\ y_{ini} \\ u \\ y \end{pmatrix} \\ & u_k \in \mathcal{U}, \forall k \in \{0, ..., N-1\}, \\ & y_k \in \mathcal{Y}, \forall k \in \{0, ..., N-1\}. \end{aligned} \tag{4.7}$$

where

$$\|y_k - r_{t+k}\|_Q^2 = y_k^T Q y_k - y_k^T Q r_{t+k} - r_{t+k}^T Q y_k + r_{t+k}^T Q r_{t+k} \tag{4.8}$$

and

$$\|u_k\|_Q^2 = u_k^T R u_k. \tag{4.9}$$

In this optimization problem, u and y are dependent decision variables. They rely on the fixed data matrices U_f and Y_f, respectively, as well as on the independent decision vector g. Note, that this formulation based on measured data trajectories allows us to replace the model and state estimation in (4.2) and hence makes the optimization problem of the DeePC algorithm completely model-free and purely based on data.
Based on the above described relations, Coulson et al. present the DeePC algorithm for linear systems.

Algorithm 2: DeePC (Algorithm 2 in [7])

Input: $col(u, y) \in \mathcal{B}_T$, reference trajectory $r \in \mathbb{R}^{Np}$, past input/output data $col(u_{ini}, y_{ini}) \in \mathcal{B}_{T_{ini}}$, constraint sets $\mathcal{U}$ and $\mathcal{Y}$, weighting matrices Q and R.
1) Solve (4.7) for g^*.
2) Compute the optimal input sequence $u^* = U_f g$.
3) Apply input $(u(t), ..., u(t+s)) = (u_0^*, ..., u_s^*)$ for some $s \leq N-1$.
4) Set t to $t+s$ and update u_{ini} and y_{ini} to the T_{ini} most recent input/output measurements.
5) Return to 1).

4.3 DeePC applied to linear Systems

For an illustration of the performance of DeePC on linear systems we use the developed simulation setup of the linearized 2-link robotic arm described in Section 2.1. We focus on the task of reference tracking. In particular, we would like the 2-link to start from a vertical downward position and move to a constant position as schematically illustrated in Figure 4.1. Putting DeePC and MPC in contrast to each other, Figure 4.2 gives a high-level overview of the two approaches. DeePC represents the system and its behaviour via data trajectories, while MPC relies on first-principle models or another form of parametric model. This becomes also clear when having a look at the mathematical formulation describing the system within the optimization problem. While DeePC proposes a data-based solution (see (4.7)), MPC relies on a linear state-space formulation referring to the previously derived system matrices (see (4.2)). Despite these fundamental differences, both approaches lead to an identical behaviour for LTI systems.

The data collection for the DeePC algorithm is illustrated in Figure 4.3. Note, that the inputs are completely random and therefore the trajectories represented by the angles ϑ and φ don't follow a certain scheme either. Out of the presented 25 s of data collection, we use the first 10 s of data to build the underlying Hankel matrices for the DeePC algorithm. The block-rows of our Hankel matrices are of size two, since we have two angles, which are stored in the output Hankel matrix,

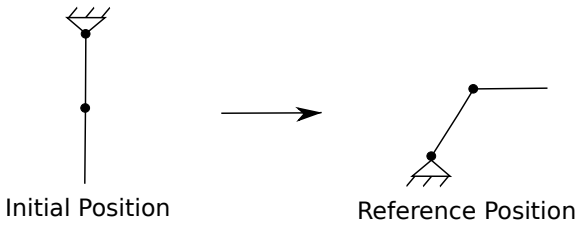


Figure 4.1: Reference tracking task for DeePC and MPC in the simulation of the linearized 2-link robotic arm.

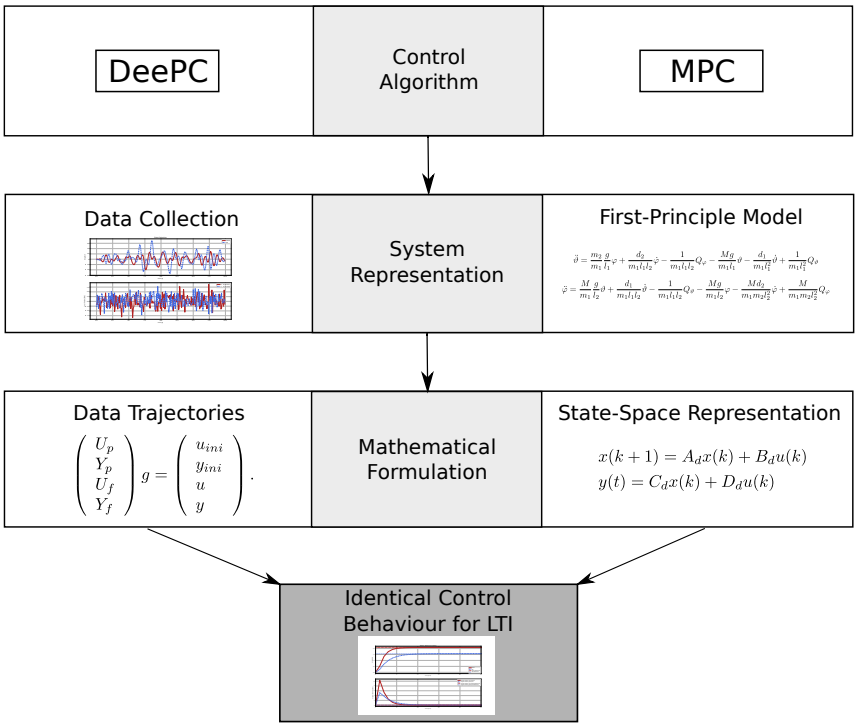


Figure 4.2: Schematic comparison of DeePC and MPC.

as well as two torques, which are stored in the input Hankel matrix. Based on the input and output Hankel matrix we can now form the data sets U_p, Y_p, U_f and Y_f for the formulation of the optimization problem. Solving this optimization problem in the context of Algorithm 2 provides the tracking performance illustrated in Figure 4.4. Furthermore, Figure 4.4 also provides the result for MPC. Having a closer look at the performance of the two algorithms, it becomes clear, that their control behaviour is identical. Hence, the application on the linearized 2-link confirms the theoretical guarantees provided by the Fundamental Lemma in Section 4.1. In the next chapter, we investigate the application of DeePC to nonlinear systems.

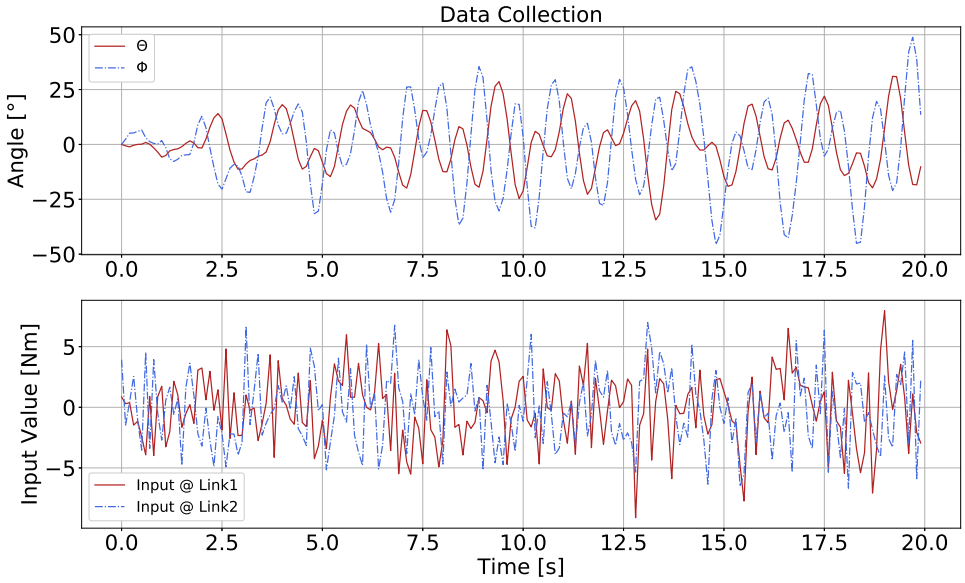


Figure 4.3: Illustration of data collection for DeePC applied to the linearized 2-link.

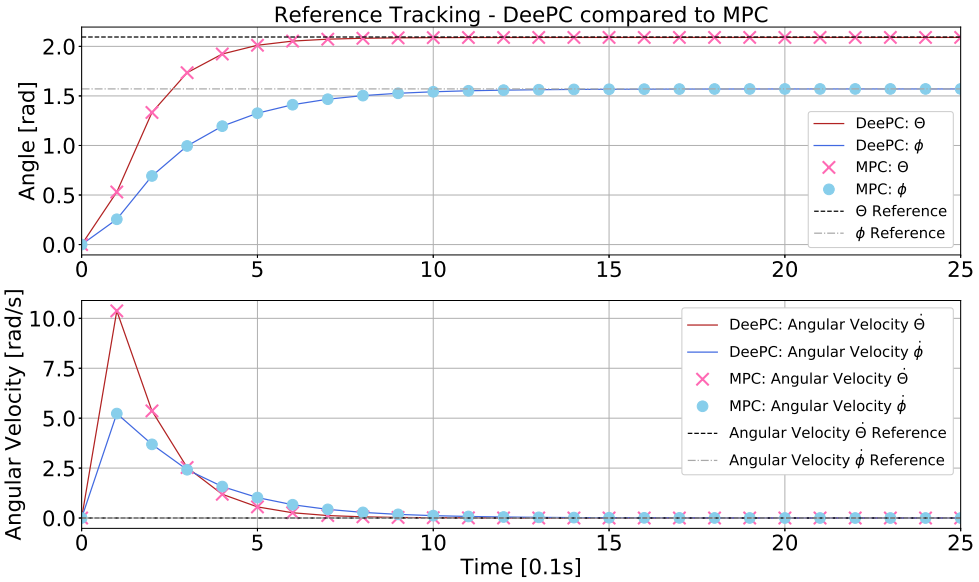


Figure 4.4: Performance of DeePC and MPC for the reference tracking task on the linearized 2-link. As guaranteed by the Fundamental Lemma, they provide identical results for the simulation of the linearized 2-link robotic arm.

Chapter 5

DeePC applied to nonlinear Systems

In this chapter we apply DeePC to nonlinear systems. We state DeePC-related challenges and their underlying reasons when operating in the nonlinear domain. We also investigate solutions to these challenges and use the simulations of a 2-link robotic arm and a M545 excavator for an analysis of the influence of data collection and hyperparameters.

5.1 DeePC-related nonlinear Challenges and their Solutions

When applying DeePC to nonlinear systems as opposed to LTI systems, we are facing emerging challenges. This is rooted in the fact that the previously introduced DeePC algorithm is based on linear systems theory. Recalling the mathematical preliminaries illustrated in Section 4.1, it becomes clear that all guarantees provided within that chapter are strongly tied to LTI systems. As a consequence, we need to answer the following two questions:

- How far can we transfer the linear concepts of DeePC to nonlinear systems?
- At which points do we need to add extensions or change the initial algorithm?

For answering these questions, we go through the structure of the DeePC algorithm and orient our thoughts on the topics of system representation by data, as well as extensions to the DeePC optimization problem.

System Representation by Data

In order to properly represent system behaviour we need an input signal that is sufficiently rich and long enough. When dealing with LTI systems we could refer to the signal property of persistence of excitation in order to check for such a sufficiently rich and long enough signal. By identifying the lag of the system, which is upper-bounded by $n(\mathcal{B})$ according to the Cayley-Hamilton theorem, we could formulate a Hankel matrix of full row rank of order $t+n(\mathcal{B})$. Hence, we were able to refer to the Fundamental Lemma and let our signal cover all the trajectories possible of the investigated system. When facing nonlinear systems, there are two conditions that change. First, we cannot refer to the lag of the system in order to determine the minimum value of T_{ini}. Here, we need to think about a new way of identifying the minimum length of our initial data trajectory. Second, even if we use our input signal to build a Hankel matrix of sufficient row rank, our nonlinear system is not globally controllable and therefore we can not rely on the guarantees provided by the Fundamental Lemma. Previously, when operating on linear systems, superposition principle allowed the decision vector g to linearly combine trajectories in order to describe the system behaviour in an ideal manner. For nonlinear systems, this does not work anymore, since linear combinations of trajectories not necessarily represent a trajectory of the nonlinear system, which results in a major challenge for DeePC. As a consequence, we are unable to ensure that our signal covers all system trajectories possible. On the contrary, we must assume that we do not sufficiently, in worst case even wrongly,

represent the system's behaviour due to nonlinear system's heavy diversions of system behaviour depending on the operating point.
In both cases, we suggest to counteract these challenges by means of more sophisticated data collection. Therefore, we investigate the impact of quantity and quality of data in Section 5.2.1. On the one hand, we study the influence of Hankel matrix formation and the related parameters T, T_{ini}, and N. On the other hand, we introduce guidelines for the qualitative assessment of data in the context of DeePC applied to nonlinear systems.

Adaptations of the Optimization Problem

The second part of the DeePC algorithm which we investigate is about the underlying optimization problem. We recall Equation (4.7) and note that we have a set of hard constraints as a part of the optimization problem, making it a constrained optimization problem. When applying DeePC to a nonlinear system, which is possibly corrupted by noise, these hard constraints would not be feasible at all times. Hence, following the suggestion in [7], we introduce a slack variable σ on the initial outputs y_{ini} (see (5.1)). Slack variables are a well established method for handling inequalities in linear programming as well as in nonlinear programming [6]. While it guarantees us feasibility of our constraints at all times, we also need to keep down the values of σ in order to achieve a precise as possible initial state estimation resulting from $Y_p g = y_{ini} + \sigma$. In order to do so, we introduce a penalty λ_y on the ℓ_1-norm of σ and consider it by including it to the cost function (see (5.1)).
A second addition to the initial DeePC algorithm suggested in [7], is a regularization of the decision vector g. For increasing the robustness of the DeePC algorithm against uncertainty, noise, and badly-fitted data, we introduce a penalty λ_g on the ℓ_1-norm of g and include it to the cost function. Like illustrated in [27], recent applications of ℓ_1-regularization in the field of, e.g., signal processing show a positive impact on denoising and signal recovery for incomplete measurements. Using the Wasserstein metric, [12] illustrates performance guarantees and tractable reformulations for data-driven distributionally robust optimization. This approach also motivates the work presented in [8], introducing a distributionally robust and chance-constrained version of the DeePC algorithm. Additionally, a theoretical motivation and interpretation of regularization in context of the DeePC algorithm is provided in [22].
Considering the penalty λ_y on the norm of the newly introduced slack variable σ, and the penalty λ_g on the norm of the decision vector g, we formulate the optimization problem J_{DeePC} of the so-called regularized DeePC algorithm [7] as

$$
\begin{aligned}
&\min_{g,u,y} \quad \sum_{k=0}^{N-1} \left(\|y_k - r_{t+k}\|_Q^2 + \|u_k\|_R^2\right) + \lambda_g \|g\|_1 + \lambda_y \|\sigma\|_1 \\
&\text{s.t.} \quad \begin{pmatrix} U_p \\ Y_p \\ U_f \\ Y_f \end{pmatrix} g = \begin{pmatrix} u_{ini} \\ y_{ini} \\ u \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ \sigma \\ 0 \\ 0 \end{pmatrix} \\
&\quad u_k \in \mathcal{U}, \forall k \in \{0, ..., N-1\}, \\
&\quad y_k \in \mathcal{Y}, \forall k \in \{0, ..., N-1\},
\end{aligned}
\tag{5.1}
$$

where $\sigma \in \mathbb{R}^{T_{ini}p}$ and λ_g, $\lambda_y \in \mathbb{R}_{\geq 0}$, while the other parameters, variables and sets are identical to (4.7). Since regularization takes up an essential role when applying DeePC to nonlinear systems, we dedicate Section 5.2.2 to the investigation of the influence of λ_g and λ_y.

5.2 Investigations of DeePC based on Simulations

As illustrated in Chapter 5.1, there are several hyperparameters as well as increased requirements on data collection when applying DeePC to nonlinear systems. Based on the python-embedded simulations of the 2-link robotic arm and the M545 excavator, this chapter provides heuristics for the influence of hyperparameters and proposes guidelines for the qualitative assessment of the

underlying data collection.
In order to assess the performance, we refer to three metrics:

- Objective value c_{obj}: The objective value is the summed result of the cost function in the minimization problem

$$c_{obj} = \sum_{i=1}^{n} J^*_{DeePC}(i) \tag{5.2}$$

over all simulation steps $i \in \{1, ..., n\}$, where n represents the total number of simulation steps per simulation.

- Precision value c_{prec}: The precision value is the sum over all simulation steps of the time-related cartesian distance between simulated excavator position $\hat{y}_k$ and reference position. The summation is normalized by the total number of simulation steps n, leading to

$$c_{prec} = \frac{\sum_{i=1}^{n} \|\hat{y}_i - r_i\|}{n}, \tag{5.3}$$

where $\hat{y}$ and r are row vectors consisting of the horizontal x-position and vertical z-position of the shovel contact and the horizontal x-reference and the vertical z-reference, respectively.

- Average solve time per time step $t_{solve,avg}$: The average solve time per time step is the sum of time needed for OSQP solving the Quadratic program $t_{solve,i}$ at each time step $i \in \{1, ..., n\}$ divided by the total number of simulation steps n of each simulation.

$$t_{solve,avg} = \frac{\sum_{i=1}^{n} t_{solve,i}}{n} \tag{5.4}$$

5.2.1 The Influence of Data Fitness

Recalling the long version of the abbreviation DeePC, which is Data-enabled Predictive Control, it is clear that data is essential for the algorithm. As previously illustrated, the minimization problem is not based on a model, but on data trajectories. Hence, it is essential for a successful application of the DeePC algorithm, to be aware of the significant impact of data collection. In order to describe the suitability of data sets for the purpose of reference tracking, we introduce the term of data fitness, which distinguishes between quantity and quality of the data. Using the idea of data fitness, we provide an orientation for the identification of well-fitted and badly-fitted data collections.

Data Quantity

The parameter T defines the quantity of data for the formulation of the DeePC algorithm. Together with the parameters T_{ini} and N, they determine the size and form of the Hankel matrix in which the relevant data is stored and arranged. A good indicator for the relation of T, T_{ini}, and N is the ratio of the length and the width of the Hankel matrix $\mathcal{H}$

$$hr(\mathcal{H}) = \frac{\#cols([U_p^T, Y_p^T, U_f^T, Y_f^T]^T)}{\#rows([U_p^T, Y_p^T, U_f^T, Y_f^T]^T)} \tag{5.5}$$

which we define as

$$hr_u(\mathcal{H}(u)) = \frac{T - (T_{ini} + N) + 1}{m(T_{ini} + N)} \tag{5.6}$$

for the inputs $u \in \mathbb{R}^m$, and

$$hr_y(\mathcal{H}(y)) = \frac{T - (T_{ini} + N) + 1}{p(T_{ini} + N)} \tag{5.7}$$

for the outputs $y \in \mathbb{R}^p$, respectively.
Based on empirical observations, we orient our investigations on hr_y, while hr_u could be subject to further research. Figure 5.1 illustrates the influence of hr_y. We observe a strong deterioration of performance for values which implicate a smaller width than length of the Hankel matrix ($hr \leq 1$) illustrated by the graph for c_{prec}. With an increasing hr performance imporves and reaches a sweat spot around $hr = 2.5$. In general, we can assume, that more data leads to more information about the system behaviour, and therefore results in a better performance. Nevertheless, the analysis of c_{prec} outlines that we should respect an appropriate balance of T and $T_{ini} + N$ as performance worsens for matrices with a width much higher than their length ($hr_y \approx 3.5$). Hence, we suggest $2 \leq hr_y \leq 2.5$ as an orientation when determining the dimensions of the Hankel matrix. Since an increasing amount of data means more mathematical operations of the algorithm, the limiting factor in context of data quantity is most often the solve time. Again, beyond choosing the right amount of data, it is important to choose an appropriate rate of Hankel matrix length and Hankel matrix width. Presenting $t_{solve,avg}$ over hr_y, the lower subplot of Figure 5.1 confirms $2 \leq hr_y \leq 2.5$ as a good starting point for forming the Hankel matrix also in terms of solve time.
By choosing T and following the recommendation for hr_y, a rough orientation for T_{ini} and N is provided. The detailed values for both parameters must be chosen for each system individually. Nevertheless, if available, it is a good help to consider system properties like damping and time delays between input and output. Recall the following: The higher the damping of a system, the more transient the influence of the input at time step k is on the system. Hence, it is good to have in mind, that a less-/undamped systems needs a comparatively high value, while more damped systems allow decreasing the size of T_{ini} and N if necessary. Even clearer is the case for time delays between inputs and outputs of a system like occurring in electric circuits or hydraulic applications. If such a time delay is known, we should define T_{ini} and N as least as long as the time delay.

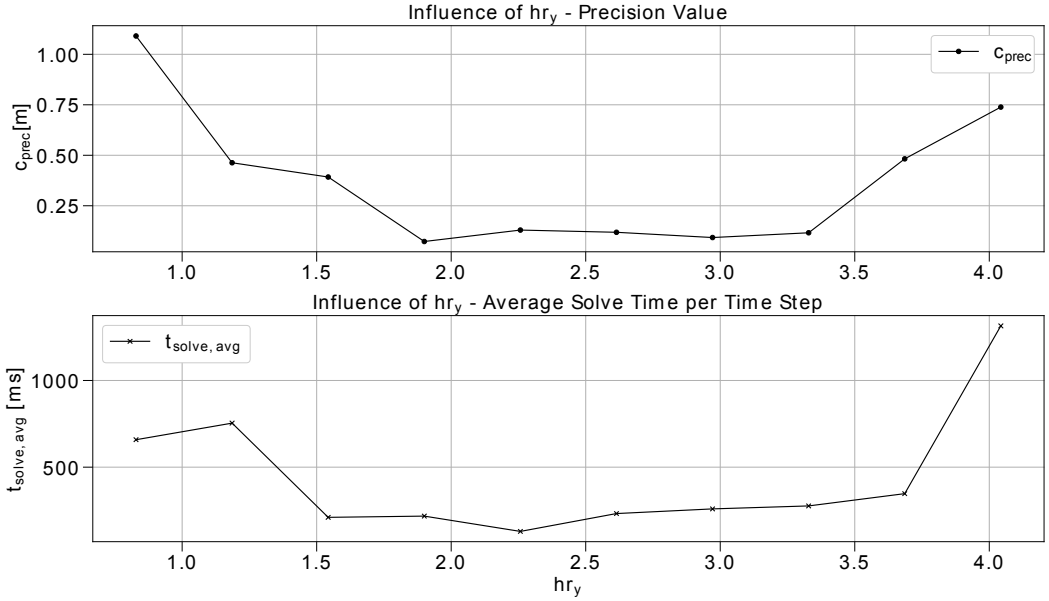


Figure 5.1: c_{prec} and $t_{solve,avg}$ over hr_y. The plot illustrates that one should aim for a width/length-ratio of the output Hankel matrix of at least 1.5. More precise, also in regards to solve time, choosing $2 \leq hr_y \leq 2.5$ is recommended.

Data Quality

Besides choosing the right quantity of data, it is also essential to use data with a high quality. In order to enable an assessment of data quality, we provide three empirically derived qualitative guidelines. For a better illustration, we serve a visual example for each guideline. The examples are based on the simulation of the M545 excavator with two flexible (boom, dipper) and two frozen joints (telescope, shovel). They use the cartesian shovel position for the task of reference tracking. The three guidelines for choosing well-fitted data sets for DeePC are:

- Operational Coverage: It can be observed that data sets, which describe the operational area of the tracked reference, outperform data sets, which do not cover the operational spectrum. This is intuitive as we know that the properties of nonlinear systems highly depend on the operating point. Furthermore, it is strongly related to the needed regularization for DeePC (investigated in Section 5.2.2). As it becomes clear at a later point, the better we cover the operating range, the less regularization we need. Figure 5.2 illustrates the control performance for two simulations based on almost identical data sets and equal hyperparameters. In example A, the reference is located in an area, which is well-covered by data collection. This leads to a superior control performance. In example B, we move the reference to an area, without data coverage. Note, that we refer to the simulation of the excavator and that the spacial difference of the two references already makes a difference for the related inputs due to the high masses and related moments of inertia of the excavator. Hence, in example B, as DeePC needs to rely on system data, which is not primarily describing the operating area, DeePC bases the control actions on insufficient data, which leads to reduced control performance.

- Input Smoothness: One of the most essential challenges for the DeePC algorithm is to identify the relation between inputs and outputs. It is obvious, that this challenge becomes easier, the less chaotic the input signal of the underlying data collection is. A chaotic signal tends to create its excitation by being very high-frequent, noisy and nervous (Figure 5.3, example B). In contrast, a more suitable signal creates its excitation by an appropriately varying frequency and magnitude as well as by combining different characteristic signals (Figure 5.3, example A, a changing offset overlapped by a sinus wave with varying frequency and marginal noise). Figure 5.3 compares two examples based on identical hyperparameters. Their tracking performance, in terms of precision, is comparable. Nevertheless, there is a significant difference in terms of acceleration, which can be traced back to the applied inputs. In Example A, there is a smooth input signal provided by data collection leading to very small accelerations when controlling the system. In Example B, there is a jerky input signal captured by data collection leading to drastic accelerations with much higher magnitude compared to Example A. The comparison outlines clearly that the characteristics of the input signal provided by data collection can be recognized in control behaviour. In favour of a smooth tracking performance, one should prefer a signal that guarantees persistency of excitation by methodically varying its course and characteristics instead of a signal that creates excitation by high noise-levels and extreme frequencies.

- Trajectory Recognition: Recall, that every column of the Hankel matrix is a snippet of the underlying data collection and that by using the Hankel matrix formation of the data, we emphasize the timely relation of the data points. For explaining the idea of trajectory recognition we refer to Figure 5.4. Please note that it is only a very simple schematic illustration for purposes of better understanding. For both examples we would like to follow the reference highlighted in green. Furthermore, both examples cover exactly the same points best-located for doing so (highlighted in red). In example A, there is only one trajectory, which includes all the relevant points and lets us directly recognise the reference trajectory. In example B, in order to cover the reference, we need to combine the points of three different trajectories, which have little in common with the reference trajectory. Intuitively, we would always choose the trajectory of example A, which is almost identical to the reference. But let's consider the two following theoretic aspects to confirm this intuition. First, we already

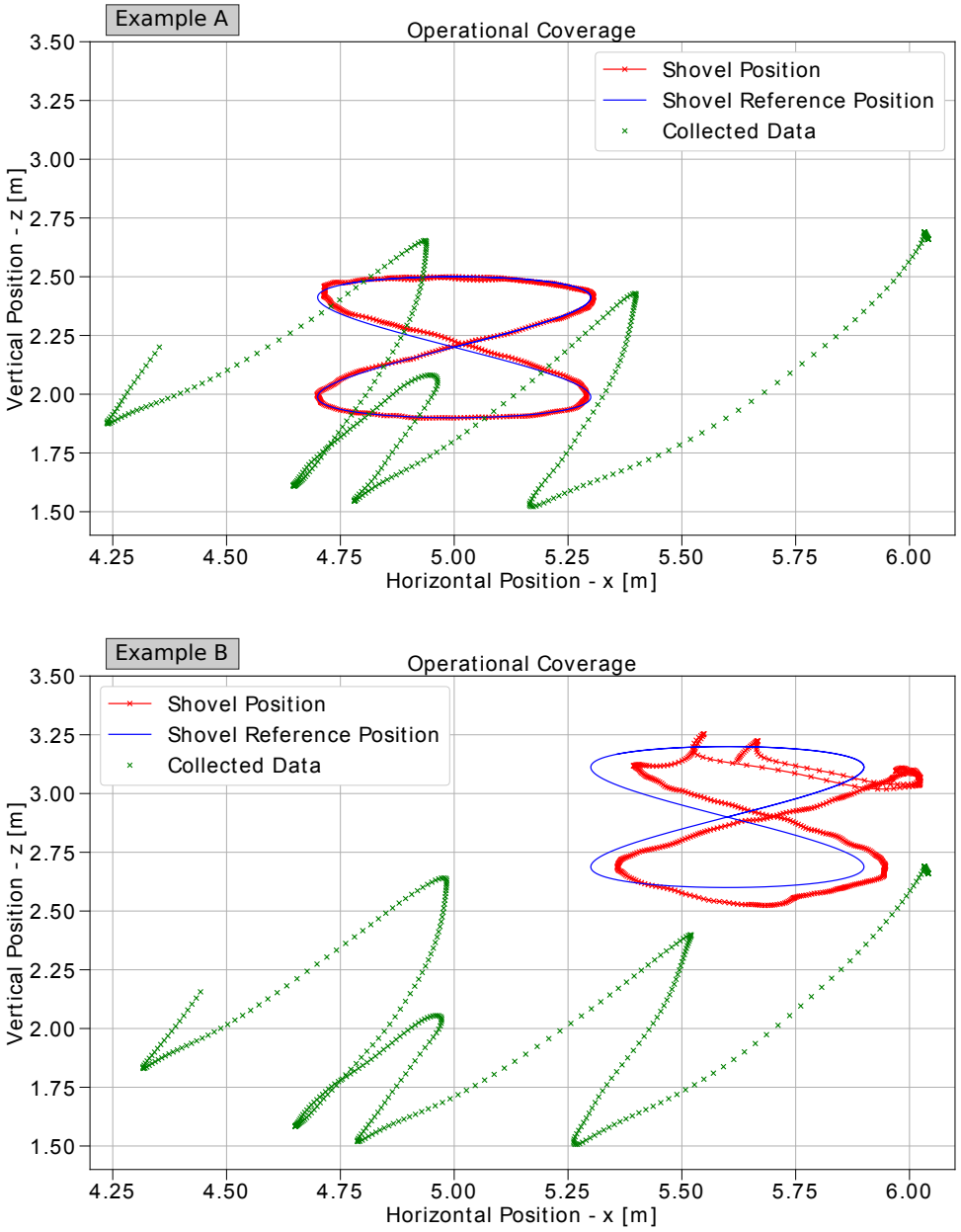


Figure 5.2: Comparison of the influence of operational coverage. In example A, the reference is located in an area, which is well-described by the underlying data. The related tracking performance is precise and smooth. In example B, the reference is located in an area without data coverage. The tracking performance clearly deteriorates, which is expressed by inconsistent and imprecise tracking behaviour. The hyperparameters are identical for both simulations.

mentioned that linearly combining trajectories ($\widehat{=}$ columns of the Hankel matrix) can cause trouble for nonlinear systems (this is strongly related to the role of λ_g, see Section 5.2.2). Hence, if possible, we prefer reducing the amount of linear combinations of trajectories. Second, by combining the trajectories we are able to cover the locations of the data points. But we also have to mind about the related inputs. In example B, there is a completely

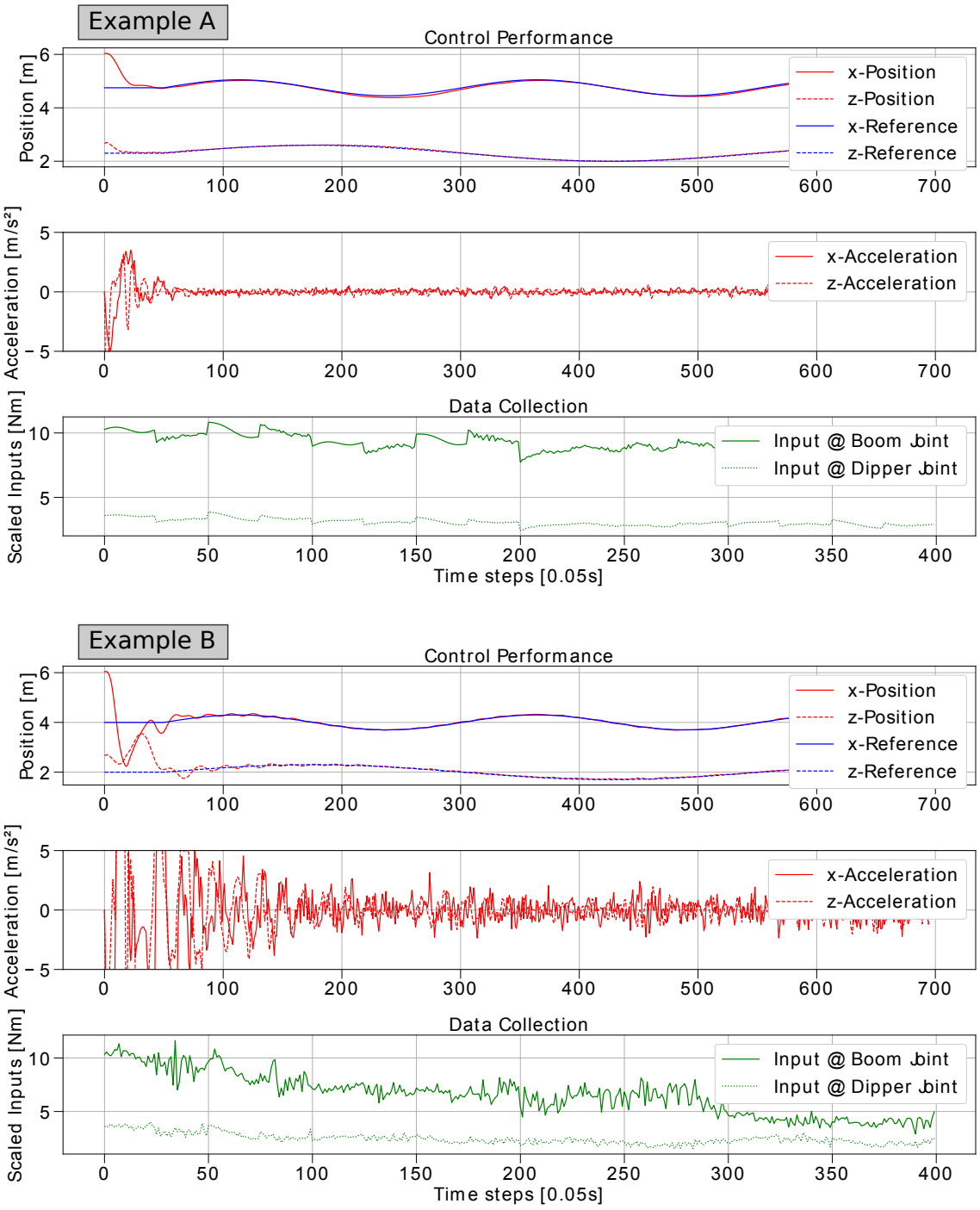


Figure 5.3: Influence of signal smoothness. In example A, a smooth input signal leads to smooth control performance. In example B, a jerky input signal leads to high accelerations of the system. The comparison outlines that the input signal characteristics, in terms of smoothness, are transferred to the control behaviour.

different input-output relation as it would need for following the reference, since the upper trajectories are rather horizontal than vertical and the lower trajectory goes vertical but has an opposing curvature. This makes it much harder for DeePC to follow the reference and

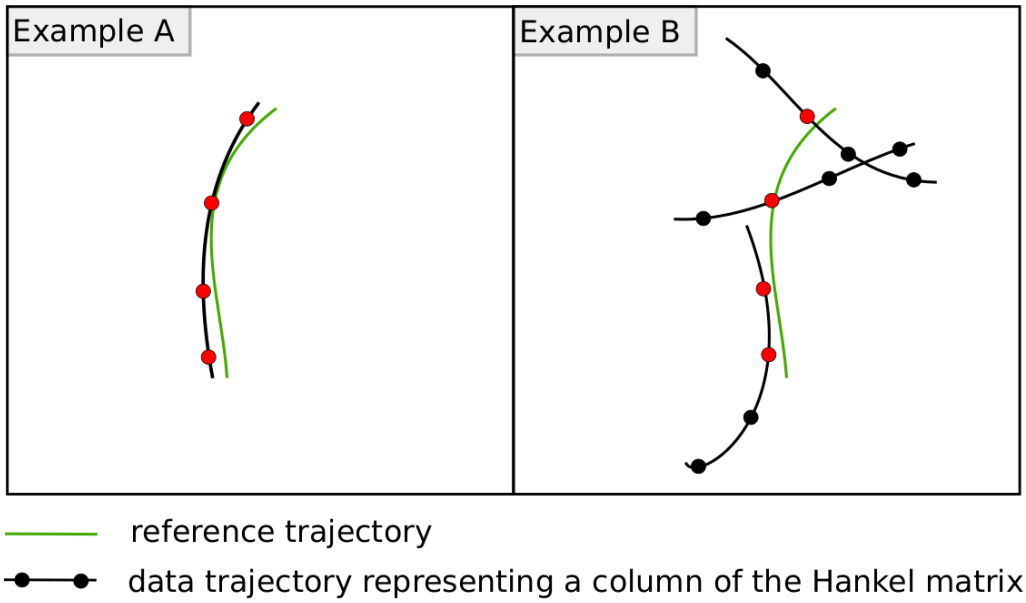


Figure 5.4: Schematic illustration of the advantage of trajectory recognition within data collection. In example A, there is a trajectory, which is a slight deviation of the targeted reference trajectory. Schematically, it is described by four data points highlighted in red. The identical four data points are also present in example B, represented by a combination of three trajectories, which have little in common with the reference trajectory. Example A is preferable due to two reasons. First, linearly combining the trajectories of the Hankel matrix in example B does not necessarily represent the nonlinear system. Second, despite covering the same local points, the input-output relation describing the system behaviour is completely different for the two examples. While targeting a vertical reference, the two upper trajectories in example B move almost horizontally, which leads to completely different inputs compared to the ones needed for following a vertical trajectory. By neglecting the dimension of time and only focusing on location, we can not represent the underlying physical correlation, which is particularly insufficient for the real world application on mechanical systems like a 12 t excavator.

the negative impact might become even worse, if we start thinking about systems with time-delays dependent on moving direction. With the help of this simple and schematic example, we are able to highlight how advantageous it can be if we recognize certain parts of the reference as trajectory snippets within the collected data.

5.2.2 The Influence of Hyperparameters

This sub-chapter provides an overview of the influence of the hyperparameters λ_g and λ_y.

λ_g - Robustness for DeePC

As announced in Section 5.1, we introduce the penalty λ_g on the norm of the decision vector g. By penalizing the norm of the decision vector we improve the robustness of the DeePC algorithm against nonlinearities, noise, and time delays. The higher λ_g is, the more selective is the algorithm in terms of how many columns and how strongly these are prioritized for the representation of

system behaviour mathematically modelled in

$$\begin{pmatrix} U_p \\ Y_p \\ U_f \\ Y_f \end{pmatrix} g = \begin{pmatrix} u_{ini} \\ y_{ini} \\ u \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ \sigma \\ 0 \\ 0 \end{pmatrix}. \tag{5.8}$$

Recall, that every entry of the vector g is directly related to a column of our Hankel matrix, and that every column of the Hankel matrix represents a data trajectory of length $T_{ini} + N$ out of the underlying data collection of length T. Choosing λ_g comparatively low means for the algorithm to consider more trajectories and balance their weighting for system representation. Choosing a comparatively high λ_g makes the algorithm more selective, and therefore only refer to a small number of columns in our Hankel matrix and prioritize them significantly. Figure 5.5 provides a comparison between two different values for λ_g. Every bar within the figure represents the value of an entry of the decision vector g. The values themselves are normalized by the maximum absolute value of each data set. The extract is based on the same underlying data set and both setups have exactly the same hyperparameters except λ_g. Furthermore, the reference is identical and the data represents g for exactly the same time step of each simulation. Thanks to the direct comparison, the previously explained influence of λ_g in terms of selectivity of the algorithm becomes clear. For $\lambda_g = 1$, the algorithm considers almost every entry of g and balances the values. Visually speaking, every peak is accompanied by an ascent and descent. In contrast, for $\lambda_g = 4000$, the algorithm selects much less entries and prioritizes those clearly. Visually, this leads to many much more isolated peaks in the lower subplot.

Intuitively, on the one hand, one would choose a rather low λ_g if there is a well-fitted data set, little nonlinearity of the system, and low uncertainty due to, e.g., noise or time delays. On the other hand, one would decide for a higher λ_g for badly-fitted data sets, strong nonlinearities, and a signal heavily influenced by noise and other factors leading to uncertainty. It also helps to rethink the difference between linear and nonlinear systems. As system behaviour of linear systems follows the same principles at every operating point of the system, it is not necessary to apply regularization, and choose for the best-fitting elements of g. On the contrary, making oneself clear, that the system behaviour of nonlinear systems heavily depends on the operating point and that the underlying data collection is most certainly not only covering that exact operating point, it becomes the logical consequence to apply regularization, and select and prioritize the best-fitting vector entries of g each representing a certain snippet of our data collection. Being aware of the principle of λ_g, one can get a qualitative orientation by recalling the consequences of regularization at the boundaries of a very low and a very high value for λ_g:

- If one chooses λ_g too low, then too much not relevant or wrong information about the system behaviour is considered by DeePC. As a consequence, the algorithm is unable to provide a precise representation of the system behaviour at the relevant operating point, which leads to a poor control performance.

- If one chooses λ_g too high, then we might encounter over-regularization. This means that the algorithm narrows down its understanding of system behaviour to a very limited number of entries of g, and therefore loses the ability to confirm its understanding by considering other trajectories within the Hankel matrix. For a good underlying data collection with an increased chance of finding an ideal data trajectory this is not a problem. For a bad data collection, this might have severe consequences, since the chosen data trajectories, although being the best-fitting parts of the Hankel matrix, still represent the system behaviour wrongly and mislead the DeePC algorithm.

Considering these two border cases, it becomes obvious that the trade-off when selecting λ_g is about prioritizing the right information while using as diverse information as possible. Hence, the ideal value of λ_g enables DeePC to find the most representative parts of our data collection in terms of system behaviour, while backing up its system understanding by using as many parts of the data collection as possible without considering misleading or wrong information.

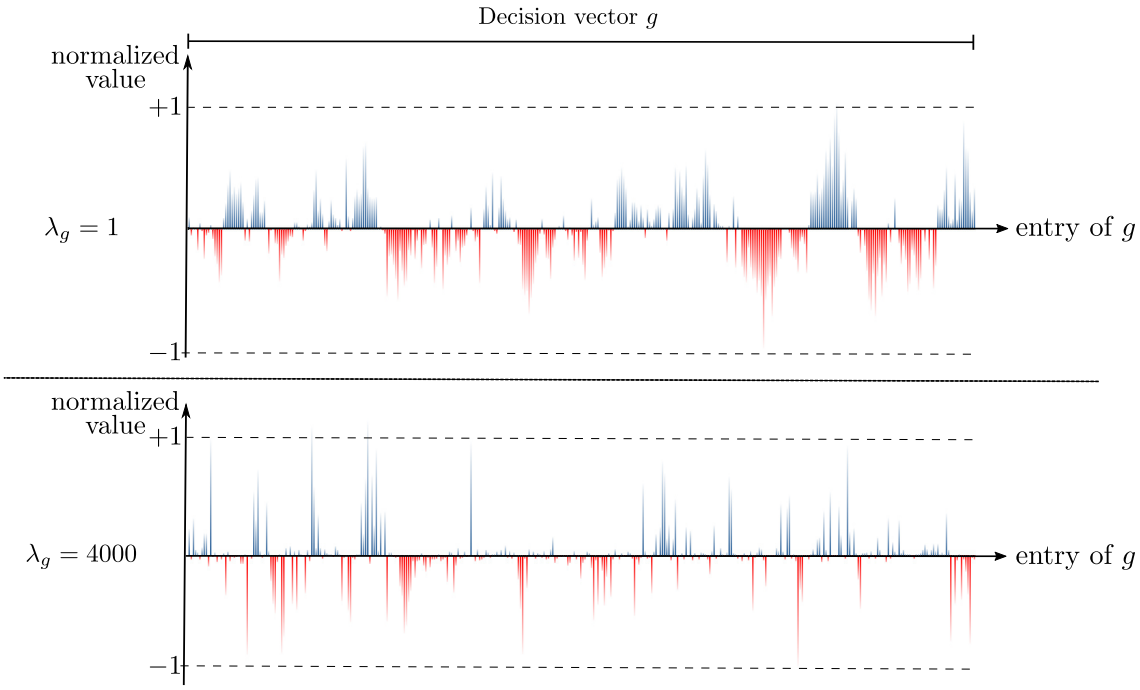


Figure 5.5: Comparison of the entries of the decision vector g depending on λ_g. Every bar represents the normalized value of a column entry. The values are normalized by the maximum entry of each data set ($\max(\lambda_g(1)) = 7.877; \max(\lambda_g(4000)) = 0.548$). For $\lambda_g = 1$, the algorithm takes into account almost every entry and balances their values. For $\lambda_g = 4000$, the algorithm considers significantly less entries and emphasises those, which leads to more isolated peaks in the view of the lower subplot.

In order to quantify this qualitative understanding of the influence of λ_g, we refer to the previously introduced precision value c_{prec}. Figure 5.6 illustrates the metrics for different values of λ_g based on a data set with moderate white measurement noise quantified by a signal-to-noise ratio $\eta = 0.02$. For simulation, we define the noise-corrupted output $\tilde{y}$ as

$$\tilde{y}(i) = \begin{pmatrix} \tilde{y}_1(i) \\ \vdots \\ \tilde{y}_p(i) \end{pmatrix} = \begin{pmatrix} y_1(i) + \eta \frac{w_1(i)}{|w_1(i)|} |y_1(i)| \\ \vdots \\ y_p(i) + \eta \frac{w_p(i)}{|w_p(i)|} |y_p(i)| \end{pmatrix} \qquad \forall i \in \{1, ..., n\}, \tag{5.9}$$

where $w(i) \sim \mathcal{N}(\vec{0}, I)$ is white measurement noise and n is the total number of simulation steps. In Figure 5.6 we notice a clear impact of the regularization, particularly in the beginning. While encountering under-regularization for $\lambda_g \leq 100$, we observe a zone of good control performance around a sweet spot of $\lambda_g \approx 1000$. By further increasing the value for λ_g, we approach the zone of over-regularization, which can be anticipated starting from $\lambda_g \approx 5000$. Using the insights of the plot, at the point of selecting a value for λ_g, it is necessary to choose a value that is high enough to achieve sufficient regularization, and a value that is low enough to avoid over-regularization. Hence, it is necessary to iteratively converge to the optimal value located between the two zones.

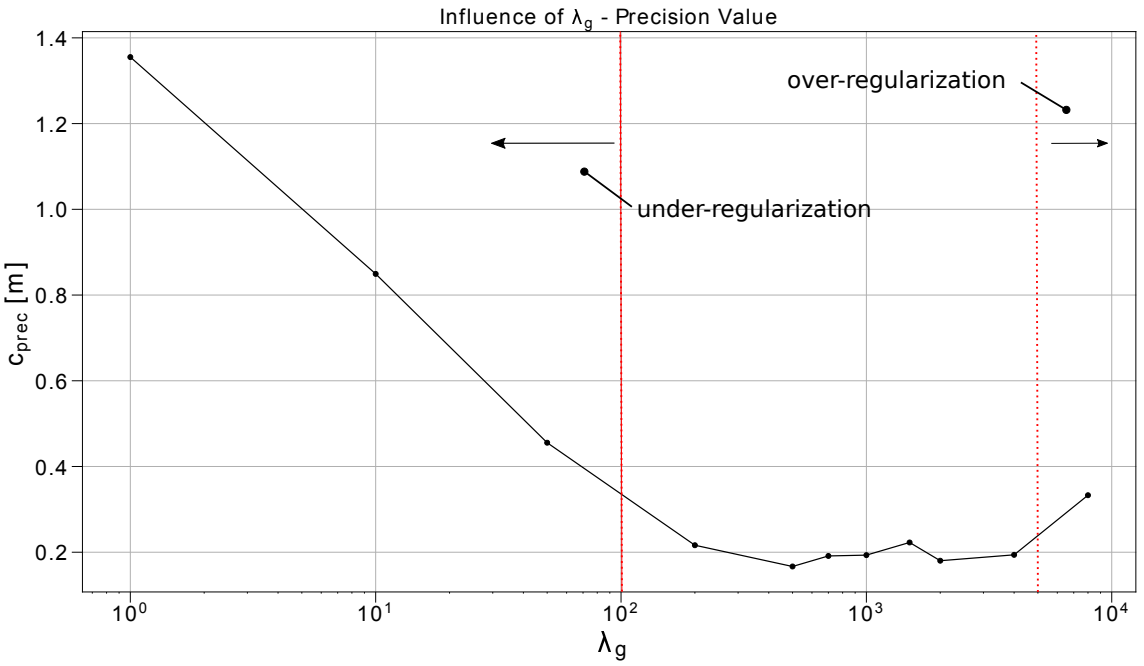


Figure 5.6: Precision value c_{prec} over λ_g. In the left part of the figure, we observe improving performance for increasing values of λ_g. Departing the zone of under-regularization at $\lambda_g \approx 100$, we note an area of good control behaviour around a sweet spot for $\lambda_g \approx 10^3$. By further increasing λ_g, we approach the zone of over-regularization, which we can anticipate starting at a value of $\lambda_g \approx 5000$.

λ_y - initial State Estimation

As a second hyperparameter we investigate the role of λ_y as penalty on the norm of the slack variable σ. On the one hand, the slack variable σ is introduced to guarantee feasibility of our constraints at all time. On the other hand, within the constraint formulation

$$Y_p g = y_{ini} + \sigma \tag{5.10}$$

it directly influences the initial state estimation of our system. Hence, while including the slack variable for feasibility, it is necessary to choose a suitable value for λ_y. This keeps the value of the slack variable to an appropriate dimension and therefore allows a precise initial state estimation. Figure 5.7 provides two strongly related values dependent on λ_y. The graph related to the left axis illustrates the course of an exemplary value of the ℓ_1-norm on the slack variable σ. For each simulation, representing a different value of λ_y, we choose the value at the time step $k = 75 \,\widehat{=}\, 3.75\,\text{s}$. Furthermore, all results are based on the identical data set and all hyperparameters except λ_y are the same. As expected, the value of the ℓ_1-norm of σ decreases significantly when increasing λ_y. Hence, for theoretical purposes, we can derive that a high λ_y allows a precise initial state estimation and should therefore be favoured when selecting λ_y. For practical reasons, the graph related to the right axis of the figure, presents the results for c_{prec}. Note, that these values are not based on exemplary time steps, but on a whole simulation with a total number of time steps $n = 350 \,\widehat{=}\, 17.5\,\text{s}$ each. In general, the graph aligns with the results provided for $||\sigma||_1$ by decreasing for an increasing λ_y. Nevertheless, for the practical application, there is an important difference to be noted. Focusing on the area of $\lambda_y \geq 10^7$, we note an increase of c_{prec}, which is equal to an unintended loss of tracking accuracy. The observed increase of c_{prec} is related to a negligence of the other hyperparameters relevant for DeePC as a consequence of choosing an extremely high value for λ_y. Due to the direct comparison of $||\sigma||_1$ and c_{prec} it becomes clear, that when tuning

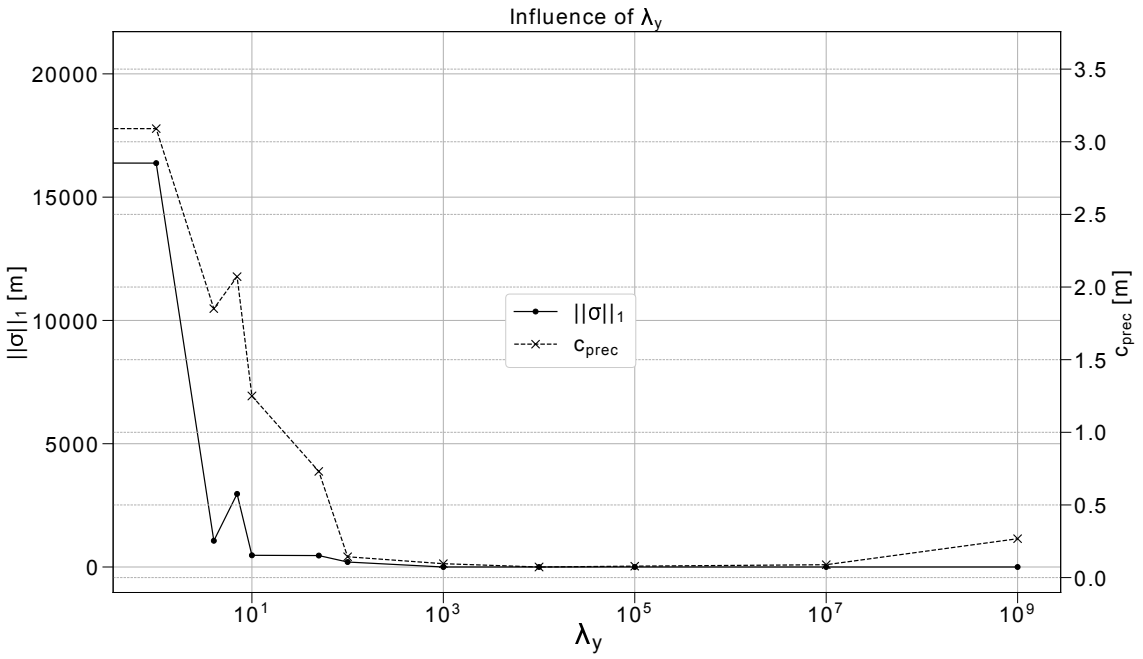


Figure 5.7: The value of the ℓ_1-norm on the left axis and precision value c_{prec} on the right axis. It is illustrated that an increasing value for λ_y reduces the magnitude of the slack variable and therefore allows a precise initial state estimation. Nevertheless, the comparison between $||\sigma||_1$ and c_{prec} for high values of λ_y highlights that λ_y should not be chosen endlessly high as this leads to a decrease of tracking performance due to deprioritization of other hyperparameters.

λ_y, it is necessary to find a balance between a sufficiently precise initial state estimation and a deprioritization of the other hyperparameters of the algorithm. For orientation when selecting a value for λ_y, we re-state the two border cases:

- If λ_y is too low, the slack variable increases and DeePC is not able to get a precise initial state estimation. As a consequence, reference tracking is poor.
- If λ_y is too high, we reduce the slack variable to a minimum, which is most certainly lower than needed for a sufficient initial state estimation. Furthermore, choosing λ_y too high, leads to a deprioritization of other hyperparameters. As a consequence, we pass the optimal value for λ_y and tracking performance decreases.

5.2.3 Summary of Influences

Since there is a vast amount of opportunities to influence the performance and behaviour of DeePC, Figure 5.8 summarizes the findings of the previous sub-chapters and highlights the most important influences on the three metrics robustness, tracking performance, and computational effort.
Section 5.2.1 outlines that data collection is the basis of DeePC. Since everything is built on the data collection, it strongly biases the control behaviour. Data fitness describes the suitability of a data set. It distinguishes between data quantity and data quality as influencing factors. In terms of quantity, it is possible to influence the outcome via the parameters T, T_{ini}, and N. In terms of quality, we observe that it is important to have a data set that describes the targeted operating area, which is based on smooth inputs while being persistently exciting, and ideally represents at least parts of the reference trajectory's characteristics. Overall, the data collection biases the performance of DeePC at an initial point and is highly related to success or failure of the DeePC algorithm. Focusing on the hyperparameters, Figure 5.8, illustrates the impact of λ_g on the

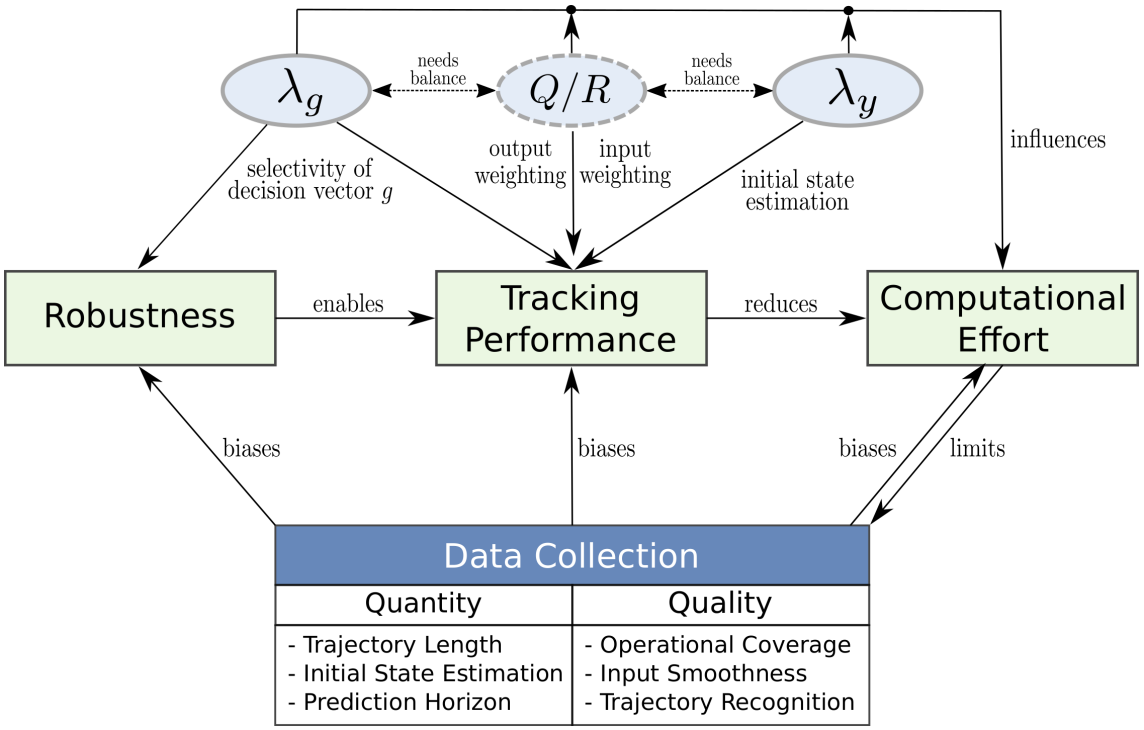


Figure 5.8: Overview of DeePC hyperparameters and their influence and relations. For achieving high robustness, precise tracking performance, and low computational effort, it is necessary to tune λ_g, λ_y, as well as Q and R. Furthermore, a well-fitted data collection in terms of quantity and quality is a key ingredient for the successful application of the DeePC algorithm.

selectivity of the decision vector g and the related influence on robustness and tracking performance. In general it states that an increased robustness of the algorithm allows more accurate tracking performance. The tracking performance is influenced by the weighting matrices Q and R, as well as λ_y. While λ_y influences the initial state estimation, Q and R can be used to balance precision and input cost. The computational effort depends on tracking performance, data collection and hyperparameter selection. Initially, when setting up DeePC, computational effort and the related solve time are often the limiting factors for the amount of data included.

5.2.4 Procedure for Hyperparameter Tuning

The previous chapters outline that the hyperparameter tuning is of significant importance for the success of DeePC. With every application to a new system and the related change of underlying data, it is necessary to re-tune the hyperparameters. Due to the high number of influencing factors and their relations, this can be a time-consuming task. In order to decrease the effort and the time needed for hyperparameter tuning, Figure 6.1 presents a procedure for finding the right setup.
In the beginning it is necessary to define the minimum tracking precision p_{tr}, the maximum solve time per time step $t_{solve,max}$, the maximum number of iterations n_{max}, as well as the minimum precision for the initial state estimation p_{ISE} and the allowed deviation η_{ISE} from that value. If there is knowledge about a time delay t_{delay} between inputs and outputs, T_{ini} and N are set to that value. If there is no information available, we recommend a starting value of 1 s. Depending on the choice for T_{ini} and N, T is determined via the output Hankel ratio $hr_y = 2.5$. The first hyperparameter to be set is λ_y. A good starting point is $\lambda_y = 10^5$. We use the value of $||\sigma||_1$ to investigate the impact of λ_y and to adjust its value. Once we find a suitable value for λ_y ($||\sigma||_1 \leq p_{ISE}$), the next hyperparameter is λ_g. Iteratively finding a value located between the

area of under- and over-regularization, we always take into account the impact of λ_g on λ_y. This is necessary to ensure the right balance and not to unintentionally sacrifice the initial state estimation for robustness. With λ_g and λ_y selected, we can now check for the solving time needed. If the solve time with the current setup is lower than 75% of the maximum allowed solve time, it is reasonable to increase the amount of data. The value of of 75% is based on experience but can be chosen differently depending on possible knowledge of the impact of the setup on solve time. If we reach the upper threshold of solve time, we can now start fine tuning the algorithm. This can be done via weighting matrices Q and R. If it is not possible to achieve the targeted precision by adjustment of the weighting matrices, and the hyperparameter values reach numerically difficult dimensions, one option is input and output scaling. By scaling the data, we free up new reserves for adjusting the values. A second option, which is not explicitly illustrated in Figure 6.1, is choosing a more aggressive value for λ_g. Recall the influence of λ_g illustrated in Section 5.2.2. If one can not reach the precision and solve time targets within the defined amount of maximum iterations n_{max}, a solution is to chose a different underlying data collection and, if possible, to soften the requirements.

1 define maximum solve time $t_{solve,max}$ and maximum iterations of tuning n_{max}
2 define precision p_{ISE} and acceptable error η_{ISE} for initial state estimation
3 define precision p_{tr} for reference tracking
4 set $\lambda_g = \lambda_y = 1$, set $Q = R = I$
5 if applicable, define $T_{ini,min} = N_{min} = t_{delay}$
6 set $T_{ini} = N = min\{T_{ini,min}; 1s\}$
7 set T s.t. $hr_y = 2.5$
8 set $\lambda_y = 10^5$
9 run DeePC
10 reduce λ_y
11 increase λ_y
12 scale inputs/outputs
13 set $\lambda_g = 500$
14 run DeePC
15 reduce λ_g
16 increase λ_g
17 increase T
18 adjust T_{ini}, N for $hr_y = 2.5$
19 reduce T
20 increase Q
21 collect new data and revise p_{tr}, $t_{solve,max}$ and n_{max}

A input-output time delay t_{delay}
B $\lambda_y = 1$
C $||\sigma||_1 \leq p_{ISE}$
D $||\sigma||_1 \cdot \eta_{ISE} \leq p_{ISE}$
E $\lambda_y \geq 10^9$
F $\lambda_g \neq 1$
G over-regularization
H under-regularization
I $t_{solve} \leq 0.75 \cdot t_{solve,max}$
J $t_{solve} \leq t_{solve,max}$
K tracking error $\leq p_{rt}$
L $n \geq n_{max}$
M $Q \geq 10^{10}$

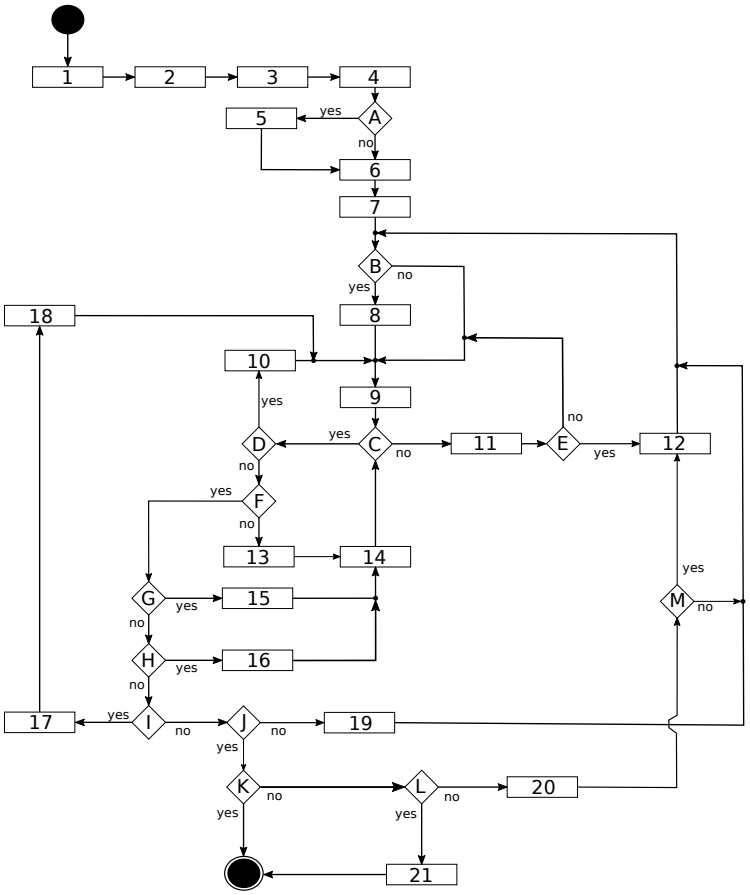


Figure 5.9: Illustration of the workflow for hyperparameter tuning. The boxes are actions and the diamonds are decisions.

Chapter 6

DeePC on a M545 Excavator

In this chapter we outline the implementation of DeePC on a real-world M545 excavator (presented in Section 2.2) and the results of an experimental study. Applying DeePC to the M545 is an ambitious and exciting task, since DeePC has never before been applied to a real world mechanical system of such dimensions. As a consequence, the work presented in this chapter serves as proof of concept for the feasibility of successful application of DeePC to strongly nonlinear mechanical systems. Furthermore, it allows an investigation of the following questions:

- How representative are the heuristics from simulation and how helpful are they when setting up DeePC on a real-world machine?
- How flexibly and intuitively can we adjust DeePC to system-specific requirements?
- How robust is DeePC against real machine noise and incomplete measurements?
- How well can the algorithm handle time delays between sensing and actuation, e.g., caused by hydraulics?

6.1 Implementation of DeePC

This section outlines the control architecture of the M545 and how DeePC is integrated into it. Furthermore, it introduces system-specific extensions of the DeePC algorithm.

Control Architecture

The provided M545 excavator is regularly used for experimental purposes and therefore offers an interface for the integration of the DeePC controller. The so-called high-level controller organizes all operations and connected controllers, and hence allows us to incorporate a C++-version of DeePC. The data sets for the DeePC controller are collected offline. To do so, we manually run the excavator and use a signal logger to store the data. In general, we operate the system based on a sampling rate of 20 Hz. Out of the overall data collection, we select the best-fitting extracts of length T following the guidelines presented in Section 5.2.1. Consecutively we arrange the data in the form of Hankel matrices and derive the matrices U_p, Y_p, U_f and Y_f, which concludes the offline procedure. Going online with the controller, DeePC starts tracking the reference r and therefore predicts the optimal inputs for the excavator arm. The inputs provided by DeePC are velocity commands for each joint. Hence, it is necessary to interpose the so-called low-level controller between DeePC controller and excavator arm. It maps the velocity commands to valve set points, which can then be applied to the hydraulic valves. The mapping of the low-level controller is empirically tuned. Referring to the valve set points, we can now control the excavator. The control loop is closed by the high-level controller, which updates and provides the necessary information about shovel position, joint angles, and reference to the DeePC controller at each time step.

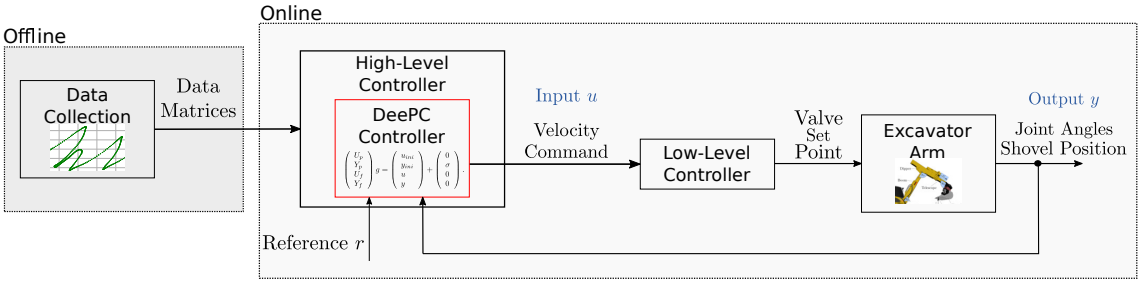


Figure 6.1: Integration of DeePC into the control architecture of the M545.

System-specific Extensions of DeePC

Due to the hydraulically actuated legs of the M545, the system can be interpreted as being mounted on four springs. As a consequence, the system is very vulnerable to oscillations. The main source for possible oscillations is the movement of the excavator arm. Hence, in order to avoid critical oscillations, it is necessary to apply very smooth control sequences. As previously outlined in Section 5.2.1, the control inputs can be biased by data collection. Nevertheless, in order to influence the control behaviour independent of data collection, we extend the cost function of the DeePC algorithm by two terms.
First, we add a penalty λ_u on the norm of the slew rate Δu_k of the inputs, which we define as

$$\begin{aligned} \Delta u_k &= u_k - u_{ini}(T_{ini}) \quad &&, \text{for } k = 0, \\ \Delta u_k &= u_k - u_{k-1} &&, \forall k \in \mathbb{Z}^+ \end{aligned} \tag{6.1}$$

with $u_k \in \mathbb{R}^m$ as the input at time step k. We choose $||\cdot||_\infty$ as norm type, in order to penalize the maximum element of the input slew rate vector.
Second, we add a penalty λ_a on the norm of the output accelerations a_k for directly suppressing high accelerations. We express a_k via finite difference scheme as the discrete derivative of second order

$$\begin{aligned} \Delta a_k &= \frac{y_{k+1} - 2y_k + y_{ini}(T_{ini})}{\Delta t^2} \quad &&, \text{for } k = 0, \\ \Delta a_k &= \frac{y_{k+1} - 2y_k + y_{k-1}}{\Delta t^2} &&, \forall k \in \mathbb{Z}^+, \end{aligned} \tag{6.2}$$

with $y_k \in \mathbb{R}^p$ as the output at time step k and Δt as the sampling rate. Similar to the slew rate, we choose $||\cdot||_\infty$ in order to penalize the maximum acceleration.

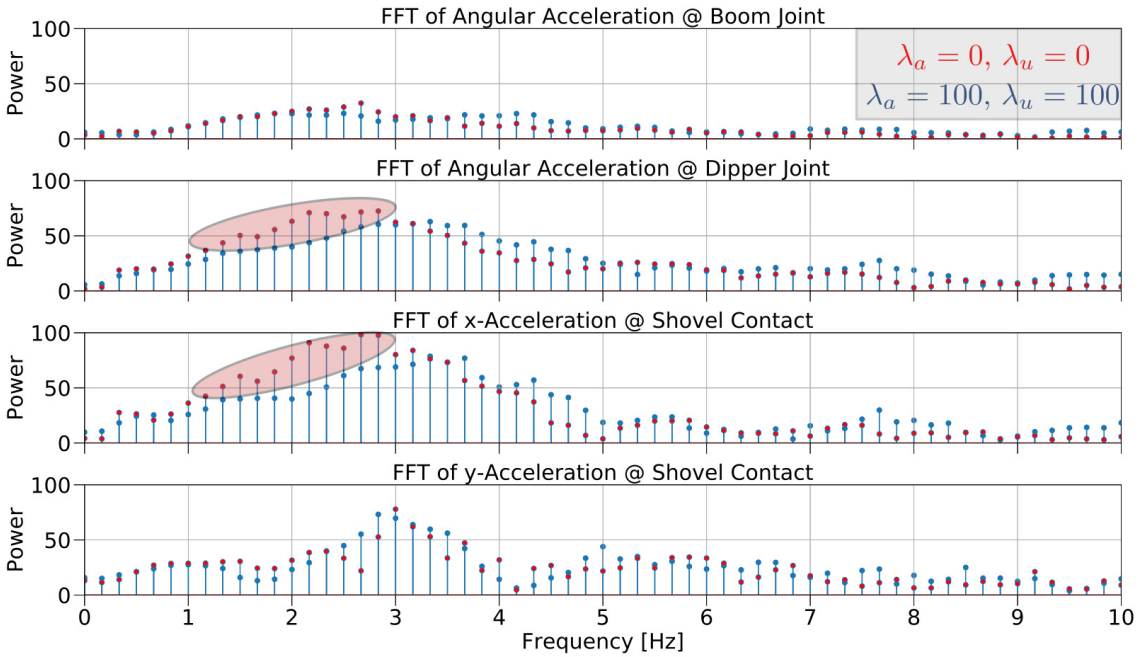


Figure 6.2: FFT for different setups of λ_u and λ_a based on simulation. The figure outlines that the excitation can be reduced in the critical area of 1 Hz - 2 Hz, which is highlighted in red.

We can now formulate the minimization problem used for controlling the M545 as follows

$$
\begin{aligned}
\min_{g,u,y} \quad & \sum_{k=1}^{N} \left(\|y_k - r_k\|_Q^2 + \|u_k\|_R^2 + \lambda_a \|a_k\|_\infty + \lambda_u \|\Delta u_k\|_\infty \right) + \lambda_g \|g\|_1 + \lambda_y \|\sigma\|_1 \\
\text{s.t.} \quad & \begin{pmatrix} U_p \\ Y_p \\ U_f \\ Y_f \end{pmatrix} g = \begin{pmatrix} u_{ini} \\ y_{ini} \\ u \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ \sigma \\ 0 \\ 0 \end{pmatrix} \\
& \Delta a_k = \frac{y_{k+1} - 2y_k + y_{ini}(T_{ini})}{\Delta t^2}, \text{ for } k = 0, \\
& \Delta a_k = \frac{y_{k+1} - 2y_k + y_{k-1})}{\Delta t^2}, \forall k \in \mathbb{Z}^+, \\
& \Delta u_k = u_k - u_{ini}(T_{ini}), \text{ for } k = 0, \\
& \Delta u_k = u_k - u_{k-1}, \forall k \in \mathbb{Z}^+, \\
& u_k \in \mathcal{U}, \forall k \in \{0, ..., N-1\}, \text{where} \mathcal{U} \subseteq \mathbb{R}^m, \\
& y_k \in \mathcal{Y}, \forall k \in \{0, ..., N-1\}, \text{where} \mathcal{Y} \subseteq \mathbb{R}^p.
\end{aligned}
\tag{6.3}
$$

The most critical frequencies for the excitation of the excavator are located between 1 Hz and 2 Hz. We use the newly introduced hyperparameters to reduce the occurrence of these frequencies. Based on the results of a simulation, Figure 6.2 illustrates the impact on the joint as well as the shovel accelerations by means of a Fast-Fourier-Transformation (FFT). It provides a direct comparison of a setup with $\lambda_u = \lambda_a = 0$ and a setup with $\lambda_u = \lambda_a = 100$. It is shown that λ_u and λ_a help to reduce the excitation by reducing the occurrence of particularly critical frequencies.

6.2 Proof of Concept and Experimental Study

After describing the implementation of the DeePC controller on the M545, we can now present the proof of concept. We provide results for a setup with two activated joints (boom, dipper) and four activated joints (boom, dipper, telescope, shovel). Beyond operating a system of 12 t and the related moments of inertia, this means forces in the dimension of 100 kN and a maximum reach of almost 10 m of the excavator arm.

For both setups of the excavator arm, we are able to provide stable control behaviour with DeePC.

There are several findings coming with this success, which we elaborate on the following pages. In particular, we investigate system representation and reveal the question about online adaptability, examine the role of T_{ini} for robustness, and confirm and specify the heuristics from simulation. Furthermore, we derive opportunities for improvement based on the observed strengths and weaknesses during the experiments.

In order to give a first impression of the system, we provide an example of data collection and control performance for a setup with four degrees of freedom (4DOF). Figure 6.3 illustrates the data collection. The upper subplot shows the operating range in cartesian coordinates referenced to the cabin of the excavator. In this example we illustrate a comparatively small extract with roughly a width of 1 m and a height of 1.5 m. We can identify characteristic trajectories of the individual joints as parts of the overall trajectory. Highlighted in red, A marks a movement of the telescope. The straight horizontal trajectory is connected to the input marked in the lowest subplot. Similar, B highlights a change of the shovel angle. While boom and dipper already allow a very flexible and smooth trajectory, the additional shovel joint enables particularly tight radii. In this example, and for most of our experiments, the data collection is of length $T = 450$, which corresponds to a time span of 22.5 s due to the underlying sampling rate of 20 Hz. Besides the advanced kinematics, a very important difference compared to the setup with two degrees of freedom (2DOF), is the fact, that there is the same amount of time steps while having twice the amount of joints and therefore inputs. This leads to less information exclusively referred back to the individual joint, which on the opposite leads to more information being the result of a superposition of movements of many joints. Intuitively, as we recall that all of DeePC's knowledge about the system behaviour is stored in the Hankel matrices, which are of same width for both setups, this simple relation highlights that the task for 4DOF is much more difficult than the task for 2DOF. A more quantitative illustration is related to the Hankel matrix ratio hr introduced in Section 5.2.1. For both setups, the Hankel matrices are of the same width, but for 4DOF of twice the height, leading to a reduced Hankel matrix ratio. Nevertheless, note, that this metric only takes into account the dimensional aspects and not the aspects related to the superposition of movements, which we assume as a major influence, too.

Figure 6.4 illustrates the control performance resulting of the data collection presented in Figure 6.3. The upper subplot presents a smooth trajectory of the excavator shovel based on 4DOF. Nevertheless, we must recognize that the tracking precision is not satisfactory. A very positive aspect can be gathered from the inputs. While they are oscillating in the beginning, they are dampened by DeePC, which leads to a very smooth trajectory. It is also notable that despite the large radius of 0.75 m, which leads to a reference that could be easily tracked by boom joint and dipper joint only, the tracking is based on all four joints including telescope and shovel.

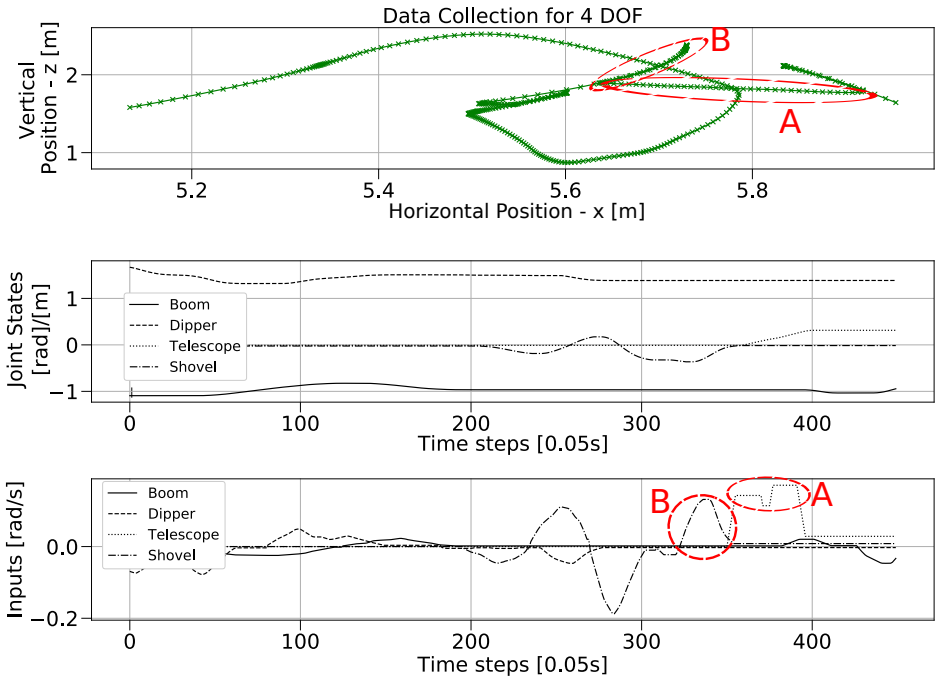


Figure 6.3: Data collection for four a arm setup with four degrees of freedom. While boom and dipper already cover a wide area, the shovel joint enables particularly tight radii and the telescope stroke allows an extended operating range.

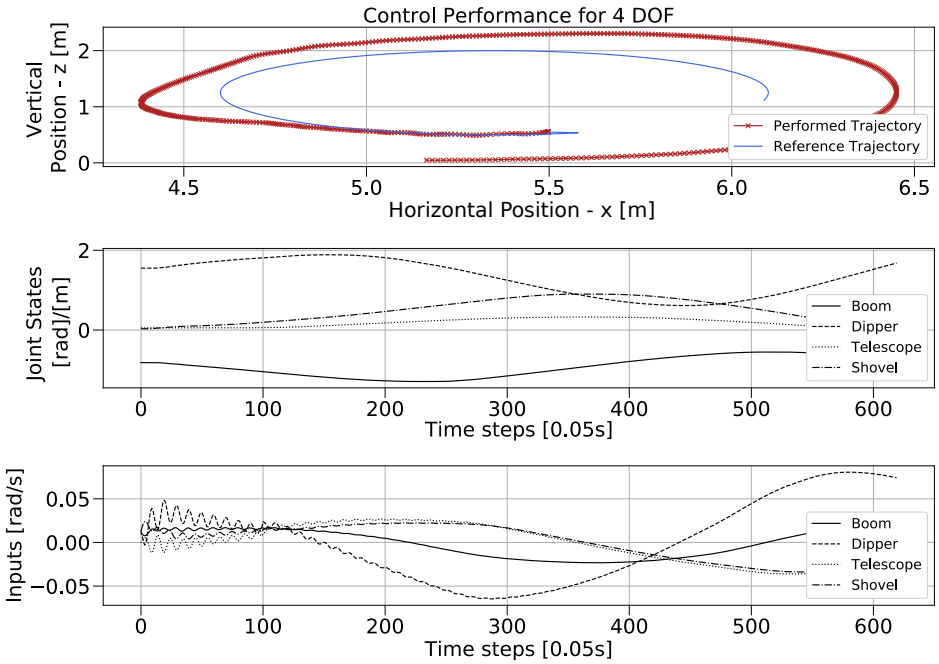


Figure 6.4: Tracking performance for a setup with 4 DOF. DeePC provides a very smooth trajectory, but lacks precision for this exemplary setup.

System Representation

The above described tracking performance is a result of system representation. The excavator setup with 2 DOF, like used in Figure 6.5, is an ideal environment to test the degree of reflection of underlying data. Using only boom and dipper joint for the task of reference tracking, we can fix the telescope and the shovel. Figure 6.5 illustrates a tracking performance which presents itself with a rotation of roughly 25° compared to the reference trajectory. The interesting point is that two different system setups were used for this plot. For data collection, the fixed shovel angle is ≈ 0°. For tracking, the fixed shovel angle is ≈ 45°. Hence we are controlling a system, which is different than the system the Hankel matrices are actually describing. The plot illustrates that DeePC re-models the system kinematics and highlights the importance of a correct system representation. Furthermore, it reveals the question about online adaptability of DeePC, which is a topic at a later point in this chapter.

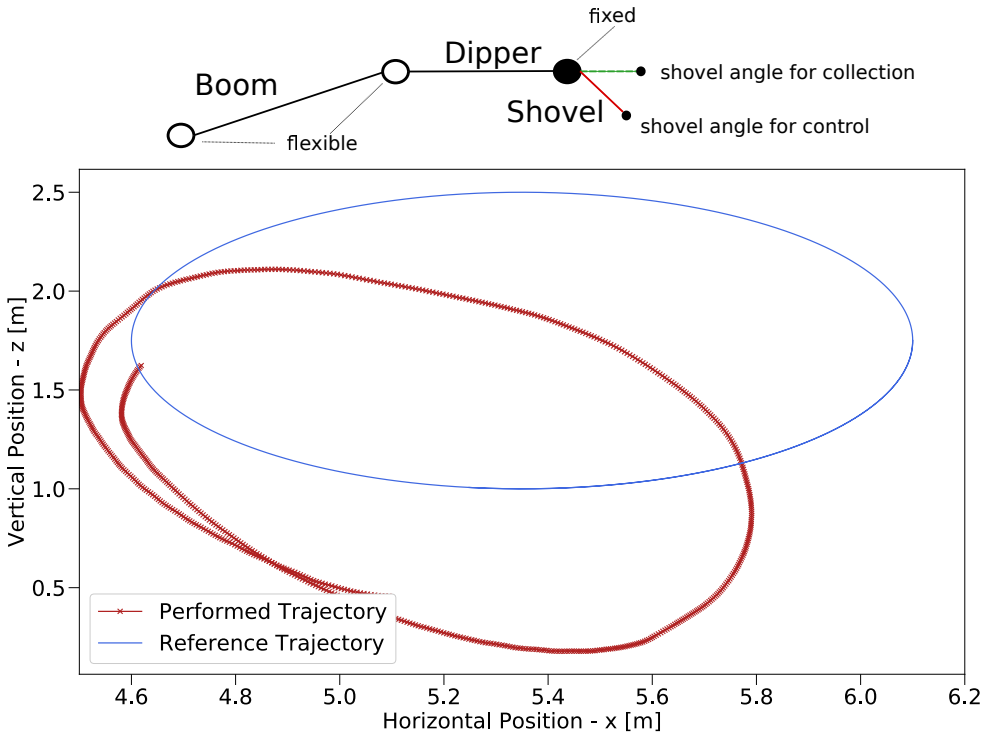


Figure 6.5: Trajectory plot for 2DOF and a schematic illustration of the system setup for data collection compared to the system setup when controlling the excavator. The data collection is based on a setup with a fixed shovel angle of 0°, while the tracking is based on a setup with a fixed shovel angle of 45°. The difference of setups is clearly recognizable in the tracking performance and reveals the question about online adaptability.

The Role of T_{ini} for Robustness

Section 5.2.1 and Section 5.2.2 use simulations to outline the importance of an accurate initial state estimation. Strongly related to the initial state estimation is the parameter T_{ini} since it defines the length of the past data sets U_p and Y_p. Figure 6.6 provides data points based on the excavator real world experiments. For the two data points with $T_{ini} = 10$ and $T_{ini} = 15$ we observe very poor control behaviour. Meanwhile, when increasing T_{ini}, starting from a value of $T_{ini} = 25$, we recognize stable and appealing performance. This is once more a confirmation for

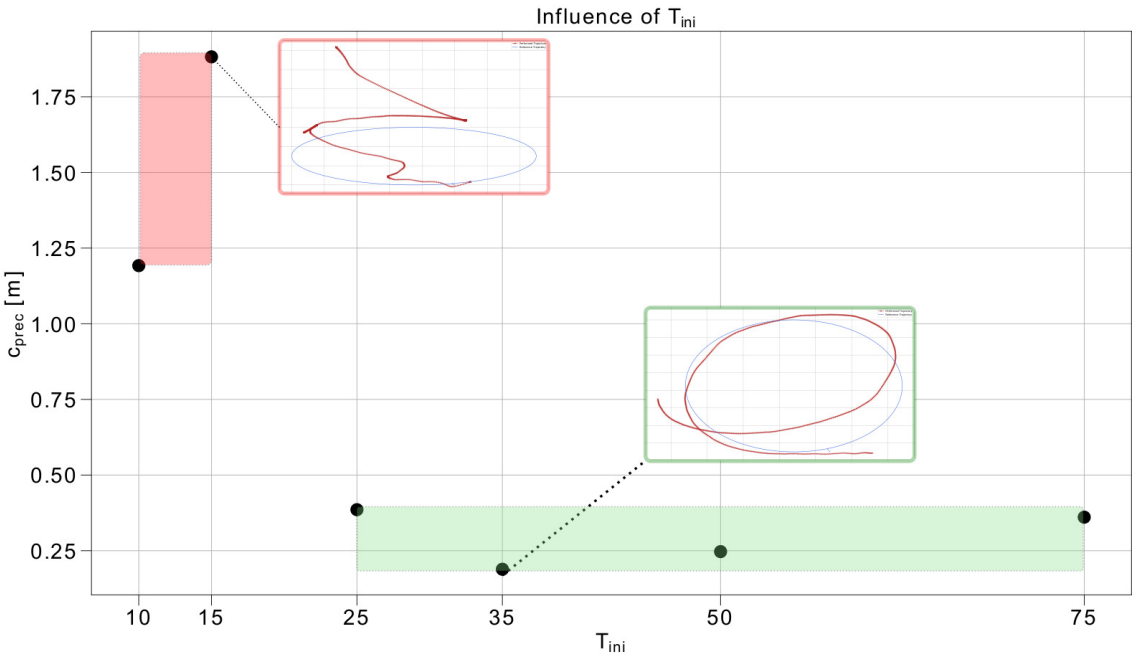


Figure 6.6: T_{ini} significantly influences control performance. Furthermore, when referring the results back to the simulation of the linearized and nonlinear systems, it underlines, that it is necessary to increase T_{ini} with increasing uncertainty of the system

the recommendation of choosing T_{ini} as large as possible when dealing with nonlinear systems. Inspecting the results for the real world experiment, it is also interesting to recall the context of uncertainty including nonlinearity, noise and time-delays. For the linearized system, we could define a lower threshold for the minimum size of T_{ini} referring to the lag of the system. Back then, the value of T_{ini} had a sufficient character. When changing to the nonlinear domain that character of T_{ini} changes from sufficient to necessary. Instead of setting up on a sufficiently large lower boundary, we now necessarily need to choose T_{ini} as large as possible in order to establish a chance for good control performance. Also consider the development when moving from simulation to the real machine. On the one hand, in simulation, setups with $T_{ini} = 10$ lead to promising results. On the other hand, for the real machine, we experience very poor performance for the same value. Recalling that the simulation does not consider the time-delays, and the real machine might be even more nonlinear at some points, this observation is nothing unexpected. Much more it confirms, that an increase of T_{ini} robustifies the DeePC algorithm and is necessary when approaching systems related to higher uncertainty.

Heuristics for Hyperparameters

In Section 5.2.2 we investigate the influence of λ_g and λ_y on the DeePC algorithm. Despite, representing a major guidance for simulation purposes, we could not know in advance, if they can be transferred to the real world application. We can clarify this point by the results presented in the two following graphs.
Figure 6.7 is the real-world-equivalent to Figure 5.6 based on simulation. Comparing the two plots, we can partly confirm the simulation-based characteristic course of the control performance dependent on λ_g. Nevertheless we need to point out three major differences:

- An improvement of control performance starts at a higher value of λ_g. When operating the real world machine, we need a more aggressive regularization. For too low values of λ_g, we observe no control behaviour leading to the excavator falling down to the ground.

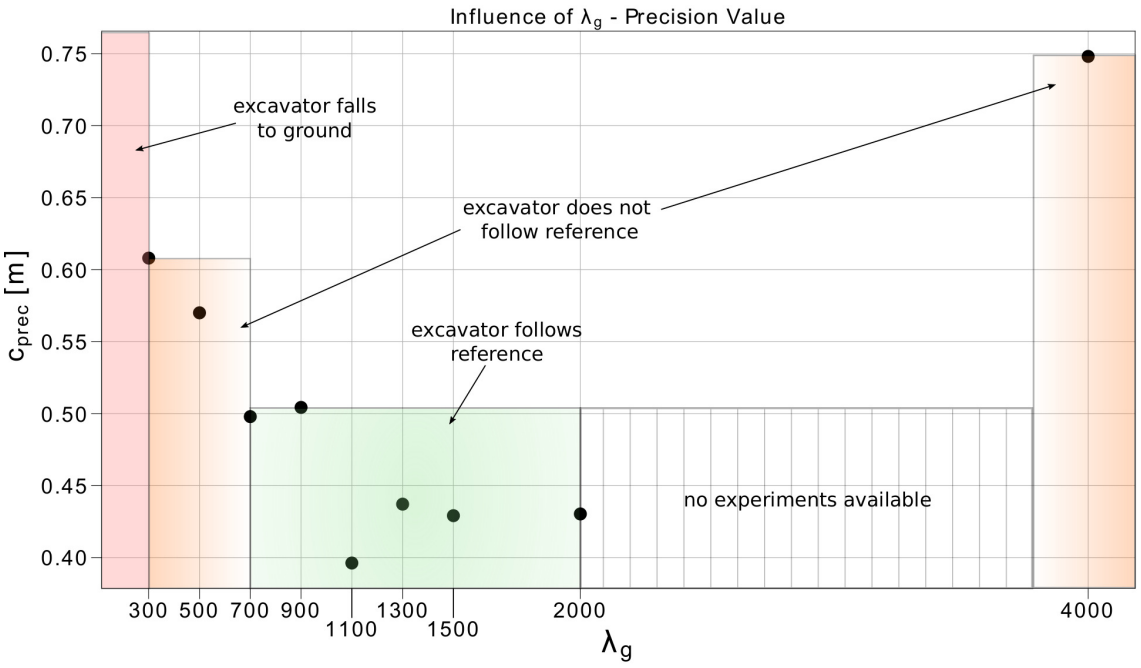


Figure 6.7: Influence of λ_g. With a regularization of $\lambda_g \leq 300$ we do not recognize any intention of controlling the system which leads to the excavator falling to the ground. With a stronger regularization expressed by a higher value for λ_g, we observe an imporving control performance with a sweet spot for $\lambda_g = 1100$. Further increasing the value to $\lambda_g = 4000$ leads to a deterioration of control performance due to over-regularization.

- The real world machine is more sensitive for over-regularization. Compared to the simulation, where we only mentioned, but did not encounter the problem of over-regularization, we can observe a deterioration of control performance for too high values.
- As a consequence of the more aggressive regularization needed and the earlier appearance of over-regularization, the sweet spot surrounding the optimal λ_g becomes tighter and therefore a more precise determination of λ_g is necessary.

The results for λ_y on the M545, presented in Figure 6.8, confirm the heuristic from simulation (see Figure 5.7). Nevertheless, a higher value is needed for achieving a positive impact, while considering a loss of performance for too high values. Hence, similar to λ_g, one must determine the value of λ_y more precise than in simulation.
Summarizing the insights on the hyperparameters, we can confirm the heuristics from simulation and refer to them as orientation for hyperparameter tuning. Nevertheless, we need to be aware, that finding the right hyperparameter setup becomes a more delicate balance when approaching real systems, which can be traced back to an increase of nonlinearity, noise, and time-delays between sensing and actuation.

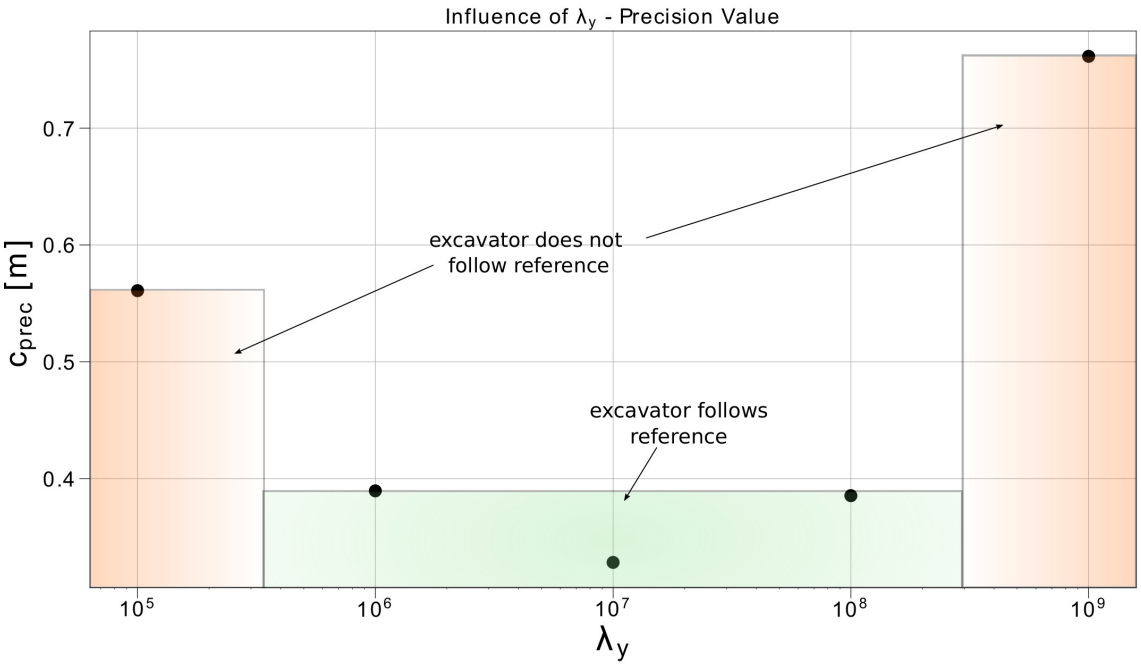


Figure 6.8: Influence of λ_y. When choosing $\lambda_y = 10^5$, we face an imprecise initial state estimation and therefore a poor control performance. With increasing λ_y we experience a better initial state estimation which leads to a better control performance. For $\lambda_y \geq 10^9$, control performance worsens due to deprioritization of the other hyperparameters.

Tracking Performance

For assessing the tracking performance, we refer to c_{prec}, which quantifies the average distance between shovel contact point and reference point per time step. Furthermore, we state the maximum error e_{max} between shovel contact point and reference point per experiment. For 2DOF we can present a setup achieving $c_{prec} = 0.1881\,\text{m}$ and a maximum error of $e_{max} = 0.3702\,\text{m}$. The corresponding tracking performance is illustrated in Figure 6.9. The best setup within the experimental series for 4DOF has an average distance per time step of $c_{prec} = 0.8404\,\text{m}$ and a maximal error of $e_{max} = 1.166\,\text{m}$. When facing these values, it is necessary to note that all experiments had the goal to highlight the relation of hyperparamters, and not to achieve an optimal tracking accuracy. Hence, there are several opportunities for improvement of precision with many of them related to the points suggested in the following paragraph.

Opportunities based on experimental Insights

Based on the insights from the first application of DeePC on a strongly nonlinear mechanical system we propose ideas and starting points for the further development of the algorithm.

- We outline at several points that the part of data collection is crucial. We present a guideline consisting of three aspects. For future applications it is crucial to quantify and automate the process of data scanning. This would be a huge benefit, particularly for the workflow in the context of a real world machine. A first starting point could be a restriction to joint-based applications.
- On the one hand, in Section 6.1 we show how to adapt the DeePC algorithm via the cost function to counteract oscillations. This extension helps us when bringing DeePC to the real machine, but must be done offline. On the other hand, we outline the characteristic control behaviour depending on the underlying data set. While showing robustness against system

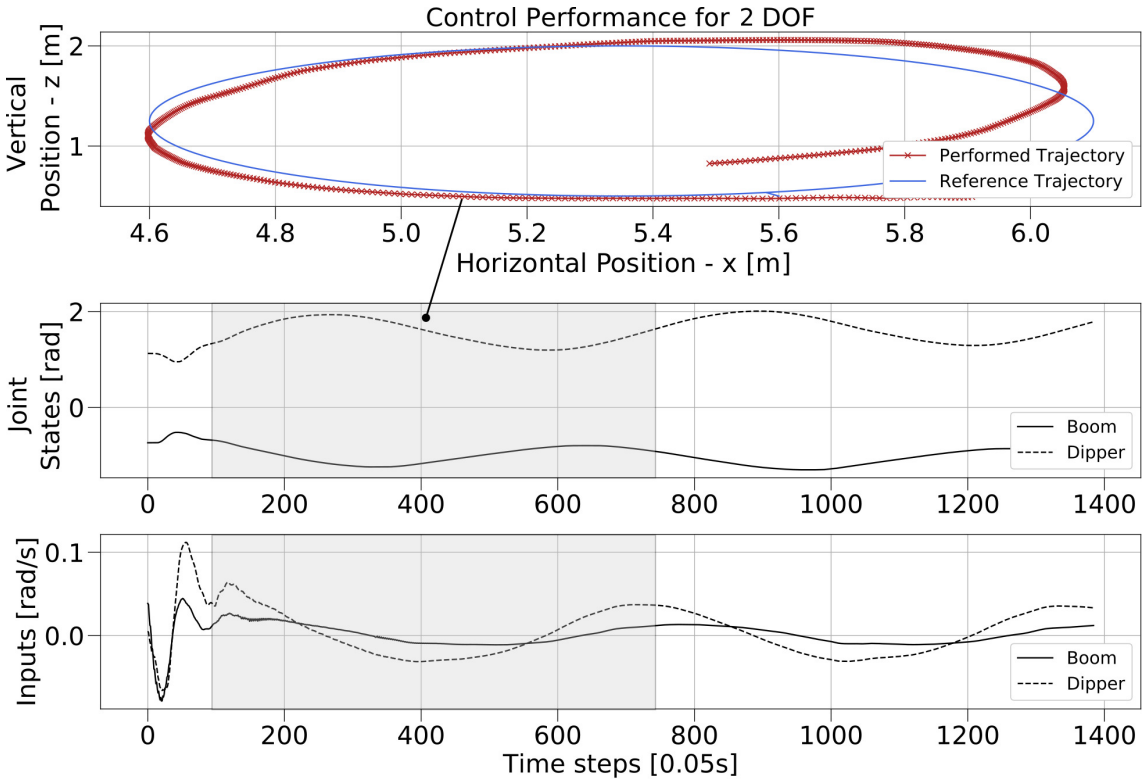


Figure 6.9: The most precise control performance for 2DOF within the experimental series. $c_{prec} = 0.1881\,\text{m}, e_{max} = 0.3702\,\text{m}$.

deviations, DeePC also shows that it cannot adjust to minor system deviations online. Hence, an approach allowing online updates of the Hankel matrices would be very promising for the feature of domain adaptability. Within the experimental series we tried to implement such an adaptive method, but were not successful due to not-achieving real-time capability.

- Reducing the solve time in general is a large potential for DeePC. Within this thesis we refer to basic methods for the mathematical operations. Applying some more sophisticated approaches could most certainly reduce the solve time and therefore generate a potential for including more data, which could be turned into an improvement of performance.

- We outline in the paragraphs above that the heuristics from simulation can be transferred to the real world application. This creates two opportunities. First, with the confirmed heuristics, it is possible to refer to the workflow for hyperparameter tuning presented in Section 5.2.4, and work into the direction of an automated hyperparameter tuning. Second, coupled to a method for the quantification of data fitness, one could use the heuristics for an adaptive hyperparameter scheme. This would allow to balance the hyperparameters depending on the situation and the available data and therefore contribute to the efficiency of DeePC.

Chapter 7

Conclusion

We start this thesis with an illustration of the experimental setup (Chapter 2) consisting of a simulated 2-link robotic arm and a Menzi Muck M545 12 t excavator. In the state of the art (Chapter 3), we investigate common control frameworks for the task of reference tracking. We distinguish between model-based and data-driven approaches and outline the advantages and drawbacks of one representative each. Consecutively, we use the insights from the literature review to bring DeePC in line with the current state of the art. Based on the mathematical preliminaries, we introduce DeePC and apply it to a linear system in Chapter 4. Entering the nonlinear domain, we refer to a simulation of a 2-link robotic arm and two high-fidelity simulations of the excavator, in order to investigate and illustrate the influence of hyperparameters, as well as the importance of data collection. Based on the simulations, we derive heuristics for the hyperparameters and introduce a routine for efficient tuning (Chapter 5). In Chapter 6, we then present the proof of concept for DeePC applied to a strongly nonlinear real world system. We confirm the hyperparameter heuristics from simulation and illustrate that we can use DeePC to control the M545 excavator with two and four degrees of freedom. Key findings resulting of this thesis are:

- It is possible to control strongly nonlinear mechanical real world systems with DeePC.
- Data collection is crucial, particularly for real world applications with increased noise levels and/or time-delays. An objective and quantifying method is definitely needed.
- The hyperparameter heuristics from simulation are confirmed by the real world experiments and can be used as an orientation. Nevertheless, we note that tuning the hyperparameters becomes a more delicate balance, which asks for a more precise determination of the individual optimal values.

As subjects for future research we recommend three topics. First, as already mentioned, it would be a major advantage to have a method that objectively quantifies data fitness. This would speed up the workflow, which is particularly important for real world applications, and eliminate the dependency on an individual's experience with data evaluation. Second, we suggest investigations in the direction of adaptive Hankel matrices. This would possibly give DeePC a feature similar to domain adaptability. Furthermore, in combination with a evaluation method for data fitness, one could adjust the Hankel matrices in order to provide an as accurate as possible system representation. Third, as the hyperparameter heuristics are confirmed on the real machine, it could be worth working into the direction of adaptive hyperparameters based on the heuristics. This could possibly increase the efficiency and performance of DeePC, while reducing the computational effort.

Appendix A

Technical Preliminaries

For our research in context of the DeePC algorithm, we rely on several technical resources including RaiSim, OSQP, and CVXPY.

RaiSim - Simulating physical Environments

The M545 simulation (introduced in Section 2.2) is built on RaiSim. RaiSim is a closed-source physics engine, allowing cross-platform projects in the domain of robotics and AI [24]. Within this thesis, we use the underlying C++-version of RaiSim, as well as the python-based version RaisimPY. Besides others, the advantages of RaiSim are speed [26], accuracy [23] [49], simplicity, as well as a minimum number of dependencies. In general, all resources of the M545 simulation are managed in a world environment. The RaiSim server then serializes the world environment and streams data to clients via tcp/ip. A visualization of the world environment is possible via RaiSimUnity. In simulation, the M545 is built as an articulated system, which allows access to several system characteristics like body type, as well as interactions with the environment like setting external forces.

OSQP - Solving Quadratic Programs

OSQP (Operator Splitting Quadratic Program) is a package that provides a numerical optimization method for solving convex quadratic programs. OSQP solves the QP of the form:

$$\begin{aligned} &\min_{x} \frac{1}{2} x^T P x + q^T x \\ &\text{s.t.} \quad l \leq Ax \geq u \end{aligned} \tag{A.1}$$

where $x \in \mathbb{R}^n$ is the optimization variable, while the objective function is defined by the positive semi-definite Matrix $P \in \mathbb{S}_+^n$ and vector $q \in \mathbb{R}^n$. The matrix $A \in \mathbb{R}^{mxn}$ and the vectors l and u define the linear constraints such that $l_i \in \mathbb{R} \cup \{-\inf\}$ and $u_i \in \mathbb{R} \cup \{+\inf\}$ for all $i \in \{i, ..., m\}$ [51].
OSQP is based on the ADMM algorithm, which allows fast computation as well as infeasibility detection [1]. Furthermore, additional features like polishing for high-precision solutions, adaptive ρ-step-size for fast convergence, as well as warm-start for increased efficiency are available. An illustration of code generation is provided in [2].

CVXPY - Formulating Minimization Problems in Python

As parts of the system simulations used within this thesis are based on Python, we use CVXPY to re-formulate the quadratic program at the core of the DeePC algorithm. CVXPY is a python-embedded modeling language which allows the expression of minimization problems in a mathematically natural way [9]. Perfectly matching our purposes, CVXPY also provides an interface to OSQP and supports all the previously introduced OSQP-related features [52].

Appendix B

Access to Video Material

We provide two videos of DeePC applied to the M545. The first video shows the M545 with 2 degrees of freedom tracking a circular reference. It can be accessed via:

- `https://polybox.ethz.ch/index.php/s/KMDSrVXq18gEVYV`

The second video shows the M545 using all 4 degrees of freedom to the track a circular reference. It can be accessed via:

- `https://polybox.ethz.ch/index.php/s/T3s23fWfbvPDXac`

In the case of any technical issues, please contact the author via e-mail:
`felix.wegner@rwth-aachen.de`

Bibliography

[1] G. Banjac, P. Goulart, B. Stellato, and S. Boyd. Infeasibility detection in the alternating direction method of multipliers for convex optimization. *Journal of Optimization Theory and Applications*, 183(2):490–519, 2019.

[2] G. Banjac, B. Stellato, N. Moehle, P. Goulart, A. Bemporad, and S. Boyd. Embedded code generation using the OSQP solver. In *IEEE Conference on Decision and Control (CDC)*, 2017.

[3] Andrew Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems: Theory and Applications*, 13, 12 2002.

[4] Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition, 1957.

[5] Lucian Busoniu, Robert Babuska, Bart De Schutter, and Damien Ernst. *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC Press, Inc., USA, 1st edition, 2010.

[6] Andrew Conn, Nicholas Gould, and Philippe Toint. A note on exploiting structure when using slack variables. *Math. Program.*, 67:89–97, 10 1994.

[7] J. Coulson, J. Lygeros, and F. Dörfler. Data-enabled predictive control: In the shallows of the DeePC. In *European Control Conference*, pages 307–312, 2019.

[8] Jeremy Coulson, John Lygeros, and Florian Dörfler. Distributionally robust chance constrained data-enabled predictive control, 2020.

[9] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[10] A. D'Souza, Sethu Vijayakumar, and Stefan Schaal. Learning inverse kinematics. volume 1, pages 298 – 303 vol.1, 02 2001.

[11] Pascal Egli. Towards rl-based hydraulic excavator automation. 2020-07-06.

[12] Peyman Esfahani and Daniel Kuhn. Data-driven distributionally robust optimization using the wasserstein metric: Performance guarantees and tractable reformulations. *Mathematical Programming*, 171, 05 2015.

[13] W. Favoreel, B. De Moor, P. Van Overschee, and M. Gevers. Model-free subspace-based lqg-design. In *Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251)*, volume 5, pages 3372–3376 vol.5, 1999.

[14] Maryam Fazel, Ting Pong, Defeng Sun, and Paul Tseng. Hankel matrix rank minimization with applications to system identification and realization. *SIAM Journal on Matrix Analysis and Applications*, 34, 07 2013.

[15] Rolf Findeisen and Frank Allgöwer. An introduction to nonlinear model predictive control. 01 2002.

[16] Michael Forbes, Rohit Patwardhan, Hamza Hamadah, and Bhushan Gopaluni. Model predictive control in industry: Challenges and opportunities. *IFAC-PapersOnLine*, 48:531–538, 12 2015.

[17] Carlos E. GarcÃa, David M. Prett, and Manfred Morari. Model predictive control: Theory and practiceâa survey. *Automatica*, 25(3):335–348, 1989.

[18] Huadong Guo, Lizhe Wang, Fang Chen, and Dong Liang. Scientific big data and digital earth. *Chinese Science Bulletin (Chinese Version)*, 59:1047, 12 2014.

[19] Sebastian Hörl, Francesco Ciari, and Kay Axhausen. Recent perspectives on the impact of autonomous vehicles, 09 2016.

[20] Zhong-Sheng Hou and Zhuo Wang. From model-based control to data-driven control: Survey, classification and perspective. *Information Sciences*, 235:3–35, 2013. Data-based Control, Decision, Scheduling and Fault Diagnostics.

[21] Linbin Huang, Jeremy Coulson, John Lygeros, and Florian Dörfler. Data-enabled predictive control for grid-connected power converters. pages 8130–8135, 12 2019.

[22] Linbin Huang, Jianzhe Zhen, John Lygeros, and Florian Dörfler. Quadratic regularization of data-enabled predictive control: Theory and application to power converter experiments, 12 2020.

[23] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26), 2019.

[24] Jemin Hwangbo, Joonho Lee, and Marco Hutter. Per-contact iteration method for solving contact dynamics. *IEEE Robotics and Automation Letters*, 3(2):895–902, 2018.

[25] Ryan Luke Johns, Martin Wermelinger, Ruben Mascaro, Dominic Jud, Fabio Gramazio, Matthias Kohler, Margarita Chli, and Marco Hutter. Autonomous dry stone: On-site planning and assembly of stone walls with a robotic excavator. *Construction Robotics*, 4, 12 2020.

[26] Dongho Kang and Jemin Hwangho. Simbenchmark.

[27] Seung-Jean Kim, K. Koh, M. Lustig, Stephen Boyd, and Dimitry Gorinevsky. An interior-point method for large-scale l1-regularized least squares. *Selected Topics in Signal Processing, IEEE Journal of*, 1:606 – 617, 01 2008.

[28] Jens Kober, J. Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32:1238–1274, 09 2013.

[29] K. Kristinsson and G. A. Dumont. System identification and control using genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(5):1033–1046, 1992.

[30] J. G. Kuschewski, S. Hui, and S. H. Zak. Application of feedforward neural networks to dynamical system identification and control. *IEEE Transactions on Control Systems Technology*, 1(1):37–49, 1993.

[31] W. Kwon and Soohee Han. Receding horizon control: Model predictive control for state models. 2005.

[32] Francoise Lamnabhi-Lagarrigue, Anuradha Annaswamy, Sebastian Engell, Alf Isaksson, Pramod Khargonekar, Richard M. Murray, Henk Nijmeijer, Tariq Samad, Dawn Tilbury, and Paul Van den Hof. Systems control for the future of humanity, research agenda: Current and future roles, impact and grand challenges. *Annual Reviews in Control*, 43:1–64, 2017.

[33] Adam Daniel Laud. *Theory and Application of Reward Shaping in Reinforcement Learning.* PhD thesis, USA, 2004. AAI3130966.

[34] I. Markovsky and P. Rapisarda. On the linear quadratic data-driven control. In *2007 European Control Conference (ECC)*, pages 5313–5318, 2007.

[35] Ivan Markovsky and Paolo Rapisarda. Data-driven simulation and control. *International Journal of Control*, 81(12), 12 2008.

[36] Ivan Markovsky, J.C. Willems, Sabine Huffel, and Bart De Moor. Exact and approximate modeling of linear systems: A behavioral approach. 03 2006.

[37] Nathan Melenbrink, Justin Werfel, and Achim Menges. On-site autonomous construction robots: Towards unsupervised building. *Automation in Construction*, 119:103312, 2020.

[38] Teodor Mihai Moldovan and Pieter Abbeel. Safe exploration in markov decision processes, 2012.

[39] K. S. Narendra and P. Kudva. Stable adaptive schemes for system identification and control-part i. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-4(6):542–551, 1974.

[40] Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, page 278â287, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[41] Robotic Systems Lab of ETH Zurich.

[42] Claudio De Persis and Pietro Tesi. Formulas for data-driven control: Stabilization, optimality and robustness, 2019.

[43] G.V. Raffo, Manuel Ortega, and Francisco Rubio. Path tracking of a uav via an underactuated control strategy. *European Journal of Control*, 17:194â213, 12 2011.

[44] Carl Edward Rasmussen. *Gaussian Processes in Machine Learning*, pages 63–71. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[45] Anja Richert, Sarah Müller-Abdelrazeq, Stefan Schröder, and Sabina Jeschke. Anthropomorphism in social robotics: empirical results on human-robot interaction in hybrid production workplaces. *AI SOCIETY*, 33, 08 2018.

[46] Stuart Russell. Learning agents for uncertain environments (extended abstract). In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, page 101â103, New York, NY, USA, 1998. Association for Computing Machinery.

[47] Allison Sander and Wolfgang Mel. The rise of robotics. *bcg.perspectives*, 2014.

[48] Berthold Schuppar. *Elementare numerische Mathematik.* Vieweg+Teubner Verlag, 1999.

[49] Fan Shi, Timon Homberger, Joonho Lee, Takahiro Miki, Moju Zhao, Farbod Farshidian, Kei Okada, Masayuki Inaba, and Marco Hutter. Circus anymal: A quadruped learning dexterous manipulation with its limbs, 2020.

[50] A. Simpkins. System identification: Theory for the user, 2nd edition (ljung, l.; 1999) [on the shelf]. *IEEE Robotics Automation Magazine*, 19(2):95–96, 2012.

[51] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672, 2020.

[52] Bartolomeo Stellato and Goran Banjac.

[53] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018.

[54] Jonathan Tilley. Automation, robotics, and the factory of the future, 2017.

[55] Matteo Turchetta, Andreas Krause, and Sebastian Trimpe. Robust model-free reinforcement learning with multi-objective bayesian optimization. pages 10702–10708, 05 2020.

[56] C. J. Turner, J. Oyekan, L. Stergioulas, and D. Griffin. Utilizing industry 4.0 on the construction site: Challenges and opportunities. *IEEE Transactions on Industrial Informatics*, 17(2):746–756, 2021.

[57] Jan C. Willems. From time series to linear system: Part i. finite dimensional linear time invariant systems. *Automatica*, 22(5):561–580, 1986.

[58] J.C. Willems, Ivan Markovsky, Paolo Rapisarda, and Bart De Moor. A note on persistency of excitation. volume 3, pages 2630– 2631 Vol.3, 01 2005.

[59] J. Yamaguchi and A. Takanishi. Development of a biped walking robot having antagonistic driven joints using nonlinear spring mechanism. In *Proceedings of International Conference on Robotics and Automation*, volume 1, pages 185–192 vol.1, 1997.

[60] Bin Yao and Li Xu. Adaptive robust motion control of linear motors for precision manufacturing. *Mechatronics*, 12(4):595–616, 2002.

[61] M. Zucker and J. A. Bagnell. Reinforcement planning: Rl for optimal planners. In *2012 IEEE International Conference on Robotics and Automation*, pages 1850–1855, 2012.