Master's thesis

Nonlinear Model Predictive Control for Concentrating Solar Power Receiver based on a Transformer Neural Network

Juan Ignacio Escorza Chavez
Matr. 230377

First examiner:	Prof. Dr.-Ing. Sergio Lucia
Second examiner:	M. Sc. Niklas Kemerling
Advisor:	M. Sc. Kevin Iding
Year:	2024

Technische Universität Dortmund
Faculty of Biochemical and Chemical Engineering
Laboratory for Process Automation Systems

Deutsches Zentrum für Luft- und Raumfahrt e. V. (DLR)
Institute of Solar Research
Department of Concentrating Solar Technologies

technische universität dortmund

process automation systems

DLR Institute of Solar Research

Eidesstattliche Versicherung

(Affidavit)

ESCORZA CHAVEZ, Juan Ignacio

Name, Vorname
(surname, first name)

230377

Matrikelnummer
(student ID number)

☐ Bachelorarbeit
(Bachelor's thesis)

■ Masterarbeit
(Master's thesis)

Titel
(Title)

Nonlinear Model Predictive Control for Concentrating Solar Power Receiver based on a Transformer Neural Network

Ich versichere hiermit an Eides statt, dass ich die vorliegende Abschlussarbeit mit dem oben genannten Titel selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

I declare in lieu of oath that I have completed the present thesis with the above-mentioned title independently and without any unauthorized assistance. I have not used any other sources or aids than the ones listed and have documented quotations and paraphrases as such. The thesis in its current or similar version has not been submitted to an auditing institution before.

Dortmund, 01.Jan.2024

Ort, Datum
(place, date)

Unterschrift
(signature)

Belehrung:
Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50.000,00 € geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler/die Kanzlerin der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden. (§ 63 Abs. 5 Hochschulgesetz - HG -).

Die Abgabe einer falschen Versicherung an Eides statt wird mit Freiheitsstrafe bis zu 3 Jahren oder mit Geldstrafe bestraft.

Die Technische Universität Dortmund wird ggf. elektronische Vergleichswerkzeuge (wie z.B. die Software „turnitin") zur Überprüfung von Ordnungswidrigkeiten in Prüfungsverfahren nutzen.

Die oben stehende Belehrung habe ich zur Kenntnis genommen:

Official notification:
Any person who intentionally breaches any regulation of university examination regulations relating to deception in examination performance is acting improperly. This offense can be punished with a fine of up to EUR 50,000.00. The competent administrative authority for the pursuit and prosecution of offenses of this type is the Chancellor of TU Dortmund University. In the case of multiple or other serious attempts at deception, the examinee can also be unenrolled, Section 63 (5) North Rhine-Westphalia Higher Education Act (*Hochschulgesetz, HG*).

The submission of a false affidavit will be punished with a prison sentence of up to three years or a fine.

As may be necessary, TU Dortmund University will make use of electronic plagiarism-prevention tools (e.g. the "turnitin" service) in order to monitor violations during the examination procedures.

I have taken note of the above official notification:*

Dortmund, 01.Jan.2024

Ort, Datum
(place, date)

Unterschrift
(signature)

***Please be aware that solely the German version of the affidavit ("Eidesstattliche Versicherung") for the Bachelor's/ Master's thesis is the official and legally binding version.**

Acknowledgements

Ich möchte meinen größten Dank an M.S. und die Familie S. aussprechen. Ohne ihre unablässige Unterstützung und bedingungslosem Vertrauen wäre dies alles nicht möglich gewesen. Ich habe Glück, dass ich euch alle habe.

Also, I am grateful to my supervisor, M.Sc. Iding, for giving me the chance to be part of this project, the experience at the Deutsches Zentrum für Luft- und Raumfahrt e. V. (German Aerospace Center, DLR) and his invaluable guidance and support throughout the research process.

To Prof. Lucia and M.Sc. Kemmerling, thank you for the opportunity and honor to write my thesis with you at the chair.

Finally, to my new *Heimat* and the education it provided; the TU Dortmund University, family and friends, as well as HiWi and Internship supervisors and colleagues,

THANK YOU

"[...]'Every time you victimized someone,' I said, 'you were victimizing yourself.
Every act of kindness you've done, you've done to yourself.
Every happy and sad moment ever experienced by any human was,
or will be, experienced by you.'[...]"
The Egg, by Andy Weir

"It is not the flag moving, nor the wind blowing,
but rather the movement of your minds."
Mumonkan koan 29, by Huineng (638-713), in TED-Ed Zen koans.

Abstract

The purpose of this thesis is to explore the use of Transformer Neural Network (T-NN) to model complex system dynamics, to evaluate their potential for use in modern control techniques, and to investigate their applicability in a Model Predictive Control (MPC) controller for the Solar Tower Power Plant Jülich (STJ) at the Deutsches Zentrum für Luft- und Raumfahrt e. V. (German Aerospace Center, DLR) in Jülich.

The multi-step ahead prediction capability of a Neural Network (NN) with transformer architecture is used to model the dynamics of the nonlinear multivariable system of the receiver based on real data. By incorporating information about future disturbances or setpoint changes, the predictive behavior of the model facilitates counteracting expected external influences. This property is then applied in a MPC through an Optimal Control Problem (OCP) formulation.

To solve the OCP, the PyTorch and SciPy libraries are tested with different optimizers. The first, while being an unconstrained optimizer, is used to solve the constrained problem by means of proposed barrier functions to explore its potential.

It is also observed that despite the state of the art NN, its accuracy depends on the training data and that interpolation outside the training region leads to inaccurate or unexpected predicted dynamics.

In this work, the PyTorch approach provides similar solutions to constrained optimizers while having faster computational times and better performance based on simulation results.

Within the distribution of data on which the NN was trained, it is shown that the proposed Transformer Neural Network (T-NN) enabled MPC controller is capable of tracking the reference while satisfying the constraints and rejecting disturbances or setpoint changes in the proposed test scenarios and in the presence of measurement noise. Moreover, prediction errors are fitted in a Gaussian Process Regressor (GPR) to obtain Uncertainty Quantification (UQ) information insights to be displayed to the operator of the system.

For our application, the results show the feasibility of this type of data-based controller and its potential to increase the efficiency and resilience of the system. In-situ test campaigns are needed to confirm these results.

Kurzfassung

Ziel dieser Arbeit ist es, die Verwendung neuartiger künstlicher Neuronaler Netze (NN) zur Modellierung komplexer Systemdynamiken zu erforschen, ihr Potenzial für den Einsatz in modernen Regelungstechniken zu bewerten und ihre Anwendbarkeit in einem modellprädiktiven Regler (MPC) für das Solarturmkraftwerk (STJ) des Deutschen Zentrums für Luft- und Raumfahrt (DLR) in Jülich zu untersuchen.

Die mehrschrittige Vorhersagefähigkeit eines NN mit Transformer-Architektur wird genutzt, um die Dynamik des nichtlinearen multivariablen Systems des Strahlungsempfängers auf der Basis realer Daten zu modellieren. Durch die Einbeziehung von Informationen über zukünftige Störungen oder Sollwertänderungen erleichtert das Vorhersageverhalten des Modells die Kompensation erwarteter externer Einflüsse. Diese Eigenschaft wird in einem MPC verwendet.

Zur Berechnung des MPC werden die Bibliotheken PyTorch und SciPy mit verschiedenen Optimierern getestet. Die erstgenannte Bibliothek wird, obwohl es sich um einen unbeschränkten Optimierer handelt, zur Lösung des beschränkten Problems mit Hilfe der vorgeschlagenen Schrankenfunktionen verwendet, um sein Potential zu erkunden.

In dieser Arbeit liefert der PyTorch-Ansatz ähnliche Lösungen wie die beschränkten Optimierer, hat aber schnellere Berechnungszeiten und eine bessere Leistung basierend auf den Simulationsergebnissen. Innerhalb der Datenverteilung, auf der das NN trainiert wurde, wird gezeigt, dass der vorgeschlagene NN-gesteuerte MPC in der Lage ist, den Sollwert zu verfolgen, die Nebenbedingungen zu erfüllen und Störungen oder Sollwertänderungen in den vorgeschlagenen Testszenarien abzulehnen. Zudem werden Vorhersagefehler in einem Gaussian Process Regressor (GPR) angepasst, um über die Unsicherheitsquantifizierung (UQ) Information zu gewinnen, die dem Systembediener angezeigt werden.

Für unsere Anwendung zeigen die Ergebnisse die Machbarkeit dieses datenbasierten Reglers und sein Potenzial, um die Effizienz und Widerstandsfähigkeit des Systems zu erhöhen. Testkampagnen vor Ort sind erforderlich, um diese Ergebnisse zu bestätigen.

Contents

Acronyms and Abbreviations

ASI All-sky imager

BFGS Broyden–Fletcher–Goldfarb–Shanno

CO2 Carbon Dioxide

COBYLA Constrained Optimization BY Linear Approximation

CSP Concentrated Solar Power

CSTR Continuous Stirred-Tank Reactor

DAE Differential-Algebraic Equation

DLR Deutsches Zentrum für Luft- und Raumfahrt e. V. (German Aerospace Center)

DNI Direct Normal Irradiance

DOF Degrees of Freedom

FOPDT First-Order Plus Dead-Time

GHG Greenhouse Gas

GP Gaussian Process

GPR Gaussian Process Regressor

GPT Generative Pre-trained Transformer

HTM Heat Transfer Medium

IEA International Energy Agency

IR Infrared

L-BFGS Limited-memory Broyden–Fletcher–Goldfarb–Shanno

LCOE Levelized Cost of Electricity

LLM Large Language Model

LP Linear Programming

LSTM Long Short-Term Memory

LTI Linear Time-Invariant

MAP Maximum A Posteriori estimation

MHE Moving Horizon Estimator

MIMO Multiple Input Multiple Output

ML Machine Learning

MLE Maximum Likelihood Estimation

MLP Multilayer Perceptron

MPC Model Predictive Control

MSA Multi Step-Ahead

MSE Mean Squared Error

NLP Non-linear Programming

NLP Natural Language Processing

NN Neural Network

OCP Optimal Control Problem

ODE Ordinary Differential Equation

OSA One Step-Ahead

OVR Open Volumetric Receiver

PID Proportional-Integral-Derivative

PV Photovoltaic

QP Quadratic Programming

RBF Radial Basis Function

RES Renewable Energy Source

RMSE Root Mean Squared Error

RNN Recurrent Neural Network

S-MPC Stochastic Model Predictive Control

SF Institut für Solarforschung (Institute of Solar Research)

SLSQP Sequential Least SQuares Programming

SNN Self-normalizing Neural Network

SNR Signal-to-Noise Ratio

STJ Solar Tower Power Plant Jülich

T-NN Transformer Neural Network

TES Thermal Energy Storage

Trust-constr trust-region interior point method

UBA Umwelt Bundesamt (German Environment Agency)

UQ Uncertainty Quantification

XAI Explainable Artificial Intelligence

Notation

Numbers and Arrays

a	A scalar (integer or real)
$\boldsymbol{a}$	A vector
$\boldsymbol{a}^*$	An optimal vector
$\boldsymbol{a}_{[1:N]}$	A sequence of vectors from 1 to N, such that $\boldsymbol{a}_{[1:N]} = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N)$
$\boldsymbol{a}_{[1:N]} = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N)$	
$\boldsymbol{A}$	A matrix
$\mathbf{A}$	A tensor
$\vec{\mathbf{E}}$	A sequence embedding
$\boldsymbol{I}$	Identity matrix with dimensionality implied by context
$\mathcal{C}$	Context window
p	Prediction horizon
w	Look-back window
$\vec{\mathbf{Q}}$	A vector product of a matrix vector multiplication
$\boldsymbol{W}$	A neural network weights matrix
b_i	A neural network bias

Sets and Graphs

$\mathbb{A}$	A set
$\mathbb{R}$	The set of real numbers
$\{0, 1\}$	The set containing 0 and 1
$\{0, 1, \dots, n\}$	The set of all integers between 0 and n
$[a, b]$	The real interval including a and b

Indexing

a_i	Element i of vector $\boldsymbol{a}$, with indexing starting at 0
a_{-i}	Last elements of vector $\boldsymbol{a}$ except for element
$A_{i,j}$	Element i, j of matrix $\boldsymbol{A}$
$\boldsymbol{A}_{i,:}$	Row i of matrix $\boldsymbol{A}$
$\boldsymbol{A}_{:,i}$	Column i of matrix $\boldsymbol{A}$
$A_{i,j,k}$	Element (i, j, k) of a 3-D tensor $\mathbf{A}$
$\mathbf{A}_{:,:,i}$	2-D slice of a 3-D tensor
$\dim(\boldsymbol{x})$	Dimension of vector $\boldsymbol{x}$

Linear Algebra Operations

$\boldsymbol{A}^\top$	Transpose of matrix $\boldsymbol{A}$
$\boldsymbol{A} \odot \boldsymbol{B}$	Element-wise (Hadamard) product of $\boldsymbol{A}$ and $\boldsymbol{B}$

Probability and Information Theory

P(a)	A probability distribution over a given b
P(a | b)	A probability distribution over a discrete variable
a $\sim P$	Random variable a has distribution P

$\mathbb{E}_{\mathrm{x}\sim P}[f(x)]$	Expectation of $f(x)$ with respect to $P(\mathrm{x})$
$\sigma^2(f(x))$	Variance of $f(x)$ under $P(\mathrm{x})$
$\mathrm{Cov}(f(x), g(x))$	Covariance of $f(x)$ and $g(x)$ under $P(\mathrm{x})$
$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$	Gaussian distribution over $\boldsymbol{x}$ with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$

Functions

$f(\boldsymbol{x}; \boldsymbol{\theta})$	A function of $\boldsymbol{x}$ parametrized by $\boldsymbol{\theta}$.
$NN(x)$	A neural network mapping function
$\log x$	Natural logarithm of x
$\sigma(x)$	Logistic sigmoid, $\dfrac{1}{1+\exp(-x)}$
$\|\|\boldsymbol{x}\|\|^2$	L^2 norm of $\boldsymbol{x}$
$MSE(\boldsymbol{y}, \hat{\boldsymbol{y}})$	Mean squared error between $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$
$RMSE(\boldsymbol{y}, \hat{\boldsymbol{y}})$	Root mean squared error between $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$

Datasets and Distributions

$\hat{p}_{\mathrm{data}}$	The empirical distribution defined by the training set
$\mathbb{X}$	A set of training examples
$\mathcal{D}$	The dataset with measured values

List of Figures

List of Tables

Chapter 1

Introduction

The purpose of this thesis is to explore the use of Transformer Neural Network (T-NN) to model complex system dynamics, evaluate their potential for use in modern control techniques, and investigate their applicability in an Model Predictive Control (MPC) controller for the Solar Tower Power Plant Jülich (STJ) at the Deutsches Zentrum für Luft- und Raumfahrt e. V. (German Aerospace Center, DLR) in Jülich.

1.1 Context and motivation

In response to the current global energy demands and pressing concerns regarding Greenhouse Gas (GHG) emissions and climate change driven by rising global temperatures [1], there has been a growing interest in promoting the use of Renewable Energy Sources (RESs) [2]. In 2023, Germany, for instance, achieved a reduction of Carbon Dioxide (CO2) emissions by 10.1%, according to the Umwelt Bundesamt (German Environment Agency, UBA) [3]. Besides the decrease in fossil fuel use as energy sources and an overall reduction in energy consumption, an increase in RESs has played an important role in achieving this result.

Despite these contributions, some technologies, although mature enough to prove economic viability for energy production, have the potential for increased efficiency that would enable scaled production and a higher market share. One such technology is Concentrated Solar Power (CSP) plants.

Solar thermal power, or CSP, primarily works by heating a Heat Transfer Medium (HTM) which can be used in an electric steam generator or in thermal storage. Solar irradiance is concentrated using tracking mirrors called heliostats to focus sunlight onto a receiver. Due to the high process temperatures that can be achieved, CSP can also be used for other industrial processes or to produce hydrogen or synthetic fuels, thus further contributing to sector decarbonization in addition to electric power generation.

According to the International Energy Agency (IEA), CSP plants are expected to remain 100% policy-driven until 2028 [2]. In its World Energy Outlook 2023 report, IEA calculates CSPs to represent 0.18% (16 TWh) of the world's total RES generation (8,599 TWh), and 0.19% (7 GW) of the world's total RES capacity (3,629 GW) [4]. This reflects a niche for further development that could make CSP more economically attractive and promote wider adoption by reducing the Levelized Cost of Electricity (LCOE) as an effect of increasing the number of suppliers in the energy market and improving overall system efficiency.

Ultimately, as remarked by the World Bank in its Concentrating Solar Power report, CSP with energy storage can absorb more energy from low-cost RESs, contributing overall to a lower-cost energy mix while increasing flexibility compared to Photovoltaics (PVs) [5]. An additional increase in the use of RESs can contribute to the European Union's 2050 long-term strategy to reach carbon neutrality [6], as well as to the United Nations Paris Agreement's goals [7].

To contribute to technology development, improving the efficiency and reliability of these systems is crucial. Modern control techniques can enhance disturbance rejection, improve set-point tracking of the HTM temperature, and enforce operational constraints. One such approach is Model Predictive Control (MPC).

MPC is an advanced process control method that uses a model of the system's dynamics and information about future state values, including possible future disturbance predictions. This makes it an appropriate candidate for research in solar thermal power plants. However, one of the main difficulties with this approach is that, although a sufficiently accurate model using first principles can be obtained, its complexity grows rapidly, and so does the computation time needed to obtain a numerical result.

Research has predominantly used simplified system models to address these challenges. Yet, only a few studies have explored data-based approaches, such as Machine Learning (ML) with NNs. Even fewer empirical investigations have tested these methods on real plants.

1.2 Objectives, scope, and contribution

The purpose of this thesis is to explore the use of novel artificial Neural Network (NN) architectures to model complex system dynamics and to evaluate their potential for use in modern control techniques. This is then utilized in the Solar Tower Power Plant Jülich (STJ) at the Deutsches Zentrum für Luft- und Raumfahrt e. V. (German Aerospace Center, DLR) in Jülich to investigate feasible efficiency and robustness improvements.

To achieve this goal, a NN with a Transformer architecture is used to predict the system dynamics for a certain prediction horizon in the future. The NN will then be

used as a model of the system in an MPC controller. An attempt to find an optimal combination of parameters for the MPC formulation is made by training with different prediction horizon and look-back window values.

The controller seeks to remedy the influence of external factors such as cloud coverage by using solar irradiance predictions. Lack of predictive behavior during disturbances could damage the materials or cause stresses in the receiver by driving the system outside safe operating conditions, such as maximum receiver surface temperature or allowable change of surface temperature. For instance, these issues can be caused by sudden changes in solar power during a cloud transition.

The aim of the controller is to provide a stable HTM temperature around a desired set point by controlling the inputs in a predictive manner in the presence of disturbances or changes in the set point while satisfying operation constraints.

To assess the performance of the controller, a second NN is trained with different initial conditions and randomly distributed training data to be used as the simulated true plant model in a software-in-the-loop analysis. Metrics on constraints violations, set-point Root Mean Squared Error (RMSE), total effort, and change of enthalpy are quantified to assess the results, based on four different testing scenarios.

The work also compares the use of constrained (SciPy minimize) and unconstrained (PyTorch) optimizers, and in addition to its primary focus, it provides a brief introduction to Gaussian Processes within the context of CSP control. While the UQ values will not be directly part of the design of the MPC controller, they aim to inform the operator and serve as an option to further investigate the feasibility and potential application of Gaussian Processes in this domain.

The raw data from the results cannot be disclosed directly. However, the results are presented in a normalized format for interpretation.

The key research question of this study is to explore the potential of Transformer Neural Networks to model system dynamics and investigate their applicability in an MPC controller for the Solar Tower Power Plant Jülich (STJ).

1.3 Structure

The overall structure of the thesis takes the form of nine chapters, including this introductory chapter. This work first provides in Chapter 2 an overview of CSPs, explaining the fundamentals of the Solar Tower Power Plant Jülich (STJ), the experimental setup, and the data used for this research. Chapter 3 illustrates the related state of the art. The following Chapter 4 offers a theoretical framework for Neural Networks (NNs), subsequently Model Predictive Control (MPC), and Uncertainty Quantification (UQ). Chapter 5 discusses the NNs architecture used and its components within an MPC controller. It is followed by Chapter 6, where the MPC controller is studied as a

solution to the OCP with constrained and unconstrained optimization tools. As a brief introduction to the topic, in Chapter 7, Gaussian processes are proposed as a tool for uncertainty quantification applied to solar thermal power plants. Finally, the Conclusions gives a brief summary and critique of the findings, followed by identifying and citing areas for further research in the Outlook.

The organization of this thesis is structured to resemble the progression of research, workflow, and findings that occurred during the development of this work. Hence, starting from Chapter 5, each subsequent chapter will outline its methods and present its findings.

Chapter 2

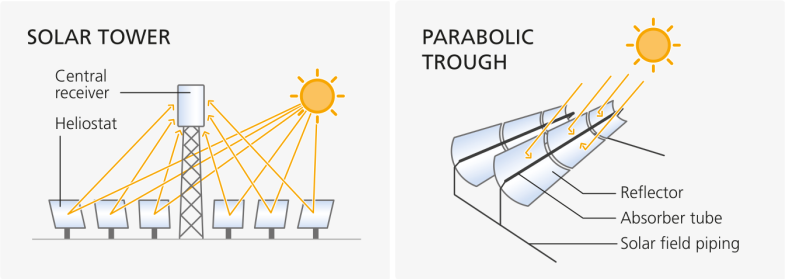
Experimental Setup

This chapter provides a brief overview of CSP plants and the fundamentals of the STJ, which is the core focus of this thesis. It also describes the experimental setup used for the research.

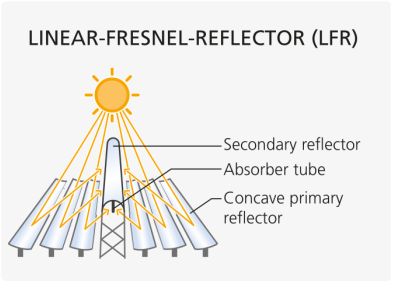
2.1 Concentrated solar power

Solar thermal power, or CSP, operates by heating a HTM using reflected solar irradiance. The heated HTM can then be used in an electric steam generator cycle, thermal storage, or industrial processes. Overall, examples of HTMs are molten salts, gases, solid particles and liquid metals, to name a few. The solar irradiance is concentrated onto a focal point or focal line using tracking mirrors called heliostats or reflectors depending in the system, and they focus or reflect the sunlight onto a receiver or absorber. The heat generated by the concentrated solar power is then transferred to the HTM as it passes through the receiver.
From the different types of solar thermal power plants that exist, Figure 2.1 depict the three most widely used [8]: solar tower (e.g. Figure 2.1a), parabolic trough (e.g. Figure 2.1b) and Linear Fresnel Reflector (LFR) systems (e.g. Figure 2.1c). In their own configuration, their core components are highlighted: Receiver or absorber, and heliostat or reflector.

An optional yet very advantageous element of CSPs is the Thermal Energy Storage (TES). Such a system stores thermal energy instead of electric energy, which is cheaper compared to other energy storages [4]. During operation, part or excess of the HTM's energy can be passed through the TES to store energy for later use. This contributes to the energy supply security and flexibility, since it enables the plant to operate during non-solar radiation periods using the energy stored. Because they can adjust power generation to demand flexibly, solar thermal power plants are also known as adjustable power plants [8].

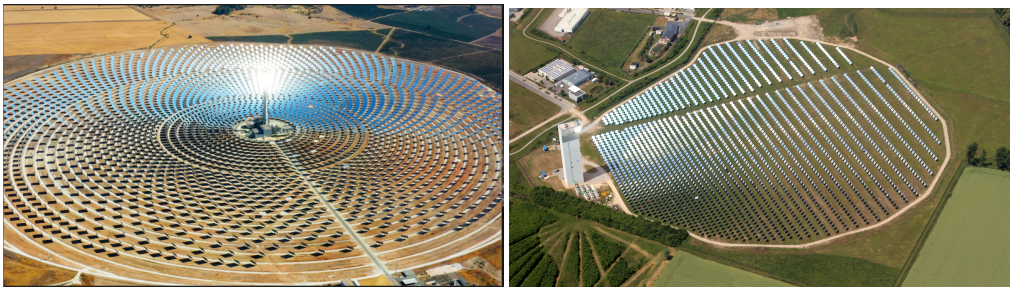


(a) Solar Tower CSP. (b) Parabolic Trough CSP.



(c) Linear Fresnel Reflector.

Figure 2.1: Different types of concentrated solar power plants. DLR [8].



(a) Surround field. Gemasolar, Spain. (b) Polar field. STJ, Germany.

Figure 2.2: Heliostat field layouts. Adobe Stock (left), DLR (right).

2.2 Solar tower power plant Jülich

The research facility in Jülich is operated by the Institut für Solarforschung (Institute of Solar Research, SF) at the Deutsches Zentrum für Luft- und Raumfahrt e. V. (German Aerospace Center, DLR). It is a solar thermal power with a solar tower configuration, thus the reflection of the solar irradiance is done by sun tracking heliostats. These mirrors have typically a 2-Degrees of Freedom (DOF) kinematic structure to direct the sunlight onto the receiver's surface.

Heliostats are placed and arranged with respect to the tower in two major configurations depicted in Figure 2.2: surround field (e.g. Figure 2.2a) like in Gemasolar CSP in Spain, and polar (e.g. Figure 2.2b), which is the configuration present at the STJ.

This heliostat field operating at the STJ is composed of over 2000 heliostats, each with a surface of 8.3 m^2. They are controlled by an in-house system called "HeliOS", where the target points are directed towards the receiver at a height of 55 m.

For HTM, the STJ utilizes ambient air to absorb the energy collected by the receiver and it is transported to either a thermal storage unit or a steam generator. This is possible thanks to the porous structure of the receiver's materials which allow the medium to flow through, absorbing the energy, hence the name of Open Volumetric Receiver (OVR).

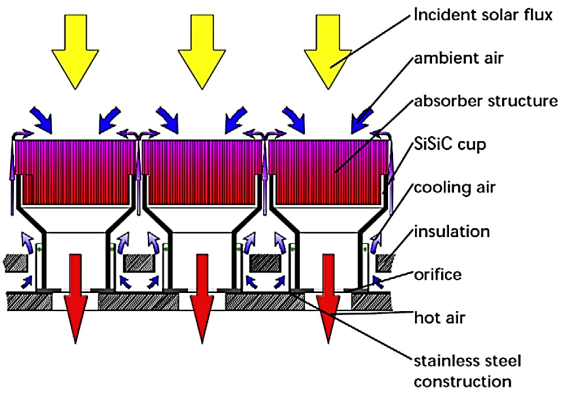
Among different solar tower CSPs configurations, the most widely adopted receivers are be tubular or open volumetric. The material of these receivers can be among others made of ceramic or metallic materials [9]. The system described in this work utilizes an open volumetric receiver, made of 1080 ceramic absorber cups, with a total surface of 22.7 m^2 [10].

Figure 2.3a show the assembly between the ceramic cups and the mounting surface of the receiver, as well as its different parts and the air interaction. The ambient fluid is drawn towards inside the cups due to the negative pressure created by a compressor in the system, while at the same time, the reflected incident solar flux heats the body of the ceramic cups. As the HTM passes through the receiver, it absorbs energy and is carried to the output of the receiver into the system through the orifice of the mounting apparatus.

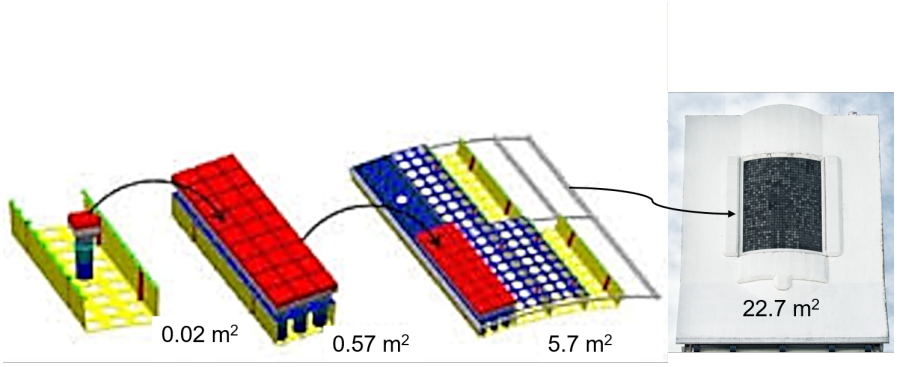
The tower's design considers air temperature reuse to increase performance. This is achieved by recirculating air back towards the receiver's forefront through gaps between the absorber modules as shown in Figure 2.3a. This approach attempts higher extraction of residual exergy inherent within the HTM. Is worth to notice, that the exposure of the fluid to ambient conditions facilitate direct interaction with the surrounding atmosphere thus make it susceptible to energy losses due to weather conditions.

The complete assembly of the tower's receiver is depicted in Figure 2.3b. It is composed of 4 quadrants with 10 holding frames each, where single cups mounted in a connecting stainless-steel pipe are connected to each frame in a 3 by 9 arrangement. Due to its ceramic materials, limitations on temperature gradients have to be considered for means of control. From a modelling perspective, the receiver's surface material, the structure of the porous absorbing cups, its assembly and the number of elements translate into a complex system of algebraic and Ordinary Differential Equations (ODEs) [15].

With the key components of a CSP described, in particular those of the STJ, an overall system description can be done. In Figure 2.4, the heliostat field, the receiver, the actuating compressor and the downstream processes are depicted. A heat storage and steam generation stages are shown for completion, but they are not directly part of the scope of this thesis, hence, the elements inside the dotted line are part of this



(a) Receiver cups construction. [11]



(b) Receiver assembly and surface. Adaptation. [12], [13], DLR [10].

Figure 2.3: Solar Tower Power Plant Jülich (STJ) receiver.

work.

One main component of this thesis is to explore the feasibility to model the STJ as a simplified system and estimate states based on direct and indirect measurements. Therefore, five variables are selected and explored in this thesis, shown in Figure 2.4: receiver's surface temperature T_{surface} , receiver's surface apparent brightness $I_{\dot{Q}}$, hot air temperature $T_{\text{hot air}}$, cold air temperature $T_{\text{cold air}}$, and air mass flow $\dot{m}_{\text{air}}$.

2.3 Measured data

The plant is equipped with a variety of sensors and cameras whose data is captured and stored in a central system for both online or offline use, allowing the data to be used for research purposes. Control and peripherals integration happen within an OPC UA system.

With respect to the measured variables that are used to model the system, the plant is equipped with air mass flow and temperature sensors. In particular, to indirectly measure the solar irradiance reflected by the heliostats onto the receiver the

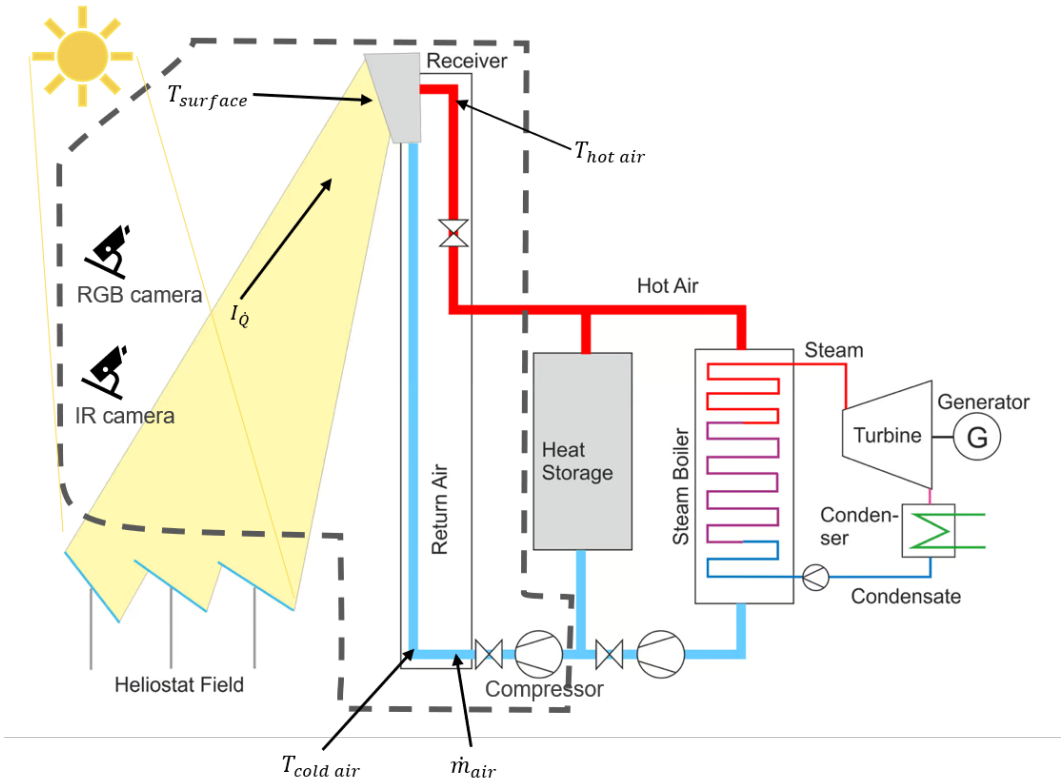
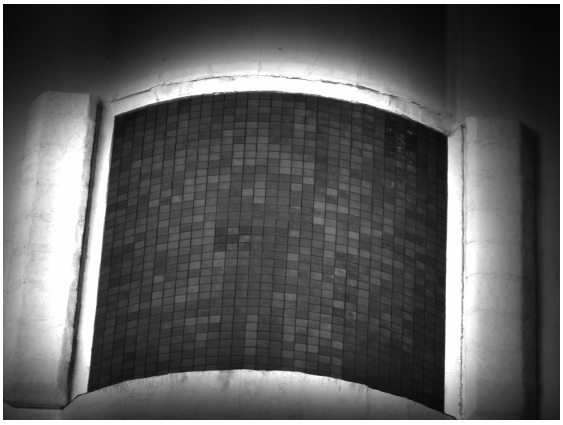


Figure 2.4: Diagram of solar tower power plant Jülich with open volumetric receiver. Adaptation. DLR, Pabst [14].

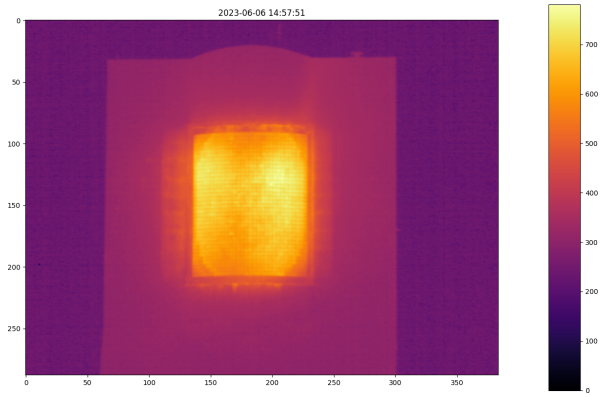
tower uses an optical gray-scale camera system pointing to the receiver. This gives the receiver's surface apparent brightness. These images can be used to make an estimated measurement of the irradiated flux density based on the apparent pixel brightness taken from the images of the surface [16]. Another important part is the receiver's surface temperature measurement system, which is composed by an Infrared (IR) camera.

In Figure 2.5a a sample image of the gray-scale camera system is shown. Images have 12-bit precision values and a resolution of 1392x1040 pixels. In Figure 2.5b a 384x288 resolution IR image of the receiver's surface temperature is shown along a scale bar. For later use, images are first rectified, stabilized and cut to the receiver's edges. Then pixels groups are averaged for a final image resolution of 30 by 36, equal to the number of ceramic absorbers in the receiver's surface.

For the variables $I_{\dot{Q}}$and T_{surface} , measured by the gray-scale and IR camera systems respectively, image-based methods offer an easier and cost-effective way to take the measurements compared to installing individual sensors directly around or in front each absorber cup. Additionally, temperature sensors are placed only behind a number of absorber cups. While the surface temperature could be measured directly,



(a) Receiver's gray-scale wrapped image



(b) Receiver's IR wrapped image with temperature bar

Figure 2.5: Solar Tower Power Plant Jülich (STJ) receiver. DLR.

the irradiated solar power is technically not viable and operational not feasible, since this will mean placing the instruments in between the heliostats and receiver.
To circumvent this issue, Thelen [16] developed an indirect measurement technique based on the receiver's reflection or brightness captured in gray-scale images. With the absorber and irradiance parameters a calibration process is performed. Measurements show correlation compared to direct techniques like Gardon and Kendall radiometers and Suncatch calorimeters, showing the viability of this measuring technique. In this thesis, the system's measured irradiance in $[W/m^2]$ in not directly used since it is purpose of study to explore the use of indirect measurements data as input to the NN. Instead, the mean of the pixels raw values (intensity, brightness) of the rectified images is considered. This is done as well for the surface temperature, with the difference that the values of the pixels correspond to the temperature directly. For the indirectly measured irradiation intensity (receiver's mean surface apparent brightness) $I_{\dot{Q}}$, the value is divided by the exposure time of the camera, and for the receiver surface temperature T_{surface} the value is taken as is.

To measure the hot air temperature $T_{\text{hot air}}$, 10% of the receiver's cups have a temperature probe, backed by a redundant system after the receiver's mounting modules junction with a number of sensors, allowing for direct measurement of the hot air after the receiver. Hence, this work uses the redundant system as they reflect the temperature of the HTM after passing through the complete receiver. The difference between the measured temperatures in the different zones of the receiver surface is outside the scope of this thesis as the irradiation distribution is considered to be uniform across the surface. Cold air $T_{\text{cold air}}$ is measured in a similar way. With a fixed number of sensors, air returning from the steam or storage process is measured before being exhausted towards the receiver surface in between the gaps as presented before. For training purposes, each series of hot and cold air measurements are averaged individually to make a single array of values for the NN input. Last, the air mass flow $\dot{m}_{\text{air}}$ driven by the system's compressor is measured using a flow indicating sensor. Therefore when measured, the five variables of interest (receiver's surface temperature T_{surface} , receiver's Receiver's mean surface apparent brightness $I_{\dot{Q}}$, hot air temperature $T_{\text{hot air}}$, cold air temperature $T_{\text{cold air}}$, and air mass flow $\dot{m}_{\text{air}}$) are scalars organized as time-series data or unidimensional arrays. Values are read and stored at a frequency of 1 [Hz].

Related to the receiver model, a short-term solar irradiance forecast system called "Nowcasting" is available [17], [18]. Composed of All-sky imagers (ASIs), pyrheliometers and weather stations, the system provides Direct Normal Irradiance (DNI) predictions considering the observed clouds for up to 60 minutes ahead. This data will later be used in the MPC as part of NN input to exploit the predictive behavior of the model in the controller formulation.

To present the structure of the dataset, Figure 2.6 shows an extract of the data collected which is available for this thesis. Values are normalized for visualization purposes. The measurements shown were captured on May 31st, 2023. During this period, a constant $\dot{m}_{\text{air}}$ (green line) was attained while the $I_{\dot{Q}}$(purple line) was varied. For each change in $I_{\dot{Q}}$, a change in the measured hot air temperature ($T_{\text{hot air}}$, orange line) is observed. At a later stage, the $I_{\dot{Q}}$is kept constant and $\dot{m}_{\text{air}}$ is changed to explore the influence in $T_{\text{hot air}}$. Different configurations of variables were tested and logged to try to capture a variety of system dynamics to be modelled by the NN. The fulfillment of a persistence of excitation requirement in the dataset is not part of this thesis, and therefore it is assumed that sufficient data is available to model the dynamic to some extent. To account for irradiation dynamics, the number of heliostats reflecting to the receiver is controlled. The heliostat(s) that are then manipulated to either focus or defocus are selected in a random manner from a pool of available heliostats. In difference to an homogeneous selection of mirror(s), the randomized selection decreases the influence of cloud shadows over the heliostat field by distributing the disturbance

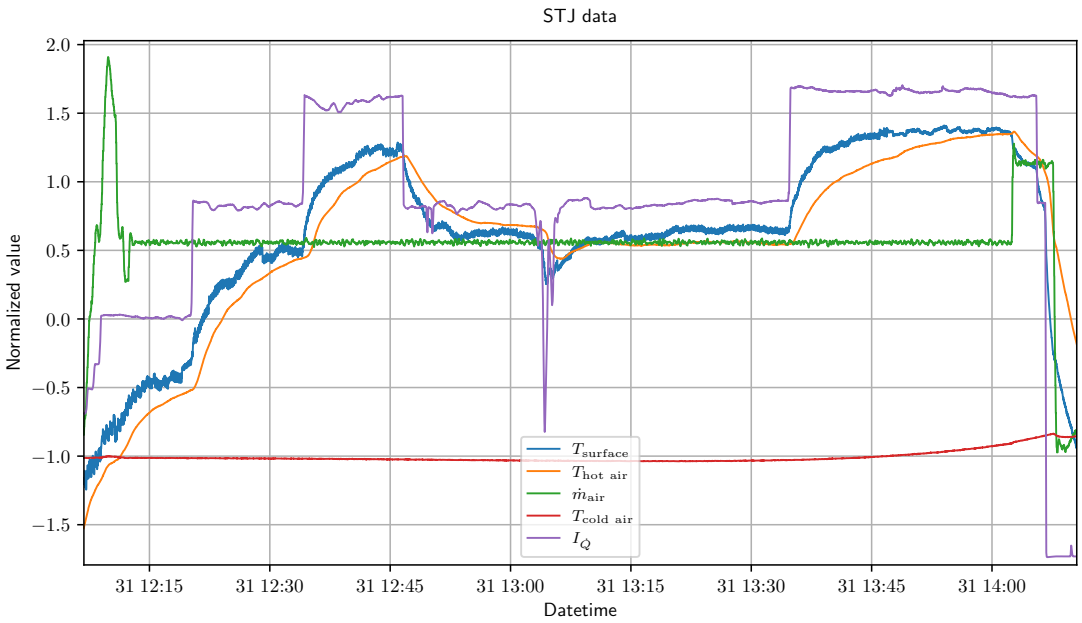


Figure 2.6: STJ normalized measured data.

across a random amount of heliostats and the available surface. For technical reasons unrelated to this work, more true measured data was not available for training.

In the following, the values of air mass flow $\dot{m}_{\text{air}}$, cold air temperature $T_{\text{cold air}}$ and receiver's surface apparent brightness $I_{\dot{Q}}$ are considered as state variables of the system. Receiver's surface temperature T_{surface} and hot air temperature $T_{\text{hot air}}$ the output variables. This is summarized in Table 2.1

Table 2.1: Solar tower Jülich model variables

Variable	Description	Units	Type
$\dot{m}_{\text{air}}$	Air mass flow	[kg/s]	controlled variable
$T_{\text{cold air}}$	Cold air temperature	[°C]	state variable
$I_{\dot{Q}}$	Receiver's mean surface apparent brightness	[1/s]	state variable
T_{surface}	Receiver's mean surface infrared temperature	[°K]	output variable
$T_{\text{hot air}}$	Hot air temperature	[°C]	output variable

In the context of NNs, these variables are visited again in Chapter 5. For the MPC, the variable air mass flow $\dot{m}_{\text{air}}$ is considered as the controlled variable and its use is further explained in Chapter 6.

Chapter 3

State of the Art

This chapter reviews the current advancements and ongoing challenges in CSP found in the literature. Key areas include Heat Transfer Medium (HTM), receiver configurations, system modeling, control techniques and uncertainty quantification methods.

3.1 Concentrated solar power

In the field of CSPs, research continuously contributes to topics such as HTMs, receiver arrangement and material, system modeling, and heliostat online control. For example, Yerudkar et al. [19] and the DLR [8] extensively present the different types of available technologies and offer a global energy-market-oriented perspective, focusing on the potential of CSPs plants and their economic viability as a driver for further technical development. However, these discussions often lack specifics on modeling and control techniques.

3.2 Heat transfer medium, materials, and receiver configuration

To address this, Merchán et al. [9] review technological advances in HTMs, solar receiver materials and configurations, as well as options for TES. This is shown in the work of Ding and Bauer [20], who analyze current and new research on HTM materials. These materials are studied in the context of ”next generation” CSP plants, where higher process temperatures can be achieved, leading to higher thermal efficiencies. More research is needed to reliably compare these materials and their viability. Directly related to the STJ, Capuano et al. [21] examine ceramic absorbers on the surface of the receiver and provide a baseline mathematical model of this solar tower plant. The study highlights the influence of variable environmental conditions on the model's

accuracy compared to real data, leaving room for further analysis and suggesting the mitigation of numerical complexity due to higher-order models.

3.3 Modeling and control

To enhance system efficiency through control, an accurate system model is needed. This model should account for system dynamics as well as disturbances. In CSPs, disturbances mainly come in the form of passing clouds that obstruct sunlight reaching the reflecting mirrors, thus reducing the reflected irradiated solar power. Since the nominal operation of a plant is usually planned in advance, the desired set point can also be inferred. Another disturbance or time-variant influence could be the energy demand of the grid, but it is considered decoupled thanks to the use of TES. Therefore, it can be considered negligible for the controllers but may affect the cold side of the HTM. Additionally, all plants have inherent minimum and maximum operating conditions for safe operation. A control mechanism that can use this information is beneficial for increasing system efficiency by extending the lifetime of parts, reducing maintenance and operation costs, and stabilizing energy generation. Literature related to CSPs control can be divided into two groups: receiver model control and irradiance aim point control.

Receiver model control

Given this context, Hirsch et al. [22] and Gall et al. [15] model the dynamics of the STJ using the modeling language Modelica. They introduce the use of Differential-Algebraic Equation (DAE) solving software in combination with an MPC controller. The main limitation of these studies is the lack of detail in their mathematical methods and OCP formulation. The latter considers a linear MPC, and full nonlinear model analysis was left open.

Another example in this area of research can be found in Popp et al. [23], where a reduced model of a CSP receiver with molten salt as HTM is studied. However, this paper considers only a traditional Proportional-Integral-Derivative (PID) controller, and the model accuracy is limited to this type of material. Later, with advances in computation power, software capabilities, and modern technologies, novel approaches revisited and tried to combine CSP dynamic models with control techniques. In his later work, Popp et al. [24] compares the PID controller against an MPC formulation using the simplified model and the HTM mass flow as a single controlled variable.

In a new approach, Iding et al. [25] proposed a single absorber cup simplified model based on Gall et al. [15], combined with an MPC to control the HTM temperature, showing good validation results against real data from the STJ. The model was then

extended to all receiver cups, but although the results showed significantly worse performance, it highlighted the importance of the inhomogeneous parameter set and system complexity.

Aim point optimization techniques and control

Surveys such as those conducted by García et al. [26], [27] delve into heliostat aim point control and related solar irradiance on the receiver based on a model published by Sánchez-González [28]. These works propose a control loop methodology and group subsets of heliostats as controlled variables in both steady state [26] and transients [27]. In the realm of dynamic aiming, Zhu and Dong [29] compare a particle swarm optimization algorithm as a control strategy against an MPC formulation.

Similar to Belhomme et al. [30], where the heliostat's aim point is explored as a controlled variable, Geschonneck [31] maximizes the receiver's absorption power in an OCP formulation based on previous models. The irradiated power, as a function of the heliostat aim point, is then controlled by an MPC in response to the measured states, specifically the receiver's surface temperature and the HTM outlet temperature.

Likewise, Ostermann [32] extends the work of Geschonneck [31] by including the HTM temperature directly in the MPC control objective as a second controlled variable. To account for the simplified model mismatch, the author uses a Moving Horizon Estimator (MHE) that minimizes the difference between the simplified model and the measured states. Despite these advances, in-situ experiments are still needed to validate the results, and the parameters of the MHE are subject to tuning.

Regarding modeling and considering the additional benefits of modern control techniques, the addition of predicted state values in the model can be achieved. Samu et al. [17] extend cloud and solar irradiance prediction, which Geschonneck [31] integrates into an MPC controller. Geschonneck evaluates the performance of this system in proposed test cloud shadowing scenarios, considering both accurate and inaccurate predictions. The results demonstrate the potential to enhance the overall efficiency and reliability of CSPs.

3.4 Data-based control

Despite breakthroughs in CSP control strategies, research consistently shows the complexity and required balance between computation time and model accuracy. Until recently, few studies have explored the connection between solar thermal power plants and data-driven control systems, which aim to identify the system and describe its dynamics based on real data. Ruiz Moreno [33] demonstrates one such integration, where data from a solar parabolic-trough plant with an MPC controller is collected.

This data is later used to train an Multilayer Perceptron (MLP) NN to learn the policies of the controller, thus reducing computation time with comparable mean results.

Another approach is shown in Pargmann et al. [34], where heliostat calibration data is used in a Self-normalizing Neural Network (SNN) to improve the calibration method. Results show an increase in calibration accuracy compared to current algorithms.

More literature can be found outside data-driven control applied to CSPs. For instance, Wong et al. [35] explore the use of an Recurrent Neural Network (RNN) to model a pharmaceutical manufacturing process described as a Continuous Stirred-Tank Reactor (CSTR) to make multiple single-step-ahead predictions (multiple shooting) of the system and control it using an MPC. However, a weakness of the study is the use of a numerical model of the equations to generate training data, deviating from the use of real process data and falling short in the study of its applicability. Utama et al. [36] take a similar approach to Ruiz Moreno [33], where the MPC controller policies are used as training data for a deep learning NN in an energy management system test case. The key difference lies in the use of Explainable Artificial Intelligence (XAI), providing insights on how the NN arrived at a given result through a series of techniques.

An attempt to explore the use of Long Short-Term Memory (LSTM) NNs as a system model in an MPC controller is made by Jung et al. [37], where the focus is on the impact of different solvers and differentiation algorithms used in the OCP formulation in an extensive analysis. Compared to other ML architectures, this type of NN showed good performance in representing the system dynamics against ODE models of a two-tank problem and CSTR study cases. Challenging the performance results of the LSTM architecture and with the growing adoption in NNs of the Self-Attention mechanism presented by Vaswani et al. [38], Park et al. [39] explore and measure the performance of a T-NN in three study cases. These report better performance compared to other types of architectures. The paper also highlights the exploitable capability of T-NNs to make Multi Step-Ahead (MSA) predictions, reducing computation time in comparison to recursive approaches like LSTM and RNN models, while also addressing the vanishing gradients problem.

3.5 Uncertainty quantification and model predictive control

Previous publications have not addressed the inherent problem of model uncertainty, which arises when assumptions and simplifications are made in mathematical models. This is especially true in NNs, where the architecture is adapted to the training data, but an accurate generalization of a system is not achieved and its representation is highly depending on the regions at which it was trained, making it potentially unreliable outside of the data distribution. Therefore, there is increasing interest in UQ

methods and their possible integration with ML. UQ seeks to characterize and quantify the discrepancy between the model and the true observed data. The sources of uncertainties may vary, from its presence in the model parameters, modelling and simplifications, measurement and output noise, delays, disturbances, among others [40].

Different UQ methods are applied to various fields of study, types of data, and experiments. In this work, interest is focused on time series values, as they constitute the core component of the data-driven control method used. One UQ technique that has gained high interest and adoption in research is Bayesian optimization, particularly Gaussian Processes, which have proven to be a useful non-parametric tool that fits a regression with mean and standard deviation as a function of its kernel.

Gaussian processes

The papers of Kocijan et al. [41]–[43] and Likar and Kocijan [44] introduce a GP-based MPC model. This is later extended to demonstrate how this method can be used for data-based system identification and dynamic system control [45].

Regarding the use of UQ applied to CSPs plants, little research has been done. One example is presented by Luo [46], who performs a multi-objective robust optimization of a molten salt thermal solar power plant under uncertainty using a model approximation and solved by a Monte Carlo simulation and simulated annealing. On the other hand, Mohammadzadeh et al. [47] use UQ in a stochastic mixed integer linear program formulation to dispatch electrical energy and maximize expected profit with a CSP plant as a generator.

Despite the lack of literature in this area, possible approaches can be explored in other fields of engineering. For example, Torrente et al. [48] use a first-principles model of a quadrotor, where the discrepancies of the model are fitted with a GPR and used in an MPC for trajectory control. Eckel [49] explores its applicability for controlling towing kites, where the dynamics of the system are modeled both online and offline depending on the flying regime and conditions. Elsheikh and Engell [50] propose a hybrid MPC approach consisting of first-principles and data-based models. Training regions are identified using a support vector machine, and an online GP model error model is added when the controller is outside of that region. A significant limitation of the previous works is that the authors' methods do not consider the use of the standard deviation or variance information that the UQ method provides, but rather only the mean of the model error. Lastly, in an attempt to take UQ a step further and integrate it into NNs, Fiedler et al. [51] present a NN with a Bayesian last layer and explore its extrapolation capabilities. This enables direct quantification of the output uncertainty, addressing the increasing complexity of fitting a GP for large datasets.

Stochastic model predictive control

Another approach to leveraging UQ to account for model mismatch involves its application within an OCP formulation that results in an Stochastic Model Predictive Control (S-MPC). Currently, no publications investigate the use of S-MPC in CSP plants, but much can be learned from other areas of study where this is applied. Langåker [52] uses a GP to model a four-tank system and a car system for obstacle avoidance. Hewing et al. [53], [54] and Kabzan et al. [55] explore its use in autonomous systems based on first-principles models, the first in an embedded system for miniature race cars and the latter for competition racing. Bradford et al. [56] design an MPC controlled with Monte Carlo samples of an offline GPR for constraint tightening in a semi-batch bioprocess case study. Additionally, model prediction sub-setting by stochastic admissible solutions is proposed by Wabersich and Zeilinger [57], and Mesbah et al. [58] conduct extensive research into ML-enabled MPC under uncertainty. For trajectory control, Polcz et al. [59] use this approach to control a robotic arm, and Fiedler and Lucia [60] present a multi-step prediction S-MPC using a linear state-space system in combination with a GP. Conversely, an approximation of data-based dynamics can be expressed as Linear Time-Invariant (LTI) systems by calculating the Henkel matrix. This is later transformed to a state-space representation, where the mean and standard deviation of the GP can be used in an S-MPC fashion, as shown by Pan et al. [61].

Chapter 4

Fundamentals

This chapter introduces the theoretical background for artificial neural networks as black-box system dynamics models, along with the necessary elements for this thesis. It also covers the theory behind Model Predictive Control (MPC), which forms the basis for the methods used in this study, and briefly introduces Uncertainty Quantification (UQ) methods.

4.1 Neural network models

There are different approaches to create models of dynamic systems: first-principles models (white-box models), typically defined by ODEs or state-space representation, gray-box models, and black-box models. Black-box models identify the dynamics using system data by fitting the best combination of parameters. Approaches like nonlinear ARX (auto-regressive exogenous) (NARX) and Hammerstein-Wiener are available [62], but they have limitations in fitting certain dynamics [50], [63]. In these cases, a ML model, particularly a Neural Network (NN) model, can be beneficial [64], [65].

The dataset $\mathcal{D}$ used in the system identification step will be fit to a regression model. $\mathcal{D}$ consists of $\mathbb{X}$ and $\mathbb{Y}$ [65]. $\mathbb{X}$ contains pairs of states $x_k \in \mathbb{R}^x$ and inputs $u_k \in \mathbb{R}^u$, and $\mathbb{Y}$ contains outputs $y_k \in \mathbb{R}^y$ with $k = \{0, 1, \ldots, p\}$. The systems this work aims to identify are nonlinear discrete-time systems of the form [66]:

$$x_{k+1} = f(x_k, u_k), \tag{4.1a}$$

$$y_k = h(x_k, u_k) \tag{4.1b}$$

A NN is a mathematical function that approximates data values from $\mathbb{X} = \{x, u\}$ by mapping and estimating the output $\hat{y}$. The model can be written in compact form as $\hat{y}_k = NN(x_k, u_k, w)$, where $NN(\cdot)$ is the function represented by the neural net-

work and w the vector of parameters [67]. The perceptron, first introduced by Rosenblatt [68], was inspired by the behavior of neurons in the human brain (synapses). This perceptron emulates activation through activation functions, learning the relationship (mapping) of the inputs to represent an output. The equation of a single layer perceptron is [67]:

$$\hat{y} = \phi(b_0 + \sum_{i=1}^{n} w_i u_i) \tag{4.2}$$

where $\hat{y}$ is the output of the perceptron, $\phi(\cdot)$ is the activation function, b_0 is the bias term, w_i are the weights associated with the input features u_i, and n is the number of input features. Weights and biases $w_i, b_0 \in \mathbb{R}$ are model parameters that are learned during training by back-propagation [69].

Activation functions introduce non-linearity into neural networks, enabling the learning of complex patterns [64], [67]. In literature, commonly used activation functions $\phi(\cdot)$ include Sigmoid [67], [69], hyperbolic tangent (tanh) [67], Rectified Linear Unit (ReLU) [70], and Softmax[38], [71]. These are defined as:

$$\sigma(x) = \frac{1}{1+e^{-x}}, \tag{4.3a}$$

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}, \tag{4.3b}$$

$$\text{ReLU}(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}, \tag{4.3c}$$

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}. \tag{4.3d}$$

The sigmoid function (Equation 4.3a) maps the input to a value between 0 and 1, while the tanh function (Equation 4.3b) maps the input to a value between -1 and 1. On the other hand, the ReLU function (Equation 4.3c) outputs the input directly if it is positive; otherwise, it outputs zero. The Softmax function (Equation 4.3d) is used in the output layer of a classifier to represent a categorical probability distribution.

In this context, several layers and configurations of perceptrons can be cascaded to increase accuracy and learn more complex dynamics [72]. This is described as multi-layer perceptron [67]. As research progressed, NNs emerged and evolved beyond multi-layer perceptrons in diverse architectures to address a broader spectrum of problems such as regression and classification [64].

For black-box models, data is typically stored in a timed or sequential manner, implying that the current system state is a function of previous states [73]. To learn the effect of this sequential behavior and make a regression that better fits the data, RNNs were created. RNN is an architecture that interconnects ”cells” that interact with past cells. The hidden state h_k uses the input at time k to make predictions y_{k+1}. It then

passes the prediction at time $k+1$ in a recurrent manner through sequentially connected cells to propagate the state up to a certain time in the future $k+p$, where p is the number of recurrent cells. This is defined as [74], [75]:

$$h_k = \phi_{\text{hidden layer}}(W_{RNN} \cdot h_{k-1} + W_U \cdot x_k + b_h), \tag{4.4a}$$

$$\hat{y}_k = \phi_{\text{output layer}}(W_y \cdot h_k + b_y), \tag{4.4b}$$

with x_k (input at discrete time step k), h_k (hidden state), W_{RNN} (weight matrix for the recurrent connections), W_U (weight matrix for the input connections), W_y (weight matrix for the output connections), b_h (bias term for the hidden state), and b_y (bias term for the output).
The drawback of this approach is the loss of information during training in the form of vanishing gradients, which worsens with longer recurrent networks [76]. An attempt to overcome this challenge was implemented in Long Short-Term Memory (LSTM) neural networks, a variation of RNNs. This architecture introduces cells that deliberately forget short-term information to store long-term dependencies in sequential data. This is achieved with a memory cell structure composed of three gates: the input gate, forget gate, and output gate, along with a cell state. The LSTM model presented by Staudemeyer and Morris [77] is given by:

$$C_k = f_k \odot C_{k-1} + i_k \odot \tilde{C}_k, \tag{4.5a}$$

$$i_k = \phi(W_{xi}x_k + W_{hi}h_{k-1} + b_i), \tag{4.5b}$$

$$f_k = \phi(W_{xf}x_k + W_{hf}h_{k-1} + b_f), \tag{4.5c}$$

$$\tilde{C}_k = \tanh(W_{xc}x_k + W_{hc}h_{k-1} + b_c), \tag{4.5d}$$

$$o_k = \phi(W_{xo}x_k + W_{ho}h_{k-1} + b_o), \tag{4.5e}$$

$$\hat{y}_k = h_k = o_k \odot \tanh(C_k), \tag{4.5f}$$

with C_k (cell state at discrete time step k), i_k (input gate output), f_k (forget gate output), $\tilde{C}_k$ (new candidate values to be added to the cell state), o_k (output gate output), h_k (hidden state/output), x_k (input at discrete time step k), W_{xi}, W_{hi}, W_{xf}, W_{hf}, W_{xc}, W_{hc}, W_{xo}, W_{ho} (weight matrices for input, hidden state, and output connections), and b_i, b_f, b_c, b_o (bias terms for input, forget, candidate, and output gates)[75].

Although these architectures are extensively used as surrogate models, they are limited by their memory and sequential structure [78]. To exploit parallel computation and include context information, the Transformer Neural Network (T-NN) architecture using the "Attention mechanism" was created [38]. This is further detailed in Section 4.3.

4.2 Residual connection and persistence model

In traditional neural network architectures, each layer learns a mapping from its input to its output. However, as networks become deeper, as in the case of some RNNs, where deeper layers are needed to include more information, they can become more difficult to train due to the vanishing or exploding gradient problems. This means that gradients diminish or take large values as they propagate through the layers of the network, losing information needed to update the parameters and weights of early layers effectively during back-propagation [79].

Residual connections alleviate this issue by introducing interconnections or shortcuts that allow the gradient to bypass certain layers, enabling the network to learn functions more directly. This approach uses the difference between the input and the output of a layer, also called residuals, rather than trying to learn the entire mapping. The approach arises from the observation that certain architectures or layers of a network benefit from learning the residual mapping instead of the full mapping. He et al. [79] introduced this approach in deep neural networks for image recognition.

Figure 4.1 shows a generalized structure and the connection shortcut made by the residual connection. An application of this is shown in Section 4.3. Let $F(\cdot)$ be the function or mapping done by the layer i to n of the neural network. x is the input of the function, $G(x)$ the output of the last layer, and $F(x)$ the output of the function block. Without residual connection (e.g. Figure 4.1 left), the output $F(x)$ of the function $F(\cdot)$ is equal to $G(x) = F(x)$. If $F(\cdot)$ makes a perfect representation of x, then it is equal to $F(x)$, but if there is a deviation in this mapping, there is no element to account for this residual. In other words, $F(\cdot)$ has to learn $F(x)$. Instead, with a residual connection (e.g. Figure 4.1 right), the output $F(x)$ of the function $F(\cdot)$ is equal to $G(x) + x$. Therefore, $F(\cdot)$ has to learn $G(x) = F(x) - x$, facilitating the identity mapping $F(x) = x$ to adjust the weights and biases inside the layer(s).

Residual connections simplify the learning process in deep networks by focusing on the residuals. When these residuals are zero, the network essentially performs identity mapping, meaning the input is directly transferred to the output without alteration. Consequently, this leads to the natural development of persistence in the network's behavior. If the residuals are zero, the network's output matches its input, aligning with the persistence model's idea of maintaining consistency.

While a residual connection attempts to improve learning across deep networks, a baseline is needed to check if a model has learned the dynamics of the system. In the context of NN models, a persistence model refers to a simple baseline model that predicts future values in a time series based only on the current or previous values, effectively using the persistence effect seen when residuals equal zero. This is also known as the naive approach [80].

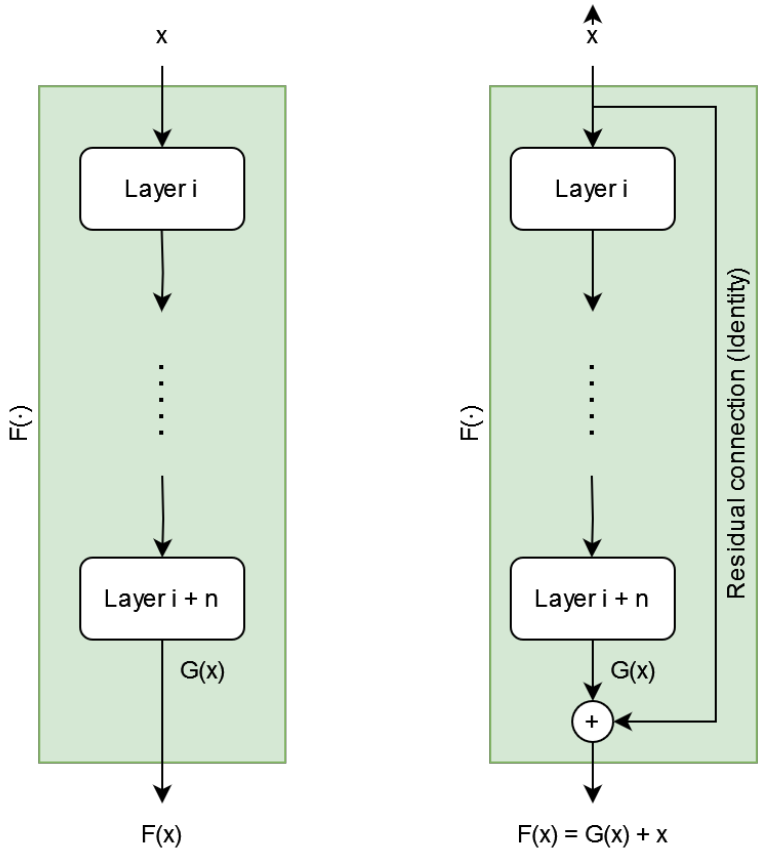


Figure 4.1: Feed-forward neural network layers with (right) and without (left) residual connection

A persistence model is defined as $\hat{y}_{k+1} = y_k$, where $\hat{y}_{k+1}$ is the predicted value at time step $k+1$, and y_k is the observed value at time step k. After training a NN, a persistence model can be used as a reference to compare the performance of trained models. If a trained NN has a lower loss value than the persistence model, it suggests that the network has learned from the data beyond what can be captured by simple persistence.

4.3 Transformer neural network

Driven by the limited memory scalability of recurrent neural networks and their variant LSTM NNs in the context of Large Language Models (LLMs), Vaswani et al. [38] introduced a new architecture in 2017 named Transformer. This NN was built upon the "Attention mechanism" proposed by Bahdanau et al. [81]. Initially, it was used for language translation, but its context awareness soon found applications in other fields, including LLMs and Generative Pre-trained Transformers (GPTs) based on Natural Language Processing (NLP).

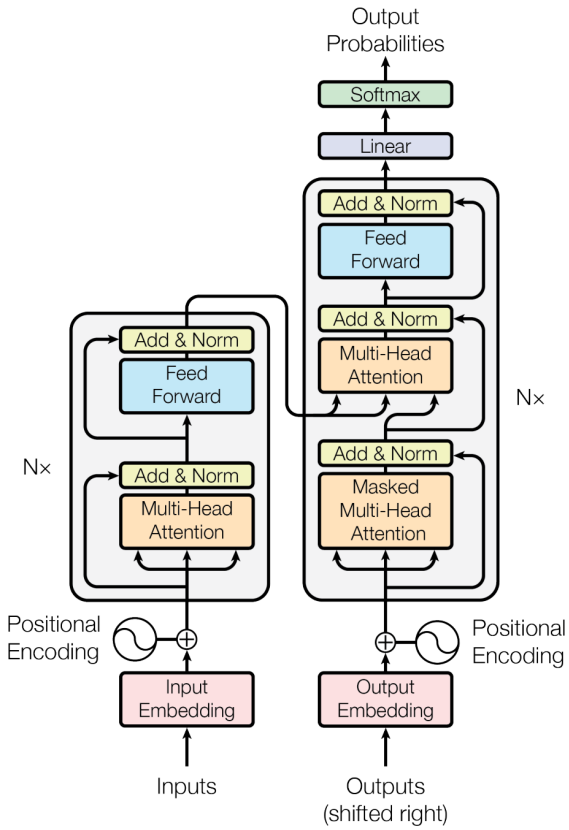


Figure 4.2: The Transformer - model architecture. Vaswani et al. [38]

The term "Transformer" reflects the model's ability to transform representations during training by adapting its weights to include contextual information in its input. In this way, the predictions carry a refined representation of what the value should be based on the context. With this approach, a full memory of the input is not needed. Instead, the NN updates its prediction by adding a change that better encodes the context [82]. As an added benefit, the formulation of the "Attention Pattern" computes a prediction for each element of the input, meaning that a Multi Step-Ahead (MSA) prediction is possible as a single forward pass in the network [39].

Figure 4.2 shows the original "Transformer model architecture" as presented by Vaswani et al. [38]. It consists of an encoder-decoder structure. Inside each encoder, input is encoded into embeddings $\vec{\mathbf{E}}_i$ and fed into a given number of transformer blocks added sequentially, each composed of a "multi-head attention" layer, a residual connection, and a pass through a feed-forward layer. This structure also makes use of a positional encoding to embed positional information into the embeddings. The output of this architecture gives the probabilities relation between the embeddings.

In NLP and GPTs, the length of the input and therefore the number of embeddings i is variable, but in this work, it is assumed to be constant. Since the whole length of the input is considered, the context window $\mathcal{C}$ is equal to the number of embeddings i.

Additionally, in LLM, words are said to be embedded or "transformed" into vectors of a higher-dimensional space, represented as elements of $\mathbb{R}^{d_e}$, where d_e is the dimension of the embedding space. This enables the model to learn a higher number of distinct ways to represent the input. In the case of time-series predictions and physical system dynamics regression used in this thesis, the dimension of the embedding space $\mathbb{R}^{d_e}$ is equal to the number of variables.

The key contribution of the T-NN architecture is the attention mechanism, also referred to as scaled dot-product attention. It decomposes the embedded input into query, key, and value vectors ($\vec{\mathbf{Q}}_i$, $\vec{\mathbf{K}}_i$, and $\vec{\mathbf{V}}_i$), which together represent the contextual influence of the surrounding embeddings. The attention mechanism is given by the following equation [38]:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_{qk}}}\right)V \tag{4.6}$$

with Q (query matrix), K (key matrix), V (value matrix), d_{qk} (query-key space dimension), and softmax(·) (softmax function).

The query matrix Q is used to compute attention scores indicating the relevance of each element in the key matrix K with respect to the queries in Q. In other words, how much the embeddings $\vec{\mathbf{E}}_i$ of the keys "attend to" the embeddings $\vec{\mathbf{E}}_i$ of the queries, or where the queries are focusing on based on the input.
These attention scores represent the probability of an embedding $\vec{\mathbf{E}}$ influencing another; a contextual meaning. They are then used to weigh the corresponding elements in the value matrix V, producing the output of the attention mechanism.

Matrices Q, K, and V are the horizontal concatenation of the query, key, and value vectors $\vec{\mathbf{Q}}_i$, $\vec{\mathbf{K}}_i$, and $\vec{\mathbf{V}}_i$ respectively (Equations 4.7). These vectors are obtained by multiplying the corresponding weight matrices $\boldsymbol{W}_Q$, $\boldsymbol{W}_K$, and $\boldsymbol{W}_V$ by each of the embeddings $\vec{\mathbf{E}}_i$. The weight matrices are parameters of the network, and their values are calculated during training via back-propagation.

$$Q_{[1:i]}, \tag{4.7a}$$
$$K_{[1:i]}, \tag{4.7b}$$
$$V_{[1:i]}, \tag{4.7c}$$
$$\vec{\mathbf{Q}}_i = \boldsymbol{W}_Q\,\vec{\mathbf{E}}_i, \quad \vec{\mathbf{Q}}_i \in \mathbb{R}^{d_{qk}}, \tag{4.7d}$$
$$\vec{\mathbf{K}}_i = \boldsymbol{W}_K\,\vec{\mathbf{E}}_i, \quad \vec{\mathbf{K}}_i \in \mathbb{R}^{d_{qk}}, \tag{4.7e}$$
$$\vec{\mathbf{V}}_i = \boldsymbol{W}_V\,\vec{\mathbf{E}}_i, \quad \vec{\mathbf{V}}_i \in \mathbb{R}^{d_v} \tag{4.7f}$$

with $\boldsymbol{W}_i^Q \in \mathbb{R}^{d_{qk}}$, $\boldsymbol{W}_i^K \in \mathbb{R}^{d_{qk}}$, $\boldsymbol{W}_i^V \in \mathbb{R}^{d_v}$ (query, key, and value weight matrices).

An important component of Equation (4.6) is the softmax function. This can be interpreted as the calculation of the probability of an event (or embedding $\vec{\mathbf{E}}$) given the other. This can be written as [38]:

$$\text{Attention}(Q, K, V) = P(K \mid Q)V, \tag{4.8a}$$

$$P(K \mid Q) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_{qk}}}\right), \tag{4.8b}$$

$$\text{where,} \quad \text{softmax}(\cdot) = P(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{\mathcal{C}} e^{z_j}} \tag{4.8c}$$

The scaled dot-product attention (Equation (4.8b)) is one approach to the attention mechanism. It refers to the dot-product of Q and K (or written in compact form as QK^T) divided by $\sqrt{d_{qk}}$ for numeric stability. It is called "self-attention" when Q, K, V are all from the same sequence of embeddings [38], in contrast to "cross-attention" where the matrices come from different sequences [81].
When the embeddings $\vec{\mathbf{E}}_i$ of the keys "attend to" the embeddings $\vec{\mathbf{E}}_i$ of the queries, the dot-product results in a positive value, suggesting that the vectors are related. If the value is zero, the vectors are unrelated, and if negative, they are opposite. The result, after being normalized and mapped into a distribution function, is called the "attention score." Presented in a matrix form, it is also known as the "attention pattern." A representation of this pattern is shown in Figure 4.3, before normalization and softmax(·) for simplicity.

Query Key Pair	$\vec{\mathbf{E}}_1$	$\vec{\mathbf{E}}_2$	**...**	$\vec{\mathbf{E}}_j$
$\vec{\mathbf{E}}_1$	$\vec{\mathbf{Q}}_1 \cdot \vec{\mathbf{K}}_1$	$\vec{\mathbf{Q}}_1 \cdot \vec{\mathbf{K}}_2$	...	$\vec{\mathbf{Q}}_1 \cdot \vec{\mathbf{K}}_j$
$\vec{\mathbf{E}}_2$	$\vec{\mathbf{Q}}_2 \cdot \vec{\mathbf{K}}_1$	$\vec{\mathbf{Q}}_2 \cdot \vec{\mathbf{K}}_2$	...	$\vec{\mathbf{Q}}_2 \cdot \vec{\mathbf{K}}_j$
$\vec{\mathbf{E}}_3$	$\vec{\mathbf{Q}}_3 \cdot \vec{\mathbf{K}}_1$	$\vec{\mathbf{Q}}_3 \cdot \vec{\mathbf{K}}_2$	...	$\vec{\mathbf{Q}}_3 \cdot \vec{\mathbf{K}}_j$
...	...	...	...	...
$\vec{\mathbf{E}}_i$	$\vec{\mathbf{Q}}_i \cdot \vec{\mathbf{K}}_1$	$\vec{\mathbf{Q}}_i \cdot \vec{\mathbf{K}}_2$	...	$\vec{\mathbf{Q}}_i \cdot \vec{\mathbf{K}}_j$

Figure 4.3: Attention pattern. In self-attention, the sequences and corresponding embeddings are the same, therefore $i = j$ and $d_{\text{e}\,1} = d_{\text{e}\,2}$. When using cross-attentions, the sequences are different from one another, therefore $i \neq j$ and $d_{\text{e}\,1} \neq d_{\text{e}\,2}$.

As the last part of the attention head, this score is multiplied by the matrix V. This results in the weighted sum of the contributions of each embedding to the meaning of the others. In the literature, the result of the attention layer is written as the matrix Z, which is then fed to the feed-forward NN.

Each feed-forward layer applies a linear transformation followed by a non-linear activation function to capture the nonlinear dynamics. This is done by projecting the embeddings into a higher-dimensional space, applying a non-linear activation function, and projecting the result back into the original dimension. This process enhances the model's ability to extract meaningful features and relationships from the more refined representation of the embedding coming from the preceding attention head.

In the context of time series predictions and physical systems, T-NN utilize "masking" to prevent future embeddings $\vec{\mathbf{E}}$ from influencing the computation of attention scores for preceding ones. This masking occurs within the attention pattern computation, where elements corresponding to future values are set to $-\infty$ before calculating attention scores. By masking future embeddings, the model ensures that each embedding attends only to past or current embeddings, facilitating the capture of temporal dependencies sequentially without being influenced by future information.

The advantage of the T-NN architecture is that it achieves parallelization through its self-attention mechanism and feed-forward networks, which is multiplied by the number of attention heads h in the network. This multi-head attention allows the model to learn different representations and is given by:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(head_1, ..., head_h)\boldsymbol{W}^0, \tag{4.9a}$$

$$\text{where,} \quad \text{head}_i = \text{Attention}(Q\boldsymbol{W}_i^Q, K\boldsymbol{W}_i^K, V\boldsymbol{W}_i^V) \tag{4.9b}$$

with $\boldsymbol{W}_i^Q$, $\boldsymbol{W}_i^K$, $\boldsymbol{W}_i^V$ (query, key, and value weight matrices for each attention head i), Q, K, and V (query, key, and value matrices for each attention head i), and $\boldsymbol{W}^0$ (output matrix) [38].

The self-attention mechanism inside each head allows embeddings of the input sequence to attend to each other in a concurrent fashion. This enables parallel computation across all of the embeddings facilitated by the matrix vector multiplications. Additionally, the architecture inherently supports making predictions for each embedding in parallel up to $\mathcal{C}$. This is better captured in Figure 4.3. By allowing each embedding to attend to all others and incorporating multiple layers of self-attention and feed-forward networks, the Transformer model generates predictions for every token simultaneously [82].

The feed-forward networks within each layer operate independently on the final embeddings coming from the attention heads. This parallelization allows efficient use of computational resources, resulting in faster training and inference times while capturing complex system dynamics across the input sequence more effectively [82].

Finally, the adapted architecture proposed by Park et al. [39] does not include the explicit use of positional encoding. The literature has shown that transformers can

still produce accurate predictions when used with casual masking for time series prediction [83], [84].

4.4 Constrained and unconstrained optimization

Optimization is the process of finding an optimal solution from a set of feasible solutions subject to an objective function $f(x)$ and its decision variables. In most cases, solving optimization problems analytically is impractical. Therefore, iterative algorithms are used in practice. These start with an initial guess and generate further iterations to improve the cost function until a stopping criterion is met. For iterative methods, there are two main types: direct methods, which rely solely on function evaluations, and indirect methods, which use gradient information [85], [86].

Optimization problems can be either constrained or unconstrained. The objective function is expressed mathematically as the goal that the optimizer should achieve, typically minimization written as minimize$f(x)$. The decision variable is then optimized to reach the goal of the objective function (e.g., x in $f(x)$) [87]. The unconstrained optimization problem is therefore written as:

$$\underset{x}{\text{minimize}} \quad f(x) \tag{4.10}$$

Depending on the problem formulation, there are different algorithms to solve a particular optimization problem, including linear and non-linear problems, integer and continuous variables, among others [85].

Unconstrained optimization problems seek to minimize an objective function without any constraints. Common methods for solving these problems include gradient descent (1^{st} order derivative), Newton's method (2^{nd} order derivative approximation), and Quasi-Newton methods such as Broyden–Fletcher–Goldfarb–Shanno (BFGS) [88].

Constrained optimization problems involve minimizing the objective function subject to constraints. These constraints can be equality or inequality constraints, expressed as $h(x) = 0$ and $g(x) \leq 0$ respectively [87]. Equation (4.10) is then rewritten as:

$$\underset{u}{\text{minimize}} \quad f(x) \tag{4.11a}$$

$$\text{subject to:} \quad h_j(x) = 0, \quad j = 1, \ldots, p \tag{4.11b}$$

$$g_i(x) \leq 0, \quad i = 1, \ldots, m \tag{4.11c}$$

Methods to solve these problems include Linear Programming (LP), Quadratic Programming (QP), and Non-linear Programming (NLP) [85], [89]. The solution $\boldsymbol{u}$ of the problem is called feasible if it satisfies all constraints and conditions (Equations 4.11b

and 4.11c). For some problems, there might exist more than one feasible solution. These solutions then form the feasible set of inputs $\mathbb{U} = \{\boldsymbol{u} \in \mathbb{R}^m\}$. Moreover, the solution is called optimal $\boldsymbol{u}^*$ if[87]:

$$\begin{aligned} & f(x^*) \leq f(x) \quad \forall x \in \mathbb{X} \\ \text{such that:} \quad & g(x) \leq 0, \quad h(x) = 0. \end{aligned} \tag{4.12}$$

4.4.1 Barrier functions

While iterative algorithms may handle equality constraints as a set of algebraic equations inside the optimization loop, the analytical solution of inequality constraints requires a different approach. To solve them, constrained optimizers are extended to handle these inequalities by methods such as interior point (trust-region), barrier functions, active set, linear approximation, or projection-based methods, to name a few [90].

Barrier functions, for example, add a penalty to the objective function for violating constraints. They lead the gradient of the solution away from the constraint, transforming a constrained problem into an equality-constrained one or enabling an unconstrained solver to deal with constraints. Thus, the constrained optimization problem shown in Equation (4.11) can be reformulated as:

$$\underset{x}{\text{minimize}} \quad f(x) + \sum_{i=1}^{m} \phi(g_i(\boldsymbol{x})) \tag{4.13a}$$

$$\text{subject to:} \quad h_j(x) = 0, \quad j = 1, \ldots, p \tag{4.13b}$$

$$g_i(x) \leq 0, \quad i = 1, \ldots, m \tag{4.13c}$$

where $\phi(\cdot)$ is a barrier function for $g(x)$ [86].

Barrier functions approximate an indicator function $I\_(v)$. This is a piecewise function of the form:

$$I\_(v) = \begin{cases} 0 & \text{if } v \leq 0, \\ +\infty & \text{otherwise.} \end{cases} \tag{4.14}$$

The indicator function introduces a large penalty (approaching infinity) as v approaches zero from the negative side, thus enforcing the constraints indirectly by making infeasible regions where $I\_(v) \geq 0$ have an increasing cost relative to the objective func-

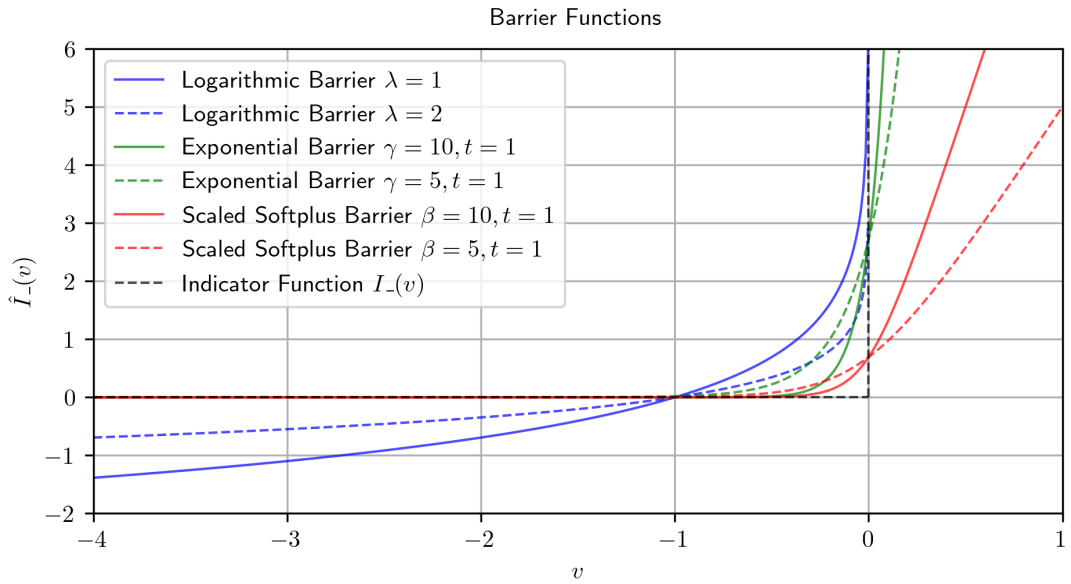


Figure 4.4: Examples of logarithmic, exponential, and scaled Softplus barrier functions.

tion[87]. For numerical stability reasons, a continuous function approximation is preferred over a piecewise function [85]. Examples of barrier functions are:

$$\text{Logarithmic barrier:} \quad \hat{I}_-(v) = -(1/\lambda)\log(-v) \tag{4.15a}$$

$$\text{Exponential barrier:} \quad \hat{I}_-(v) = e^{(T+v\gamma)} \tag{4.15b}$$

$$\text{Scaled Softplus barrier:} \quad \hat{I}_-(v) = (1/T)\frac{1}{\beta/T}\log(1+\exp(\frac{\beta v}{T})) \tag{4.15c}$$

with parameters T (function temperature), and λ and γ (approximation quality of $\hat{I}_-(v)$). A depiction of the barrier functions is shown in Figure 4.4.

The parameters of the barrier functions must be carefully tuned to ensure the values remain smaller relative to the objective function. This helps prevent the solution from deviating from the optimum [86], [90]. In summary, both unconstrained and constrained optimization techniques, including the use of barrier functions to handle constraints, constitute the basis for implementing advanced control strategies such as MPC, which uses optimization to predict and optimize future system dynamics [91].

4.5 Model predictive control

Model Predictive Control (MPC) is an optimal control approach that determines an input sequence $u(t)$ to minimize an objective function $J(\cdot)$ while satisfying constraints, formulated as an Optimal Control Problem (OCP) [91]. The benefits of this controller in-

clude enhanced control performance and efficiency through constraints handling, due to its predictive capability given by the described system model [91]. Based on optimization, MPC can handle Multiple Input Multiple Output (MIMO) and non-linear systems [92]. State-of-the-art formulations also address robustness by managing model uncertainties [93].

Considering the nonlinear discrete-time system in Equation (4.1), we define the MPC problem with prediction horizon p as:

$$
\begin{aligned}
\underset{\boldsymbol{u}_{[k+1:k+p]}}{\text{minimize}} \quad & J(\boldsymbol{x}(k), \boldsymbol{u}(k)) && \text{(4.16a)}\\
\text{subject to:} \quad & \boldsymbol{x}(k+1) = f(\boldsymbol{x}(k), \boldsymbol{u}(k)), && \text{(4.16b)}\\
& \boldsymbol{x}(0) = \boldsymbol{x}_0, && \text{(4.16c)}\\
& h_k(\boldsymbol{x}(k), \boldsymbol{u}(k)) = 0, && \text{(4.16d)}\\
& g_k(\boldsymbol{x}(k), \boldsymbol{u}(k)) \leq 0, && \text{(4.16e)}\\
& \forall k \in \{k, \ldots, p-1\},\\
& \boldsymbol{u}(k) \in \mathbb{U}, \quad \boldsymbol{x}(k) \in \mathbb{X}
\end{aligned}
$$

with $\boldsymbol{x}(k)$ and $\boldsymbol{u}(k)$ (states and control inputs at time k), $J(\boldsymbol{x}(k), \boldsymbol{u}(k))$ (cost function), $\boldsymbol{x}(k+1)$ (next state), $f(\boldsymbol{x}(k), \boldsymbol{u}(k))$ (system's model), $\boldsymbol{x}_0$ (initial state), $h_k(\cdot)$ (equality constraints), $g_k(\cdot)$ (inequality constraints). For this thesis, since the data available are discrete observed values, the discrete time form k is used.

The problem is solved at each discrete time-step k iteratively by an optimizer, given the observed (or estimated) states $\boldsymbol{x}$, the initial state value $\boldsymbol{x}_0$, and the system model $f(\boldsymbol{x}(k), \boldsymbol{u}(k))$ [94]. The solution is an optimal sequence of input variables $\boldsymbol{u}^*_{[k:k+p-1]}$ given the predicted system states $\boldsymbol{x}^*_{[k:k+p]}$ and the optimal inputs.

In Figure 4.5, a representation of the MPC algorithm is depicted. At time k=0, with information on past measured or estimated states up to look-back window w, and current states (initial state x_0), the optimal sequence of input variables $\boldsymbol{u}^*$ is iteratively calculated in a closed-loop up to prediction horizon p, where the predicted system states $\boldsymbol{x}^*$ show the behavior of the system as a function of the optimal sequence [91].

For the closed-loop control application, the first element of the sequence of inputs $\boldsymbol{u}^*$ is applied to the system or plant. In summary, an MPC controller calculates the optimal control input sequence $\boldsymbol{u}$ considering (or predicting) how the system will behave in the future using the system model and the states $\boldsymbol{x}$. Such a closed loop of a MPC controller is depicted in Figure 4.6, and its implementation is described in Algorithm 1.

The objective of the MPC is to minimize the cost function $J(\boldsymbol{x}(k), \boldsymbol{u}(k))$ given the optimization variables and the future states up to the prediction horizon p of the system given the model. The formulation of this function is done as a quadratic problem

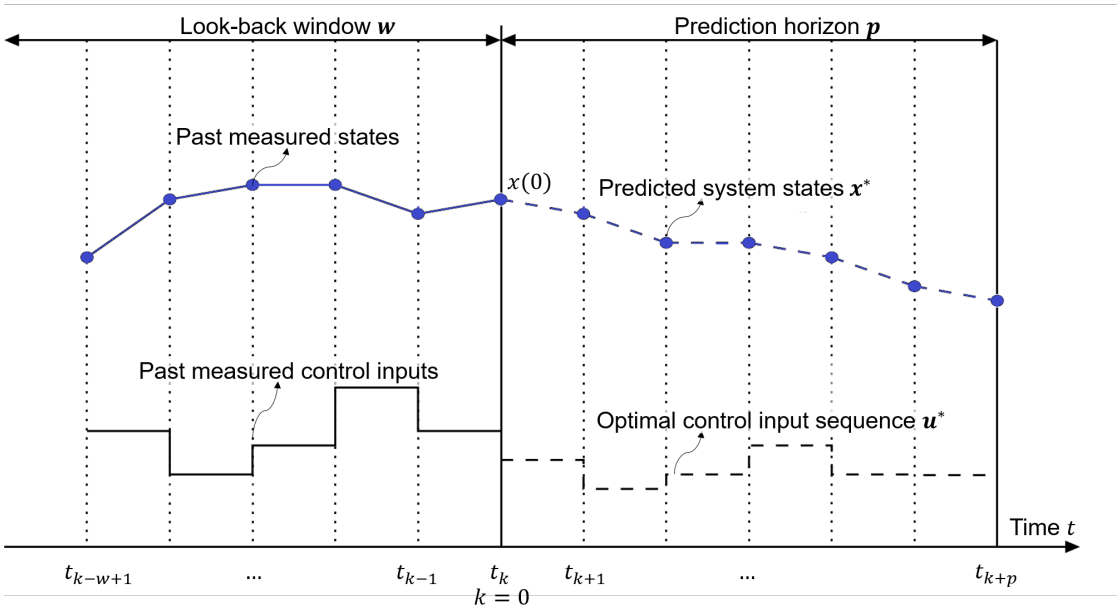


Figure 4.5: MPC system control algorithm example. Adaptation. Grüne [93].

Algorithm 1 MPC Closed-Loop Algorithm

Initialize $k \leftarrow 0$, system state $\boldsymbol{x}(0)$, and desired output y_{ref}.
while control loop is active **do**
  Provide $y_{\text{ref}}(k)$ to MPC controller.
  Predict future states $\boldsymbol{x}(k+1)$ using plant model and current system state $\boldsymbol{x}(0)$.
  Initialize optimizer iteration $i \leftarrow 0$.
  while not converged & $k < p - 1$ **do**
    Iteration i: Calculate candidate control input sequence $\boldsymbol{u}_i(k)$.
    Evaluate cost function $J(\boldsymbol{x}, \boldsymbol{u}_i)$.
    Evaluate constraints.
    Update control input sequence $\boldsymbol{u}_i(k)$.
    $i \leftarrow i + 1$
  end while
  Set $\boldsymbol{u}^*(k)$ as the optimal control input sequence from the optimizer.
  Apply the first element of $\boldsymbol{u}^*(k)$ to the real plant.
  Measure output $y(k)$ from the real plant.
  Feed $y(k)$ back to the MPC controller.
  Update plant model with the new state $\boldsymbol{x}(k+1)$.
  $k \leftarrow k + 1$
end while

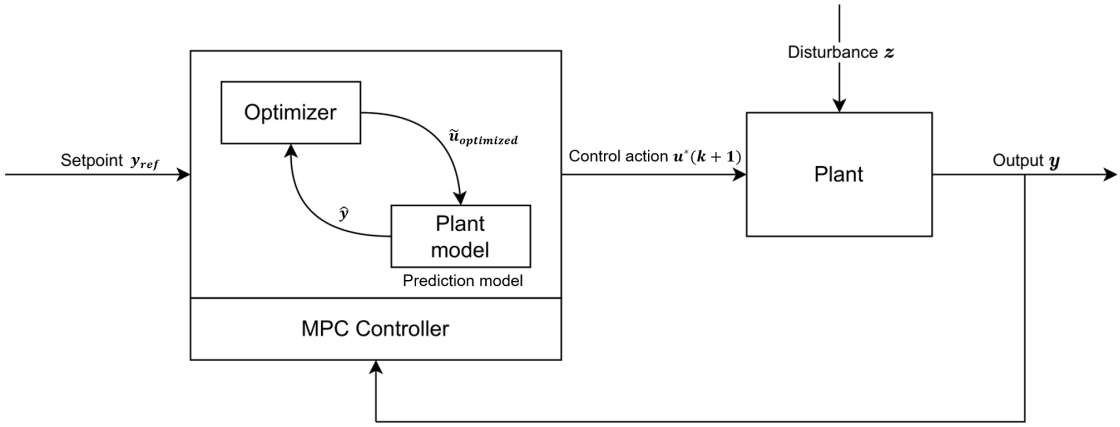


Figure 4.6: Control closed-loop with a MPC. Adaptation. Schwenzer et al. [95].

that describes a convex function and can consider different effects such as regulation, tracking, minimum time, and economic [87]. For this work, the cost function described in Equation (4.16a) considers the reference tracking effect, which takes the form:

$$J(\boldsymbol{x}, \boldsymbol{u}, \Delta\boldsymbol{u}) = \sum_{k=0}^{p-1} (l(\boldsymbol{x}(k), \boldsymbol{u}(k)) + r(\Delta\boldsymbol{u}(k))), \tag{4.17a}$$

$$\text{where,} \quad l(\boldsymbol{x}(k), \boldsymbol{u}(k)) = f'(\boldsymbol{x}(k), \boldsymbol{u}(k))^\top \boldsymbol{Q} f'(\boldsymbol{x}(k), \boldsymbol{u}(k)), \tag{4.17b}$$

$$r(\Delta\boldsymbol{u}(k)) = \Delta\boldsymbol{u}(k)^\top \boldsymbol{R} \Delta\boldsymbol{u}(k), \tag{4.17c}$$

$$\Delta\boldsymbol{u}(k) = \boldsymbol{u}(k) - \boldsymbol{u}(k-1), \tag{4.17d}$$

$$f'(\boldsymbol{x}(k), \boldsymbol{u}(k)) = \boldsymbol{x}(k+1) - \boldsymbol{x}_{ref}(k+1) \tag{4.17e}$$

with $\boldsymbol{Q}$ (tracking gain), and $\boldsymbol{R}$ (control effort gain). Tracking effect is divided into two parts: the stage cost $l(\boldsymbol{x}(k), \boldsymbol{u}(k))$ and the control effort cost $r(\Delta\boldsymbol{u}(k))$. Since the OCP is formulated as a minimization problem, the goal of the former is to penalize the distance to a reference or desired value (Equation (4.17b)), while the latter penalizes the rate of change in the control inputs (Equation (4.17c)). The convex quadratic formulation of the reference tracking MPC is then redefined as follows:

$$\underset{\boldsymbol{u}_{[k+1:k+p]}}{\text{minimize}} \quad J(\boldsymbol{x}, \boldsymbol{u}, \Delta\boldsymbol{u}) \tag{4.18a}$$

$$\text{subject to:} \quad 4.16b, 4.16c, 4.16d, 4.16e$$

The use of equality and inequality constraints (Equation (4.16d) and Equation (4.16e)) aims to consider operation limits or safety conditions during the state's evolution of the system. They can represent input constraints $u(\cdot) \in \mathbb{U}$, state constraints $\boldsymbol{x}(\cdot) \in \mathbb{X}$, boundary conditions, and system dynamics.

Input constraints capture limitations in actuators or controlled variables, such as a motor's angular velocity, valve opening, among others. The set $\mathbb{U}$ is usually described as a compact set since the parameters of the actuators are known, e.g., the minimum and maximum angular velocity of a compressor or the rate of change of a motor. As this set is considered to be closed and bounded, the input constraints usually take the form of inequality constraints, described as box constraints for the inputs as $\mathbb{U} = \{\boldsymbol{u} \in \mathbb{R}^m \mid \boldsymbol{u}_{\text{lb}} \leq \boldsymbol{u}(k) \leq \boldsymbol{u}_{\text{ub}}\}$, or for the inputs rate of change $\Delta\mathbb{U} = \{\Delta\boldsymbol{u} \in \mathbb{R}^m \mid \Delta\boldsymbol{u}_{\text{lb}} \leq \boldsymbol{u}(k+1) - \boldsymbol{u}(k) \leq \Delta\boldsymbol{u}_{\text{ub}}\}$, where $\boldsymbol{u}_{\text{ub}}, \Delta\boldsymbol{u}_{\text{ub}}$ and $\boldsymbol{u}_{\text{lb}}, \Delta\boldsymbol{u}_{\text{lb}}$ are the upper and lower bounds respectively [87].

State constraints represent physical limitations of the system, safety restrictions, or quality parameters, among others [86]. The first refer to the physical properties or capabilities of the system, while the latter ensure that the system operates within safe and acceptable boundaries. An example of this can be temperature limits or rates of change in temperature. The set of state constraints $\mathbb{X}$ is considered to form a single, uninterrupted region in the state space, with no disjoints in the feasible set $\mathbb{X} = \{\boldsymbol{x} \in \mathbb{R}^m \mid \boldsymbol{x}_{\text{lb}} \leq \boldsymbol{x}(k) \leq \boldsymbol{x}_{\text{ub}}\}$ [87].

Boundary conditions encompass initial conditions and terminal constraints, and system dynamics path and output constraints. For certain formulations, soft constraints can also be used to allow some flexibility in constraint satisfaction in the optimization by introducing penalty terms in the cost function for constraint violations [85]. This is done by adding slack variables ϵ. For input constraints, for example, it is formulated such that $\boldsymbol{u}_{\text{lb}} \leq \boldsymbol{u}(k) - \epsilon \leq \boldsymbol{u}_{\text{ub}}$, where the difference is then penalized and included as a part of the cost function. Therefore, the OCP from Equation (4.17a) can be rewritten as:

$$J(\boldsymbol{x}, \Delta\boldsymbol{u}, \epsilon) = \sum_{k=0}^{p-1} (l(\boldsymbol{x}(k), \boldsymbol{u}(k)) + r(\Delta\boldsymbol{u}(k))) + \rho\|\epsilon\|^2 \tag{4.19}$$

where ϵ represents the slack variables and ρ is a penalty weight.

For state-space, ODE, and some NN system models, the predictions are a single future prediction, also known as One Step-Ahead (OSA) prediction. To get the full system's path $\boldsymbol{x}^*$, a recursive iteration needs to be implemented. This may lead to two disadvantages: more computational time to solve the optimization problem is needed, which might be relevant in real-time applications [91]. Additionally, simplified models, direct multiple shooting approaches or those that do not account for uncertainties are prone to have deviated predictions from the true plant dynamics. These approaches also do not consider past data other than the current state and the vanishing/exploding gradients problem.

In comparison, there exist some data-based approaches like T-NN whose architecture enables Multi Step-Ahead (MSA) predictions up to the prediction horizon p as

part of the model itself [38], [78], and include information further into the past up to the look-back window w. With this, MPC can be enhanced and its relevance as a modern control method improved. Due to its disturbance rejection enabled by its model predicted behavior and the handling of constraints, it is a suitable tool for controlling the STJ receiver.

4.6 Uncertainty quantification

While data-based approaches offer benefits over first-principles models, uncertainty in the models often accompany the resulting black-box models. Sources of uncertainty include parametric variations, unmodeled dynamics or simplifications, disturbances, measurement and communication noise, linearization errors, and time - varying parameters, among others [96]. Addressing this uncertainty is crucial for improving control [97].

Uncertainty Quantification (UQ) involves identifying, characterizing, and managing uncertainty in mathematical models and their posterior predictions as regressors [98]. Common approaches to address uncertainty include probabilistic methods, which model uncertainty using probability distributions. Examples include Monte Carlo simulations, Bayesian inference, and Gaussian Processes (GPs) [98].

This chapter demonstrates how GPs can incorporate UQ methods into NN, which usually do not include probabilistic data in their predictions [45], [49]. The main objective is to provide operators with valuable probabilistic insights regarding the prediction accuracy of the regression model, thereby improving reliability and aiding decision-making. This implementation examines both the practicality and advantages of utilizing GPs in this context. Although a comprehensive overview of GP and UQ techniques is provided for context, the focus remains on practical application rather than detailed theoretical exploration. The goal is to demonstrate the motivation and encourage rigorous theoretical analysis for future work.

4.6.1 Gaussian processes

Among approaches for uncertainty quantification, GP regressors are used in state-of-the-art control techniques due to their flexibility thanks to the non-parametric solution leveraged by kernels and theoretical probabilistic modeling [45], [48], [54].

Probabilistic regression using GPs extended from the linear regression problem and considers a normally distributed additive noise. The linear approximation y_k in Equation 4.1b is rewritten in compact form as [98], [99]

$$y = \boldsymbol{x}^T \boldsymbol{w} + \epsilon \tag{4.20a}$$

where $\boldsymbol{w} \in \mathbb{R}^i$ is the vector of parameters and $\epsilon \sim \mathcal{N}(\mu, \Sigma)$. In this case, the mean value μ of the normal distribution $\mathcal{N}(\cdot)$ is assumed to be 0, and that the distribution has a covariance $\Sigma \in \mathbb{R}^{i \times i}$.

In the literature, there are two methods to approximate nonlinear functions as mappings between the inputs $\boldsymbol{x}$ and outputs $\boldsymbol{y}$. The first is using a set of basis functions with nonlinear features $\phi(\boldsymbol{x}) = [\phi_1(x), \ldots, \phi_i(x)]^T$, which are linearly combined to approximate the function [45]. The second uses kernel methods [100], which utilize similarity functions to find the modeling relations based on observed data.

Kernel functions that describe the similarity of the observed data are defined as $k(\boldsymbol{x}, \boldsymbol{x}')$ [100]. A commonly used kernel function in the literature is the Radial Basis Function (RBF) kernel [48], [101]. The advantage of the kernel method is that it avoids the calculation of basis functions and instead relies on training data to estimate the parameters [45].

To solve the regression in a probabilistic fashion, the uncertainty in the prediction of $\boldsymbol{y}$ for every new observed value $\boldsymbol{x} \in \tilde{\mathbb{Z}}$ is modeled as the predictive distribution $P(\boldsymbol{y} \mid \boldsymbol{z})$. The likelihood of observing the model output y subject to the given parameters $\boldsymbol{w}$ is given by:

$$P(y \mid \boldsymbol{w}) = \mathcal{N}(y; \tilde{\mathbb{Z}}\boldsymbol{w}, \Sigma(w)) \quad \text{(Likelihood)} \tag{4.21}$$

The approach also assumes a previous known information on the parameters that follow a normal distribution such that:

$$P(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}; 0, \Sigma(w)) \quad \text{(Prior)} \tag{4.22}$$

where distribution is defined as a normal distribution $\mathcal{N}(\mu, \Sigma)$ with mean $\mu = 0$ and covariance Σ [98], [99].

Utilizing Bayes' theorem, the probability of the parameters of the mapping function given the observed data, called the posterior, is calculated as:

$$P(\boldsymbol{w} \mid y) = \frac{P(y \mid \boldsymbol{w}) \times P(\boldsymbol{w})}{P(y)} = \mathcal{N}(w; E(w), \Sigma(w)) \tag{4.23a}$$

$$\text{Posterior} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Marginal likelihood}} \tag{4.23b}$$

where $P(\boldsymbol{w})$ is the prior information on the distribution of the parameters, $P(y \mid \boldsymbol{w})$ is the likelihood of the observed data given the parameters, and $P(y)$ is the probability of observing the data (marginal likelihood) [45], [99].

Defined in the literature as "a collection of random variables" [45], a GP is a distribution of multivariate normal distribution functions $f(w; \boldsymbol{z})$ that describe observed points $\boldsymbol{z}$ [100]. Formally, a GP is denoted as [97]:

$$f(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}'))$$

with $m(\boldsymbol{x})$ (mean), and $k(\boldsymbol{x}, \boldsymbol{x}')$ (covariance) [102]. The solution to the problem is solved by minimization of the negative log likelihood (Maximum Likelihood Estimation (MLE)) or by the maximization of the posterior subject to the parameters (Maximum A Posteriori estimation (MAP)). The solution can be formulated for both the basis functions and kernel formulations, based on the assumption of a normal distribution given as:

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu_x)^2}{2\sigma^2}} \tag{4.24}$$

This results in a predicted y with a mean μ and covariance Σ as a function of the parameters w of the basis functions f or hyperparameters of the kernels.

Assuming that the parameters describing the noise ϵ are calculated, the model can be used to make predictions based on the inputs. This is described as $\hat{\boldsymbol{y}} = \hat{\boldsymbol{x}}^T \boldsymbol{w}$, and the probability of the predicted value as [99]:

$$P(\hat{\boldsymbol{y}} \mid \boldsymbol{y} = \boldsymbol{z}) = \mathcal{N}(\hat{\boldsymbol{y}}; \hat{\boldsymbol{x}}^T \mathbb{E}(w), \hat{\boldsymbol{x}}^T \Sigma(w) \boldsymbol{x}) \tag{4.25}$$

Chapter 5

Artificial Neural Network Dynamic Models

Looking to research the applicability of data-based models with modern control techniques, this work explores the use of a Transformer Neural Network (T-NN) with Model Predictive Control (MPC). Our NN is based on the model presented in the paper by Park et al. [39]. Therefore, this chapter presents the methods and results of using T-NNs applied to two case studies. The scope of this chapter includes the structure and training processes of the NNs used, demonstrating their application in MPC scenarios.

5.1 Transformer neural network architecture and training data stucture

Figure 5.1 show the Transformer Neural Network (T-NN) architecture used in this thesis. The implementation is based on the approach presented by Park et al. [39]. The network consists of an encoder-decoder structure. Inside each encoder, input is encoded into embeddings $\vec{\mathbf{E}}_i$ and fed into a given number of transformer blocks added sequentially, each composed of a "multi-head attention" layer, a residual connection, and a pass through a feed-forward layer.

The dataset $\mathbb{D}$ used for training was adapted to have a specific structure to be used in the T-NN time-series model. For this thesis, the Multi Step-Ahead (MSA) prediction capability of the architecture is explored. The input $\mathbb{X}$, containing pairs of states $x_k \in \mathbb{R}^x$ and inputs $u_k \in \mathbb{R}^u$ with $k = \{0, 1, \ldots, n\}$, takes the form of a three-dimensional array called a tensor $\mathbf{X}$. The set $\mathbb{Y}$, with outputs $y_k \in \mathbb{R}^y$, is also added to the tensor $\mathbf{X}$. In the context of NNs, the first dimension of the tensor represents the batch, which is the number of samples in the tensor. This will be used during training and evaluation.

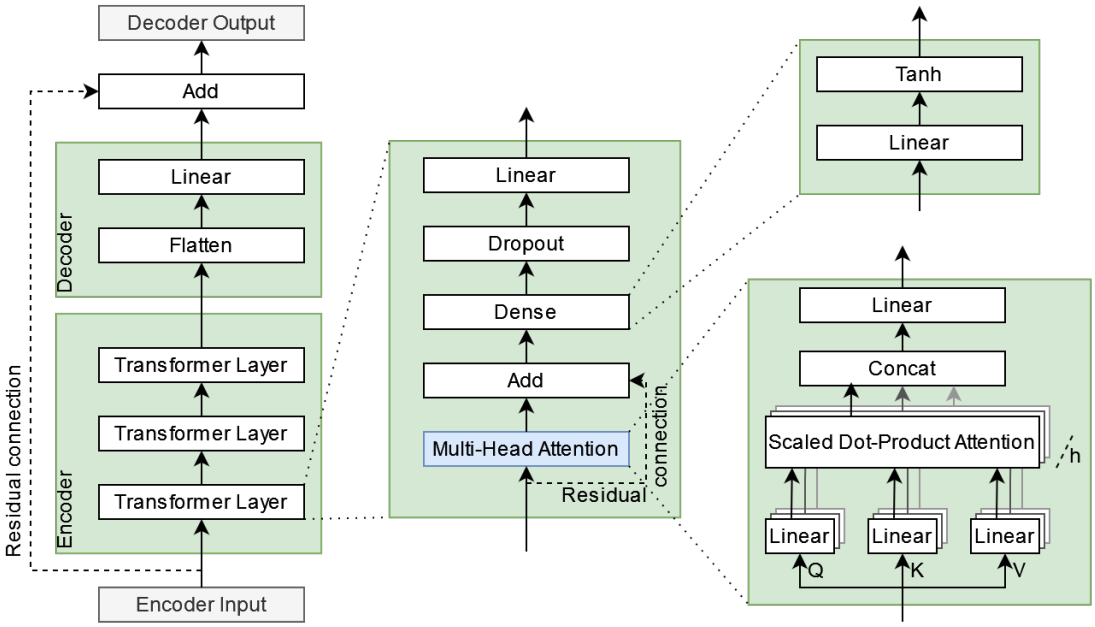


Figure 5.1: Transformer Neural Network (T-NN) architecture.

The latter two dimensions of the tensor correspond to the length of the time window and the number of embedding dimensions d_{e}. The time window for which the network will be used is also known as the context size $\mathcal{C}$. This time window is considered to be constant, and its length is equal to the length of the look-back window w plus the prediction horizon p. The size of the embedding dimension d_{e} is equal to the number of variables in the system as $\mathbb{R}^{(y,u,x)}$. For simplicity of representation, the latter two dimensions of the tensor will be used for defining the equations and examples.

Furthermore, the attention mechanism present in the T-NN architecture has data requirements in the tensor $\mathbf{X}$. This is due to the T-NN using contextual information from both future and past embeddings. For time-series predictions, this behavior must be modified. When used as a regressor, the model must prevent future values from influencing past ones. This is called masking or causal masking. As presented in Section 4.3, one approach to achieve this effect is by setting the corresponding values inside the attention pattern to $-\infty$. Another approach, as presented by Park et al. [39], is to make all the output values y for future time steps equal to y_k. As an analogy, this serves as labels l for the NN to predict those values given the inputs, where

Table 5.1: Data structure for T-NN model. Note repetition of y_k in y' from time $t = k$ to $t = k + p$. u is the optimization variable.

			Context window at time t					
Description	**Size**		$k-w+1$	$\dots$	k	$k+1$	$\dots$	$k+p$
NN output **Y**	$p \times l$	y				y_{k+1}	$\cdots$	y_{k+p}
NN input **X**	$(w+p) \times d_e$	y'	y_{k-w+1}	$\cdots$	y_k	y_k	$\cdots$	y_k
		u	u_{k-w+1}	$\cdots$	u_k	u_{k+1}	$\cdots$	u_{k+p}
		x	x_{k-w+1}	$\cdots$	x_k	x_{k+1}	$\cdots$	x_{k+p}

$\dim(l) = \dim(y)$. With this, the mathematical equations for the data structure are given as:

$$\mathbf{Y} = (y_{k+1}, \dots, y_{k+p}) \in \mathbb{R}^{p \times l}, \tag{5.1a}$$

$$\mathbf{X} = \begin{pmatrix} (y_{k-w}, \dots, y'_{k+1}, \dots, y'_{k+p}) \\ (u_{k-w}, \dots, u_{k+p}), \\ (x_{k-w}, \dots, x_{k+p}) \end{pmatrix} \in \mathbb{R}^{(w+p) \times d_e}, \tag{5.1b}$$

$$\text{where:} \quad y'_{k+i} = y_k, \quad i = 1, \dots, p \tag{5.1c}$$

Table 5.1 describes the structure of both the input **X** and output **Y** tensors for the T-NN model, detailing their respective components and dimensions. The NN output tensor **Y** consists of predicted values for future time steps $(y_{k+1}, \dots, y_{k+p})$ with a size of $p \times l$. The input **X** contains both the prior and posterior context from $k = 0$, represented by sequences of states $(x_{k-w+1}, \dots, x_{k+p})$ and control inputs $(u_{k-w+1}, \dots, u_{k+p})$ over a window size $(w + p)$. Moreover, **X** includes the current output value $y_{k-w}, \dots, y'_{k+1}, \dots, y'_{k+p}$ for future steps. This structure enables the model to use historical data while preventing future information leakage and influencing past values.

For the following sections and during the evaluation of this work, the parameters of the NN are shown in Table 5.2. Moreover, the methodology presented in this chapter is based on that presented by Park et al. [39]. The dataset and source code are available at https://github.com/BYU-PRISM/Transformer_MPC.

Taking advantage of the T-NN structure and its MSA prediction capabilities, it is possible to include past measurements to try to better predict the future states. Known as the look-back window w, this enables the model to include past dynamics to better model future ones up to the prediction horizon p.

Table 5.2: Options for Neural Network Model Configuration

Name	Type	Description
Embedding dimension d_e	int	Features/variables in input **X**
Feed-forward dimension	int	Feed-forward NN dimension
w	int	Look-back window
p	int	Prediction horizon
Attention heads	int	Number of heads in the multihead attention
Transformer layers	int	Number of sequential transformer layers
Dropout %	float	Dropout percentage in evaluation mode
Output dimension l	int	Features/variables to be fit/predicted. Output **Y**
Optimizer	Literal['Adam', 'AdamW']	Optimizer used for training
Learning rate	float	Optimization learning rate
Residual connection	bool	Enables last layer residual connection
Persistence model	bool	Enables persistence model. Serves as baseline

While the use of masking for the attention head is important for the predicted variables, it can be reformulated to achieve predictive behavior in the states. Future values can be added in states variables as predictions of possible future changes or disturbances, hence enabling the model to better predict how the system will behave, thereby leveraging MPC.

To program and train NNs, Python 3.11 [103] is used in this work. While Park et al.[39] used TensorFlow [104] for the NN, our approach was based on PyTorch 2.11 [105] and PyTorch Lightning 2.1.3 [106]. TensorBoard 2.16.2 was used as logger.

Table 5.3: First order plus dead-time neural network parameter configuration

Name	Value	Name	Value
Embedding dimension d_e	2	Dropout	20%
w	5	Output dimension l	1
p	10	Optimizer	Adam
Attention heads	2	Learning rate	1e-4
Transformer layers	3		
Trainable parameters		1888	

5.2 First order plus dead-time model

To explore the use of T-NN as dynamic models, the dataset from the First-Order Plus Dead-Time (FOPDT) model described in Park et al. [39] is used. Its dynamics are mathematically expressed as:

$$\dot{y}(t) = (1/\tau_p)(-y(t) + K_p u(t - \theta_p)) \tag{5.2}$$

with $K_p = 1$ (process gain), $\tau_p = 2$ (time constant), and $\theta_p = 0$ (dead time).

In an attempt to reproduce the NN architecture presented by Park et al. [39], the implementation in PyTorch was adapted to have a number of trainable parameters close to that reported by the authors of the paper. In this work, the number of trainable parameters was 1888 for the FOPDT model. In contrast, the TensorFlow implementation shown in the paper has a total of 1758 trainable parameters. Moreover, the last layer residual connection was disabled to replicate the paper's architecture. The parameters of the NN are described in Table 5.3. As described in the paper, the dataset was created using simulation for random inputs u for a length of 1600 data points.

Using the validation set and its Mean Squared Error (MSE) as early stopping criteria, the T-NN achieved on average a test loss value of 1.537×10^{-3} ($\sigma = 5.35 \times 10^{-4}$ with 5 experiments), and 3.639×10^{-3} ($\sigma = 2.65 \times 10^{-4}$ with 5 experiments) with 0% and 10% noise respectively. Noise is added directly to the measurements as a percentage of the standard deviation of the dataset. These values are obtained for the complete dataset.

Figure 5.2 shows an example of the T-NN model of the FOPDT system model, and an ODE simulation for the same input u. The graph shows that the NN model has been capable of capturing the dynamics of the system to a certain extent. In this example,

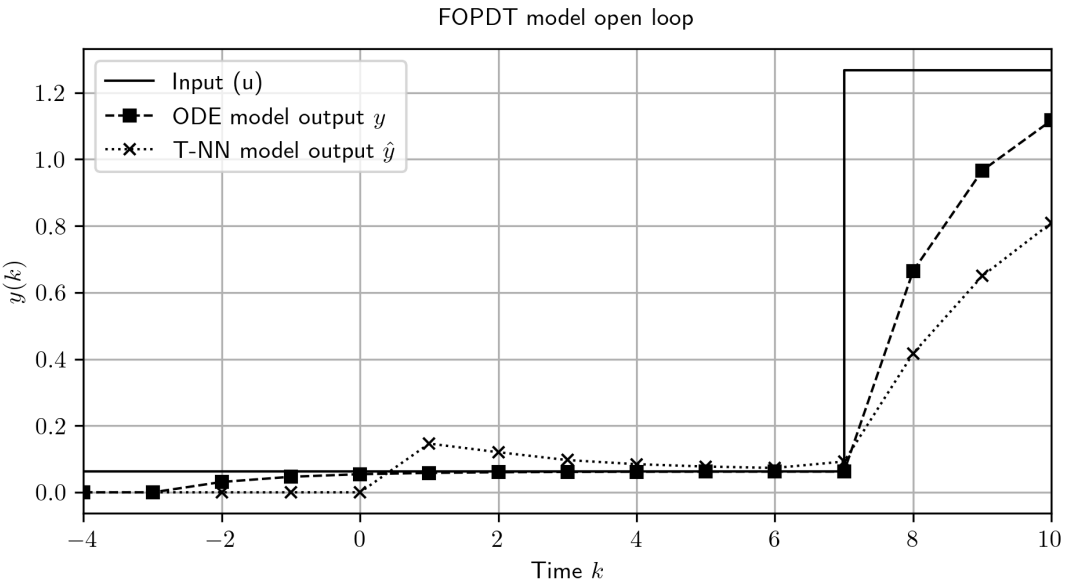


Figure 5.2: Example of FOPDT and T-NN model output y for a given input u

the MSE is equal to 0.027066. Although the accuracy of the model depends on the region of validity at which the model was trained, and the model deviated from the ODE simulation, it shows the capability to model system dynamics. The model accuracy can be further enhanced with extended training, improved parameter selection, and model structure.

5.3 Solar tower Jülich model

The Solar Tower Power Plant Jülich (STJ) NN model was trained using the dataset $\mathcal{D}$ shown in Chapter2. The total length of the dataset consists of 109,821 measured data points, obtained during 6 days of data acquisition in May and June 2023. Using the TensorBoard library, MSE values were logged for $\hat{y}$, as well as for T_{surface} and $T_{\text{hot air}}$ individually.
The parameters of the NN that were kept fixed are described in Table 5.4.

5.3.1 Data pre-processing and filtering

The datasets and the resulting sequences for the input tensor **X** are a function of the look-back windows w and the prediction horizon p. In this approach these values are considered constant and set for the training. Dataset are then divided into training, validation and test subsets in a ratio of 70%-20%-10%. The training sequences were designed to be non-overlapping by creating arrays of validation and test sequence indices. These arrays were extended to include adjacent sequence indices (plus or mi-

Table 5.4: Solar tower Jülich neural network parameter configuration

Name	Value	Name	Value
Embedding dimension d_e	5	Dropout	20%
Output dimension l	2	Optimizer	Adam
Attention heads	5	Transformer layers	3
Trainable parameters		1888	

nus one). If the training sequence index was found in these arrays, it was removed from the training sequence to maintain exclusivity between the datasets. This structure ensured that the subsets were mutually exclusive, eliminating any potential for information leakage that could lead to optimistic performance estimates. While this approach prevents information leakage, it reduces the length of the training sequence.

As loss function, the Mean Squared Error (MSE) for training, validation and test were logged throughout the training process. The validation MSE was used as an early stop criterion. Early stopping was configured in 'minimum' mode with a patience of 30 epochs to avoid overfitting.

In the case of the FOPDT model, the dataset provided assumed equally spaced data points, thanks to the ODE system formulation and the numerical simulation. In the case of the Solar Tower Power Plant Jülich (STJ), the dataset requires pre-processing and filtering due to its real-world nature.
At the facility, the integrated sensor system is programmed to make measurements at a frequency of 1 Hz. For this work, the dataset was reduced to the five variables of interest shown in previously 2.1: receiver's mean surface temperature T_{surface} , receiver's mean surface apparent brightness $I_{\dot{Q}}$, hot air temperature $T_{\text{hot air}}$, cold air temperature $T_{\text{cold air}}$, and air mass flow $\dot{m}_{\text{air}}$.

Due to the data structure needed to train and evaluate the NN, the dataset needs to fulfill the following requirements: window overlap $\geq$ 50%, $\hat{\mathbf{X}} = (\mathbf{X} - \mu(\mathbf{X}))/\sigma(\mathbf{X})$, $\hat{\mathbf{X}} \in \mathbb{R}^{(w+p)\times d_e}$, $NaN \notin \hat{\mathbf{X}}$, set window overlapping. The first condition requires a sequence overlap greater or equal to 50%. The second describes that the input tensor $\mathbf{X}$ is normalized and written as $\hat{\mathbf{X}}$. The second condition requires that from the 109,821 available data points, they need to be divided into sequences of shape $(w + p) \times d_e$. Last, $\hat{\mathbf{X}}$ must have a value at every position. This also means that measurements should be equally distant at 1-second intervals. Algorithm 2 describes the steps for pre-processing and sequencing:

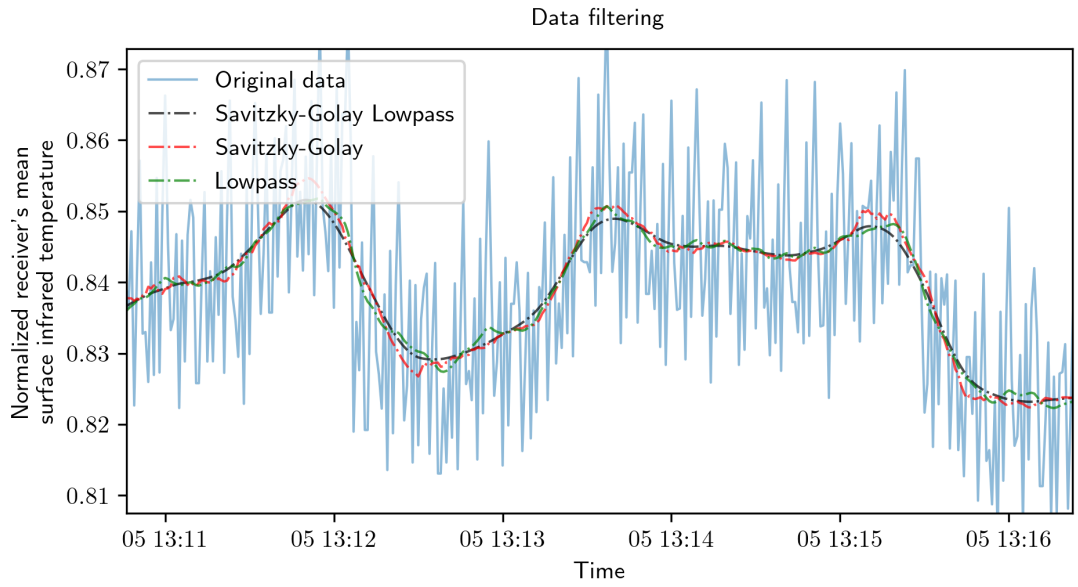


Figure 5.3: T_{surface} noisy data and comparison of different filters. Note step offsets in the measurements attributed to camera's auto-calibration.

Algorithm 2 Data pre-processing and sequencing

1: $\mathbf{X} \leftarrow$ Dataset {Read and load data to tensor}
2: $\hat{\mathbf{X}} \leftarrow \frac{\mathbf{X}-\mu(\mathbf{X})}{\sigma(\mathbf{X})}$ {Normalize}
3: $\hat{\mathbf{X}} \leftarrow$ Filter $\hat{\mathbf{X}}$ {Apply desired filtering}
4: stride $\leftarrow \lfloor (1 - \text{window overlap \%}) \times (w + p) \rfloor$ {Calculate stride}
5: **for** $i \leftarrow 0$, total_length, stride **do**
6: $\quad \hat{\mathbf{X}}_i \leftarrow \hat{\mathbf{X}}[i : i + w + p]$ {Extract sequence}
7: $\quad$ Check $\hat{\mathbf{X}}_i$ continuous at 1 Hz and no NaNs. {Check sequence}
8: **end for**
9: **return** $\hat{\mathbf{X}}_i$

In the dataset, measurement artifacts and noise are observed, particularly for the T_{surface} values obtained from the IR camera. To filter the data, two approaches are used: Direct data filtering and spectrogram analysis.

In the first method, different types of signal filters like exponential smoothing, Savitzky - Golay, low-pass, and combinations were tested. Filters are passed forward and backward to reduce phase delay. Figure 5.3 shows a snapshot of the normalized receiver's mean surface temperature data, where different filters are applied for comparison.

In the second procedure, we calculate the spectrogram of the entire T_{surface} measurements. We sum the intensities across all frequencies and then normalize them. We mark peaks where the summed intensities exceed a threshold, indicating regions with noise due to higher frequency components. Figure 5.4 shows a graph of this anal-

ysis before and after filtering for comparison. In the upper graph, T_{surface} data is plotted. The middle graph shows the spectrogram applied to the filtered data, and the bottom graph shows the added normalized intensity with detected peaks. All three graphs highlight the regions of noise around the peaks. The identified regions are discarded from the dataset and not used during training and inference. For the following, this is referred to as denoising. The first method helps with direct measurement noise, while the second helps identify regions in the data that are not suitable for training.

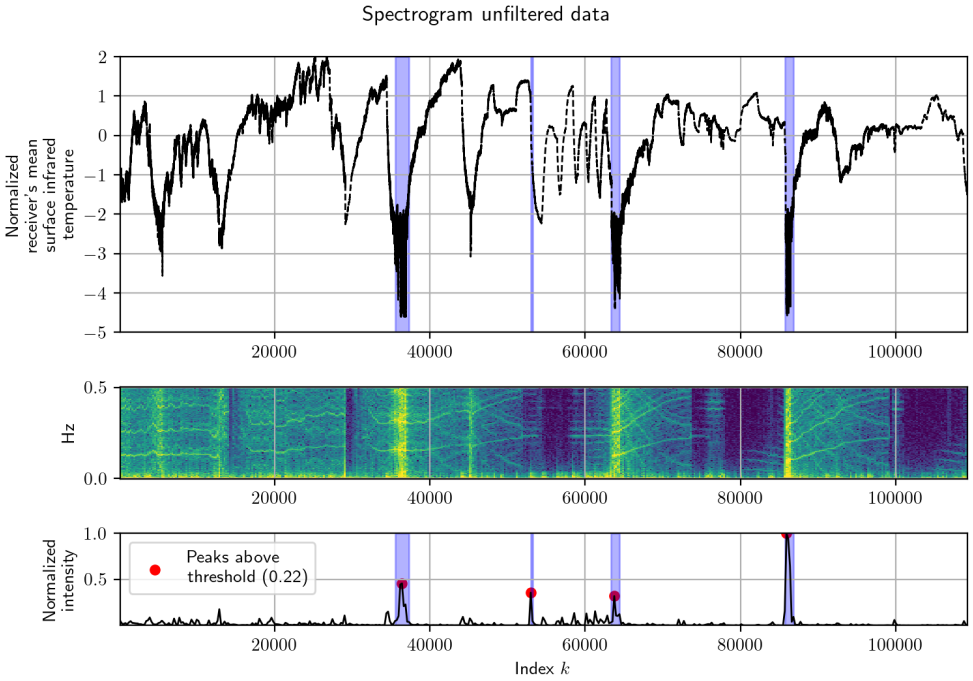
Results for the data filtering in Figure 5.3 show that there has been a marked decrease in the apparent noise in the data while preserving the trend with no observable phase delay for all filters tested. Observing the data points, measurement temporal offsets are observed. These are revealed as offset shifts in the data which are displayed as sudden steps in the measurements. In the graph, four of these offset effects are shown. These were later attributed to the IR camera auto-calibration, and the effect of these were not filtered. These remain open for future work.

For the purpose of training and inference, the low-pass filter is selected for its implementation simplicity. After applying the filter to the T_{surface} dataset, the measurements showed an Signal-to-Noise Ratio (SNR) increase of 23.40 dB and a correlation factor of 99.77%. An SNR above 0 dB implies that the signal power is higher than the noise power, suggesting lower noise influence. Additionally, with a correlation factor closer to 1, it suggests that the filtered signal retains most of the information from the original signal.

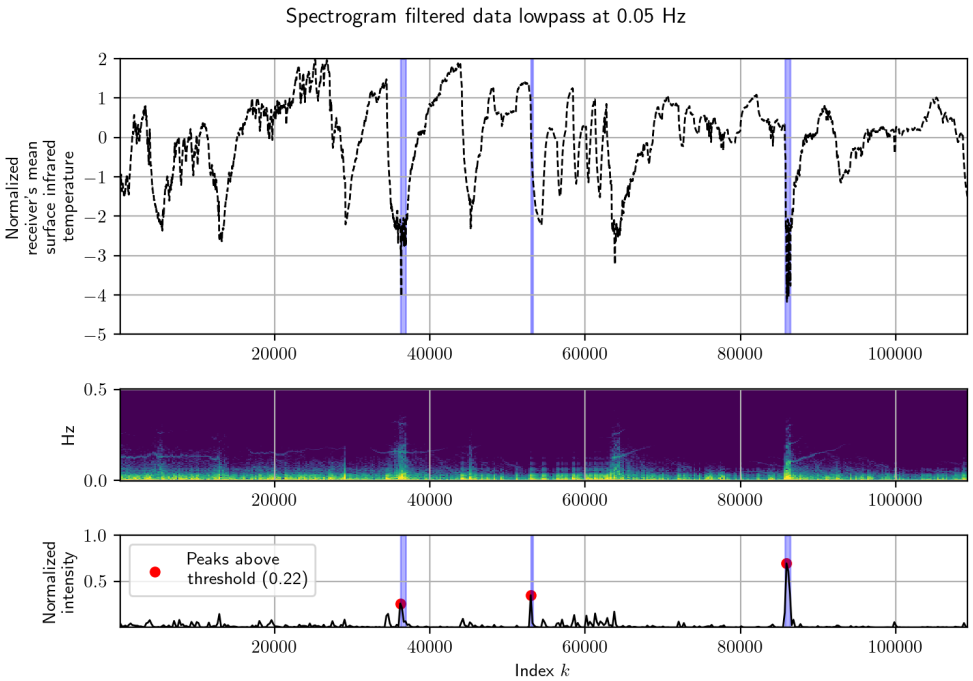
Furthermore, Figure 5.4b shows a sharp reduction in the frequency components in the spectrogram (middle graph). What stands out in this figure is the dominance of frequency components in the lower ends of the normalized values (in the region of -2.5 in the y-axis). These artifacts were later found to be related to the measurement range of the IR camera system. These represent the measured values at ambient temperature before the operation of the solar tower. With a measuring range beginning at 300 °C, measurements below this value are not reliable and prone to high measurement error. With this in mind, the method of finding regions of high noise in the dataset by means of spectrogram analysis offers a systematic approach.

For the training of the T-NN with the STJ dataset, various parameter combinations were tested. Unlike the approach by Park et al. [39] with the FOPDT model, we investigated the use of longer look-back window w values compared to the prediction horizon p. For this study, $p = \{10, 20, \ldots, 60\}$ and $w = \{30, 45, \ldots, 120\}$ were tested.

For reference in the length of available training sequences, using data with filtered measurements and deleted regions with high noise, the total number of dataset sequences results in 427 for $w = 120$ and $p = 300$. After splitting, the number of sequences in the training dataset drops from 298 to 136. Validation sequences therefore are 85 elements long, and 44 for test.



(a) T_{surface} unfiltered



(b) T_{surface} filtered

Figure 5.4: T_{surface} data spectrogram. Unfiltered data (left). Filtered data with low-pass filter with critical frequency at 0.05Hz (right). Note drop in normalized noise signal intensity and less components in frequency domain while preserving data structure.

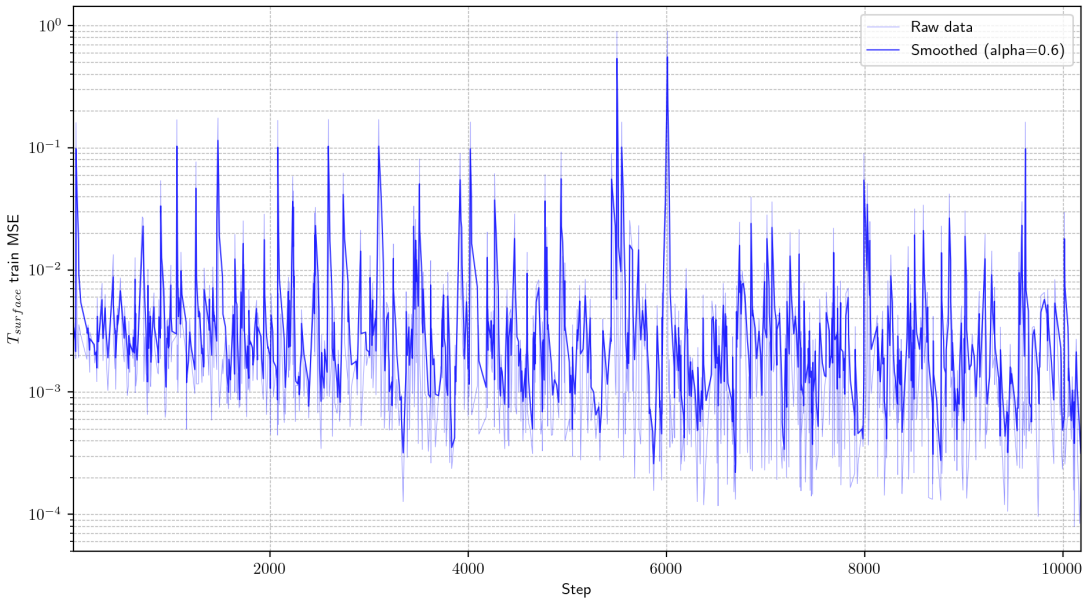


Figure 5.5: Unfiltered data receiver's mean surface temperature T_{surface} learning curve. TensorBoard logger sample.

Initially, the unfiltered dataset was used for training. During training, it was observed that the learning curve of the T_{surface} showed apparent peaks in the loss. Logging values are captured and plotted with the TensorBoard library. This effect is depicted in Figure 5.5 as an extract of the logging board. Contrary to a decreasing learning curve, the graph showed some epochs with loss values higher than the initial value. This unexpected behavior was subject to analysis.

Observing the data as shown in Figure 5.3 and Figure 5.4b, the dataset presented data points with high measurement noise. This concluded that filtering the measurement of the T_{surface} was needed for the complete scope of the work.

After selecting a low-pass filter as the standard filter, and removing regions of high noise from training (Subsection 5.3.1), the MSE in the validation step was reduced. Using a test sample with $w = 30, p = 20$ and $w = 120, p = 20$, and measuring the MSE during validation, the loss value decreased for T_{surface} from 6.743×10^{-3} and 4.449×10^{-3} respectively before filtering, to 9.8×10^{-4} and 1.369×10^{-3} respectively after filtering. This represents an 85.46% and 69.23% reduction in the loss value for validation respectively.

5.3.2 Last layer residual connection

Following the results in Section 5.2, the structure of the T-NN was analyzed and improved. A last layer residual connection was added to the NN as seen in Section 4.2,

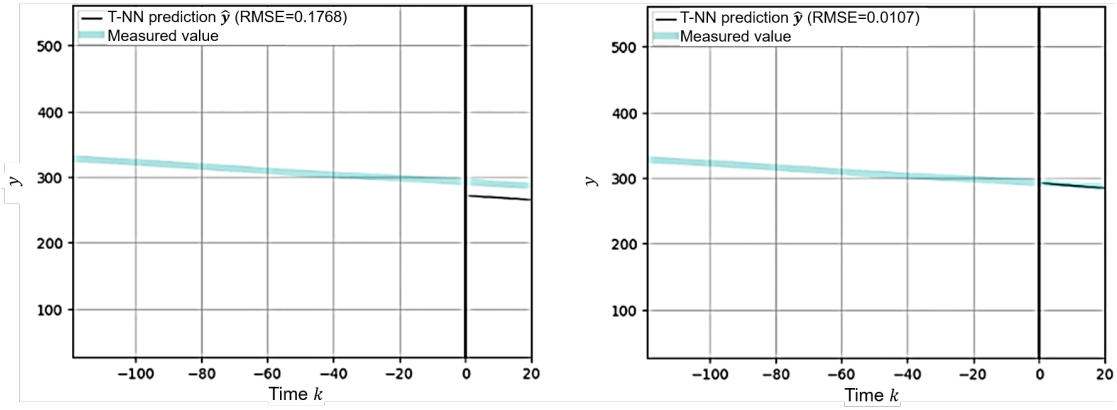


Figure 5.6: Last layer residual connection prediction comparison of receiver hot air temperature $T_{\text{hot air}}$ prediction.

whose structure is depicted in Figure 5.1. After learning the additional residuals, the trained NN showed better performance.

With parameters $w = 120, p = 20$, Figure 5.6 shows a sample of the measured dataset and the NN prediction for $T_{\text{hot air}}$.

On the left, the NN does not have a last layer residual connection. Comparing both graphs, it is clear that the residual connection improved the prediction capabilities. This is observed in the discrepancy between the measured values and the predicted values, where the T-NN with a residual connection captures the dynamics of the system more accurately. For instance, the RMSE of the measurement and prediction without a residual connection has a value of 1.798×10^{-1}, compared to 1.09×10^{-2} with a residual connection, considering unnormalized values.

This discrepancy was observed in other training sequences as well. Using a test sample with $w = 30, p = 20$ and $w = 120, p = 20$, and measuring the MSE for $T_{\text{hot air}}$ during validation, the loss value decreased from 1.711×10^{-3} and 2.95×10^{-3} respectively without a residual connection, to 9.0×10^{-5} and 7.6×10^{-5} respectively with a residual connection. This represents a 94.74% and 97.42% or 2 order of magnitude reduction in the loss value for validation respectively.

Although not investigated, the discrepancy between the ODE and NN model observed in Figure 5.2 is considered capable of reduction when using a last layer residual connection.

To assess the impact of both infrared measurement filtering, data denoising, and the last layer residual connection, a comparative analysis was made. Using a persistence model as described in Section 4.2, the validation loss value was contrasted.

A total of 5 NNs were trained and tested for each combination of parameters. With $w = \{30, 120\}$ and $p = 20$ as selected parameter variations, the MSE loss value for validation and testing was obtained for $T_{\text{hot air}}$ and T_{surface} . This was repeated for each

combination of residual connection and filtering enabled. Last, the same values were captured from a persistence model with the same parameter combinations. When the test MSE value of the trained NN is lower than the persistence model, it is marked as "Y".

As shown in Table 5.5, the results demonstrate that without the last layer residual connection and filtering on T_{surface} measurements, the trained NN only learned the dynamics of the receiver's mean surface temperature, but not for the hot air.

The table also shows that with the last layer residual connection alone, the trained T-NN was capable of learning both $T_{\text{hot air}}$ and T_{surface} better than a persistence model, suggesting effective learning from the data. When testing data filtering alone without residuals, the comparison fails, suggesting that the model did not learn the dynamics of T_{surface} .

While the residual connection enabled the model to learn both dynamics, using data filtering in parallel reduced the loss value further. Even more, while the combination improved the model overall, the reduction of the loss value was greater in the surface temperature dynamics. Thus, the overall MSE for both variables is calculated to account for their influence.

Table 5.5: Validation loss value and persistence model comparison table for w, p, residual connection and filtering parameters

w	p	**Residual connection**	**Filtering**	**MSE validation loss**			**Persistence model test MSE comparison**		
				Sum	$T_{\textbf{surface}}$	$T_{\textbf{hot air}}$	Sum	$T_{\textbf{surface}}$	$T_{\textbf{hot air}}$
30	20	N	N	4.227×10^{-3}	6.743×10^{-3}	1.711×10^{-3}	Y	Y	N
120	20	N	N	3.700×10^{-3}	4.449×10^{-3}	2.950×10^{-3}	Y	Y	N
30	20	Y	N	2.676×10^{-3}	5.262×10^{-3}	9.000×10^{-5}	Y	Y	Y
120	20	Y	N	1.789×10^{-3}	3.503×10^{-3}	7.600×10^{-5}	Y	Y	Y
30	20	N	Y	7.210×10^{-4}	9.800×10^{-4}	4.620×10^{-4}	Y	N	Y
120	20	N	Y	1.526×10^{-3}	1.369×10^{-3}	1.682×10^{-3}	Y	N	Y
30	20	Y	Y	1.850×10^{-4}	3.090×10^{-4}	6.100×10^{-5}	Y	Y	Y
120	20	Y	Y	1.910×10^{-4}	3.150×10^{-4}	6.800×10^{-5}	Y	Y	Y

With the results obtained, it is concluded that a last layer residual connection and measurement filtering are needed for the T-NN model to increase the accuracy of the predictions and therefore improve the efficiency of the MPC controller.

5.3.3 Hyperparameters: Look-back window and prediction horizon

After finding that the model prediction error was reduced by using filters in the measured data and adding a last layer residual connection to the T-NN, it was of interest to find which combination of parameters w (look-back window) and p (prediction horizon) offered higher prediction accuracy. Additionally, the set of values w and p was adapted to explore the use of longer prediction horizons due to the observed slow dynamics of the system.

To obtain prediction accuracy insights, 5 NNs for each combination $p = \{60, 120, \ldots, 300\}$ and $w = \{30, 60, \ldots, 120\}$ were trained, for a total of 20 combinations, and 100 trained NNs.

To make equivalent comparisons, MSE loss values for training, validation, and testing were obtained for each prediction horizon value. This was calculated as:

$$\text{Training loss}_{NN(w,p)} = \begin{cases} MSE(y_k, \hat{y}_k) & \forall k \in \{0, \ldots, i\}, \text{if } i \leq p, \\ 1 & \text{otherwise.} \end{cases} \tag{5.3}$$

Figure 5.7 illustrates Equation 5.3. Consider three NNs trained with a look-back window w and three distinct prediction horizons p. Predictions $\hat{y}_1, \hat{y}_2$ and $\hat{y}_3$ of the real output y are distinct from each other. The prediction deviation from the true value is expected to increase for higher values of k. Calculating the MSE for the complete prediction sequence of length $k = \{0, \ldots, p\}$ for each $\text{NN}_{\{1,2,3\}}$, the values are at most 15% distinct from each other and in the same order of magnitude. The value for each NN does not distinguish if the first predictions are closer to the true value since the overall loss is calculated with values close to each other.

In contrast, if the MSE is calculated for the minimum prediction horizon p (in the example graph $p = 60$), a NN with better prediction accuracy (lower MSE) for that time window can be found. In our example, this is the case for $\text{NN}_3(p = 180)$ $\hat{y}_3$. This is beneficial and applicable since we assume that the MPC utilizes the model where the first optimal control inputs are used before reevaluating the model.

In this example, the $\text{MSE}_{p=60}$ values are 22.0×10^{-3} and 7.6×10^{-3} for NN_1 and NN_3 respectively. This represents a reduction of the MSE loss value of 14.4×10^{-3} or 65%. The method described in Equation 5.3 is used to select the NNs that will be used in the MPC.

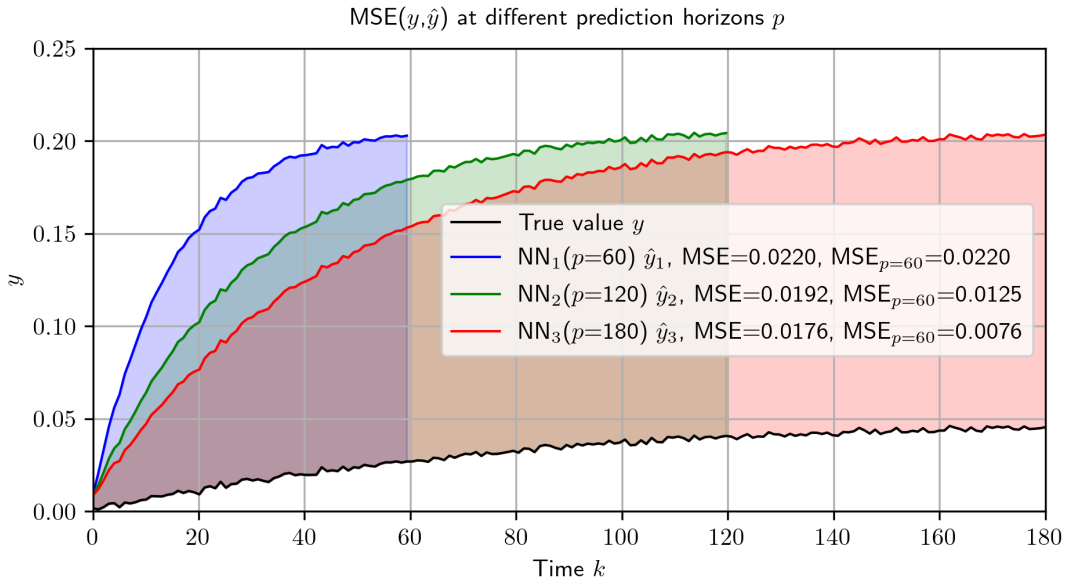


Figure 5.7: Example of neural network mean squared error comparison for different prediction horizons p

Parameters such as the look-back window w and prediction horizon p are important for capturing temporal dependencies and ensuring accurate future state predictions. For this purpose, loss values were obtained for each of the 100 NN trained. Surfaces were created using interpolation and analyzed to provide insights into the optimization surface present during neural network training at different parameter values.

Considering that the receiver hot air $T_{\text{hot air}}$ will be used for tracking MPC, the obtained MSE loss values are ranked at $p = 60$ for that variable (sort on column $T_{\text{hot air},p=60}$).

Looking at Table 5.6, it was found that $\text{NN}_{28}(w = 60, p = 120)$ has the lowest MSE validation loss at 3.01×10^{-4}. For this reason, and considering the slow dynamics present at the STJ plant, $w = 60$ and $p = 120$ are selected as fixed parameters for the following chapters. For a simulated real plant that will be used in a software-in-the-loop simulation, the trained neural network NN_{28} is selected. For the model present in the MPC controller, neural network NN_{25} is chosen as it has the same parameter combination while having worse prediction performance. Having different models and using those with worse performance is intended to explore the influence of dynamic discrepancies in the controller.

It is also noteworthy that the MSE loss value of $T_{\text{hot air}}$ is at least one order of magnitude smaller than that of T_{surface} .

For reference, using data with filtered measurements and deleted regions with high noise, the total number of dataset sequences results in 1062 for $w = 60$ and $p = 120$. After splitting, the number of sequences in the training dataset drops from

Table 5.6: Lowest 10 mean squared error validation loss for w and p combinations ranked at $p = 60$

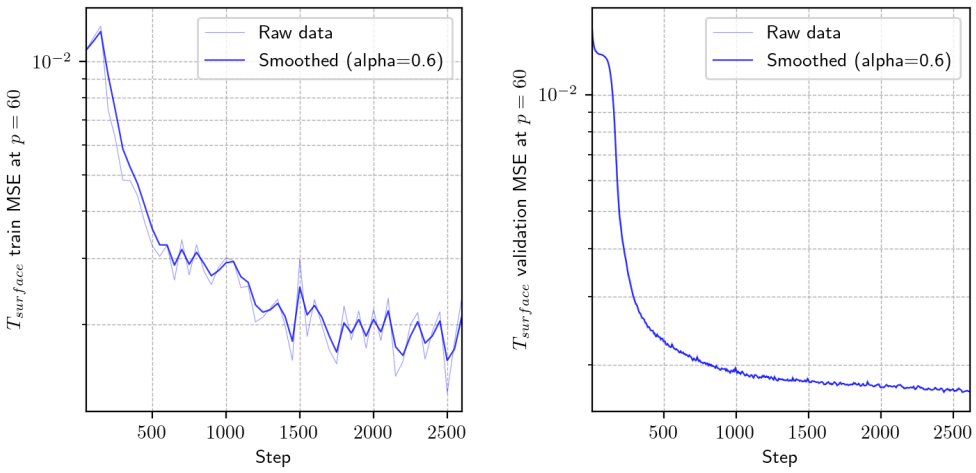
NN	w	p	**MSE validation loss**			
			Sum	$\text{Sum}_{p=60}$	$T_{\textbf{hot air}\ ,\ p=60}$	$T_{\textbf{surface}\ ,\ p=60}$
28	60	120	0.001702	0.000741	0.000301	0.001182
06	60	60	0.000738	0.000737	0.000308	0.001167
92	90	300	0.005142	0.000714	0.000321	0.001107
07	60	60	0.000835	0.000835	0.000322	0.001347
04	30	60	0.001000	0.001000	0.000324	0.001677
29	60	120	0.001889	0.000801	0.000326	0.001276
27	60	120	0.001815	0.000781	0.000337	0.001225
25	60	120	0.001853	0.000798	0.000339	0.001257
16	120	60	0.000699	0.000699	0.000351	0.001047
94	90	300	0.006895	0.000815	0.000352	0.001277

743 to 372. Validation sequences therefore are 212 elements long, and 107 for testing. Additionally, with the above-mentioned parameter configuration, the model presents a total of 219,915 trainable parameters.

As opposed to the learning curves obtained for unfiltered data as shown in Figure 5.5, the corresponding MSE loss values during training and validation did not exhibit the same noise artifacts described. For reference, Figure 5.8a shows the logged values for $p = 60$ during training, while Figure 5.8b shows the values for validation.

Regarding the corresponding loss surfaces, a 2-dimensional interpolation fit was done for all the values. The resulting surface contour plots are obtained as well. Using the calculated surface interpolation values, the minimum z value (MSE loss) was obtained with its corresponding w and p. Figure 5.9 shows the surface contour plot for training, validation, and testing at $p = 60$ using the values of Table 5.6 (Figure 5.9a, Figure 5.9b and Figure 5.9c respectively). Measured values are added as scattered points in the graph marked in red.

The graphs show that there is no apparent correlation in the loss surfaces. What is interesting in this figure is the difference between them. Three behaviors were expected to be observed: similar contours with relatively equal gradients and orders of magnitude, similar or closer minimums, and higher MSE loss at greater values of p.



(a) Training (b) Validation

Figure 5.8: MSE loss curves at $p = 60$

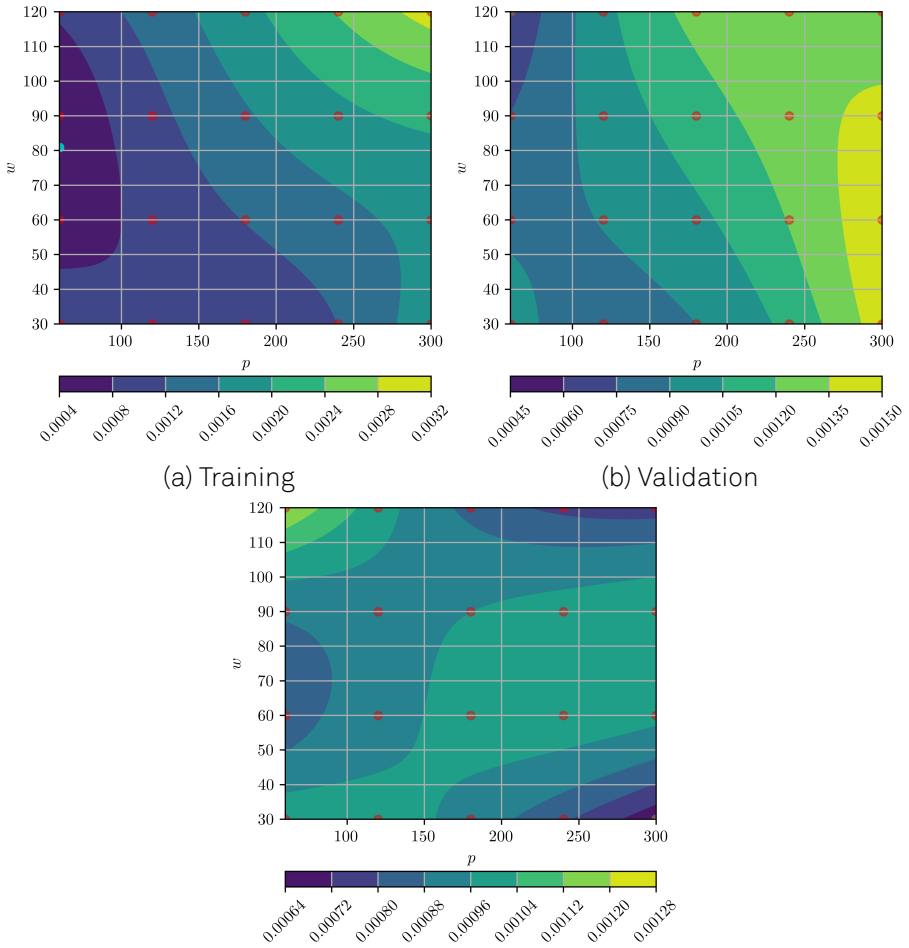
The first point was not met, as lower and higher values between the contours do not match, and the isolines dividing the values are located in different locations. Surface minimum values are 4.0×10^{-4}, 4.5×10^{-4}, and 6.4×10^{-4} for training, validation, and testing respectively. In the same order, surface maximum values are 32.0×10^{-4}, 15.0×10^{-4}, and 12.8×10^{-4}.

Second, surface minimums are located at different combinations of w and p. These are located at $w \approx 81, p = 60$ for training, $w = 120, p = 60$ for validation, and $w = 30, p = 300$ for testing. The loss value for each minimum is 4.42×10^{-4}, 5.94×10^{-4}, and 6.57×10^{-4} respectively.

Last, while training and validation MSE surface contours show a clear increase in the loss value for higher values of p, testing does not.

It is then concluded that, due to the difference between training, validation, and testing MSE loss and their surfaces, a global optimum around the combinations of w and p tested was found. This effect may be due to the size of the dataset, dynamics present in the dataset, or the prevalence of measurement noise. This also opens the line of inquiry to optimize the NN structure to better capture the dynamics and possibly find a suitable combination of parameters.

Training, validation, and testing MSE loss value tables can be found in the Appendix, Chapter C. The corresponding loss surfaces are also available in the Appendix, Chapter B.



(a) Training

(b) Validation

(c) Testing

Figure 5.9: MSE loss surface contour at $p = 60$

Chapter 6

Neural Network Model Predictive Control

This chapter presents the use of NN dynamic models, with a focus on Model Predictive Control (MPC). While MPC relies on accurate models, some dynamics can be challenging to formulate for complex systems. NNs, particularly novel structures such as Transformer Neural Network (T-NN), adopt a data-driven approach to model these dynamics. The captured nonlinearities, also possible for MIMO systems, are thus expressed in a model that can be leveraged by the controller.

This research explores how neural network-based MPC can be implemented and tested in various scenarios. The goal is to demonstrate the applicability of MPC in controlling complex systems, specifically within the context of solar power plants like the Solar Tower Power Plant Jülich (STJ). Additionally, the research investigates the use of optimization algorithms commonly found in NN training applications to solve the OCP presented by the MPC.

Similar to Chapter 5, the First-Order Plus Dead-Time (FOPDT) model presented by Park et al. [39] and the Solar Tower Power Plant Jülich (STJ) model were used. While the first case was used to explore application feasibility, the second employs a more rigorous qualitative methodology. For both models, it is assumed that all states are observable and that the systems are completely controllable in all described control scenarios.

Closed-loop simulations were done and performance of the controller subject to the given models was measured. Such a closed-loop is depicted in Figure 4.6, Section 4.5.

6.1 First oder plus dead-time MPC

To simulate the FOPDT model for testing and performance evaluation, a change in the setpoint y_{ref} was introduced in an unconstrained MPC formulation as in the work of Park et al. [39]. While the paper's authors used TensorFlow [104] for the NN and SciPy library [107] for the OCP, we compared the performance using PyTorch 2.1.1 [105] and replicate using Scipy 1.11.4. These libraries were used with Python 3.11 [103].

The formulation of the MPC was implemented as described in Park et al.[39], which is given by:

$$\underset{\Delta \boldsymbol{u}_{[k+1:k+p]}}{\text{minimize}} \quad \sum_{k=0}^{p-1} (l(\boldsymbol{x}(k), \boldsymbol{u}(k)) + r(\Delta \boldsymbol{u}(k))), \tag{6.1a}$$

$$\text{subject to:} \quad \boldsymbol{x}(k+1) = f(\boldsymbol{x}(k), \boldsymbol{u}(k)), \tag{6.1b}$$

$$\text{where,} \quad l(\boldsymbol{x}(k), \boldsymbol{u}(k)) = f'(\boldsymbol{x}(k), \boldsymbol{u}(k))^\top \boldsymbol{Q} f'(\boldsymbol{x}(k), \boldsymbol{u}(k)), \tag{6.1c}$$

$$r(\Delta \boldsymbol{u}(k)) = \Delta \boldsymbol{u}(k)^\top \boldsymbol{R} \Delta \boldsymbol{u}(k), \tag{6.1d}$$

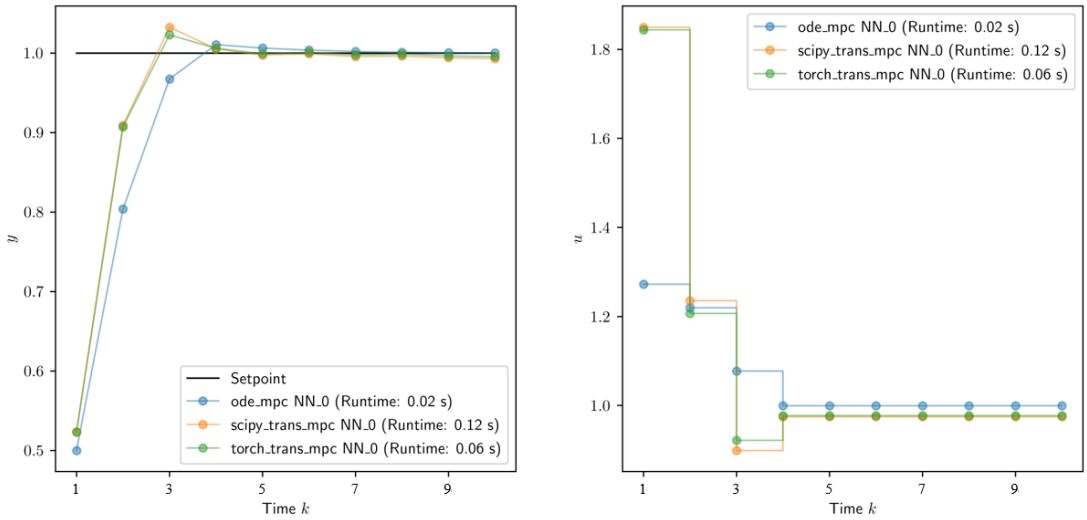
$$\Delta \boldsymbol{u}(k) = \boldsymbol{u}(k) - \boldsymbol{u}(k-1), \tag{6.1e}$$

$$f'(\boldsymbol{x}(k), \boldsymbol{u}(k)) = \boldsymbol{x}(k+1) - \boldsymbol{x}_{ref}(k+1) \tag{6.1f}$$

For the FOPDT model, a single step k was simulated to explore the behavior of the MPC controller. A change in the set point value y_{ref} was set from its initial condition at $y = 0.5$ to $y = 1.0$ using the NN model described in Chapter 5. For comparison, the simulation was also performed using an ODE model.

To compare the solution between the first-principles model and the data-based approach, 3 simulation results are obtained: ODE MPC using SciPy's numeric integrator, a T-NN MPC using SciPy as optimizer, and a T-NN MPC using the PyTorch Adam optimizer. Results are shown in Figure 6.1.

For the ODE simulation, the MPC problem was solved using SciPy's minimize Sequential Least SQuares Programming (SLSQP) optimizer (eps=1e-6, ftol=1e-3). The second simulation was solved with the same approach. The PyTorch simulation was solved using the Adam optimizer (learning rate=0.0238, eps=1e-3). Figure 6.1a shows the output variable $\boldsymbol{y}$ and its simulated dynamics for the 3 simulations. From the graph, we can see that there is a difference between the ODE model and the dynamics described by the T-NN model. In contrast, there is no significant difference between the results obtained using SciPy and PyTorch. Furthermore, while there is a difference between the models, the values converge at step $k = 3$. It is worth noting the discrepancy between the ODE and NN models, which accounts for up to 12.5% at time $k = 2$.



(a) Model output y

(b) MPC controlled input variable u

Figure 6.1: FOPDT response model y at time $k = 1$ for change in setpoint given MPC control input $\boldsymbol{u}^*$. Results replicate reported findings by Park et al. [39]. Note different $\boldsymbol{u}^*$ and $\boldsymbol{y}$ values at $k = 1$ due to model differences.

On the other hand, Figure 6.1b shows the value of the controlled variable $\boldsymbol{u}^*$ over time obtained by the MPC. While the initial guess of the optimizer is initialized with the same value across the simulations, the optimal solution at $k + 1$ is different between the ODE and NN models. This is believed to be due to the model's accuracy. Additionally, and unlike the previous graph, the final value of $\boldsymbol{u}^*$ is also different.

Results obtained also replicate those obtained by Park et al. [39] for the ODE and SciPy cases.

The final value of the cost function demonstrates a clear difference between the first-principles and NN models. With the specified optimizer parameters, the final value of the cost function was 3.5224, 25.9760, and 25.9593 for the ODE-SciPy, T-NN-SciPy, and T-NN-PyTorch models, respectively. These observations were not mentioned in the paper by Park et al. [39], but they do suggest that improvements to the model are needed to achieve higher prediction accuracy and potentially reduce the difference in the cost function due to difference in modeled dynamics.

What stands out in both figures is the different solution times of the MPC cycle. While ODE simulations are typically faster, the difference between the SciPy and PyTorch approaches is significant. On average, the PyTorch approach took 50% less time than using the constrained optimizer in SciPy. While this is partially attributed to the calculation of the gradients for each optimizer, it was not the subject of research in this work.

In this context, the SciPy library uses numerical methods to calculate gradients. Specifically, it computes the gradient using first-order one-sided differences. While

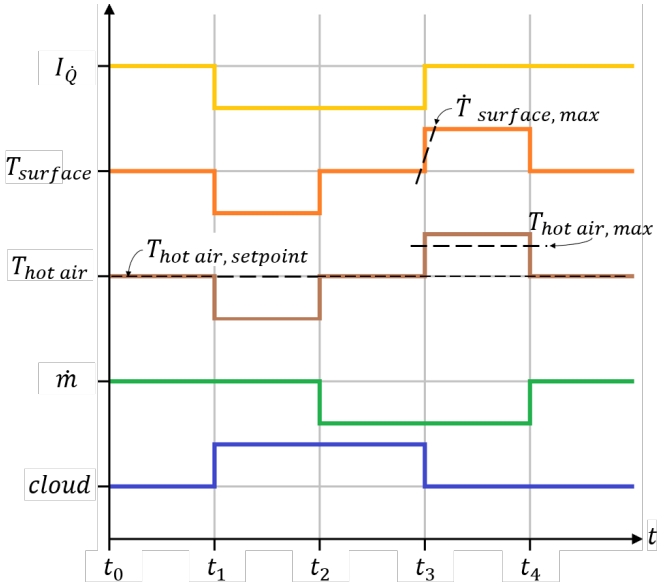


Figure 6.2: STJ motivation scenario for MPC

second-order methods are available, they are typically computationally expensive. On the other hand, PyTorch uses a technique called automatic differentiation using PyTorch's Autograd module. Moreover, the use of tensor-based libraries and parallel NN architectures such as Transformer can further reduce the solution times, especially when improved by using GPUs.
Findings on the applicability of T-NN as dynamics models and the faster MPC computation time compared to constrained optimizers extended the interest of using PyTorch in the STJ case.

6.2 Solar tower Jülich MPC

The STJ is subject to operational constraints in the following parameters: hot air temperature ($T_{\text{hot air}}$), air mass flow ($\dot{m}_{\text{air}}$), and surface temperature ($\dot{T}_{\text{surface}}$). The first of these constraints is related to the maximum operating temperature in the HTM, which is defined as the temperature at which degradation or failure of different components along the fluid's path may occur. The second constraint is associated with the maximum and minimum angular velocity of the compressor that actuates the air in the system. The third constraint is related to temperature gradients. Higher temperature changes than those permitted could damage the receiver's ceramic cups due to internal stresses.

The evolution of the system, variables and cloud disturbances as well as the appearance of these constraints over time are depicted in Figure6.2. From time t_0 to t_1, the system is in a steady state. At time t_1, a cloud blocks the heliostat field, resulting

in a decline in the apparent receiver's surface brightness $I_{\dot{Q}}$ and surface temperature T_{surface} that persists until time t_2. At this point, the controller attempts to mitigate the disturbance by reducing the controlled variable $\dot{m}_{\text{air}}$, thereby restoring the hot air temperature $T_{\text{hot air}}$ to its setpoint. At time t_3, the cloud passes and the irradiation return to their previous levels, while the air massflow remain constant. This presents a risk scenario in which a change in the $\dot{m}_{\text{air}}$ occurs at a later time. At time t_3, a sudden increase in $I_{\dot{Q}}$translates into a higher surface and hot air temperatures. Depending on the disturbance, this increase in surface temperature could also violate the temperature gradients constraints. Consequently, the rise in the temperature of the hot air could reach levels above the maximum safe operational point. This is where the advantages of the MPC could be exploited, by allowing the controller to find a controlled variable trajectory that rejects the disturbances while tracking the setpoint.

6.2.1 Testing scenarios

Based on the values observed in the Solar Tower Power Plant Jülich (STJ) plant measured data , four different testing scenarios were created, divided into two groups. The first group focuses on assessing the MPC in operation regimes close to the constraints (C.V.). The goal of the second group is to obtain insights into nominal operation conditions (N.O.). Each scenario has its initial values.

To simulate changing conditions, 7 consecutive steps were considered. For the first group, each step has a length of 5 minutes, while the second group has steps of 15 minutes. The initial conditions of both scenarios are shown in Table A.1, while the time-variant step values are shown in Table A.2, both in the Appendix, Chapter Datasheets. Direct publication of the raw data from the findings is not permissible. Nonetheless, results are shown in a normalized manner.

Constraint violation and nominal operation scenarios 1 describe a $T_{\text{hot air}}$ tracking MPC controller simulation at constant $I_{\dot{Q}}$for different values of $T_{\text{hot air}}$. In contrast, while having the $T_{\text{hot air}}$ tracking behavior, its value was kept constant in constraint violation and nominal operation scenarios 2. Additionally, these scenarios exhibit changes in $I_{\dot{Q}}$to emulate the influence of solar irradiation disturbances such as cloud passes.

In the case of scenario C.V. 2, disturbances were emulated during steps 2, 4, and 6 by a change in $I_{\dot{Q}}$of -50%, +50%, and -70% relative to the initial value. For scenario N.O. 2, disturbances are simulated in a similar fashion, with -25%, -50%, and -70% relative to the initial value.

Operating constraints are described as lower and upper bounds (L.B. and U.B., respectively) for $\dot{m}_{\text{air}}$. The same was set for $T_{\text{hot air}}$, but with different upper bound values depending on the testing scenario, being lower in the constraint violation scenarios.

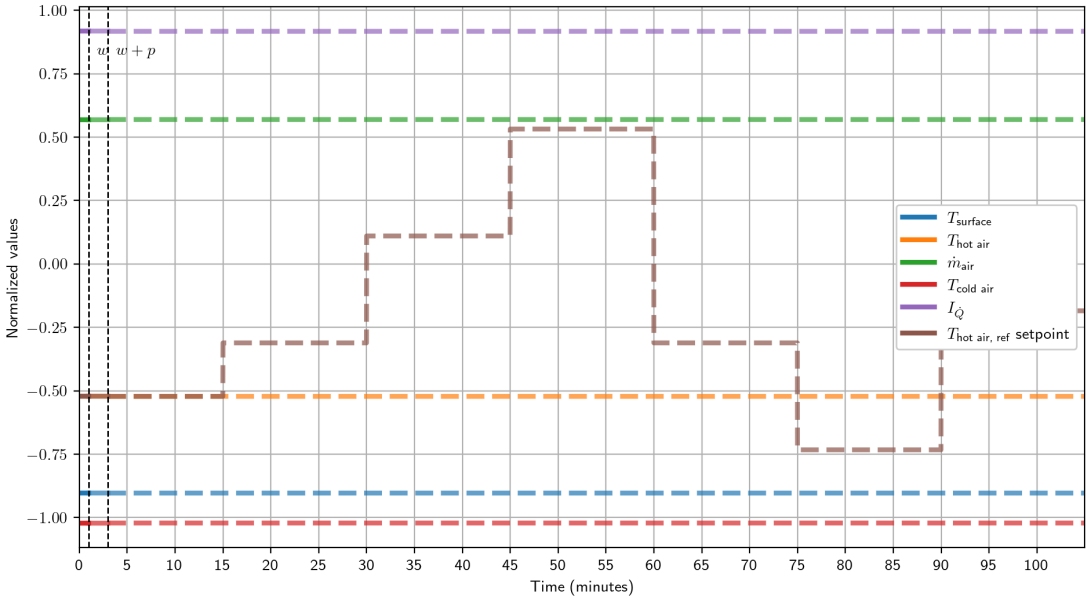


Figure 6.3: Nominal operation test scenario 1. Hot air $T_{\text{hot air}}$ temperature tracking at constant solar irradiation $I_{\dot{Q}}$. MPC moving window $w + p$ in solid line. Future values in dashed line.

Lastly, a constraint on the receiver's surface temperature gradients $\dot{T}_{\text{surface}}$ is imposed to avoid damage to the ceramic material due to rapid heating and cooling. These constraints are shown in Appendix, Table A.3.

Given the constraints, the test scenarios are designed to drive the MPC towards and outside the bounds to test the controller's performance and constraint violation rejection. This is done by setting $T_{\text{hot air}}$ setpoints greater than or equal to the upper bound constraint. The simulation steps at which this situation occurs are marked in bold in Table A.1 and Table A.2.

For purposes of comparison, Figure 6.3 depicts the time graph for the normal operation scenario 1 (N.O. 1). This graph depicts the hot air temperature setpoint $T_{\text{hot air, ref}}$ for the controller and the values for states $\boldsymbol{x}$ and outputs $\boldsymbol{y}$. While outputs, states, and inputs are shown, they illustrate the initial conditions. The scenario is initiated at time $t = 0$, and the solid line depicts the data used as the input sequence in the tensor X. The dashed lines represent future values, which are updated during the simulation. Vertical lines at time $t = w$ and $t = w + p$ are also depicted to illustrate the moving window of the MPC. A graph for all testing scenarios can be found in the Appendix, Chapter B.

Table 6.1: Optimizers used in the MPC formulation

Library	**Optimizer**			
PyTorch	Adam	AdamW	Yogi	L-BFGS
SciPy	Trust-constr	COBYLA	SLSQP	

6.2.2 Barrier functions constrained MPC

This work also tested the performance of newer optimization algorithms to solve the OCP. While Park et al. [39] used SLSQP (Kraft [108]) as the optimizer algorithm with the SciPy library (Virtanen et al. [107], SciPy 1.11.4) to obtain the optimal control sequence, this work was based on other tools. Using the PyTorch library, optimizers such as Adam (Kingma et al. [109]), AdamW (Loshchilov and Hutter [110]), Yogi (Zaheer et al. [111]), and Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) (Schmidt [112]) were tested. The L-BFGS algorithm uses a low-rank Hessian approximation for the line search.

While the mentioned optimizers are built for unconstrained formulations, they were used for constrained problems by means of barrier functions. To compare against constrained optimizers, the SciPy library (Virtanen et al. [107]) is implemented using optimizers such as Constrained Optimization BY Linear Approximation (COBYLA) ([113]–[115]), Sequential Least SQuares Programming (SLSQP) ([108]), and trust-region interior point method (Trust-constr) ([116]).

Table 6.1 shows the list of optimizers used. For further detail, SciPy optimizer options are shown in Table A.6, and PyTorch optimizer options are shown in Table A.5.

The controller is programmed and implemented in the unconstrained case based on Equation 6.2.
For completeness, the formulation of the cost function J of the MPC for the constrained case using unconstrained optimizers is defined as:

$$J(\boldsymbol{x}, \boldsymbol{u}, \Delta\boldsymbol{u}) = \sum_{k=0}^{p-1}(l(\boldsymbol{x}(k), \boldsymbol{u}(k)) + r(\Delta\boldsymbol{u}(k))) + \sum_{i=1}^{m}\phi(g_i(\boldsymbol{x})) \tag{6.2a}$$

$$\text{with:} \quad l(\boldsymbol{x}(k), \boldsymbol{u}(k)) = f'(\boldsymbol{x}(k), \boldsymbol{u}(k))^\top \boldsymbol{Q} f'(\boldsymbol{x}(k), \boldsymbol{u}(k)), \tag{6.2b}$$

$$r(\Delta\boldsymbol{u}(k)) = \Delta\boldsymbol{u}(k)^\top \boldsymbol{R}\Delta\boldsymbol{u}(k), \tag{6.2c}$$

$$\phi(g_i(\boldsymbol{x})) = \sum_{k=0}^{p-1}(\hat{I}_-(\boldsymbol{v}_j(\boldsymbol{x}))), \quad j = [1, 6] \tag{6.2d}$$

where $\hat{I}\_(\boldsymbol{v}(\boldsymbol{x}))$ represent the barrier function for the value function $\boldsymbol{v}$.

Furthermore, the MPC gain matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ are selected consistently for each study case to ensure equivalence across different scenarios. These are given as:

$$\boldsymbol{R} = \begin{bmatrix} 1 \end{bmatrix} \tag{6.3a}$$

$$\boldsymbol{Q} = \begin{bmatrix} 1 & 0 \\ 0 & 10 \end{bmatrix} \tag{6.3b}$$

Considering 6 constraints in total, 3 variables ($T_{\text{hot air}}$, $\dot{m}_{\text{air}}$, $\dot{T}_{\text{surface}}$), and 2 bounds each (lower and upper bound), the value functions $\boldsymbol{v}_j(\boldsymbol{x})$ for the constrained formulation are given as:

$$v_1 = \dot{\tilde{m}}_{\text{air}} - \dot{m}_{\text{air, U.B.}}, \tag{6.4a}$$

$$v_2 = \dot{m}_{\text{air, L.B.}} - \dot{\tilde{m}}_{\text{air}}, \tag{6.4b}$$

$$v_3 = \tilde{T}_{\text{hot air}} - T_{\text{hot air, U.B.}}, \tag{6.4c}$$

$$v_4 = T_{\text{hot air, L.B.}} - \tilde{T}_{\text{hot air}}, \tag{6.4d}$$

$$v_5 = \dot{\tilde{T}}_{\text{surface}} - \dot{T}_{\text{surface, U.B.}}, \tag{6.4e}$$

$$v_6 = \dot{T}_{\text{surface, L.B.}} - \dot{\tilde{T}}_{\text{surface}} \tag{6.4f}$$

where the tilde variable represents the obtained value at each optimization iteration. The tuning parameters β, γ, and T, each corresponding to the scaled Softplus and exponential barrier functions respectively are given in Appendix, Table A.4. While these values were manually tuned using all test scenarios to get a desired behavior, the process is not considered part of this work and therefore the methodology is not further detailed. For the following, the exponential barrier function is used.

6.2.3 Closed-loop controller

To measure controller performance in the STJ study case, metrics such as constraint violation (C.V.), total effort $\sum \Delta \boldsymbol{u}$, surface temperature gradients $\dot{T}_{\text{surface}}$, tracking RMSE $T_{\text{hot air, RMSE}}$, and estimated heat flux $\dot{\tilde{H}}_{\text{air}}$ were obtained in simulation. MPC solution time was captured as well for reference and comparison, but no major analysis was performed.

To gain insights on the power output of the receiver, a reduced model of the outlet air enthalpy flux is considered for result quantification based on the work of Ostermann [32]. This is given by the formula 6.5:

$$\dot{\tilde{H}}_{\text{air}} = C_{p,\,\text{air}}(T_{\text{hot air}} - T_{\text{cold air}})\dot{m}_{\text{air}} \tag{6.5}$$

with $C_{p,\,\text{air}} = 1.005$ (air specific heat [kJ/(kg °C)]).

Measured metrics are first normalized and then ranked. Two approaches were implemented for ranking. The first evaluates the aggregated normalized values ratio (Equation 6.6a), while the second makes a Softmax aggregated assessment of the results (Equation 6.6b). Individual ranking values are given for sorting from smaller to greater values, except for the estimated heat flux $\dot{\tilde{H}}_{\text{air}}$, where a higher value indicates better performance due to a higher energy state.

$$\begin{aligned}\text{Aggregated ratio ranking} =&(\dot{m}_{\text{air, U.B. C.V.}} + \dot{m}_{\text{air, L.B. C.V.}} \\ &+ T_{\text{hot air, U.B. C.V.}} + T_{\text{hot air, L.B. C.V.}} \\ &+ \sum \dot{T}_{\text{surface}} + \sum \Delta \boldsymbol{u} \\ &+ T_{\text{hot air, RMSE}})/(\sum \dot{\tilde{H}}_{\text{air}})\end{aligned} \tag{6.6a}$$

$$\begin{aligned}\text{Softmax aggregated ranking} =&(\text{Softmax}(\dot{m}_{\text{air, U.B. C.V.}}) + \text{Softmax}(\dot{m}_{\text{air, L.B. C.V.}}) \\ &+ \text{Softmax}(T_{\text{hot air, U.B. C.V.}}) + \text{Softmax}(T_{\text{hot air, L.B. C.V.}}) \\ &+ \text{Softmax}(\sum \dot{T}_{\text{surface}}) + \text{Softmax}(\sum \Delta \boldsymbol{u}) \\ &+ \text{Softmax}(T_{\text{hot air, RMSE}}))/(\text{Softmax}(\sum \dot{\tilde{H}}_{\text{air}}))\end{aligned} \tag{6.6b}$$

As mentioned in Chapter 5, two NNs were selected. One will model the dynamics of the plant inside the MPC controller, while the second acts as the real plant. Considering the two models and the closed-loop diagram, the simulation is then carried out as explained in Algorithm 3.
Finally, for the purpose of this work, each library (SciPy and PyTorch) computes its own gradient information. Although the MPC used PyTorch-based NNs for most of the simulations, results are only obtained using the CPU.

6.2.4 Constraint violation case

One of the advantages of using MPC is its reported higher performance compared to other approaches. This is particularly important for controlling the system in the presence of constraints. For this reason, the STJ study case was further analyzed using the four different test scenarios described in Section 6.2.

Algorithm 3 NN-based MPC simulation closed-loop

Initialize dataset $\mathcal{D}$
Define look-back window w
Define prediction horizon p
Initialize time index $k = 0$
while Simulation **do**
  $\mathbf{X} \leftarrow \boldsymbol{x}_{[k-w+1:k+p]}$ from $\mathcal{D}_{[k]}$ (Get input from dataset at k),
  repeat
    $\mathbf{Y} \leftarrow \text{NN}_{\text{MPC plant model}} \leftarrow \mathbf{X}$ (Evaluate NN model),
    $\boldsymbol{u}^*_{[k+1:k+p]} \leftarrow$ MPC (Calculate optimal control sequence)
  until break condition
  $\mathbf{X}' \leftarrow \mathbf{X} \leftarrow \boldsymbol{u}^*_{[k+1:k+p]}$ (Update tensor),
  $\mathbf{Y}' \leftarrow \text{NN}_{\text{real plant}} \leftarrow \mathbf{X}'$ (Evaluate real plant model),
  $\mathcal{D}_{[k+1]} \leftarrow \mathbf{Y}'$ (Update dataset at k+1),
  $k = k + 1$ (Increment time index)
end while

In the analysis, both ranking methods (Softmax aggregated ranking and Aggregated ratio ranking) yielded the same score results. Consequently, only the Softmax aggregated ranking method is shown in the results. For reference, the aggregated ratio ranking values can be found in the Appendix, Chapter C, Subsection Aggregated ratio ranking results C.2.2.

To evaluate performance in testing scenarios prone to constraint violations, simulations were run using test scenarios C.V. 1 and 2. Each case imposes constraints, and two approaches to solve the OCP were used: a constrained optimizer in SciPy and an unconstrained optimizer in PyTorch with barrier functions to handle constraints.

The performance of the used optimizer from both approaches (SciPy and PyTorch) were compared by ranking of the metrics. Table 6.2 (Appendix Table C.4) shows results of simulations in test scenarios C.V. 1 and 2 for all optimizers. Zero value columns are omitted for visualization purposes as they do not change the result.

This table reveals several key points. First, it directly compares the score for each optimizer for the SciPy and PyTorch approaches, showing that PyTorch performed better by having the lowest score overall. In this case, the lowest score among the SciPy optimizers was 4.04 (Trust-constr) and 3.98 (COBYLA) for C.V. 1 and 2 respectively. Between PyTorch optimizers 3.71 (Adam) and 3.80 (L-BFGS) for the same scenarios.
The data indicates that for test scenario C.V. 1, the SciPy-based MPC violated the upper bound constraints in $T_{\text{hot air}}$, an operation point which is undesirable. In addition, the PyTorch-based MPC produced less temperature change in the surfaces, reducing the gradients $\dot{T}_{\text{surface}}$ overall.

The table also shows that the PyTorch approach reported a lower total effort $\sum \Delta \boldsymbol{u}$ in the controlled variable $\dot{m}_{\text{air}}$, representing softer transitions and control of the sys-

tem. Similarly, this approach reduced the error between the desired reference hot air temperature $T_{\text{hot air, ref}}$ and the measured value $T_{\text{hot air}}$, as indicated by the lower RMSE value.

Unlike the previous metrics, where a lower value means better performance, a higher value of $\sum \dot{\tilde{H}}_{\text{air}}$ indicates a higher energy state in the system, translating to more energy available for downstream processes. The Adam optimizer in PyTorch managed to control the system into a state of higher heat flux compared to Trust-constr. The final ranking for C.V. 1 suggested that the PyTorch approach using Adam as the optimizer performed the best overall, with a score of 3.71.

Ranking results observed in test scenario C.V. 1 also apply to C.V. 2. The interesting aspect of this data is that in this scenario, both approaches violated the constraints. Nevertheless, the amount of violation was significantly higher in the SciPy approach compared to PyTorch. A metric at which constrained optimizers excelled in this scenario was $\sum \dot{T}_{\text{surface}}$, but with a probability difference no greater than 1%. Thus, the final score of 3.80 suggests that the PyTorch approach performed better relative to SciPy.

It is important to note that for C.V. scenario 2, the simulation time for the SciPy approach using optimizer SLSQP took over 150 hours. This greatly reduced the performance of this optimizer, since the simulated simulation time was only 35 minutes.

Despite these results, another relevant aspect of the investigation was the value of the cost function when using unconstrained optimizers available in Python. This is crucial since the values of the barrier functions are directly added to the cost function J in the MPC. If the parameters are not chosen carefully, barrier function values may exceed the stage cost and control effort, potentially driving the solution away from the optimum. To address this, the cost function value at the end of each cycle in all testing scenarios was captured. Particularly, the constraint violation testing scenarios were explored. Figure 6.4 shows the aggregated and disaggregated values of the cost function for test scenario C.V.2. As seen in Table 6.2, both approaches violated the constraints.

The plot consists of three graphs. The first graph shows the individual values over the simulation for the objective function, the stage and control effort costs, and penalties (barriers).
The second graph shows the individual values of the stage and control effort costs. In this scenario, the control effort was close to zero, while most of the cost went to tracking the reference in the stage cost.

From the bottom graph, each barrier function relative to each constraint is graphed. It can be seen that the major component of the barrier is to drive the system away from violating the upper bound constraint in the hot air temperature $T_{\text{hot air}}$.

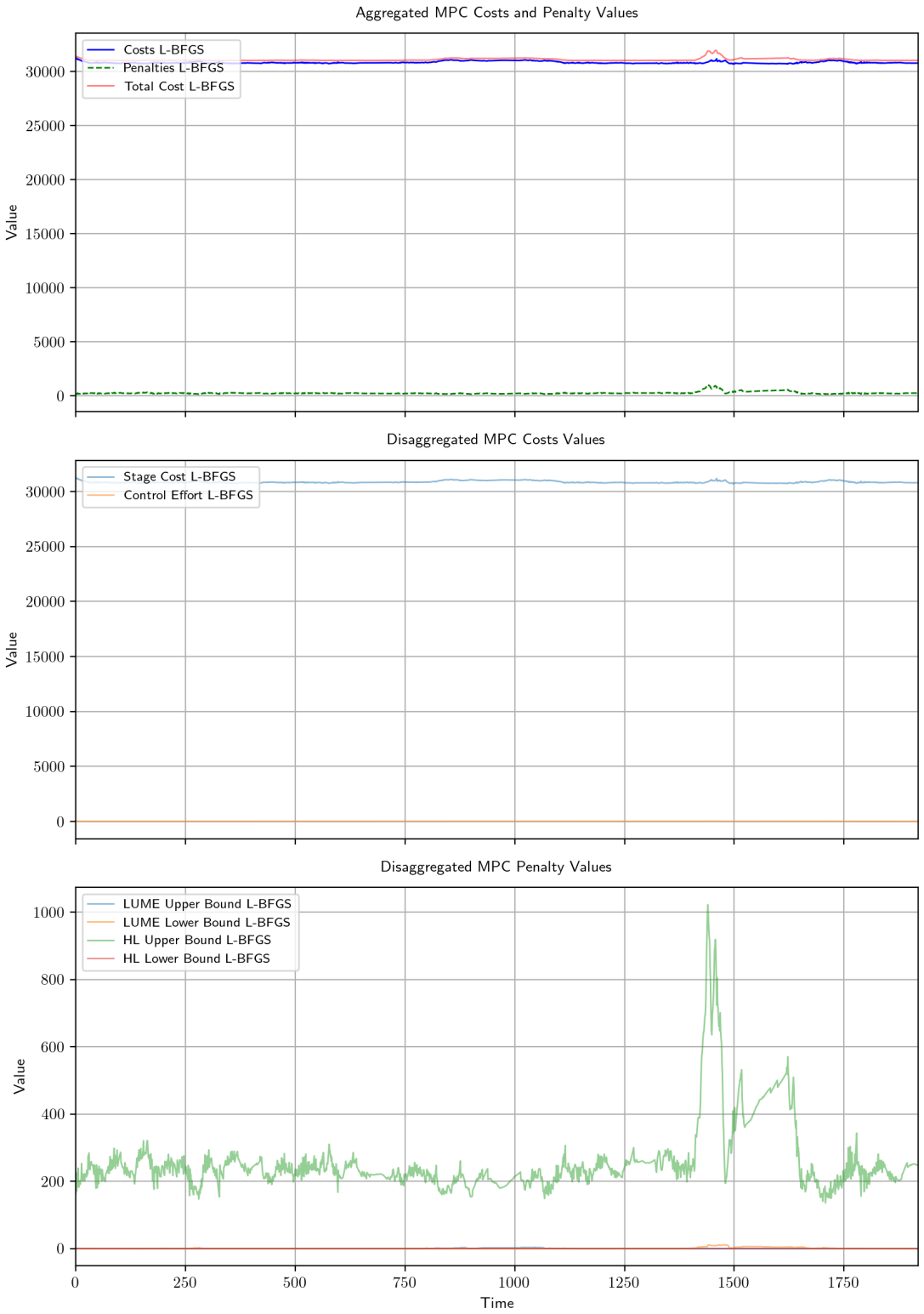


Figure 6.4: PyTorch MPC costs on test scenario constraint violation 2

The most striking observation from the graph is the difference in the order of magnitude between the costs and the penalties. This is clear in the upper graph, where the penalty values represent approximately 3% of the costs. These results were also observed in all test scenarios.

Table 6.2: Softmax aggregated ranking PyTorch vs SciPy results on test scenarios Constraint Violation (C.V.) 1 and 2 for all different optimizers. Constrained violations in $T_{\text{hot air}}$ observed. PyTorch MPC showing better performance compared to SciPy.

Scenario	**Library**	**Optimizer**	$\dot{m}_{\text{air}}$, U.B. C.V.	$\dot{m}_{\text{air}}$, L.B. C.V.	$T_{\text{hot air}}$, U.B. C.V.	Rank	$T_{\text{hot air}}$, L.B. C.V.	$\sum \dot{T}_{\text{surface}}$	Rank	$\sum \Delta \boldsymbol{u}$	Rank	$T_{\text{hot air}}$, RMSE	Rank	$\sum \dot{\tilde{H}}_{\text{air}}$	Rank	**Score**
Constraint Violation 1	SciPy	COBYLA	**1.00**	0.00	**0.12**	0.13	0.00	0.13	0.14	0.13	0.14	0.16	0.15	0.08	0.13	4.13
		SLSQP	0.00	0.00	**0.77**	0.26	0.00	0.18	0.15	0.18	0.15	0.14	0.14	0.14	0.14	4.93
		Trust-constr	0.00	0.00	**0.04**	0.12	0.00	0.15	0.14	0.15	0.14	0.15	0.14	0.11	0.14	**4.04**
	PyTorch	Adam	0.00	0.00	0.00	0.12	0.00	0.13	0.14	0.13	0.14	0.14	0.14	0.17	0.15	**3.71**
		AdamW	0.00	0.00	0.00	0.12	0.00	0.14	0.14	0.14	0.14	0.14	0.14	0.16	0.15	3.72
		Yogi	0.00	0.00	**0.01**	0.12	0.00	0.15	0.14	0.15	0.14	0.14	0.14	0.17	0.15	3.79
		LBFGS	0.00	0.00	**0.07**	0.13	0.00	0.12	0.14	0.12	0.14	0.14	0.14	0.17	0.15	3.71
Constraint Violation 2	SciPy	COBYLA	0.00	**1.00**	**0.07**	0.13	0.00	0.12	0.14	0.09	0.13	0.13	0.14	0.10	0.14	**3.98**
		SLSQP	0.00	0.00	**0.30**	0.17	0.00	0.24	0.16	0.16	0.15	0.25	0.16	0.21	0.15	4.13
		Trust-constr	0.00	0.00	**0.23**	0.16	0.00	0.12	0.14	0.08	0.13	0.13	0.14	0.10	0.14	4.14
	PyTorch	Adam	0.00	0.00	**0.11**	0.14	0.00	0.13	0.14	0.19	0.15	0.12	0.14	0.15	0.14	3.95
		AdamW	0.00	0.00	**0.09**	0.14	0.00	0.13	0.14	0.19	0.15	0.12	0.14	0.15	0.14	3.94
		Yogi	0.00	0.00	**0.14**	0.14	0.00	0.14	0.14	0.23	0.16	0.12	0.14	0.15	0.14	4.06
		LBFGS	0.00	0.00	**0.06**	0.13	0.00	0.12	0.14	0.07	0.13	0.12	0.14	0.15	0.14	**3.80**

6.2.5 Nominal operation case

Regarding testing scenarios with nominal operation regimes, Table 6.3 (Appendix Table C.5) shows results of simulations in test scenarios N.O. 1 and 2 to compare the performance of the used optimizer from both approaches (SciPy and PyTorch) by ranking of the metrics.

Interestingly, while the best performing optimizer in SciPy was Trust-constr for the constraint violation scenarios, it was no longer the case for the nominal operation scenarios. SLSQP performed best in N.O. scenarios while performing worst in C.V. scenarios. Despite starting conditions away from constraints, none of the approaches drove the system towards constraint violation regions for both nominal operation scenarios.

Closer inspection of the table shows that contrary to the constraint violation scenarios, the performance difference between both approaches was relatively smaller. For instance, in nominal operation scenario 1, the SciPy approach obtained a lowest score of 3.83 against 3.79 for PyTorch. This trend was also observed in scenario 2, where PyTorch had a better performance with a lower score of 3.85 compared to SciPy's 3.87.

Similar to the constraint violation test scenarios, the solutions obtained by the PyTorch-enabled MPC had better performance relative to the SciPy approach. The only metric where the best performing SciPy optimizer achieved a better result was in $T_{\text{hot air, RMSE}}$ in test scenario N.O. 2, which might be due to the higher $\sum \Delta \boldsymbol{u}$ in the controlled variable.

These results can also be observed in the simulation graphs. While results for nominal operation test scenarios 1 and 2 are presented in this chapter, all simulation graphs and results can be found in the Appendix, Chapter B Graphs, Section B.2. Values shown in the figures are normalized.
Figure 6.5 shows the simulation results for test scenario nominal operation 1. This figure is composed of seven graphs.

The first graph shows the simulation values for the surface apparent brightness $I_{\dot{Q}}$. For scenarios C.V. and N.O.1, this is kept constant.
In the second graph, the desired temperature value $T_{\text{hot air, ref}}$ is shown, along with the system dynamics for both approaches while tracking the set point. The data shows that both SciPy- and PyTorch-enabled MPC were able to track the desired setpoint, albeit SciPy did so slightly slower in the first two steps. The model captured the dynamics in the domain of validity sufficiently, evident at each step change. Whenever there is a higher $T_{\text{hot air, ref}}$, controllers reduce $\dot{m}_{\text{air}}$ to allow the receiver to give less energy to the HTM. This is visible in the drop in $T_{\text{hot air}}$, which later increases to the lower mass flow. The dynamics of the STJ and the predictive behavior of the controller en-

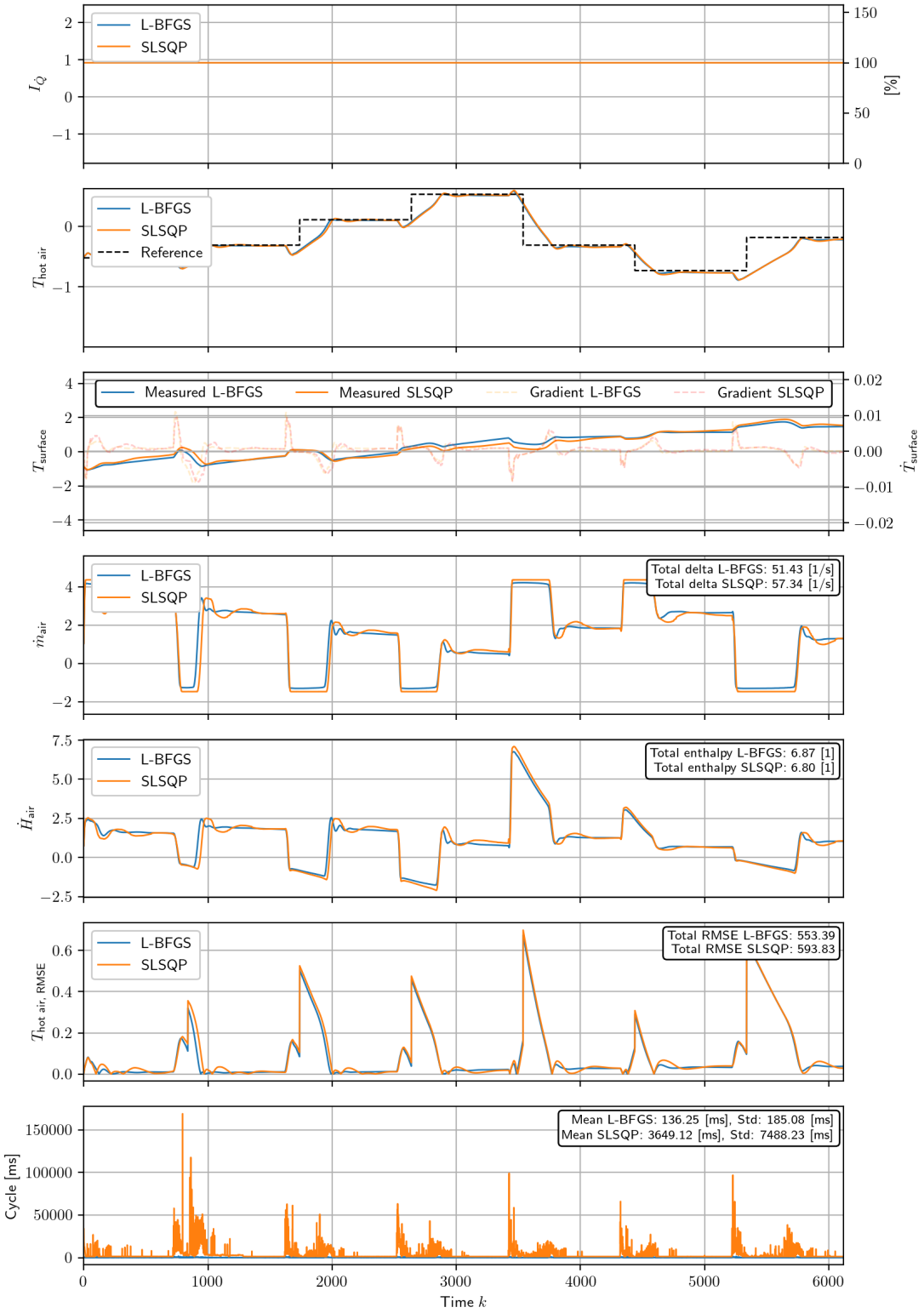


Figure 6.5: PyTorch vs SciPy simulation results on test scenario nominal operation 1

able the system to reach the new setpoint with little overshoot. During steady state, the controller efficiently tracks $T_{\text{hot air, ref}}$, keeping the $T_{\text{hot air, RMSE}}$ value close to zero.

The third graph shows the surface temperature T_{surface} measured by the IR system, with surface temperature gradients $\dot{T}_{\text{surface}}$ shown in parallel, with a second Y-label. This graphically demonstrates if the dynamics were closer to the gradient constraints. What stands out in this graph is the simulated increasing T_{surface} of the system. This result is somewhat counterintuitive. While expected during steps with increasing setpoints, the opposite was not observed during decreasing steps. This would mean that the system's energy is continuously increasing despite operating at lower points. A reason for this result could also be due to the NN model and insufficient dynamics data to reproduce the system accurately for T_{surface} . Although the gradient graph indicated higher values at setpoint changes and during steady-state regions, the expected behavior was for the surface temperature to converge to zero, suggesting minimal changes.

The fourth graph shows the evolution of the controlled variable air mass flow $\dot{m}_{\text{air}}$ over the simulation, along with gradients on $\dot{m}_{\text{air}}$ to get insights on the effort for the whole test scenario. Gradients' absolute values are added to get the total delta, or the total rate of change, shown as metric $\sum \Delta \boldsymbol{u}$ in the previous tables. The figure indicates that the PyTorch approach reacted faster to setpoint changes.

The fifth graph shows the heat flux $\dot{\tilde{H}}_{\text{air}}$ of the system calculated according to Equation 6.5. Despite reaching higher hot air temperatures, the heat flux was lower due to the constant $I_{\dot{Q}}$and the controller tracking only $T_{\text{hot air, ref}}$. The system dynamics required lower air mass flow values to achieve higher temperatures, reducing $\dot{\tilde{H}}_{\text{air}}$.

The sixth graph depicts the RMSE over time between $T_{\text{hot air}}$ and $T_{\text{hot air, ref}}$, shown as $T_{\text{hot air, RMSE}}$, illustrating the deviation from the setpoint. These values correlate with setpoint changes, approaching zero when the system achieved steady-state.

Finally, the seventh graph shows the MPC cycle time for both approaches. Notably, a significant reduction in the MPC cycle time was achieved with the PyTorch approach. On average, PyTorch computed the optimal solution of the OCP in 136.25, 107.20, 49.88, and 122.29 ms for test scenarios N.O. and C.V. 1 and 2, respectively. In comparison, the best performing SciPy optimizer controller cycle time was 3649.12, 2193.44, 17694.68, and 29957.71 ms. This represents a change of up to two orders of magnitude in the C.V. case and one order of magnitude in the N.O. case. This is much more important when a plant is desired to be controlled in Real-Time. For the STJ, PyTorch offers solutions under 1 second, lower than the measuring frequency, potentially enabling Real-Time operation. Although faster computation times can be achieved by down-scaling the model and sampling the system at lower frequencies, these results show a clear benefit of using PyTorch and Autograd gradient calculation.

The most surprising aspect of the data is the $T_{\text{hot air, RMSE}}$ relation between both approaches. While the system dynamics and control inputs might have followed different trajectories, the relative equal value in this metric suggests that both approaches followed the desired value $T_{\text{hot air, ref}}$ similarly on average. Additionally, the selection of the parameters in the barrier functions for the PyTorch approach allowed the controller to be closer to the constraints compared to SciPy.

In general, similar behavior can be seen in the simulation results for all test scenarios. The related figure shown in Appendix Section B.2 illustrates that PyTorch can compute solutions similar to a constrained optimization approach based on SciPy. Moreover, results demonstrate control with less effort and lower deviation from the setpoint.

Another important aspect is the performance of both approaches in testing scenarios where disturbances are simulated as variations in $I_{\dot{Q}}$. For instance, Figure 6.6 (Appendix Figure B.11) shows the simulation results for this case.

It stands out that the controller effectively uses the prediction capabilities of the T-NN model to reject most disturbances. The first two disturbances simulate cloud passes with brightness reductions of 25% and 50%, respectively. In these cases, the controller adjusts $\dot{m}_{\text{air}}$ to track $T_{\text{hot air, ref}}$, reducing $T_{\text{hot air, RMSE}}$. However, the controller struggled with the third disturbance, which simulates a cloud pass with a 70% reduction in $I_{\dot{Q}}$. Under this condition, the system cannot track the reference due to insufficient solar radiation to heat the receiver. This limitation is evident in the $\dot{m}_{\text{air}}$ value, which reaches its lower bound during the last disturbance.

The most surprising aspect of the graph is the system response after the disturbance. Even when $I_{\dot{Q}}$returns to the initial state, the controller did not track the reference. Moreover, the final simulation values show a clear reduction in the hot air temperature. One reason for this behavior might be the NN model, which may not have learned the dynamics accurately for this combination of inputs due to the training data points.

This is also highlighted in the T_{surface} graph. The simulation shows an upward trend in surface temperature, expected to drop when the system temperature dropped. This remains open for further research.

Additional graphs are also available to the reader. Constraint violation values are graphed for all testing scenarios. These were not shown in the chapter as their final values are captured in the tables but can be found in Appendix, Section B.2.

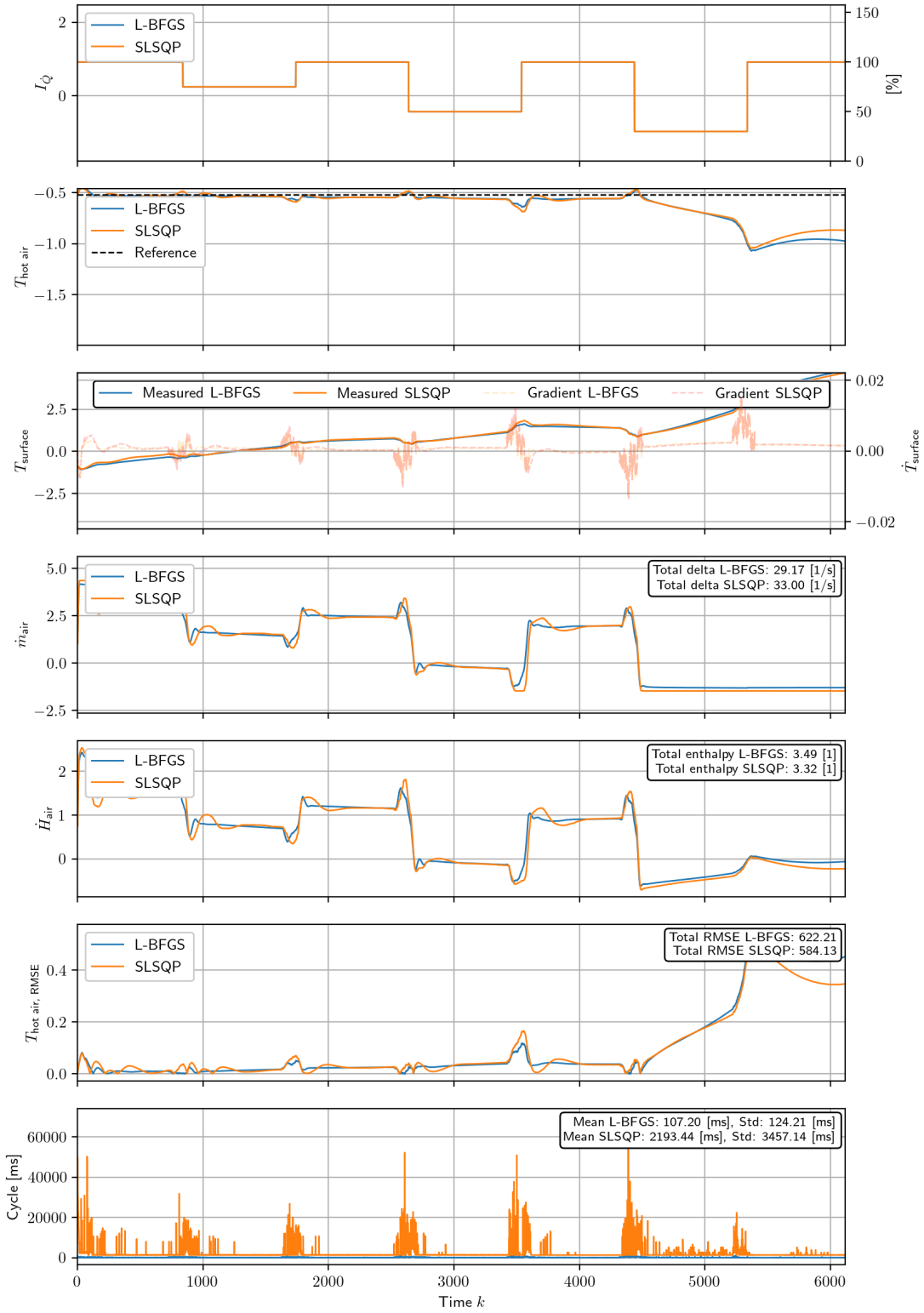


Figure 6.6: PyTorch vs SciPy simulation results on test scenario nominal operation 2

Table 6.3: Softmax aggregated ranking PyTorch vs SciPy results on test scenarios Nominal Operation (N.O.) 1 and 2 for all different optimizers. No constraints violations observed. MPC using PyTorch scores lower showing better performance compared to SciPy.

Scenario	**Library**	**Optimizer**	$\dot{m}_{\text{air}}$, U.B. C.V.	$\dot{m}_{\text{air}}$, L.B. C.V.	$T_{\text{hot air}}$, U.B. C.V.	Rank	$T_{\text{hot air}}$, L.B. C.V.	$\sum \dot{T}_{\text{surface}}$	Rank	$\sum \Delta \boldsymbol{u}$	Rank	$T_{\text{hot air}}$, RMSE	Rank	$\sum \dot{\tilde{H}}_{\text{air}}$	Rank	**Score**
Nominal Operation 1	SciPy	COBYLA	0.00	**1.00**	0.00	0.14	0.00	0.15	0.14	0.10	0.14	0.44	0.19	0.07	0.13	4.64
		SLSQP	0.00	0.00	0.00	0.14	0.00	0.15	0.14	0.14	0.14	0.07	0.13	0.17	0.15	**3.83**
		Trust-constr	0.00	0.00	0.00	0.14	0.00	0.14	0.14	0.13	0.14	0.18	0.15	0.11	0.14	4.17
	PyTorch	Adam	0.00	0.00	0.00	0.14	0.00	0.15	0.14	0.15	0.14	0.08	0.13	0.17	0.15	3.86
		AdamW	0.00	0.00	0.00	0.14	0.00	0.15	0.14	0.14	0.14	0.09	0.13	0.15	0.14	3.91
		Yogi	0.00	0.00	0.00	0.14	0.00	0.14	0.14	0.21	0.15	0.07	0.13	0.17	0.15	3.88
		LBFGS	0.00	0.00	0.00	0.14	0.00	0.13	0.14	0.13	0.14	0.07	0.13	0.17	0.15	**3.79**
Nominal Operation 2	SciPy	COBYLA	0.00	**1.00**	0.00	0.14	0.00	0.15	0.14	0.16	0.14	0.21	0.15	0.12	0.14	4.18
		SLSQP	0.00	0.00	0.00	0.14	0.00	0.14	0.14	0.13	0.14	0.09	0.14	0.16	0.14	**3.87**
		Trust-constr	0.00	0.00	0.00	0.14	0.00	0.16	0.14	0.10	0.14	0.26	0.16	0.10	0.14	4.28
	PyTorch	Adam	0.00	0.00	0.00	0.14	0.00	0.14	0.14	0.13	0.14	0.11	0.14	0.16	0.15	3.89
		AdamW	0.00	0.00	0.00	0.14	0.00	0.15	0.14	0.12	0.14	0.14	0.14	0.15	0.14	3.97
		Yogi	0.00	0.00	0.00	0.14	0.00	0.13	0.14	0.24	0.16	0.10	0.14	0.16	0.15	3.98
		LBFGS	0.00	0.00	0.00	0.14	0.00	0.13	0.14	0.12	0.14	0.10	0.14	0.16	0.15	**3.85**

6.2.6 Measurement noise

To emulate the influence of measurement noise, uniform random distribution noise was added to the measurements at time $k = 0$ in the MPC model. The amount of noise is given as:

$$\mathbf{X}_{[k=0,:]} = \mathbf{X}_{[k=0,:]} + \text{Noise}_{\text{max}} \cdot \text{Noise}_{\text{random scale}}, \tag{6.7a}$$

$$\text{where:} \quad \text{Noise}_{\text{random scale}} = 2 \cdot \mathcal{N}(\mathbf{X}_{[k=0,:]}; (0, 1]) - 1, \tag{6.7b}$$

$$\text{Noise}_{\text{max}} = \text{Noise}_{\%} \cdot max(\mathcal{D}) \tag{6.7c}$$

$max(\mathcal{D})$ represents the maximum observed value in the dataset. For example, assume a state-of-the-art Open Volumetric Receiver (OVR) with Heat Transfer Medium (HTM) temperatures of up to 500 °C. A 5% measurement noise based on the maximum value $max(\mathcal{D}) = 10°C$ measurement error would represent over 2 orders of magnitude higher inaccuracy than standard PT100 temperature sensors with tolerance class W 0.6, F type, according to DIN EN IEC 60751.

Figure 6.7 shows the results obtained from simulations with 0% and 5% measurement noise. The $T_{\text{hot air}}$ graph shows that controller performance decreased compared to simulations without noise. Furthermore, accuracy degraded in relation to the amount of noise. Despite the influence, the controller follows the $T_{\text{hot air, ref}}$ trend, albeit with higher oscillation in the presence of higher noise. This is also visible in the $T_{\text{hot air, RMSE}}$ graph, where the effect is clearly visible. Quantitatively, a $\sum T_{\text{hot air, RMSE}}$ of 812.98 and 621.88 with 5% and 0% noise respectively is observed. This represents up to 30.43% higher total RMSE respectively.

Noise effect is also visible in the third graph, which reveals different T_{surface} for each case. The striking aspect of the graph is the gradients $\dot{T}_{\text{surface}}$, which also behave different relative to each other. For 0% measurement noise, values were still within the bounds, but this was close to the constraint in the case of 5% noise.

An increase in $\sum \Delta \boldsymbol{u}$ is also observed in the $\dot{m}_{\text{air}}$ graph. Similar effects were found in $\dot{\tilde{H}}_{\text{air}}$. Most interestingly, MPC cycle times remained under 1 second on average, but clear steps with 1 second solution times are observed.

Figure 6.8 presents experimental data on increasing measurement noise for metrics absolute total $\sum \Delta \boldsymbol{u}$ and $T_{\text{hot air, RMSE}}$. With $\text{Noise}_{\%} = [0\%, 0.5\%, 1.0\%, 2.5\%, 5.0\%, 7.5\%, 10\%]$ results show a direct correlation. The increase in the metric value for increasing values of simulated measurement noise is depicted.

At noise levels of 7.5% and 10%, metric values were not captured. This was due to the NN model returning NaN or $+$ inf values in the output tensor $\mathbf{Y}$, thus ending the optimization cycle of the MPC due to non-convergence.

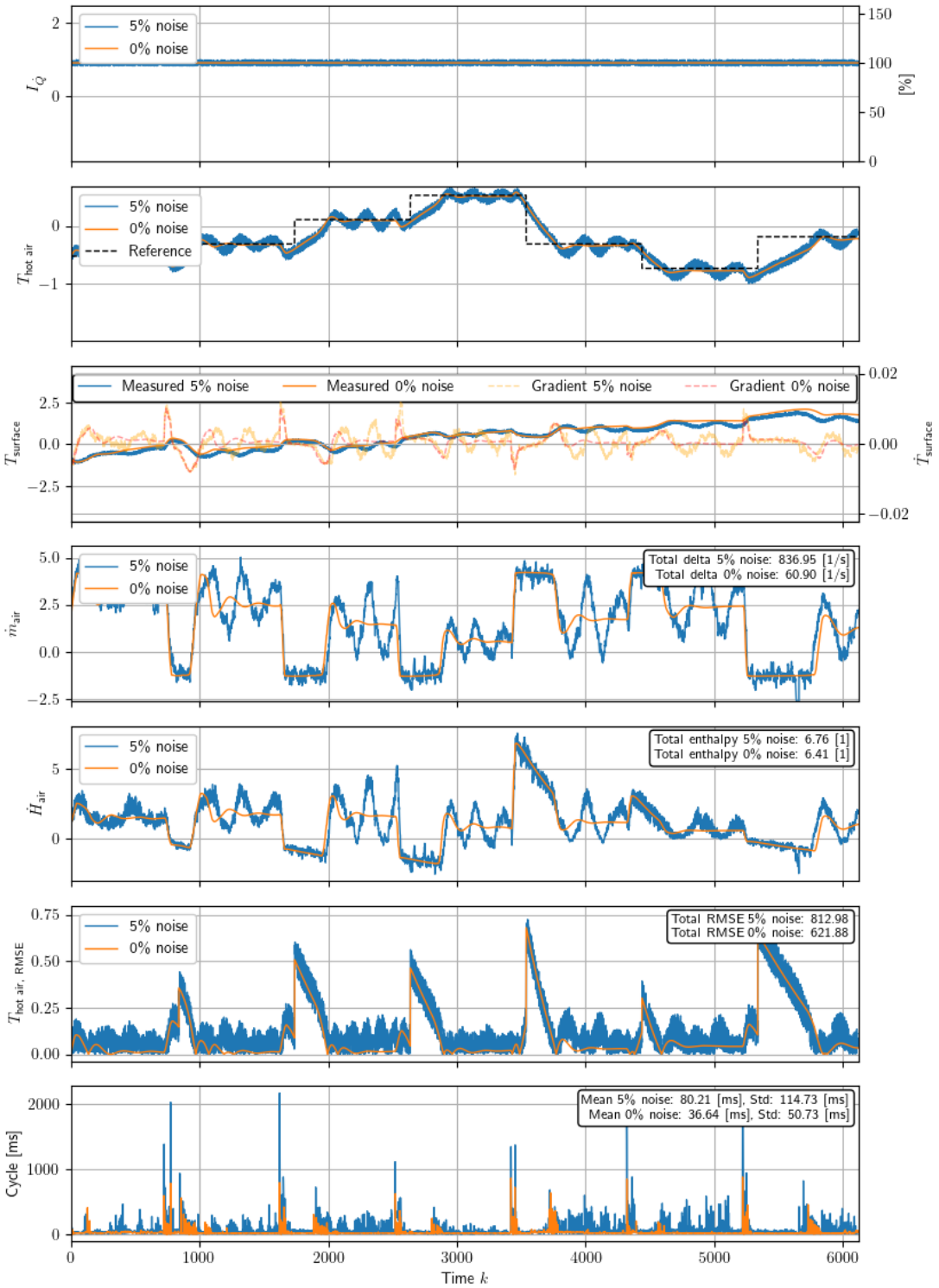
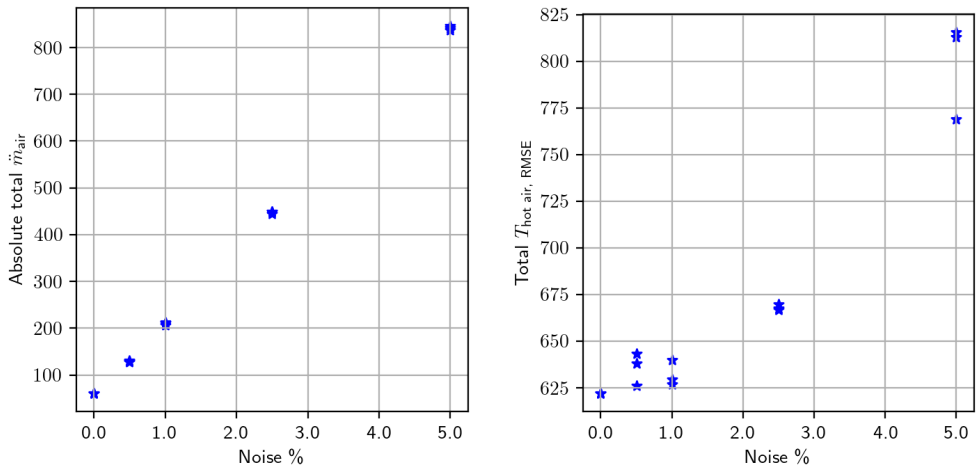


Figure 6.7: PyTorch simulation with 0% and 5% measurement noise on test scenario nominal operation 1



(a) $\sum \Delta \boldsymbol{u}$ (b) $T_{\text{hot air, RMSE}}$

Figure 6.8: Metric value per noise % of maximum value.

Overall, these results suggest that a T-NN dynamic model can be used for MPC, even in the presence of measurement noise. Additionally, the results indicate the viability of PyTorch as a constrained optimizer using barrier functions. This approach performed better relative to an off-the-shelf constrained optimizer used in the literature. Results also show that the simplified T-NN dynamic model of the STJ system is capable of Real-Time solutions.

Chapter 7

Uncertainty Quantification

The primary objective of this research is to investigate the applicability of Transformer Neural Network (T-NN) within Model Predictive Control (MPC). Additionally, this study explores Uncertainty Quantification (UQ) using Kernel-based Gaussian Process Regressor (GPR). By incorporating probabilistic information into the model's predictions, this research aims to lay the groundwork for future applications in this domain.

Recent advances in the literature have introduced innovative approaches such as incorporating a Bayesian final layer in neural networks, which allows these models to infer probabilistic information intrinsically [51]. In contrast, similar to other existing methodologies, our approach fits the model discrepancy between historical predictions and actual measurements using GPR [48]–[50].

To achieve accurate and reliable predictions, an online training and inference mechanism is employed, which augments prediction data by including the computed mean (μ) and covariance (Σ). This study adopts the First-Order Plus Dead-Time (FOPDT) model as proposed by Park et al.[39], utilizing the Gaussian Processes module from the SciKit library 1.3.2 [117]. The use of Solar Tower Power Plant Jülich (STJ) case study was also explored using the GPyTorch library 1.11 [118] in Python 3.11 [103].

Another advantage of using these libraries is their use of a Kernel approach, which is flexible for testing combinations. The main disadvantage of this method is the lack of theoretical analysis of the GP, and the parameters need to be computed every cycle, posing a limitation for real-time applications. These remain subjects for future study.

This study uses stored NN predictions $\hat{\boldsymbol{y}}$ and true plant output measurements $\boldsymbol{y}_{measured}$ to calculate a model error as $\hat{y}_{error} = \hat{\boldsymbol{y}} - \boldsymbol{y}_{measured}$. The online process inside the MPC cycle is illustrated in Algorithm 4.

The GPR parameters for both Python libraries are in Appendix, Section A.3 GP parameters Table A.7.

Algorithm 4 GP model error model fit

Initialize dataset $\mathcal{D}$
Define look-back window w
Define prediction horizon p
Initialize time index $t = 0$
Define model error model length l
Define Confidence Intervals $C.I.(\sigma)$
Initialize $\hat{y}_{error,\,[k=l,j=0:p]}$, $\hat{y}_{store,\,[k=l,j=0:p]}$, $\boldsymbol{y}_{measured,\,[k=l,j=0:p]}$
while Simulation **do**
  $\mathbf{X} \leftarrow \boldsymbol{x}_{[k-w+1:k+p]}$ from $\mathcal{D}$ (Get input from dataset)
  $\hat{y}_{store,\,[0,:]} \leftarrow NN(\mathbf{X})$ (Store NN model predictions)
  $\boldsymbol{y}_{measured,\,[0,:]} \leftarrow Real\ Plant(\mathbf{X})$ (Measure real plant model outputs)
  if $t = l$ **then**
    $\hat{\boldsymbol{y}}_{error,\,[0,:]} \leftarrow \hat{y}_{store,\,[-1,:]} - \boldsymbol{y}_{measured,\,[-1,:]}$ (Calculate prediction error)
    $\hat{\boldsymbol{y}}_{error,\,[0,:]} \sim \mathcal{GP}(\mu,\, \sigma^2)$ (Fit to a GP)
    $\hat{\boldsymbol{y}}' \leftarrow \hat{\boldsymbol{y}} + \mu_{\hat{\boldsymbol{y}}_{error}}$ (Calculate adjusted predictions)
    $\hat{\boldsymbol{y}}' \pm C.I. \cdot \sigma_{\hat{\boldsymbol{y}}_{error}}$ (Calculate prediction confidence regions)
  end if
  Shift $\hat{\boldsymbol{y}}_{error}, \hat{\boldsymbol{y}}_{store}, \hat{\boldsymbol{y}}_{measured}$: $[1:,:] \leftarrow [:-1,:]$
  Display Confidence Intervals from σ
  $t = t + 1$ (Increment time index)
end while

7.1 First order plus dead-time model

Similar to Section 5.2, the dataset from the FOPDT model described in Park et al. [39] was used, with NN parameters as presented in Section 5.2. The kernel utilized for the analysis was a product of a Constant and RBF kernels.

To simulate plant dynamics, the same random test scenario of inputs u used to generate the data set was used to obtain the NN response and fit $\hat{y}_{error}$. Figure B.23 in Appendix, Section B.3, shows the test scenario for the FOPDT case. In addition, 5% of $\sigma(u)$ normally distributed random noise was added to the input of the NN to evaluate the performance of the GPR in the presence of noise.

Figure 7.1 shows the result of the open-loop simulation for T-NN prediction at simulation time step $t = 126$. The upper graph shows the prediction from the T-NN model $\hat{y}$, and the lower graph shows the prediction of the NN model plus the GPR mean μ and with the confidence intervals (C.I.) of $\pm 3\sigma$.

What stands out from the figure is that the true data y_{truth} is inside the confidence intervals of the GPR and these intervals are narrow. In comparison, Figure 7.2 shows the simulation result at time $t = 77$, where a wider confidence region is evident. These differences can be explained in part by the delay in the uncertainty quantification, resulting from the model error being made of past measurements. This is especially

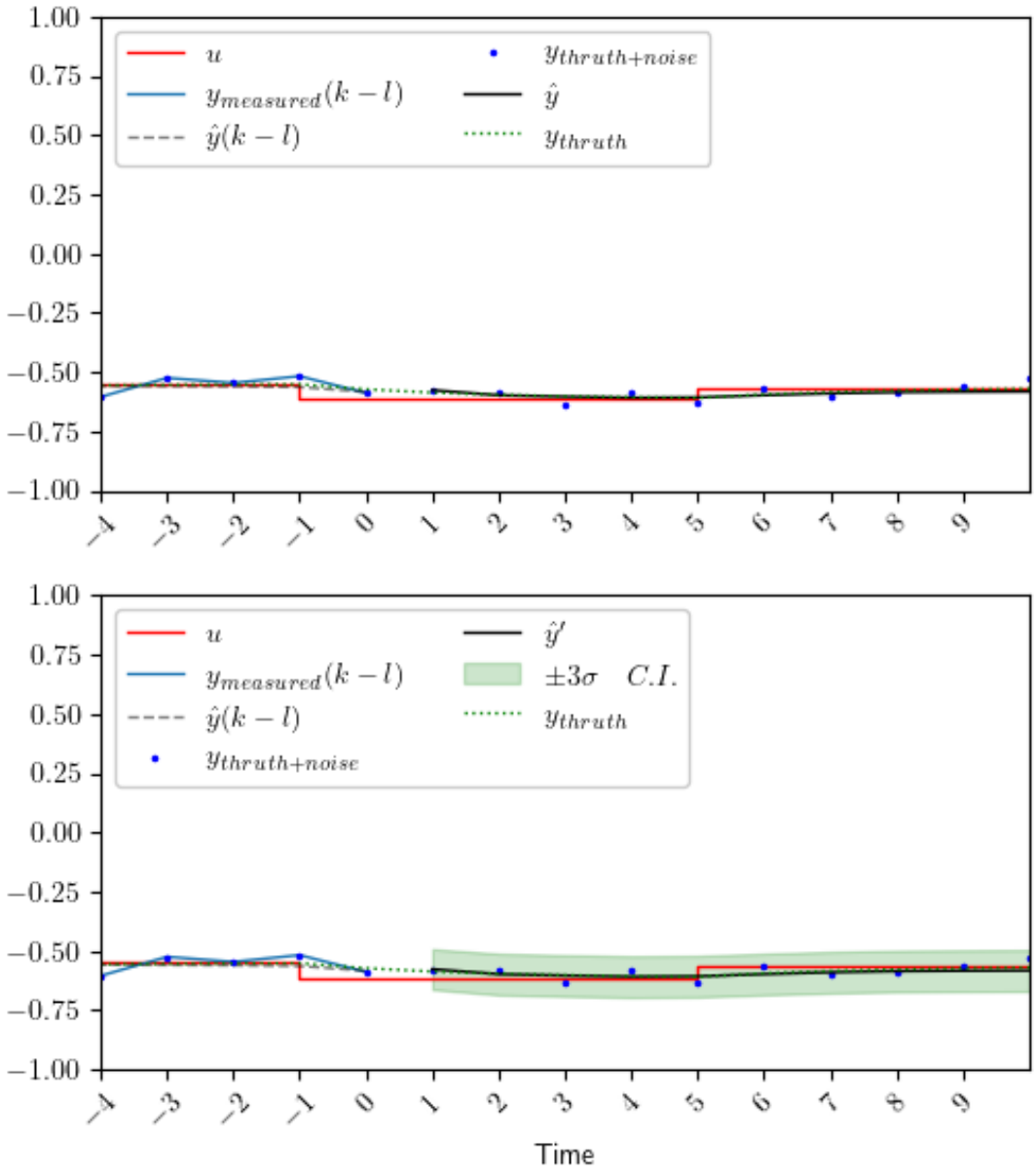


Figure 7.1: FOPDT narrow confidence interval at test scenario time $t = 126$ with input u and MPC sliding window. T-NN model (Top). T-NN model with GP model error model regressor and $\pm 3\sigma$ confidence intervals (Bottom). Note that due to low prediction error, μ_{GP} is close to zero and the difference between the two models small.

evident at points with higher changes in the input variables u or the desired setpoint y_{ref}, as illustrated in the figure. This observation also highlights the inaccuracy of the model and its potential for improvement.

An interesting result from the data is that despite the added noise, out of the 785 times for which the GPR was evaluated during the test, y_{truth} was inside the confidence interval of $\pm 3\sigma$ (99.7%) 93.50% (762 of 785) and $\pm 2.58\sigma$ (95%) 97.07% (762 of 785).

The effect of changes in the input u that contribute to wider confidence intervals can also be seen in Figure 7.3. This graph shows the RMSE of both models ($\hat{\boldsymbol{y}}$ and $\hat{y}_{error}$). From the graph, it is clear that the prediction that included the mean of the GPR worsened the prediction when changes in the setpoint occurred. In addition, no advantage was found for the RMSE over the values obtained. Overall, the $\hat{y}_{error}$ model performed 14% worse than the $\hat{y}$ model, with a total $\hat{\boldsymbol{y}}_{RMSE} = 0.0300$ and $\hat{y}_{error,\,RMSE} = 0.0342$.

It is recommended that the GPR be analyzed in the STJ case study. The results of the FOPDT model demonstrate that this approach is viable, with confidence regions aligning with the predicted trends. These results could be incorporated into the predictions.

For reference, graphs with the calculated model error $\hat{\boldsymbol{y}}_{error}$ and its corresponding GPR fit, as well as calculated $\hat{\boldsymbol{y}}_{RMSE}$ and $\hat{y}_{error,\,RMSE}$ at times $t = 77$ and $t = 126$, can be found in Appendix, Section B.3 Uncertainty Quantification. See Figure B.25 and Figure B.24, respectively.

7.2 Solar tower Jülich model

Another application of interest was the integration of statistical information in the prediction of dynamic models in closed-loop control. Focusing on the STJ case to examine the behavior of Gaussian Process (GP) model errors was studied in conjunction with T-NN models under optimal control inputs u^*. Similar to the FOPDT model, a product of constant and RBF kernels was used.

To extend the scope of UQ methods in this work, quantitative measures were incorporated to evaluate the prediction accuracy of NN dynamic models. As an introductory experiment, initial simulations were performed according to the nominal operation scenario 1 (N.O. 1) described in Section 6.2.

Figure 7.4 illustrates the closed-loop simulation results at time step $t = 338$ for the two variables predicted by the T-NN: $T_{\text{hot air}}$ and T_{surface}.

Both graphs show predictions by the NN model and measured states. An extended model, combining the T-NN model predictions with the GP mean μ_{GP}, is also shown, along with $\pm 3\sigma$ confidence intervals.

The results show that the GP model error model effectively quantifies prediction covariance and captures the dynamic trend of the system. Notably, during steady-

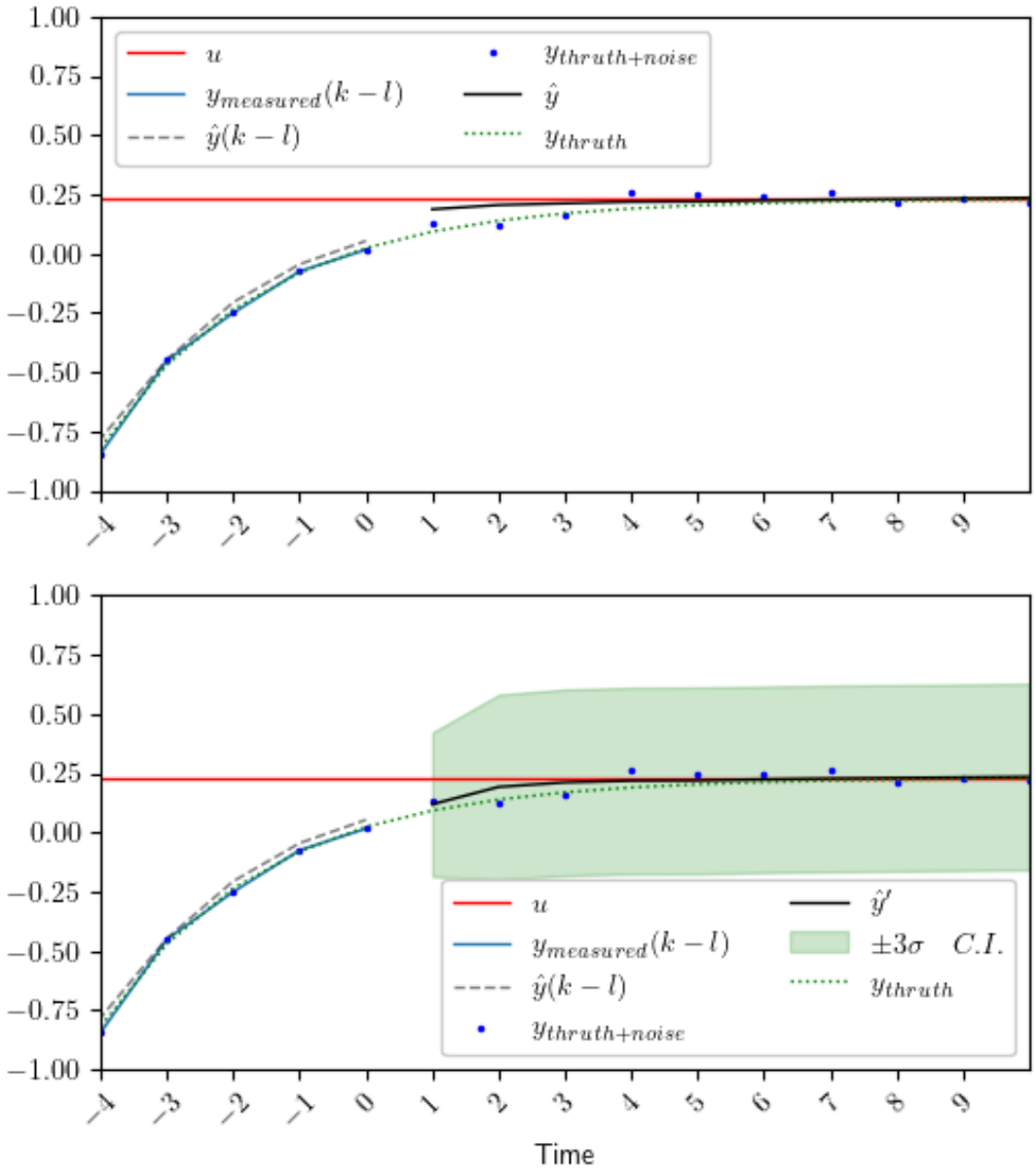


Figure 7.2: FOPDT wide confidence interval at test scenario time $t = 77$ with input u and MPC sliding window. T-NN model (Top). T-NN model with GP model error model regressor and $\pm 3\sigma$ confidence intervals (Bottom). Note that due to higher prediction error, μ_{GP} contribute to the NN predictions while the confidence intervals is wider suggesting lower certainty.

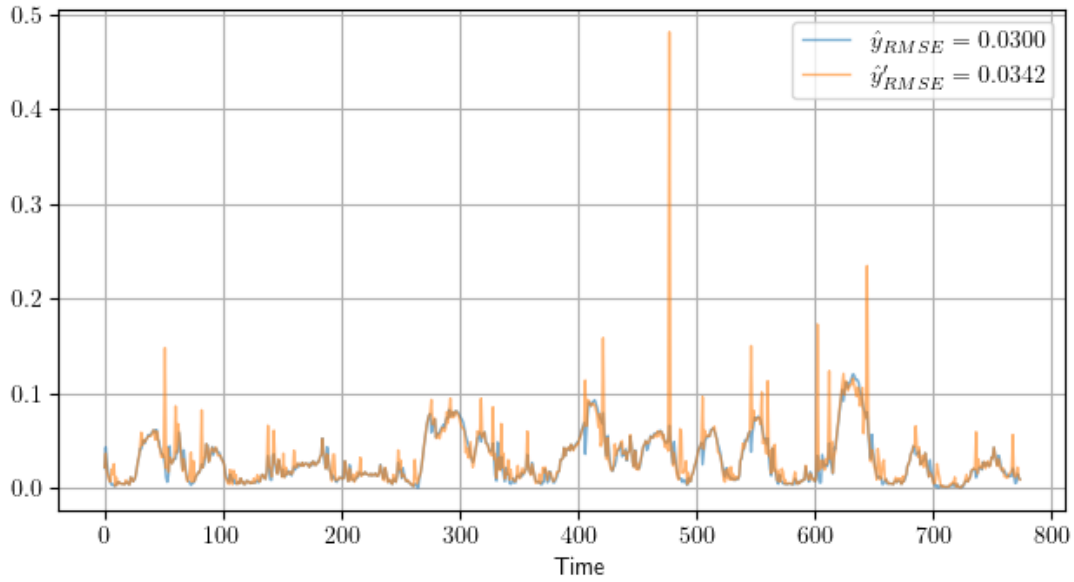


Figure 7.3: Neural network and added Gaussian Process models prediction root mean squared error

state conditions (no setpoint changes or rapid dynamics), the mean $\pm 3\sigma$ over the prediction horizon accounts for only 1.78% of T_{surface} and 3.22% of $T_{\text{hot air}}$ at this time step, suggesting a high level of confidence in the predictions.

A significant contrast was found at later time steps in the simulation. Figure 7.5 shows the simulation results when a change in the hot air temperature $T_{\text{hot air, ref}}$ was introduced (time $t = 774$).

Similar to the FOPDT model, the GP model exhibited a notable bias in response to rapid setpoint changes. It is worth noting that the mean $\pm 3\sigma$ over the prediction horizon increased to only 2.62% of T_{surface} and 10.87% of $T_{\text{hot air}}$ at this time step, suggesting a lower confidence in the predictions.

As last finding, it was observed that μ_{GP} introduced an offset in the NN predictions. This phenomenon can be attributed to the delayed effect of incorporating past measurements and the incorporation of model error in a closed-loop system. This distinction is particularly important. As Lucia explains: "open-loop predictions are not the same as closed-loop trajectories [...] because we are using a finite horizon" [119]. Open-loop control systems rely on predetermined predictions without integrating feedback for adjustments. In contrast, closed-loop control systems use continuous feedback to constantly refine and modify control actions based on real-time data [120].

These results illustrate the feasibility of combining NN models with UQ methods such as GP to include statistical information such as covariance. The findings suggest that, although there are certain constraints, this methodology shows potential for enhancing prediction accuracy and offer an overview in the uncertainty. Future

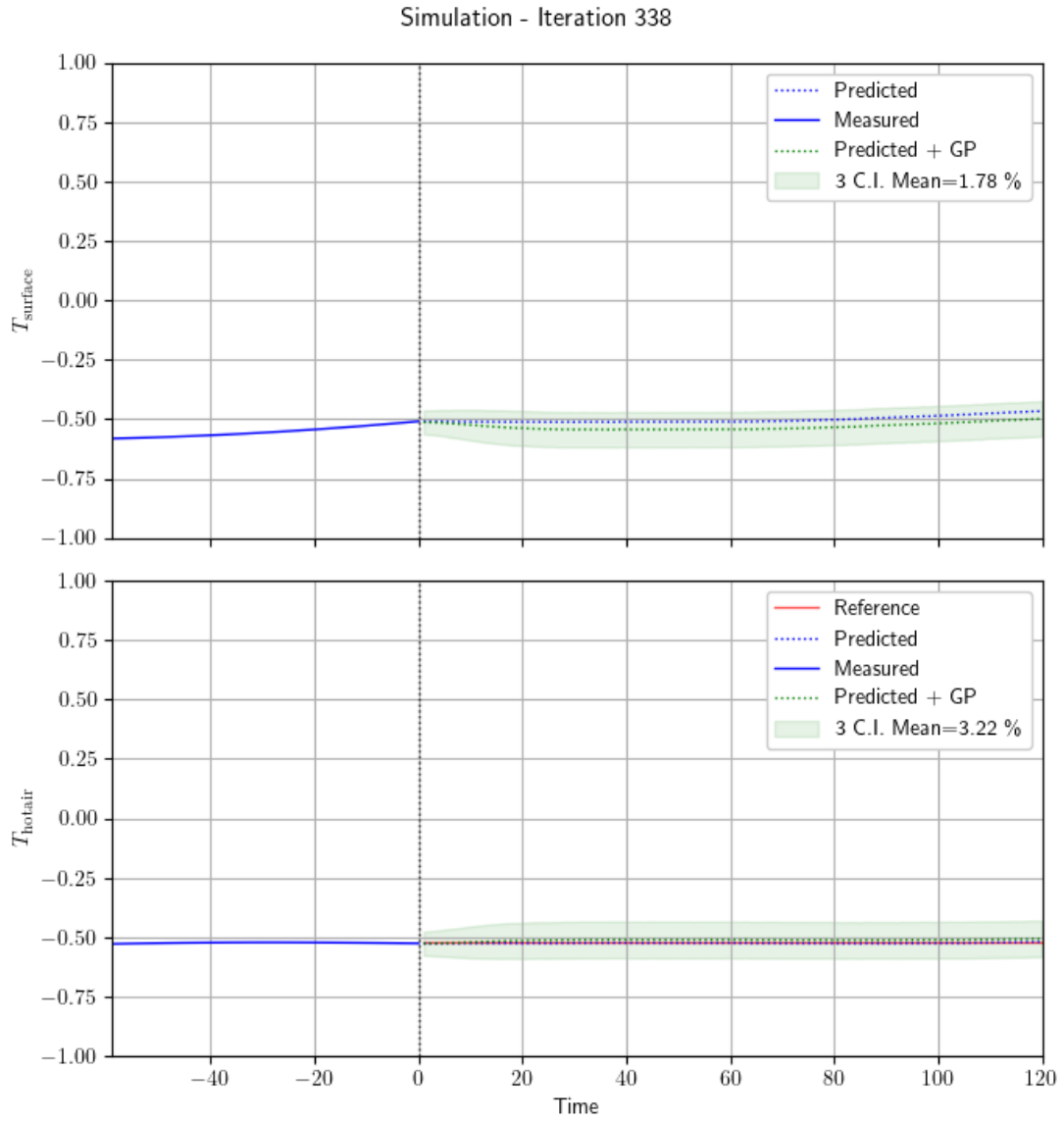


Figure 7.4: Narrow confidence interval at STJ test scenario nominal operation 1 time $t = 338$ with input u and MPC sliding window. T-NN model (Top). T-NN model with GP model error model regressor and $\pm 3\sigma$ confidence intervals (Bottom). Note that due to low prediction error, μ_{GP} is close to zero and the difference between the two models small.

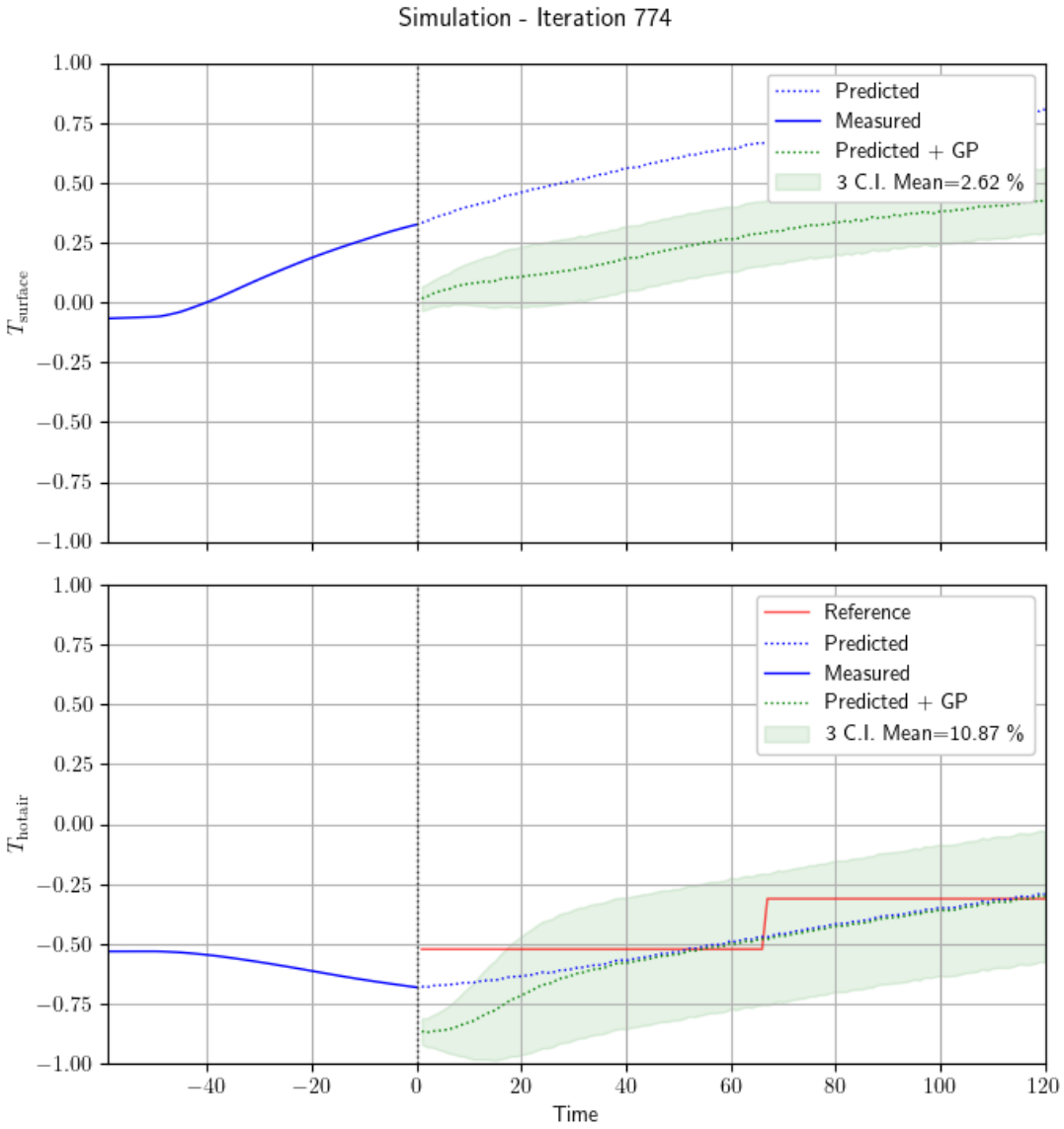


Figure 7.5: Wide confidence interval at STJ test scenario nominal operation 1 time $t = 774$ with input u and MPC sliding window. T-NN model (Top). T-NN model with GP model error model regressor and $\pm 3\sigma$ confidence intervals (Bottom). Note that due to higher prediction error, μ_{GP} contribute to the NN predictions while the confidence intervals is wider suggesting lower certainty.

investigations are suggested to mitigate these limitations and examine further applications of UQ within control systems.

Chapter 8

Conclusions

This thesis explores the integration of Model Predictive Control (MPC) with Transformer Neural Network (T-NN) to model and control the Solar Tower Power Plant Jülich (STJ). The aim was to increase the efficiency of Concentrated Solar Power (CSP) plants. This section evaluates the extent to which the study has achieved its goals. As mentioned in the Introduction 1, the lateral focus is on meeting the global need for sustainable Renewable Energy Sources (RESs). Furthermore, we present an analysis of the findings' relevance and potential impact.

Artificial Neural Networks

The present work was designed to explore the use of novel NNs to learn system dynamics. Motivated by the limitations of simplified models [25], [32], a data-based approach using T-NN was proposed [39]. Additionally, this work aimed to explore the potential of employing NN dynamic models with an MPC controller, rather than optimizing results for each system individually.

The T-NN was trained to predict the system's future states based on historical data. This study demonstrated the importance of data integrity and its impact on the performance of data-based models. During the training phase, measurement noise in one of the states limited the NN's ability to learn the system's dynamics. By applying filtering techniques, the model's predictions were significantly improved, highlighting the importance of data preprocessing and filtering to ensure the quality of the training data.

Interestingly, no clear minimum was identified in the model's prediction performance when exploring the hyperparameter space of the look-back window and prediction horizon. In order to select a suitable configuration, validation MSE values were ranked based on the metric at the lowest value of the prediction horizon. This result showed that training for longer prediction horizons drives the learning process to have better performance for the first few steps in the study case presented.

Another important contribution was the addition of a last-layer residual connection in the T-NN, which was absent in the work by Park et al. [39].The results demonstrated enhanced prediction accuracy when compared to the model without last layer residual connection.

These results suggest that a NN with an attention mechanism, as introduced by Vaswani et al. [38], can be used to model the dynamics of complex systems. The T-NN model was able to learn the system dynamics and predict future states within the region of validity. This opens up the possibility of using advanced NN architectures with modern control techniques to further improve the performance of CSP plants.

Model Predictive Control

The MPC controller was selected due to its ability to handle complex systems and MIMO configurations. Significantly, this controller enabled the system to operate under constraints and incorporated predicted disturbances in the control strategy. Focusing on the application in CSP plants, the use of MPC with the STJ receiver system was demonstrated. The key contribution was the use of T-NN to model the system dynamics. This thesis has demonstrated that NN models can be used for MPC in the STJ.

In the current study, a comparison was made between constrained and unconstrained optimization with barrier functions, which yielded interesting results. The former, implemented in Python's SciPy library as shown by Park et al. [39] and Jung et al. [37], served as a reference for an optimal control solution. The primary objective was to evaluate the performance of the MPC controller using different optimization algorithms such as Adam and L-BFGS. It was demonstrated that the unconstrained optimization approach was able to handle bound constraints using barrier functions with tuned parameters. This approach exhibited significantly lower computational effort and superior performance in simulation tests compared to the constrained optimization approach. These findings need to be confirmed for more testing scenarios and other system dynamics.

It is noteworthy that in all proposed test scenarios, both approaches successfully utilized a T-NN model to predict the system dynamics and control the system. The results demonstrated that the MPC controller was capable of tracking the set points and maintaining the system within operational bounds near constraints and during nominal operation scenarios. Another important finding was that the Multi Step-Ahead (MSA) prediction capabilities of the T-NN model were leveraged by the MPC controller to reject predicted disturbances. This was demonstrated in scenarios with simulated varying solar irradiance. However, the observed model dynamic deviation at the end of this scenario highlights the need for further research. The prediction accuracy was

observed to decline outside the domain of validity of the training data, indicating the necessity for the acquisition of more real plant data and model tuning.

The results on the PyTorch-based MPC have important implications for developing of real-time control applications. In our study, the computational time was found to be significantly reduced in comparison to the SciPy-based MPC. This difference could be attributed to the use of different tools for gradient computation, which constitutes an important area for future research.

An additional objective of this work was to explore the impact of measurement noise on the model dynamics and the MPC controller. In the STJ case study simulation results showed that the system remained controllable and was able to track the reference set points even under higher noise levels than typical industrial instrumentation standards. However, the presence of noise in the measurements led to a deterioration in the performance of the controller and an increase in the tracking error. Although this was explored with a single case study, caution must be applied, as the findings might not be transferable to all measured variables and regions of operation. Therefore, a more robust analysis is required.

Uncertainty Quantification

With these results, it can be concluded that the addition of a Gaussian Process (GP) has potential to overall enhance the prediction accuracy of the T-NN by uncertainty information. What stands out is the matching confidence interval calculation when compared to simulated true data, confirming that the probabilistic information provided by the covariance of the Gaussian Process Regressor (GPR) can be used. This suggests that while the GP model error model did not improve overall accuracy of the NN model, it was capable of quantifying the model prediction uncertainty, which is valuable for future work.

Using the example proposed by Park et al. [39] while extending the architecture and methodology, we were able to showcase the applicability of the Transformer Neural Network (T-NN) to learn and simulate nonlinear system dynamics. The model was then used with real-world data from the Solar Tower Power Plant Jülich (STJ) to model Concentrated Solar Power (CSP) system behavior. The importance of tuning parameters is presented, and the model is validated in simulation. Furthermore, the potential of novel neural network architectures such as the ”Transformer” is demonstrated, illustrating the possibility of using it in conjunction with Model Predictive Control (MPC) to enhance the efficiency and reliability of Concentrated Solar Power (CSP) plants.

Results from this work contribute to address global challenges like climate change by enhancing the efficiency and reliability of Renewable Energy Sources (RESs) tech-

nologies such as CSP plants[4]. When energy storage is considered, the implications can be extended and economic viability enhanced [2], [8]. This contributes to narrow the gap in the field of Renewable Energy Sources (RESs) model and control, and aims to serve as an example to further motivate increasing investments in scientific research.

Chapter 9

Outlook

In light of the scope of this thesis, areas for future research can further deepen on the findings. Potential topics for future investigation include topics on data pre-processing and broader test scenarios, alternative control methods and NN architectures, results validation, optimization, MPC, UQ and real plant validation.

Data pre-processing and test scenarios

Further experiments exploring lower data-sampling rates could shed more light on model prediction accuracy, control performance, and the computational cost of Optimal Control Problems (OCPs). Extending in model testing could be conducted by incorporating fast and more dynamic disturbances. These could include variable ramps and smooth transition, among other possibilities. An interesting open field rely on the controller response in the presence of inaccurate disturbance predictions.

Alternative control methods and neural network architectures

A comprehensive review of alternative controllers such as PID and Linear Quadratic Regulator could provide valuable insights. Additionally, a detailed comparison with reinforcement learning and other neural network architectures such as LSTM NNs could be pursued. Moreover, investigating replacing traditional multi-layer perceptrons with Kolmogorov-Arnold Networks (KANs) [121] presents and interesting field of study.

Results validation

Future studies might include a full discussion on uncertainty propagation, domain of validity and stability, extending the findings of this work. If the research is to be moved forward, a better understanding of theoretical assumptions, noise disturbance and

model uncertainty needs to be developed with theoretical tools such as sensitivity analysis, among others.

Optimization

A greater focus on optimization tools could produce interesting findings that account more for constrained optimization in tensor-enabled architectures. Penalty methods, barrier functions, interior point methods implementation or constrained optimization algorithms in PyTorch could be investigated. More research using SciPy and PyTorch with shared gradient information is required to enable a systematic comparison. In addition, more information on the OCP ill-conditioning would assist on establishing a greater degree of accuracy on this area.

Model predictive control

The incorporation of an aim-point and solar irradiation control system is an intriguing one which could be usefully explored in further research. Moreover, considerably more work will need to be done to determine the applicability of NN models with uncertainty quantification in MPC. A natural progression of this work is to analyze a Stochastic and Robust MPC formulation, as well as the economic formulation of the objective function. Elements in the MPC formulation such as terminal cost, terminal set, a terminal control law and recursive feasibility are presented as of particular interest to proof controller stability.

Uncertainty quantification

Future research should aim to deepen the theoretical understanding of GPs to optimize their application in different modeling scenarios. An extended theoretical analysis will enhance understanding and improve the accuracy of NN models by leveraging UQ methods. Designed operation regimes that account for limitations on added UQ methods such as gradual setpoint, could contribute to increase the reliability of the method.
Another UQ approaches applied to MPC such as NN Bayesian last layer [60], sparse GP [54] or conformal prediction [122] offer opportunities for future research. The addition of UQ in the model can enable the controller to be formulated as Robust MPC. In particular, formulations such as Stochastic, Min-max, Tube-based, Scenario-based MPC could be investigated.

Real plant test validation, economic and environmental impact assessment

This research has thrown up many questions in need of further investigation. The precise effect of the proposed T-NN MPC in the real STJ plant remains to be tested. An extensive assessment of both economic and environmental impacts could provide valuable insights to evaluate the advantages associated with the proposed MPC system. This evaluation should incorporate analyses of cost-benefit scenarios as well as potential reductions in greenhouse gas emissions. Finally, trials beyond CSP plants could provide more definitive evidence on the findings.

Appendix A

Datasheets

A.1 Parameters for MPC testing scenarios

Scenario	Initial conditions						$T_{\text{hot air}}$ setpoint
	T_{surface}	$T_{\text{hot air}}$	$\dot{m}_{\text{air}}$	$T_{\text{cold air}}$	$I_{\dot{Q}}$	$\dot{\tilde{H}}_{\text{air}}$	
C.V. 1	-0.9034	-	0.5696	-1.0215	0.9174	1.3900	-
C.V. 2	-0.9034	-0.5221	0.5696	-1.0215	-	1.3900	**4.7484**
N.O. 1	-0.9034	-	0.5696	-1.0215	0.9174	1.3900	-
N.O. 2	-0.9034	-0.5221	0.5696	-1.0215	-	1.3900	-0.5221

Table A.1: Initial conditions for each test scenario

Scenario	Time variant	Step time variant value							Step duration	Scenario duration
		1	2	3	4	5	6	7		
C.V. 1	$T_{\text{hot air}}$	-0.5221	**-0.3112**	**0.1104**	**4.7484**	**-0.3112**	-0.7329	**-0.1847**	5 min	35 min
C.V. 2	$I_{\dot{Q}}$	0.9174	-0.4334	0.9174	2.2681	0.9174	-0.9737	0.9174	5 min	35 min
N.O. 1	$T_{\text{hot air}}$	-0.5221	-0.3112	0.1104	0.5320	-0.3112	-0.7329	-0.1847	15 min	105 min
N.O. 2	$I_{\dot{Q}}$	0.9174	0.2420	0.9174	-0.4334	0.9174	-0.9737	0.9174	15 min	105 min

Table A.2: Step time variant values for each test scenario

A.2 MPC parameters

	$T_{\text{hot air}}$		$\dot{m}_{\text{air}}$	$\dot{T}_{\text{surface}}$
	C.V.	N.O.		
Lower bound	-1.9977	-1.9977	-1.4723	-0.0221
Upper bound	-0.3112	1.3753	4.3616	0.0221

Table A.3: Bound constraints for each test scenario

Barrier function	**Parameters**	**Value function** v					
		v_1	v_2	v_3	v_4	v_5	v_6
Scaled softplus	T	0.010		0.001		0.010	
	β			5			
Exponential	T	1		1		0	
	γ	40		50		100	

Table A.4: Barrier functions parameters

L-BFGS		**Adam**		**AdamW**		**Yogi**	
Option	**Value**	**Option**	**Value**	**Option**	**Value**	**Option**	**Value**
lr	0.001	lr	0.01	lr	0.01	lr	0.025
history_size	10	amsgrad	True				
line_search_fn	strong_wolfe						
loss_threshold					1e-3		
controlled_variable_loss_threshold					1e-3		

Table A.5: PyTorch optimizer options

Table A.6: SciPy optimizer options

Trust-contr		**COBYLA**		**SLSQP**	
Option	**Value**	**Option**	**Value**	**Option**	**Value**
jac	2-point	rhobeg	1	jac	3-point
factorization_method	SVDFactorization	catol	2e-8	ftol	1e-3
initial_barrier_parameter	1e-2	tol	1e-4	maxiter	100
finite_diff_rel_step	1e-2	maxiter	1000	eps	1e-3
barrier_tol	1e-6				
initial_tr_radius	1.5e0				
xtol	1.5e0				
gtol	1e-2				
maxiter	100				

A.3 GP parameters

Table A.7: Kernel parameters for each case study

Case study	Kernel	Kernel(s) parameters	
FOPDT	Constant kernel	constant_value	1.0
		constant_value_bounds	(1e-5, 1e5)
			”fixed”
	RBF kernel	length_scale	1.0
		constant_value_bounds	(1e-5, 1e5)
			”fixed”
STJ	Constant kernel	outputscale_prior	1.0
		outputscale_constraint	Positive
	RBF kernel	length_scale	1.0
		eps	1e-6
		lengthscale_constraint	Positive

Appendix B

Graphs

B.1 Artificial Neural Networks

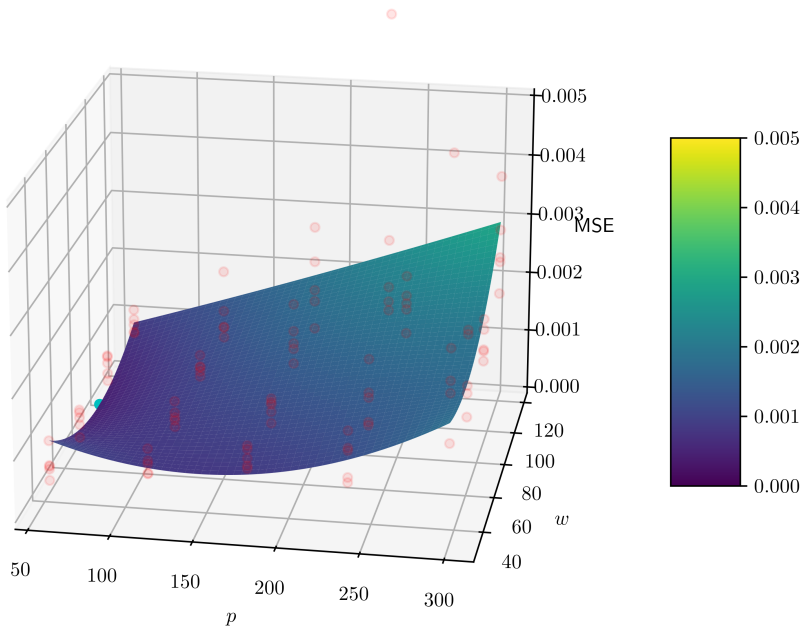


Figure B.1: Neural networks training mean squared error surface for w and p combinations ranked at $p = 60$

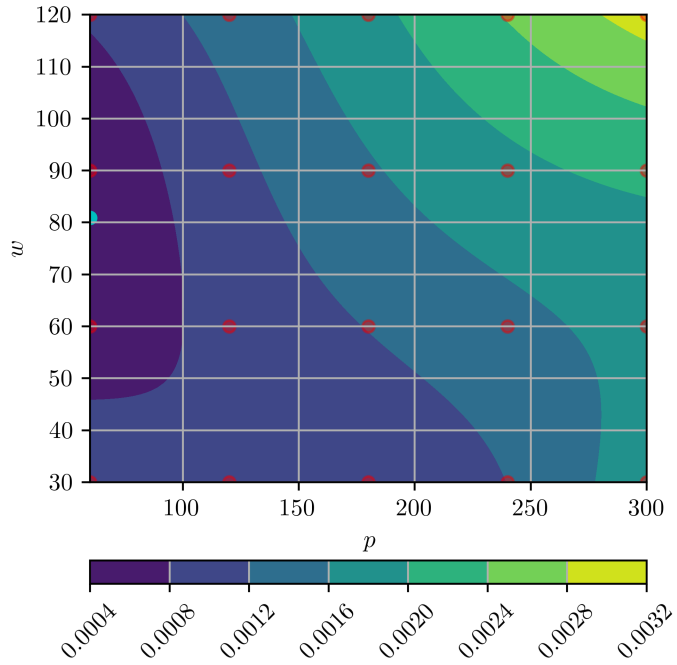


Figure B.2: Neural networks training mean squared error surface contour for w and p combinations ranked at $p = 60$

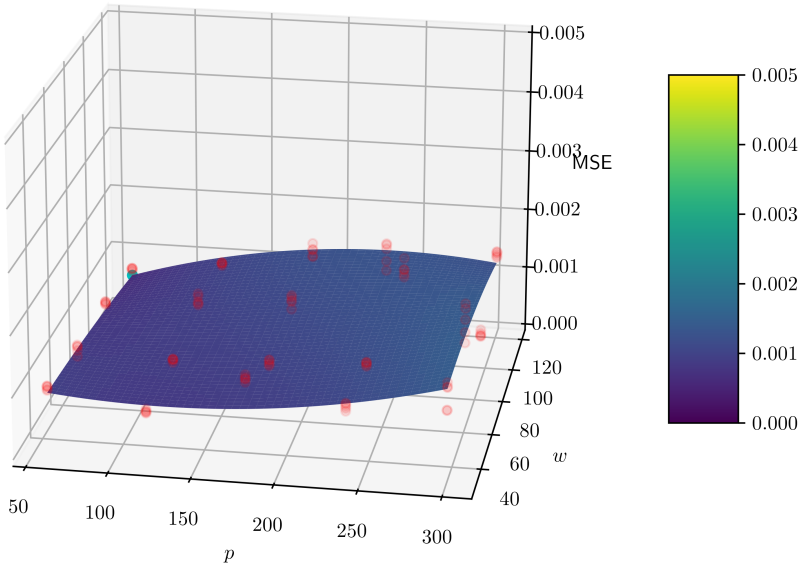


Figure B.3: Neural networks validation mean squared error surface for w and p combinations ranked at $p = 60$

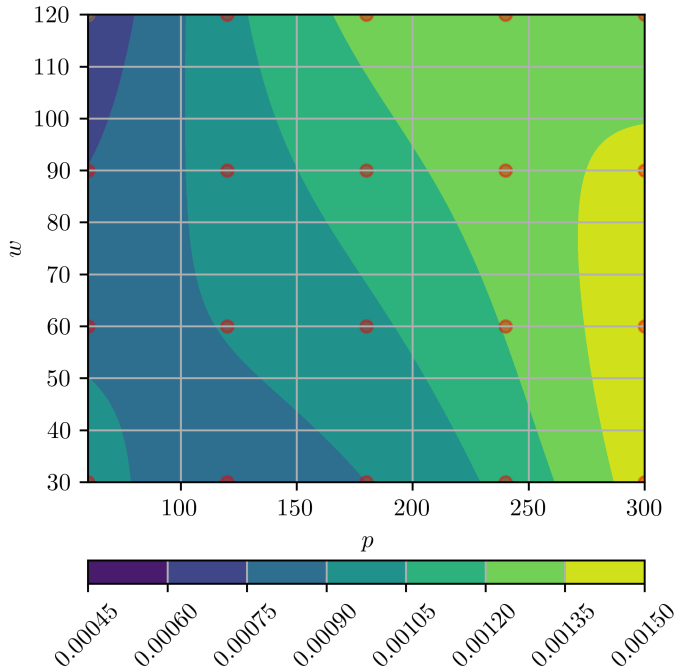


Figure B.4: Neural networks validation mean squared error surface contour for w and p combinations ranked at $p = 60$

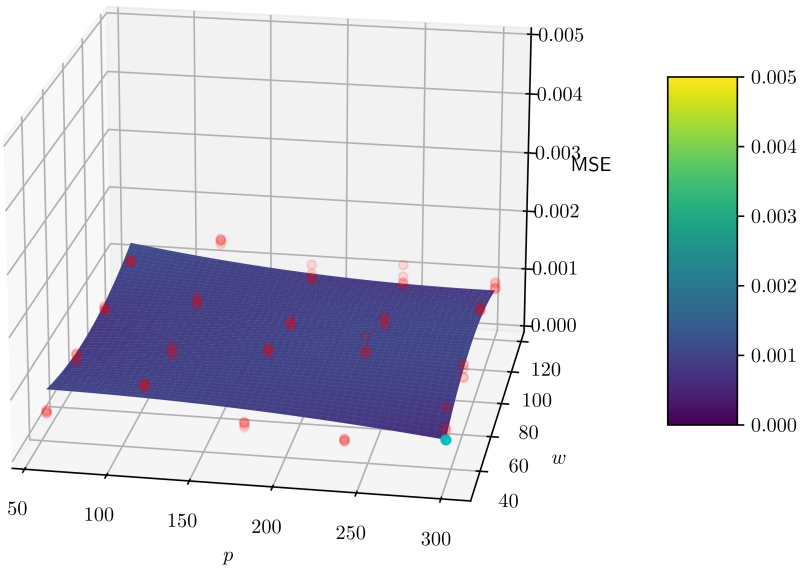


Figure B.5: Neural networks testing mean squared error surface for w and p combinations ranked at $p = 60$

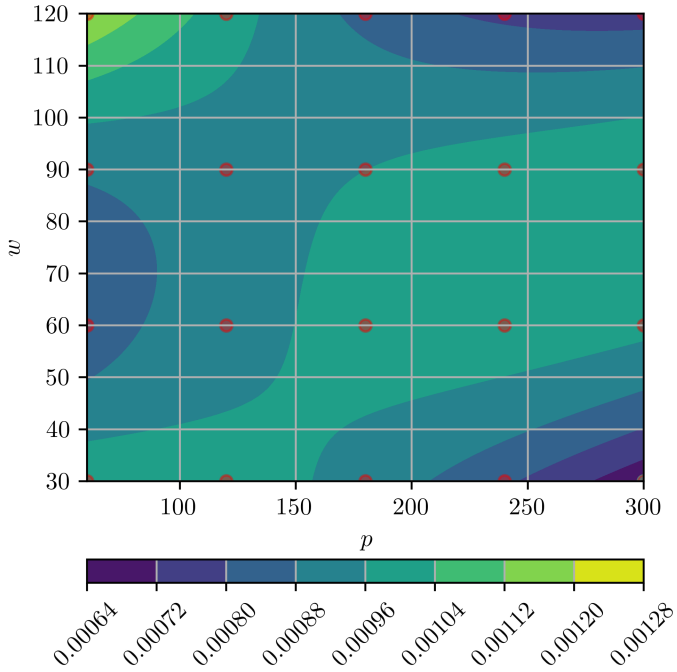


Figure B.6: Neural networks testing mean squared error surface contour for w and p combinations ranked at $p = 60$

B.2 Model Predictive Control

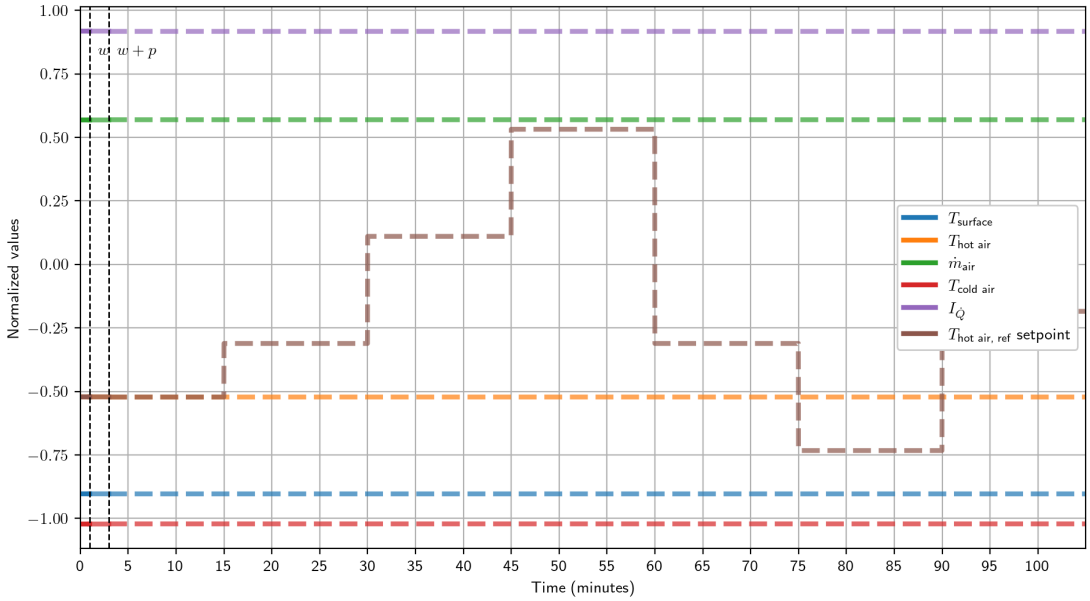


Figure B.7: Nominal operation test scenario 1. Hot air $T_{\text{hot air}}$ temperature tracking at constant solar irradiation $I_{\dot{Q}}$.

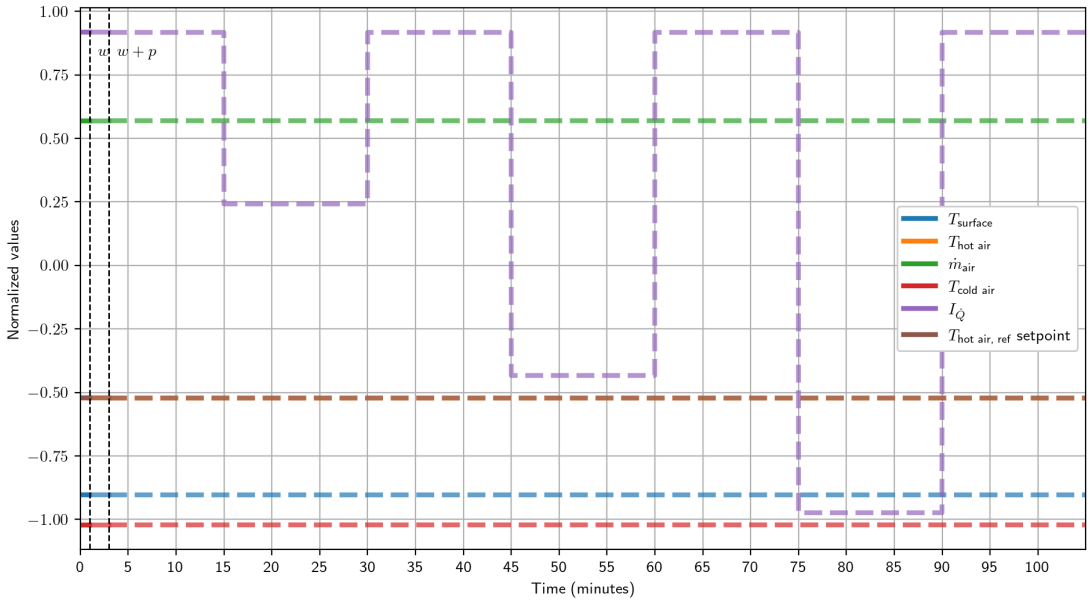


Figure B.8: Nominal operation test scenario 2. Hot air $T_{\text{hot air}}$ temperature tracking at variant solar irradiation $I_{\dot{Q}}$.

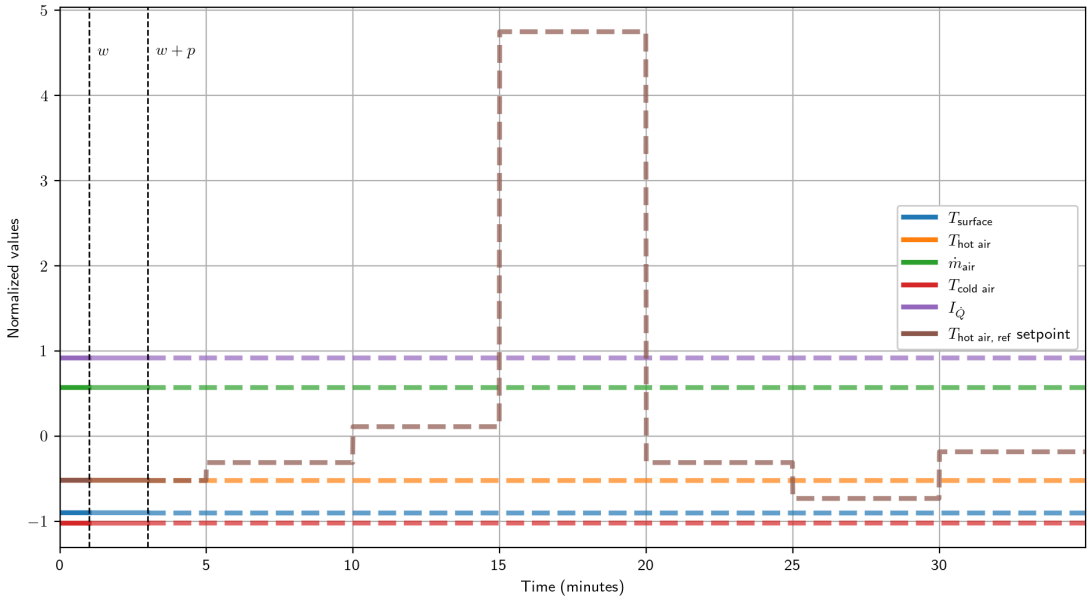


Figure B.9: Constraint violation test scenario 1. Hot air $T_{\text{hot air}}$ temperature tracking at constant solar irradiation $I_{\dot{Q}}$.

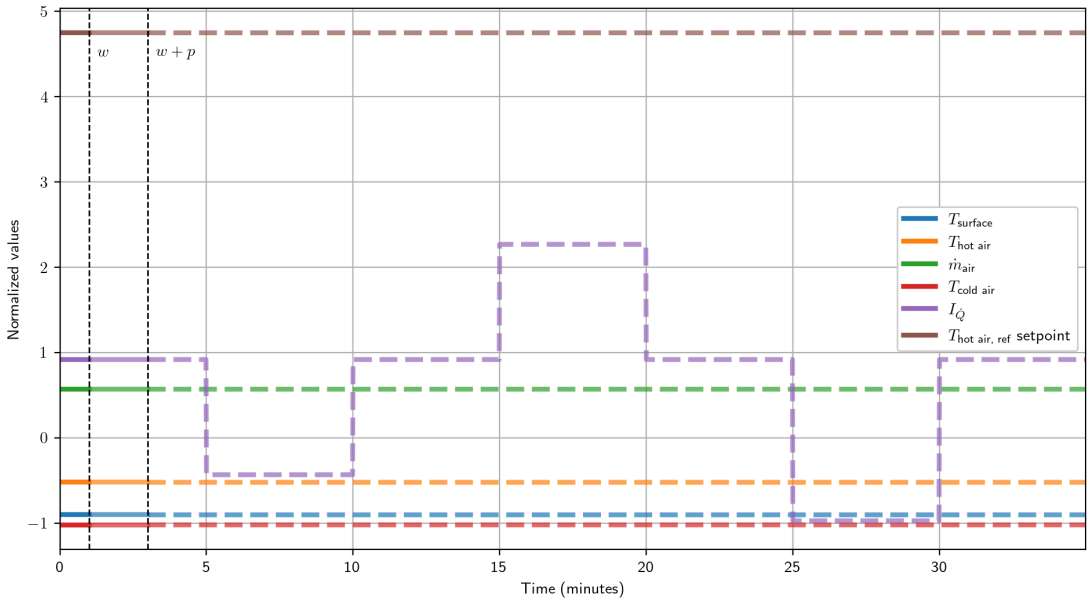


Figure B.10: Constraint violation test scenario 2. Hot air $T_{\text{hot air}}$ temperature tracking at variant solar irradiation $I_{\dot{Q}}$.

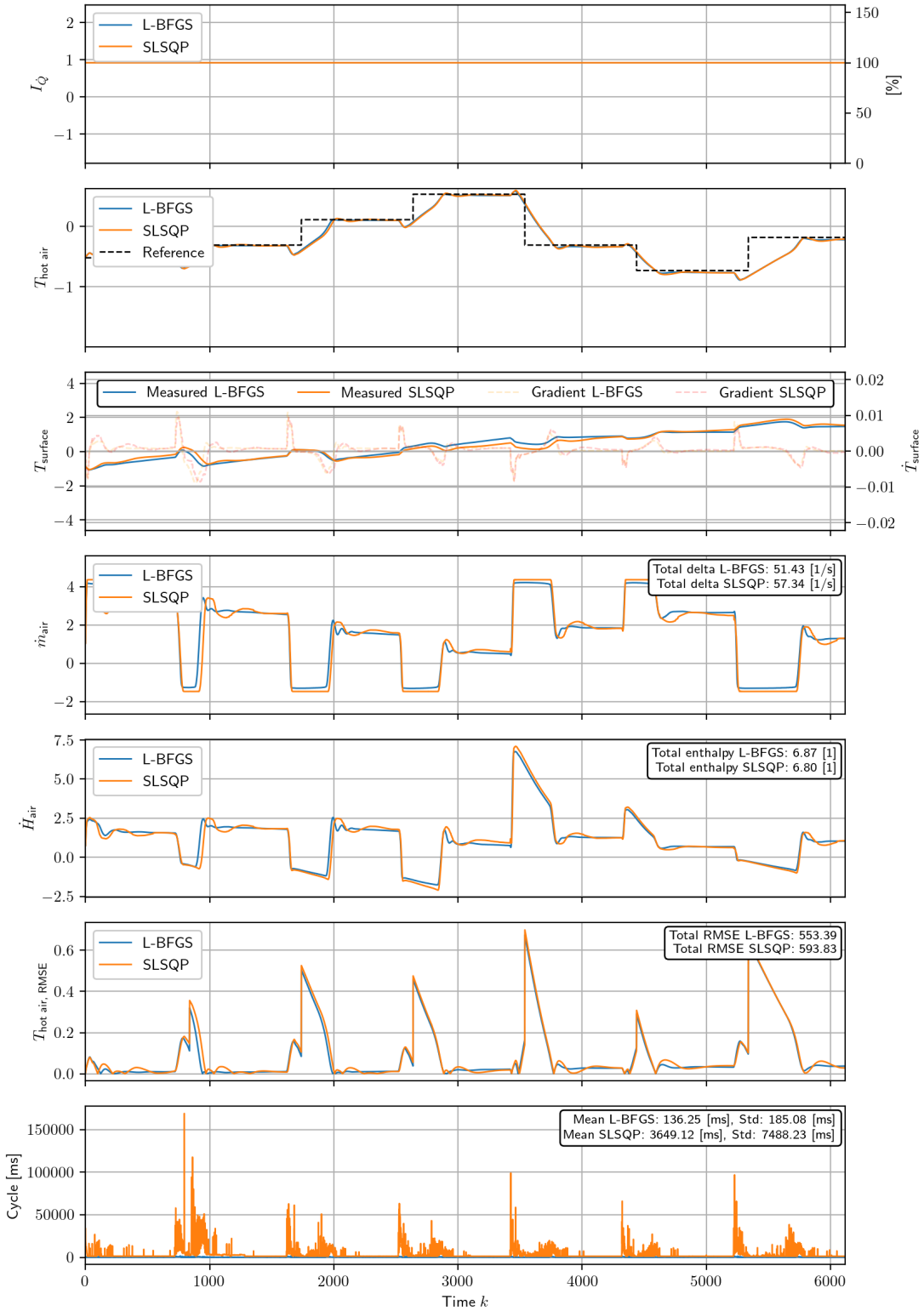


Figure B.11: Pytorch vs SciPy simulation results on test scenario nominal operation 1

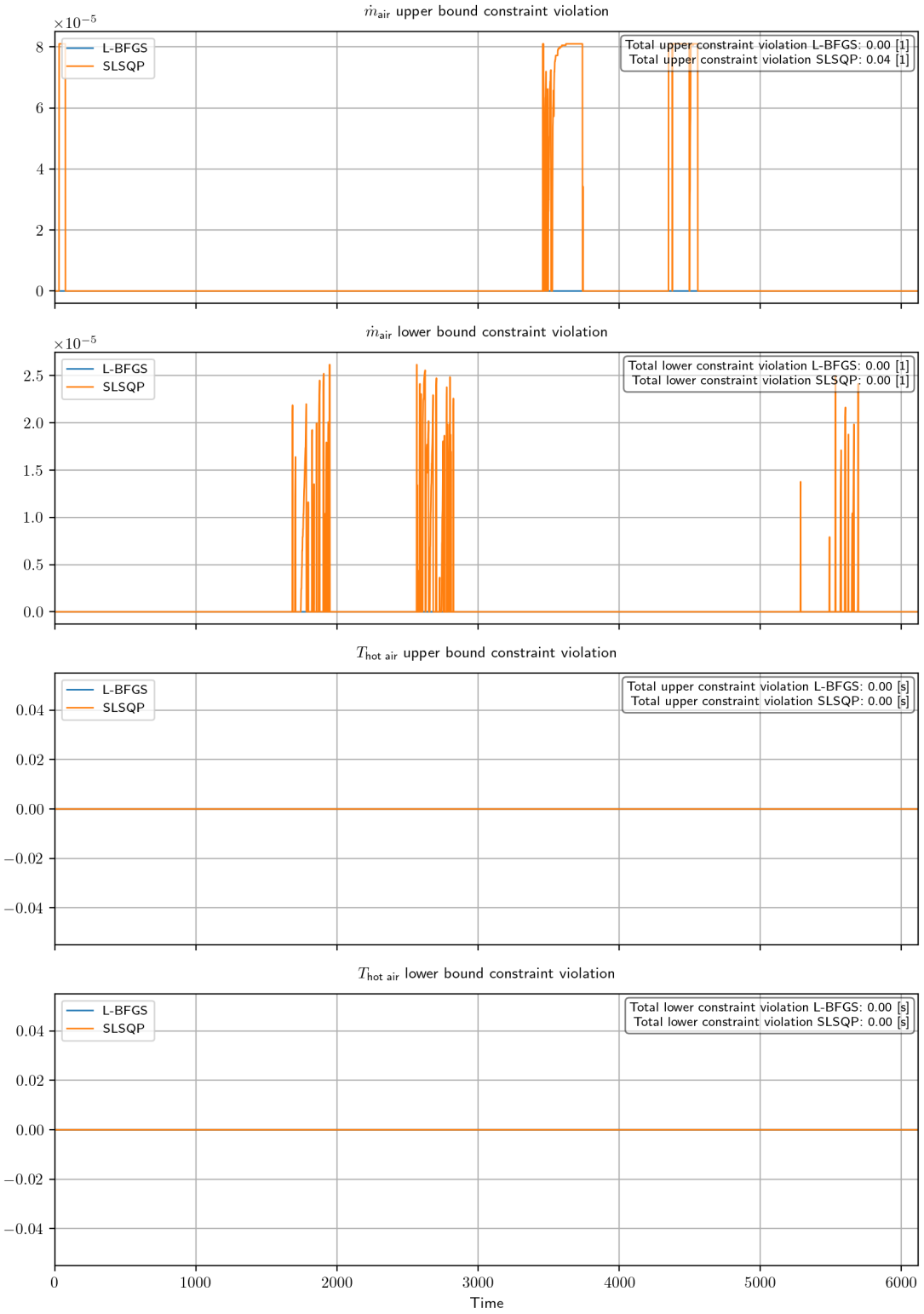


Figure B.12: Pytorch vs SciPy simulation constraint violation on test scenario nominal operation 1

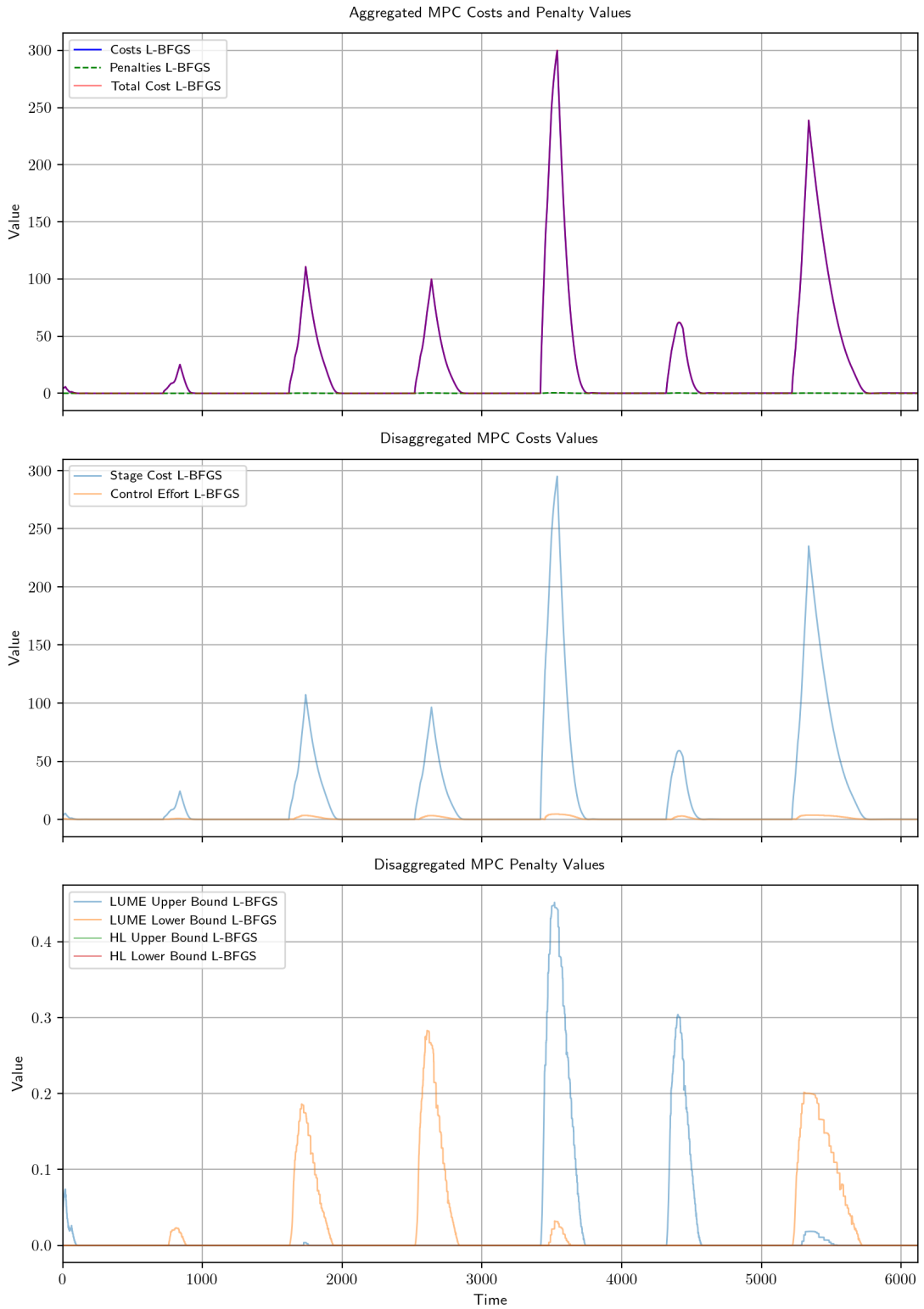


Figure B.13: Pytorch MPC costs on test scenario nominal operation 1

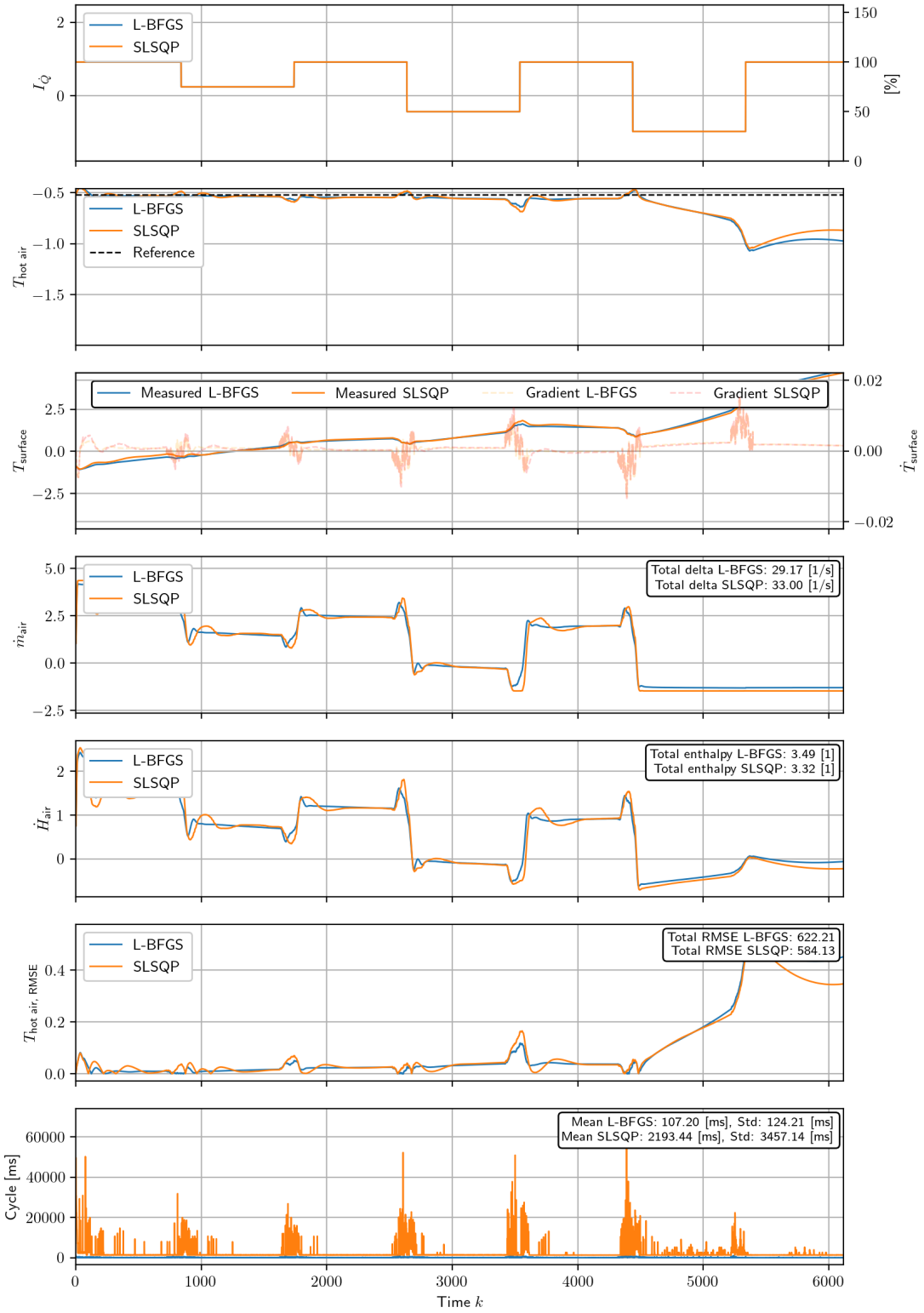


Figure B.14: Pytorch vs SciPy simulation results on test scenario nominal operation 2

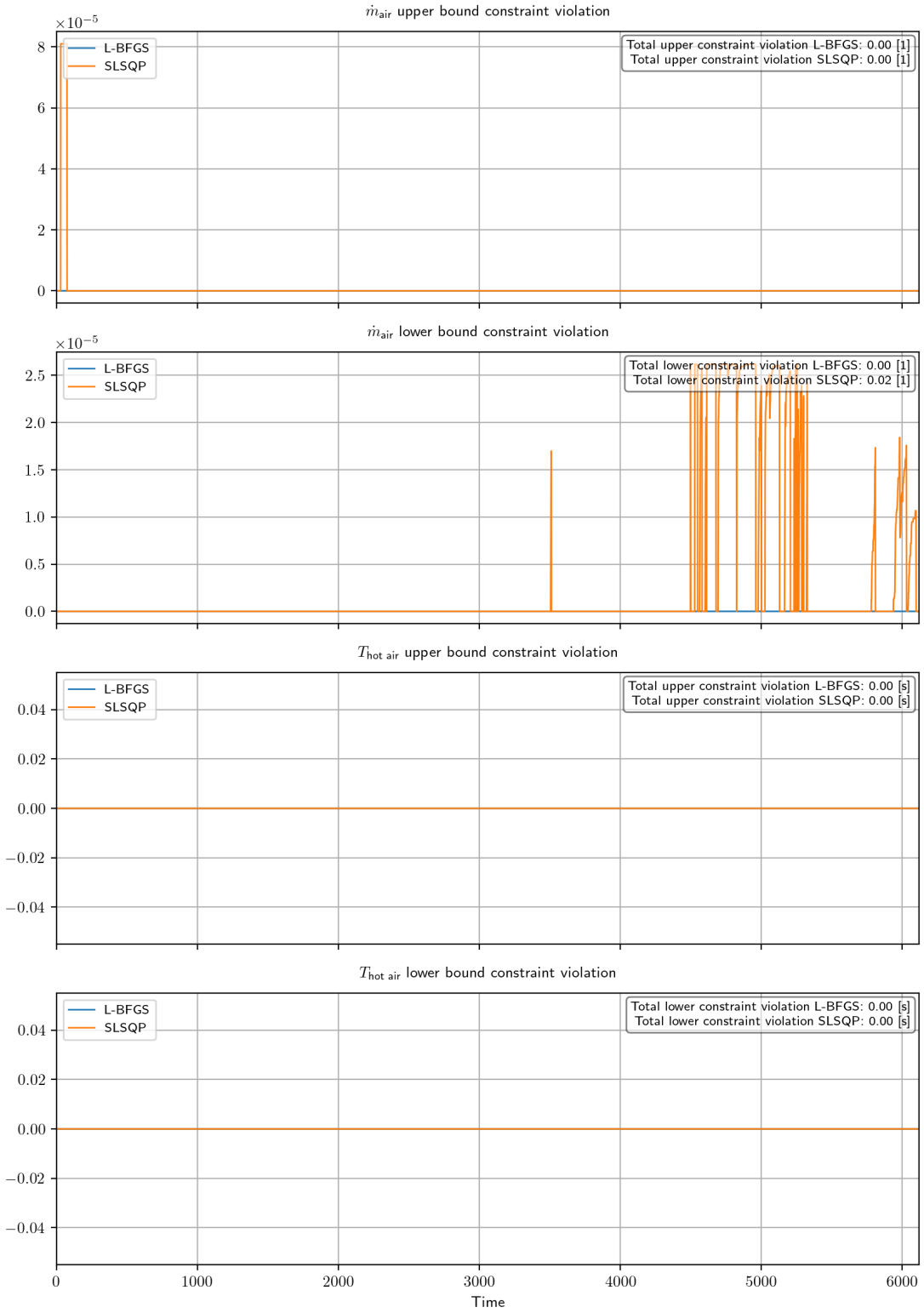


Figure B.15: Pytorch vs SciPy simulation constraint violation on test scenario nominal operation 2

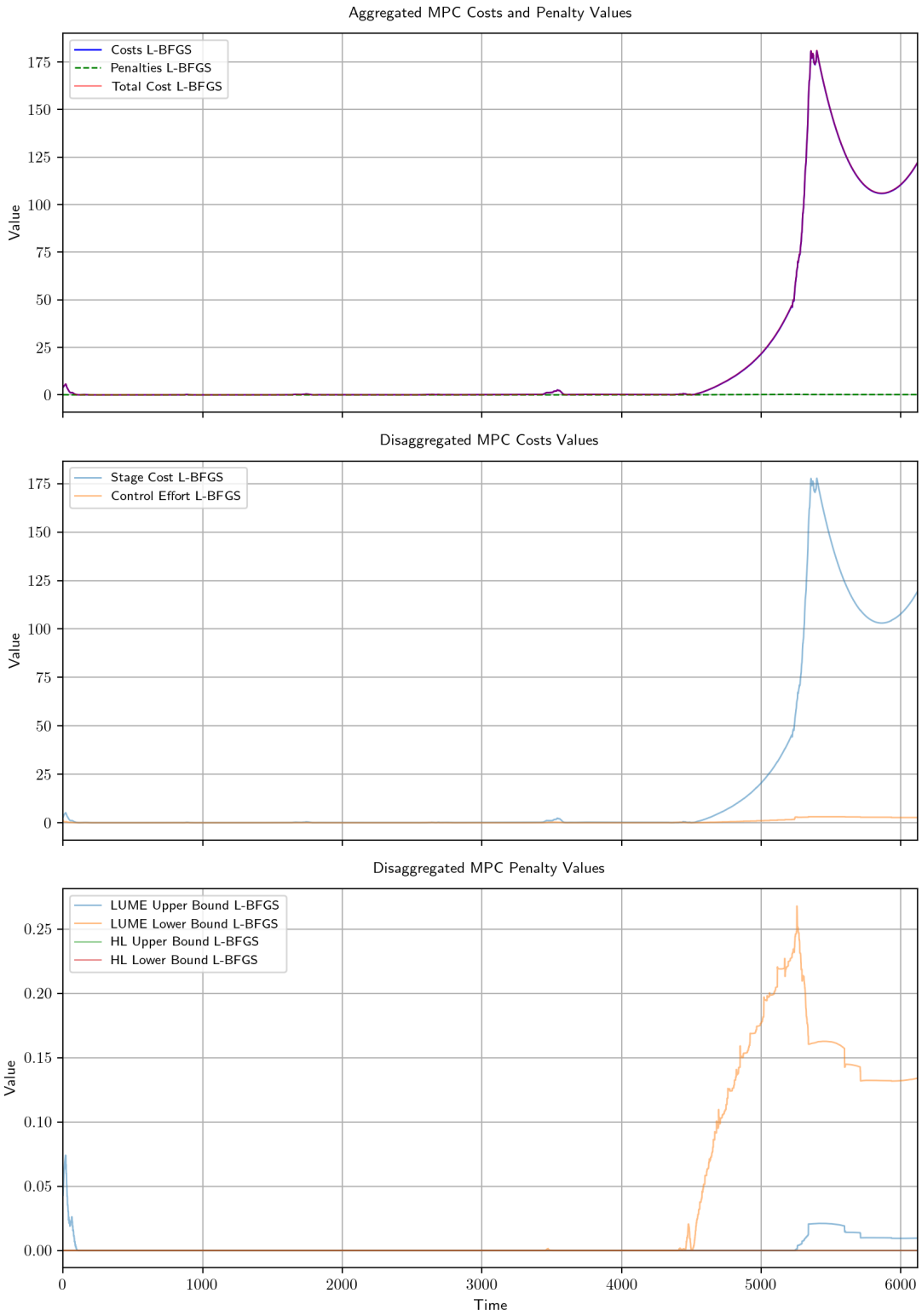


Figure B.16: Pytorch MPC costs on test scenario nominal operation 2

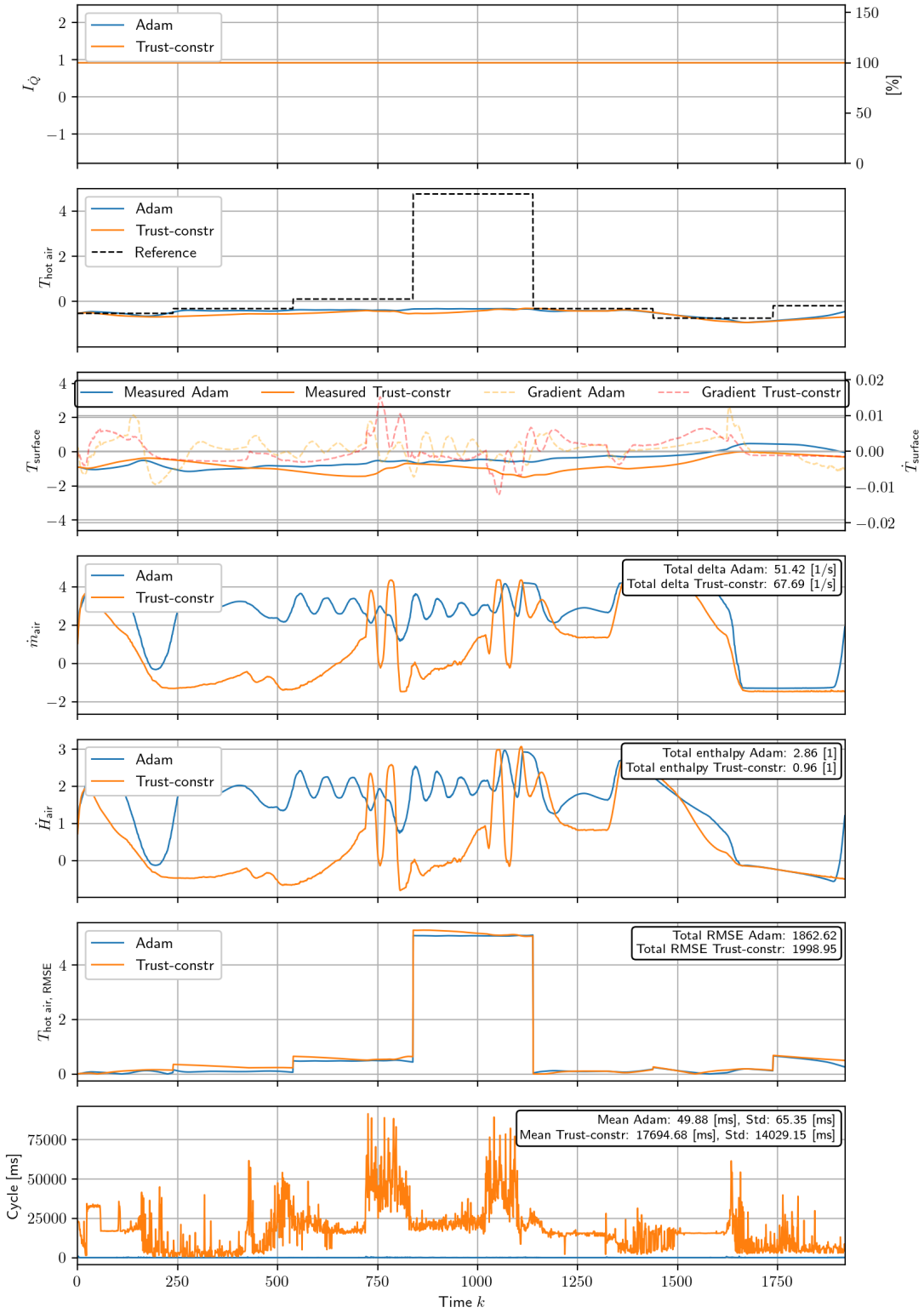


Figure B.17: Pytorch vs SciPy simulation results on test scenario constraint violation 1

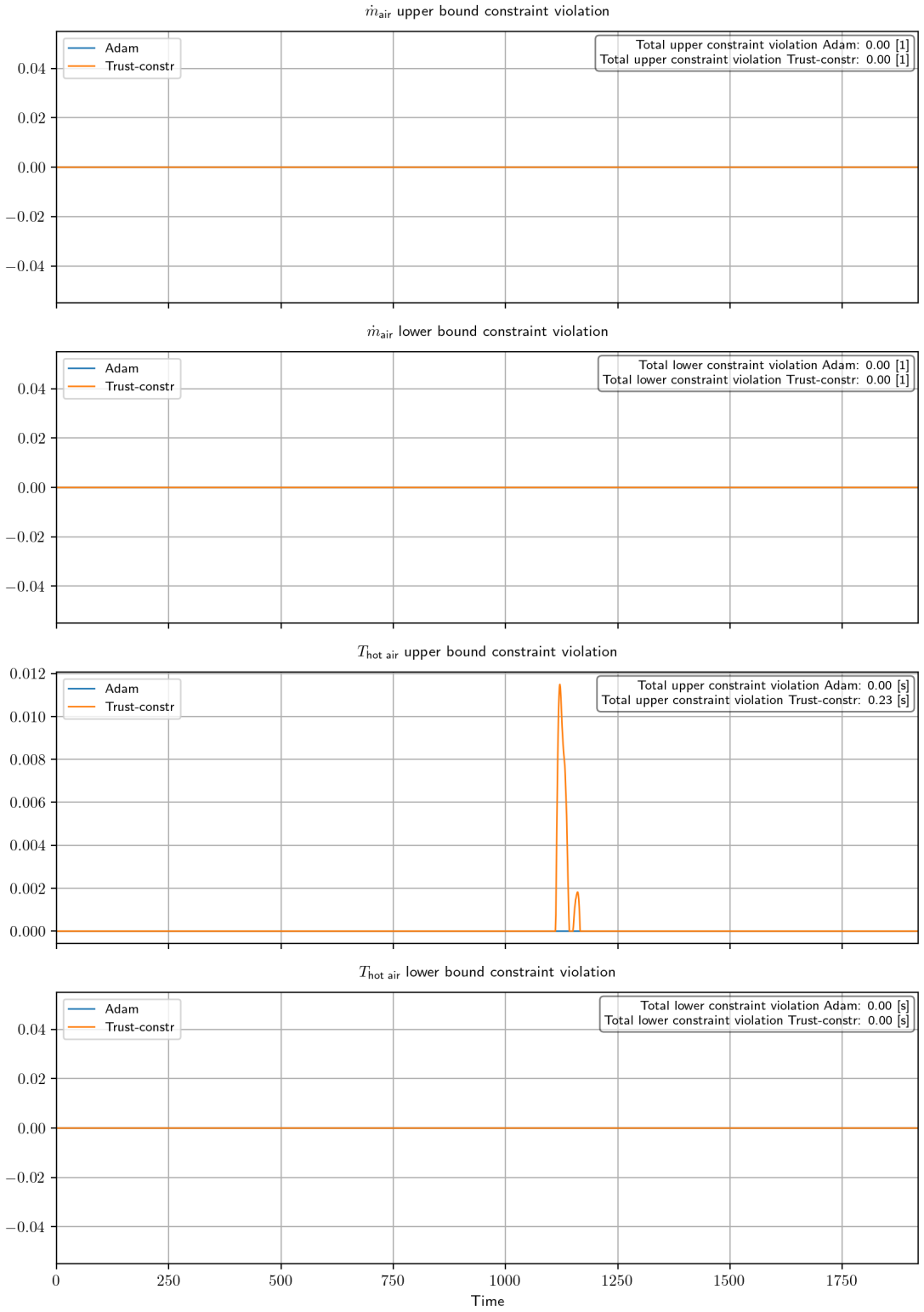


Figure B.18: Pytorch vs SciPy simulation constraint violation on test scenario constraint violation 1

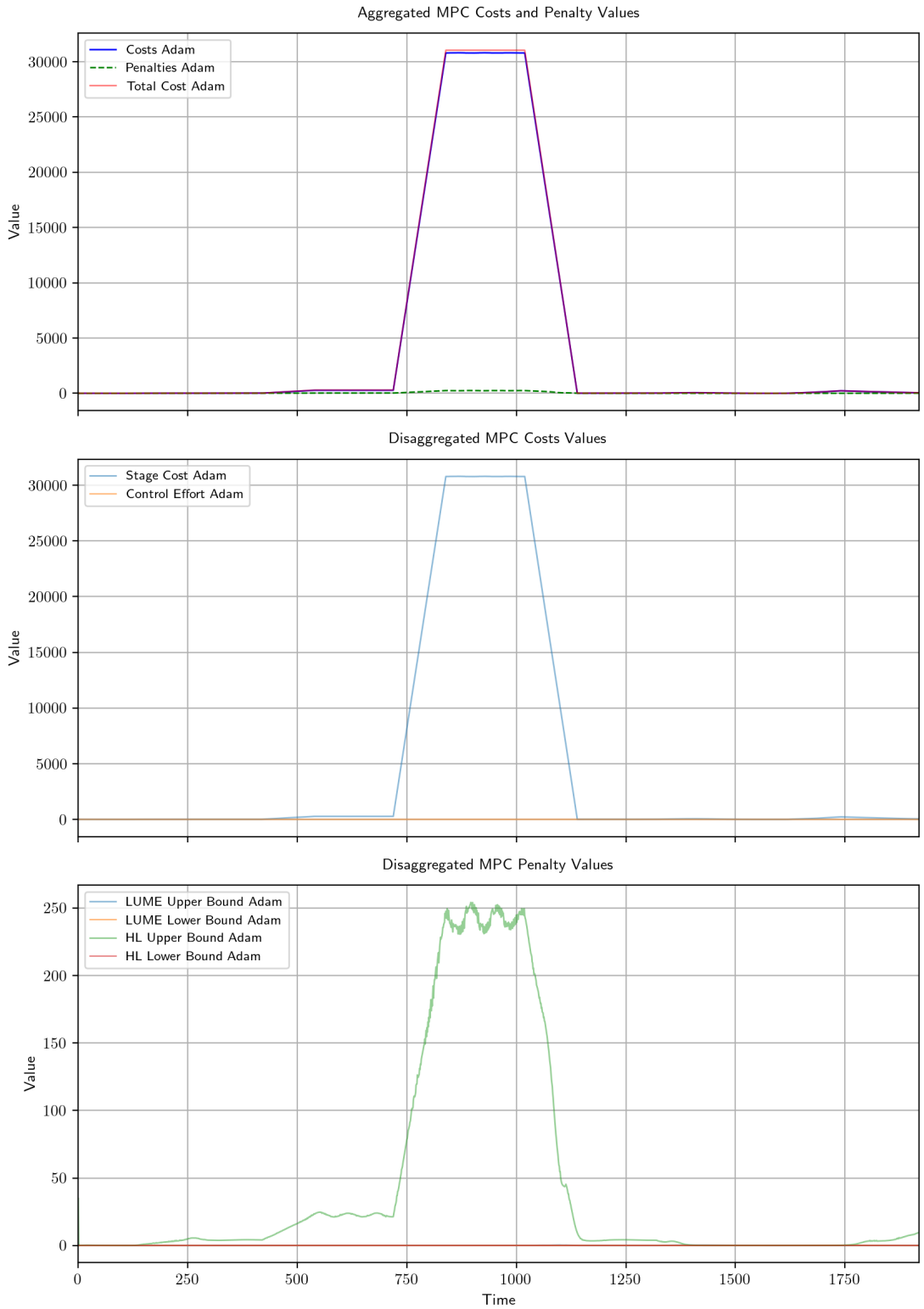


Figure B.19: Pytorch MPC costs on test scenario constraint violation 1

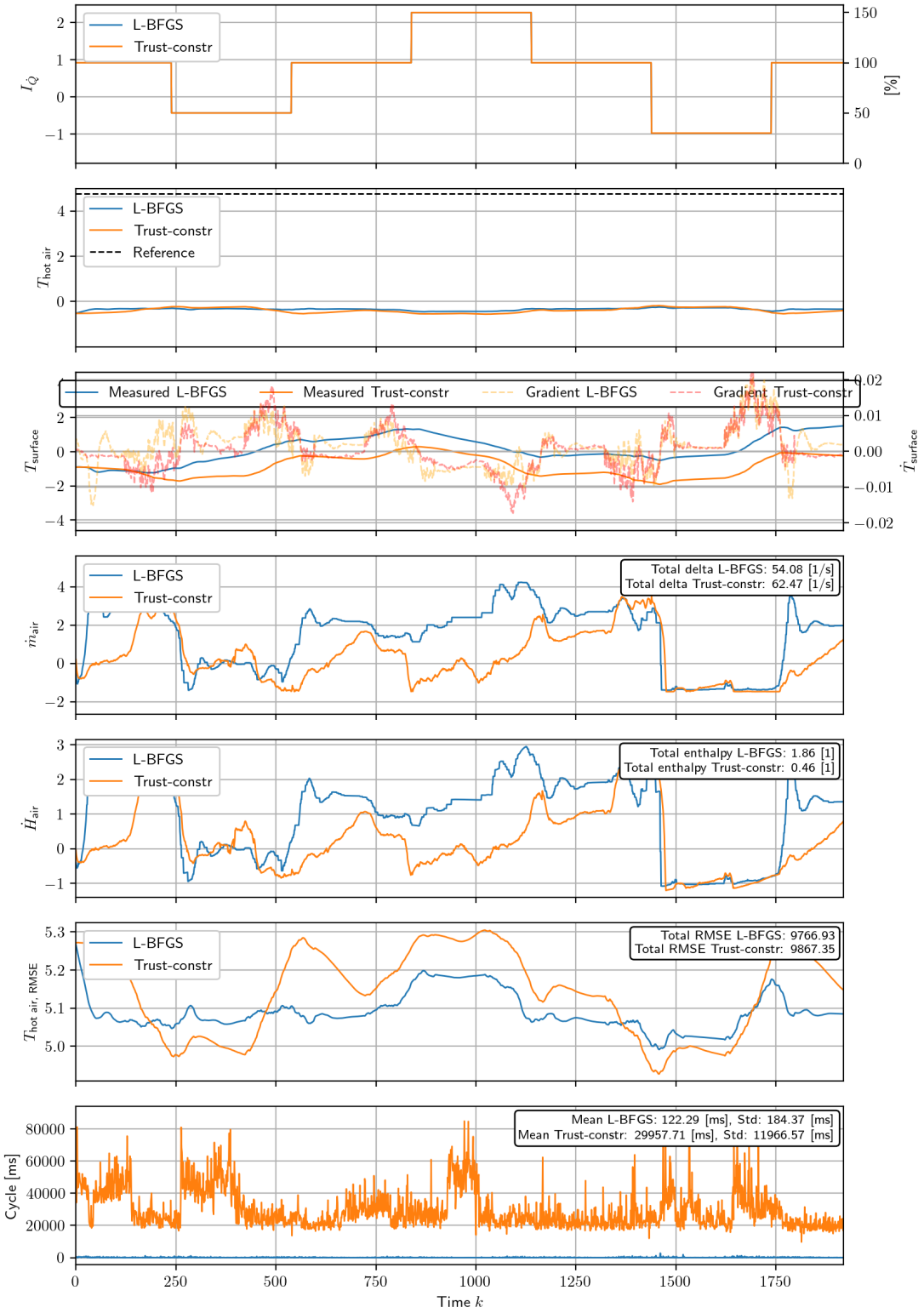


Figure B.20: Pytorch vs SciPy simulation results on test scenario constraint violation 2

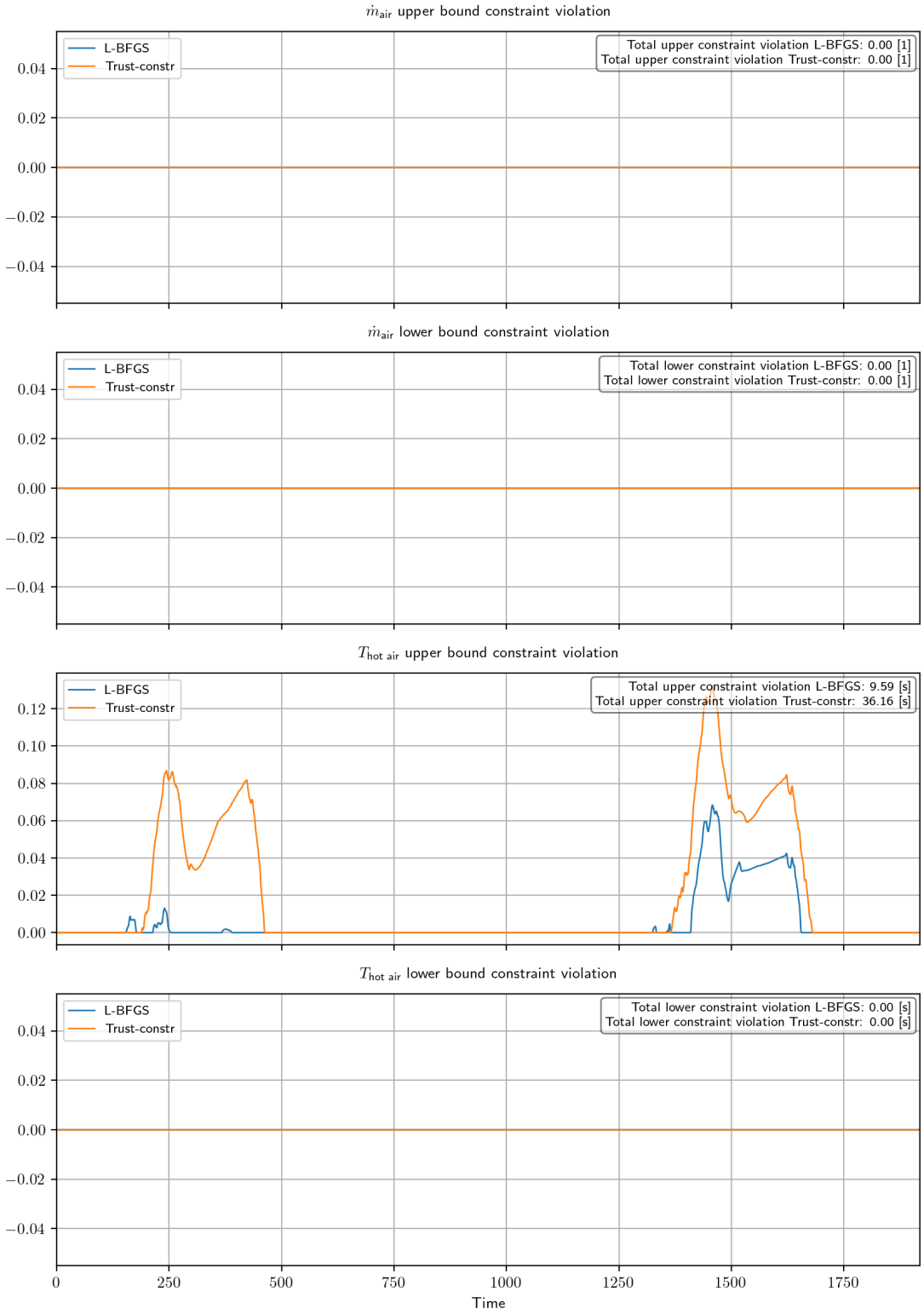


Figure B.21: Pytorch vs SciPy simulation constraint violation on test scenario constraint violation 2

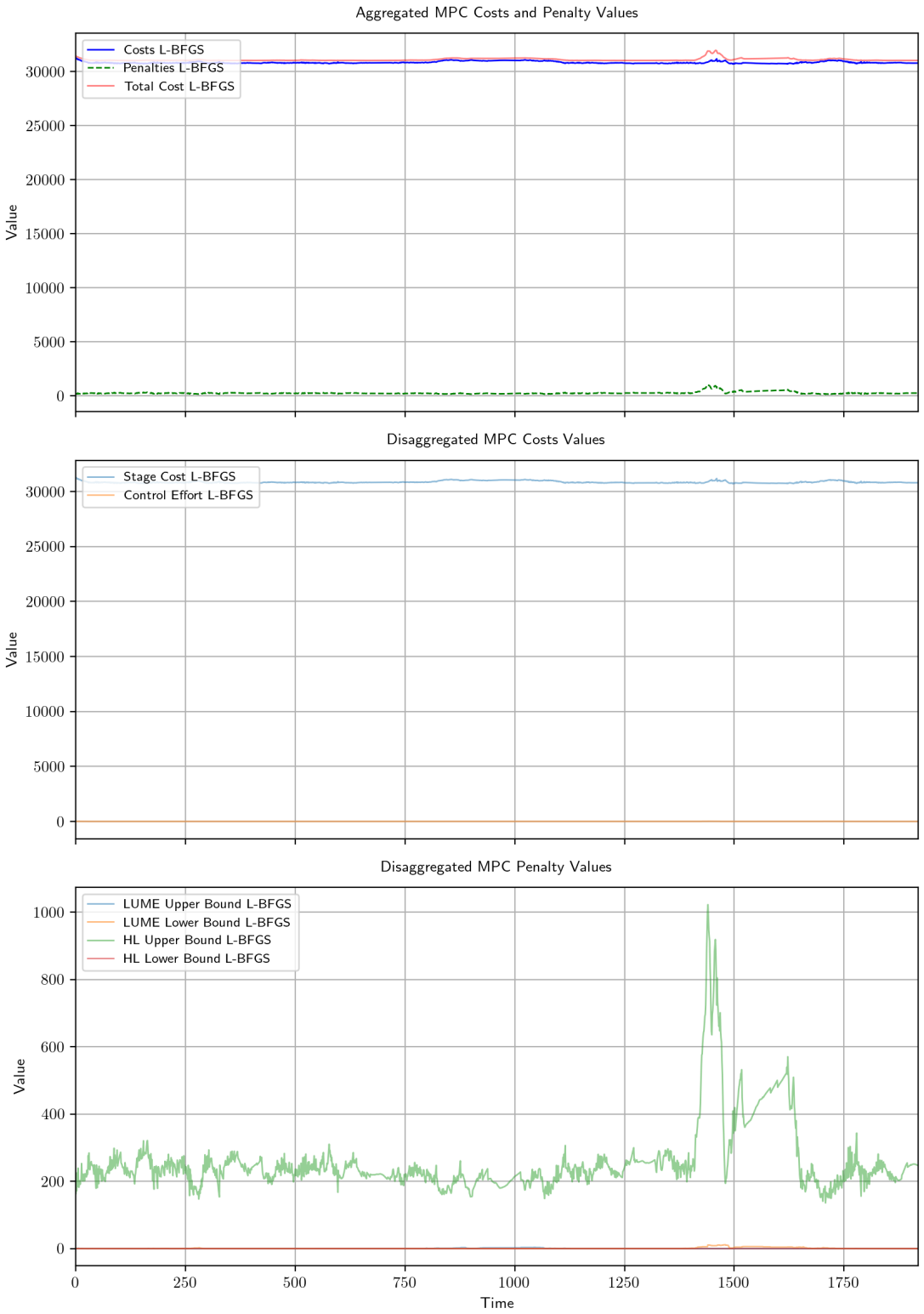


Figure B.22: Pytorch MPC costs on test scenario constraint violation 2

B.3 Uncertainty Quantification

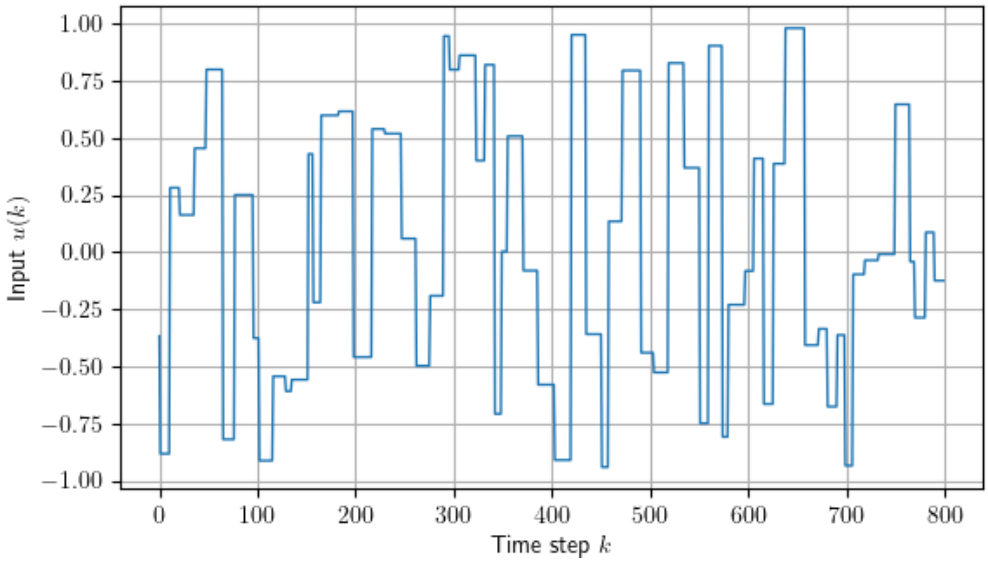


Figure B.23: Test scenario input u for NN predictions with GP model error model regressor

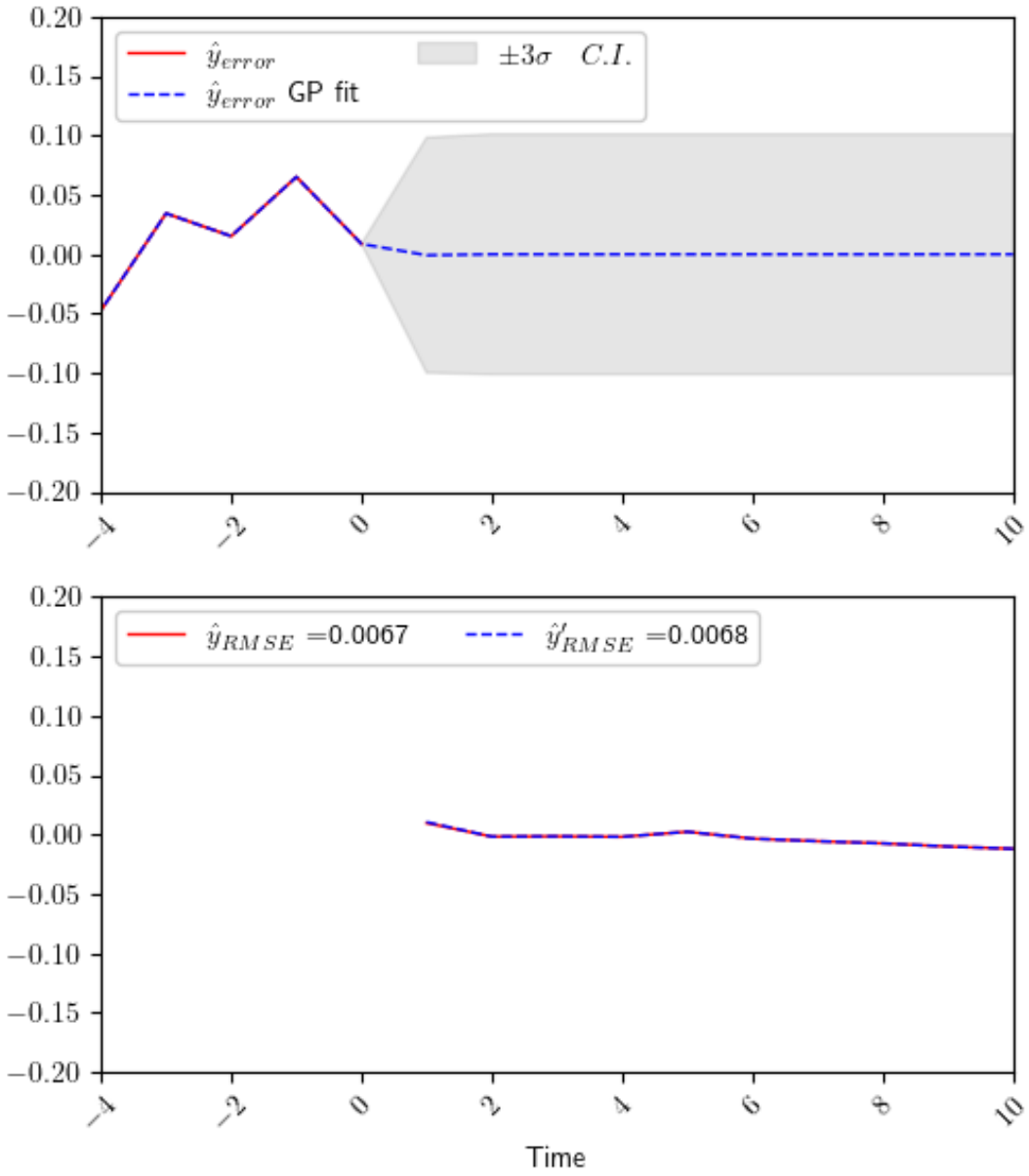


Figure B.24: PyTorch MPC costs on test scenario constraint violation 2

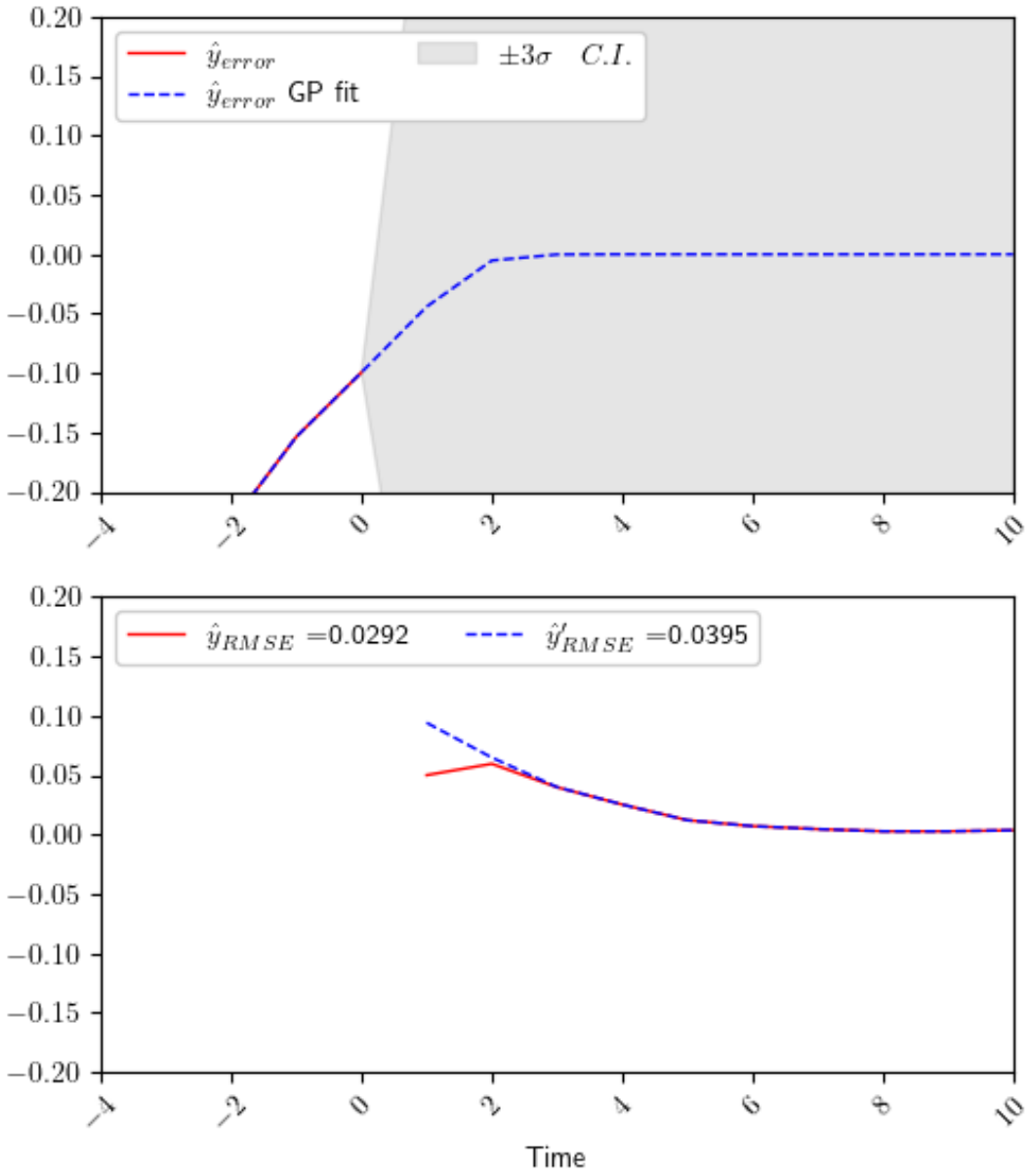


Figure B.25: PyTorch MPC costs on test scenario constraint violation 2

Appendix C

Tables

C.1 Artificial Neural Networks

Table C.1: Lowest 10 mean squared error training loss for w and p combinations ranked at $p = 60$. Filtered and denoised dataset.

NN	w	p	**Training MSE loss**					
			Sum	$T_{\textbf{hot air}}$	$T_{\textbf{surface}}$	$\text{Sum}_{p=60}$	$T_{\textbf{hot air}\ ,\ p=60}$	$T_{\textbf{surface}\ ,\ p=60}$
28	120	60	0.002366	0.003513	0.001219	0.001144	0.001906	0.000382
06	60	60	0.000435	0.000487	0.000383	0.000435	0.000487	0.000383
92	300	90	0.005318	0.006526	0.004109	0.001517	0.002115	0.000918
07	60	60	0.000852	0.001334	0.000370	0.000852	0.001334	0.000370
04	60	30	0.000543	0.000814	0.000271	0.000543	0.000814	0.000271
29	120	60	0.002382	0.002796	0.001969	0.000874	0.001177	0.000572
27	120	60	0.001374	0.001868	0.000880	0.000720	0.000985	0.000455
25	120	60	0.001965	0.003029	0.000901	0.000826	0.001301	0.000352
16	60	120	0.000926	0.001186	0.000665	0.000926	0.001186	0.000665
94	300	90	0.005970	0.007213	0.004726	0.001541	0.002048	0.001033

Table C.2: Lowest 10 mean squared error validation loss for w and p combinations ranked at $p = 60$. Filtered and denoised dataset.

NN	w	p	**Validation MSE loss**					
			Sum	$T_{\textbf{hot air}}$	$T_{\textbf{surface}}$	$\text{Sum}_{p=60}$	$T_{\textbf{hot air}\ ,\ p=60}$	$T_{\textbf{surface}\ ,\ p=60}$
28	60	120	0.001702	0.000831	0.002573	0.000741	0.000301	0.001182
06	60	60	0.000738	0.000308	0.001167	0.000737	0.000308	0.001167
92	90	300	0.005142	0.003787	0.006497	0.000714	0.000321	0.001107
07	60	60	0.000835	0.000322	0.001347	0.000835	0.000322	0.001347
04	30	60	0.001000	0.000324	0.001677	0.001000	0.000324	0.001677
29	60	120	0.001889	0.000895	0.002883	0.000801	0.000326	0.001276
27	60	120	0.001815	0.000958	0.002672	0.000781	0.000337	0.001225
25	60	120	0.001853	0.000943	0.002763	0.000798	0.000339	0.001257
16	120	60	0.000699	0.000351	0.001047	0.000699	0.000351	0.001047
94	90	300	0.006895	0.004835	0.008955	0.000815	0.000352	0.001277

Table C.3: Lowest 10 mean squared error training loss for w and p combinations ranked at $p = 60$. Filtered and denoised dataset.

NN	w	p	**Testing MSE loss**					
			Sum	$T_{\textbf{hot air}}$	$T_{\textbf{surface}}$	$\text{Sum}_{p=60}$	$T_{\textbf{hot air}\ ,\ p=60}$	$T_{\textbf{surface}\ ,\ p=60}$
28	120	60	0.002321	0.003647	0.000995	0.001021	0.001746	0.000296
06	60	60	0.000692	0.001064	0.000320	0.000692	0.001064	0.000320
92	300	90	0.005786	0.007517	0.004055	0.001187	0.001634	0.000741
07	60	60	0.000712	0.001116	0.000307	0.000712	0.001116	0.000307
04	60	30	0.000641	0.000943	0.000339	0.000641	0.000943	0.000339
29	120	60	0.002359	0.003593	0.001125	0.001034	0.001711	0.000357
27	120	60	0.002026	0.002937	0.001116	0.000891	0.001447	0.000334
25	120	60	0.002146	0.003263	0.001029	0.000946	0.001557	0.000335
16	60	120	0.000863	0.001284	0.000443	0.000863	0.001284	0.000443
94	300	90	0.007072	0.009616	0.004528	0.001213	0.001695	0.000730

C.2 Model Predictive Control

C.2.1 Softmax aggregated ranking results

Table C.4: Softmax aggregated ranking PyTorch vs SciPy results on test scenarios Constraint Violation (C.V.) 1 and 2 for all different optimizers. Constrained violations in $T_{\text{hot air}}$ observed. PyTorch MPC showing better performance compared to SciPy.

Scenario	**Library**	**Optimizer**	$\dot{m}_{\text{air}}$, U.B. C.V.	$\dot{m}_{\text{air}}$, L.B. C.V.	$T_{\text{hot air}}$, U.B. C.V.	Rank	$T_{\text{hot air}}$, L.B. C.V.	$\sum \dot{T}_{\text{surface}}$	Rank	$\sum \Delta \boldsymbol{u}$	Rank	$T_{\text{hot air}}$, RMSE	Rank	$\sum \dot{\tilde{H}}_{\text{air}}$	Rank	**Score**
Constraint Violation 1	SciPy	COBYLA	**1.00**	0.00	**0.12**	0.13	0.00	0.13	0.14	0.13	0.14	0.16	0.15	0.08	0.13	4.13
		SLSQP	0.00	0.00	**0.77**	0.26	0.00	0.18	0.15	0.18	0.15	0.14	0.14	0.14	0.14	4.93
		Trust-constr	0.00	0.00	**0.04**	0.12	0.00	0.15	0.14	0.15	0.14	0.15	0.14	0.11	0.14	**4.04**
	PyTorch	Adam	0.00	0.00	0.00	0.12	0.00	0.13	0.14	0.13	0.14	0.14	0.14	0.17	0.15	**3.71**
		AdamW	0.00	0.00	0.00	0.12	0.00	0.14	0.14	0.14	0.14	0.14	0.14	0.16	0.15	3.72
		Yogi	0.00	0.00	**0.01**	0.12	0.00	0.15	0.14	0.15	0.14	0.14	0.14	0.17	0.15	3.79
		LBFGS	0.00	0.00	**0.07**	0.13	0.00	0.12	0.14	0.12	0.14	0.14	0.14	0.17	0.15	3.71
Constraint Violation 2	SciPy	COBYLA	0.00	**1.00**	**0.07**	0.13	0.00	0.12	0.14	0.09	0.13	0.13	0.14	0.10	0.14	**3.98**
		SLSQP	0.00	0.00	**0.30**	0.17	0.00	0.24	0.16	0.16	0.15	0.25	0.16	0.21	0.15	4.13
		Trust-constr	0.00	0.00	**0.23**	0.16	0.00	0.12	0.14	0.08	0.13	0.13	0.14	0.10	0.14	4.14
	PyTorch	Adam	0.00	0.00	**0.11**	0.14	0.00	0.13	0.14	0.19	0.15	0.12	0.14	0.15	0.14	3.95
		AdamW	0.00	0.00	**0.09**	0.14	0.00	0.13	0.14	0.19	0.15	0.12	0.14	0.15	0.14	3.94
		Yogi	0.00	0.00	**0.14**	0.14	0.00	0.14	0.14	0.23	0.16	0.12	0.14	0.15	0.14	4.06
		LBFGS	0.00	0.00	**0.06**	0.13	0.00	0.12	0.14	0.07	0.13	0.12	0.14	0.15	0.14	**3.80**

Table C.5: Softmax aggregated ranking PyTorch vs SciPy results on test scenarios Nominal Operation (N.O.) 1 and 2 for all different optimizers. No constraints violations observed. MPC using PyTorch scores lower showing better performance compared to SciPy.

Scenario	**Library**	**Optimizer**	$\dot{m}_{\text{air}}$, U.B. C.V.	$\dot{m}_{\text{air}}$, L.B. C.V.	$T_{\text{hot air}}$, U.B. C.V.	Rank	$T_{\text{hot air}}$, L.B. C.V.	$\sum \dot{T}_{\text{surface}}$	Rank	$\sum \Delta \boldsymbol{u}$	Rank	$T_{\text{hot air, RMSE}}$	Rank	$\sum \dot{\tilde{H}}_{\text{air}}$	Rank	**Score**
Nominal Operation 1	SciPy	COBYLA	0.00	**1.00**	0.00	0.14	0.00	0.15	0.14	0.10	0.14	0.44	0.19	0.07	0.13	4.64
		SLSQP	0.00	0.00	0.00	0.14	0.00	0.15	0.14	0.14	0.14	0.07	0.13	0.17	0.15	**3.83**
		Trust-constr	0.00	0.00	0.00	0.14	0.00	0.14	0.14	0.13	0.14	0.18	0.15	0.11	0.14	4.17
	PyTorch	Adam	0.00	0.00	0.00	0.14	0.00	0.15	0.14	0.15	0.14	0.08	0.13	0.17	0.15	3.86
		AdamW	0.00	0.00	0.00	0.14	0.00	0.15	0.14	0.14	0.14	0.09	0.13	0.15	0.14	3.91
		Yogi	0.00	0.00	0.00	0.14	0.00	0.14	0.14	0.21	0.15	0.07	0.13	0.17	0.15	3.88
		LBFGS	0.00	0.00	0.00	0.14	0.00	0.13	0.14	0.13	0.14	0.07	0.13	0.17	0.15	**3.79**
Nominal Operation 2	SciPy	COBYLA	0.00	**1.00**	0.00	0.14	0.00	0.15	0.14	0.16	0.14	0.21	0.15	0.12	0.14	4.18
		SLSQP	0.00	0.00	0.00	0.14	0.00	0.14	0.14	0.13	0.14	0.09	0.14	0.16	0.14	**3.87**
		Trust-constr	0.00	0.00	0.00	0.14	0.00	0.16	0.14	0.10	0.14	0.26	0.16	0.10	0.14	4.28
	PyTorch	Adam	0.00	0.00	0.00	0.14	0.00	0.14	0.14	0.13	0.14	0.11	0.14	0.16	0.15	3.89
		AdamW	0.00	0.00	0.00	0.14	0.00	0.15	0.14	0.12	0.14	0.14	0.14	0.15	0.14	3.97
		Yogi	0.00	0.00	0.00	0.14	0.00	0.13	0.14	0.24	0.16	0.10	0.14	0.16	0.15	3.98
		LBFGS	0.00	0.00	0.00	0.14	0.00	0.13	0.14	0.12	0.14	0.10	0.14	0.16	0.15	**3.85**

C.2.2 Aggregated ratio ranking results

Table C.6: Aggregated ratio ranking Pytorch vs SciPy results on test scenarios Constraint Violation (C.V.) 1 and 2 for all different optimizers. Constrained violations in $T_{\text{hot air}}$ observed. PyTorch MPC showing better performance compared to SciPy.

Scenario	Library	Optimizer	$\dot{m}_{\text{air}}$, U.B. C.V.	$\dot{m}_{\text{air}}$, L.B. C.V.	$T_{\text{hot air}}$, U.B. C.V.	$T_{\text{hot air}}$, L.B. C.V.	$\sum \dot{T}_{\text{surface}}$	$\sum \Delta \boldsymbol{u}$	$T_{\text{hot air, RMSE}}$	$\sum \dot{H}_{\text{air}}$	Score
Constraint Violation 1	SciPy	COBYLA	**1.00**	0.00	0.12	0.00	0.13	0.08	0.16	0.08	6.01
		SLSQP	0.00	0.00	0.77	0.00	0.18	0.23	0.14	0.14	9.24
		Trust-constr	0.00	0.00	0.04	0.00	0.15	0.16	0.15	0.11	**4.65**
	PyTorch	Adam	0.00	0.00	0.00	0.00	0.13	0.13	0.14	0.17	**2.38**
		AdamW	0.00	0.00	0.00	0.00	0.14	0.12	0.14	0.16	2.42
		Yogi	0.00	0.00	0.01	0.00	0.15	0.18	0.14	0.17	2.85
		LBFGS	0.00	0.00	0.07	0.00	0.12	0.09	0.14	0.17	2.43
Constraint Violation 2	SciPy	COBYLA	0.00	**1.00**	0.07	0.00	0.12	0.09	0.13	0.10	**3.89**
		SLSQP	0.00	0.00	0.30	0.00	0.24	0.16	0.25	0.21	4.62
		Trust-constr	0.00	0.00	0.23	0.00	0.12	0.08	0.13	0.10	5.34
	PyTorch	Adam	0.00	0.00	0.11	0.00	0.13	0.19	0.12	0.15	3.71
		AdamW	0.00	0.00	0.09	0.00	0.13	0.19	0.12	0.15	3.58
		Yogi	0.00	0.00	0.14	0.00	0.14	0.23	0.12	0.15	4.39
		LBFGS	0.00	0.00	0.06	0.00	0.12	0.07	0.12	0.15	**2.58**

Table C.7: Aggregated ratio ranking Pytorch vs SciPy results on test scenarios Nominal Operation (N.O.) 1 and 2 for all different optimizers. Constrained violations in $T_{\text{hot air}}$ observed. PyTorch MPC showing better performance compared to SciPy.

Scenario	**Library**	**Optimizer**	$\dot{m}_{\text{air}}$, U.B. C.V.	$\dot{m}_{\text{air}}$, L.B. C.V.	$T_{\text{hot air}}$, U.B. C.V.	$T_{\text{hot air}}$, L.B. C.V.	$\sum \dot{T}_{\text{surface}}$	$\sum \Delta \boldsymbol{u}$	$T_{\text{hot air, RMSE}}$	$\sum \dot{\tilde{H}}_{\text{air}}$	**Score**
Nominal Operation 1	SciPy	COBYLA	**1.00**	0.00	0.00	0.00	0.15	0.10	0.44	0.07	10.38
		SLSQP	0.00	0.00	0.00	0.00	0.15	0.14	0.07	0.17	**2.14**
		Trust-constr	0.00	0.00	0.00	0.00	0.14	0.13	0.18	0.11	4.25
	PyTorch	Adam	0.00	0.00	0.00	0.00	0.15	0.15	0.08	0.17	2.27
		AdamW	0.00	0.00	0.00	0.00	0.15	0.14	0.09	0.15	2.50
		Yogi	0.00	0.00	0.00	0.00	0.14	0.21	0.07	0.17	2.45
		LBFGS	0.00	0.00	0.00	0.00	0.13	0.13	0.07	0.17	**1.94**
Nominal Operation 2	SciPy	COBYLA	0.00	**1.00**	0.00	0.00	0.15	0.16	0.21	0.12	4.24
		SLSQP	0.00	0.00	0.00	0.00	0.14	0.13	0.09	0.16	**2.28**
		Trust-constr	0.00	0.00	0.00	0.00	0.16	0.10	0.26	0.10	5.25
	PyTorch	Adam	0.00	0.00	0.00	0.00	0.14	0.13	0.11	0.16	2.41
		AdamW	0.00	0.00	0.00	0.00	0.15	0.12	0.14	0.15	2.84
		Yogi	0.00	0.00	0.00	0.00	0.13	0.24	0.10	0.16	2.97
		LBFGS	0.00	0.00	0.00	0.00	0.13	0.12	0.10	0.16	**2.13**

Bibliography

[1] K. Calvin, D. Dasgupta, G. Krinner, *et al.*, *IPCC, 2023: Climate Change 2023: Synthesis Report. Contribution of Working Groups I, II and III to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change*. Jul. 2023. DOI: 10.59327/ipcc/ar6-9789291691647.

[2] International Energy Agency, *Renewables 2023: Analysis and forecasts to 2028*, IEA, (accessed April 23, 2024), Jan. 2024. [Online]. Available: https://www.iea.org/reports/renewables-2023.

[3] Umweltbundesamt, *Klimaemissionen sinken 2023 um 10,1 prozent – größter rückgang seit 1990*, Umweltbundesamt.de, (accessed April 23, 2024), Mar. 2024. [Online]. Available: https://www.umweltbundesamt.de/presse/pressemitteilungen/klimaemissionen-sinken-2023-um-101-prozent.

[4] International Energy Agency, *World energy outlook 2023*, IEA, (accessed April 23, 2024), Oct. 2023. [Online]. Available: https://www.iea.org/reports/world-energy-outlook-2023.

[5] The World Bank, *Concentrating solar power: Clean power on demand 24/7*, WorldBank, (accessed April 23, 2024), Jan. 2021. [Online]. Available: https://pubdocs.worldbank.org/en/849341611761898393/WorldBank-CSP-Report-Concentrating-Solar-Power-Clean-Power-on-Demand-24-7-FINAL.

[6] Umweltbundesamt, *Europäische energie- und klimaziele*, Umweltbundesamt.de, (accessed May 8, 2024), Mar. 2023. [Online]. Available: https://www.umweltbundesamt.de/daten/klima/europaeische-energie-klimaziele#zielvereinbarungen.

[7] United Nations Framework Convention on Climate Change (UNFCCC), *Adoption of the paris agreement*, (accessed April 23, 2024), Dec. 2015. [Online]. Available: http://digitallibrary.un.org/record/831040.

[8] German Aerospace Center (DLR), *Solar thermal power plants: Heat, electricity and fuels from concentrated solar power*, DLR, (accessed April 23, 2024), May 2021. [Online]. Available: https://www.dlr.de/en/latest/news/2021/01/20210330_study-solar-thermal-power.

[9] R. Merchán, M. Santos, A. Medina, and A. Calvo Hernández, "High temperature central tower plants for concentrated solar power: 2021 overview," *Renewable and Sustainable Energy Reviews*, p. 111 828, 2022. DOI: https://doi.org/10.1016/j.rser.2021.111828.

[10] German Aerospace Center (DLR), *Volumetric receivers - heat conductors under extreme temperatures*, German Aerospace Center, (accessed December 18, 2023), 2023. [Online]. Available: https://www.dlr.de/en/sf/imported-from-cxxl/research-and-development/point-focus-systems/em-solarthermal-receivers-em/volumetrische-receiver/18599_read-43284.

[11] A. L. Ávila-Marín, "Volumetric receivers in solar thermal power plants with central receiver system technology: A review," *Solar Energy*, no. 5, pp. 891–910, 2011. DOI: https://doi.org/10.1016/j.solener.2011.02.002.

[12] K. Hennecke, P. Schwarzbözl, B. Hoffschmidt, *et al.*, "The solar power tower jülich – a solar thermal power plant for test and demonstration of air receiver," in *2007 ISES Solar World Congress, Beijing* Y. Goswami and Y. Zhao, Springer Verlag, Sep. 2007, pp. 1749–1753. [Online]. Available: https://elib.dlr.de/53295/.

[13] N. Ahlbrink, S. Alexopoulos, J. Andersson, *et al.*, "Vicerp - the virtual institute of central receiver power plants: Modeling and simulation of an open volumetric air receiver power plant," in *PROCEEDINGS MATHMOD09 Vienna - Full Papers CD Volume* I. Troch and F. Breitenecker, ser. ARGESIM Report, ARGESIM / ASIM, Feb. 2009. [Online]. Available: https://elib.dlr.de/58262/.

[14] C. Pabst, G. Feckler, S. Schmitz, *et al.*, "Experimental performance of an advanced metal volumetric air receiver for solar towers," *Renewable Energy*, pp. 91–98, 2017. DOI: 10.1016/j.renene.2017.01.016.

[15] J. Gall, D. Abel, N. Ahlbrink, *et al.*, "Simulation and control of solar thermal power plants," *Renewable Energy and Power Quality Journal*, no. 08, pp. 232–236, Apr. 2010. DOI: 10.24084/repqj08.294.

[16] M. Thelen, "Entwicklung eines optischen messsystems für strahlungsflussdichteverteilung und verifizierung anhand hochkonzen-trierter solarstrahlung," M.S. thesis, Rheinische Fachhochschule Köln, Jul. 2016. [Online]. Available: https://elib.dlr.de/109132/.

[17] R. Samu, S. G. Bhujun, M. Calais, *et al.*, "Solar irradiance nowcasting system trial and evaluation for islanded microgrid control purposes," *Energies*, no. 17, 2022. DOI: 10.3390/en15176100.

[18] B. Nouri, S. Wilbert, N. Blum, *et al.*, "Evaluation of an all sky imager based nowcasting system for distinct conditions and five sites," *AIP Conference Proceedings*, no. 1, p. 180 006, Dec. 2020. DOI: 10.1063/5.0028670.

[19] A. N. Yerudkar, D. Kumar, V. H. Dalvi, S. V. Panse, V. R. Gaval, and J. B. Joshi, "Economically feasible solutions in concentrating solar power technology specifically for heliostats – a review," *Renewable and Sustainable Energy Reviews*, 2024. DOI: 10.1016/j.rser.2023.113825.

[20] W. Ding and T. Bauer, "Progress in research and development of molten chloride salt technology for next generation concentrated solar power plants," *Engineering*, no. 3, pp. 334–347, 2021. DOI: 10.1016/j.eng.2020.06.027.

[21] R. Capuano, T. Fend, P. Schwarzbözl, *et al.*, "Numerical models of advanced ceramic absorbers for volumetric solar receivers," *Renewable and Sustainable Energy Reviews*, pp. 656–665, 2016. DOI: 10.1016/j.rser.2015.12.068.

[22] T. Hirsch, N. Ahlbrink, R. Pitz-Paal, *et al.*, "Dynamic simulation of a solar tower system with open volumetic receiver - a review on the vicerp project," in *Proceedings of the SolarPACES 2011 conference*, Sep. 2011. [Online]. Available: https://elib.dlr.de/72402/.

[23] R. Popp, R. Flesch, T. Konrad, U. Jassmann, and D. Abel, "Control-oriented model of a molten salt solar power central receiver," in *2019 18th European Control Conference (ECC)*, 2019, pp. 2295–2300. DOI: 10.23919/ECC.2019.8795914.

[24] R. Popp, K. Iding, P. Schwarzbözl, T. Konrad, and D. Abel, "A comparison between model predictive and PID-based control of a molten salt solar tower receiver," Presented at the 27th International conference on Concentrating Solar Power and Chemical Energy Systems: Solar Power and Chemical Energy Systems, SolarPACES 2021, vol. 2815, Oct. 2023. DOI: 10.1063/5.0148728.

[25] K. Iding, D. Zanger, D. Maldonado Quinto, and R. Pitz-Paal, "A real-time capable simulation of open volumetric receiver surface temperatures with spatially high resolution," *SolarPACES conference Proceedings*, Mar. 2024. DOI: 10.52825/solarpaces.v1i.885.

[26] J. García, Y. Chean Soo Too, R. Vasquez Padilla, A. Beath, J.-S. Kim, and M. E. Sanjuan, "Multivariable closed control loop methodology for heliostat aiming manipulation in solar central receiver systems," *Journal of Solar Energy Engineering*, no. 3, Mar. 2018. DOI: 10.1115/1.4039255.

[27] J. García, R. Barraza, Y. C. Soo Too, *et al.*, "Transient simulation of a control strategy for solar receivers based on mass flow valves adjustments and heliostats aiming," *Renewable Energy*, pp. 1221–1244, Feb. 2022. DOI: 10.1016/j.renene.2021.12.008.

[28] A. Sánchez-González, M. R. Rodríguez-Sánchez, and D. Santana, "Aiming strategy model based on allowable flux densities for molten salt central receivers," *Solar Energy*, pp. 1130–1144, 2017. DOI: 10.1016/j.solener.2015.12.055.

[29] R. Zhu and D. Ni, "A model predictive control approach for heliostat field power regulatory aiming strategy under varying cloud shadowing conditions," *Energies*, no. 7, p. 2997, Mar. 2023. DOI: 10.3390/en16072997.

[30] B. Belhomme, R. Pitz-Paal, and P. Schwarzbözl, "Optimization of heliostat aim point selection for central receiver systems based on the ant colony optimization metaheuristic," *Journal of Solar Energy Engineering*, no. 1, Jul. 2013. DOI: 10.1115/1.4024738.

[31] M. T. Geschonneck, *Modellprädiktive regelung eines keramischen receivers für solartürme*, BA thesis, 2023. [Online]. Available: https://elib.dlr.de/198430/.

[32] M. Ostermann, *Model predictive control and moving horizon estimation for solar tower power plants: Analysis and adaptations to the experimental facility solar tower jülich*, M.S. thesis, 2024. [Online]. Available: https://elib.dlr.de/202446/.

[33] S. Ruiz-Moreno, J. R. D. Frejo, and E. F. Camacho, "Model predictive control based on deep learning for solar parabolic-trough plants," *Renewable Energy*, pp. 193–202, 2021. DOI: 10.1016/j.renene.2021.08.058.

[34] M. Pargmann, D. Maldonado Quinto, P. Schwarzbözl, and R. Pitz-Paal, "High accuracy data-driven heliostat calibration and state prediction with pretrained deep neural networks," *Solar Energy*, pp. 48–56, 2021. DOI: 10.1016/j.solener.2021.01.046.

[35] W. C. Wong, E. Chee, J. Li, and X. Wang, "Recurrent neural network-based model predictive control for continuous pharmaceutical manufacturing," *Mathematics*, no. 11, 2018. DOI: 10.3390/math6110242.

[36] C. Utama, B. Karg, C. Meske, and S. Lucia, "Explainable artificial intelligence for deep learning-based model predictive controllers" S. S. Barbu M., Presented at the 26th International conference on System Theory, Control and Computing, ICSTCC 2022; Institute of Electrical and Electronics Engineers Inc., 2022, pp. 464–471. DOI: 10.1109/ICSTCC55426.2022.9931794.

[37] M. Jung, P. R. da Costa Mendes, M. önnheim, and E. Gustavsson, "Model predictive control when utilizing lstm as dynamic models," *Engineering Applications of Artificial Intelligence*, 2023. DOI: 10.1016/j.engappai.2023.106226.

[38] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," English G. I., F. R., W. H., *et al.*, vol. 2017-December, Neural information processing systems foundation, 2017, pp. 5999–6009. DOI: 10.48550/arXiv.1706.03762.

[39] J. Park, M. R. Babaei, S. A. Munoz, A. N. Venkat, and J. D. Hedengren, "Simultaneous multistep transformer architecture for model predictive control," *Computers and Chemical Engineering*, 2023. DOI: 10.1016/j.compchemeng.2023.108396.

[40] R. Ghanem, D. Higdon, and H. Owhadi, *Handbook of Uncertainty Quantification*. Springer, 2017. DOI: 10.1007/978-3-319-12385-1.

[41] J. Kocijan, R. Murray-Smith, C. E. Rasmussen, and B. Likar, "Predictive control with gaussian process models," Presented at the 2nd IEEE Region 8 EUROCON 2003, vol. A, Institute of Electrical and Electronics Engineers Inc., 2003, pp. 352–356. DOI: 10.1109/EURCON.2003.1248042.

[42] J. Kocijan, R. Murray-Smith, C. E. Rasmussen, and A. Girard, "Gaussian process model based predictive control," Presented at the Proceedings of the 2004 American Control conference (AAC), vol. 3, Institute of Electrical and Electronics Engineers Inc., 2004, pp. 2214–2219. DOI: 10.23919/acc.2004.1383790.

[43] J. Kocijan and R. Murray-Smith, "Nonlinear predictive control with a gaussian process model," Presented at the European Summer School on Multi-Agent Control, vol. 3355, Springer Verlag, 2005, pp. 185–200. DOI: 10.1007/978-3-540-30560-6_8.

[44] B. Likar and J. Kocijan, "Predictive control of a gas-liquid separation plant based on a gaussian process model," *Computers and Chemical Engineering*, no. 3, pp. 142–152, 2007. DOI: 10.1016/j.compchemeng.2006.05.011.

[45] J. Kocijan, *Modelling and Control of Dynamic Systems Using Gaussian Process Models*. Springer International Publishing, 2016, pp. XVI, 267. DOI: 10.1007/978-3-319-21021-6.

[46] Y. Luo, Z. Wang, J. Zhu, *et al.*, "Multi-objective robust optimization of a solar power tower plant under uncertainty," *Energy*, 2022. DOI: 10.1016/j.energy.2021.121716.

[47] N. Mohammadzadeh, H. Truong-Ba, M. E. Cholette, T. A. Steinberg, and G. Manzolini, *A stochastic-milp dispatch optimization model for concentrated solar thermal under uncertainty*, 2024. DOI: 10.48550/arXiv.2401.01133.

[48] G. Torrente, E. Kaufmann, P. Fohn, and D. Scaramuzza, "Data-driven mpc for quadrotors," *IEEE Robotics and Automation Letters*, no. 2, pp. 3769–3776, 2021. DOI: 10.1109/LRA.2021.3061307.

[49] C. Eckel, M. Maiworm, and R. Findeisen, "Optimal operation and control of towing kites using online and offline gaussian process learning supported model predictive control," Presented at the 2022 American Control conference, ACC 2022, vol. 2022-June, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 2637–2643. DOI: 10.23919/ACC53348.2022.9867371.

[50] M. Elsheikh and S. Engell, "Learning-based predictive control using a hybrid model with adaptive domain of validity," Presented at the 27th International conference on Methods and Models in Automation and Robotics, MMAR 2023, Institute of Electrical and Electronics Engineers Inc., 2023, pp. 127–132. DOI: 10.1109/MMAR58394.2023.10242519.

[51] F. Fiedler and S. Lucia, "Improved uncertainty quantification for neural networks with bayesian last layer," *IEEE Access*, pp. 123 149–123 160, 2023. DOI: 10.1109/ACCESS.2023.3329685.

[52] H.-A. Langåker, *Cautious mpc-based control with machine learning*, M.S. thesis, Nov. 2018. [Online]. Available: http://hdl.handle.net/11250/2572395.

[53] L. Hewing, A. Liniger, and M. N. Zeilinger, "Cautious nmpc with gaussian process dynamics for autonomous miniature race cars," Presented at the 16th European Control conference, ECC 2018, Institute of Electrical and Electronics Engineers Inc., 2018, pp. 1341–1348. DOI: 10.23919/ECC.2018.8550162.

[54] L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious model predictive control using gaussian process regression," *IEEE Transactions on Control Systems Technology*, no. 6, pp. 2736–2743, 2020. DOI: 10.1109/TCST.2019.2949757.

[55] J. Kabzan, L. Hewing, A. Liniger, and M. N. Zeilinger, "Learning-based model predictive control for autonomous racing," *IEEE Robotics and Automation Letters*, no. 4, pp. 3363–3370, 2019. DOI: 10.1109/LRA.2019.2926677.

[56] E. Bradford, L. Imsland, D. Zhang, and E. A. del Rio Chanona, "Stochastic data-driven model predictive control using gaussian processes," *Computers and Chemical Engineering*, 2020. DOI: 10.1016/j.compchemeng.2020.106844.

[57] K. P. Wabersich and M. N. Zeilinger, "Nonlinear learning-based model predictive control supporting state and input dependent model uncertainty estimates," *International Journal of Robust and Nonlinear Control*, no. 18, pp. 8897–8915, 2021. DOI: 10.1002/rnc.5688.

[58] A. Mesbah, "Stochastic model predictive control: An overview and perspectives for future research," *IEEE Control Systems*, no. 6, pp. 30–44, 2016. DOI: 10.1109/MCS.2016.2602087.

[59] P. Polcz, T. Péni, and R. Tóth, "Efficient implementation of gaussian process–based predictive control by quadratic programming," *IET Control Theory and Applications*, no. 8, pp. 968–984, 2023. DOI: 10.1049/cth2.12430.

[60] F. Fiedler and S. Lucia, "Probabilistic multi-step identification with implicit state estimation for stochastic mpc," *IEEE Access*, pp. 117 018–117 029, 2023. DOI: 10.1109/ACCESS.2023.3326344.

[61] G. Pan, R. Ou, and T. Faulwasser, "Towards data-driven stochastic predictive control," *International Journal of Robust and Nonlinear Control*, 2023. DOI: 10.1002/rnc.6812.

[62] P. D.-I. S. Engell, *Lecture notes on data-based dynamic modelling*, Technical University Dortmund, Chair of System Dynamics and Process Control, 2023.

[63] O. Nelles, *Nonlinear System Identification: From Classical Approaches to Neural Networks, Fuzzy Models, and Gaussian Processes*. Springer International Publishing, 2020. DOI: 10.1007/978-3-030-47439-3.

[64] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, Oct. 2016, vol. 19, pp. 305–307. DOI: 10.1007/s10710-017-9314-z.

[65] F. Fiedler, B. Karg, L. Lüken, *et al.*, "Do-mpc: Towards fair nonlinear and robust model predictive control," *Control Engineering Practice*, 2023. DOI: 10.1016/j.conengprac.2023.105676.

[66] B. Karg, T. Alamo, and S. Lucia, "Probabilistic performance validation of deep learning-based robust nmpc controllers," *International Journal of Robust and Nonlinear Control*, no. 18, pp. 8855–8876, 2021. DOI: 10.1002/rnc.5696.

[67] C. M. Bishop, *Neural networks for pattern recognition*, Reprinted. Oxford [u.a.]: Oxford University Press, 2010, 482 pp.

[68] F. Rosenblatt, "The perceptron - a perceiving and recognizing automaton," Cornell Aeronautical Laboratory, Ithaca, New York, Tech. Rep. 85-460-1, Jan. 1957.

[69] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, no. 6088, pp. 533–536, Oct. 1986. DOI: 10.1038/323533a0.

[70] B. Karg and S. Lucia, "Efficient representation and approximation of model predictive control laws via deep learning," *IEEE Transactions on Cybernetics*, no. 9, pp. 3866–3878, 2020. DOI: 10.1109/TCYB.2020.2999556.

[71] J. S. Bridle, "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition," in *Neurocomputing* F. F. Soulié and J. Hérault, Berlin, Heidelberg: Springer Berlin Heidelberg, 1990, pp. 227–236. DOI: 10.1007/978-3-642-76153-9_28.

[72] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, no. 7553, pp. 436–444, May 2015. DOI: 10.1038/nature14539.

[73] A. Simpkins, "System identification: Theory for the user, 2nd edition (ljung, l.; 1999) [on the shelf]," *IEEE Robotics and Automation Magazine*, no. 2, pp. 95–96, 2012. DOI: 10.1109/MRA.2012.2192817.

[74] A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," 2018. DOI: 10.48550/ARXIV.1808.03314.

[75] R. M. Schmidt, *Recurrent neural networks (rnns): A gentle introduction and overview*, 2019. DOI: 10.48550/ARXIV.1912.05911.

[76] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, no. 2, pp. 157–166, 1994. DOI: 10.1109/72.279181.

[77] R. C. Staudemeyer and E. R. Morris, *Understanding lstm – a tutorial into long short-term memory recurrent neural networks*, 2019. DOI: 10.48550/ARXIV.1909.09586.

[78] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, no. 8, pp. 1735–1780, Nov. 1997. DOI: 10.1162/neco.1997.9.8.1735.

[79] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015. DOI: 10.48550/ARXIV.1512.03385.

[80] S. Ahuja and A. Kumar, "Expectation-based probabilistic naive approach for forecasting involving optimized parameter estimation," *Arabian Journal for Science and Engineering*, no. 2, pp. 1363–1370, Apr. 2022. DOI: 10.1007/s13369-022-06819-0.

[81] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," Sep. 2014. DOI: 10.48550/ARXIV.1409.0473. arXiv: 1409.0473 [cs.CL].

[82] K. Cao, T. Zhang, and J. Huang, "Advanced hybrid lstm-transformer architecture for real-time multi-task prediction in engineering systems," *Scientific Reports*, no. 1, Feb. 2024. DOI: 10.1038/s41598-024-55483-x.

[83] A. Haviv, O. Ram, O. Press, P. Izsak, and O. Levy, *Transformer language models without positional encodings still learn positional information*, 2022. DOI: 10.48550/ARXIV.2203.16634.

[84] A. Kazemnejad, I. Padhi, K. N. Ramamurthy, P. Das, and S. Reddy, "The impact of positional encoding on length generalization in transformers," May 2023. DOI: 10.48550/ARXIV.2305.19466.

[85] S. J. Nocedal Jorge;Wright, *Numerical optimization*, 2nd ed. New York, NY: Springer, Dec. 2006, 1664 pp. DOI: 10.1007/978-0-387-40065-5.

[86] S. Boyd and L. Vandenberghe, *Convex Optimization*, Version 29. Cambridge: Cambridge University Press, 2004, 716 pp.

[87] P. D.-I. S. Lucia, *Advanced process control. lecture 6: Optimal control and mpc*, Technical University Dortmund, Chair of Process Automation Systems, 2022.

[88] J. E. Dennis, *Numerical methods for unconstrained optimization and nonlinear equations, Originally published: Englewood Cliffs, N.J. : Prentice-Hall, c1983*. Philadelphia, Pa: Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 1996, 1378 pp. DOI: 10.1137/1.9781611971200.

[89] R. J. Vanderbei, *Linear programming, Foundations and extensions*, Fifth edition. Cham, Switzerland: Springer, 2020, 1471 pp.

[90] Y. Nesterov and A. Nemirovskii, *Interior-Point Polynomial Algorithms in Convex Programming*. Society for Industrial and Applied Mathematics, Jan. 1994. DOI: 10.1137/1.9781611970791.

[91] J. B. Rawlings and D. Q. Mayne, *Model predictive control, Theory and design*, 1. printing. Madison, Wis.: Nob Hill Publ., 2009, 533 pp.

[92] M. Morari and J. H. Lee, "Model predictive control: Past, present and future," *Computers and Chemical Engineering*, no. 4–5, pp. 667–682, May 1999. DOI: 10.1016/s0098-1354(98)00301-9.

[93] L. Grüne, *Nonlinear Model Predictive Control, Theory and Algorithms*, 2nd ed. 2017. Cham: Springer, 2017, 4568022 pp. DOI: 10.1007/978-3-319-46024-6.

[94] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, no. 6, pp. 789–814, Jun. 2000. DOI: 10.1016/s0005-1098(99)00214-9.

[95] M. Schwenzer, M. Ay, T. Bergs, and D. Abel, "Review on model predictive control: An engineering perspective," *The International Journal of Advanced Manufacturing Technology*, no. 5–6, pp. 1327–1349, Aug. 2021. DOI: 10 . 1007 / s00170-021-07682-3.

[96] R. Ghanem, D. Higdon, and H. Owhadi, "Introduction to uncertainty quantification," in *Handbook of Uncertainty Quantification*. Springer International Publishing, 2016, pp. 1–4. DOI: 10.1007/978-3-319-11259-6_1-1.

[97] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: Toward safe learning in control," *Annual Review of Control, Robotics, and Autonomous Systems*, pp. 269–296, 2020. DOI: 10.1146/annurev-control-090419-075625.

[98] R. G. McClarren, *Uncertainty Quantification and Predictive Computational Science: A Foundation for Physical Scientists and Engineers*. Springer International Publishing, 2018, pp. 1–344. DOI: 10.1007/978-3-319-99525-0.

[99] P. D.-I. S. Lucia, *Machine learning methods for engineers. lecture 5: Gaussian processes*, Technical University Dortmund, Chair of Process Automation Systems, 2022.

[100] C. E. Rasmussen, "Gaussian processes in machine learning," in *Advanced Lectures on Machine Learning. Revised Lectures.* O. Bousquet, U. von Luxburg, and G. Rätsch. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 63–71. DOI: 10.1007/978-3-540-28650-9_4.

[101] C. M. Bishop, *Pattern recognition and machine learning*. New York, NY: Springer Science+Business Media, LLC, 2019, 758 pp.

[102] I. Bilionis and N. Zabaras, "Bayesian uncertainty propagation using gaussian processes," in *Handbook of Uncertainty Quantification*. Springer International Publishing, 2015, pp. 1–45. DOI: 10.1007/978-3-319-11259-6_16-1.

[103] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.

[104] T. Developers, *Tensorflow*, 2024. DOI: 10.5281/ZENODO.4724125.

[105] J. Ansel, E. Yang, H. He, *et al.*, "Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation," in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ser. ASPLOS '24, ACM, Apr. 2024. DOI: 10.1145/3620665.3640366.

[106] W. Falcon, J. Borovec, A. Wälchli, *et al.*, *Pytorch lightning*, 2020. DOI: 10.5281/ZENODO.3828935.

[107] P. Virtanen, R. Gommers, T. E. Oliphant, *et al.*, "Scipy 1.0: Fundamental algorithms for scientific computing in python," *Nature Methods*, no. 3, pp. 261–272, Feb. 2020. DOI: 10.1038/s41592-019-0686-2.

[108] D. Kraft, "A software package for sequential quadratic programming," DLR German Aerospace Center - Institute for Flight Mechanics, Koln, Germany, Tech. Rep. DFVLR-FB 88-28, 1988.

[109] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2014. DOI: 10.48550/ARXIV.1412.6980.

[110] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," Nov. 2017. DOI: 10.48550/ARXIV.1711.05101.

[111] M. Zaheer, S. Reddi, D. Sachan, S. Kale, and S. Kumar, "Adaptive methods for nonconvex optimization," in *Advances in Neural Information Processing Systems* S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, vol. 31, Curran Associates, Inc., 2018. [Online]. Available: https://dl.acm.org/doi/10.5555/3327546.3327647.

[112] M. Schmidt, *Minfunc: Unconstrained differentiable multivariate optimization in matlab*, 2005. [Online]. Available: https://www.cs.ubc.ca/~schmidtm/Software/minFunc.html.

[113] M. J. D. Powell, "A direct search optimization method that models the objective and constraint functions by linear interpolation," in *Advances in Optimization and Numerical Analysis* S. Gomez and J.-P. Hennart. Dordrecht: Springer Netherlands, 1994, pp. 51–67. DOI: 10.1007/978-94-015-8330-5_4.

[114] M. J. D. Powell, "Direct search algorithms for optimization calculations," *Acta Numerica*, pp. 287–336, Jan. 1998. DOI: 10.1017/s0962492900002841.

[115] M. J. D. Powell, "A view of algorithms for optimization without derivatives 1," Tech. Rep., 2007. [Online]. Available: https://optimization-online.org/?p=9121.

[116] R. H. Byrd, M. E. Hribar, and J. Nocedal, "An interior point algorithm for large-scale nonlinear programming," *SIAM Journal on Optimization*, no. 4, pp. 877–900, Jan. 1999. DOI: 10.1137/s1052623497325107.

[117] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, pp. 2825–2830, 2011.

[118] J. R. Gardner, G. Pleiss, D. Bindel, K. Q. Weinberger, and A. G. Wilson, *Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration*, 2018. DOI: 10.48550/ARXIV.1809.11165.

[119] P. D.-I. S. Lucia, *Advanced process control. lecture 8: Theoretical properties of mpc*, Technical University Dortmund, Chair of Process Automation Systems, 2022.

[120] X. Wang, J. Liu, and H. Peng, "Model predictive control: From open-loop to closed-loop," in *Symplectic Pseudospectral Methods for Optimal Control: Theory and Applications in Path Planning*. Singapore: Springer Singapore, 2021, pp. 115–119. DOI: 10.1007/978-981-15-3438-6_7.

[121] Z. Liu, Y. Wang, S. Vaidya, *et al.*, *Kan: Kolmogorov-arnold networks*, 2024. DOI: 10.48550/ARXIV.2404.19756.

[122] A. N. Angelopoulos and S. Bates, *A gentle introduction to conformal prediction and distribution-free uncertainty quantification*, 2021. DOI: 10.48550/ARXIV.2107.07511.