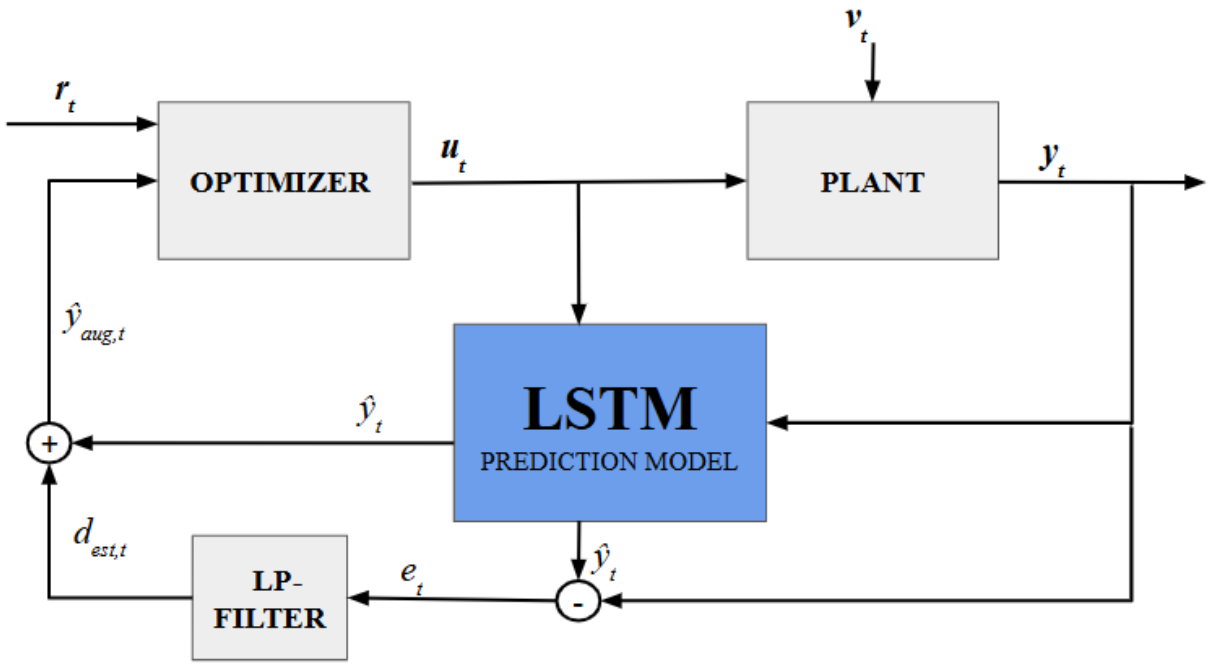
CHALMERS
UNIVERSITY OF TECHNOLOGY



AI-driven MPC/DMC: Automated Model Generation & Smart Control for the Future Process Industry

Investigation of Whether an LSTM-MPC can Match the Performance of a Traditional DMC Controller

Master's Thesis in Systems, Control and Mechatronics

Filippa Forsman

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's Thesis 2026

AI-driven MPC/DMC: Automated Model Generation & Smart Control for the Future Process Industry

Investigation of Whether an LSTM-MPC can Match the Performance of a Traditional DMC Controller

FILIPPA FORSMAN

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2026

AI-driven MPC/DMC: Automated Model Generation & Smart Control for the Future Process Industry
Investigation of Whether an LSTM-MPC can Match the Performance of a Traditional DMC Controller
FILIPPA FORSMAN

Supervisor: David Thor, VAROPreem
Examiner: Torsten Wik, Department of Electrical Engineering at Chalmers University of Technology

Master's Thesis 2026
Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Cover: Model Predictive Control (MPC) Loop with an LSTM Prediction Model.

Typeset in LaTeX
Gothenburg, Sweden 2026

AI driven MPC/DMC: Automated Model Generation & Smart Control for the Future Process Industry

FILIPPA FORSMAN
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Traditionally, Model Predictive Control (MPC) controllers applied in refinery plants rely on linear prediction models derived from step response tests, under steady state operation. Modern process industries often produce large amounts of measurement data, which brings potential for AI-driven modeling instead. Through deep learning frameworks such as Long Short-Term Memory (LSTM), a model could learn system dynamics through historical input-output data. However, identifying dynamics can be challenging, since process data often is collected in closed-loop setting with an active controller.

This thesis investigates an LSTM-based MPC controller applied to a pass-balancing problem in an industrial furnace. The LSTM prediction model was trained using either real historical process data or synthetically generated open-loop data based on a First Order Plus Dead-Time (FOPDT) process model. A Hybrid Physics-Guided LSTM-MPC was also evaluated, utilizing the FOPDT model to ensure the correct physics of the system while applying the LSTM model for nonlinearities and disturbances.

The performance of the proposed LSTM-MPC controller was evaluated in comparison to the traditional linear MPC controller in a custom simulation environment. The results demonstrate that performance and stability of an AI-driven MPC strongly depend on the quality of the training data, particularly in closed-loop settings. The prediction performance in regards of MSE error of the LSTM network alone is not enough to determine if the model will do well in closed loop control, hence evaluation through simulation is essential. Furthermore, stable closed loop performance was achieved using synthetically generated training data, showing that LSTM-based MPC can be a promising approach for future nonlinear process control, and motivating further research into automated model generation for industrial control systems.

Keywords: Long Short-Term Memory Networks, Nonlinear Model Predictive Control, Data Driven Control, Pass-Balancing

Acknowledgements

I would like to thank my thesis supervisor David Thor, and the rest of the Department of Process Control at VAROPreem, for their guidance and for sharing their knowledge in the field. I have gotten the opportunity to learn a lot and it has been a great project to work on.

Furthermore, I would like to thank Torsten Wik for examining the project and for his insightful inputs throughout.

And finally, I want to thank my friends at Chalmers and family at home for the love and support.

Filippa Forsman, Gothenburg, June 2026

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis:

MPC	Model Predictive Control
DMC	Dynamic Matrix Control
NMPC	Non-Linear Model Predictive Control
PID	Proportional-Integral-Derivative controller
MIMO	Multi-Input Multi-Output
DCS	Distributed Control System
LSTM	Long Short-Term Memory
FOPDT	First Order Plus Dead-Time
RNN	Recurrent Neural Network
PINN	Physics Informed Neural Network
MSE	Mean Squared Error
QP	Quadratic Programming
SQP	Sequential Quadratic Programming
BPTT	Backpropagation Through Time
CV	Controlled Variable
MV	Manipulated Variable
FF	Feed-Forward
FC	Flow Controller
TI	Temperature Indicator

Nomenclature

Below is the nomenclature of indices, parameters, and variables that have been used throughout this thesis.

Indices

i,j	General Indices
t	Discrete-time (sequence) index in LSTM formulation
k	Discrete time index

Parameters

Q	Penalty for prediction error
R	Penalty for aggressive input moves
S	Penalty for deviation from non-zero flow-bias sum
H_p	Prediction Horizon
H_c	Control Horizon
α_d	Filter gain for the First Order Low Pass filter
W	Learnable weight matrices for the LSTM
b	Bias vectors for the LSTM
τ	Time constant
θ	Dead Time
K	Gain
$T_{\min}$	Minimum temperature under normal operation
$T_{\max}$	Maximum temperature under normal operation
$F_{\min}$	Minimum flow under normal operation (per pass)
$F_{\max}$	Maximum flow under normal operation (per pass)
I	Integral State

δ	Dead Band
K_i	Integral Gain

Variables

u_k	Input
x_k	State
y_k	Measured Output
$\hat{y_k}$	Predicted Output
d_k	Disturbance Estimate
e_k	Raw Residual Error
$e_{gaussian,k}$	Gaussian disturbance on synthetically generated data
v_k	Measurement Noise
r	Reference Signal
Y^{seq}	Output sequence
X^{seq}	Input sequence
U	Vector of Manipulated Flows
D	Measured Disturbance
Y_{meas}	Current Measured Temperature
ΔU^*	Optimal Control moves

Contents

List of Figures

List of Tables

1
Introduction

1.1 Background

At the VAROPreem refinery in Lysekil, there are process units controlled using Model Predictive Control (MPC) or Dynamic Matrix Control (DMC) controllers, the purpose of which to ensure safe control and optimizing production as well as resource use. This is a widely used control method in the process industry for complex systems, since it can account for constraints on plant and actuators. However, the performance of an MPC controller is closely related to how well the model of the system is formulated. Model identification can be a time consuming and difficult task in the MPC implementation and typically requires plant experiments followed by system identification, unless there is a simple physical model. Currently implemented MPC/DMC controllers often use step response models as prediction models, which are reliable but can fail to handle complex non-linear behavior.

In this thesis project, a pass-balancing problem was considered for a furnace at VAROPreem Lysekil's refinery. The furnace H2201, which has eight feed-passes, is used for preheating. The aim is to keep an equal temperature in all passes and the temperature in the furnace is not homogeneous. To do this, an average temperature is calculated from sensor data from each of the eight passes, and then the error for each pass can be calculated as the deviation from that average temperature. The goal is for the controller to drive these errors to zero, by regulating eight control valves to adjust the flow rate through each pass.

The aim of this thesis was to investigate if the modeling of the dynamics for the controller can be AI-driven, specifically through a deep learning framework called Long Short-Term Memory (LSTM) which is a type of Recurrent Neural Network (RNN). These modeling methods are known to do well in the prediction of dynamic systems, according to Wu et al. [1]. However, their reliability in terms of robustness, stability and computational time in a closed loop controller system needed to be further investigated.

1.2 State of the Art

In the literature regarding AI-driven MPC, there is previous research motivating the study directions of deep learning and an LSTM network for this thesis. Advances in machine learning have enabled the use of neural network based prediction mod-

els within MPC, which can allow nonlinear dynamics to be captured and thereby improving control of nonlinear systems. RNN and LSTM networks are suitable for dynamic process modeling because they can capture temporal dependencies and process "memory" directly from historical input and output data [2].

Wu et.al [1] describe how because of recent improvements in the data handling field, machine learning techniques are now more accessible in traditional engineering, such as the process industry. They state that modeling a system for MPC using RNN has been successful. However, the formulation of formal stability guarantees for RNN-based MPC remains challenging, as does the need for computational time reduction.

Seel et al. [3] also discuss the problem with closed loop stability and the lack of formal guarantees for stability overall. While AI-driven MPC through neural networks has been successful in capturing non-linear dynamics for several applications, the proof of input to state stability remains a challenge.

Furthermore, Rajasekhar et al. [4] discuss the choice of RNN specifically as the preferred deep learning model, because of the fact that data in the process industry is dynamic, which makes RNN a good choice because they are designed to handle time series data. Specifically, the more recent developments of RNN such as LSTM have the ability to overcome the issue with the vanishing gradient problem (diminishing gradients during backpropagation, hence losing long term dependencies) when learning long sequences of data, which is why it is of particular interest in this thesis.

However, passive process data (collected while the plant is operating under normal feedback control) is often not enough for training. Because the controller is already keeping the system at setpoint, the data lacks the excitation needed to show how the plant behaves when things change. Rajasekhar et al. [4] state that deep learning can overcome the manual effort of formulating a prediction model of a dynamic system, by automatically learning the system behavior directly from process data. Rajasekhar et al. use a synthetic plant simulator to perform system identification, by generating data from a known model. By applying a range of randomized inputs such as step changes, pulses, and stochastic noise built on a known model, they create an open loop dataset that covers the system dynamics, including couplings and dead times which the LSTM can be trained on [4].

Huang et al. [5] propose an LSTM-MPC to handle multi-mode process control, where operating range and conditions change and therefore lead to a change in model. Their work successfully lead to an LSTM model that could match the modes, as well as being implemented as prediction model in an MPC setting with stable and robust results.

According to Kong et al. [6], deep learning methods have become increasingly popular for time series forecasting due to their ability to capture complex temporal dependencies directly from data. Architectures like RNNs, LSTMs, Gated Recurrent Units (GRUs), and Transformer models, for example, have shown promising

prediction performance across some applications. Among the architectures, LSTM networks is one of the most widely used approaches for industrial time series forecasting because of its ability to capture long-term dependencies while also handling the vanishing gradient problem [6], [7].

1.3 Application: Pass-Balancing an Industrial Furnace

The dynamic system which the LSTM model was attempted to identify is an industrial furnace, for which the aim is to solve a global pass-balancing optimization control problem. To improve the lifespan of the furnace and avoid buildup formation in the pipes, it is desirable to keep an equal temperature in each pass. The heating capacity of the furnace is adjusted by the flows of fuel gas to the burners. The heat distribution in the furnace is not homogeneous. Hence, the heat transfer to the passes will differ depending on where in the furnace the pass is located, as well as on buildup differences in the pipes. To keep the same temperature in each pass, the flow (and therefore the residence time in the furnace) through each pass is adjusted using flow control valves.

1.3.1 H2201 - A Vacuum Distillation Preheating Furnace

The furnace which the thesis considers is for preheating feed to a distillation unit. The furnace has 8 separate passes which the feed flows through in order to reach a desired temperature before the distillation tower. The system is illustrated in Figure 1.1 and shows the MVs (manipulated variables), FF (feed forward) disturbance and CVs (controlled variables) of the pass balancing problem. The control objective is to keep all outlet pass temperatures (TI4 to TI11) equal. The average temperature is mainly affected by the total flow through the furnace, which is the input to the total flow controller FC_{master} (FC0 in Figure 1.1), as well as the effectiveness of the furnace burners.

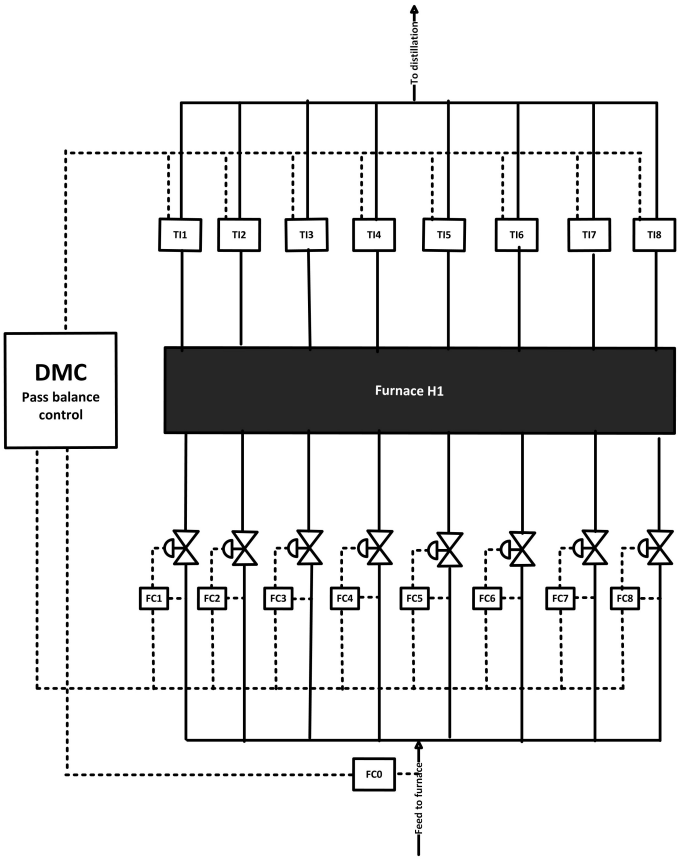


Figure 1.1: Pass balancing of an eight pass furnace.

The oven has eight control valves on ground level of the furnace, which control the flow through eight passes entering the oven at slightly different levels on the upper part of the oven, and then flows downwards through the pipes inside of the oven. The temperature sensors measure the temperature in each pass individually as they exit the furnace, before all converging into the main pipe heading to the vacuum distillation tower. The control valves (the actuators) that control the eight inlet flows are controlled by PID controllers based on the setpoint of FC_{master}, which decides the base output for the control valves (opening percent of the valves). However, these PID controllers simply adjust the actuators such that they correctly add up to the total flow by FC_{master}. For the outlet temperatures of each pass to be equal, the valve actuators will need to have different outputs to adjust the individual flow rates, which is fed back as a bias to the PID to adjust the valve. The control valves are not all of the same type and of the same physical instrument tuning and they also wear over time.

The setpoints for FC1 to FC8 hence consists of the output from the master PID controller FC_{master} to hold an eighth of the total inlet flow per pass, and then an additional bias from the DMC controller as an external target to keep all temperatures equal, meaning no deviation from the mean outlet temperature. If a pass is too hot (positive temperature-bias from the average outlet temperature), a flow-bias should be added to the corresponding flow controller to increase the flow rate through that

specific pass. Hence, all flow biases will not be equal once the DMC biases have been applied.

However, the sum of all pass biases applied by the external DMC controller should always add up to zero, in order to not increase or decrease the total flow rate through the furnace. The DMC will hence redistribute the total flow requested by the the PID in order to minimize the temperature error from the average temperature. If the flow bias is increased on one pass, it must decrease in other passes. By the DMC, FC_{master} is considered a disturbance variable which is not controlled.

The independent variables for this system are hence the flow-bias to the FCs (MVs) as well as a disturbance from the total inlet flow. The controlled variables (CVs) are the eight outlet pass temperatures TI4 to TI11 (Temperature Indicators) deviation from the average, which the goal is to drive to zero. Measured is each individual pass flow rates and individual temperatures (Process Values, PV). Therefore, the matrix formulation of the model is:

Manipulated Variables:

$$u_k = \begin{bmatrix} u_{1,k} \\ \vdots \\ u_{8,k} \end{bmatrix} = \begin{bmatrix} FC1.PV_k \\ \vdots \\ FC8.PV_k \end{bmatrix}$$

Measured Feedforward Disturbance:

$$v_k = \begin{bmatrix} v_{\text{total\_flow}_k} \end{bmatrix} = \begin{bmatrix} FC_{master}.PV_k \end{bmatrix}$$

Controlled Variables:

$$y_k = \begin{bmatrix} y_{1,k} \\ \vdots \\ y_{8,k} \end{bmatrix} = \begin{bmatrix} TI4.PV_k \\ \vdots \\ TI11.PV_k \end{bmatrix}$$

Unmeasured Disturbance (Observer Output for plant-model mismatch):

$$d_k = \begin{bmatrix} d_{1,k} \\ \vdots \\ d_{8,k} \end{bmatrix}$$

The operating modes of the plant are described in Table 1.1. When the DMC controller is active, FC_{master} should be in cascade control (RCAS) and the pass flow controllers in Remote Out (ROUT), meaning they are driven by an external source.

Table 1.1: Operating Modes with DMC Active - H2201

Flow Controller	**Normal Operating Mode**	**Backup-Mode**
FC1	ROUT	AUTO/MAN
FC2	ROUT	AUTO/MAN
FC3	ROUT	AUTO/MAN
FC4	ROUT	AUTO/MAN
FC5	ROUT	AUTO/MAN
FC6	ROUT	AUTO/MAN
FC7	ROUT	AUTO/MAN
FC8	ROUT	AUTO/MAN
FC_{master}	RCAS	AUTO/MAN

The MVs and CVs for the system are defined in Table 1.2 .

Table 1.2: Manipulated variables (MVs) and controlled variables (CVs) used in the control structure.

Manipulated Variables (MVs)		**Controlled Variables (CVs)**	
$FC1_{BIAS}$	Flow bias Pass 1	$TI4_{BIAS}$	Temperature bias Pass 1
$FC2_{BIAS}$	Flow bias Pass 2	$TI5_{BIAS}$	Temperature bias Pass 2
$FC3_{BIAS}$	Flow bias Pass 3	$TI6_{BIAS}$	Temperature bias Pass 3
$FC4_{BIAS}$	Flow bias Pass 4	$TI7_{BIAS}$	Temperature bias Pass 4
$FC5_{BIAS}$	Flow bias Pass 5	$TI8_{BIAS}$	Temperature bias Pass 5
$FC6_{BIAS}$	Flow bias Pass 6	$TI9_{BIAS}$	Temperature bias Pass 6
$FC7_{BIAS}$	Flow bias Pass 7	$TI10_{BIAS}$	Temperature bias Pass 7
$FC8_{BIAS}$	Flow bias Pass 8	$TI11_{BIAS}$	Temperature bias Pass 8
FC_{master}	Total flow to H2201		

The operating range for the model is defined by the alarm limits implemented for H2201, shown in Table 1.3. The full flow and temperature range is larger, however, during normal operation these ranges apply.

Table 1.3: Operating limits

Parameter	Value
$T_{\min}$	360°C
$T_{\max}$	450°C
$F_{\min}$	9.6 m^3/h
$F_{\max}$	37.5 m^3/h

The currently implemented DMC controller uses a step response prediction model,

a First Order Plus Dead Time (FOPDT) that defines the dead times, time constants and gains of the system, all dimensioned as in Figure 1.2. The dark shaded diagonal of the matrices corresponds to the pass flow-temp bias pair, for example the direct effect of the temperature on pass 1 if the flow on pass 1 is changed ($i, j = 1, 1$ etc.). The light shaded areas are the cross-correlations between the passes. For example, if the flow bias on pass 1 is increased, not only TI_1 will be affected, but the change will carry over to the other pass temperatures as well.

The gain matrix has a negative diagonal, and positive cross correlations. That means that if the flow bias of the first pass is increased, then the temperature of the pass is decreased by the negative gain and for all other passes the temperature will increase by a positive gain.

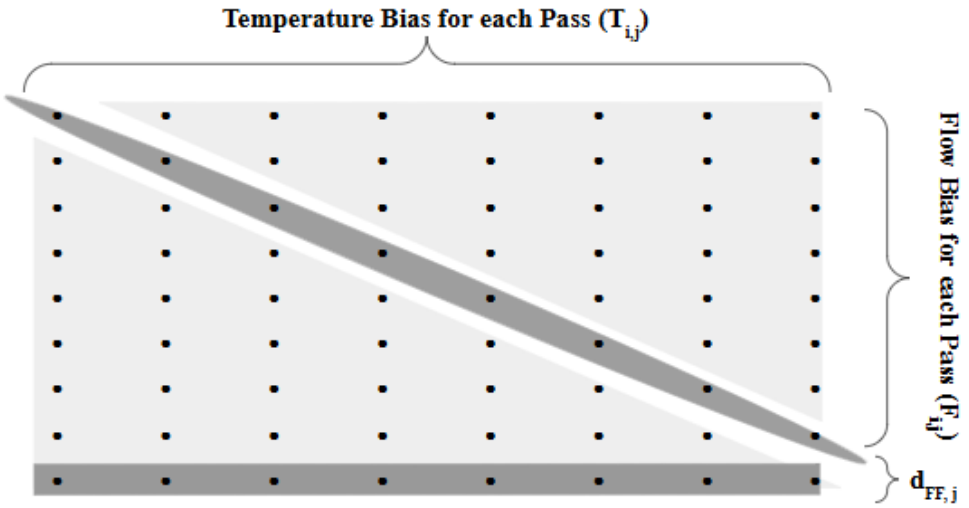


Figure 1.2: The FOPDT matrix format for the H2201 deadtime-, time constant- and gain-matrix.

The control module for the pass balancing DMC controller for H2201 is described in Figure 1.3. It consist of the PID as master and the DMC as an external controller. The sum of the inputs from the DMC and the PID is used for regulating the valve actuators.

$$u_i = u_{\mathrm{PID},i} + \Delta u_{\mathrm{DMC},i}$$

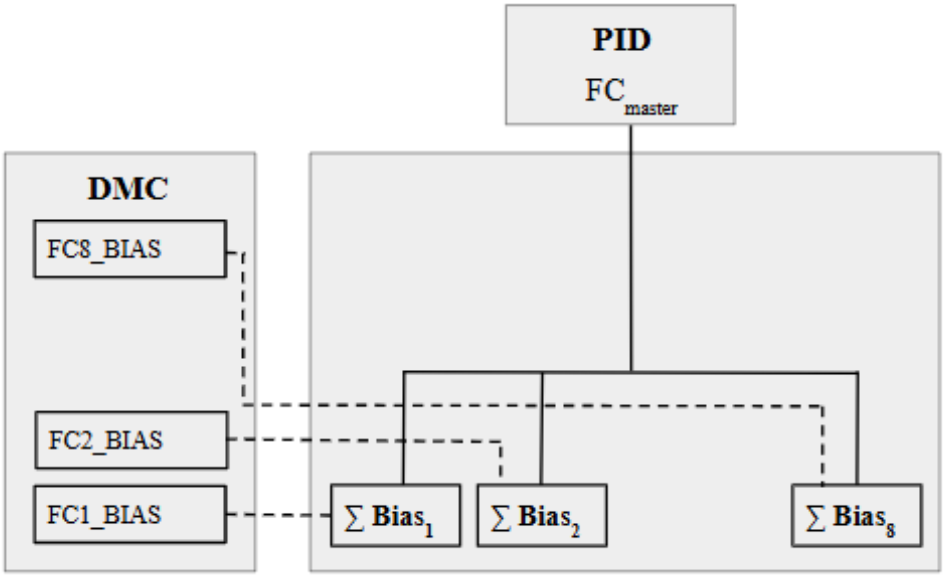


Figure 1.3: Control module H2201

1.4 Purpose and aim

The aim of this thesis was to develop and evaluate a data-driven model, specifically a Recurrent Neural Network (RNN) through Long Short-Term Memory (LSTM), for an MPC controller in a process control problem. Comparison of the performance of a traditional step response DMC/MPC controller is needed to evaluate if the LSTM-MPC would improve the control of the pass-balancing problem for H2201. This was done by building a Python-based environment where the LSTM-model was formulated using PyTorch, and a simulation environment was created to test both controllers and compare their closed loop performance.

The following research questions, the main aim of the thesis, are the base for the methods that will be presented in Chapter 3. The results are evaluated in a separate discussion chapter.

- To what extent can an MPC with a data driven LSTM prediction model match or surpass the closed-loop performance of traditional linear MPC/DMC, and what are the trade-offs regarding computational cost and operability?
- To what extent can we trust standard accuracy metrics to guarantee that an AI-based controller will operate safely and effectively in a closed-loop environment?
- How does the robustness of LSTM-MPC and traditional MPC/DMC differ when the system is exposed to errors and disturbances?

1.5 Scope, limitations and assumptions

The following points were not considered during the project:

- The project considers the pass-balancing problem of H2201, and does not generalize to other process units.
- Implementation on the company control hardware is outside the scope of the project; the work was done using Python through algorithms and control logic.
- The mathematical proof of stability and interpretation of the internal model is also outside of the scope, mainly because of the nonlinear model. The model will be validated through control results in simulation, and not theoretically.
- The currently implemented FOPDT step test model on the pass-balancing of H2201 is considered to be the "true" system dynamics, in comparison.

2

Theory

Industrial process control systems are often based on PID controllers, sometimes with more complex systems having an external controller, such as an MPC or DMC [8]. In this case, while the PID takes care of the local objective such as keeping the flow through a pass at a certain value by adjusting the valve position based on a temperature reading, the MPC acts as a global optimizer by aiming to keep all pass temperatures equal. MPC builds on a dynamic model of the system to predict the next output of the system in order to determine the next input or control move, while the PID uses the model to formulate the transfer function that relates the input to the output.

2.1 Model Predictive Control - MPC

Model Predictive Control (MPC) is a control strategy that uses a prediction model of a system's dynamics to predict future system behavior over a time period, a prediction horizon H_p. With this, an optimizer finds an ideal sequence of control moves over a control horizon H_c, minimizing an objective function subject to (s.t) some constraints [9].

Classical MPC solves an optimization problem (specifically, a Quadratic Program, QP) where the solutions determine the values of the MVs to be used in the plant until the next control interval. MPC is also known as Receding Horizon Control, where the optimizer calculates a whole sequence of future moves for the next control horizon. But only the very first optimal move is executed. Then, time moves forward by one time step, the loop takes a new sensor reading, and the procedure is repeated [10].

The MPC loop used in this thesis follows the closed-loop architecture illustrated in Figure 2.1. In this formulation, $r(t)$ denotes the reference trajectory, $u(t)$ represents the manipulated variables, $y(t)$ is the measured controlled output, $\hat{y}(t)$ is the predicted output generated by the prediction model, $d(t)$ denotes external disturbances acting on the system, and $e(t)$ represents the prediction error. The feedforward disturbance takes the measured disturbance FC_{master} and adjusts the input before it is applied to the plant.

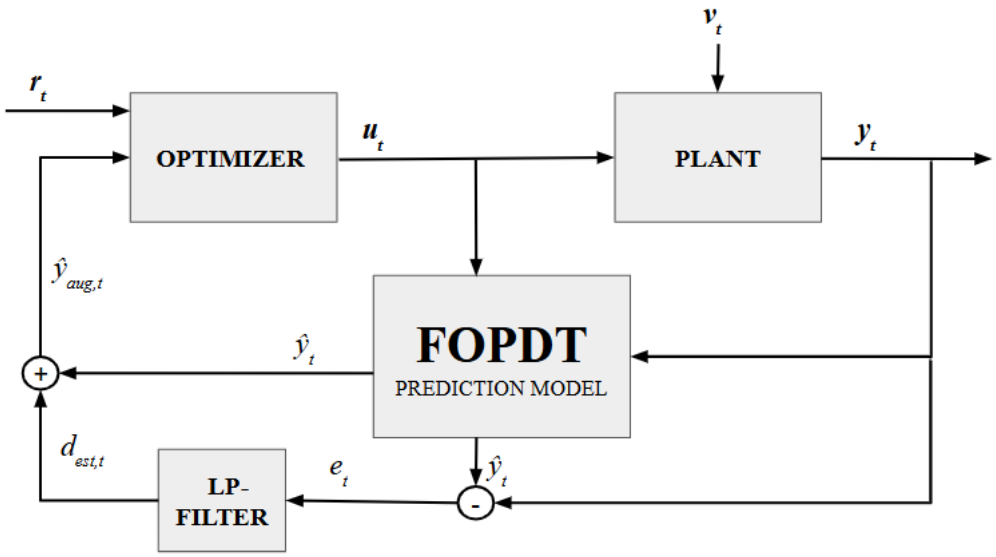


Figure 2.1: MPC-loop used in the thesis

Prediction Models for MPC

For the MPC algorithm to calculate the next move, a prediction model is used to estimate the next expected output. The prediction model is not always correct. However, with a prediction error feedback as in Figure 2.1, the error can be accounted for. It is well known that the quality and performance of the MPC controller relies heavily on the accuracy of the prediction model [10], [9].

The prediction model describing the system dynamics can be derived using physical relations, which results in differential equations formulated as a state space model. However, a physics derived equation can be difficult to formulate since reality often does not follow ideal behavior.

A way to identify the plant dynamics commonly used in the process industry is by various types of system identification methods. Through a step response test on the plant, a mathematical model of dynamic systems based on observed input-output data can be formulated. The plant starts at equilibrium and an input step (a sudden change in input) is applied, then the behavior can be observed through its output [11]. The model order is determined and, for example, a First Order Plus Dead Time (FOPDT) model is formulated. This type of model can take multiple inputs and multiple outputs, and capture cross correlations between inputs and outputs.

The FOPDT model is of first order and formulated by injecting a change to the plant while it is in equilibrium and then looking at the response curve of how the plant settle after the change. This results in a gain matrix ($K_{i,j}$, how much the temperatures will eventually change), a time constant matrix ($\tau_{i,j}$, how fast it moves to get there, specifically the time it takes to reach 63 percent of the final change), and a dead time matrix ($\theta_{i,j}$, the delay before it starts moving at all) [11]. In continuous

time, the FOPDT is formulated as

$$\tau \frac{dy_t}{dt} + y_t = K_{i,j} \cdot u_{t-\theta}. \tag{2.1}$$

Using the step invariance method (in this case with a 1-minute sample time), the system is converted to discrete time:

$$y_k = \alpha \cdot y_{k-1} + K_{adj} \cdot u_{k-1-\theta} \tag{2.2}$$

Where $T_s = 1$ is the sample time and $\alpha = e^{-1/\tau}$ is the memory decay factor. If τ is large, α is close to 1, meaning the furnace holds its heat for a long time, and $K_{adj} = K \cdot (1 - \alpha)$ [12].

Although this method is commonly used in the process industry for modeling system dynamics, other types of system identification are possible such as classical system identification of general transfer functions or through deep learning [13], which is discussed in Section 2.2.

Solving the Optimization problem

In classical MPC, the prediction model is a linear model (usually linearized around an operating point). This results in a convex quadratic optimization problem that can rapidly be solved using quadratic programming (QP) techniques [8], [10]. The objective of the MPC is to minimize a cost function or an objective function (in this thesis, formulated as in Equation 2.3). At each sampling instant, the controller predicts the future system response over a finite prediction horizon and determines an optimal sequence of future control actions by minimizing the cost function while satisfying process constraints, such as (2.4) and (2.5) [10]. In this thesis there are three terms in the cost function to minimize: The tracking error term ($y_k - r$, output deviation from the reference signal), the control action term (Δu_k, penalty to avoid aggressive control moves) and a penalty for deviation from zero flow bias sum ($u_{i,k}$).

$$\textbf{min } J = \frac{Q}{H_p} \sum_{k=1}^{H_p} \|\hat{y}_k - r\|^2 + R \sum_{k=1}^{H_c} \|\Delta u_k\|^2 + \frac{S}{H_p} \sum_{k=1}^{H_p} (\sum_{i=1}^{8} u_{i,k}^2) \tag{2.3}$$

$$\textbf{s.t. } \Delta u \leq \Delta u_{\max} \tag{2.4}$$

$$y_{tot} = 0 \tag{2.5}$$

Specifically, we solve Equation 2.3 such that the equality- (Equation 2.5) and inequality constraints (Equation 2.4) are satisfied. Here,

- H_p is the prediction horizon (how far into the future the output is predicted).
- H_c is the control horizon (how many future control moves are optimized).
- r is the reference signal (setpoint).
- y_k is the temperature bias at time step k.
- Δu_k is the change in control input.
- $u_{i,k}$ is the control move (change in flow bias) per pass.

- R is the move penalty, which penalizes aggressive control actions.
- Q is the reference tracking penalty, which penalizes prediction error.
- S is the flow-bias sum penalty, which penalizes deviations from zero flow bias sum.

AI-driven MPC

With the growth in industrial data availability, data driven prediction models for MPC has become a possible solution for process units with nonlinear dynamics. However, introducing a nonlinear prediction model transforms the MPC optimization problem into a nonlinear programming problem (Nonlinear MPC, NMPC), which can no longer be solved using the QP solvers commonly used in linear MPC [8]. Therefore, nonlinear MPC often use Sequential Quadratic Programming (SQP) instead. SQP solves the nonlinear optimization problem iteratively by approximating it as a sequence of quadratic programming subproblems. At each iteration, the nonlinear objective function is locally approximated by a quadratic model, while the constraints are linearized around the operating point [14]. The resulting QP is solved and used to update the control trajectory until it converges to a feasible solution. This SQP approach allows MPC to handle nonlinear prediction models while still keeping advantages of quadratic optimization [15].

2.2 Deep Learning and Recurrent Neural Networks

As mentioned, system identification is a way of identifying a model of a dynamic system using measured input and output data. Traditionally, this has been achieved using parametric model structures such as transfer functions, ARX models, and state-space representations. However, recent advances in machine learning have introduced deep neural networks as an alternative model structure for system identification [13].

According to Ljung et al. [13], deep learning can be viewed as an extension of classical black box modeling, where the model structure is represented by a neural network rather than a physical or parametric model. The network parameters are estimated directly from data by minimizing a prediction error criterion, similarly to traditional identification methods.

Deep learning is a subset of machine learning that uses multi-layered artificial neural networks to identify patterns within data. For time-series forecasting, Recurrent Neural Networks (RNN) are a commonly used deep learning architecture because of the feedback memory [7]. RNNs process sequences by maintaining a feedback loop, where the output of a previous step is fed back into the network as part of the input for the current step, which allows the model to maintain a memory of past system states, as pictured in Figure 2.2. The red arrows represent the recurrent step, which distinguishes the RNN from a Feedforward Neural Network. The red circles represents cells in the hidden layer, and in the figure a single layer RNN is shown.

However, standard RNNs are affected by the vanishing gradient problem, where information from early in a sequence is lost as the network propagates errors over time, which is where the RNN version Long Short-Term Memory (LSTM) comes in handy [7].

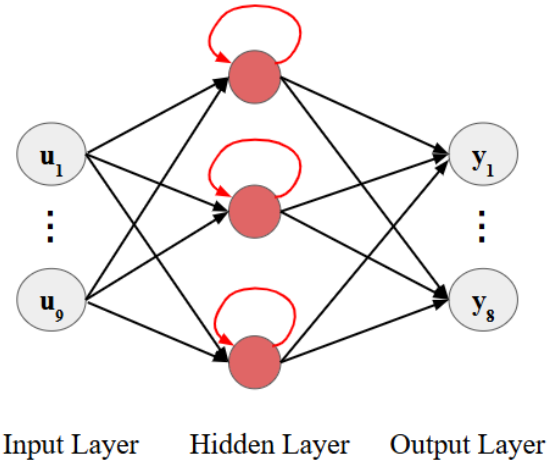


Figure 2.2: Recurrent Neural Network

2.2.1 Long Short-Term Memory - LSTM

LSTM is a type of RNN commonly used for time series and sequential data analysis and can map complex nonlinear dynamics. While plain RNNs struggle to learn long term dependencies due to the vanishing gradient problem, the LSTM introduces internal gates that allows it to remember or discard information over long periods [7].

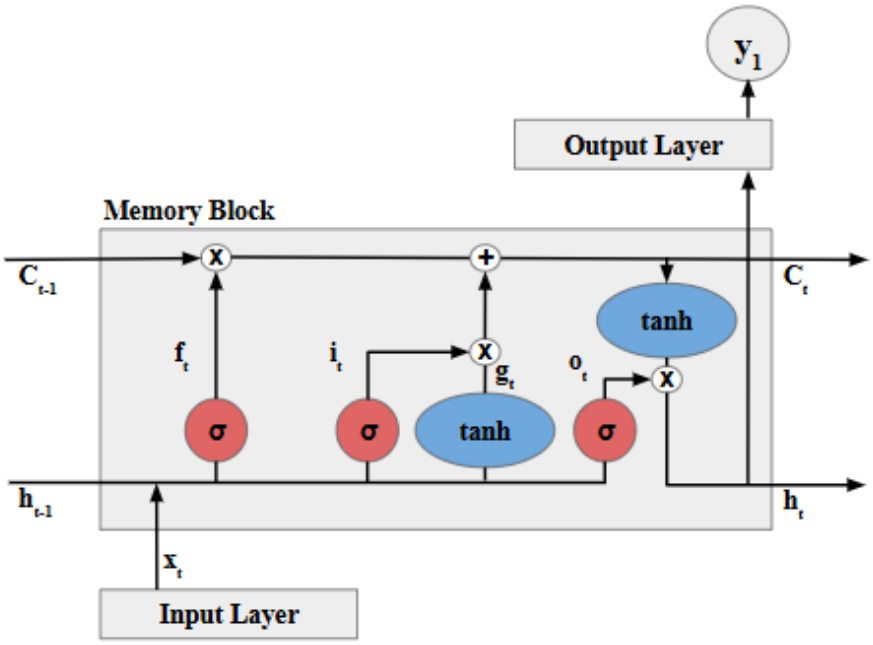


Figure 2.3: LSTM Network (one cell)

Figure 2.3 shows a typical LSTM network, a single pass (i.e. movement of input data through the internal hidden layers to the output layer), where x_t represents the input, C_t represents the long term memory and h_t represents the short term working memory. The cell state C_t holds all dimensions of historical data. h_t should only hold the information that is relevant at the current time-step [16]. The input to the LSTM is in the form of sequences instead of a single input vector, in order for it to learn the long term dependencies of the data and not simply a single relation map. Hence, the LSTM works with tensors.

In Equations 2.6 through 2.8 below, it is shown how each layer computes the output from the input sequence, with the weights W and the biases b updated at each iteration. For example, W_{ii} denotes the weight for the input (state, x_t) connected to the input gate (i_t), W_{if} denotes the weight for the input (x_t) connected to the forget gate (f_t) [16]. In the following equations, t denotes a discrete time (sequence) index.

$$\begin{aligned} C_t &= f_t \cdot C_{t-1} + i_t \cdot g_t \\ h_t &= o_t \cdot \tanh(c_t) \end{aligned} \tag{2.6}$$

$$\begin{aligned} i_t &= \sigma(W_{ii} \cdot x_t + b_{ii} + W_{hi} \cdot h_{t-1} + b_{hi}) \\ f_t &= \sigma(W_{if} \cdot x_t + b_{if} + W_{hf} \cdot h_{t-1} + b_{hf}) \end{aligned} \tag{2.7}$$

The current cell state C_t passes through the tanh function so that the values ends up between 1 and -1 preventing the maths to explode during training. The forget gate (f_t) acts as a filter that decides which parts of the previous long-term memory are no longer useful and should be discarded. At the same time, the input gate (i_t) looks at the new incoming data and decides which parts are important enough to be added to the long term memory. Then combining these two, the model updates its long term understanding (C_t) to keep only the most valuable information [5], [16]. Finally, the output gate (o_t, a sigmoid activation function shown in Equation 2.8) decides what piece of information in C_t is relevant right now. If the output gate outputs 0 (gate closed) for a specific dimension, that piece of memory is hidden from h_t, and if it outputs 1 then the piece of memory is sent to the linear layer and used in the temperature prediction. The output gate acts as a "gatekeeper" that translates this updated long term memory into the short term memory (h_t), which is what the model actually uses to make its prediction. g_t represents the updated new state, as in Equation 2.8 [5], [16].

$$\begin{aligned} o_t &= \sigma(W_{io} \cdot x_t + b_{io} + W_{ho} \cdot h_{t-1} + b_{ho}) \\ g_t &= \tanh(W_{ig} \cdot x_t + b_{ig} + W_{hg} \cdot h_{t-1} + b_{hg}) \end{aligned} \tag{2.8}$$

Finally, the network produces the prediction y_t by projecting the hidden state h_t through a final linear output layer, and the because of the tanh and sigmoid functions the relation to the input is nonlinear.

The training mechanism for an LSTM network is through a gradient based algorithm, where it learns through Backpropagation Through Time (BPTT) to address the vanishing gradient problem [7]. The network calculates the gradient of the error and propagates it backward through the sequence window to update the networks

weights. Figure 2.3 illustrates the forward-pass architecture of an LSTM. Following the forward pass, MSE loss calculation, BPTT and an Adam optimizer minimizeing the loss and optimize the LSTM weights W and b [17]. One full pass of the entire training dataset through this process is defined as an epoch.

The number of layers in the LSTM network refers to how many LSTM cells are stacked one after another. When there are multiple layers in the network, the output of the first layer becomes the input to the second one. The first layer captures more immediate dynamics and the second one should identify more long term trends. Adding more layers can make the model learn more complex dynamics, however it also increases the risk of overfitting to noise. The number of hidden units (the hidden size) refers to the number of units within each LSTM cell. It represents the number of cells (neurons) available to store information at that point in time. If the hidden size is too small, the model will forget information because it runs out of memory. However, if it is too large the model will memorize noise without learning the physics and overfit the data [16].

The number of training epochs can be determined by either an early stopping algorithm, or by an iterative process by observing after how many epochs the loss curve starts to level out, i.e. when the MSE stops decreasing. The batch size is how many samples (datapoints) there are in each batch, and the baches determine how many iterations per epoch are made. If the batch size equals the amount of samples, there will be one iteration per epoch. If the batch size is half the number of samples, there will be two iterations per epoch. The learning rate controls how much of the models weights are adjusted during training [18].

In terms of tuning these hyper-parameters, this can be done by observation of the loss curve to determine a balanced network between overfitting and underfitting to the data. To avoid overfitting, a dropout layer can be added which randomly deactivates a fraction of cells during training, forcing the network to learn the dynamics rather than just memorizing the training data [16].

3
Methods

To formulate an LSTM-MPC and evaluate its performance compared to a classic FOPDT-DMC controller, Python with PyTorch [16] and other relevant packages such as SciPy Minimize [19] was used. Data was collected from VAROPreems database, processed and filtered, and then used to train an LSTM model for use in an MPC controller. A step() function using the currently implemented FOPDT step response model was formulated, to be used as a simulation environment for controller evaluation, acting as the "real" model of the plant. A baseline MPC was developed to be similar to the currently implemented DMC, using the FOPDT model as prediction model as well as the currently implemented penalty weights and horizons.

Four main MPC controller versions were formulated: (1) the Baseline MPC, to be used as a reference for performance evaluation; (2) a synthetically trained LSTM-MPC, to evaluate the LSTM prediction models performance without the closed loop data problem; (3), the main LSTM-MPC, trained on historical process data from H2201; and (4) a hybrid LSTM-MPC where the FOPDT was used in combination with LSTM predicted residuals.

3.1 Model Formulation - Passbalancing H2201

As described in Section 1.1 (Table 1.1), the pass balancing problem for furnace H2201 was formulated using nine MVs and eight CVs. It can therefore be interpreted as a multivariable control problem, with couplings between inputs and outputs [11]. The aim is to formulate an LSTM model to predict the trajectory of eight pass outlet temperatures $T_i(k)$, $i = 1, ..., 8$ and use the error to formulate a new input signal $F_i(k)$, $i = 1, ..., 8$. The temperature bias is formulated as a deviation from the average pass temperature, and the pass flow bias is formulated as the deviation from an eighth of the total inlet flow, i.e. not as a deviation from the average pass flow. In order to keep the total inlet flow constant, the sum of all pass flow biases at each time step needs to equal zero.

The bias-sum constraint is applied on the sum of the flow biases in the real historical data, in order to keep the total inlet flow constant. This means that the historical data will have correlation between passes. Hence, the system dynamics for each pass is not purely individual. If one pass has a too high temperature bias and the optimizer wants to lower it, it will try to increase the flow bias through that

pass. To keep the bias sum zero, the controller must then decrease the flow bias on other passes in the furnace, even if they also have a temperature bias that is too high.

The correlation problem as well as the active constraint in the historical closed loop process data leads to one less degree of freedom in the dynamic modeling. This means that the degrees of freedom are 7 instead of 8. A way to overcome this is to predict seven biases, and then reconstruct the eighth one.

Pass Temperature Bias

$T_i(k)$ denotes the temperature in pass i at time step k. The average temperature across all eight passes is defined as

$$\bar{T}(k) = \frac{1}{8}\sum_{i=1}^{8} T_i(k). \tag{3.1}$$

The temperature bias for each pass is then defined as the deviation from this average:

$$T_i^{\text{bias}}(k) = T_i(k) - \bar{T}(k), \quad i = 1, \ldots, 7. \tag{3.2}$$

Only the first seven temperature biases are considered as independent outputs. The eighth temperature bias is reconstructed using the zero-sum constraint:

$$\sum_{i=1}^{8} T_i^{\text{bias}}(k) = 0 \tag{3.3}$$

which gives:

$$T_8^{\text{bias}}(k) = -\sum_{i=1}^{7} T_i^{\text{bias}}(k). \tag{3.4}$$

Pass Flow Bias

$F_i(k)$ denotes the flow rate in pass i, and $F_{\text{total}}(k)$ denotes the total inlet flow to H2201. The average base-flow per pass is defined as

$$\bar{F}(k) = \frac{F_{\text{total}}(k)}{8}. \tag{3.5}$$

The pass flow bias is defined as a deviation from an eighth of the total inlet flow:

$$F_i^{\text{bias}}(k) = F_i(k) - \bar{F}(k), \quad i = 1, \ldots, 7. \tag{3.6}$$

Just like with the predicted temperature, the eighth pass flow bias (Pass H) is reconstructed using the zero-sum constraint:

$$\sum_{i=1}^{8} F_i^{\text{bias}}(k) = 0 \tag{3.7}$$

which gives:

$$F_8^{\text{bias}}(k) = -\sum_{i=1}^{7} F_i^{\text{bias}}(k) \tag{3.8}$$

This method is only needed when training the LSTM on the real historical closed loop process data. In the baseline formulation, using the FOPDT model, the bias-sum constraint is not active in the step response test since the model is formulated considering a change in one pass at a time. In the baseline formulation, the pass flow bias and temperature bias are simply calculated using Equations 3.2 and 3.6.

Furthermore, limits on how fast the pass flows are allowed to change per time step was considered as a constraint in the model (e.g., cannot change the flow more than ± 0.5 per minute). However, the limit on the total valve position or the total flow bias are left to be constrained in the DSC and not included in the model. Other constraints, such as turning off the external controller when the temperature bias for a pass is too high, is also left to the DCS.

3.2 Historical Process data from H2201

In order to train the LSTM prediction model, historical process data was extracted from the database used at VAROPreem. The data available from the temperature and flow sensors in the H2201 system reaches three years back and has a sampling time of 20 seconds, i.e. three samples per minute. The CVs and MVs from which historical data was extracted are shown in Table 3.1.

3.2.1 Data Extraction

The historical process data was extracted to Excel using the company's database Add-On. Extracting data with 1 minute interpolation, the files were then arranged with 6 months of data in each file, converted to CSV files for easier handling using Python. The following tags in Table 3.1 were extracted from the company database. The MODE columns were converted to numbers from string values for filtering possibilities in Python.

Table 3.1: Historical Process data extraction

Tags	Comment
TIME	Time stamp for each data sample ($T_s = 1$ minute)
TI4	Temperature measurement for Pass 1
⋮	⋮
TI11	Temperature measurement for Pass 8
FC_{master}	Total flow (sum of all pass flows)
FC1	Flow through pass 1
⋮	⋮
FC8	Flow through pass 8
$FC_{master}.MODE$	Operating mode of the total flow controller
FC1.MODE	Operating mode of pass 1
⋮	⋮
FC8.MODE	Operating mode of pass 8

3.2.2 Data Pre-Processing

The CSV files with the H2201 data contains data from several different operating modes, which are described in Chapter 1 in Table 1.1. The plant could have been under complete shutdown for a period of time, or restarting from a trip or simply running a single pass in manual mode for whatever reason. The LSTM should be trained on normal operating conditions to learn the plant physics where the furnace and pass balance is running normally, since it may otherwise learn behavior from for example shutdowns where the physics may be different due to burner setup and so on. To ensure reliable and relevant training and testing data for the LSTM, the following steps were taken.

1. The data where the MODE is in not in ROUT (normal operating modes from Table 1.1) was filtered out.
2. To make sure data that the LSTM trains on is within the usual operating range, data outside of the alarm limits in Table 1.3 is filtered out.
3. NaN (not a number) data points were removed.
4. Limit (*gap_interpolation_limit*) on how many time steps that can be interpolated through. Because the training data in this thesis was sampled with a 1 minute interval and the dead time and time constant are only a few minutes each, a maximum 3 minute interpolation gap is accepted.
5. The data was divided into sequences X_{seq} and Y_{seq} of length *sequence_length* since the LSTM takes time series in the form of sequences (sliding windows). Because of the time constant and dead time of the process, a sequence length of 60 minutes was used. This would give enough time to capture the full system dynamics.
6. Any sequences in which gaps in the data is larger than the *gap_interpolation_limit*

were removed. If large gaps in data (filtered out in either step 1, 2 or 3) was interpolated, the model wold make an assumption that the data between two time steps has a gradient that is not necessarily the correct one, which would lead to the LSTM learning on incorrect data.

It is however fine for the LSTM to not take sequences that are connected to each other, since the length of a sequence should cover the system dynamics (dead time, delay and gain). Long term dynamics will be handled by the MPC through disturbance estimation.

7. The raw temperature measurements and pass flow measurements are recalculated to biases as in Section 3.1.
8. The data was scaled using RobustScaler or StandardScaler from Scikit Learn [20].
9. The remaining data is divided into 80% training and 20% testing (validation) data.

After the data has been used for training it is converted back from PyTorch tensors into NumPy arrays as well as being un-scaled.

3.3 Simulation Environment: H2201 Digital Twin

To validate the proposed LSTM controllers as well as for comparison to the classical controller, a simulation environment was built based on a step response model of the plant. As the FOPDT matrix for the MIMO (Multi Input Multi Output) system format in Figure 1.2 in Chapter 1, a 9x8 Gain Matrix (K), 9x8 Time Constant Matrix (τ) and 9x8 Dead Time Matrix (θ) were used in the FOPDT formulation of the dynamics of the plant. The plant model was discretized and a pass temperature initial condition (intial temperature offsets) of $[4.0, 3.0, 2.0, 1.0, -1.0, -2.0, -3.0, -4.0]$ was defined for a benchmark for comparison. This creates a clear controller goal in the simulation; to bring these temperature bias values towards zero, making it simple to visually see how the controller handles the challenge.

First, the continuous time formulation of the plant was defined. For a single input-output pair, a transfer function is described in the Laplace domain as a FOPDT model [11]:

$$G(s) = \frac{Y(s)}{U(s)} = \frac{Ke^{-\theta s}}{\tau s + 1} \tag{3.9}$$

Then, since the MPC controller operates digitally and since the data collection in this thesis is interpolated to 1 minute sampling intervals ($T_s = 1$), the transfer function must be discretized:

$$\alpha = e^{-T_s/\tau} \tag{3.10}$$

$$K_{\text{discrete}} = K(1 - \alpha) \tag{3.11}$$

This leads to a recursive difference equation:

$$y_k^{i,j} = \alpha^{i,j} \cdot y_{k-1}^{i,j} + K_{discrete}^{i,j} \cdot u_{k-\theta^{i,j}}^{i} \tag{3.12}$$

Then the final temperature of pass j is calculated as:

$$y_k^j = \sum_{i=1}^{8} y_k^{i,j} + e^j_{\text{gaussian},k} \tag{3.13}$$

Where

$$e^j_{\text{gaussian},k} \sim \mathcal{N}(0, \sigma^2). \tag{3.14}$$

A Gaussian measurement noise has been added to make the simulation environment more realistic and to evaluate controller robustness. Such noise models are commonly used in system identification and discrete time stochastic modeling to represent sensor uncertainty and unmodeled dynamics [12]. This digital twin of H2201 was used as the plant in the MPC loops as well as the basis for the prediction model in the baseline MPC controller.

3.3.1 Generating Synthetic Plant Data

Because of difficulties overcoming the closed loop data problem when training the LSTM to be used as prediction model in the MPC controller, a synthetic version of the LSTM model was made. This model was trained on data generated by the FOPDT model, meaning it was trained on data that would approximately equal open loop data from H2201 (with valves in manual mode and no feedback). To generate synthetic open loop process data, the MIMO FOPDT Simulation Plant (the digital twin of H2201) python file is used. However, the synthetic data will be tweaked in order to introduce some variance to the values so that the LSTM learns on diverse data.

Large uniform random step changes are applied in order to capture high frequency response and cross-coupling [12]:

$$\Delta u_k^i \sim \mathcal{U}(-0.3, 0.3) \tag{3.15}$$

For evaluating settling time and delays [12], high amplitude pulses are added to the baseline

$$\Delta u_k^i = \begin{cases} \mathcal{U}(-0.4, 0.4) & \text{if } k \pmod{50} = 0 \\ \mathcal{U}(-0.05, 0.05) & \text{else} \end{cases} \tag{3.16}$$

To simulate gradual stochastic variations in the input data, a Gaussian random walk was used. This process is the discrete time equivalent of Brownian motion (Wiener process) [21]:

$$\Delta u_k^i \sim \mathcal{N}(0, \sigma^2), \quad \sigma = 0.05 \tag{3.17}$$

To provide data that can be used in the LSTM training, the data is captured via a sliding observation window to generate a tensor of windows. For the defined sequence length (N=60 minutes), X_k^{seq} and Y_k^{seq} are produced as follows:

$$X_k^{\text{seq}} = \begin{bmatrix} U_{k-N+1} & D_{k-N+1} & Y_{k-N+1}^{\text{meas}} \\ \vdots & \vdots & \vdots \\ U_k & D_k & Y_k^{\text{meas}} \end{bmatrix} \tag{3.18}$$

$$Y_k^{\text{seq}} = Y_{k+1}^{\text{meas}} \tag{3.19}$$

where U is the vector of manipulated flows, D is the measured disturbance, and Y^{meas} is the current measured temperature implemented in bias space instead of full range flow and temperature measurements.

3.4 MPC Formulation

The proposed MPC controllers follow a receding horizon principle where the optimizer minimizes the objective function that penalizes trajectory error (temperature bias), control moves (flow bias) and sum of flow bias sum deviation to zero, subject to constraint on flow adjustments per step. For the baseline classic MPC, the linear FOPDT prediction model with a Sequential Quadratic Programming (SQP) cost function is used. For the LSTM-MPC, the nonlinear LSTM prediction model is used with a SQP cost function for non-convex optimization. Algorithm 1 describes the MPC which was implemented in Python, using SciPy 'SLSQP' to minimize the objective function [19].

Algorithm 1 MPC formulation

1: **Inputs:** Current state x_k, setpoint r, penalty weights Q. R and S, an LSTM model or FOPDT model. Nudge ΔU_0 so that the optimizer does not get stuck.
2: **while** system is running **do**
3: Measure current output y_k
4: **Disturbance estimation:**
5: Update d_k using filtered residual $e_k = y_k - \hat{y}_k$
6: Initialize with previous control sequence ΔU_{k-1} (warm start)
7: **Optimize:**
8: Solve $\Delta U^* = \arg\min J$, where:

$$J = \frac{Q}{H_p} \sum_{k=1}^{H_p} \|y_k - r\|^2 + R \sum_{k=1}^{H_c} \|\Delta u_k\|^2 + \frac{S}{H_p} \sum_{k=1}^{H_p} (\sum_{i=1}^{8} u_{i,k})^2$$

9: **s.t**
10:

$$\Delta u_{\min} \leq \Delta u \leq \Delta u_{\max}$$

11: Compute $F_8 = -\sum_{i=1}^{7} F_i$ (for LSTM real data version)
12: Execute first optimal control move Δu_0^* and shift horizon
13: **end while**

The baseline FOPDT-MPC was tuned in line with the currently implemented DMC controller, with a prediction horizon of $H_p = 20$ and $H_c = 9$. The penalty weights

are defined as diagonal matrices, with a constant diagonal of 1.0 for trajectory error (Q), 0.1 for control moves (R), 10.0 penalty for bias sum deviation from zero (S), hence, a high priority on keeping the flow bias sum close to zero to avoid drift.

For the LSTM-MPC tuning, the first iteration of the controller began by choosing the same MPC parameters as the baseline model, however further tuning was required for the different LSTM-MPC versions. Additional logic was added to the MPC to improve performance and stability, including feedforward disturbance, offset correction using bias disturbance estimation and integral action.

To ensure controller consistency when adjusting the prediction horizon H_p, the temperature and bias-sum penalties were normalized by H_p, ensuring the relative importance of them remaining constant, regardless of how far into the future the controller plans.

3.4.1 Feed-Forward Disturbance Control

The total flow to the furnace is a measured disturbance and was hence used for feedforward, meaning disturbances could be anticipated and accounted for. Therefor, the total flow to H2201 was included as an input to the LSTM prediction model in training, so that the model could learn how the total inlet flow affects the pass balancing dynamics.

During the closed-loop simulation, the feedforward disturbance is measured and appended to the historical input window. When predicting the future trajectory over (H_p), the measured disturbance is held constant at its most recently measured value throughout the simulated future window. Because the LSTM model has learned the relation it predicts how the temperatures will respond to the current measured disturbance. The optimizer does not have to wait for the tracking error e_k to be calculated, the optimizer can help adjusts the valves in advance.

3.4.2 Feedback Offset Correction

Even with feed-forward estimation, steady state error may occur in the simulation. The aim of offset correction is therefore to make the steady state error go to zero. This error includes both model-plant mismatch as well as process disturbances that appear over time.

Offset correction can be done using different techniques, such as disturbance estimators and (forced) integral action. Bias disturbance estimator (d_k) adjusts the model prediction, and d_k is added to the LSTMs future prediction horizon. An "integral action" adjusts the control decision based on accumulated error. It is implemented as an additional penalty in the MPC cost function, so that accumulated error (a permanent offset) is penalized.

Bias Disturbance Estimation

In order to adjust the model prediction based on the real plant output, a disturbance estimator is used in the MPC formulation to supply a disturbance estimation model d_k. This disturbance model is then added to the prediction to achieve a better next prediction. There are two types of noise to either include in the disturbance model or filter out; process noise (a physical change to the plant, such as buildup in the pipes, etc.) and measurement noise (for example high frequency disturbance in the measurement equipment). An observer (a state estimator) was used, specifically a First-Order Low-Pass Filter, according to Algorithm 2 to approximate a bias disturbance model. Though a Kalman filter is often used for this purpose, because the model is nonlinear and not represented in state space form, the low pass filter was chosen [10]. The low pass filter is also illustrated in Figure 2.1.

Algorithm 2 Disturbance Estimation with Low-Pass Filtering (Bias Estimator)

1: **Initialize:** Disturbance estimate $d_0 = 0$
2: Define filtering factor $\alpha_d \in (0, 1)$
3: **for** each control step k **do**
4: Measure process output y_k
5: Compute model prediction $\hat{y}_k$ (without disturbance correction)
6: Compute raw prediction error:

$$e_k = y_k - \hat{y}_k$$

7: **Low-pass filtering the disturbance**
8: Update disturbance estimate:

$$d_{est,k} = (1 - \alpha_d)\, d_{est,k-1} + \alpha_d\, e_k$$

9: Correct/augment model prediction:

$$\hat{y}_k^{aug} = \hat{y}_k + d_{est,k}$$

10: Use $\hat{y}_k^{aug}$ in MPC prediction and optimization
11: Apply first control action to the plant
12: **end for**

Row eight of Algorithm 2 shows the First-Order Low-Pass Filter equation, where the filter constant α decides how much of the disturbance model should filter out high frequency disturbance such as measurement noise without removing the process disturbance or model inaccuracy. A low α, such as $\alpha = 0.2$, makes the controller robust to high frequency sensor noise. If the controller takes too long to return to the setpoint after a disturbance is injected, α may need to be increased.

Integral Action

While the disturbance estimator corrects the model's internal baseline, it does not penalize the optimizer for remaining at a slight offset. To account for error accumulation over time, a forced integral action through added optimizer penalty was implemented according to Algorithm 3 to remove steady state offset and get knowledge of unmeasured disturbances [8].

Algorithm 3 MPC with Integral Action and Dead band

Initialize: Integral state $I_0 = 0$
Define dead band $\delta = 0.05$
Define integral gain $K_i = 0.5$
Define integral limits $I_{\min} = -10, I_{\max} = 10$
for each control step k **do**
    Measure process output y_k
    Compute control error:

$$e_k = r_k - y_k$$

    Deadband formulation
    if $|e_k| > \delta$ **then**
        Update integral state:

$$I_k = I_{k-1} + K_i e_k$$

    else
        Decay integral state:

$$I_k = 0.8 I_{k-1}$$

    end if
    Anti-windup saturation:

$$I_k = \text{clip}(I_k, I_{\min}, I_{\max})$$

    Solve MPC optimization problem using:

$$e_k + I_k$$

    as the augmented tracking error
    Apply first control action to the plant
end for

Persistent errors is summed and will cause increased penalty in the MPC cost function, because now the total error $e_k + I_k$ augmented with I_k is squared and penalized in the MPC optimizer for tracking error. At each control step k, the tracking error $e_k = r_k - y_k$ is calculated and if the error is significant (larger than an allowed range, 0.05 chosen for H2201 pass-balancing problem), it is added to the running integral state using a deadband δ. The dead band is added because if we are close enough to setpoint we are fine without it and want to avoid oscillations. If the system is within the allowed dead band, the integral state is instead decaying to relax previous

accumulation. The integral gain is tuned to balance oscillations.

If the the controller suggest a change that reaches an actuator limit, the accumulated integral action will grow toward infinity. To prevent this, an anti-windup clipping function was used that bounds the integral state between $I_{\min}$ and $I_{\max}$. Integral action with too high gain can cause oscillations, so tuning was required in the simulation environment.

3.5 LSTM Implementation

The furnace H2201 has slow thermal dynamics as well as dead time after actuator adjustments, and therefore the LSTM memory gates comes in use. The nonlinearity of the LSTM network leads to a nonlinear QP optimization problem, and therefore a nonlinear solver is needed. In this thesis, Sequential Least Squares Programming (SLSQP) solver from the SciPy library was used for the nonlinear MPC formulation.

3.5.1 LSTM model training using PyTorch

To prepare data for the LSTM training, a sliding window technique was used. The historical furnace data is transformed into sequences, where each input tensor X_{seq} contains a window of past valve positions, disturbances, and temperatures (the last 60 minutes as per the defined sequence length). The model then outputs a prediction for the next time step. Once trained, the MPC controller feeds a sequence of future valve moves into this model, and the LSTM returns the predicted future temperature trajectory.

In the FOPDT-MPC, the previous temperatures are not needed as inputs because the model assumes the system is linear and stationary. However, in the LSTM-MPC is nonlinear and needs to know the exact state of the furnace to determine what will happen next, i.e. it needs memory to use in its internal gates. Therefore, instead of providing 9 inputs and 8 outputs like the FOPDT does, it processes an input vector of 17 features with 8 flow biases, the total flow FC_{master}, and the 8 current temperature biases (at t-1). Feeding back these 8 temperature inputs is needed because the LSTMs response is dependent on the system current state. Therefore, the inputs and outputs used in the training of the LSTM are listed in Table 3.2, in comparison to the inputs and outputs to the FOPDT-MPC in Table 1.2 (although these are still the MVs and CVs of the system).

Table 3.2: Inputs and Outputs to the LSTM network

Inputs		**Outputs**	
Variable	**Time**	**Variable**	**Time**
$FC1_{BIAS}$	t	$TI4_{BIAS}$	t
$FC2_{BIAS}$	t	$TI5_{BIAS}$	t
$FC3_{BIAS}$	t	$TI6_{BIAS}$	t
$FC4_{BIAS}$	t	$TI7_{BIAS}$	t
$FC5_{BIAS}$	t	$TI8_{BIAS}$	t
$FC6_{BIAS}$	t	$TI9_{BIAS}$	t
$FC7_{BIAS}$	t	$TI10_{BIAS}$	t
$FC8_{BIAS}$	t	$TI11_{BIAS}$	t
FC_{master}	t		
$TI4_{BIAS}$	$t-1$		
$TI5_{BIAS}$	$t-1$		
$TI6_{BIAS}$	$t-1$		
$TI7_{BIAS}$	$t-1$		
$TI8_{BIAS}$	$t-1$		
$TI9_{BIAS}$	$t-1$		
$TI10_{BIAS}$	$t-1$		
$TI11_{BIAS}$	$t-1$		

Hyperparameter Tuning

In regards to tuning the training parameters for the LSTM training, an iterative method was used. However, possible less tedious methods for this are discussed in Chapter 4.

Addressing Closed-Loop Data Challenges

Because the available historical process data is data from a unit that is under closed loop control and kept in steady state by the active controller, the physical response to any changes in the system is suppressed. For the LSTM to learn the system physics, i.e. if the flow through a pass increased the temperature should decrease due to a shorter residence time, it needs to learn that by seeing past the correlation between the passes caused by the controller balancing them under constraints.

- **Autoregressive Multi-Step Training**
  Instead of training the LSTM to predict only a single step ahead, autoregressive training is used. The model is trained to predict a multi step trajectory 15 time steps into the future (so called free flight steps, length chosen to encompass the dead time and time constant), to learn long time dynamics of the system and thereby potentially overcome the controller caused dynamics

and correlations [7].

- **Excitation in training data**
  To ensure the model learns the system dynamics the training data must show some excitation or at least show low signal to noise ratio. Because the system is in equilibrium most of the time, a filter was added to the LSTM training data in the training file, where data where the flow actually changed is kept and data with low activity (in terms of standard deviation) is filtered out, hence improving the signal to noise ratio [13].

Algorithm 4 Autoregressive LSTM Training

Input: Training data, sequence length $T = 60$, free flight $T_f = 15$
Initialize LSTM parameters θ
for each epoch **do**
  for each training batch (X, Y) **do**
    Reset optimizer gradients
    Warm-up phase (teacher forcing)
    for $t = 1$ to $T - T_f$ **do**
      Propagate true inputs through the LSTM
    end for
    Autoregressive prediction phase Free Flight
    for $t = T - T_f + 1$ to T **do**
      Predict future temperatures $\hat{y}_t$
      Feed predictions back into the next input sequence
    end for
    Compute trajectory loss:

$$L_{\text{traj}} = \text{MSE}(\hat{Y}, Y)$$

    Compute final-step loss:

$$L_{\text{final}} = \text{MSE}(\hat{y}, y)$$

    Compute total loss:

$$\mathcal{L} = 0.7L_{\text{traj}} + 0.3L_{\text{final}}$$

  (tunable)
    Backpropagate loss through time (BPTT)
    Update parameters using Adam optimizer
  end for
end for

3.5.2 Absolute vs differential LSTM training

The LSTM can be trained on either an absolute relation, where it learns if we have a certain flow we should expect a certain temperature:

$$y_{t+1} = f(x_t) \tag{3.20}$$

The LSTM can also be trained on differential data, which means it learns how a change in flow bias results in a change in temperature bias:

$$\Delta y_t = f(\Delta x_t) \tag{3.21}$$

$$y_{t+1} = y_t + \Delta y_t \tag{3.22}$$

3.5.3 Grey Box LSTM

In this thesis, two different versions of Grey Box modeling or physics guided LSTM are evaluated, to compare with simply physical or simply Black Box prediction models.

Physics-Informed Neural Network (PINN)

A PINN is trained based on the knowledge that if the flow bias is decreased, and predicted temperature goes down, a penalty should be added to the loss function. By adding such a penalty to the loss function, the LSTM needs to find a set of weights that agrees with the defined physics. The resulting model is more robust when dealing with noisy or limited sensor measurements, as the penalty counteracts physically impossible predictions [22]. The total training loss in the LSTM algorithm is then defined as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{data}} + \mathcal{L}_{\text{physics}}. \tag{3.23}$$

Physics-Guided Hybrid LSTM Model

To improve prediction accuracy while preserving physical consistency, a hybrid model is proposed. The FOPDT model acts as the physical predictor, capturing the system dynamics, while the machine learning model (the LSTM in this case) serves as a correction layer that learns the mismatch between the physical model and reality [23]. The total temperature prediction is then formulated as:

$$\mathbf{y}_{k+1}^{\text{hybrid}} = \mathbf{y}_{k+1}^{\text{FOPDT}} + \mathbf{y}_{k+1}^{\text{LSTM}} \tag{3.24}$$

3.6 Model and Controller Evaluation

To evaluate the performance and stability of the LSTM-MPC compared to the baseline DMC controller, it needs to be tested on unseen data. Both the LSTM model in regards to prediction loss (Mean Squared Error, MSE) as well as how it works in a closed loop control setting are evaluated. Instability with no convergence to setpoint as well as oscillations should be avoided for a safe and efficient control of the plant. The controller needs to be robust to changes in conditions and be able to handle disturbances well.

3.6.1 Model Validation

After training the LSTM network on a section of the historical process data, it is validated, or tested, on an unseen fraction of the historical data. The root mean square error is calculated using PyTorch $nn.MSELoss()$, and plotted over each training epoch:

$$\text{Loss} = \frac{1}{N}\sum_{i=1}^{N} |\hat{\mathbf{y}}_i - \mathbf{y}_i|^2 \tag{3.25}$$

If the training loss, as well as the testing loss, shows a downward trend and are within a reasonably small range from each other, the model is considered to have learned the data well. If the training loss is lower than the test loss, the model may be overfitting, meaning that it has adapted more closely to the training data than to unseen data, and therefore requiring adjusting the LSTM parameters. However, this does not tell if the model has learned the correct physics of the system, just that it has learned the data relations. Therefore, the model needs to be tested in a controller simulation on the plant.

3.6.2 Controller Validation

To evaluate the LSTM prediction model, the custom made simulation environment with the step() function was used, implemented as an online receding horizon control simulation. First, an open loop evaluation was made, where the optimizer generates an optimal control sequence over the entire horizon (H_p, H_c) without executing the first move. This reveals the controllers plans to take the system to setpoint, without applying any feedback etc. If the planned trajectory does not move the system toward the setpoint, the model is not fit for control, even if its predictive error is low. This step shows whether problems come from the prediction model or the controller logic, and was used for both trouble shooting and evaluation.

When the model predicts the temperature well in the open loop predictions, a closed loop simulation was performed. The control move is performed, and feedback is active. In this simulation, disturbance injections are made on the plant to evaluate the controller's performance to correct it. This also shows how the controller should be tuned in regards to penalty on input aggressiveness and reference trajectory, as well as the penalty for the flow bias-sum constraint in order to avoid drift.

Closed loop stability, robustness as well as computational time are important metrics in controller evaluation. If using a state space model, controller stability could be evaluated using formal proofs and theorems. However, since using a data driven model, these metrics were evaluated empirically instead.

To evaluate the controller robustness, i.e. investigating how the controller handles changes that the model has not seen before, a step test in a pass flow is introduced. The system is considered input-to-state stable if the temperature bias converges to zero without sustained oscillations or divergence.

The closed loop stability is evaluated in simulation where a starting bias is injected and the controller has the challenge of bringing the temperature biases to setpoint. This should happen without growing oscillations, without drift in the flow biases or diverging temperatures, and the temperatures should converge to zero, which can all be revealed by plotting the closed loop simulation.

To prove that the controller can operate in real time, a timer is added to the MPC formulation, evaluating computational efficiency.

4

Results

The results of the LSTM modeling using historical process data from H2201A as well as synthetic data from the FOPDT model are presented in this chapter. The LSTM-MPC controller and comparison to the baseline controller is shown through simulations and step tests. The setpoint is defined to bring the temperatures to zero deviation from the average temperature, and since the data is formulated into individual biases from this average the setpoint to reach is zero temperature bias.

4.1 Raw Historical Process Data from H2201

Figures 4.1 and 4.2 shows an extract of the raw furnace data, to get an idea of the dynamics. Its is clear that pass 5 and pass 6 has much harsher flow movements compared to the other passes. Also, it can be seen that there are some changes to the average temperature over time, although the changes are fairly slow.

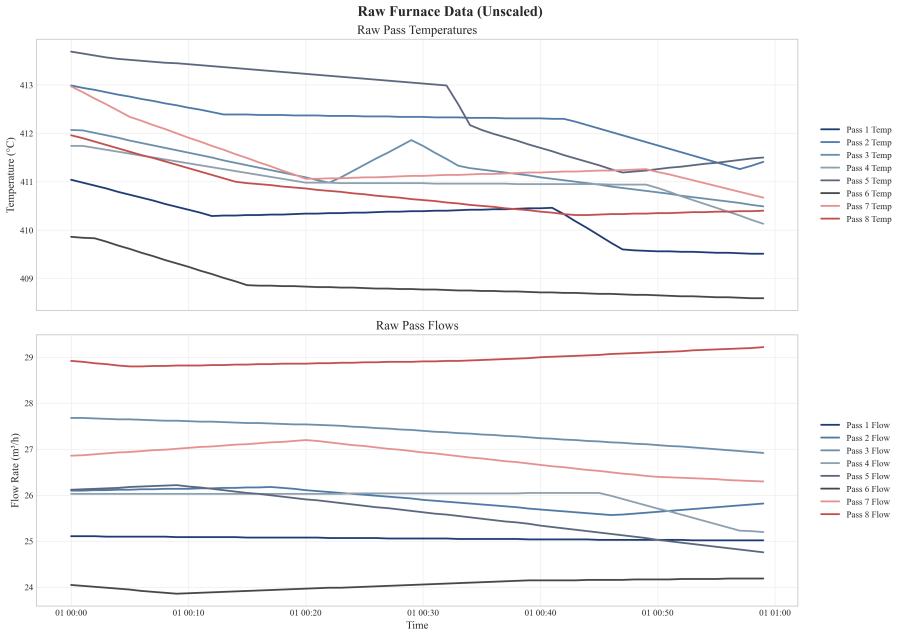


Figure 4.1: 60 minutes of raw data extract - Historical Process Data

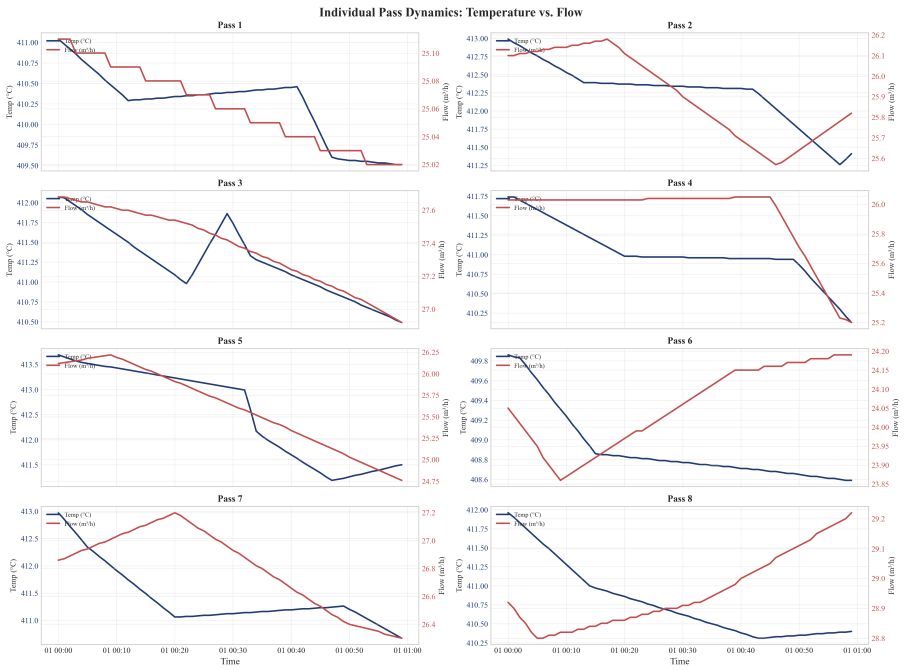


Figure 4.2: 60 minutes of individual temperature and flow bias versus time - Historical Process Data

To get an idea of a longer period of the furnace balancing, 1000 data points (about 17 hours) of raw historical process data is plotted in Figures 4.3 and 4.4.

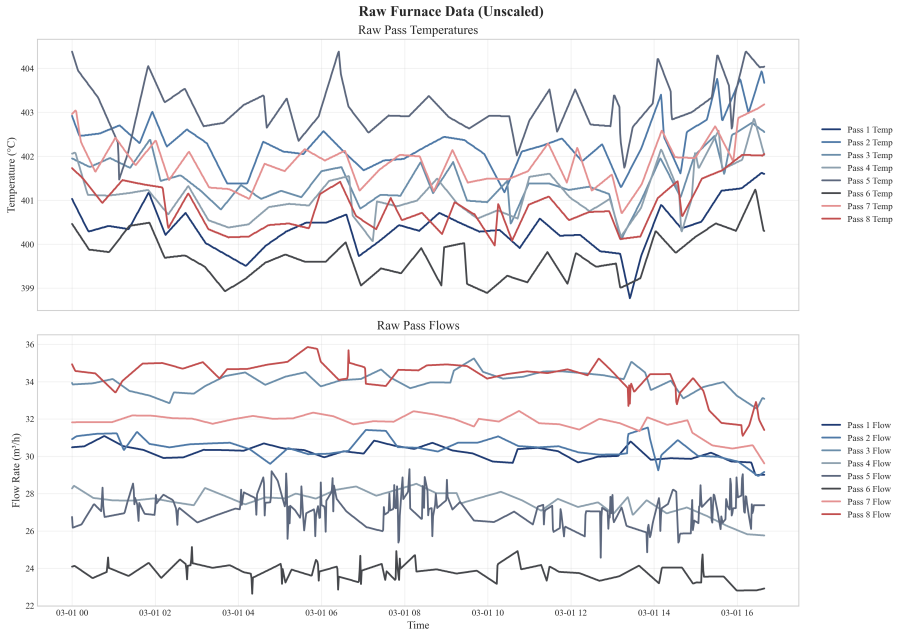


Figure 4.3: Raw data extract - Historical Process Data

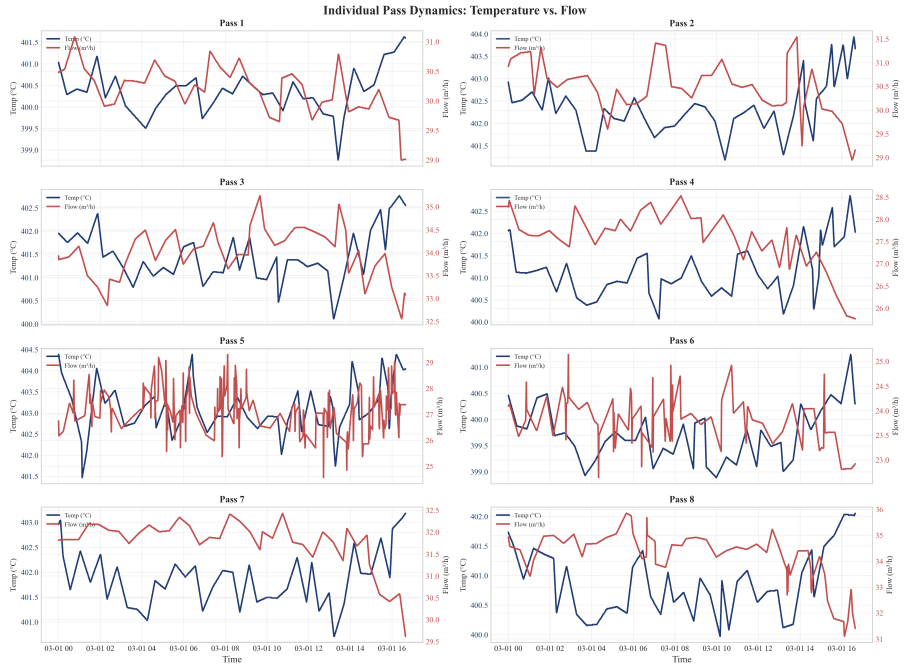


Figure 4.4: Individual temperature and flow bias versus time - Historical Process Data

4.2 Synthetic FOPDT generated data H2201

Synthetic data was generated using the FOPDT model described in Chapter 3 and has the following appearance over a 60 minute window. Notable is that the synthetic data is noisier than the historical process data, which is because the historical process data is in a controlled loop and thereby processed.

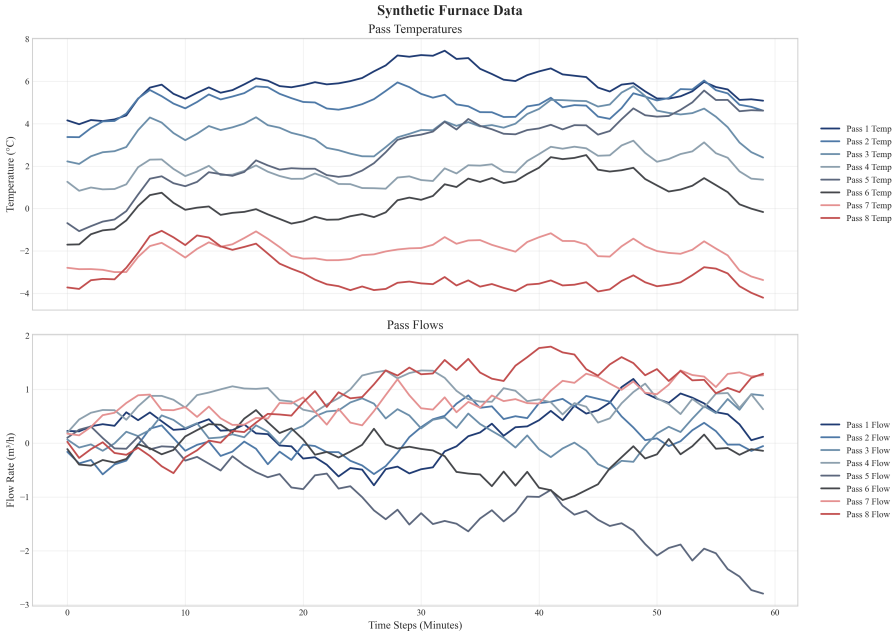


Figure 4.5: Synthetic Data extract

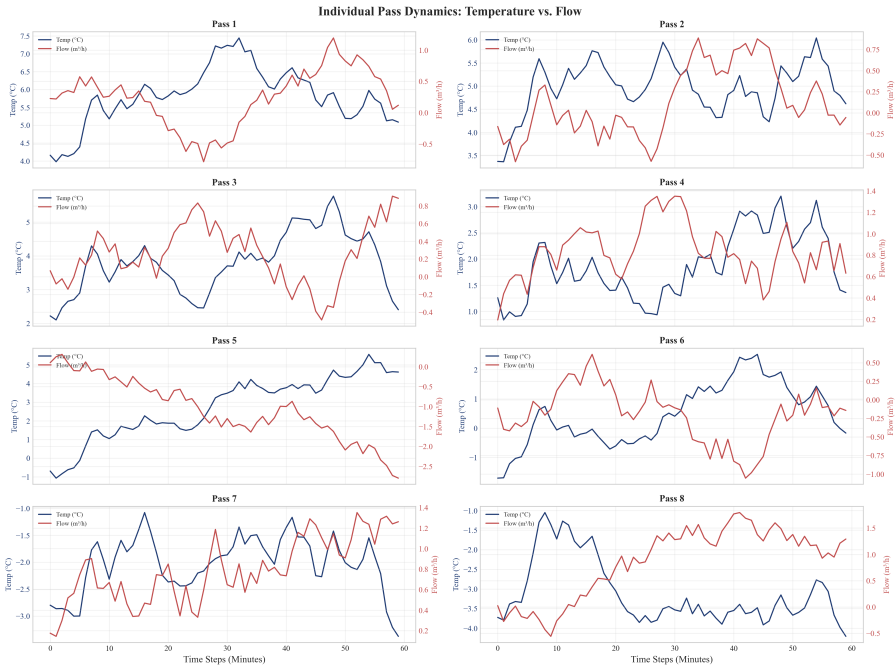


Figure 4.6: Individual temp and flow bias vs time

4.3 Baseline MPC - Ideal FOPDT model

The currently implemented DMC based on a step response FOPDT model act as a benchmark for comparison to the data driven MPCs that are formulated in this

thesis. The prediction model in the DMC loop is formulated by step response matrices generated by plant experiments, as described in Chapter 3. In the closed loop simulation, the plant model is based on the FOPDT model as well, however with added disturbance tweaks for synthesizing a less ideal case. Of course, this prediction and control performance of the baseline controller will still be highly ideal, since the plant model is so similar to the prediction model.

The MPC parameter that are stated in Table 4.1 are close to what is implemented in the real DMC controller. All pass temperatures are treated as equally important, hence the tracking error penalty (Q), the input penalty (R) and the bias sum penalty (S) are all diagonal matrices.

Table 4.1: Baseline model - MPC parameters

MPC Parameters				
H_p	H_c	**Q**	**R**	**S**
20	9	1.0	0.1	10.0

A starting point (initial condition) of temperature biases over the eight passes is [4.0,3.0,2.0,1.0,-1.0,-2.0,-3.0,-4.0] and the goal for the optimizer is hence to bring these to zero while abiding the constraints. The resulting baseline one step prediction curve (computing the plan over the prediction and control horizons) and closed loop 60 minute simulation curve are shown in Figures 4.7 and 4.8.

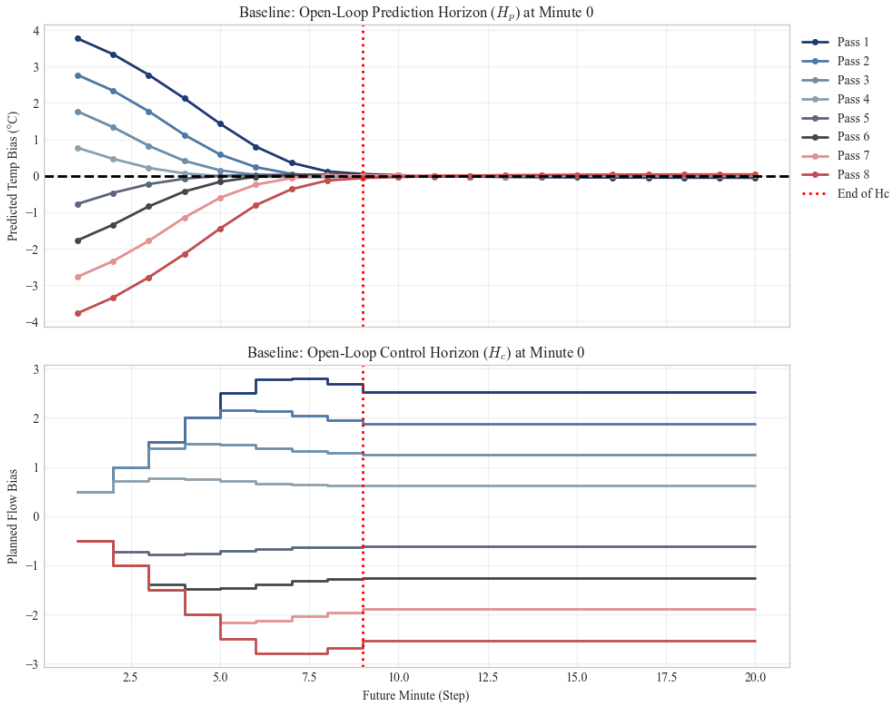


Figure 4.7: Baseline MPC first step H_c and H_p

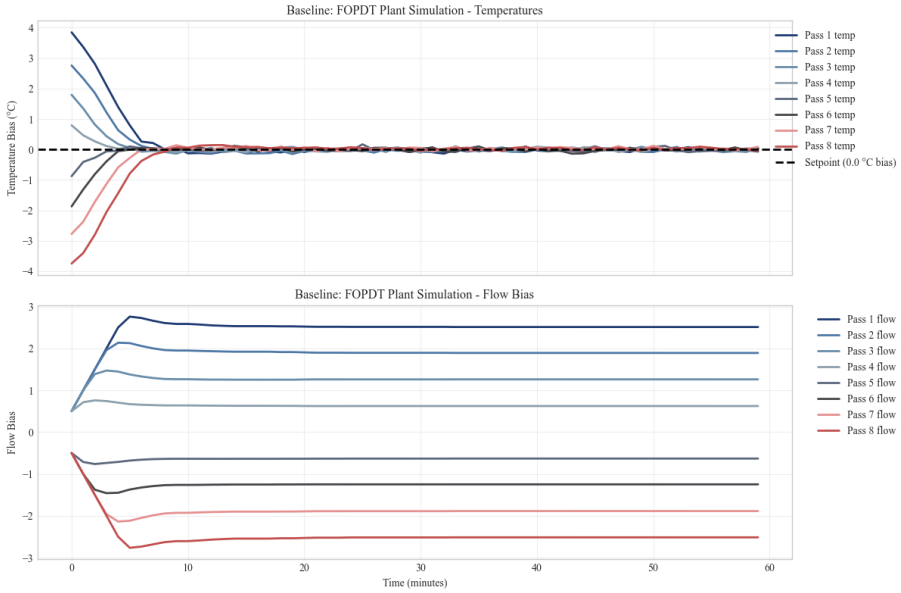


Figure 4.8: Baseline Closed Loop Simulation

A step on Pass 1 by adding a persistent temperature bias spike of +2 degrees Celsius was performed in order to evaluate controller sensitivity, shown in Figure 4.9. The controller brings the bias back to setpoint smoothly and timely while abiding the flow bias sum constraint.

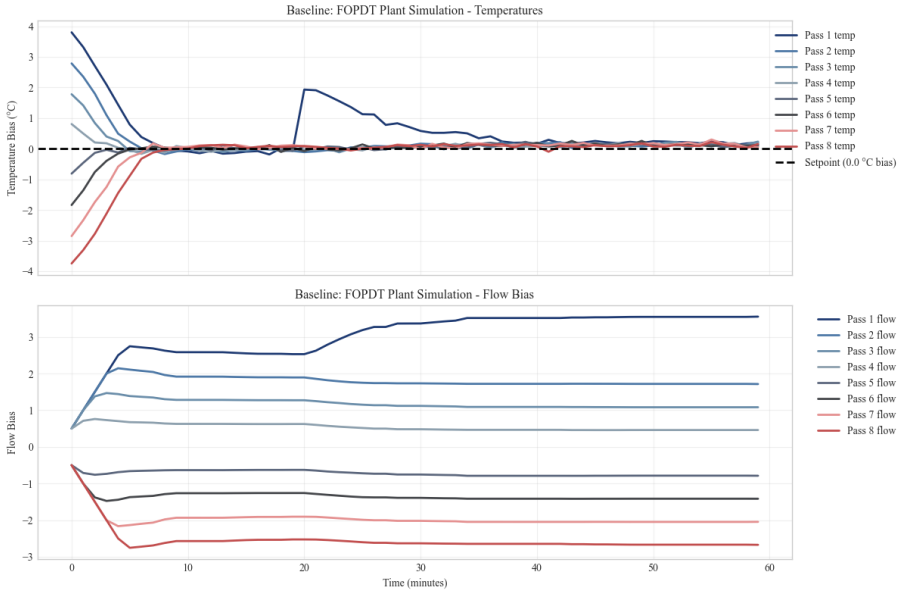


Figure 4.9: Baseline MPC: step test

A timer was added to the baseline MPC loop in order to evaluate computational efficiency, with the resulting simulation times stated in Table 4.2.

Table 4.2: Computational Time for 60 minute simulation - Baseline MPC

Timing Metrics			
Mean step time	**Max step time**	**Min step time**	**Total simulation time**
0.18 s	3.32 s	0.04 s	17.61 s

4.4 LSTM-MPC

4.4.1 Synthetic-LSTM MPC

To investigate the concept of an LSTM-MPC without the closed loop data issue, the synthetically generated FOPDT data is used for this version LSTM training. The LSTM training parameters and MPC parameters are listed in Table 4.3. A bach size of 256 was used.

Table 4.3: Model and MPC parameters for the Synthetic LSTM-MPC

Model Parameters				
Nr of Layers	**Hidden Size**	**Epochs**	**Learning Rate**	**Drop Out**
1	16	15	0.001	0.2
MPC Parameters				
H_p	H_c	**Q**	**R**	**S**
20	9	1.0	0.1	10.0

The LSTM trained on synthetic data gave the MSE loss vs training epoch, shown in Figure 4.10, apparently learning the data well.

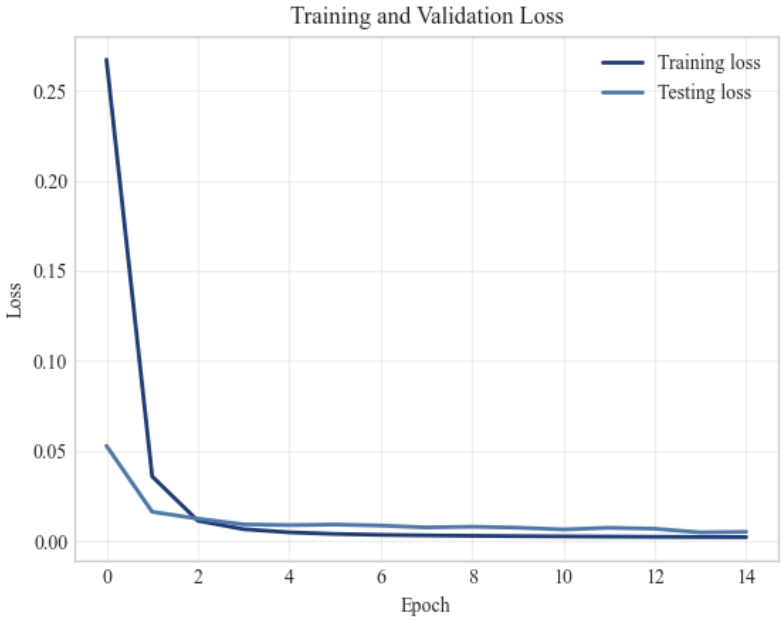


Figure 4.10: LSTM trained on synthetic data - loss curve

Using the synthetically trained LSTM as prediction model in the MPC loop results in the open loop predictions and the closed loop simulation shown in Figures 4.11 and 4.12, applying the same initial condition as in the baseline case. The resulting controller manages to minimize the temperature bias and bring it to setpoint.

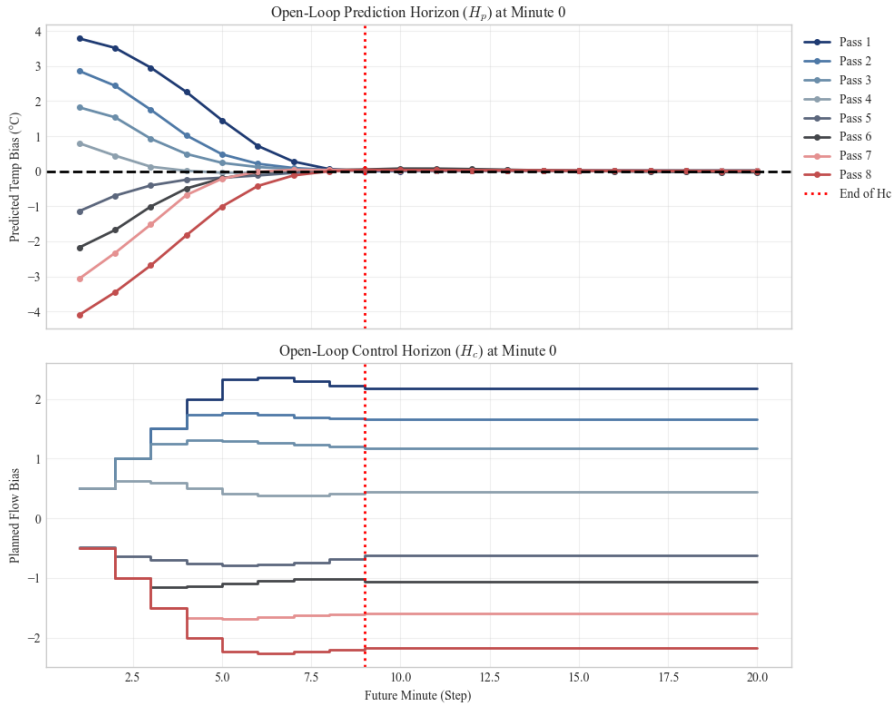


Figure 4.11: LSTM trained on synthetic data - MPC first step H_c and H_p

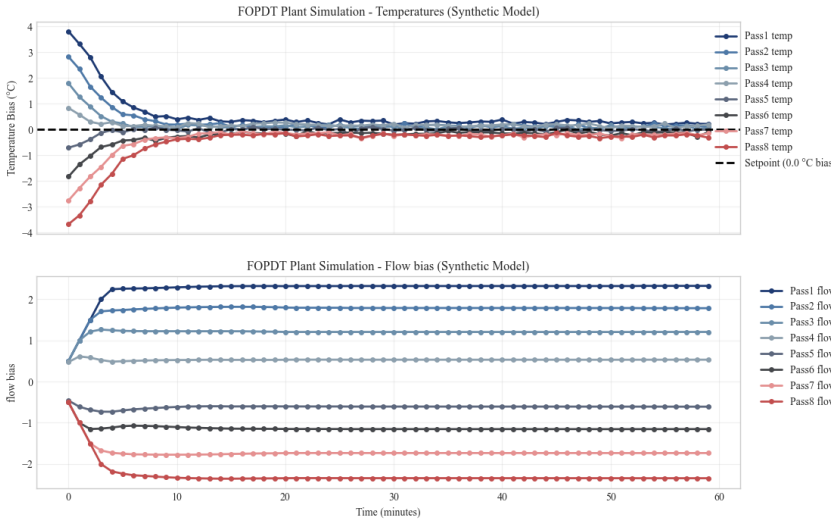


Figure 4.12: LSTM trained on synthetic data - MPC Closed Loop Simulation

Performing a step test on Pass 1 in simulation of the synthetic LSTM-MPC, the controller reduces the error slightly but settles with an offset, as can be seen in Figure 4.13.

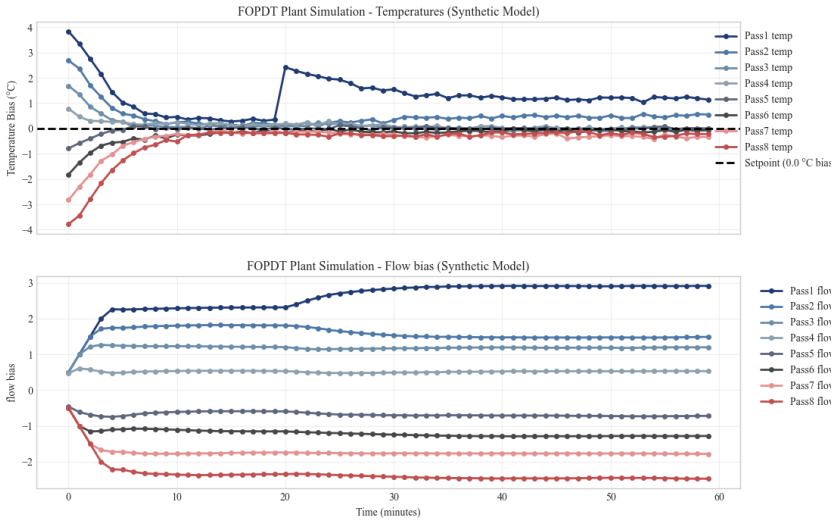


Figure 4.13: LSTM trained on synthetic data - step test

To remove the steady state offset, integral action with dead band is added as explained in Chapter 3. The results are shown in Figure 4.14, with the steady state bias offset being reduced. However, some overshoot and oscillations appear.

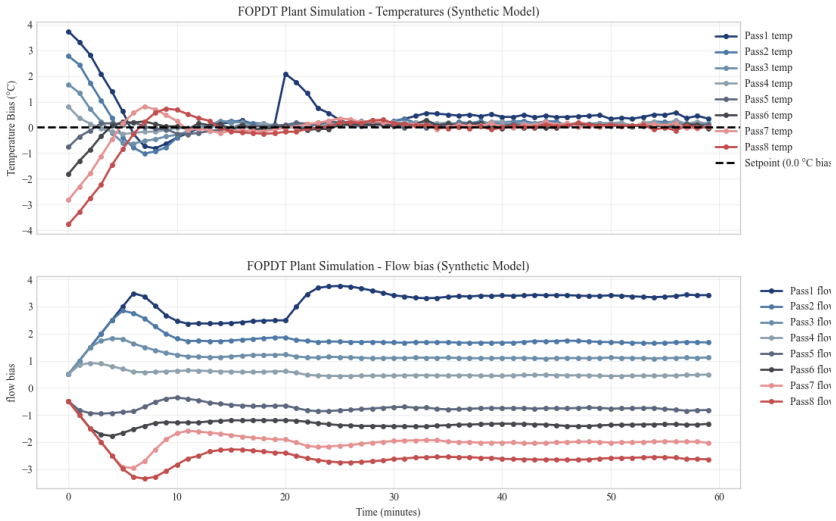


Figure 4.14: LSTM trained on synthetic data with integral action - step test

Regarding the computational time for the synthetic LSTM-MPC cycle, the results are stated in Table 4.4.

Table 4.4: Computational Time for 60 minutes simulation - Synthetic LSTM-MPC

Step Timing Metrics		
Mean step time	**Max step time**	**Min step time**
2.77 s	8.76 s	0.36 s
Optimizer Timing Metrics		
Mean optimizer time	**Max optimizer time**	**Total simulation time**
2.72 s	6.77 s	151.90 s

4.4.2 Base LSTM-MPC

The base LSTM was trained on the real historical data, formulated as deviations (change in flow causing change in temperature) as well as with auto-regressive training with multi step loss. State reconstruction was used as described in Chapter 3, to overcome the degrees of freedom issue. The parameters used for the LSTM training and the MPC are stated in Table 4.5.

Table 4.5: Base-LSTM: Model and MPC parameters

Model Parameters				
Nr of Layers	**Hidden Size**	**Epochs**	**Learning Rate**	**Drop Out**
1	16	15	0.001	0
MPC Parameters				
H_p	H_c	**Q**	**R**	**S**
30	15	9.0	0.001	10.0

As a first indicator on whether the LSTM model trained on real historical process data would predict the same behavior as the the FOPDT model, the LSTM was tested (using the synthetic data as testing data instead of the training data as test data) on the synthetically generated FOPDT data, with fitting results shown in Figure 4.15.

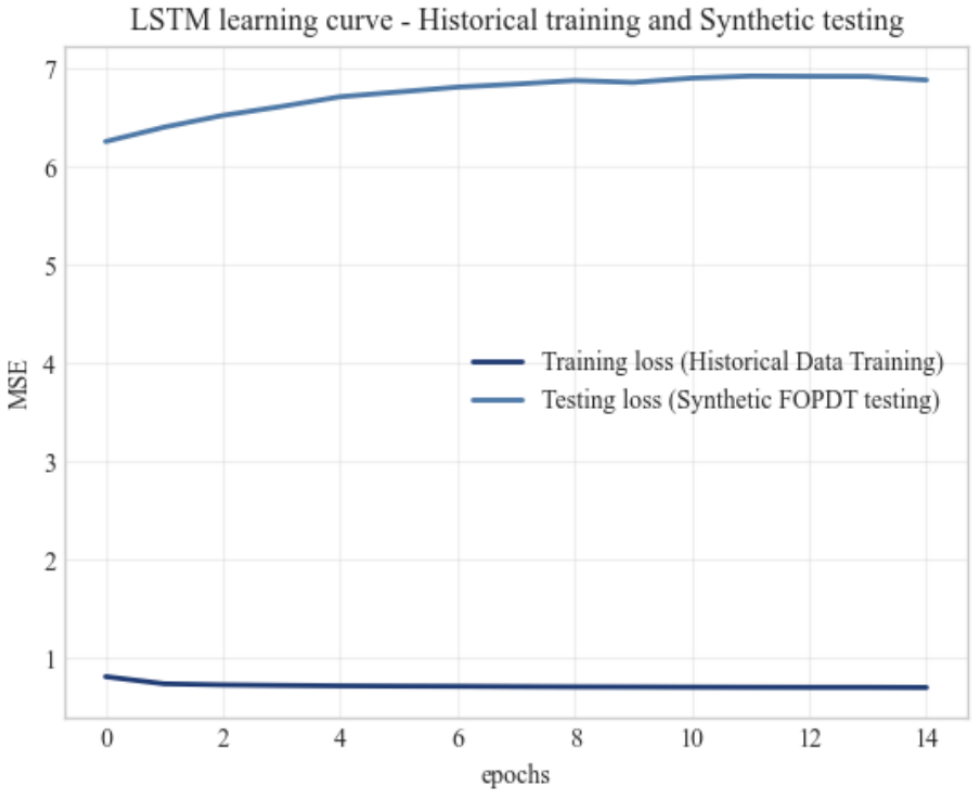


Figure 4.15: LSTM loss curve: Trained on Historical data and tested on Synthetic FOPDT data

Then the MSE loss curve was computed for the Base-LSTM training and testing on real historical data, shown in Figure 4.16. Note that the training loss is higher than the testing loss, suggesting overfitting.

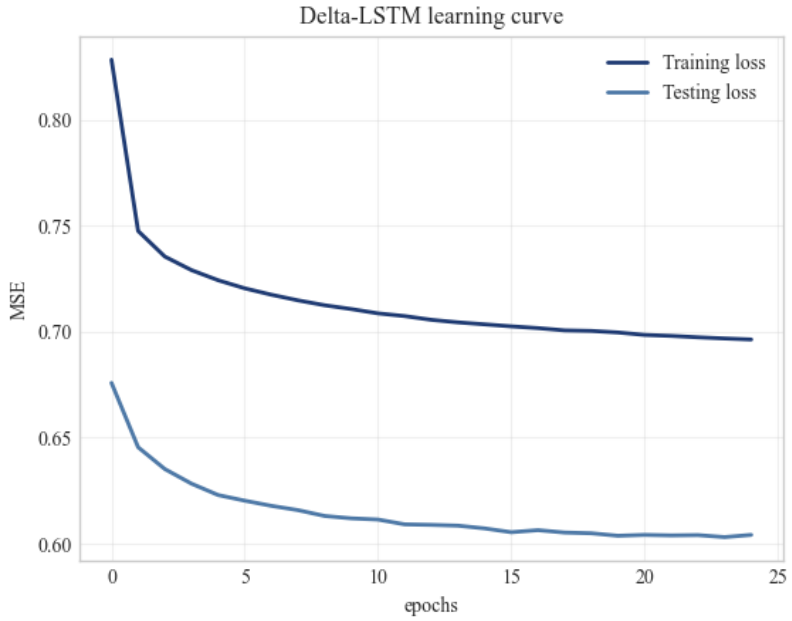


Figure 4.16: Base-LSTM: trained on historical process data - loss curve

Applying the Base LSTM model in an MPC setting (with integral action applied) to evaluate open loop predictions and closed loop simulation gave the following results in Figures 4.17 and 4.18.

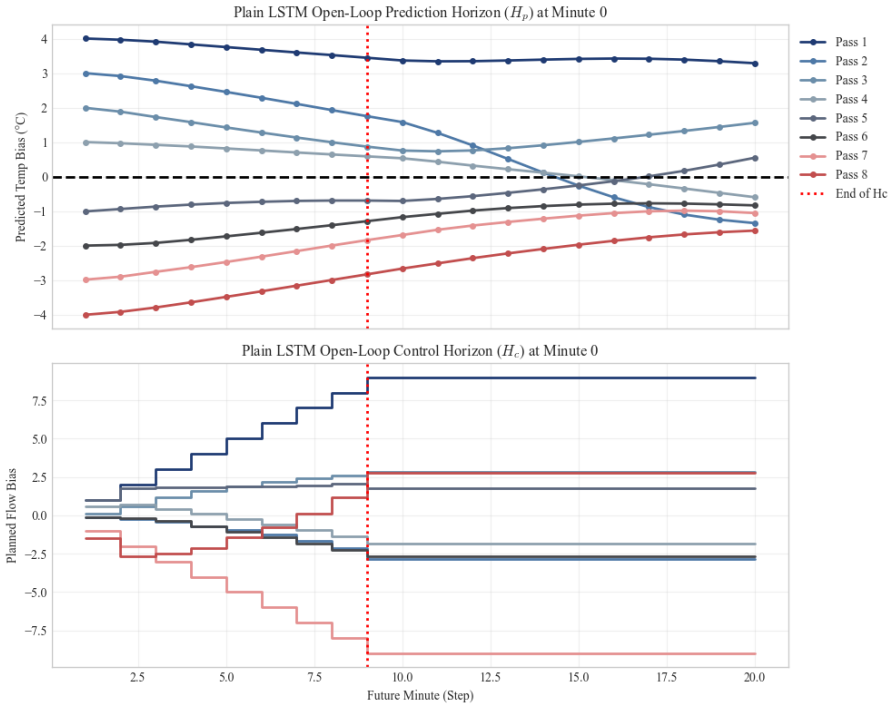


Figure 4.17: Base LSTM: MPC H_c and H_p with integral action

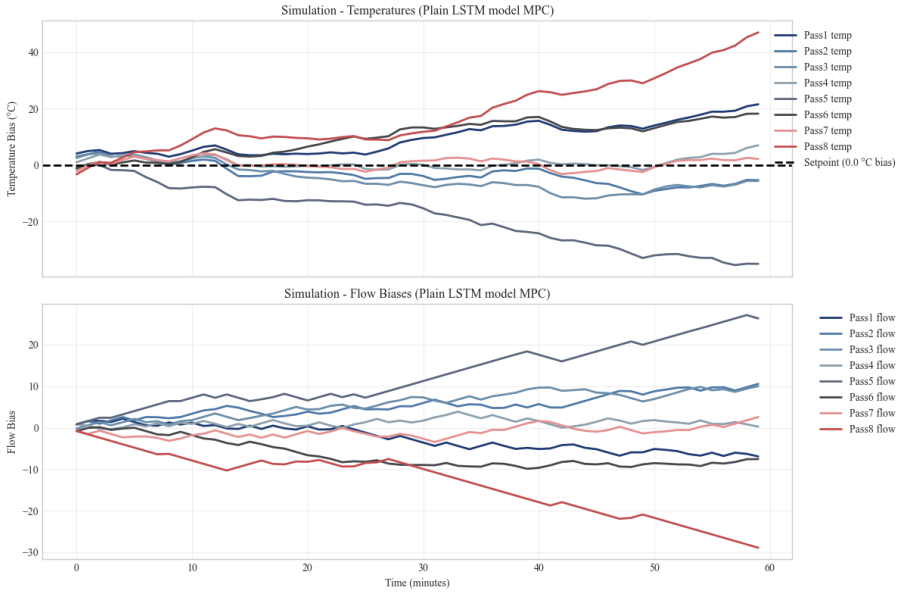


Figure 4.18: Base LSTM: MPC Closed Loop Simulation H_c and H_p with integral action

The MPC does not settle in the chosen prediction and control horizon, the results diverge from the setpoints. Because of this, further testing in regards to robustness and computational time was not evaluated.

4.4.3 Hybrid LSTM-MPC

A Hybrid LSTM-MPC was built, utilizing the FOPDT model to achieve reasonable physics for the system, while using the LSTM model from the Base LSTM to account to mismatch between the model and reality. To settle, it required a slightly longer prediction horizon and a slightly longer control horizon, as stated in Table 4.6.

Table 4.6: Hybrid-LSTM: Model and MPC parameters

Model Parameters				
Nr of Layers	**Hidden Size**	**Epochs**	**Learning Rate**	**Drop Out**
1	16	25	0.001	0
MPC Parameters				
H_p	H_c	**Q**	**R**	**S**
30	15	9.0	0.001	10.0

The loss curve for the Hybrid LSTM network is shown in Figure 4.19.

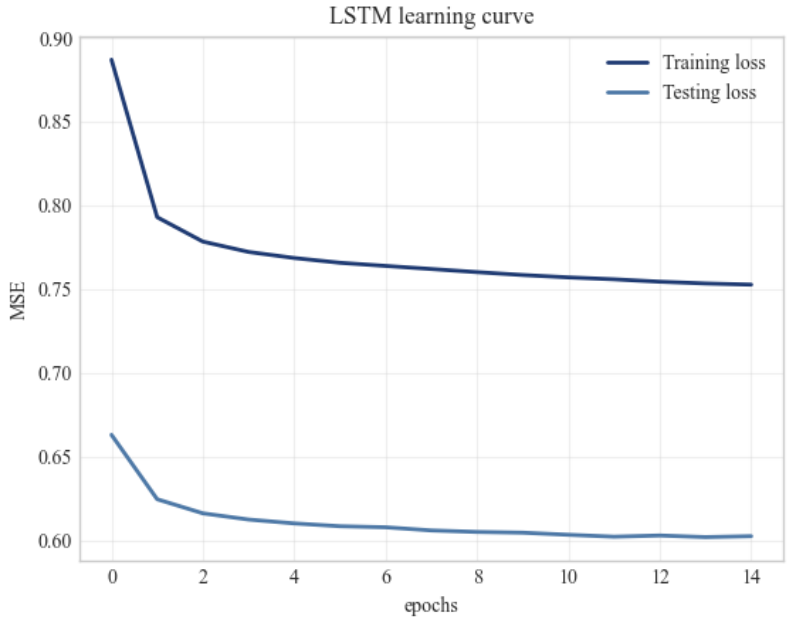


Figure 4.19: Hybrid-LSTM - loss curves

The open loop first step prediction as well as the closed loop simulation results when using integral action are displayed in Figures 4.20 and 4.21.

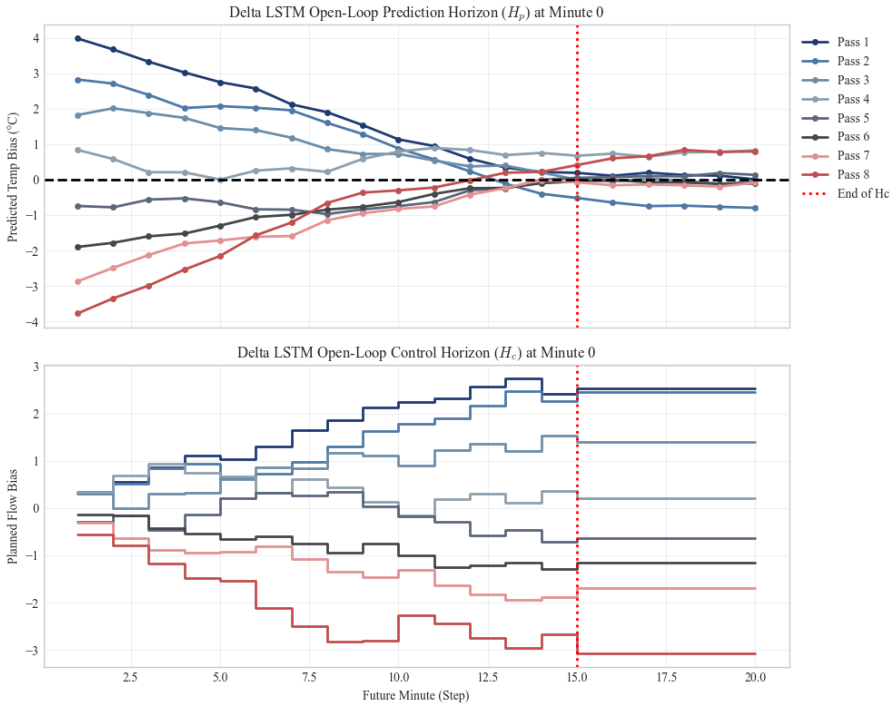


Figure 4.20: Hybrid-LSTM: MPC H_c and H_p with integral action

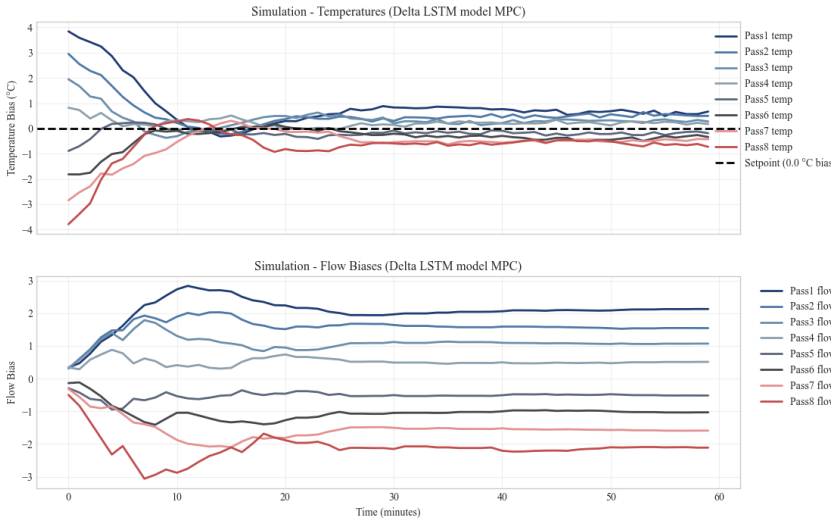


Figure 4.21: Hybrid-LSTM: MPC Closed Loop Simulation H_c and H_p with integral action

The step test of adding +2 degrees temperature bias to Pass 1 is shown in Figure 4.22.

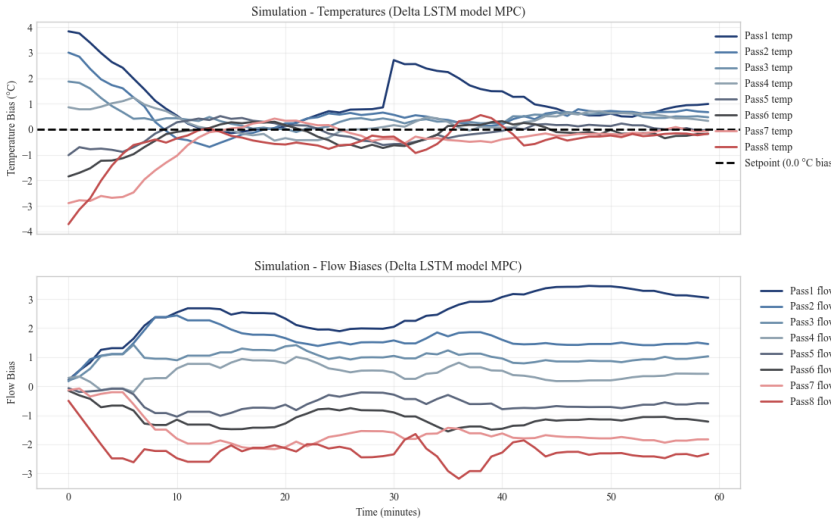


Figure 4.22: Hybrid-LSTM MPC with integral action: step test

The computational times for the Hybrid LSTM formulation are shown in Table 4.7.

Table 4.7: Computational Time for 60 minutes simulation - Hybrid LSTM-MPC

Step Timing Metrics		
Mean step time	**Max step time**	**Min step time**
14.84 s	43.20 s	0.91 s
Optimizer Timing Metrics		
Mean optimizer time	**Max optimizer time**	**Total Simulation Time**
14.79 s	43.19 s	890.07 s

5
Discussion

5.1 LSTM-MPC evaluation

5.1.1 Performance of LSTM-MPC

The LSTM trained on the real historical process data from H2201 predicted the data well, with a decreasing loss in both training and validation test. For this system, a simple single layer with 16 hidden cells performed well in predicting the data. A more complex LSTM led to overfitting.

As a first indicator on whether the LSTM model trained on real historical process data would predict the same behavior as the the FOPDT model, the LSTM was tested (validated) on the synthetically generated FOPDT data. The results in Figure 4.15 show that the loss is low for the model training meaning that the LSTM network learns the data well, however the testing loss is much higher. This tells us that the models are significantly deviating from each other, which is not unexpected since the historical process data used for learning is in closed loop with feedback and constraints applied. This explains why the LSTM seemingly learns the controller behavior of the closed loop data and struggles to reveal the physics behind it. The FOPDT is still used as plant in all MPC loops in simulation because it is considered the most true version available of the system dynamics.

The proposed Base LSTM-MPC controller confirms that the base LSTM network trained on historical process data does not learn the physics of the system properly. As can be seen in Figure 4.18, the MPC controller with the Base LSTM prediction model does not manage to control the temperature biases. Looking at the open loop predictions, it can be seen that the controller does not manage to plan for the predicted temperatures to reach zero, and in the closed loop simulation, this issue is not fixed by applying feedback. This confirms that the issue is not with the MPC formulation but with the prediction model. Compared to the baseline controller, it is clear that the gain for a control move to a temperature output for the Base LSTM-MPC is much smaller than in the baseline with the FOPDT model. Hence, larger control moves are proposed by the optimizer to try and minimize the temperature error. However, it still fails to bring the error down. Some of the pass predictions also go in the wrong direction, for example Pass 4 shows a decreasing temperature over the horizon even if the pass flow bias decreases, which should be physically impossible because the residence time in the furnace decreases. Increasing the prediction and control horizons did not help in simulation, it only lead to heavier computations.

The main issue is that the Base LSTM learns on steady state, closed loop historical process data with little excitation and therefore struggles to see past the correlation between passes that the active controller causes. This results in incorrect mapping of the input to output. In an attempt to overcome this, state reconstruction was evaluated to account for loss in degrees of freedom from the active controller constraints. And, auto-regressive multi-step loss training was evaluated to improve the model understanding of the long term physics of the system with added trajectory loss. However, this only improved predictions slightly. The LSTM training parameters were tuned to improve predictions. Increasing the number of layers from 1 to 2 had no significant impact on the LSTM prediction quality. The hidden size of 16 seemed to yield the most accurate model. Increasing the hidden size did not significantly improve predictions. This shows the importance of the data quality and data formulation in LSTM training, as well as the importance of an accurate prediction model in an MPC setting.

In order to prove the concept that an LSTM-MPC works in practice and confirm that the issue is an LSTM model trained on closed loop historical data, the synthetic LSTM-MPC was evaluated. Trained on the synthetically generated FOPDT data, it received data with no active control or constraints and could therefor learn the dynamics of the system. These results are in line with what can be seen in literature for an LSTM-MPC, such as in the study by Rajasekhar et al. regarding deep learning for nonlinear MPC model building [4]. The results of the synthetically trained LSTM-MPC were comparable to the baseline DMC controller in performance. The computational time was longer but withing CPU capacity bound (less than 20 seconds), the convergence to set-point was stable and smooth. However, during the disturbance step test in Figure 4.13, an integral action penalty needed to be added to the controller. With tuning of the penalties and the addition of integral action to avoid a bias offset, the controller performs well. The synthetic version of the LSTM-MPC proves that excitation rich data is the needed for successful LSTM identification, so that the network does not only learn on steady-state data. The results support the idea that successful application of deep learning in process control is dependent on the principles of system identification, including data quality, process excitation, and model validation [13].

The Hybrid LSTM-MPC is proposed as a possible solution to the closed loop data problem, with the base physics defined by the FOPDT model and the residuals predicted by the trained LSTM network. The performance of the resulting controller does converge to values close to the setpoint, without large oscillations or runaways. However, the remaining bias offset is larger than in the baseline case, but the baseline case is highly ideal. The training and testing MSE loss for the Hybrid model is larger than the synthetic LSTM (testing loss landed on about 0.02 for the synthetic LSTM and 0.6-0.7 for the LSTM trained on real data). This alternative is a safer option than the pure Base LSTM, as the FOPDT will provide safe physically accurate predictions. The LSTM could capture possible nonlinear dynamics and model change over time etc., but without making the furnace unstable if it gets the

predictions wrong, just less optimized. However, the computational time for the Hybrid LSTM is too long.

The Physics Informed LSTM for MPC (PINN-MPC) that was suggested in the methods did not improve the performance of the Base LSTM, and was hence discarded.

5.1.2 Comparison to the baseline controller

Comparing the baseline MPC with the proposed LSTM controllers, closed loop stability, robustness and computational time was evaluated. The baseline MPC shows a shorter computational time and slightly lower error. However, in regards to robustness and closed loop stability, the synthetic LSTM showed similar results.

However, adjustments needed to be made to the MPC formulation to improve the performance with the LSTM as prediction model in the control loop. Integral action, dead band, longer prediction and control horizon as well as more aggressive tuning was required. Both disturbance estimation and integral action was implemented, in comparison to the baseline model that did well using only disturbance estimation. This suggests that the LSTM nonlinearity introduces a form of model-plant uncertainty that differs from traditional linear errors. In contrast to first-principles or linear models, neural-network prediction models introduce approximation errors that accumulate over time. Consequently, horizon selection becomes more important, and longer prediction horizons may be needed to ensure that the optimizer captures the long-term effect of control moves despite model uncertainty [2].

Regarding computational time, the synthetic LSTM controller is comparable to the baseline DMC, however the Hybrid model takes much longer, and is in this formulation too slow to be implemented with a 20 second sampling time. It is slower since it has to evaluate both the FOPDT model as well as the LSTM model with the SLSQP solver.

The LSTM-MPC handled unmeasured disturbances well, in about the same time as the baseline controller. The LSTM-MPC was made robust to measurement noise through the proposed disturbance estimation using the Low-Pass filter, and made robust to model uncertainties through the forced integral action. Regarding the closed loop stability, the Base LSTM-MPC does not become stable at all but shows growing oscillations and no convergence to setpoint. The Synthetic and Hybrid LSTM MPC keep the system stable in closed loop.

Comparing with the raw data plots in Figure 4.1, there is also a bias range of about 4 degrees over the passes, meaning that the currently implemented DMC also has a bias offset. Comparing this temperature bias to the resulting temperature bias of the Hybrid LSTM, the Hybrid LSTM lands the system on a smaller range closer to zero, however, in an ideal simulation.

5.2 Limitations

There are a few limitations to the thesis, affecting the results, which should be taken into consideration in the interpretation.

The baseline comparison model is an ideal representation of the system, with manually added tweaks to make it more true to reality. However, the real system may have other factors affecting its physics, such as buildup over time, that makes the model less accurate and has to be handled with feedback. Hence, the comparison to the baseline model is not necessarily a fair comparison, which should be taken into account when crediting its results.

The main limitation to the LSTM-MPC implementation is the quality of the available training data. Even though there are three years of data available, which is a sufficient amount to train the LSTM, the furnace is, under normal operation, constantly being balanced by a controller. This leads to closed loop data that the AI-network struggles to see the physics in. In system identification, the quality of the prediction model is strongly dependent on the excitation level of the data, as the input needs to excite the system dynamics in order to ensure informative input to output data [13]. To overcome this, the data collection for model training and validation should be adjusted. If the pass balancing was operated in manual mode, open loop data would be collected. However, operating the plant in manual mode for longer periods of time requires a lot of operator attention and surveillance, in order to manually adjust the output for the eight flow control valves at every minute by observing the eight outlet temperatures. Therefore, methods for closed loop system identification should be researched further if this is to be attempted.

Going back to the raw data, in Figure 4.3 it can be seen that for the 1000 plotted data points, Pass 5 seems to have much larger oscillations and more aggressive movements compared to the others. As could be seen from the predictions of Pass 5 compared to the raw data plot, this seems to have a strong effect on the AI's ability to learn the physics of it. In Figure 4.17, Pass 5 seem to get an increase in temperature bias when there is an increase in flow bias, which should be physically impossible.

5.3 Suggestions for Practical Implementations

To apply the LSTM-MPC in practice in a live refinery control, there are some suggestions regarding implementation.

The computational time of the LSTM controller is higher than for the traditional DMC due to the repeated evaluation of the LSTM model within the optimization. Therefore, execution time should be evaluated before implementation to ensure that the optimization can be completed within the control interval.

The quality of the prediction model is dependent on the quality of the training data. The results of this thesis indicate that historical closed loop data may not provide sufficient patterns to identify the system dynamics. Additional data collection strategies may be required, such as further step-response tests or manually generated operating data. However, it needs enough excitation variation and preferably collected over time to capture shifts. From a system identification perspective, this creates a challenging learning environment, as feedback introduces dependence between inputs and measurement noise, and reduces the richness of the data available for dynamic modeling [12].

A challenge for AI-driven control is ensuring operability and safe operation. If something goes wrong in the predictions that cause a failure in the plant, it will be difficult to pinpoint where the issue roots from because of the black box structure of the LSTM. For this reason, a purely black box LSTM model may be difficult to justify in a industrial environment. The Hybrid LSTM approach investigated in this thesis has a potential solution by combining data driven predictions with a physically realistic process model. The linear FOPDT model reduces the risk of unrealistic predictions. Additional safety measures should also be implemented. In particular, hard constraints on predicted temperatures and control actions should be enforced to prevent physically impossible values from being processed by the MPC optimizer.

5.4 Future Work

The proposed LSTM-MPC framework demonstrated promising results in simulation, although some areas remain for future investigation.

The LSTM models investigated in this thesis were trained offline using a fixed dataset. Future work could explore online adaptation strategies, allowing the prediction model to continuously update as new process data is available. That approach could improve robustness to for example buildup in pipes, actuator wear and changing operating conditions etc. Automated data collection to take live data from tags to retrain the model continuously could be considered. Instead of manually exported datasets to excel, direct connections to process databases could enable automated data collection and model retraining.

Another future improvement would be to investigate tools for the LSTM hyperparameter tuning if the method is to be deployed for other process units. The LSTM network is configurable through the hyperparameters explained in Section 2.2.1. The chosen hyperparameters directly affects the models performance, and hence it is important to optimize them. Tuning by trial and error takes time, which is both experienced in this thesis as well as stated in the article by [24]. For example, a genetic algorithm could be used for LSTM hyperparameter tuning. [24] state that genetic algorithm has been investigated in several forecasting settings using LSTM networks.

The pass balancing problem considered in this thesis can quite confidently be approximated by linear dynamics within its normal operating range. Therefore, the full benefits of an LSTM-based predictor may not be evident in this case, since the LSTM could have a larger advantage compared to baseline FOPDT if the system dynamics are nonlinear. However, for proof of concept it is an intuitive system to work on. Future work could therefore evaluate the LSTM framework on systems with known nonlinear behavior. This could further assess whether LSTM prediction models can outperform traditional linear models in MPC, under conditions where linear models are less accurate. Furthermore, ways to improve computational time for an LSTM-MPC could be an interesting future work.

The Physics-Guided LSTM-MPC investigated in this thesis represents an approach for combining physical process knowledge with data driven learning. Future research could explore more advanced PINN or Hybrid formulations. In this thesis penalties were introduced for physically impossible predictions or unrealistic process behavior. To improve the safety of the predictions and making sure none of these wrongful predictions come through in the learnt model they could potentially be completely rejected.

6
Conclusion

An LSTM model can be successfully integrated into an MPC framework and used for closed-loop control of the pass-balancing problem, by using synthetically generated process data. The synthetic LSTM-MPC shows closed loop stability, robustness to process disturbances and is computationally viable in a real time control system, proved by simulation. Its performance is similar to that of the currently implemented traditional DMC controller.

The performance of an LSTM-MPC trained on real process data, however, is highly dependent on the quality and excitation level of the training data. Historical closed-loop process data presented challenges for model identification, indicating that data availability may be an obstacle for industrial deployment an that open loop data with the furnace controlled on manual mode should be considered.

The investigated hybrid Physics-Guided LSTM-MPC architecture showed potential for combining physical process knowledge with data-driven prediction models. This can be a relevant topic for further research, because it may provide model improvements while ensuring safe operations.

Bibliography

[1] Z. Wu, A. Tran, D. Rincon, and P. D. Christofides, "Machine learning-based predictive control of nonlinear processes. part i: Theory," *AIChE Journal*, vol. 65, no. 11, 2019.

[2] Y. M. Ren, M. S. Alhajeri, J. Luo, S. Chen, F. Abdullah, Z. Wu, and P. D. Christofides, "A tutorial review of neural network modeling approaches for model predictive control," *Computers & Chemical Engineering*, vol. 165, 2022.

[3] K. Seel, E. I. Grøtli, S. Moe, J. T. Gravdahl, and K. Y. Pettersen, "Neural network-based model predictive control with input-to-state stability," in *2021 American Control Conference (ACC)*, 2021.

[4] N. Rajasekhar, K. K. Nagappan, T. K. Radhakrishnan, and N. Samsudeen, "Effective mpc strategies using deep learning methods for control of nonlinear system," *International Journal of Dynamics and Control*, 2024.

[5] K. Huang, K. Wei, F. Li, C. Yang, and W. Gui, "Lstm-mpc: A deep learning based predictive control method for multimode process control," in *IEEE*, 2022.

[6] X. Kong, Z. Chen, W. Liu, K. Ning, L. Zhang, S. M. Marier, Y. Liu, Y. Chen, and F. Xia, "Deep learning for time series forecasting: A survey," *arXiv preprint arXiv:2503.10198*, 2025. [Online]. Available: https://arxiv.org/abs/2503.10198

[7] A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," *Physica D: Nonlinear Phenomena*, vol. 404, 2020.

[8] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control Engineering Practice*, vol. 11, 2003.

[9] B. Bernhardsson and K. J. Åström, "Model predictive control (mpc)," 2016, lecture Notes, Department of Automatic Control, Lund University (LTH). [Online]. Available: https://www.control.lth.se/fileadmin/control/Education/DoctorateProgram/ControlSystemsSynthesis/2016/MPC.pdf

[10] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, 2000.

[11] T. Glad and L. Ljung, *Control Theory: Multivariable and Nonlinear Methods*, 1st ed. Taylor & Francis, 2000.

[12] T. Söderström and P. Stoica, *System Identification*. Prentice-Hall, 1989.

[13] L. Ljung, C. Andersson, and K. Tiels, "Deep learning and system identification," in *IFAC-PapersOnLine*, vol. 53, 2020.

[14] P. T. Boggs and J. W. Tolle, "Sequential quadratic programming," *Acta Numerica*, 1995.

[15] D. Boiroux and J. B. Jørgensen, "Sequential l1 quadratic programming for nonlinear model predictive control," in *IFAC-PapersOnLine*, vol. 52, 2019.

[16] PyTorch, "torch.nn.lstm pytorch documentation." [Online]. Available: https://docs.pytorch.org/docs/main/generated/torch.nn.LSTM.html

[17] PyTorch, "torch.optim.adam - pytorch documentation." [Online]. Available: https://docs.pytorch.org/docs/main/generated/torch.optim.adam.Adam_class.html

[18] GeeksforGeeks, "Epoch in machine learning," n.d. [Online]. Available: https://www.geeksforgeeks.org/machine-learning/epoch-in-machine-learning/

[19] SciPy, "minimize(method='slsqp') scipy," https://docs.scipy.org/doc/scipy/reference/optimize.minimize-slsqp.html, 2024.

[20] "Scikit-learn: Machine learning in python." [Online]. Available: https://scikit-learn.org/

[21] B. Øksendal, *Stochastic Differential Equations: An Introduction with Applications*. Springer, 1998.

[22] Y. Zheng and Z. Wu, "Physics-informed online machine learning and predictive control of nonlinear processes with parameter uncertainty," *Industrial & Engineering Chemistry Research*, vol. 62, 2023.

[23] F. E. Bock, S. Keller, N. Huber, and B. Klusemann, "Hybrid modelling by machine learning corrections of analytical model predictions towards high-fidelity simulation solutions," *Materials*, vol. 14, no. 8, 2021.

[24] H. Dhake, Y. Kashyap, and P. Kosmopoulos, "Algorithms for hyperparameter tuning of lstms for time series forecasting," *Remote Sensing*, 2023.

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY