Diplomarbeit

Model Predictive Control for Power Control of a Fluidized Bed Furnace

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Diplom-Ingenieurs
unter der Leitung von

Ao.Univ.Prof. Dr Martin Kozek und
Univ.Prof. Dr. Stefan Jakubek
Institut für Mechanik und Mechatronik
Abteilung für Regelungstechnik und Prozessautomatisierung

eingereicht an der Technischen Universität Wien

Fakultät für Maschinenwesen und Betriebswissenschaften

von

Lukas Stanger
[redacted]

Wien, am 15. Juni 2021

Lukas Stanger

Eidesstattliche Erklärung

Ich erkläre eidesstattlich, dass ich die Arbeit selbständig angefertigt, keine anderen als die angegebenen Hilfsmittel benutzt und alle aus ungedruckten Quellen, gedruckter Literatur oder aus dem Internet im Wortlaut oder im wesentlichen Inhalt übernommenen Formulierungen und Konzepte gemäß den Richtlinien wissenschaftlicher Arbeiten zitiert, durch Fußnoten gekennzeichnet bzw. mit genauer Quellenangabe kenntlich gemacht habe.

Wien, am 15. Juni 2021

Lukas Stanger

Danksagung

Ganz besonders bedanken möchte ich mich bei Herrn Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Martin Kozek und Herrn Dipl.-Ing. Alexander Lukas Gratzer für die Betreuung meiner Diplomarbeit. Die vielen Besprechungen, in denen wir Probleme und Lösungsansätze diskutierten und ich Feedback von euch erhalten habe, haben nicht nur maßgeblich zum Erfolg meiner Arbeit beigetragen, sie haben mir auch viel Freude bereitet und die Motivation für meine Arbeit aufrecht erhalten. Herzlichen Dank auch an Herrn Univ.Prof. Dipl.-Ing. Dr.techn. Stefan Jakubek für die vielen hilfreichen Inputs zur Arbeit.

Außerdem möchte ich mich bei der Wien Energie für die Zusammenarbeit und die Bereitstellung der Messdaten bedanken.

Vielen lieben Dank auch an meine Mitstudenten, die durch gemeinsames Lernen, Mittagessen und Kaffeepausen ganz besonders für Spaß und Motivation im Studium gesorgt haben. Besonderen Dank an dich, lieber Michael Lehner, für die unzähligen Unitage, die wir gemeinsam verbracht haben.

Abschließend möchte ich mich noch ganz herzlich bei meiner Familie für die Unterstützung in der Zeit meines Studiums bedanken. Außerdem vielen Dank an dich, liebe Sophia, für's Korrekturlesen meiner Arbeit.

Kurzfassung

Wirbelschichtöfen werden unter anderem in Müllverbrennungsanlagen eingesetzt, da sie eine effiziente und gleichmäßige Verbrennung verschiedenster Müllzusammensetzungen ermöglichen. Diese Arbeit befasst sich mit der Entwicklung einer modellprädiktiven Regelung (engl. *Model Predictive Control*, MPC) für einen Wirbelschichtofen. Zur Identifikation des dafür benötigten Modells wird eine physikalische Modellierung mit experimentellen Identifikationsmethoden kombiniert: Energie- und Massenbilanzen liefern eine initiale Modellstruktur. Diese wird um den Betriebspunkt linearisiert und diskretisiert, die erhaltenen Gleichungen werden als Zustandsraummodell formuliert. Anschließend wird ein evolutionärer Algorithmus vorgestellt, der die auf physikalischen Gleichungen basierende Modellstruktur optimiert: Zusätzliche Kopplungen zwischen Eingängen, Zuständen und Ausgängen werden freigegeben, sofern dies physikalisch plausibel erscheint. Die im Modell enthaltenen Parameter werden in weiterer Folge mittels Messdaten geschätzt. Zur Validierung der Leistungsfähigkeit der linearen Zustandsraummodelle wird ein künstliches neuronales Netz trainiert, mit dem diese dann verglichen werden. Ein MPC wird entworfen, der ein lineares Zustandsraummodell zur Prädiktion verwendet. Das Modell wird um zusätzliche Störgrößenzustände erweitert, was eine Regelung ohne stationären Regelfehler erlaubt. Ein Kalman-Filter wird zur Zustandsschätzung der originalen und der erweiterten Zustände verwendet. Der MPC wird in zwei verschiedenen Zeitskalen betrieben, da sich die Zeitkonstanten des zu regelnden Prozesses stark unterscheiden. Simulationen werden durchgeführt, um die Funktionsfähigkeit des Reglers zu validieren.

Abstract

Fluidized bed furnaces are used in waste incineration plants for efficient and uniformly combustion of different waste compositions. The aim of this work is the development of an model predictive control (MPC) scheme for a fluidized bed furnace. In order to obtain a dynamic model suitable for MPC, first-principle equations are combined with experimental identification methods: Physical equations in terms of energy and mass balances give an initial structure for the model. These equations are linearized with respect to the operating point and discritized. An evolutionary-based algorithm is presented that optimizes the resulting model structure. This algorithm extends the model with additional coupling between inputs, states and outputs, whereby only physically plausible couplings are allowed to be added by the algorithm. The parameters in the model are estimated using measurement data. To validate the performance of the linear state-space models, an artificial neural network is developed as a benchmark model. A MPC is designed that uses the linear state-space model. The model is extended with additional disturbance states, which allows offset-free tracking of stationary set points. A Kalman filter is designed as an observer to estimate the system's original states and the additional disturbance states. The MPC operates in two different time scales in order to handle strongly different time constants in the process under consideration. Simulations are carried out to validate the effectiveness of the controller.

Contents

Chapter 1

Introduction

Waste incineration is a common treatment for the disposal of high volumes of waste in modern countries. It offers the possibility both for efficient disposal of waste and the usage of energy from combustion for conversion to electrical energy and for district-heating. In the European Union, 28 % of the municipal solid waste (MSW) is incinerated in 492 waste-to-energy plants. However, there are big differences between European countries regarding waste handling. In Austria, 39 % of the MSW is treated in waste-to-energy plants [1][2].

The most common waste incineration procedures are grate firing and fluidized bed combustion (FBC). Although grate firing is more common, fluidized bed combustion offers several advantages: Fluidized bed furnaces are better in handling the combustion of varying waste compositions. Moreover, the air supply in grate firing is more difficult as the air supply needs to fit the fuel conditions on the grate. The main differences between grate firing and fluidized bed combustion are discussed in [3].

The control of waste incineration plants is quite challenging due to coupling between the controlled variables, non-linearities, largely differing time constants and model parameters that are hard to be identified. Different control strategies for waste incineration plants have been applied in [7][8][9].

Model predictive control enables efficient control of multiple-input multiple-output systems. Moreover, constraints on control variables, output variables and inner states of the system can be taken into account explicitly. The aim of this work is to develop a model predictive controller that ensures process control at the desired operating point while minimizing fluctuations regarding temperatures and oxygen concentrations in the flue gas.

The work is structured into four parts: The first chapter gives a physical description of fluidized bed furnace, including mathematical equations modelling the process. In the second chapter, a linear model structure is derived from the physical equations and parameters are estimated using measurement data. In the third chapter, an algorithm is presented to optimize the model structure in order to get a better mathematical description of the fluidized bed furnace. The forth chapter deals with the model predictive control architecture that utilizes the identified models to control the process under consideration.

Research Questions

In this thesis, the following research questions are covered:

- Which model gives the best mathematical description of the fluidized bed furnace and can then be used to implement a linear model predictive control scheme?
- What is an effective model predictive control algorithm that can handle the process dynamics of the fluidized bed furnace?

Chapter 2

Fluidized Bed Furnace

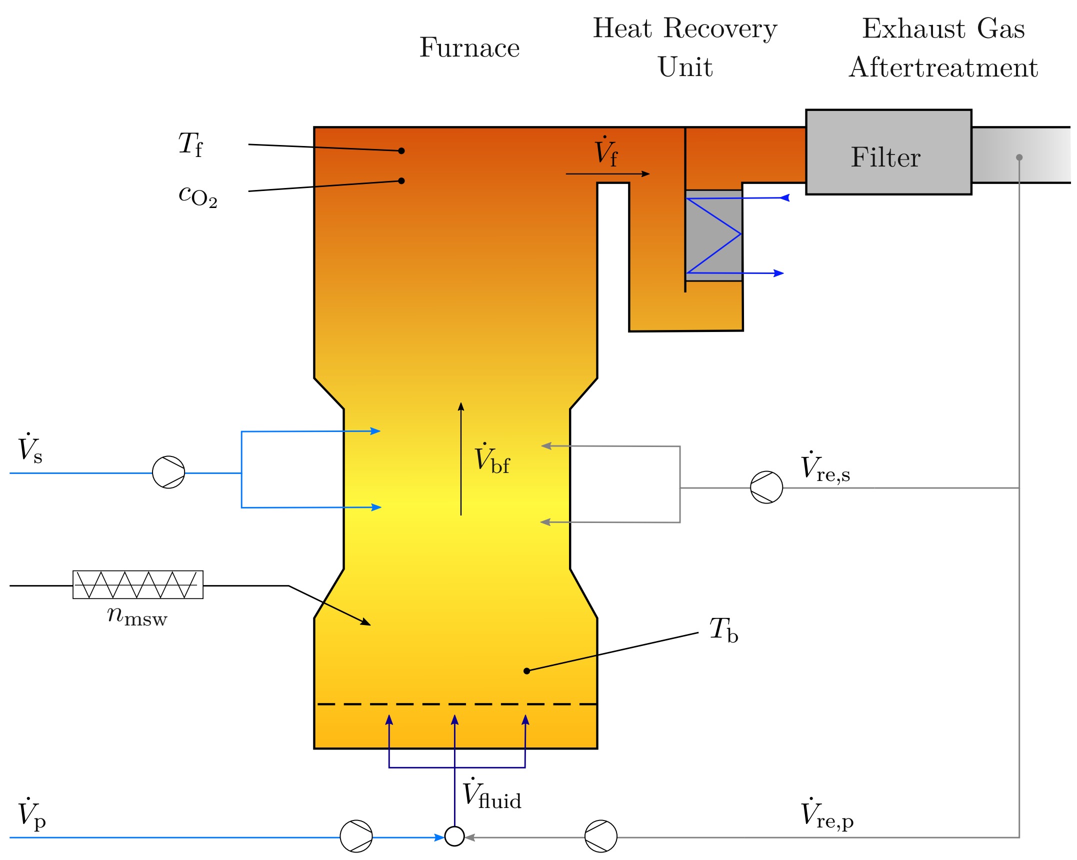
This chapter is about the process of a fluidized bed furnace. After a brief introduction, physical equations are developed in order to get an analytical description of the furnace.

2.1 Introduction

A fluidized bed furnace consists of a bed and a freeboard. The bed contains a high amount of inert material, typically sand. This material is *fluidized* by upstreaming air. *Fluidization* means that the particles in the bed are kept in suspension by an upstreaming gas and the bed therefore gets properties similar to a fluid: objects with a higher density than the bed material sink, whereas those with a lower density float. The fluidized bed furnace allows a good heat transfer and a uniformly mixing of the bed material, the fluidization gas and the fuel. Depending on the velocity of the fluidization gas, it is distinguished between different types of fluidized bed combustors. At waste incineration plants, typically three different types are used: the bubbling fluidized bed (BFBC), the rotating fluidized bed (RFBC) and the circulating fluidized bed [4]. In bubbling fluidized beds, the velocity of the fludization gas is rather low, leading to a well-defined surface of the bed. The same applies to the rotating fluidized bed, however, the fluidization gas is not even distributed, what leads to an internal rotation of the bed. In circulating fluidized beds, the velocity of the fluidization gas is higher. Particles of the bed are carried upwards to the freeboard and need to be separated using a cyclone, so that they can be fed back to the bed. In the fluidized bed furnace under consideration, a BFBC is used and a mix of air and recirulated gas is taken as a fluidization gas. More detailed insights on fluidized beds can be found in [4][5][6].

2.2 Physical Modelling

The main parts of the waste incineration plant are the furnace, the heat recovery boiler and the exhaust after-treatment systems like filters and catalysts. The furnace itself consists of



n_{msw}	waste feed screw conveyor rotational speed
$\dot{V}_p$	primary air volume flow
$\dot{V}_{re,p}$	primary recirculated gas volume flow
$\dot{V}_s$	secondary air volume flow
$\dot{V}_{re,s}$	secondary recirculated gas volume flow
$\dot{V}_{fluid}$	gas for fluidization volume flow
$\dot{V}_{bf}$	Volume flow from bed to freeboard
$\dot{V}_f$	flue gas volume flow
T_b	fluidized bed temperature
T_f	furnace head temperature
c_{O_2}	furnace head oxygen concentration

Figure 2.1: Fluidized bed furnace.

the fluidized bed at the bottom and the freeboard at the top. Figure 2.1 gives a schematic drawing of the plant.

There can be different types of fuels fed into the furnace. The most important task of the plant is to incinerate municipal solid waste. However, also sewage sludge can be burned. Moreover, auxiliary fuels, like oil, can be fed into the furnace if necessary. Air and recirculated gas are blown into the furnace at different heights. Fans and valves are used to control the gas flows.

Primary air and a proportion of the recirculated gas are blown into the fluidizied bed from below. This gas flow is called the gas flow for fluidization. It always needs to be constant at a certain level in order to achieve the fluidization. Within the fluidized bed the majority of the fuel is gasified, the rest remains as ash. The part remaining as ash is neglected later when writing energy- and mass balances for the furnace. Some of the gasified fuel is already burnt within the fluidized bed. However, there is not enough oxygen available in the bed in order to burn the whole amount of gasified fuel in here. The proportion that is not burned in the bed is burned at the freeboard later, where additional air (secondary air) and recirculated gas are blown into the furnace. At the freeboard, there is more oxygen inside the furnace than needed for combustion. The oxygen left at the top of the furnace is measured and desired to be at a certain level, which is necessary for the exhaust aftertreatment.

To get a mathematical description of the fluidized bed furnace, physical equations are used to describe the process. Energy balances for the fluidized bed and the whole furnace are used to model the temperature in the bed and at the top of the furnace, respectively. In order to get a description for the oxygen concentration, for flue gas mass flow and for fluidization gas mass flow, mass balances are used.

Although different kinds of fuel can be burned in the furnace, like waste, sewage sludge, coke or oil, the following equations are modelling waste incineration only.

2.2.1 Bed Temperature

The temperature of the fluidized bed can be modelled as

$$C_{\mathrm{b}} \frac{\mathrm{d}T_{\mathrm{b}}}{\mathrm{d}t} = \dot{H}_{\mathrm{p}} + \dot{H}_{\mathrm{re,p}} + \dot{H}_{\mathrm{msw}} - \dot{H}_{\mathrm{bf}} + P_{\mathrm{cb}}\,, \tag{2.1}$$

where C_{b} indicates the heat capacity of the fluidized bed. The enthalpy flow of the primary air $\dot{H}_{\mathrm{p}}$, the enthalpy flow of the primary recirculated gas $\dot{H}_{\mathrm{re,p}}$, the enthalpy flow of the waste $\dot{H}_{\mathrm{msw}}$, the enthalpy flow of the gas leaving the bed $\dot{H}_{\mathrm{bf}}$ and the power released by combustion

P_{cb} are

$$\dot{H}_{\mathrm{p}} = c_{\mathrm{p,air}} \rho_{\mathrm{air}} \dot{V}_{\mathrm{p}} T_{\mathrm{p}}, \tag{2.2}$$
$$\dot{H}_{\mathrm{re,p}} = c_{\mathrm{p,re}} \rho_{\mathrm{re}} \dot{V}_{\mathrm{re,p}} T_{\mathrm{re}}, \tag{2.3}$$
$$\dot{H}_{\mathrm{msw}} = c_{\mathrm{p,msw}} \dot{m}_{\mathrm{msw}} T_{\mathrm{msw}}, \tag{2.4}$$
$$\dot{H}_{\mathrm{bf}} = c_{\mathrm{p,bf}} \rho_{\mathrm{bf}} \dot{V}_{\mathrm{bf}} T_{\mathrm{b}}, \tag{2.5}$$
$$P_{\mathrm{cb}} = x_{\mathrm{c}} H_{\mathrm{c}} \dot{m}_{\mathrm{msw}}, \tag{2.6}$$

with the specific heat capacities c_{p}, the temperatures T, the densities ρ, the proportion of waste burnt in the bed x_{c} and the net calorific value H_{c}. The volume flow rates $\dot{V}$ are shown in Figure 2.1 and are always given as standard volume flow rates at 1.013 25 bar and 0 °C according to DIN 1343. Therefore, the densities ρ refer to the density at standard temperatures and pressures.

The mass flow of waste into the bed $\dot{m}_{\mathrm{msw}}$ can be written as a function of the waste feed screw conveyor rotational speed n_{msw}

$$\dot{m}_{\mathrm{msw}} = k_{\mathrm{msw}} n_{\mathrm{msw}}, \tag{2.7}$$

where k_{msw} is just a constant describing the relation between $\dot{m}_{\mathrm{msw}}$ and n_{msw}. If the mass of the fluidized bed is assumed to be constant, the mass flow leaving the bed is

$$\rho_{\mathrm{bf}} \dot{V}_{\mathrm{bf}} = \dot{m}_{\mathrm{msw}} + \rho_{\mathrm{air}} \dot{V}_{\mathrm{p}} + \rho_{\mathrm{re}} \dot{V}_{\mathrm{re,p}}. \tag{2.8}$$

The proportion of the waste burned in the fluidized bed can be stated as

$$x_{\mathrm{c}} = \frac{\rho_{\mathrm{air}} \dot{V}_{\mathrm{p}} c_{\mathrm{O_2,m,air}} + \rho_{\mathrm{re}} \dot{V}_{\mathrm{re,p}} c_{\mathrm{O_2,m,re}}}{k_{\mathrm{f}} \dot{m}_{\mathrm{msw}}}, \tag{2.9}$$

where k_{f} denotes the specific oxygen demand of the fuel and $c_{\mathrm{O_2,m}}$ are the oxygen concentrations as a mass fraction. As mentioned above, a substoichiometric combustion is assumed for the fluidized bed. Note that it is assumed that there is never enough oxygen available so that the whole fuel can be combusted in the fluidized bed. Another assumption is that the temperature in the bed is always high enough in order that all the oxygen in the bed can be used for combustion.

With (2.9) the power released by combustion in (2.6) can be written as

$$P_{\mathrm{cb}} = \frac{H_{\mathrm{c}}}{k_{\mathrm{f}}} (\rho_{\mathrm{air}} \dot{V}_{\mathrm{p}} c_{\mathrm{O_2,m,air}} + \rho_{\mathrm{re}} \dot{V}_{\mathrm{re,p}} c_{\mathrm{O_2,m,re}}). \tag{2.10}$$

2.2.2 Furnace Head Temperature

For head of the furnace, an energy balance equation can be written as

$$C_\mathrm{f}\frac{\mathrm{d}T_\mathrm{f}}{\mathrm{d}t} = \dot{H}_\mathrm{bf} + \dot{H}_\mathrm{s} + \dot{H}_\mathrm{re,s} - \dot{H}_\mathrm{f} + P_\mathrm{cf}, \tag{2.11}$$

with the heat capacity C_f. The enthalpy flow from the fluidized bed to the head of the furnace $\dot{H}_\mathrm{bf}$, the enthalpy flow of the secondary air $\dot{H}_\mathrm{s}$, the enthalpy flow of the secondary recirculated gas $\dot{H}_\mathrm{re,s}$, the enthalpy flow of the flue gas $\dot{H}_\mathrm{f}$ and the power released by combustion in the furnace head P_cf can be specified as

$$\begin{aligned}
\dot{H}_\mathrm{bf} &= c_\mathrm{p,bf}\rho_\mathrm{bf}\dot{V}_\mathrm{bf}T_\mathrm{b}, && (2.12)\\
\dot{H}_\mathrm{s} &= c_\mathrm{p,air}\rho_\mathrm{air}\dot{V}_\mathrm{s}T_\mathrm{s}, && (2.13)\\
\dot{H}_\mathrm{re,s} &= c_\mathrm{p,re}\rho_\mathrm{re}\dot{V}_\mathrm{re,s}T_\mathrm{re}, && (2.14)\\
\dot{H}_\mathrm{f} &= c_\mathrm{p,f}\rho_\mathrm{f}\dot{V}_\mathrm{f}T_\mathrm{f}, && (2.15)\\
P_\mathrm{cf} &= (1-x_\mathrm{c})H_\mathrm{c}\dot{m}_\mathrm{msw}, && (2.16)
\end{aligned}$$

with the heat capacities c_p and temperatures T. Again, the volume flow rates are shown in Figure 2.1 and are given as standard volume flow rates. Inserting x_c from (2.9) the power P_cf can be written as

$$P_\mathrm{cf} = H_\mathrm{c}\dot{m}_\mathrm{msw} - \frac{H_\mathrm{c}}{k_\mathrm{f}}(\rho_\mathrm{air}\dot{V}_\mathrm{p}c_{\mathrm{O_2,m,air}} + \rho_\mathrm{re}\dot{V}_\mathrm{re,p}c_{\mathrm{O_2,m,re}}). \tag{2.17}$$

2.2.3 Oxygen Concentration in the Furnace Head

In order to model the oxygen concentration at furnace head, a mass balance can be written for the oxygen:

$$M_\mathrm{f}\frac{\mathrm{d}c_{\mathrm{O_2,m}}}{\mathrm{d}t} = \rho_\mathrm{air}\dot{V}_\mathrm{s}c_{\mathrm{O_2,m,air}} + \rho_\mathrm{re}\dot{V}_\mathrm{re,s}c_{\mathrm{O_2,m,re}} - \rho_\mathrm{f}\dot{V}_\mathrm{f}c_{\mathrm{O_2,m}} - \dot{m}_{\mathrm{O_2,m,comb}}, \tag{2.18}$$

with the gas mass in the freeboard M_f, the oxygen concentration of air $c_{\mathrm{O_2,m,air}}$, the oxygen concentration in the recirculated gas $c_{\mathrm{O_2,m,re}}$ and the oxygen concentration $c_{\mathrm{O_2,m}}$ in furnace head that is assumed to be the oxygen concentration in the flue gas as well. Note that these oxygen concentrations are mass fractions. The amount of oxygen needed for combustion $\dot{m}_{\mathrm{O_2,comb}}$ is specified as

$$\dot{m}_{\mathrm{O_2,comb}} = k_\mathrm{f}\dot{m}_\mathrm{msw}(1-x_\mathrm{c}). \tag{2.19}$$

Inserting x_c from 2.9 leads to

$$\dot{m}_{\mathrm{O_2,comb}} = k_\mathrm{f}\dot{m}_\mathrm{msw} - \rho_\mathrm{air}\dot{V}_\mathrm{p}c_{\mathrm{O_2,m,air}} - \rho_\mathrm{re}\dot{V}_\mathrm{re,p}c_{\mathrm{O_2,m,re}}. \tag{2.20}$$

On the plant, the oxygen concentrations are measured as percentage of volume. Therefore, the relationship between mass fraction and volume fraction is needed:

$$c_{\mathrm{O_2}} = \frac{V_{\mathrm{O_2}}}{V_{\mathrm{f}}}, \tag{2.21}$$

$$c_{\mathrm{O_2,m}} = \frac{m_{\mathrm{O_2}}}{m_{\mathrm{f}}} = \frac{V_{\mathrm{O_2}}\rho_{\mathrm{O_2}}}{V_{\mathrm{f}}\rho_{\mathrm{f}}}. \tag{2.22}$$

The index m denotes a mass fraction. As most of the time the volume fraction is of interest, no index is used for the volume fraction. Now, the density ρ is described using the ideal gas law

$$\rho = \frac{p}{RT}. \tag{2.23}$$

This leads to

$$c_{\mathrm{O_2,m}} = \frac{V_{\mathrm{O_2}}}{V_{\mathrm{f}}} \frac{\frac{p_{\mathrm{O_2}}}{R_{\mathrm{O_2}}T_{\mathrm{O_2}}}}{\frac{p_{\mathrm{f}}}{R_{\mathrm{f}}T_{\mathrm{f}}}}. \tag{2.24}$$

Temperature and pressure must be the same for oxygen and total gas. Hence, the oxygen mass fraction can be described using the oxygen volume fraction and the gas constants R:

$$c_{\mathrm{O_2,m}} = \frac{V_{\mathrm{O_2}}}{V_{\mathrm{f}}} \frac{R_{\mathrm{f}}}{R_{\mathrm{O_2}}} = c_{\mathrm{O_2}} \frac{R_{\mathrm{f}}}{R_{\mathrm{O_2}}}. \tag{2.25}$$

2.2.4 Flue Gas Mass Flow

The flue gas mass flow leaving the furnace can be modeled as the sum of the mass flows entering the furnace, which leads to the mass balance

$$\rho_{\mathrm{f}}\dot{V}_{\mathrm{f}} = \dot{m}_{\mathrm{msw}} + \rho_{\mathrm{air}}\dot{V}_{\mathrm{p}} + \rho_{\mathrm{re}}\dot{V}_{\mathrm{re,p}} + \rho_{\mathrm{air}}\dot{V}_{\mathrm{s}} + \rho_{\mathrm{re}}\dot{V}_{\mathrm{re,s}}. \tag{2.26}$$

Note that the remaining ash is neglected. Moreover, there is sand taken out from the bed, cleaned from the ash and brought back to the furnace. However, the amount of sand in the fluidized bed should stay constant. Therefore, the sand circuit is not taken into account.

2.2.5 Mass Flow for Fluidization

The mass flow that is fluidizing the bed is the sum of the primary air and the primary recirculated gas and can therefore be written as

$$\rho_{\mathrm{fluid}}\dot{V}_{\mathrm{fluid}} = \rho_{\mathrm{air}}\dot{V}_{\mathrm{p}} + \rho_{\mathrm{re}}\dot{V}_{\mathrm{re,p}}. \tag{2.27}$$

Chapter 3

System Identification

For the control algorithm, which will be discussed in Chapter 5, a model of the process under consideration is needed. This chapter is about finding this model in terms of a mathematical description, using both physical knowledge and measurement data of the process.

3.1 Introduction

The aim of this work is to develop an efficient, robust and real time capable model predictive controller that keeps temperatures, oxygen levels and flow rates at a desired level. In order to do that, the controller needs to know how these outputs are influenced by the inputs, e.g. air flow rates into the furnace or the waste feed. Therefore, a plant model is developed which gives a mathematical description of the relations between inputs and outputs.

Besides using a model for controller design, there are many other cases where models are useful: Models are used to simulate a system's behavior and to get a better understanding of how a process works [10]

There are different ways to obtain models. One way is to build *first-principle models*. These models are built by the usage of physical principles, e.g. Newton's law or laws of thermodynamics. Another approach is to build the model by using input and output data from the process under consideration (Figure 3.1). This kind of modelling is called *system identification* [10]. There are basically two types of models that can be build using measurement data: *Grey box models* and *black box models*. Grey box models have a certain structure that is recieved from physical equations. However, there are unknown parameters in these equations that are then identified using measurement data. A black box model is a model where the structure is not based on physical equations. An artificial neural network is an expamle for a black box model [11][12][13].

To get measurement data to find the parameters in the model, experiments need to be done. These experiments can include choosing the input signals to excite the system under consideration. The input signals should be chosen appropriate to the process in terms of amplitudes

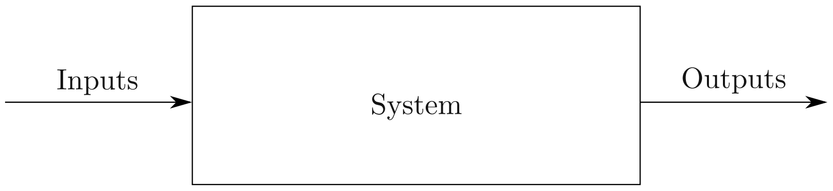


Figure 3.1: The aim of system identification is to find a model of a dynamical system using measured input and output data.

and frequencies. As the input signal should contain a wide frequency range, pseudorandom binary sequence (PRBS) signals or swept sinusoid signals are a popular choice [10].

Yet, there are cases where the input signal cannot be freely chosen. If the system is unstable, a controller is needed to stabilize the system while carrying out the experiments. Moreover, it can be too expensive to carry out experiments, what is often the case in industrial applications. Then the measurement data are collected under regular operations, where the system is operating in closed-loop. However, problems can appear when using closed-loop data for system identification. This is because in closed-loop data, the disturbances acting on the system are correlated with the input signals applied by the controller. Research on closed-loop identification have been done in [14][15]. When identifying systems operating in closed-loop, it can be helpful if some signal (e.g. white noise or impulses) is added to the control signal in order to additionally excite the system.

For this work, historical data form a plant operating in closed-loop is taken to identify the system. No additional experiments could have been done to generate data for system identification.

3.2 Data Preprocessing

Before the data collected on the plant can be used for system identification, some data preprocessing is necessary. This includes removing outliers, filtering, resampling and normalizing the data. Furthermore, the data set is divided into identification data, validation data and test data. In Section 3.2.3 it is described, why different data sets are needed.

3.2.1 Normalizing Data

The data collected are measured in different units. Hence, the signals are of very different scales. However, performing computations with data lying in very different ranges can lead to higher numerical errors. Therefore, the data are normalized by subtracting the mean μ and dividing by the standard deviation σ:

$$\boldsymbol{\mathcal{Y}}_i{}' = \frac{\boldsymbol{\mathcal{Y}}_i - \mu_i}{\sigma_i}, \tag{3.1}$$

with

$$\begin{aligned} \mu_i &= \frac{1}{N}\sum_{k=1}^{N} y_i(k) \\ \sigma_i^2 &= \frac{1}{N-1}\sum_{k=1}^{N}(y_i(k)-\mu)^2, \end{aligned} \tag{3.2}$$

where $y_i(k)$ is the output number i at the time step k and N is the number of data points collected. The whole signal of one output stored in a vector is denoted as $\boldsymbol{\mathcal{Y}}_i$.

3.2.2 Filtering and Resampling

The data collected at the plant is not sampled uniformly. In many cases, a new data point is only stored if the measured signal changes within a certain range. As long as changes in the signal are very small, no new data points are stored. Thus, the data points are collected at different times for different signals. However, for the system identification procedure, the data needs to be uniformly sampled.

Before the signals are resampled, an anti-aliasing filter is applied. This is necessary because signals can have been collected with higher sampling frequencies than used for system identification later. Hence, the signals can contain high-frequency components, what leads to aliasing when downsampling the signals.

After filtering the signal, the data points are interpolated using piecewise cubic Hermite interpolating polynomials (pchip). This interpolation method is used because the signals under consideration are physical signals and change smoothly, not abruptly.

Filtering and resampling is carried out using the MATLAB function `resample`. Figure 3.2 shows two signals before and after filtering and resampling.

3.2.3 Data Division

The data used in this work have been collected from April 6, 2020 at 10 p.m to April 12, 2020 at 8 a.m. This data set is divided into three parts:

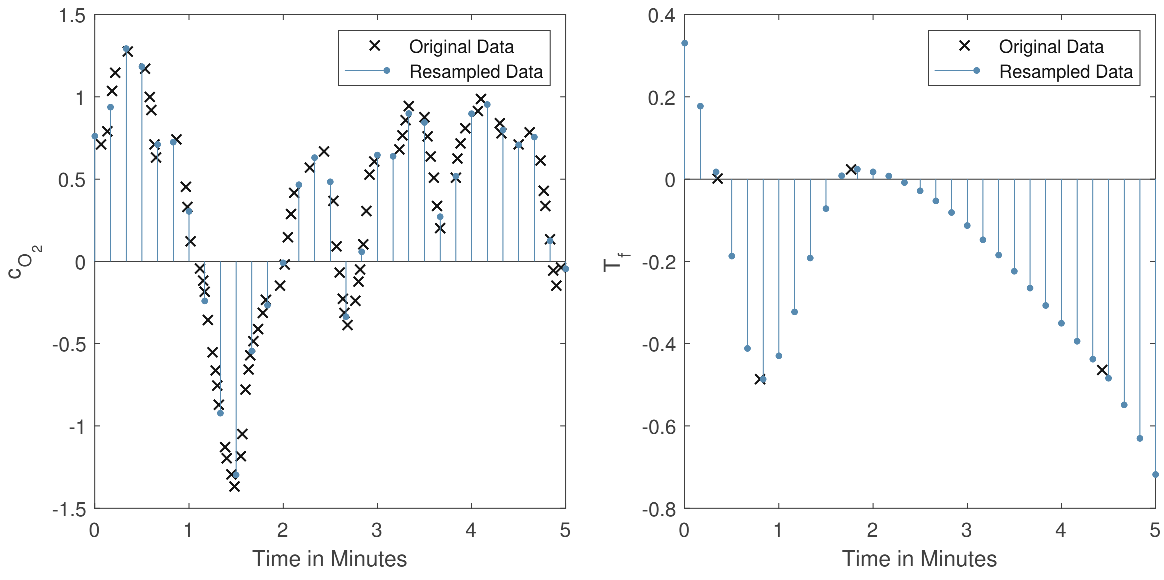


Figure 3.2: Data Preprocessing: The original data is sampled fast for c_{O_2} and slow for T_f. After resampling, the data sequences are uniformly sampled.

- Identification data (50 %)
- Validation data (25 %)
- Test data (25 %)

The identification data is used for parameter estimation as described in Section 3.5.2. For model validation, it is important to use a data set that has not been used for parameter estimation. Therefore, the validation data is used for model validation as described in Section 3.7. In Chapter 4, a third data set is needed for the computation of the fitness value. Therefore, the test data is used.

3.3 Definition of Input and Output Variables

In Chapter 2, both differential and algebraic equations have been developed to describe the fluidized bed furnace. These equations contain constants and variables. The variables are separated into input variables $\boldsymbol{u}$ and output variables $\boldsymbol{y}$: Output variables are desired to be controlled and lie as close as possible to a certain reference value or trajectory. These variables are influenced by the input variables.

Note that usually inputs are only variables that can be manipulated. Variables, that cannot be manipulated, but also have an influence on the outputs, are called disturbances. However, for the purpose of identification it does not matter if variables can be manipulated or not, as long as the variables can be measured.

The input and output variables are defined as follows:

u_1 :	waste feed screw conveyor rotational speed (n_{msw})
u_2 :	primary air volume flow ($\dot{V}_{\mathrm{p}}$)
u_3 :	primary recirculated gas volume flow ($\dot{V}_{\mathrm{re,p}}$)
u_4 :	secondary air volume flow ($\dot{V}_{\mathrm{s}}$)
u_5 :	secondary recirculated gas volume flow ($\dot{V}_{\mathrm{s,re}}$)
u_6 :	primary air enthalpy flow ($\dot{V}_{\mathrm{p}}T_{\mathrm{fluid}}$)
u_7 :	primary recirculated gas enthalpy flow ($\dot{V}_{\mathrm{re,p}}T_{\mathrm{re}}$)
u_8 :	secondary air enthalpy flow ($\dot{V}_{\mathrm{s}}T_{\mathrm{s}}$)
u_9 :	secondary recirculated gas enthalpy flow ($\dot{V}_{\mathrm{re,s}}T_{\mathrm{re}}$)

y_1:	flue gas volume flow ($\dot{V}_{\mathrm{f}}$)
y_2:	furnace head temperature (T_{f})
y_3:	fluidized bed temperature (T_{b})
y_4:	oxygen concentration furnace head ($c_{\mathrm{O_2}}$)
y_5:	gas for fluidization volume flow ($\dot{V}_{\mathrm{fluid}}$)

The inputs u_6 to u_9 are the products of a volume flow (as a standard volume flow) with its temperature. These products are proportional to the corresponding enthalpy flows. This allows formulating equations that are linear in inputs, even though the original equations are non-linear because of the product of mass flow and temperature.

3.4 From Physical Equations to Polynomial Models

The equations developed in Chapter 2 are now reformulated, linearized and discretized before estimating the parameters.

The aim of this section is to find a description for each output using a first-order polynomial model in discrete-time:

$$y_i(k+1) = a_{i1}y_1(k) + \ldots + a_{i5}y_5(k) + b_{i1}u_1(k) + \ldots + b_{i9}u_9(k), \tag{3.3}$$

that is now a function of the sample number k, not of the time t.

For more convenience, this can be written using a matrix formulation:

$$y_i(k+1) = \boldsymbol{a_i}\boldsymbol{y}(k) + \boldsymbol{b_i}\boldsymbol{u}(k). \tag{3.4}$$

3.4.1 Flue Gas Mass Flow

The flue gas mass flow can be written as

$$\rho_\mathrm{f}\dot{V}_\mathrm{f} = k_\mathrm{msw}n_\mathrm{msw} + \rho_\mathrm{air}\dot{V}_\mathrm{p} + \rho_\mathrm{re}\dot{V}_\mathrm{re,p} + \rho_\mathrm{air}\dot{V}_\mathrm{s} + \rho_\mathrm{re}\dot{V}_\mathrm{re,s} - \dot{m}_\mathrm{ash}. \tag{3.5}$$

Equation (3.5) is then written using the inputs and outputs from Section 3.3. The parameters are combined in new parameters b_{11} to b_{15}:

$$y_1(k) = b_{11}u_1(k) + b_{12}u_2(k) + b_{13}u_3(k) + b_{14}u_4(k) + b_{15}u_5(k), \tag{3.6}$$

what is an algebraic equation now. However, a better model of the flue gas mass flow can be achieved using a first-order difference equation:

$$y_1(k+1) = \begin{bmatrix} a_{11} & 0 & 0 & 0 & 0 \end{bmatrix} \boldsymbol{y}(k) + \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} & b_{15} & 0 & 0 & 0 & 0 \end{bmatrix} \boldsymbol{u}(k). \tag{3.7}$$

3.4.2 Furnace Head Temperature

The energy balance for the furnace head led to the following differential equation:

$$\begin{aligned} C_\mathrm{f}\frac{\mathrm{d}T_\mathrm{f}}{\mathrm{d}t} &= c_\mathrm{p,bf}(k_\mathrm{msw}n_\mathrm{msw} + \rho_\mathrm{air}\dot{V}_\mathrm{p} + \rho_\mathrm{re}\dot{V}_\mathrm{re,p})T_\mathrm{b} \\ &+ c_\mathrm{p,air}\rho_\mathrm{air}\dot{V}_\mathrm{s}T_\mathrm{s} \\ &+ c_\mathrm{p,re}\rho_\mathrm{re}\dot{m}_\mathrm{re,s}T_\mathrm{re} \\ &- c_\mathrm{p,f}\rho_\mathrm{f}\dot{V}_\mathrm{f}T_\mathrm{f} \\ &+ H_\mathrm{c}k_\mathrm{msw}n_\mathrm{msw} - \frac{H_\mathrm{c}}{k_\mathrm{f}}(\rho_\mathrm{air}\dot{V}_\mathrm{p}c_{\mathrm{O_2,m,air}} + \rho_\mathrm{re}\dot{V}_\mathrm{re,p}c_{\mathrm{O_2,m,re}}). \end{aligned} \tag{3.8}$$

Now, this equation is rewritten using the input and output notation:

$$\begin{aligned} C_\mathrm{f}\frac{\mathrm{d}y_2}{\mathrm{d}t} &= c_\mathrm{p,bf}(k_\mathrm{msw}u_1 + \rho_\mathrm{air}u_2 + \rho_\mathrm{re}u_3)y_3 \\ &+ c_\mathrm{p,air}\rho_\mathrm{air}u_8 \\ &+ c_\mathrm{p,re}\rho_\mathrm{re}u_9 \\ &- c_\mathrm{p,f}\rho_\mathrm{f}y_1y_2 \\ &+ H_\mathrm{c}k_\mathrm{msw}u_1 - \frac{H_\mathrm{c}}{k_\mathrm{f}}(\rho_\mathrm{air}u_2c_{\mathrm{O_2,m,air}} + \rho_\mathrm{re}u_3c_{\mathrm{O_2,m,re}}). \end{aligned} \tag{3.9}$$

The first and fourth term on the right-hand side are non-linear, because of the product of input and output or output and output. These products are linearized with respect to the operating point:

$$u_iy_j \approx u_iy_{j,0} + u_{i,0}y_j, \tag{3.10}$$

where $u_{i,0}$ and $y_{j,0}$ are input and output at the operating point. This is done for all of the products in (3.9).

The next step is to discretize the equation. Although methods exist to estimate parameters for continuous-time models, it seems more convenient in this case to estimate parameter for a discretized model, because measurement data are discrete anyway and the model used for control should be a discrete-time model as well. Therefore, the derivative is approximated using a first-order divided difference:

$$\frac{\mathrm{d}y_2(t)}{\mathrm{d}t} \approx \frac{y_2(t) - y_2(t - T_\mathrm{s})}{T_\mathrm{s}}, \tag{3.11}$$

with the sampling time T_s. All measured signals are given in discrete time and sampled uniformly, therefore $y(t) = y(kT_\mathrm{s})$.

After linearization and discretization, Equation (3.9) can be written as a function of k:

$$y_2(k+1) = \begin{bmatrix} a_{21} & a_{22} & a_{23} & 0 & 0 \end{bmatrix} \boldsymbol{y}(k) + \begin{bmatrix} b_{21} & b_{22} & b_{23} & 0 & 0 & 0 & 0 & b_{28} & b_{29} \end{bmatrix} \boldsymbol{u}(k), \tag{3.12}$$

where all the constants in (3.9), the operating points and the sampling time T_s are combined in new parameters a_{2j} and b_{2j}.

3.4.3 Fluidized Bed Temperature

The differential equation for the fluidized bed temperature

$$\begin{aligned} C_\mathrm{b}\frac{\mathrm{d}T_\mathrm{b}}{\mathrm{d}t} &= c_\mathrm{p,air}\rho_\mathrm{air}\dot{V}_\mathrm{p}T_\mathrm{p} \\ &+ c_\mathrm{p,re}\rho_\mathrm{re}\dot{V}_\mathrm{re,p}T_\mathrm{re} \\ &+ c_\mathrm{p,msw}k_\mathrm{msw}n_\mathrm{msw}T_\mathrm{msw} \\ &- c_\mathrm{p,bf}(k_\mathrm{msw}n_\mathrm{msw} + \rho_\mathrm{air}\dot{V}_\mathrm{p} + \rho_\mathrm{re}\dot{V}_\mathrm{re,p})T_\mathrm{b} \\ &+ \frac{H_\mathrm{c}}{k_\mathrm{f}}(\rho_\mathrm{air}\dot{V}_\mathrm{p}c_{\mathrm{O_2,m,air}} + \rho_\mathrm{re}\dot{V}_\mathrm{re,p}c_{\mathrm{O_2,m,re}}) \end{aligned} \tag{3.13}$$

is again written using inputs and outputs:

$$\begin{aligned} C_\mathrm{b}\frac{\mathrm{d}y_3}{\mathrm{d}t} &= c_\mathrm{p,air}\rho_\mathrm{air}u_6 \\ &+ c_\mathrm{p,re}\rho_\mathrm{re}u_7 \\ &+ c_\mathrm{p,msw}k_\mathrm{msw}u_1T_\mathrm{msw} \\ &- c_\mathrm{p,bf}(k_\mathrm{msw}u_1 + \rho_\mathrm{air}u_2 + \rho_\mathrm{re}u_3)y_3 \\ &+ \frac{H_\mathrm{c}}{k_\mathrm{f}}(\rho_\mathrm{air}u_2c_{\mathrm{O_2,m,air}} + \rho_\mathrm{re}u_3c_{\mathrm{O_2,m,re}}). \end{aligned} \tag{3.14}$$

Now, the equation is linearized according to (3.10) and discretized according to (3.11). This leads to the following polynomial model:

$$y_3(k+1) = \begin{bmatrix}0 & 0 & a_{33} & 0 & 0\end{bmatrix} \boldsymbol{y}(k) + \begin{bmatrix}b_{31} & b_{32} & b_{33} & 0 & 0 & b_{36} & b_{37} & 0 & 0\end{bmatrix} \boldsymbol{u}(k). \tag{3.15}$$

3.4.4 Oxygen Concentration Furnace Head

A description for the oxygen concentration at the furnace head has been received by writing a mass balance for the furnace head as

$$\begin{aligned} M_\mathrm{f} \frac{\mathrm{d}c_{\mathrm{O_2,m}}}{\mathrm{d}t} &= \rho_\mathrm{air} \dot{V}_\mathrm{s} c_{\mathrm{O_2,m,air}} + \rho_\mathrm{re} \dot{V}_\mathrm{re,s} c_{\mathrm{O_2,m,re}} - \rho_\mathrm{f} \dot{V}_\mathrm{f} c_{\mathrm{O_2}} \frac{R_\mathrm{f}}{R_{\mathrm{O_2}}} \\ &- (k_\mathrm{f} k_\mathrm{msw} n_\mathrm{msw} - c_{\mathrm{O_2,m,air}} \rho_\mathrm{air} \dot{V}_\mathrm{p} - c_{\mathrm{O_2,m,re}} \rho_\mathrm{re} \dot{V}_\mathrm{re,p}). \end{aligned} \tag{3.16}$$

Again, the equation is written using inputs and outputs:

$$\begin{aligned} M_\mathrm{f} \frac{\mathrm{d}y_4}{\mathrm{d}t} &= \rho_\mathrm{air} u_4 c_{\mathrm{O_2,m,air}} + \rho_\mathrm{re} u_5 c_{\mathrm{O_2,m,re}} - \rho_\mathrm{f} y_1 y_4 \frac{R_\mathrm{f}}{R_{\mathrm{O_2}}} \\ &- (k_\mathrm{f} k_\mathrm{msw} u_1 - c_{\mathrm{O_2,m,air}} \rho_\mathrm{air} u_2 - c_{\mathrm{O_2,m,re}} \rho_\mathrm{re} u_3). \end{aligned} \tag{3.17}$$

After linearization, discretization and combining parameters, the equation can be written as

$$y_4(k+1) = \begin{bmatrix}a_{41} & 0 & 0 & a_{44} & 0\end{bmatrix} \boldsymbol{y}(k) + \begin{bmatrix}b_{41} & b_{42} & b_{43} & b_{44} & b_{45} & 0 & 0 & 0 & 0\end{bmatrix} \boldsymbol{u}(k). \tag{3.18}$$

3.4.5 Mass Flow for Fluidization

The mass flow for fluidization can be written as the following sum:

$$\rho_\mathrm{fluid} \dot{V}_\mathrm{fluid} = \rho_\mathrm{air} \dot{V}_\mathrm{p} + \rho_\mathrm{re} \dot{V}_\mathrm{re,p}. \tag{3.19}$$

However, due to inertia and time delays in the system, a polynomial model better describes the mass flow for fluidization:

$$y_5(k+1) = \begin{bmatrix}0 & 0 & 0 & 0 & a_{55}\end{bmatrix} \boldsymbol{y}(k) + \begin{bmatrix}0 & b_{42} & b_{43} & 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix} \boldsymbol{u}(k). \tag{3.20}$$

3.5 Identification of Linear State-Space Models

For multiple-input and multiple-output (MIMO) systems, a state-space model is a convenient way to describe the physical processes. Therefore, the parameter estimation procedure is done in the state-space. State-space representation uses first-order differential or difference equations to describe a continuous-time or discrete-time system, respectively. In discrete time, a state-space model has the following form:

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k), \\ \boldsymbol{y}(k) &= \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k), \end{aligned} \tag{3.21}$$

with the variables

$\boldsymbol{A} \in \mathbb{R}^{n \times n}$	state matrix,	$\boldsymbol{x} \in \mathbb{R}^{n}$	state vector,
$\boldsymbol{B} \in \mathbb{R}^{n \times r}$	input matrix,	$\boldsymbol{u} \in \mathbb{R}^{r}$	input vector,
$\boldsymbol{C} \in \mathbb{R}^{m \times n}$	output matrix,	$\boldsymbol{y} \in \mathbb{R}^{m}$	output vector,
$\boldsymbol{D} \in \mathbb{R}^{m \times r}$	feedthrough matrix,		

where n indicates the order of the model (the number of the states), r indicates the number of inputs and m indicates the number of outputs. Note that a state-space representation is not unique, whereas other forms, like transfer functions, give a unique description of a dynamical system.

For the identification procedure the system is assumed to be time-invariant. This means the matrices $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ and $\boldsymbol{D}$ are constant. The matrix $\boldsymbol{D}$ is assumed to be the zero matrix: The input $\boldsymbol{u}$ cannot influence the output $\boldsymbol{y}$ within the same sample k, what is the case for almost all physical systems.

The aim of system identification is now to determine the matrices in (3.21).

3.5.1 Model Structure

In Section 3.4, polynomial models have been developed for every single output y_{i} to describe the input-output behavior. These polynomial models are now combined to get a closed-form representation for the whole MIMO system. As these polynomial models are already first-order difference equations, they can easily be written in state-space form with the state vector $\boldsymbol{x}$ equal to the output vector $\boldsymbol{y}$:

$$\begin{aligned}\boldsymbol{x}(k+1) = &\begin{bmatrix} a_{11} & 0 & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 \\ 0 & 0 & a_{33} & 0 & 0 \\ a_{41} & 0 & 0 & a_{44} & 0 \\ 0 & 0 & 0 & 0 & a_{55} \end{bmatrix} \boldsymbol{x}(k) \\ &+ \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} & b_{15} & 0 & 0 & 0 & 0 \\ b_{21} & b_{22} & b_{23} & 0 & 0 & 0 & 0 & b_{28} & b_{29} \\ b_{31} & b_{32} & b_{33} & 0 & 0 & b_{36} & b_{37} & 0 & 0 \\ b_{41} & b_{42} & b_{43} & b_{44} & b_{45} & 0 & 0 & 0 & 0 \\ 0 & b_{52} & b_{53} & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \boldsymbol{u}(k), \end{aligned} \tag{3.22}$$

with the input vector

$$\boldsymbol{u} = \begin{bmatrix} n_{\text{msw}}, & \dot{m}_{\text{p}}, & \dot{m}_{\text{p,re}}, & \dot{m}_{\text{s}}, & \dot{m}_{\text{s,re}}, & \dot{m}_{\text{p}}T_{\text{re}} & \dot{m}_{\text{p,re}}T_{\text{p}}, & \dot{m}_{\text{s}}T_{\text{s}}, & \dot{m}_{\text{s,re}}T_{\text{re}}, & \end{bmatrix}^T$$

and the output vector

$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} = \boldsymbol{I} \begin{bmatrix} \dot{m}_{\text{f}}, & T_{\text{f}}, & T_{\text{b}}, & c_{\text{O2}}, & \dot{m}_{\text{fluid}} \end{bmatrix}^T.$$

The output matrix $\boldsymbol{C}$ is the identity matrix in this case, since state vector and output vector contain the same variables.

Additionally to the model developed so far, a higher-order model is developed now. These two models are later compared to each other. So far, first-order difference equations have been used to describe the input-output behavior. However, a higher order model could possibly perform better at modeling the system's behavior. Thus, the first-order difference equations with the form

$$y_i(k+1) = \boldsymbol{a_i}\boldsymbol{y}(k) + \boldsymbol{b_i}\boldsymbol{u}(k), \tag{3.23}$$

with the parameter vectors $\boldsymbol{a_i}$ and $\boldsymbol{b_i}$ are extended to second-order difference equations:

$$y_i(k+1) = \begin{bmatrix} \tilde{\boldsymbol{a}}_{\boldsymbol{1i}} & \tilde{\boldsymbol{a}}_{\boldsymbol{2i}} \end{bmatrix} \begin{bmatrix} \boldsymbol{y}(k) \\ \boldsymbol{y}(k-1) \end{bmatrix} + \begin{bmatrix} \tilde{\boldsymbol{b}}_{\boldsymbol{1i}} & \tilde{\boldsymbol{b}}_{\boldsymbol{2i}} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}(k) \\ \boldsymbol{u}(k-1) \end{bmatrix}. \tag{3.24}$$

This equation is now also a function of $\boldsymbol{y}(k-1)$ and $\boldsymbol{u}(k-1)$. The non-zero elements in the parameter vectors $\tilde{\boldsymbol{a}}_{\boldsymbol{1i}}$ and $\tilde{\boldsymbol{a}}_{\boldsymbol{2i}}$ are the same as in $\boldsymbol{a_i}$. The same applies for $\tilde{\boldsymbol{b}}_{\boldsymbol{1i}}$, $\tilde{\boldsymbol{b}}_{\boldsymbol{2i}}$ and $\boldsymbol{b_i}$. This means that the dependencies of the outputs $y_i i$ on the other outputs and on the inputs did not change when increasing the model order.

To get a state-space representation of the increased-order model, each second-order difference equation needs to be written as two first-order difference equations. This is done by introducing a new state

$$\begin{aligned} y_i(k+1) &= \tilde{\boldsymbol{a}}_{\boldsymbol{1i}}\boldsymbol{y}(k) + \tilde{\boldsymbol{b}}_{\boldsymbol{1i}}\boldsymbol{u}(k) + \tilde{x}_i(k), \\ \tilde{x}_i(k+1) &= \tilde{\boldsymbol{a}}_{\boldsymbol{2i}}\boldsymbol{y}(k) + \tilde{\boldsymbol{b}}_{\boldsymbol{2i}}\boldsymbol{u}(k) \end{aligned} \tag{3.25}$$

for every output y_i. The increased-order model can be written in state-space representation:

$$\begin{aligned} \begin{bmatrix} y_1(k+1) \\ y_2(k+1) \\ y_3(k+1) \\ y_4(k+1) \\ y_5(k+1) \\ \tilde{x}_1(k+1) \\ \tilde{x}_2(k+1) \\ \tilde{x}_3(k+1) \\ \tilde{x}_4(k+1) \\ \tilde{x}_5(k+1) \end{bmatrix} &= \begin{bmatrix} a_{11} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & a_{33} & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ a_{41} & 0 & 0 & a_{44} & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & a_{55} & 0 & 0 & 0 & 0 & 1 \\ a_{66} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ a_{76} & a_{77} & a_{78} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & a_{88} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ a_{96} & 0 & 0 & a_{99} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & a_{1010} & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} y_1(k) \\ y_2(k) \\ y_3(k) \\ y_4(k) \\ y_5(k) \\ \tilde{x}_1(k) \\ \tilde{x}_2(k) \\ \tilde{x}_3(k) \\ \tilde{x}_4(k) \\ \tilde{x}_5(k) \end{bmatrix} \\ &+ \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} & b_{15} & 0 & 0 & 0 & 0 \\ b_{21} & b_{22} & b_{23} & 0 & 0 & 0 & 0 & b_{28} & b_{29} \\ b_{31} & b_{32} & b_{33} & 0 & 0 & b_{36} & b_{37} & 0 & 0 \\ b_{41} & b_{42} & b_{43} & b_{44} & b_{45} & 0 & 0 & 0 & 0 \\ 0 & b_{52} & b_{53} & 0 & 0 & 0 & 0 & 0 & 0 \\ b_{11} & b_{12} & b_{13} & b_{14} & b_{15} & 0 & 0 & 0 & 0 \\ b_{21} & b_{22} & b_{23} & 0 & 0 & 0 & 0 & b_{28} & b_{29} \\ b_{31} & b_{32} & b_{33} & 0 & 0 & b_{36} & b_{37} & 0 & 0 \\ b_{41} & b_{42} & b_{43} & b_{44} & b_{45} & 0 & 0 & 0 & 0 \\ 0 & b_{52} & b_{53} & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_1(k) \\ u_2(k) \\ u_3(k) \\ u_4(k) \\ u_5(k) \\ u_6(k) \\ u_7(k) \\ u_8(k) \\ u_9(k) \end{bmatrix} \end{aligned} \tag{3.26}$$

with a state vector $\boldsymbol{x}$ containing the five outputs $\boldsymbol{y}$ followed by five additional states as mentioned in (3.25). Thus, the output matrix becomes

$$\boldsymbol{C} = \begin{bmatrix} \boldsymbol{I}_{5\times5} & \boldsymbol{0}_{5\times5} \end{bmatrix}. \tag{3.27}$$

3.5.2 Parameter Estimation

The next step is to estimate the unknown parameters in the state-space matrices in (3.22) and (3.26) using measurement data.

There are data points collected for the input $\boldsymbol{u}$ and output $\boldsymbol{y}$. The parameters in the model should be identified in a way that the output $\hat{\boldsymbol{y}}$, that is generated by the model, fits the measured output as good as possible. The error of the model output $\hat{\boldsymbol{y}}$ is defined for every sample point k as

$$\boldsymbol{e}(k) = \boldsymbol{y}(k) - \hat{\boldsymbol{y}}(k). \tag{3.28}$$

To evaluate the goodness of the fit, the mean squared error over N samples is defined as a cost function:

$$V(\boldsymbol{\theta}) = \frac{1}{N}\sum_{k=1}^{N} \boldsymbol{e}(k)^T \boldsymbol{e}(k), \tag{3.29}$$

where $\boldsymbol{\theta}$ is the parameter vector, containing all the unknown parameters in the model. To choose a quadratic cost function like in (3.29) is a common choice for a cost function [10].

The parameters are estimated by minimizing the cost function

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}\in \mathrm{D}} V(\boldsymbol{\theta}). \tag{3.30}$$

Computation of the Model Output

The model output $\hat{\boldsymbol{y}}$ can be computed in different ways: one way is to use only the measured input signal $\boldsymbol{u}$. This input signal is fed into the model and the model output $\hat{\boldsymbol{y}}$ is computed. This is like simulating the response using the model and the measured input, what works for both polynomial models and state-space models.

Another way of calculating $\hat{\boldsymbol{y}}$ is to use both the measured input signal $\boldsymbol{u}$ and the measured output signal $\boldsymbol{y}$. For a polynomial model like (3.23) or (3.24) this can easily be done by using measurement signals $\boldsymbol{y}$ on the right-hand side of the equation to compute $\hat{\boldsymbol{y}}$:

$$\hat{y}_i(k+1) = \boldsymbol{a_i}\boldsymbol{y}(k) + \boldsymbol{b_i}\boldsymbol{u}(k). \tag{3.31}$$

When computing $\hat{\boldsymbol{y}}$ this way, the one-step-ahead prediction error is going to be minimized when minimizing the cost function $V(\boldsymbol{\theta})$.

For state-space models, this cannot be done that easily, because the state vector $\boldsymbol{x}$ cannot be measured in general. However, the measured output $\boldsymbol{y}$ can be used to correct the calculated state vector $\hat{\boldsymbol{x}}$:

$$\begin{aligned} \hat{\boldsymbol{x}}(k+1) &= \boldsymbol{A}\hat{\boldsymbol{x}}(k) + \boldsymbol{B}\boldsymbol{u}(k) + \boldsymbol{K}(\boldsymbol{y}(k) - \boldsymbol{C}\hat{\boldsymbol{x}}(k)), \\ \hat{\boldsymbol{y}}(k) &= \boldsymbol{C}\hat{\boldsymbol{x}}(k), \end{aligned} \tag{3.32}$$

where $\boldsymbol{K}$ can be interpreted as a Kalman-Filter gain matrix. The parameters in $\boldsymbol{K}$ can be estimated using measurement data as well. If this is done, this model structure is called *directly parametrized innovations form* [10].

If $\hat{\boldsymbol{y}}$ is calculated using a one-step-ahead prediction, the sampling time of the model must be chosen very appropriate to the dynamics of the system under consideration. If the model is sampled very fast, the current output $\boldsymbol{y}(k)$ would always be a good prediction for the next output $\hat{\boldsymbol{y}}(k+1)$. For the fluidized bed furnace, the dynamics are very different for each output: The furnace bed temperature changes very slow compared to the other outputs, due to the high heat capacity of the bed. Therefore, it is impossible to choose a sampling time appropriate for all outputs. Thus, a simulation is used to compute $\hat{\boldsymbol{y}}$, using only the measured input signal $\boldsymbol{u}$, not the measured output signal $\boldsymbol{y}$.

Minimization of the Cost Function

For some cases, e.g. if a polynomial model is used and the model response is computed using a one-step-ahead prediction, there are analytical solutions to minimize the cost function $V(\boldsymbol{\theta})$. However, in this case a numerical, iterative technique must be applied to find a minimum because the model output $\hat{\boldsymbol{y}}(k+1)$ is a function of a previous model output $\hat{\boldsymbol{y}}(k)$ and cannot be given explicit in terms of measurement data. An iterative approach also needs to be applied when parameters of nonlinear models are estimated, since the parameter estimation problem is then a nonlinear least-squares problem (if a quadratic cost function as in (3.29) is defined). For this work, the MATLAB System Identification Toolbox [16] is used for parameter estimation. This toolbox uses different numerical algorithms for minimizing the cost function, such as the Levenberg-Marquardt algorithm :

$$\hat{\boldsymbol{\theta}}_{i+1} = \hat{\boldsymbol{\theta}}_i + \left(\frac{1}{N}\sum_{k=1}^{N}\boldsymbol{J}(k,\hat{\boldsymbol{\theta}})\boldsymbol{J}^T(k,\hat{\boldsymbol{\theta}}) + \lambda\boldsymbol{I}\right)^{-1}\left(-\frac{1}{N}\sum_{k=1}^{N}\boldsymbol{J}(k,\hat{\boldsymbol{\theta}})\boldsymbol{e}(k,\theta)\right), \tag{3.33}$$

with the parameter estimate $\hat{\boldsymbol{\theta}}_i$ at the i^{th}-iteration step, the Jacobi matrix $\boldsymbol{J}(k,\hat{\boldsymbol{\theta}}) \in \mathbb{R}^{n_\theta \times m}$ of the model output $\hat{\boldsymbol{y}}(k,\hat{\boldsymbol{\theta}})$, a scalar parameter λ and the error $\boldsymbol{e}(k,\hat{\boldsymbol{\theta}})$ as specified in (3.28). Also the Gauss-Newton algorithm is used by the System Identification Toolbox, where $\lambda = 0$. More detailed information and other search schemes can be found in [10].

3.6 Identification of Artificial Neural Networks

In order to evaluate the performance of the linear state-space models, an artificial neural network (ANN) is designed and trained. The performance of the ANN is then compared to those of the linear state-space modes. The same measurement data are used to train the ANN.

3.6.1 Network Architecture

The MATLAB Deep Learning Toolbox [17] is used to design and train the ANN. Figure 3.3 shows the architecture of the network. The input and output vectors are those that are specified in Section 3.3. The ANN is designed as a recurrent radial basis function network. This means, not only the acctual inputs $\boldsymbol{u}$ are inputs to the network, but the network's outputs $\hat{\boldsymbol{y}}$ are fed back as inputs to the network as well [12], and a radial basis function is used as an activation function.

The network uses inputs $\boldsymbol{u}$ and the network outputs $\hat{\boldsymbol{y}}$ from two prior periods as network inputs. These network inputs are now called $\boldsymbol{a_0}$.

Before the network inputs reach the neurons in the hidden layer, they are weighted. In the hidden layer, a constant bias term is added and a non-linear function φ is applied. This leads to the output of the hidden layer

$$\boldsymbol{a_1} = \varphi(\boldsymbol{b_1} + \boldsymbol{W_{10}} \boldsymbol{a_0}), \tag{3.34}$$

with the network inputs $\boldsymbol{a_0} \in \mathbb{R}^{28}$, the weighting matrix $\boldsymbol{W_{10}} \in \mathbb{R}^{10\times 28}$, the bias vector $\boldsymbol{b_1} \in \mathbb{R}^{10}$ and the output of the hidden layer $\boldsymbol{a_1} \in \mathbb{R}^{10}$. The elements in the bias vector $\boldsymbol{b_1}$ determine the centers of the radial basis functions.

The non-linear function applied in the neurons is call activation function. In this work, a radial basis function is used as an activation function:

$$\varphi(\boldsymbol{x}) = \frac{e^{-\boldsymbol{x}^2}}{\sum_{i=1}^{n} x_i}. \tag{3.35}$$

The output $\hat{\boldsymbol{y}}(k)$ is then computed as a linear combination of $\boldsymbol{a_1}$. Again, a bias term is added:

$$\hat{\boldsymbol{y}} = \boldsymbol{b_2} + \boldsymbol{W_{21}} \boldsymbol{a_1}, \tag{3.36}$$

where $\boldsymbol{W_{21}} \in \mathbb{R}^{5\times 10}$ is again a weighting matrix and $\boldsymbol{b_2} \in \mathbb{R}^{5}$ is a bias vector.

3.6.2 Network Training

The aim of network training is to use measurement data to estimate the weights and biases in the network. This is a similar task to the parameter estimation described in Section 3.5.2. Again, the mean-squared-error is used as a cost function as defined in (3.29).

For the minimization of the cost function, a Levenberg–Marquardt algorithm is used (see (3.33)). The calculation of the Jacobian is done by backpropagation. A deeper insight on the training of ANNs using a Levenberg-Marquardt algorithm and backpropagation is given in [13].

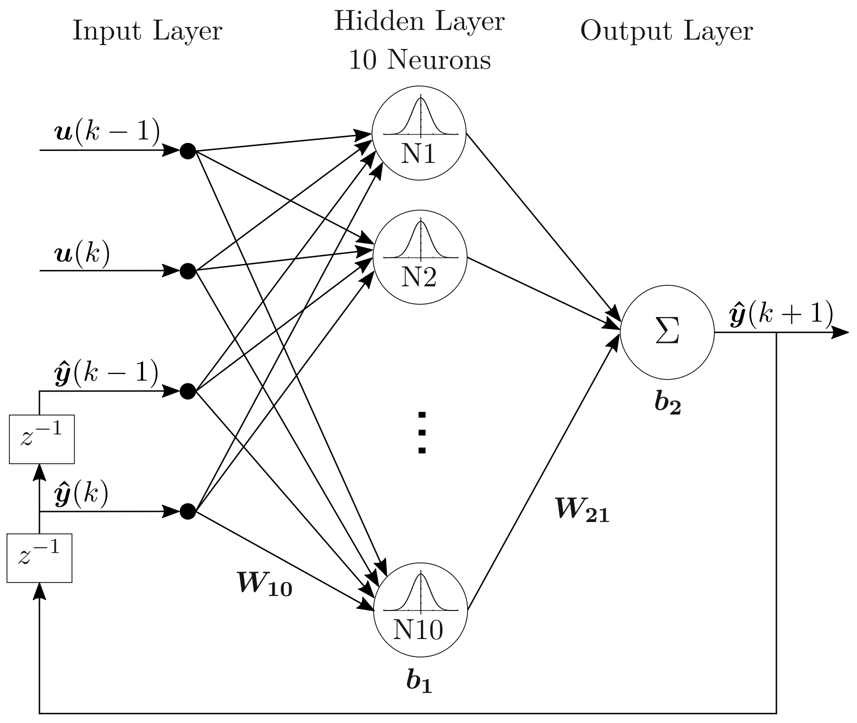


Figure 3.3: Artificial Neural Network Architecture.

3.7 Model Validation

So far, two state-space models of different orders and an artificial neural network have been identified to describe the fluidized bed furnace. The next step is to validate these models. On the one hand, this means to compare these different models and have a look on what model gives the best description of the plant. On the other hand, it should be evaluated, if the models are appropriate to use them for model predictive control. Simulations are done using measurement data and the model outputs are compared to the measured outputs. Moreover, the physical plausibility of the models is validated.

3.7.1 Simulations Using Measurement Data

In this section, the models are validated using *crossvalidation*. This means to take a data set that was not used for parameter estimation and feed the input to the model. Then the output generated by the model is compared to the measured output. This corresponds to calculate a simulation as described in Section 3.5.2. Of course, also a one-step-ahead prediction or a k-step-ahead prediction can be used to validate the model.

Table 3.1: Model performance on validation data.

Output	1st order model	2nd order model	ANN
$\dot{V}_\mathrm{f}$	36.4 %	44.0 %	**45.4 %**
T_f	36.5 %	**46.0 %**	20.0 %
T_b	15.2 %	**32.5 %**	17.0 %
c_O2	47.7 %	51.0 %	**60.2 %**
$\dot{V}_\mathrm{fluid}$	43.9 %	**45.6 %**	35.0 %

There are different metrics to evaluate the goodness of the model output compared to the measured output. A cost function as defined in (3.29) can be taken. However, a more intuitive number that describes the goodness of one model output is to compute the following fit in percent:

$$\mathrm{fit}_i = 100\left(1 - \frac{\|\boldsymbol{\mathcal{Y}}_i - \hat{\boldsymbol{\mathcal{Y}}}_i\|}{\|\hat{\boldsymbol{\mathcal{Y}}}_i - \overline{\hat{\boldsymbol{\mathcal{Y}}}_i}\|}\right), \tag{3.37}$$

where $\|\cdot\|_2$ indicates the Euclidean norm of a vector, $\boldsymbol{\mathcal{Y}}_i$ is vector containing the N measurement points of one output y_i, $\hat{\boldsymbol{\mathcal{Y}}}_i$ is a vector with the corresponding model output and $\overline{\hat{\boldsymbol{\mathcal{Y}}}_i}$ its mean. This fit gets 100 % if the model output fits the measured output perfectly and can be negative as well.

The results of the simulations are shown in Table 3.1 and in the Figures 3.4 and 3.5. The model denoted as *1st-Order Model* is the state-space model in (3.22) that is build up from first-order polynomial models. The model denoted as *2nd-Order Model* is the state-space model in (3.26). *ANN* refers to the artificial neural network with the structure described in Section 3.6.1. The state-space matrices of the identified models can be found in the Appendix.

3.7.2 Physical Interpretation

Another important step of validation is to check the physical plausibility of the model. This can be done by taking a look on the estimated parameters, if they allow a physical interpretation. Here, the model's step responses are analyzed. Figure 3.6 shows the first output y_1 (flue gas, $\dot{V}_\mathrm{f}$). It can be seen that according to the both models, the amount of flue gas is decreasing if the amount of recirculated secondary air increases. Obviously, this is not describing the correct physical behavior in this case. Moreover, according to the 1st-order model, a higher amount of waste fed into the furnace is decreasing the amount of flue gas leaving the furnace, what is not making sense either. Note that the step responses are computed for the normalized models.

In this work, the measurement data used for parameter estimation has been collected in closed-loop. As stated in Section 3.1, this can lead to parameter estimates. In [14] the problems of

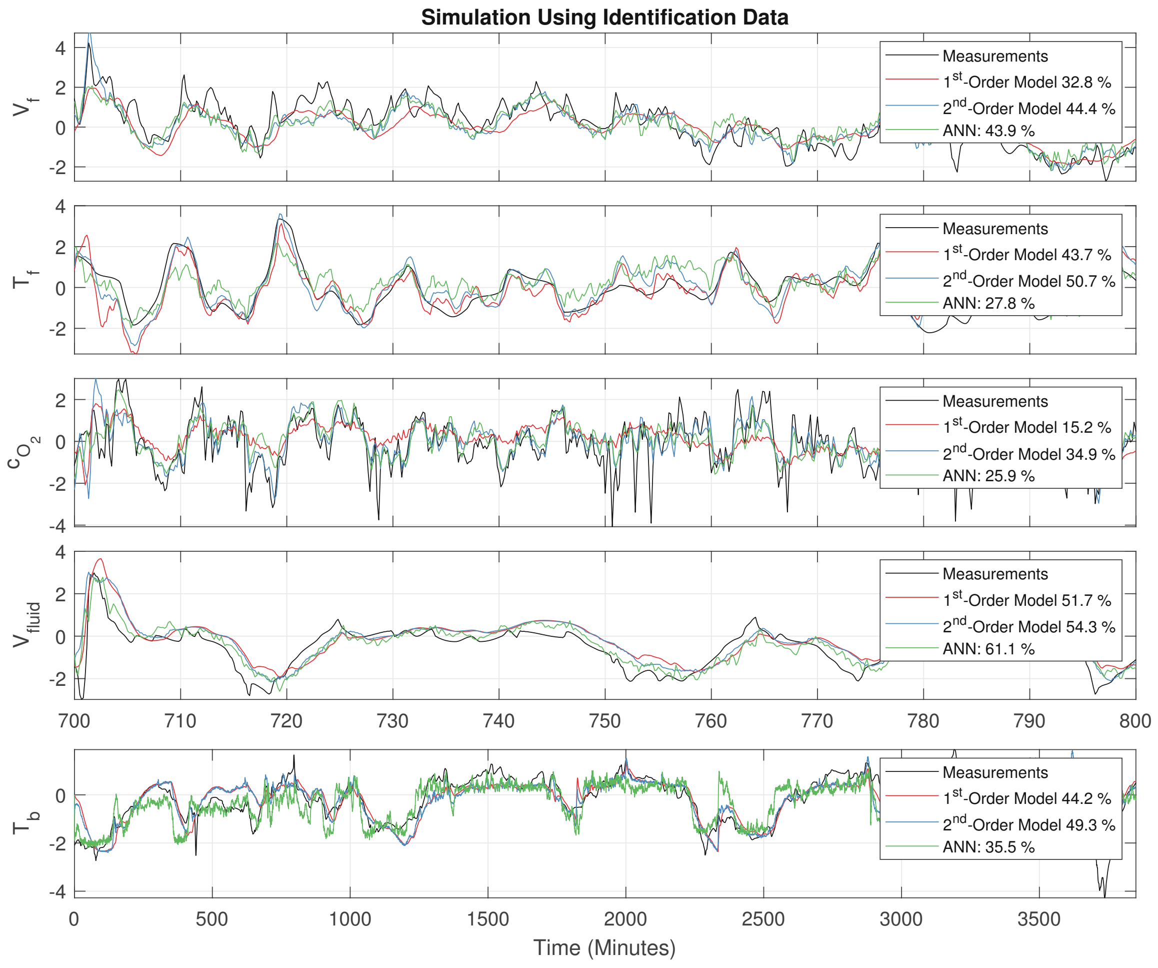


Figure 3.4: The Model outputs are computed using data that have been used for parameter identification (identification data). The Model outputs are compared to measured outputs. The plot of the bed temperature (T_b) shows the whole data set, the other outputs are plotted for a shorter time period. Note that the data is normalized.

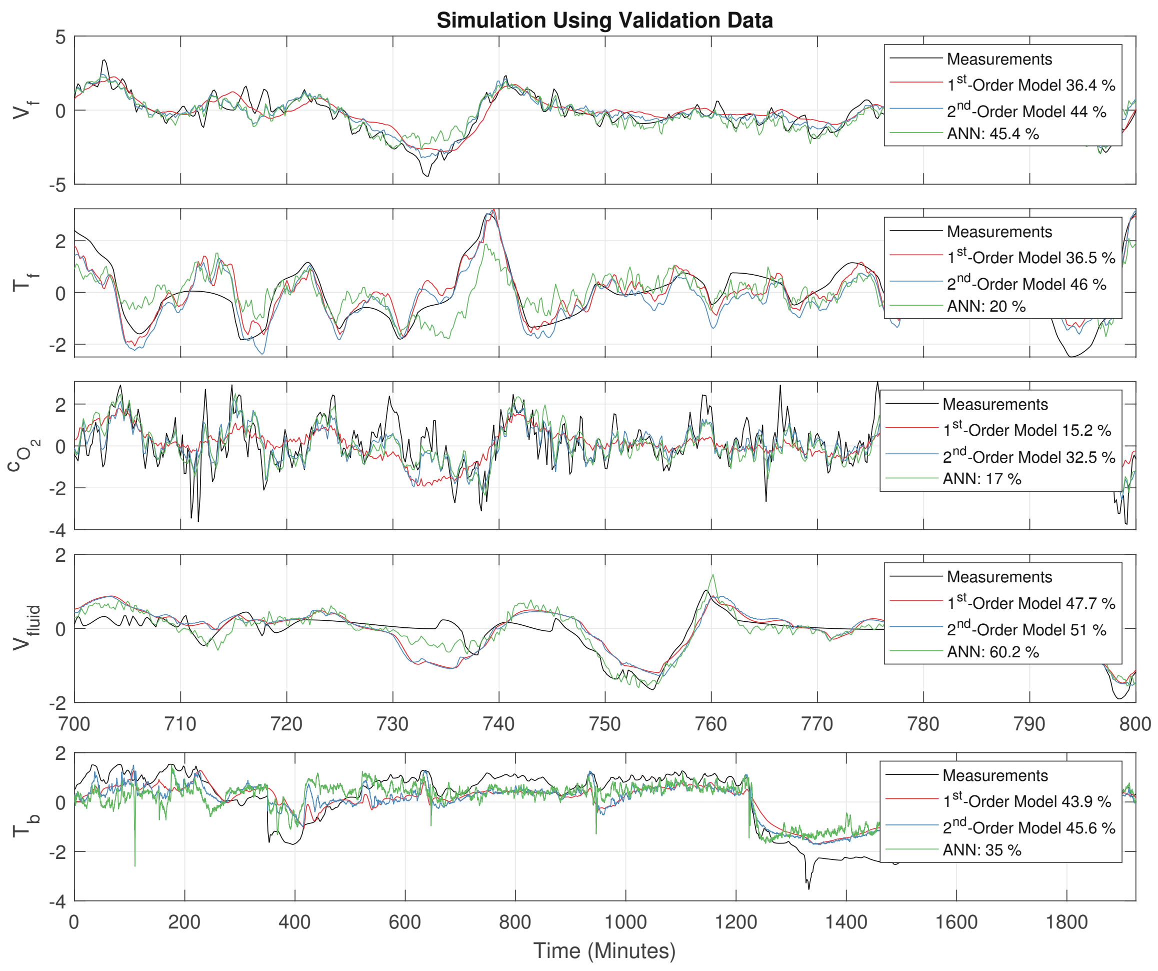


Figure 3.5: Crossvalidation. The Model outputs are computed using data that have not been used for parameter identification (validation data). The Model outputs are compared to measured outputs. The plot of the bed temperature (T_b) shows the whole data set, the other outputs are plotted for a shorter time period. Note that the data is normalized.

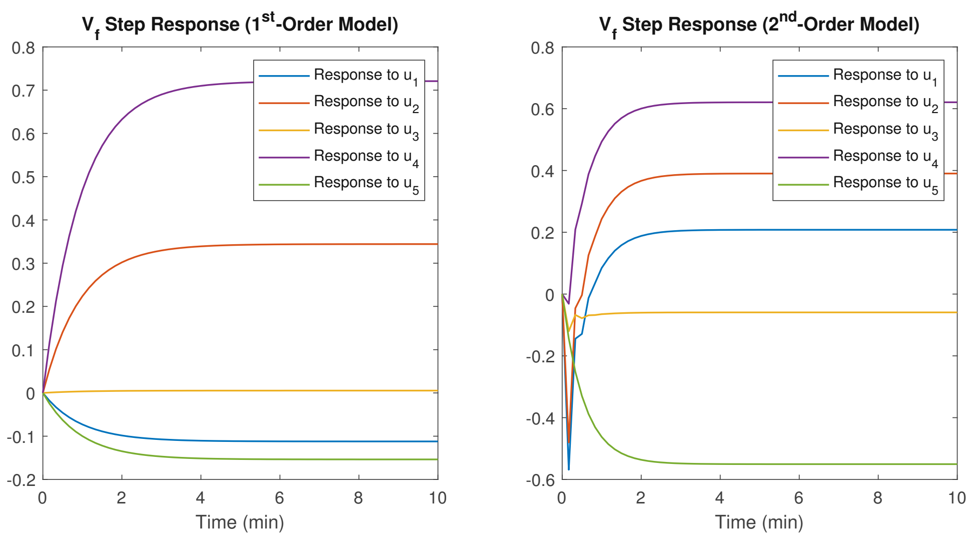


Figure 3.6: The left figure shows the response of the first output ($\dot{V}_\mathrm{f}$) when unit steps in the inputs are fed into the 1st-Order Model. The left figure shows the step response of the 2nd-Order Model.

closed-loop identification and stated and the biases in parameter estimates depending on the model structure and the estimation procedure are presented.

Chapter 4

Model Structure Optimization

In this chapter an algorithm is presented that improves the structure of a state-space model using measurement data. The approach is based on a genetic algorithm.

4.1 Introduction

The state-space models in Chapter 3 have been developed based on physical equations. These equations led to a certain model structure, characterized by the zero and non-zero elements in the state-space matrices $\boldsymbol{A}$ and $\boldsymbol{B}$. This structure can be described using the structure matrices $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$, with zeros for zero elements and ones for elements that are estimated using measurement data. These structure matrices have the following form for the model in (3.22):

$$\boldsymbol{\mathcal{A}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \boldsymbol{\mathcal{B}} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{4.1}$$

The physical equations, that have been used to get a certain model structure, are based on simplifications of the fluidized bed furnace. This brings up the question if another model structure can give a better mathematical description of the process under consideration.

The work flow so far was to determine a model structure, characterized by $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$, and then estimate the parameters using measurement data. The parameter estimation is done using a numerical, iterative algorithm and takes several seconds. This makes it impossible to estimate the parameters for every possible model structure and compare these models, since the amount of possible model structures is extremely high (for the 1st-order model, there are $2^{25} \approx 3.4 \times 10^7$ possible combinations, when taking into account the elements that are allowed to be changed according to Section 4.3. For the 2nd-order model there are even $2^{100} \approx 1.2 \times 10^{30}$ possible combinations). Hence, a genetic algorithm based approach is tried.

This algorithm iteratively changes the structure matrices $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$ in order to find an optimal model structure.

Genetic algorithms have been widely used to solve optimization problems [18][19]. They are biologically-inspired and based on the principle of evolution. At the beginning, an initial generation is created. A generation is a set of solutions which are called individuals. The initial generation can be created randomly or manually. The next step is to create an offspring generation: This is done by combining two individuals from the parental generation (*crossover*) or modifying one individual randomly (*mutation*). Then the *fitness value* is computed for the individuals in the offspring generation. The fitness value is a numerical value evaluating the goodness of one solution. The individuals with the highest fitness are then selected for a new generation. This procedure is repeated for several generations until the fitness of one individual is sufficient high or the number of generations reaches a predefined limit. There are different approaches for the genetic algorithm procedure, e.g. crossover and mutation can be carried out parallel or one after another.

Evolutionary-based algorithms have been used for modelling as well. Genetic programming is an algorithm, that tries out different non-linear model structures by combining different features (e.g. non-linear functions) [20][21].

In this chapter an algorithm for the optimization of the model structure is presented. Afterwards, the results are shown for the fluidized bed furnace model.

4.2 Structure Optimization Using Evolutionary Algorithm

The structure optimization is carried out for the state-space models (3.22) and (3.26). The original structure is taken to create an initial generation with a population size of n_{pop} individuals. This is done by mutating the original structure as described later in Section 4.2.3. Afterwards, the fitness value is calculated for each model according to Section 4.2.1. This procedure is shown in Algorithm 1.

Algorithm 1: Create the first generation of models using an initial model

```
Input: Initial Model $\Sigma_{\text{init}}$
Output: First Generation $G = \{\Sigma_1, \Sigma_2 \dots \Sigma_{\text{n}_{\text{pop}}}\}$
for $i = 1$ to $n_{\text{pop}}$ do
    $\Sigma_i = \text{Mutation}(\Sigma_{\text{init}})$;
    $\Sigma_i = \text{ParameterEstimation}(\Sigma_i)$;
    $f_{\text{i}} = \text{FitnessCalculation}(\Sigma_i)$
end
$G = \{\Sigma_1, \Sigma_2 \dots \Sigma_{\text{n}_{\text{pop}}}\}$
```

This first Generation G is now used to create an offspring generation $\tilde{G}$. This is done by selecting two individuals from G, combining these two individuals (crossover) and modifying the resulting model (mutation). Then, the model parameters are estimated and the fitness value is computed. This is done several times, what leads to an offspring generation $\tilde{G}$. Additionally, the best models n_{surv} from the parental generation are added to the offspring generation. This should prevent good models to get lost. The population size of the offspring generation $\tilde{G}$ is greater than the population size of the parental generation G. The best individuals from the offspring generation are selected to create the next parental generation (survival of the fittest). This procedure is repeated for a predefined number of generations n_{G} and is described in Algorithm 2.

```
Algorithm 2: Create further generations
Input: First Generation $G$
Output: Final Generation $G_{\text{best}}$
for $i = 2$ to $n_{\text{G}}$ do
    for $j = 1$ to $n_{\text{off}}$ do
        $\Sigma_{\text{a}} = \text{SelectIndividual}(G)$;
        $\Sigma_{\text{b}} = \text{SelectIndividual}(G)$;
        $\Sigma_{\text{co}} = \text{Crossover}(\Sigma_{\text{a}},\Sigma_{\text{b}})$;
        $\Sigma_{\text{mut}} = \text{Mutation}(\Sigma_{\text{co}})$;
        $\tilde{\Sigma}_j = \text{ParameterEstimation}(\Sigma_{\text{mut}})$;
        $\tilde{f}_j = \text{FitnessCalculation}(\Sigma_j)$
    end
    $\tilde{G} = \{\tilde{\Sigma}_1, \tilde{\Sigma}_2 \ldots \tilde{\Sigma}_{\text{n}_{\text{off}}}, \Sigma_1, \Sigma_2, \ldots, \Sigma_{\text{n}_{\text{surv}}}\}$;
    $G = \text{SurvivalOfTheFittest}(\tilde{G})$
end
$G_{\text{best}} = G$
```

4.2.1 Fitness

To evaluate the fitness of an individual Σ_j, a fitness function is defined. The fit of each model output $\hat{y}_i$ is computed for a test data set (a data set that has been used neither for parameter estimation nor for crossvalidation) as described in Section 3.7:

$$\text{fit}_i = 100 \left(1 - \frac{\|\boldsymbol{\mathcal{Y}}_i - \hat{\boldsymbol{\mathcal{Y}}}_i\|}{\|\hat{\boldsymbol{\mathcal{Y}}}_i - \overline{\hat{\boldsymbol{\mathcal{Y}}}_i}\|}\right), \tag{4.2}$$

with again the Euclidean norm of a vector $\|\cdot\|_2$, the vector containing the N measurement points of one output $\boldsymbol{\mathcal{Y}}_i$, a vector with the corresponding model output $\hat{\boldsymbol{\mathcal{Y}}}_i$ and its mean $\overline{\hat{\boldsymbol{\mathcal{Y}}}_i}$.

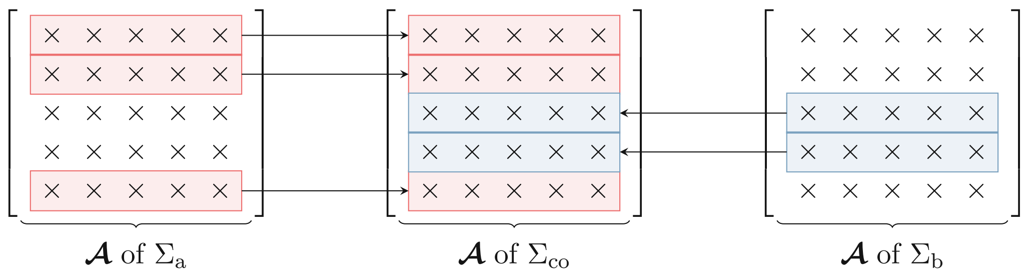

$\mathcal{A}$ of Σ_a
$\mathcal{A}$ of Σ_co
$\mathcal{A}$ of Σ_b

Figure 4.1: Crossover of Σ_a and Σ_b for the $\mathcal{A}$ matrix. Example for Σ_a having higher fits for the outputs y_1, y_2 and y_5 whereas Σ_b has higher fits for y_3 and y_4.

The fitness value f_j of an individual Σ_j is then computed as the average fit plus a term that penalizes a high number of parameters in the model:

$$f_j = \frac{1}{r}\sum_{i=1}^{r} \mathrm{fit}_i - \lambda(n_{\mathrm{p},j} - n_{\mathrm{p,init}}), \tag{4.3}$$

where r is the number of outputs, $n_{\mathrm{p},j}$ is the number of parameters in the model Σ_j, $n_{\mathrm{p,init}}$ is the number of parameters in the initial model Σ_init and λ is a tuning parameter.

4.2.2 Crossover

Crossover means the combination of two models Σ_a and Σ_b by combining their structure matrices $\mathcal{A}$ and $\mathcal{B}$.

A crossover is usually carried out by randomly combining two solutions. In this work, two models are combined according to their ability to describe one output y_i. For each output, a fit is computed according to (3.37) using the test data set. If one model performs better in describing one output y_i, the part of this model responsible for this output is taken to create the new model Σ_co. For example, if Σ_a has a higher fit for the first output, the first lines of $\mathcal{A}$ and $\mathcal{B}$ are taken from Σ_a to create Σ_co.

Figure 4.1 illustrates the crossover procedure for the $\mathcal{A}$ matrix. In this example, the fits of Σ_a are higher for the outputs y_1, y_2 and y_5. The same thing is done for the matrix $\mathcal{B}$. For the second-order model (3.26), the corresponding augmented states need to be selected as well.

Selection

For the crossover operation, two individuals need to be selected. There are different methods how individuals can be selected from a population, such as *tournament selection*, *proportional*

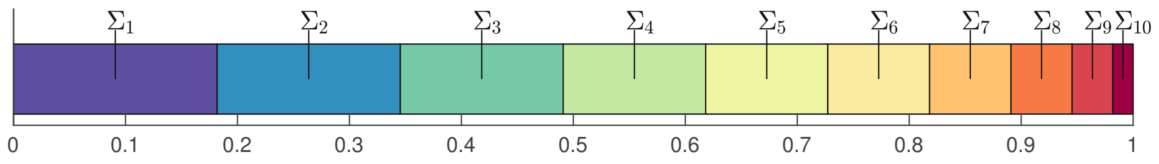


Figure 4.2: Models with a higher fitness value have a higher probability to be selected for the crossover operation. Here, Σ_1 has the highest chance to be selected.

roulette wheel selection or *rank-based roulette wheel selection*. In [22], these three selection methods are compared for solving an optimization problem. All methods have in common that they use randomness for individual selection.

In tournament selection, some (commonly two) individuals are randomly selected from the parental generation. The one with the higher fitness value is chosen.

If proportional roulette wheel selection is applied, the probability for an individual to be selected is proportional to its fitness value.

In this work, rank-based roulette wheel selection is applied. The individuals within one generation are sorted according to their fitness values. The higher they are ranked, the higher their probability to be selected. The probability for one individual to be selected is specified as

$$P_i = \frac{\mathrm{pos}_i}{\sum_{j=1}^{n_\mathrm{pop}} \mathrm{pos}_j}, \tag{4.4}$$

whereby the individual with the highest fitness value has $\mathrm{pos} = 1$ and the individual with the lowest fitness value has $\mathrm{pos} = n_\mathrm{pop}$. Figure 4.2 illustrates the selection method: Every individual has its segment on an interval between 0 and 1. The higher a model is ranked, the larger its field on this interval. A uniformly distributed random number between 0 and 1 is created. An individual is selected, if the random number is lying in its segment. One can imagine this as a roulette wheel, where the fields of the numbers are differently-sized, so that the chance for the ball to fall on a field is different for every number.

4.2.3 Mutation

The mutation of the model structure is about changing the structure of the state-space matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ randomly. Mutation is now done by changing elements in $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$ from one to zero or from zero two one. However, not all elements are allowed to be changed.

The changes in the structure matrices are carried out by changing the elements in a random way: Each zero-element has a probability of p_+ to be flipped to one and each one has a

Table 4.1: Hyperparameter in the model optimization algorithm.

Parameter	Description	Value
n_{G}	Number of generations	50
n_{pop}	Number of individuals in a parental generation	15
n_{off}	Number of individuals in an offspring generation	25
n_{surv}	Number of best models survive additionally to the offspring generation	3
$p_{+,0}$	Probability for a parameter be released (starting value)	0.5
$p_{-,0}$	Probability for a parameter to be locked and set to 0 (starting value)	0.5
λ_{p}	Probability decreasing rate	0.95
λ	Penalty weight for number of parameters in the model	0.2

probability of p_- to be flipped to zero. The values of p_+ and p_- are hyperparameters of the structure optimization algorithm.

The search process can be improved, if the values for p_+ and p_- are higher in the first generations and decrease with the number of generations. Therefore, p_+ is defined as

$$p_+ = p_{+,0}\lambda_{\mathrm{p}}^i, \tag{4.5}$$

where $p_{+,0}$ is a starting value, λ_{p} is the decreasing rate and i is the i-th generation. The same applies for p_-.

If elements in $\boldsymbol{\mathcal{A}}$ are flipped from zero to one, the corresponding value in the matrix $\boldsymbol{A}$ is set to zero as well.

4.3 Results

The algorithm is now applied to the state space models (3.22) and (3.26) and the results are presented. There are some hyperparameters in the algorithm that can be tuned. The values of these parameters influence the success of the genetic algorithm. In [18], techniques are presented to find and tune these parameters. Table 4.1 shows how the parameters appearing in the algorithm have been chosen in this work.

The $\boldsymbol{A}$ and the $\boldsymbol{B}$ matrix of the first-order model have the following structure after applying the genetic algorithm:

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & \boxed{a_{12}} & 0 & \boxed{a_{13}} & 0 \\ a_{21} & a_{22} & a_{23} & \boxed{a_{24}} & 0 \\ 0 & 0 & a_{33} & \boxed{a_{34}} & \boxed{a_{35}} \\ a_{41} & \boxed{a_{42}} & \boxed{a_{43}} & a_{44} & 0 \\ 0 & 0 & 0 & 0 & a_{55} \end{bmatrix}, \qquad \boldsymbol{B} = \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} & b_{15} & 0 & 0 & 0 & 0 \\ b_{21} & b_{22} & b_{23} & 0 & 0 & 0 & 0 & b_{28} & b_{29} \\ b_{31} & b_{32} & b_{33} & 0 & 0 & b_{36} & b_{37} & 0 & 0 \\ b_{41} & b_{42} & b_{43} & b_{44} & b_{45} & 0 & 0 & 0 & \boxed{b_{49}} \\ 0 & b_{52} & b_{53} & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

The elements with the green background are elements that are allowed to be changed by the algorithm. Elements, that are framed in green are elements, that have been changed. These elements have been zeros in the original structure. For y_5 ($\dot{m}_{\text{fluid}}$) the algorithm is not allowed to make any changes, since it is physically not possible that this output depends on other outputs or inputs except of $\dot{m}_{\text{p}}$ and $\dot{m}_{\text{re,p}}$. Moreover, y_3 (T_{b}) cannot depend on any secondary gas flows. Thus, these parameters are not allowed to be changed either.

For the second-order model, the algorithm is allowed to set parameters to zero. This is done because it may be beneficial, if a future output depends only on a current input, not on the previous one. The resulting state-space matrices of the second-order model have the following form:

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & 0 & 0 & \boxed{a_{14}} & 0 & 1 & 0 & 0 & 0 & 0 \\ \boxed{0} & a_{22} & \boxed{0} & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & \boxed{a_{32}} & a_{33} & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ \boxed{0} & \boxed{a_{42}} & 0 & a_{44} & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & a_{55} & 0 & 0 & 0 & 0 & 1 \\ a_{61} & 0 & 0 & \boxed{a_{64}} & 0 & 0 & 0 & 0 & 0 & 0 \\ a_{71} & \boxed{0} & \boxed{0} & \boxed{a_{74}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \boxed{a_{82}} & \boxed{0} & 0 & \boxed{a_{85}} & 0 & 0 & 0 & 0 & 0 \\ a_{91} & 0 & 0 & a_{94} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & a_{10,5} & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$\boldsymbol{B} = \begin{bmatrix} b_{11} & b_{12} & \boxed{0} & b_{14} & \boxed{0} & \boxed{b_{16}} & 0 & 0 & \boxed{b_{19}} \\ b_{21} & \boxed{0} & \boxed{0} & \boxed{b_{24}} & 0 & 0 & 0 & b_{28} & b_{29} \\ b_{31} & \boxed{0} & b_{33} & 0 & 0 & b_{36} & \boxed{0} & 0 & 0 \\ b_{41} & \boxed{0} & \boxed{0} & \boxed{0} & \boxed{0} & \boxed{b_{46}} & 0 & 0 & \boxed{b_{49}} \\ 0 & b_{52} & b_{53} & 0 & 0 & 0 & 0 & 0 & 0 \\ b_{61} & \boxed{0} & \boxed{0} & b_{64} & b_{65} & 0 & 0 & 0 & 0 \\ b_{71} & \boxed{0} & \boxed{0} & 0 & 0 & 0 & 0 & \boxed{0} & b_{79} \\ b_{81} & \boxed{0} & \boxed{0} & 0 & 0 & b_{86} & \boxed{0} & 0 & 0 \\ b_{91} & \boxed{0} & \boxed{0} & \boxed{0} & \boxed{0} & \boxed{b_{96}} & 0 & 0 & \boxed{b_{99}} \\ 0 & b_{10,2} & b_{10,3} & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Again elements with a green background are elements that are allowed to be changed and green-framed elements are elements that have been unlocked by the algorithm. Now, there are also elements that have been set to zero. These elements are framed in red.

The results of the crossvalidation for the first-order model are shown in Figure 4.3, the results of the crossvalidation for the second-order model are shown in Figure 4.4. For the first-order model, significant improvements could be achieved for the outputs y_1, y_2 and y_5. The numbers of parameters in the model changed from 30 to 38. For the second order model, the performance increased for y_1 and y_5, for the three other outputs it stayed constant or decreased by some percent points. However, the number of parameters decreased from 60 to 46.

The changes in the fitness value over the generations are shown in Figure 4.5. It can be seen that for the first-order model, the solution converges within a few iteration almost to the final solution. For the second-order model it takes more generations. This is because there are far more possibilities for changes in the model structure for the second-order model.

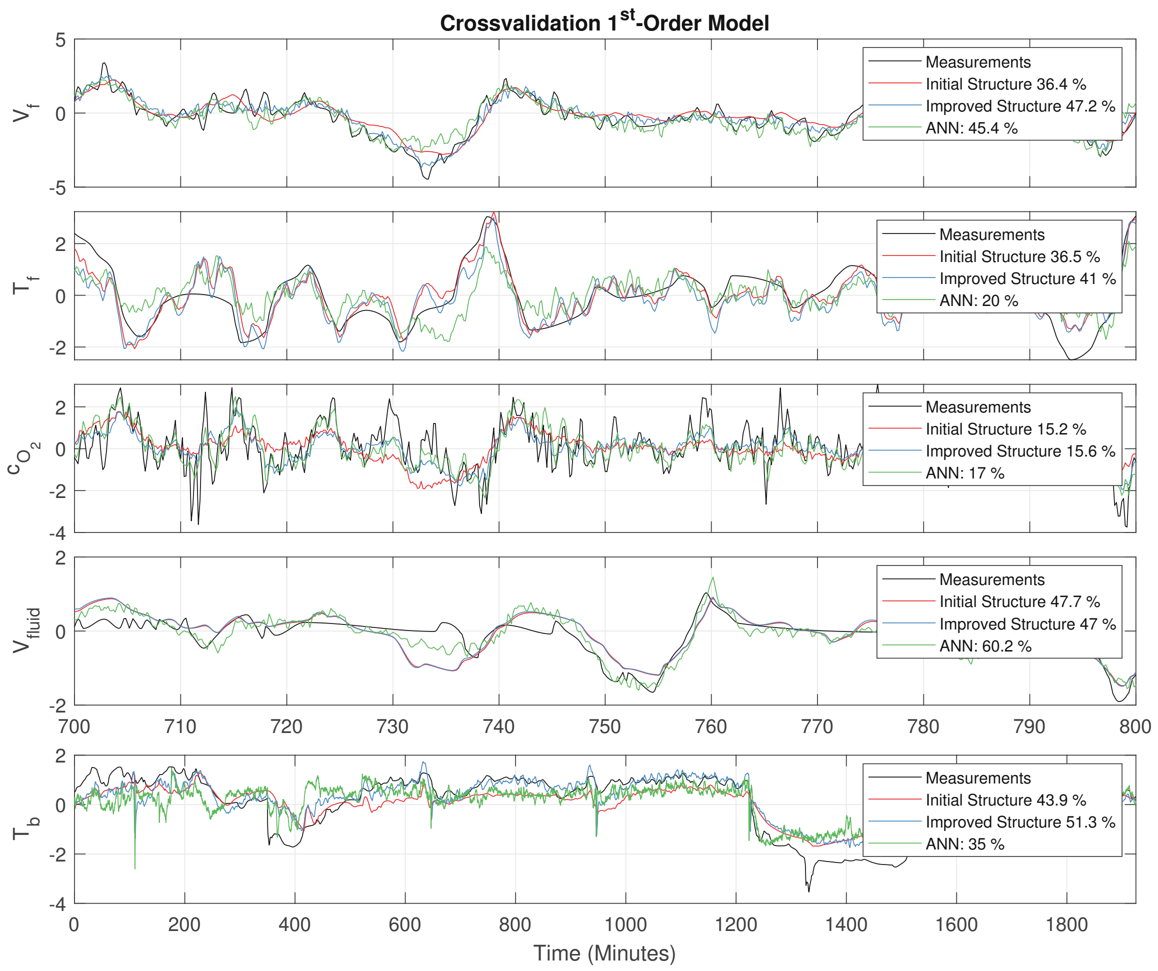


Figure 4.3: Crossvalidation for the original model and the model with the improved structure. Again with the ANN for comparison.

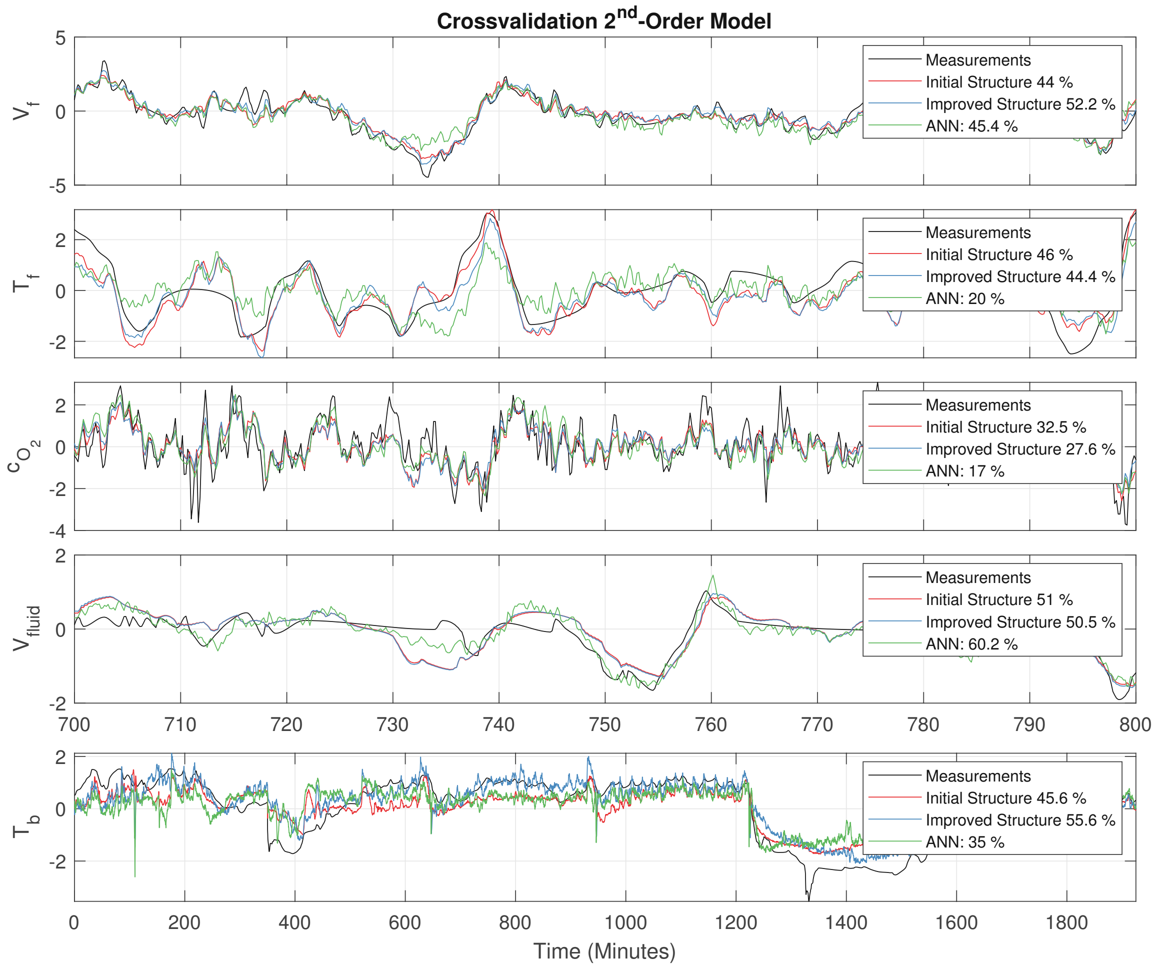


Figure 4.4: Crossvalidation for the original model and the model with the improved structure. Again with the ANN for comparison.

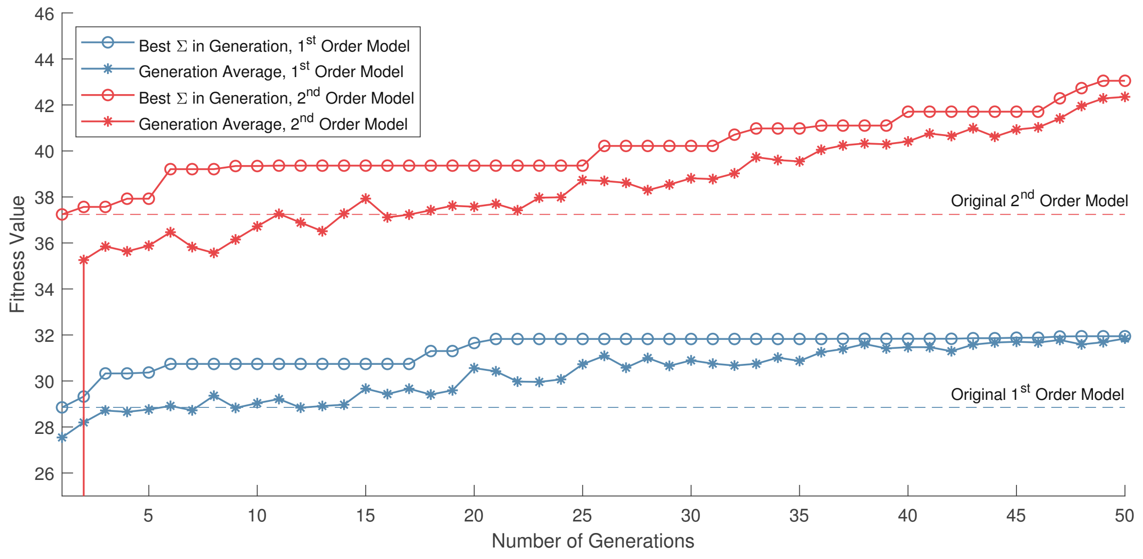


Figure 4.5: Change of the fitness value over generations.

Chapter 5

Model Predictive Control

In this chapter, a model predictive control (MPC) algorithm is proposed to control the system described in Chapter 2. The MPC algorithm uses the model which was identified in Chapter 3. To achieve offset-free control, this model is augmented with disturbance states. An observer is designed to estimate both the original model's states and the augmented disturbance states. In the last part, the controller's performance is validated by simulation studies.

5.1 Introduction

The idea of model predictive control is to derive the optimal control input to the plant for every time step (e.g. every second, the sampling time of how often this optimization problem needs to be solved is determined by the time constants of the process under consideration). In order to implement a model predictive control scheme, the requirements are as follows: Firstly, a model of the process is necessary that contains the information how the inputs to the system influence the outputs to be controlled. Secondly, in order to make clear what is optimal, a cost function needs to be defined. This cost function can penalize e.g. a difference between the system's output to its reference in the future. Thirdly, constraints are formulated, for example for control inputs due to physical limitations. These constraints are later taken into account when solving the optimization problem.

As mentioned above, the cost function evaluates the system's behavior in the future. However, this is done for a finite period of time. The length of this time period is called the prediction horizon N_p and chosen appropriately. A model is used to compute the system's behavior, depending on the control input, within this prediction horizon. The control input within the prediction horizon is chosen in a way that the costs are getting minimized. The control input needs to satisfy the constraints both in absolute value and in rate of change. Although the control input can be chosen different for every time step within the prediction horizon, it is normally held constant after an amount of time shorter than the prediction horizon. This shorter time span is called the control horizon N_c. Setting the control horizon shorter than the prediction horizon decreases the size of the optimization problem and therefore requires

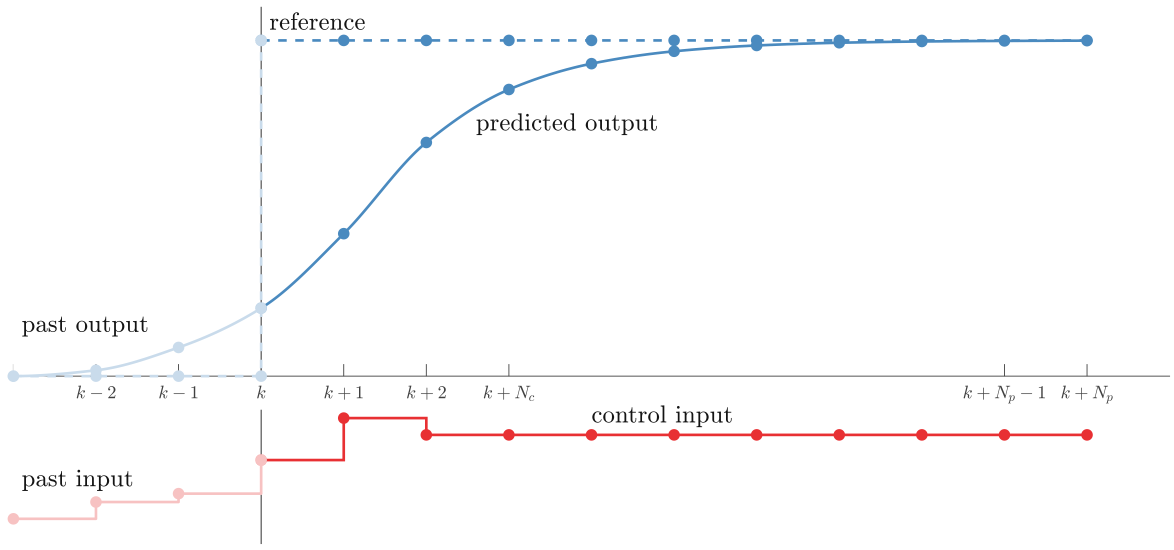


Figure 5.1: Principle of model predictive control. The future control input is computed to get a optimal system behavior.

less computational effort. Figure 5.1 illustrates the principle of model predictive control with the reference, the output and the input within the prediction horizon.

Since the optimization problem is solved every time step, a new sequence of control inputs for the whole prediction horizon is computed every time. Only the first control input is implemented to the plant. At the next time step, a new sequence of control inputs is computed, and again, only the first one is implemented to the plant. This procedure is known as receding horizon control, because the time frame, for which the optimal control input is computed, is shifted with every time step.

A deeper insight into the design and implementation of model predictive controllers is given in [23][24].

5.2 Process Description for Predictive Control

This section is about the model that is used for MPC and the constraints implemented in the MPC algorithm.

5.2.1 Linear State-Space Model

In Chapter 3, a linear discrete-time state-space model of the following structure was identified that will now be used for model predictive control:

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k), \\ \boldsymbol{y}(k) &= \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k), \end{aligned} \tag{5.1}$$

with the variables

$\boldsymbol{A} \in \mathbb{R}^{n \times n}$	system matrix,	$\boldsymbol{x} \in \mathbb{R}^{n}$	state vector,
$\boldsymbol{B} \in \mathbb{R}^{n \times r}$	input matrix,	$\boldsymbol{u} \in \mathbb{R}^{r}$	input vector,
$\boldsymbol{C} \in \mathbb{R}^{m \times n}$	output matrix,	$\boldsymbol{y} \in \mathbb{R}^{m}$	output vector,
$\boldsymbol{D} \in \mathbb{R}^{m \times r}$	feedthrough matrix		

and the dimensions

$n = 5$	state space dimension,
$r = 9$	input space dimension,
$m = 5$	output space dimension.

Model denormalization

The parameters in the state-space matrices have been estimated using measurement data. These measurement data have been normalized before the parameter estimation (see Section 3.2.1). To get the relationship between the real inputs $\boldsymbol{u}$ and the real outputs $\boldsymbol{y}$, the model is now denormalized.

The measurement data have been normalized by division through its standard deviations:

$$\begin{aligned} \boldsymbol{u}' &= \boldsymbol{\Sigma_u}^{-1}\boldsymbol{u}, \\ \boldsymbol{y}' &= \boldsymbol{\Sigma_y}^{-1}\boldsymbol{y}, \end{aligned} \tag{5.2}$$

where $\boldsymbol{u}'$ and $\boldsymbol{y}'$ are the normalized input and output vector, respectively, and $\boldsymbol{\Sigma_u}$ and $\boldsymbol{\Sigma_y}$ are the standard deviation matrices containing the standard deviations of $\boldsymbol{u}$ and $\boldsymbol{y}$ that had been used to normalize the measurement data in its diagonals.

Using these vectors for parameter estimation leaded to a normalized state-space system:

$$\begin{aligned} \boldsymbol{x}'(k+1) &= \boldsymbol{A}'\boldsymbol{x}'(k) + \boldsymbol{B}'\boldsymbol{u}'(k), \\ \boldsymbol{y}'(k) &= \boldsymbol{C}\boldsymbol{x}'(k). \end{aligned} \tag{5.3}$$

For the case that $\boldsymbol{C} = \boldsymbol{I}$, (5.2) can be substituted into (5.3):

$$\begin{aligned}\boldsymbol{\Sigma_y}^{-1}\boldsymbol{x}(k+1) &= \boldsymbol{A'}\boldsymbol{\Sigma_y}^{-1}\boldsymbol{x}(k) + \boldsymbol{B'}\boldsymbol{\Sigma_u}^{-1}\boldsymbol{u}(k),\\ \boldsymbol{\Sigma_y}^{-1}\boldsymbol{y}(k) &= \boldsymbol{C}\boldsymbol{\Sigma_y}^{-1}\boldsymbol{x}(k).\end{aligned} \tag{5.4}$$

Multiplying $\boldsymbol{\Sigma_y}$ from the left leads to a system with the real inputs and outputs:

$$\begin{aligned}\boldsymbol{x}(k+1) &= \boldsymbol{\Sigma_y}\boldsymbol{A'}\boldsymbol{\Sigma_y}^{-1}\boldsymbol{x}(k) + \boldsymbol{\Sigma_y}\boldsymbol{B'}\boldsymbol{\Sigma_u}^{-1}\boldsymbol{u}(k),\\ \boldsymbol{y}(k) &= \boldsymbol{C}\boldsymbol{x}(k).\end{aligned} \tag{5.5}$$

This results in the relationship between the the normalized model matrices and the real input-output model matrices:

$$\begin{aligned}\boldsymbol{A} &= \boldsymbol{\Sigma_y}\boldsymbol{A'}\boldsymbol{\Sigma_y}^{-1},\\ \boldsymbol{B} &= \boldsymbol{\Sigma_y}\boldsymbol{B'}\boldsymbol{\Sigma_u}^{-1},\end{aligned} \tag{5.6}$$

with the system matrix of the normalized model $\boldsymbol{A'}$, the input matrix of the normalized model $\boldsymbol{B'}$ and the standard deviation matrices $\boldsymbol{\Sigma_u}$ and $\boldsymbol{\Sigma_y}$, containing the standard deviations of $\boldsymbol{u}$ and $\boldsymbol{y}$ that had been used to normalize the measurement data in its diagonals.

Input vector reduction

The input vector $\boldsymbol{u}$ in (5.1) was defined as

$$\boldsymbol{u} = \left[n_{\text{msw}}, \quad \dot{V}_{\text{p}}, \quad \dot{V}_{\text{p,re}}, \quad \dot{V}_{\text{s}}, \quad \dot{V}_{\text{s,re}}, \quad \dot{V}_{\text{p}}T_{\text{fluid}} \quad \dot{V}_{\text{p,re}}T_{\text{fluid}}, \quad \dot{V}_{\text{s}}T_{\text{s}}, \quad \dot{V}_{\text{s,re}}T_{\text{s,re}}, \quad \right]^T.$$

A description of the inputs is given in Section 3.3. The input vector consists of the waste input, volume flow rates and enthalpy flow rates. This is because the identification procedure led to better results with an input vector containing both volume flow rates and enthalpy flow rates. However, it can be seen that not all of the 9 inputs can be chosen by the MPC independently. The inputs 6-9 are product of the inputs 2-5 with the corresponding temperatures, so the volume flow rates in the inputs 2-5 must be the same as in the inputs 6-9, obviously. The temperatures in the inputs 6-9 cannot be influenced at all and change slowly over time. Therefore, they are assumed to be constant within the prediction horizon.

This allows a reduction of the input vector from 9 to 5 inputs:

$$\begin{aligned}
\boldsymbol{B}\boldsymbol{u} &= \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} & b_{15} & b_{16} & b_{17} & b_{18} & b_{19} \\ b_{21} & b_{22} & b_{23} & b_{24} & b_{25} & b_{26} & b_{27} & b_{28} & b_{29} \\ b_{31} & b_{32} & b_{33} & b_{34} & b_{35} & b_{36} & b_{37} & b_{38} & b_{39} \\ b_{41} & b_{42} & b_{43} & b_{44} & b_{45} & b_{46} & b_{47} & b_{48} & b_{49} \\ b_{51} & b_{52} & b_{53} & b_{54} & b_{55} & b_{56} & b_{57} & b_{58} & b_{59} \end{bmatrix} \begin{bmatrix} n_{\text{msw}} \\ \dot{V}_{\text{p}} \\ \dot{V}_{\text{p,re}} \\ \dot{V}_{\text{s}} \\ \dot{V}_{\text{s,re}} \\ \dot{V}_{\text{p}} T_{\text{fluid}} \\ \dot{V}_{\text{p,re}} T_{\text{fluid}} \\ \dot{V}_{\text{s}} T_{\text{s}} \\ \dot{V}_{\text{s,re}} T_{\text{s,re}} \end{bmatrix} \\
&= \begin{bmatrix} b_{11} & b_{12} + T_{\text{fluid}} b_{16} & b_{13} + T_{\text{fluid}} b_{17} & b_{14} + T_{\text{s}} b_{18} & b_{15} + T_{\text{s,re}} b_{19} \\ b_{21} & b_{22} + T_{\text{fluid}} b_{26} & b_{23} + T_{\text{fluid}} b_{27} & b_{24} + T_{\text{s}} b_{28} & b_{25} + T_{\text{s,re}} b_{29} \\ b_{31} & b_{32} + T_{\text{fluid}} b_{36} & b_{33} + T_{\text{fluid}} b_{37} & b_{34} + T_{\text{s}} b_{38} & b_{35} + T_{\text{s,re}} b_{39} \\ b_{41} & b_{42} + T_{\text{fluid}} b_{46} & b_{43} + T_{\text{fluid}} b_{47} & b_{44} + T_{\text{s}} b_{48} & b_{45} + T_{\text{s,re}} b_{49} \\ b_{51} & b_{52} + T_{\text{fluid}} b_{56} & b_{53} + T_{\text{fluid}} b_{57} & b_{54} + T_{\text{s}} b_{58} & b_{55} + T_{\text{s,re}} b_{59} \end{bmatrix} \begin{bmatrix} n_{\text{msw}} \\ \dot{V}_{\text{p}} \\ \dot{V}_{\text{p,re}} \\ \dot{V}_{\text{s}} \\ \dot{V}_{\text{s,re}} \end{bmatrix} \\
&= \tilde{\boldsymbol{B}}\tilde{\boldsymbol{u}}.
\end{aligned} \tag{5.7}$$

The matrix $\tilde{\boldsymbol{B}}$ now contains the temperatures that are varying over time and is therefore a function of k. As a result, the matrix $\tilde{\boldsymbol{B}}$ needs to be updated every time step with the latest temperature measurements.

Note that the input reduction was now presented for a general matrix $\boldsymbol{B}$. In the model identified in Chapter 3, some of the entries of the $\boldsymbol{B}$-matrix are zero.

In the following sections, the model with the reduced input vector will be used for model predictive control, even though the input vector will be described with $\boldsymbol{u}(k)$ instead of $\tilde{\boldsymbol{u}}(k)$.

5.2.2 Constraints

As mentioned in the previous sections, the model predictive controller needs to find an input sequence in order to optimize the plant's behavior within the prediction horizon. However, these inputs cannot be arbitrary high, since the actors cannot accomplish any inputs. Therefore, constraints are formulated. These constraints are taken into account when solving the optimization problem later. Constraints are formulated both for the absolute values and for the rate of change. Furthermore, constraints are formulated for the output vector, since there are restrictions how high or low these variables are allowed to be. Output constraints are necessary in order to not violate emission regulations or to cause damages to the plant due to too high temperatures.

Constraints on the rate of change are received from measurement data by computing a central difference quotient and taking the maximum value of an observation period. This leads to the maximum value for $\Delta \boldsymbol{u}$.

5.3 Offset-Free MPC

In general, model predictive control does not lead to offset-free control. This means, there can be an offset between the outputs and its reference values for $t \to \infty$. In [25], a strategy for linear model predictive control is presented that leads to offset-free control. A method for offset-free control applied for non-linear MPC can be found in [26]. In this section, it is shown, how this offset-free control strategy is implemented based on [25]. Note that this control strategy can be used for constant reference values only.

Consider the following discrete linear state-space system:

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) + \boldsymbol{w}(k), \\ \boldsymbol{y}(k) &= \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{v}(k), \end{aligned} \tag{5.8}$$

with the process noise $\boldsymbol{w}(k) \in \mathbb{R}^n$ and the output noise $\boldsymbol{v}(k) \in \mathbb{R}^m$.

In order to receive offset-free control, the model needs to be augmented with disturbance states. In general, the number of disturbance states n_d added to the model is equal to the number of outputs desired to be offset-free. Here, the model is augmented with m states, so all outputs are going to be offset-free.

This leads to the augmented model

$$\begin{aligned} \begin{bmatrix} \boldsymbol{x}(k+1) \\ \boldsymbol{d}(k+1) \end{bmatrix} &= \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B_d} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}(k) \\ \boldsymbol{d}(k) \end{bmatrix} + \begin{bmatrix} \boldsymbol{B} \\ \boldsymbol{0} \end{bmatrix} \boldsymbol{u}(k) + \tilde{\boldsymbol{w}}(k), \\ \boldsymbol{y}(k) &= \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{C_d}\boldsymbol{d}(k) + \boldsymbol{v}(k), \end{aligned} \tag{5.9}$$

with the disturbance states $\boldsymbol{d}(k) \in \mathbb{R}^{n_d}$, the process noise acting on the augmented system $\tilde{\boldsymbol{w}}(k) \in \mathbb{R}^{n+n_d}$ and the output noise $\boldsymbol{v}(k) \in \mathbb{R}^m$. These equations include now the new matrices $\boldsymbol{B_d}$ and $\boldsymbol{C_d}$. These matrices need to be chosen in a way that the augmented model is observable. This is the case, if

$$\begin{bmatrix} \boldsymbol{A} - \boldsymbol{I} & \boldsymbol{B_d} \\ \boldsymbol{C} & \boldsymbol{C_d} \end{bmatrix} \text{ has full rank.} \tag{5.10}$$

A proof can be found in [25]. Now $\boldsymbol{C_d}$ can be chosen to be the identity matrix $\boldsymbol{I}$. As mentioned above, in this case $n_d = m$. Therefore, the matrix in condition (5.10) is a square matrix and the condition leads to

$$\det \begin{bmatrix} \boldsymbol{A}-\boldsymbol{I} & \boldsymbol{B_d} \\ \boldsymbol{C} & \boldsymbol{C_d} \end{bmatrix} = \det(\boldsymbol{A}-\boldsymbol{I}-\boldsymbol{B_d}\boldsymbol{C}) \neq 0. \tag{5.11}$$

If the plant has no integrator poles, what is the case in this application, $\det(\boldsymbol{A}-\boldsymbol{I}) \neq 0$. Therefore, $\boldsymbol{B_d}$ can be chosen to be a zero matrix ($\boldsymbol{B_d} = \boldsymbol{0}$) and the condition (5.11) still holds.

For the usage of the augmented model, the original and the augmented states need to be estimated. To do so, an observer will be designed in the next section.

5.3.1 Observer Design

The observer for the augmented model (5.9) is now designed to estimate both the original states and the augmented disturbance states:

$$\begin{bmatrix} \hat{\boldsymbol{x}}(k+1) \\ \hat{\boldsymbol{d}}(k+1) \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B_d} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}}(k) \\ \hat{\boldsymbol{d}}(k) \end{bmatrix} + \begin{bmatrix} \boldsymbol{B} \\ \boldsymbol{0} \end{bmatrix} \boldsymbol{u}(k) + \begin{bmatrix} \boldsymbol{L_x} \\ \boldsymbol{L_d} \end{bmatrix} (-\boldsymbol{y_{\mathrm{m}}}(k) + \boldsymbol{C}\hat{\boldsymbol{x}}(k) + \boldsymbol{C_d}\hat{\boldsymbol{d}}(k)), \tag{5.12}$$

where $\boldsymbol{y_{\mathrm{m}}}(k)$ is the measured output vector at time step k and $\boldsymbol{B_d} = \boldsymbol{0}$ and $\boldsymbol{C_d} = \boldsymbol{I}$ as mentioned above. $\boldsymbol{L_x}$ and $\boldsymbol{L_d}$ need to be chosen so that the observer is stable.

In this work, $\boldsymbol{L_x}$ and $\boldsymbol{L_d}$ are found by designing the observer as a Kalman filter for the augmented model. This is done in [27] for offset-free MPC as well. Let's therefore introduce the augmented matrices

$$\tilde{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B_d} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}, \; \tilde{\boldsymbol{C}} = \begin{bmatrix} \boldsymbol{C} & \boldsymbol{C_d} \end{bmatrix}, \; \tilde{\boldsymbol{L}} = \begin{bmatrix} \boldsymbol{L_x} \\ \boldsymbol{L_d} \end{bmatrix}.$$

The observer gain $\tilde{\boldsymbol{L}}$ is then

$$\tilde{\boldsymbol{L}} = \tilde{\boldsymbol{A}}\boldsymbol{\Sigma}\tilde{\boldsymbol{C}}^T(\tilde{\boldsymbol{C}}\boldsymbol{\Sigma}\tilde{\boldsymbol{C}}^T + \boldsymbol{R}_v)^{-1}, \tag{5.13}$$

where $\boldsymbol{R}_v \in \mathbb{R}^{m\times m}$ is the covariance matrix of the output noise $\boldsymbol{v}(k)$ and the matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{(n+n_d)\times(n+n_d)}$ is the solution to the discrete algebraic Riccati equation

$$\boldsymbol{\Sigma} = \tilde{\boldsymbol{A}}\boldsymbol{\Sigma}\tilde{\boldsymbol{A}}^T + \boldsymbol{Q}_w - \tilde{\boldsymbol{A}}\boldsymbol{\Sigma}\tilde{\boldsymbol{C}}^T(\tilde{\boldsymbol{C}}\boldsymbol{\Sigma}\tilde{\boldsymbol{C}}^T + \boldsymbol{R}_v)^{-1}\tilde{\boldsymbol{C}}\boldsymbol{\Sigma}\tilde{\boldsymbol{A}}^T. \tag{5.14}$$

The matrix $\boldsymbol{Q}_w \in \mathbb{R}^{(n+n_d)\times(n+n_d)}$ is the covariance matrix of the process noise $\tilde{\boldsymbol{w}}$ acting on the augmented system. The discrete algebraic Riccati equation is solved in MATLAB with the command `idare`.

For this application, the covariance matrix $\boldsymbol{R}_v$ is chosen to be the identity matrix $\boldsymbol{I}$ and $\boldsymbol{Q}_w$ is chosen to be $10^3 \cdot \boldsymbol{I}$.

5.3.2 Cost Function

For the offset-free MPC, the cost function that is going to be minimized at each time step k is specified as

$$J = \sum_{i=0}^{N_\mathrm{p}} (\boldsymbol{x}(i) - \bar{\boldsymbol{x}}_t)^T \boldsymbol{Q} (\boldsymbol{x}(i) - \bar{\boldsymbol{x}}_t) + \sum_{j=0}^{N_\mathrm{c}} (\boldsymbol{u}(j) - \bar{\boldsymbol{u}}_t)^T \boldsymbol{R} (\boldsymbol{u}(j) - \bar{\boldsymbol{u}}_t), \tag{5.15}$$

with the matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$, weighting deviations from the target values of the state vector and the input vector, respectively.

Note that the state vector $\boldsymbol{x}(i)$ in (5.15) is the state vector of the augmented system (5.9). At the initial time step $i = 0$, the estimated state vector is used: $\boldsymbol{x}(i = 0) = \hat{\boldsymbol{x}}(k)$. The state equations of the augmented system (5.9) are used to compute the future states within the prediction horizon.

$\bar{\boldsymbol{x}}_t$ and $\bar{\boldsymbol{u}}_t$ are the target values for the state vector and the control input, respectively. These target values are computed by solving the system of equations

$$\begin{bmatrix} \boldsymbol{A} - \boldsymbol{I} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{x}}_t \\ \bar{\boldsymbol{u}}_t \end{bmatrix} = \begin{bmatrix} -\boldsymbol{B}_d \hat{\boldsymbol{d}}(k) \\ \boldsymbol{r}(k) - \boldsymbol{H}\boldsymbol{C}_d \hat{\boldsymbol{d}}(k) \end{bmatrix}. \tag{5.16}$$

The matrix $\boldsymbol{H}$ selects the outputs to be tracked among all outputs. Since in this case all outputs are desired to be tracked, $\boldsymbol{H}$ is chosen to be the identity matrix ($\boldsymbol{H} = \boldsymbol{I}$).

Note that for this application the system of equations (5.16) specifies $\bar{\boldsymbol{x}}_t$ and $\bar{\boldsymbol{u}}_t$ uniquely. If this is not the case, this can be seen as an optimization problem and it can be solved e.g. by minimizing $\|\bar{\boldsymbol{u}}_t\|_2$, where $\|\cdot\|_2$ indicates the Euclidean norm of a vector.

5.4 Extended Prediction Horizon

If we take a look on the model of the fluidized bed furnace, we can see that the time constants strongly differ. This is because of the high heat capacity of the fluidized bed, due to the tons of sand in the bed. Therefore, its temperature T_b changes very slow compared to the other output variables. The sampling time of the MPC algorithm is set to 10 s what seems appropriate for the outputs $\dot{V}_\mathrm{f}$, T_f, c_{O_2} and $\dot{V}_\mathrm{fluid}$. Nevertheless, this is a very short time period for the dynamics of the fluidized bed temperature T_b. If the reference for T_b should be tracked successfully, the prediction horizon N_p needs to be extended for the MPC to "see" the slow dynamic of T_b within the prediction horizon. However, a long prediction increases the size of the optimization problem what leads to high computational effort. Therefore, two different sampling times are used for the MPC algorithm: a coarse sampling time, in order to track the slowly changing bed temperature T_b and a fine one for the other four output variables.

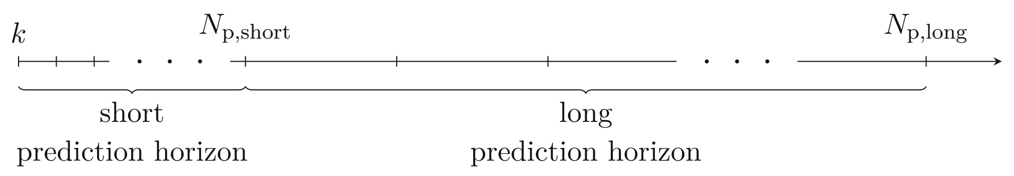


Figure 5.2: Extended prediction horizon principle. The first predictions are computed using a fast sampled model, whereas for predictions beyond $N_{\mathrm{p,short}}$ a slow sampled model is used.

Imagine that at any time step k we want to predict the system's future behavior. A fast sampled model with $T_{\mathrm{s}} = 10\,\mathrm{s}$ is used to predict the outputs within a prediction sufficient for the fast outputs ($N_{\mathrm{p,short}}$). Afterwards, a slow sampled model with $T_{\mathrm{s}} = 300\,\mathrm{s}$ is used to predict the outputs for the time period between $N_{\mathrm{p,short}}$ and $N_{\mathrm{p,long}}$. Figure 5.2 gives an illustration of the extended prediction horizon principle.

5.5 Results

To show the performance of the model predictive controller, simulations are carried out. For the simulations, the 1st-order model with the improved model structure is used (the state-space matrices can be found in the Appendix). The length of the short prediction horizon $N_{\mathrm{p,short}}$, the length of the long prediction horizon $N_{\mathrm{p,long}}$ as well as the length of the control horizon N_{c} are set to 10. The matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ in the cost function (5.15) are chosen as

$$\boldsymbol{Q} = \mathrm{diag}\left(\begin{bmatrix} 1 & 10^2 & 10^3 & 10^9 & 1 \end{bmatrix}\right), \; \boldsymbol{R} = 10^{-3} \cdot \boldsymbol{I}. \tag{5.17}$$

The model predictive controller is implemented using the YALMIP MATLAB toolbox [28]. This toolbox uses different solvers, depending on the type of the optimization problem. In this case, the QUADPROG solver is used, which is part of the MATLAB Optimization toolbox [29].

In the simulation, a step in the reference is applied for each output. The results are shown in Figure 5.3. The second output (T_{f}) cannot reach the reference value due to constraints, The other four outputs can follow their reference values. As expected, the third output (T_{b}) is slower than the other outputs, due to the high inertia of the fluidized bed.

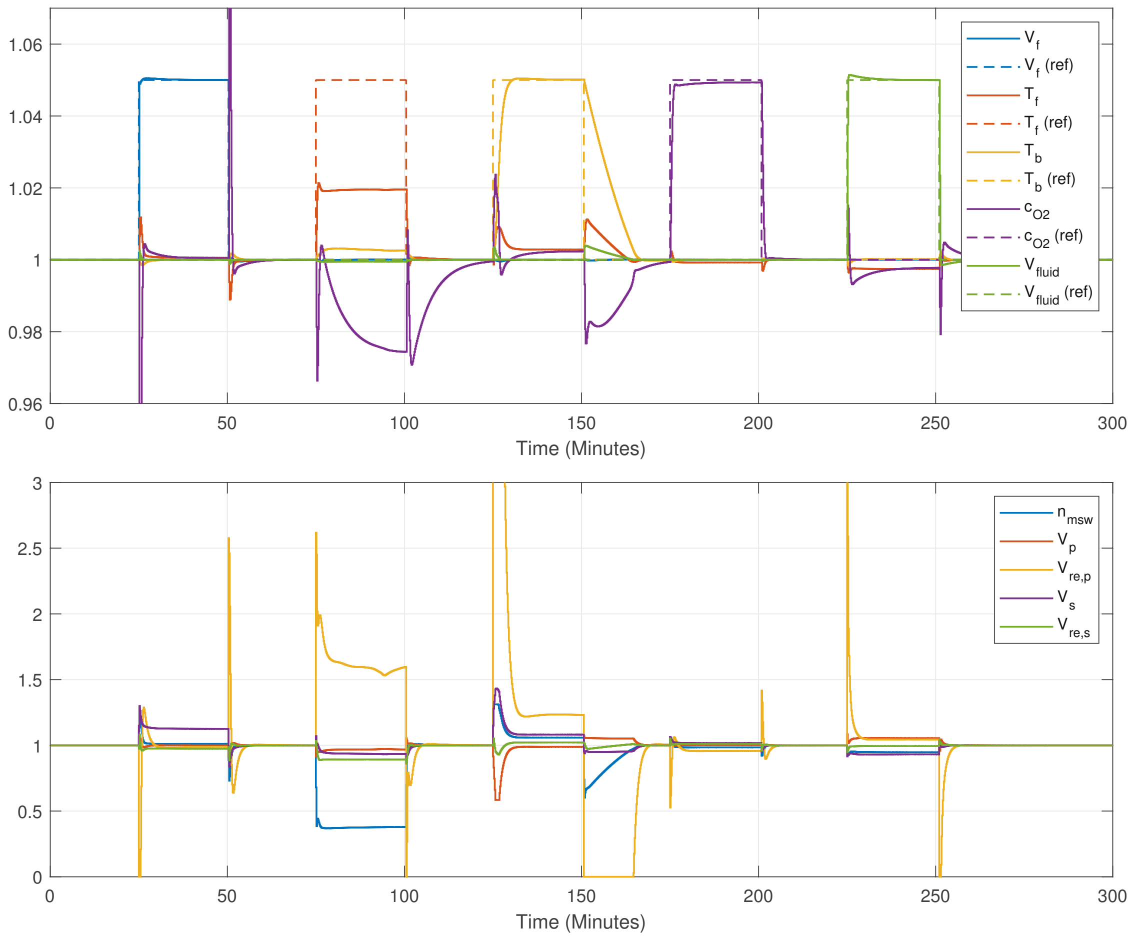


Figure 5.3: Simulation of the system's response to steps in the references for the 5 outputs with the corresponding inputs. Note the normalized representation.

Chapter 6

Conclusion

The research questions defined in the Chapter 1 can be answered as follows:

Which model gives the best mathematical description of the fluidized bed furnace and can then be used to implement a linear model predictive control scheme?

State-space models allow a convenient description of multiple-input multiple-output systems. Furthermore, MPC algorithms are most often designed for state-space models. If the model used for control is linear, the optimization problem in the MPC algorithm can be solved analytically. Therefore, a linear state-space model is used to describe the process of the fluidized bed furnace. However, nonlinearities of the process are taken into account by adding time-varying properties to the input matrix. The models with the structure improved by an evolutionary-based algorithm give an even better fit on the crossvalidation data. Thus, this model is used for the MPC.

For the parameter estimation, measurement data from the waste incineration plant is used. The data is preprocessed in terms of filtering, resampling and normalization. The data has been collected while the plant was operating in closed-loop. Drawbacks of the models identified with this data can be seen by looking on the step responses of the models: In some cases, the models behave not as physically expected.

What is an effective model predictive control algorithm that can handle the process dynamics of the fluidized bed furnace?

The standard linear MPC scheme has been extended with two features: To address the problem, that the time constants of the process are strongly different, the MPC uses two different sampled models and operates therefore in two different time scales. Moreover, the linear-state space model has been extended with additional disturbance states, what allows offset-free tracking of stationary set points. The developed MPC algorithm is then able to control the fluidized bed furnace in simulations successfully.

Before the MPC is implemented at the plant, more simulations can be carried out for different scenarios, like different kinds of disturbances acting on the plant.

Appendix A

Appendix

A.1 Linear State-Space Models

Sampling Time for all models $T_\mathrm{s} = 10\,\mathrm{s}$.

A.1.1 Initial Structure

These models are developed in Chapter 3 based on energy and mass balances for the furnace.

1st-Order Model

$$\boldsymbol{A} = \begin{bmatrix} 0.8396 & 0 & 0 & 0 & 0 \\ -0.6069 & 0.7391 & 0.0032 & 0 & 0 \\ 0 & 0 & 0.9967 & 0 & 0 \\ 0.6632 & 0 & 0 & 0.2959 & 0 \\ 0 & 0 & 0 & 0 & 0.8164 \end{bmatrix}$$

$$\boldsymbol{B} = \begin{bmatrix} -0.018 & 0.0552 & 0.0009 & 0.1156 & -0.0246 & 0 & 0 & 0 & 0 \\ -0.1201 & 0.1104 & -0.0155 & 0 & 0 & 0 & 0 & 0.42 & -0.0666 \\ 0.0008 & 0.0294 & -0.275 & 0 & 0 & -0.0308 & 0.2773 & 0 & 0 \\ 0.1403 & 0.0007 & 0.0225 & -0.0274 & 0.1819 & 0 & 0 & 0 & 0 \\ 0 & 0.2548 & 0.1492 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

2nd-Order Model

$$
\begin{bmatrix}
0.3846 & 0 & 0 & 0 & 0 & 1.0 & 0 & 0 & 0 & 0 \\
0.3512 & 1.584 & -0.2267 & 0 & 0 & 0 & 1.0 & 0 & 0 & 0 \\
0 & 0 & 0.4908 & 0 & 0 & 0 & 0 & 1.0 & 0 & 0 \\
2.6853 & 0 & 0 & 1.5999 & 0 & 0 & 0 & 0 & 1.0 & 0 \\
0 & 0 & 0 & 0 & 0.0056 & 0 & 0 & 0 & 0 & 1.0 \\
0.2593 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-0.3104 & -0.6244 & 0.2309 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.5055 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1.9942 & 0 & 0 & -0.6778 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.8114 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

$$
\boldsymbol{B} = \begin{bmatrix}
-0.5685 & -0.4802 & -0.1203 & -0.0314 & -0.1439 & 0 & 0 & 0 & 0 \\
-0.1562 & -0.1844 & -0.061 & 0 & 0 & 0 & 0 & 0.2888 & -0.0678 \\
-0.0446 & 0.4863 & 0.524 & 0 & 0 & -0.8264 & -0.4741 & 0 & 0 \\
1.3872 & 1.5029 & 0.3497 & 0.2595 & 0.3237 & 0 & 0 & 0 & 0 \\
0 & 1.3704 & 0.2444 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.6426 & 0.6191 & 0.0992 & 0.2525 & -0.0521 & 0 & 0 & 0 & 0 \\
0.1424 & 0.1525 & 0.0608 & 0 & 0 & 0 & 0 & -0.3249 & 0.0794 \\
0.0457 & -0.4503 & -0.8676 & 0 & 0 & 0.7885 & 0.8201 & 0 & 0 \\
-1.5414 & -1.762 & -0.3064 & -0.6558 & 0.077 & 0 & 0 & 0 & 0 \\
0 & -1.1061 & -0.0895 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

A.1.2 Improved Structure

In Chapter 4, the initial model structure is improved using an evolutionary algorithm. The resulting models are presented in this section.

1st-Order Model

$$
\boldsymbol{A} = \begin{bmatrix}
0.3087 & -0.5057 & 0 & -0.2697 & 0 \\
-0.4746 & 0.4305 & -0.0176 & -0.5935 & 0 \\
0 & 0 & 0.9975 & 0.0072 & 0.0379 \\
0.5026 & 0.4903 & -0.0152 & 1.2193 & 0 \\
0 & 0 & 0 & 0 & 0.7786
\end{bmatrix}
$$

$$\boldsymbol{B} = \begin{bmatrix} -0.1051 & 0.1524 & -0.0706 & 0.6397 & -0.2635 & 0 & 0 & 0 & 0 \\ -0.24 & 0.0793 & -0.0259 & 0 & 0 & 0 & 0 & 0.585 & -0.0206 \\ 0.0034 & -0.0295 & -0.23 & 0 & 0 & -0.0236 & 0.2024 & 0 & 0 \\ 0.1052 & -0.0607 & 0.0666 & -0.4618 & 0.2621 & 0 & 0 & 0 & -0.0539 \\ 0 & 0.3069 & 0.1715 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

2$^{\text{nd}}$-Order Model

$$\begin{bmatrix} 1.0836 & 0 & 0 & -0.0157 & 0 & 1.0 & 0 & 0 & 0 & 0 \\ 0 & 0.834 & 0 & 0 & 0 & 0 & 1.0 & 0 & 0 & 0 \\ 0 & -0.0816 & 0.9971 & 0 & 0 & 0 & 0 & 1.0 & 0 & 0 \\ 0 & 0.1177 & 0 & 0.6397 & 0 & 0 & 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & 0 & -0.0139 & 0 & 0 & 0 & 0 & 1.0 \\ -0.2951 & 0 & 0 & 0.1526 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.0865 & 0 & 0 & -0.1992 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.0681 & 0 & 0 & 0.0037 & 0 & 0 & 0 & 0 & 0 \\ 0.2103 & 0 & 0 & 0.1057 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.798 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{B} = \begin{bmatrix} -0.5596 & 0.0452 & 0 & 0.1884 & 0 & 0.0488 & 0 & 0 & -0.1901 \\ -0.4876 & 0 & 0 & 0.5936 & 0 & 0 & 0 & -0.463 & -0.06 \\ -0.1437 & 0 & -0.0009 & 0 & 0 & -0.2779 & 0 & 0 & 0 \\ 1.3561 & 0 & 0 & 0 & 0 & 0.4255 & 0 & 0 & 0.269 \\ 0 & 1.4424 & 0.183 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.6111 & 0 & 0 & -0.1092 & 0.0502 & 0 & 0 & 0 & 0 \\ 0.4561 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0439 \\ 0.1429 & 0 & 0 & 0 & 0 & 0.2703 & 0 & 0 & 0 \\ -1.409 & 0 & 0 & 0 & 0 & -0.4659 & 0 & 0 & -0.0568 \\ 0 & -1.1318 & -0.0003 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Bibliography

[1] CEWEP waste-to-energy plants in europe in 2018. https://www.cewep.eu/waste-to-energy-plants-in-europe-in-2018/. Accessed: 2021-02-18.

[2] CEWEP latest eurostat figures: Municipal waste treatment 2018. https://www.cewep.eu/municipal-waste-treatment-2018/. Accessed: 2021-02-18.

[3] Bo Leckner and Fredrik Lind. Combustion of municipal solid waste in fluidized bed or on grate–a comparison. *Waste Management*, 109:94–108, 2020.

[4] J. Van Caneghem, A. Brems, P. Lievens, C. Block, P. Billen, I. Vermeulen, R. Dewil, J. Baeyens, and C. Vandecasteele. Fluidized bed waste incinerators: Design, operational and environmental issues. *Progress in Energy and Combustion Science*, 38(4):551–582, 2012.

[5] Derek Geldart. Types of gas fluidization. *Powder technology*, 7(5):285–292, 1973.

[6] HT Bi and JR Grace. Flow regime diagrams for gas-solid fluidization and upward transport. *International Journal of Multiphase Flow*, 21(6):1229–1236, 1995.

[7] M. Leskens, L.B.M. van Kessel, and O.H. Bosgra. Model predictive control as a tool for improving the process operation of msw combustion plants. *Waste Management*, 25(8):788–798, 2005.

[8] M. Leskens, P.P. van't Veen, L.B.M. van Kessel, O.H. Bosgra, and P.M.J. Van den Hof. Improved economic operation of mswc plants with a new model based pid control strategy. *IFAC Proceedings Volumes*, 43(5):655–660, 2010. 9th IFAC Symposium on Dynamics and Control of Process Systems.

[9] Yingmin Jia, H. Kokame, and J. Lunze. Simultaneous adaptive decoupling and model matching control of a fluidized bed combustor for sewage sludge. *IEEE Transactions on Control Systems Technology*, 11(4):571–577, 2003.

[10] Lennart Ljung. *System identification : theory for the user*. Prentice Hall information and system sciences series. Prentice Hall, Upper Saddle River, N.J., 2nd ed.. edition, 1999.

[11] Mohamad H Hassoun et al. *Fundamentals of artificial neural networks*. MIT press, 1995.

[12] Jerome T Connor, R Douglas Martin, and Les E Atlas. Recurrent neural networks and robust time series prediction. *IEEE transactions on neural networks*, 5(2):240–254, 1994.

[13] Martin T Hagan and Mohammad B Menhaj. Training feedforward networks with the marquardt algorithm. *IEEE transactions on Neural Networks*, 5(6):989–993, 1994.

[14] Urban Forssell and Lennart Ljung. Closed-loop identification revisited. *Automatica*, 35(7):1215–1241, 1999.

[15] Paul Van den Hof. Closed-loop issues in system identification. *Annual reviews in control*, 22:173–186, 1998.

[16] Inc. The MathWorks. *System Identification Toolbox*. Natick, Massachusetts, United State, 2021.

[17] Inc. The MathWorks. *Deep Learning Toolbox*. Natick, Massachusetts, United State, 2021.

[18] Oliver Kramer. Genetic algorithms. In *Genetic algorithm essentials*, pages 11–19. Springer, 2017.

[19] Otoniel Buenrostro-Delgado, Juan Manuel Ortega-Rodriguez, Kevin C Clemitshaw, Carlos González-Razo, and Iván Y Hernández-Paniagua. Use of genetic algorithms to improve the solid waste collection service in an urban area. *Waste management*, 41:20–27, 2015.

[20] Ben McKay, Mark Willis, and Geoffrey Barton. Steady-state modelling of chemical process systems using genetic programming. *Computers & chemical engineering*, 21(9):981–996, 1997.

[21] Kemal Özkan, Şahin Işık, Zerrin Günkaya, Aysun Özkan, and Müfide Banar. A heating value estimation of refuse derived fuel using the genetic programming model. *Waste Management*, 100:327–335, 2019.

[22] Noraini Mohd Razali, John Geraghty, et al. Genetic algorithm performance with different selection strategies in solving tsp. In *Proceedings of the world congress on engineering*, volume 2, pages 1–6. International Association of Engineers Hong Kong, 2011.

[23] Liuping Wang. *Model predictive control system design and implementation using MATLAB®*. Springer Science & Business Media, 2009.

[24] Basil Kouvaritakis and Mark Cannon. *Model predictive control : classical, robust and stochastic*. Springer, 2016.

[25] Urban Maeder, Francesco Borrelli, and Manfred Morari. Linear offset-free model predictive control. *Automatica*, 45(10):2214–2222, 2009.

[26] Manfred Morari and Urban Maeder. Nonlinear offset-free model predictive control. *Automatica*, 48(9):2059–2067, 2012.

[27] Gabriele Pannocchia and James B Rawlings. Disturbance models for offset-free model-predictive control. *AIChE journal*, 49(2):426–437, 2003.

[28] Johan Lofberg. Yalmip: A toolbox for modeling and optimization in matlab. In *2004 IEEE international conference on robotics and automation (IEEE Cat. No. 04CH37508)*, pages 284–289. IEEE, 2004.

[29] Inc. The MathWorks. *Optimization Toolbox*. Natick, Massachusetts, United State, 2021.