Multilinear Design of Decentralized Controller Networks for Building Automation Systems

Vorgelegt im Promotionsausschuss der
HafenCity Universität Hamburg

zur Erlangung des akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)

Dissertation

von
Kai Kruppa

aus
Elmshorn

2018

1. Gutachter: Prof. Dr.-Ing. Harald Sternberg
2. Gutachter: Prof. Dr.-Ing. habil. Gerwald Lichtenberg

Vorsitzender der Prüfungsausschusses: Prof. Dr.-Ing. Jochen Schiewe
Zusätzliche Professorin im Prüfungsausschuss: Prof. Dr.-Ing. Annette Eicker

Tag der mündlichen Prüfung: 30.10.2018

Abstract

In Germany 35% of the end energy is currently consumed in the buildings sector. Model-based controller design methods offer the possibility to use the high energy saving potential to reduce this energy demand. Several tools for model-based design are available for linear state space models. But linear methods may fail when nonlinear effects are essential as e.g. in the area of buildings. Multilinear time-invariant (MTI) models extend the linear model class and lead to better modeling properties, e.g. for heating systems. Today's applications focus more and more on complex systems. To deal with these large-scale systems a MTI model representation with different decomposed tensor methods is derived here to get a memory efficient model description. Tensor methods are used to develop different model analysis and controller design methods especially for MTI models. Since decomposed MTI models allow to describe the dynamics of large-scale systems, methods for distributed controller design of such systems are derived based on the decomposed tensor structure of the models to reduce the communication effort as well as the computational complexity. The controller algorithms are applied in simulation to different heating systems examples to show their advantages for building automation. The results of a successful realtime application of a predictive controller at a realworld office building using the MTI tensor structure are presented.

In Deutschland werden 35% der Endenergie im Gebäudebereich verbraucht. Modellbasierte Reglerentwurfsmethoden ermöglichen das große Energieeinsparpotenzial zu nutzen, um den Energieverbrauch zu reduzieren. Für lineare Zustandsraummodelle sind dazu zahlreiche Methoden bekannt. Jedoch können diese Methoden versagen, wenn nichtlineare Effekte entscheidend sind, wie im Gebäudebereich. Multilineare zeitinvariante (MTI) Modelle erweitern die lineare Modellklasse und führen zu besseren Modellierungseigenschaften, z.B. für Heizungssysteme. In heutigen Anwendungen werden die betrachteten Systeme immer komplexer. Zur Modellierung solch großer Systeme werden hier Tensorzerlegungsverfahren angewendet, um eine speicherplatzeffiziente Modelldarstellung zu erreichen. Tensormethoden werden weiterhin dazu verwendet, verschiedene Methoden zur Modellanalyse und zum Reglerentwurf zu entwickeln. Da sich MTI Modelle dazu eignen das Verhalten sehr großer Systeme zu beschreiben, werden, basierend auf der Tensorstruktur, Methoden für verteilte Regelungen entworfen, um den Kommunikationsaufwand und die Komplexität der Regler zu reduzieren. Die entworfenen Regler werden in der Simulation auf verschiedene Beispiele aus dem Bereich der Heizungssysteme angewendet, um deren Vorteile zu zeigen. Zudem erfolgt die erfolgreiche Echtzeitanwendung eines prädiktiven Reglers auf ein reales Bürogebäude, der die multilineare Tensorstruktur des Modells nutzt.

Acknowledgement

During the years of research resulting in this thesis many people supported me in an extraordinary way. I am very thankful for that. I like to mention some special people that had a lot of influence on me during that time.

I like to thank you, Gerwald Lichtenberg, for being my supervisor. You gave many inspirations as well on topic as off topic and gave me a lot of support through my whole studies from Bachelor to Master degree and finally for my PhD research. Our discussions opened a wide view on control engineering for me. Thank you, Harald Sternberg for giving me the possibility to come to the HafenCity University and being my supervisor. Thank you, Jochen Schiewe and Annette Eicker for leading the examination committee and being part of it.

Another important factor was the team at the Hamburg University of Applied Sciences and my other PhD colleagues. Björn Lautenschlager, we took the whole journey of the research project together. This was a great time inside and outside the office together with you. Thank you for that. Without you Georg Pangalos I might never started this PhD adventure. Thank you for all the fruitful discussions we had. Our coffee rounds with Dagmar Rokita and Carsten Westarp were always great possibilities to open the mind again. Thank you for always taking time for talking about the current topics or problems. I also like to thank Erik Sewe and Thorsten Müller-Eping for the way through the project and the successful fights with the tensors.

A very special and great thank you goes to my parents, girlfriend and my whole family. You always believed in me and supported me in a way that is amazing.

You and all others who supported me, gave me a great background for my work. You are great people. All this would not have been possible without you.

Thank you!!!

Contents

List of Figures

List of Tables

List of Abbreviations

ADMM . . . . . alternating direction method of multipliers
AHU . . . . . air handling unit
ALS . . . . . alternating least squares
AMPC-SL . . . . . adaptive model predictive control with successive linearization

BACnet . . . . . Building Automation and Control Networks
BMS . . . . . building management system

CP . . . . . canonical polyadic

DDC . . . . . direct digital control
DMPC . . . . . distributed model predictive control

EMPC . . . . . economic model predictive control

GPIO . . . . . general purpose input output

HIL . . . . . hardware-in-the-loop
HOSVD . . . . . higher order singular value decomposition
HT . . . . . hierarchical Tucker
HVAC . . . . . heating, ventilation and air conditioning

LPV . . . . . linear parameter-varying
LQR . . . . . linear quadratic regulator

MIMO . . . . . multiple-input multiple-output
MPC . . . . . model predictive control
MTI . . . . . multilinear time-invariant

NMPC . . . . . nonlinear model predictive control

ODE . . . . . ordinary differential equation

PI . . . . . proportional and integral

SISO . . . . . single-input single-output

SVD . . . . . singular value decomposition

TT . . . . . tensor train

UDP . . . . . user datagram protocol

1 Introduction

At the beginning of this introduction a motivation for modeling and control of heating systems is given. The research questions are stated and the current state of research is described. A simple example motivates the description of the thermal behavior of heating systems components by multilinear models. This leads to the main contributions of the thesis. Finally an outline is given.

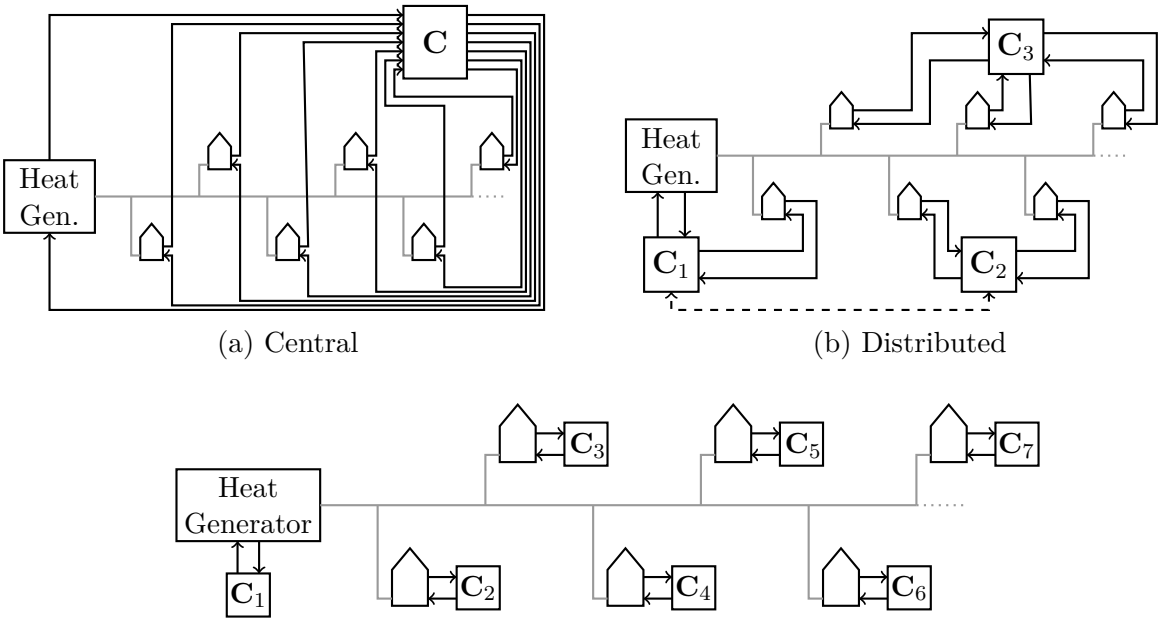
1.1 Motivation

Heating and warm water generation have a great impact on the overall energy consumption in Germany. Current reports state, that 35% of the end energy is consumed in the buildings sector. With a high rate of 37%, non-residential buildings contribute significantly to this consumption. A relevant group of buildings in this sector are office buildings. The main part of the energy is consumed for room temperature regulation, [10, 100, 111]. To reduce this high energy consumption e.g. the German government stated the following goals in its sixth energy research program concerning the buildings sector. It is declared, that until 2020 the heat demand of the building sector has to be reduced by 20% and the greenhouse gas emission should be decreased by 40% compared to the year 1990. Until 2050, a nearly climate-neutral building stock should be realized, meaning a reduction of the primary energy demand by 80%. The rate of building renovation has to be doubled.
Current studies showed, that these ambitious goals will not be reached with the actual activities, [10]. New strategies have to be taken into consideration. It is not sufficient to construct new buildings only with high energy efficiency standards to fullfill the requirements. The existing building stock has to be optimized, too. Making constructive changes e.g. at the building envelope is often related to high costs. But very often a high energy consumption is caused because the provided energy is not used properly. In many buildings basic control concepts like simple switching rules or PID controllers are used. Experiences show that furthermore these controllers are often badly tuned. In [29] it is estimated, that an appropriate controller management in buildings offer a energy saving potential of nearly 30%. This shows, that with an improved building control a high amount of energy can be saved and a waste of energy is prevented. To achieve that, no major constructional changes have to be made, but the controller structure must be improved. It is stated that these potentials can just be exploited by applying advanced controller concepts using e.g. models of the buildings or weather forecasts. The application of those advanced methods is challenging but offers the possibility to save a lot of energy.

Standard controller design leads to a centralized controller who needs the full system information, i.e. measurements of all system states to compute the control inputs for the whole

plant. Today's applications lead to more and more complex systems, like in smart grids, complex heating, ventilation and air conditioning (HVAC) systems or multi-agent systems. A centralized design would result on the one hand in large communication effort and very complex communication infrastructure and on the other hand to very much computational effort for the central controller node, [70]. This suggests the implementation of controller networks, where the control task is distributed to several nodes as shown exemplary in the following example, [6, 102].

Example 1.1 *Consider a large-scale system, where a heat generation unit supplies many houses of a district with heat. Each house and the heat generation unit has its own sensors and actuators. The setup is schematically illustrated in Figure 1.1.*



(a) Central (b) Distributed

(c) Decentralized

Figure 1.1: Heat distribution with state feedback controller

In central design each of the sensors and actuators have to be connected to the central controller $\mathbf{C}$*. As shown in Figure 1.1a, this leads to a large communication effort especially when the consumers are locally distanced. Furthermore large computations inside the controller are necessary since many control signals for the heat generation unit and each consumer have to be determined. The idea is to find a controller with an acceptable loss in control performance, that does not need access to the full system information. Cutting some of the communication links as shown in Figure 1.1b leads to a less complicated implementation of the control system since less components, e.g. cables, have to be installed. During operation the communication traffic is reduced, too. Maybe some communication between the controllers* $\mathbf{C}_1$*,* $\mathbf{C}_2$ *and* $\mathbf{C}_3$ *is necessary. The problems, that have to be solved inside the controllers are of lower complexity too. A very high degree of decentralization would have been reached if each component has its own feedback controller that uses only the sensors and actuators of its own subsystem without any information of the other systems as in Figure 1.1c. The*

controller simply have to compute the control signals of their own subsystem. But this might have a negative effect on the controller performance. Which degree of decentralization is possible depends on the application, i.e. the plant that should be controlled. How to determine a suitable structure is investigated in Section 4.2.

The exemplary illustration shows, that with a distributed controller setup the complexity of the communication infrastructure and the computational complexity of the controller nodes can be reduced, which is necessary, when advanced controller strategies are applied to systems of large-scale, as they occur in today's applications.

1.2 Research questions

The scientific interest in this thesis is based on the development of new methods for model-based controller design and their application to heating systems to benefit from the unused energy saving potential of these plants. In control theory a well established approach is the linear model-based controller design. Therefore, at first a nonlinear model of the considered plant is derived. To determine a simpler model description and a controller design of lower complexity, the nonlinear model is approximated around an operating point by a linear model. This linear model is used for controller synthesis or inside the controller depending on the particular design method. The class of linear models is very well analyzed and a widepread theory is available, [4, 23]. But the linear approximation of the plant behavior is only valid with a sufficient accuracy in a limited surrounding area of the operating point. This leads to inaccuracies, if this area is too small compared to the operating range of the plant. In this case the performance of the controller designed with the linear model is limited, because the control quality depends directly on the model accuracy. To avoid this modeling error due to the linear approximation, the nonlinear model may be used directly for a nonlinear controller design. Then the controller works with a model that captures the dynamics of the model better than a linear one. But because of the general model structure without any restrictions of the nonlinear models the computational effort for controller design and operation can get very high. An efficient computation gets complicated, which can lead to problems for real-time implementation.

To close this gap between linear and nonlinear controller design, in this thesis methods for multilinear controller design are investigated. In [63] and [85] the model class of MTI state space models were introduced, that are very efficiently represented by using tensor decomposition methods. The multilinear models extend the class of linear models and thus capture more dynamics but they are not as general as nonlinear models since they imply a certain model structure. The idea is to benefit from the better description of the system dynamics compared to the linear case with a robust controller design method of medium complexity compared to nonlinear design with the specialization to multilinear models. The focus here is on optimal control strategies, since they allow by definition an optimal operation of the plant. Because multilinear models are described by decomposed tensors in a very efficient way, modeling and controller design for large-scale system is considered, too, leading to distributed methods. This results in the following research questions.

- Which tensor decomposition methods are suitable for the complexity reduction of MTI models and their controller design methods?
- How can tensor methods and the multilinear model structure improve the controller design process of known methods like feedback linearization or predictive control?
- How to derive efficient distributed controller design methods for a multilinear model in tensor representation?
- What are the benefits for the application of decomposed models and design methods to heating systems?
- Is it possible to implement a multilinear model-based controller on real-time hardware?

1.3 State of research

Model-based controller design methods are applied in many areas in industry. Especially in the automotive and process industry they are widely used, [30]. However in the field of building automation they are mostly unused in application. But since heating system structures get more and more complex, advanced controller methods will be necessary to use the high energy saving potential, that is available, [3]. In research modeling and controller design is an active field. In a current large control conference, the IFAC World congress 2017 many papers where presented related to this task, e.g. [25, 38, 69, 77, 93, 105, 110] and a special session *"Modeling, Control and Fault Diagnosis for Building Energy Management Systems"* was provided, which shows the importance of this task.

A big challenge for the construction of building models is, that they are unique and individually planned in many cases. A building is not a mass product. In many modelling approaches this leads to very complex nonlinear models, that are very detailed, [33]. Controlling nonlinear systems by either linear or nonlinear controllers are challenges of today's applications. But such complex nonlinear models result in complex controller design processes in many cases, [43]. That is why models of lower complexity are of interest. In control theory many methods exist for modeling and controller design of linear systems, [4, 23]. For systems, where nonlinear effects are essential, these methods may fail, since the linear model does not capture the dynamical behavior of the plant sufficiently well. Other approaches are available, that focus on special classes of nonlinear systems, like the bilinear, [26]. Usually specialization on specific classes of systems lowers the complexity and improves the robustness of the design process. Therefore, e.g. the nonlinear design method of feedback linearization was investigated for a subclass of nonlinear systems, that can be approximated well by bilinear systems, [78]. But for some applications, smaller model classes are too restrictive, like the bilinear one for heating systems, [86].

Many physical systems can be modeled by mass, energy or heat flow balances, like heating systems or chemical systems, resulting in a state space model. The right hand sides often show a multilinear structure in relevant applications, i.e. they are linear if all but one state or input are held constant, [85]. It also extends the class of bilinear systems. Even though the systems have no inherent multilinear structure, it was shown, that the system behavior can be approximated adequately by an MTI model, [50]. This model class offers a structure,

that fits to the application of heating systems, since they are modeled by heat balances, [62]. This allows to develop models of heating systems, that capture more dynamics than a linear model and with less generality than nonlinear models. By modeling subsystems, that occur several times in different applications in this structure, helps to reuse the models and to encounter the individuality of the plants in the building field.

It was shown, that MTI systems can be presented in a tensor framework, [63, 85]. In mathematics tensor calculus is an active field of research and applied in the areas of signal processing, neuroscience, data analysis and machine learning, [19, 31, 60]. Since the number of parameters for MTI systems increase exponentially with the number of states and inputs, a full tensor representation leads to a severe memory demand for large-scale systems. Thus, tensor decomposition techniques can be used to reduce the storage effort and makes them computationally manageable. The applicability of decomposed tensors in the field of controller design and diagnosis was shown in [85] or [79]. From mathematics several decomposition methods are offered to reduce the storage demand of tensors. The four most common are canonical polyadic (CP), [44], Tucker, [18], tensor train (TT), [83] and hierarchical Tucker (HT) decomposition [45], that are focused on here for model representation, simulation and controller design for MTI systems. Several toolboxes for tensor calculus are available for MATLAB which supports to use these methods from mathematics in control, [5, 45, 84, 109]. The decomposed tensor representation offers the possibility to represent large-scale heating systems with multilinear models with a manageable amount of parameters, that will be investigated in this thesis. Good modeling properties of MTI systems in the field of heating system were already shown in [86]. First applications of the multilinear model structure in model-based controller design were successfully tested in simulation by [85].

Because of its capability to control multiple-input multiple-output (MIMO) systems with large time constants and delay as well as considering e.g. disturbance estimations and plant constraints directly in the design process, the model-based control method model predictive control (MPC) turned out to be a promising control strategy for heating systems, e.g. for temperature regulation, [35]. Also for temperature regulation problems in other application areas, like the precise control of the cooling system of are large free electron laser, the MPC control strategy showed good results, [51]. Much research is going on in this field, [75, 101]. In many cases MPC is applied using linear models, [1]. With a quadratic cost function, this leads to a convex optimization problem, that can be solved efficiently by standard solvers during controller operation to determine an optimal control input to the plant, [73]. But if nonlinear effects have a lot of influence the resulting modeling error leads to a control error directly. Using a nonlinear model inside the MPC leads in general to a high computational effort, because of the in general non-convexity of the optimization problem. For the class of bilinear systems different adapted optimization algorithms were derived, [42]. Convexity could be reached by linearization around an input trajectory with the drawback of a suboptimal solution of the control problem caused by the approximation, [11, 26]. With MTI models a better approximation compared to the linear case can be achieved, [50]. The MPC optimization problem with multilinear models was investigated in [56]. Further investigations are given in this thesis.

To apply the controller at a realworld plant, several real-time implementations of MPC in the field of heating systems were published, [37, 74]. Different hardware platforms were used.

Due to the convexity properties of the optimization problems, linear MPC designs were used for these applications. In this work the focus is on MTI models.

In the area of heating systems the plants get more and more complex by e.g. larger buildings, a mix of different generation units, the integration of renewable energy sources or the connection to the smart grid, [88]. This leads on the one hand to a large communication effort and on the other hand to complex computations, if a central controller is used. Decentralized or distributed controller networks help to overcome this problem, [6, 70, 102]. To reduce the communication effort, [66], [89] and [97] proposed methods to find a decentralized state feedback structure with less communication links than in the central case. This could lead to an enormous simplification of the communication structure.

For optimal control methods like MPC large-scale systems lead to a huge computational effort, if a central design is used, since a large optimization problem has to be solved. Therefore, methods were developed to split the global control task to several controller nodes, such that subproblems of lower complexity have to be solved only, [16, 80, 95]. But the published approaches do not consider the special multilinear system structure and the representation with the help of decomposed tensors as it is investigated in this thesis.

1.4 Modeling of heating systems

Models of the thermal dynamics of physical systems can be derived by thermal balances using the law of conservation of energy. The thermal energy is described by

$$Q = c\rho VT,$$

with specific heat coefficient c, density ρ, volume V and temperature T of the medium, [99]. The heat flow follows from the time derivative by

$$\dot{Q} = c\rho\dot{V}T + c\rho V\dot{T}.$$

For a water flow $\dot{V} \neq 0$ the heat flow simplifies to

$$\dot{Q} = c\rho\dot{V}T.$$

The change of heat stored inside a system is given by

$$\dot{Q} = c\rho V\dot{T},$$

since the volume of the system is assumed to be constant, which holds for many heating systems, [85]. The heat balance is built by setting the sum of all heat flows entering and exiting the system equal to the change of heat stored inside a system. The flow into the system contributes with a positive sign to the sum and the flow leaving the system with a negative sign.

The approach is illustrated by a simple heating systems example of a radiator, as depicted in Figure 1.2.

Figure 1.2: Radiator example

The radiator is fed with water of a given supply temperature T_s. A pump is installed, that generates a flow $\dot{V}$ through the radiator. The pump is controlled by a signal $\varphi \in [0, 1]$, such that a maximal flow $\dot{V}_{max}$ can be generated. It is assumed, that the pump does not react instantaneously on a change in the control signal, such that the flow is modeled as a first order system by

$$\ddot{V} = -\frac{1}{\tau_{\dot{V}}}\dot{V} + \frac{\dot{V}_{max}}{\tau_{\dot{V}}}\varphi,$$

with time constant $\tau_{\dot{V}}$. To heat the room, a constant heat flow $\dot{Q}_{heat}$ of the radiator to the environment is assumed for reasons of simplicity for this example. Cooled down water with return temperature T_r leaves the radiator. Thus, with the assumption, that the water inside the radiator is perfectly mixed at temperature T_r, the heat balance for the radiator is given by

$$c\rho V_{rad}\dot{T}_r = c\rho\dot{V}T_s - c\rho\dot{V}T_r - \dot{Q}_{heat},$$

where V_{rad} is the volume of the radiator. These two equations describe the dynamical behavior of the radiator leading to a state space model

$$\begin{aligned}\dot{x}_1 &= p_1\left(u_1x_2 - x_1x_2\right) + p_2,\\ \dot{x}_2 &= p_3x_2 + p_4u_2.\end{aligned}$$

The parameters of the model are given by

$$\begin{aligned} p_1 &= \tfrac{1}{V_{rad}}, \quad p_2 = -\tfrac{1}{c\rho V_{rad}}\dot{Q}_{heat},\\ p_3 &= -\tfrac{1}{\tau_{\dot{V}}}, \quad p_4 = \tfrac{\dot{V}_{max}}{\tau_{\dot{V}}}.\end{aligned}$$

The model has two states $\mathbf{x} = \begin{pmatrix} x_1 & x_2 \end{pmatrix}^T = \begin{pmatrix} T_r & \dot{V} \end{pmatrix}^T$ and two inputs $\mathbf{u} = \begin{pmatrix} u_1 & u_2 \end{pmatrix}^T = \begin{pmatrix} T_s & \varphi \end{pmatrix}^T$. The output of the model is the return temperature $y = T_r$. This simple example shows already, that the models of heating systems do not belong to the class of linear models. Multilinear models extend the class of linear models by allowing multiplications between

states and inputs, such that the model is linear if all but one state or input is held constant. The model of the radiator contains terms x_1x_2 and u_1x_2. The multiplication of inputs and sates is included in the bilinear model class. The multiplication of two states is not bilinear anymore but multilinear, such that the model is not linear or bilinear, but multilinear. For control applications often linear approximations around an operating point are used. The effect of the approximation should be illustrated by two example simulations.

Example 1.2 *The multilinear model perfectly describes the thermal plant dynamics. Figure 1.3 shows the simulation result of the radiator with a constant flow.*

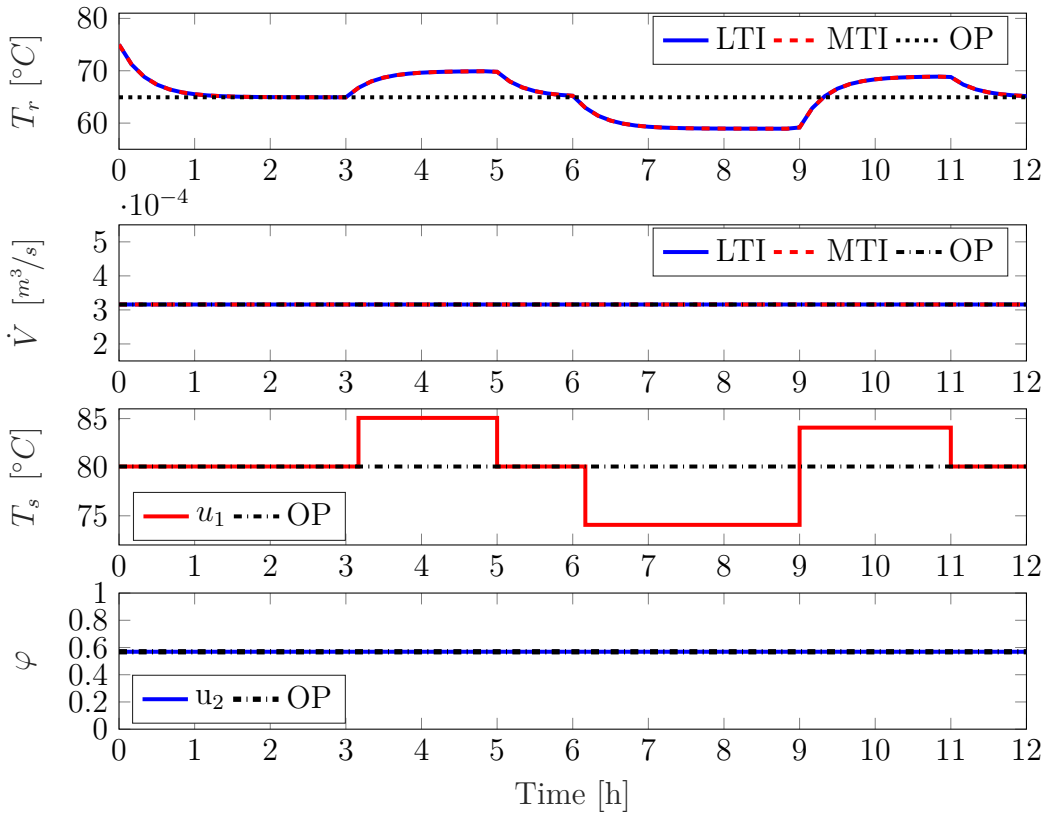


Figure 1.3: Simulation result of the radiator with a constant flow

In the figure the resulting trajectories of the state, i.e. the return temperature and the volume flow and the inputs, i.e. the pump signal and the supply temperature are depicted together with the corresponding operating point. In this case the linear model describes the plant dynamics resulting from a changing supply temperature as well perfectly as can be seen in the simulation results of the return temperature T_r. The reason for that is, that the flow is held constant at the operating point. The situation changes, if the flow is varying, as Figure 1.4 illustrates.

The figure shows the same signals with same linearization and supply temperature input. But now the pump signal is varying - leading to flow changes. The simulated return temperature of the linear approximation shows differences to the multilinear model. Whereas the multi-

linear model can deal with the changing flow and still describes the dynamics perfectly, with the linear model an approximation error occurs, since the changing flow dynamics are not captured by the linear model. Thus already this simple model shows the benefits of a multilinear model for applications in the field of heating systems. This is especially of interest for model-based controller design, since an error of the model as it is observed here with a linear approximation leads to a control error directly. With a multilinear model it is expected, that this error can be reduced for heating systems applications in general, leading to a better controller performance.

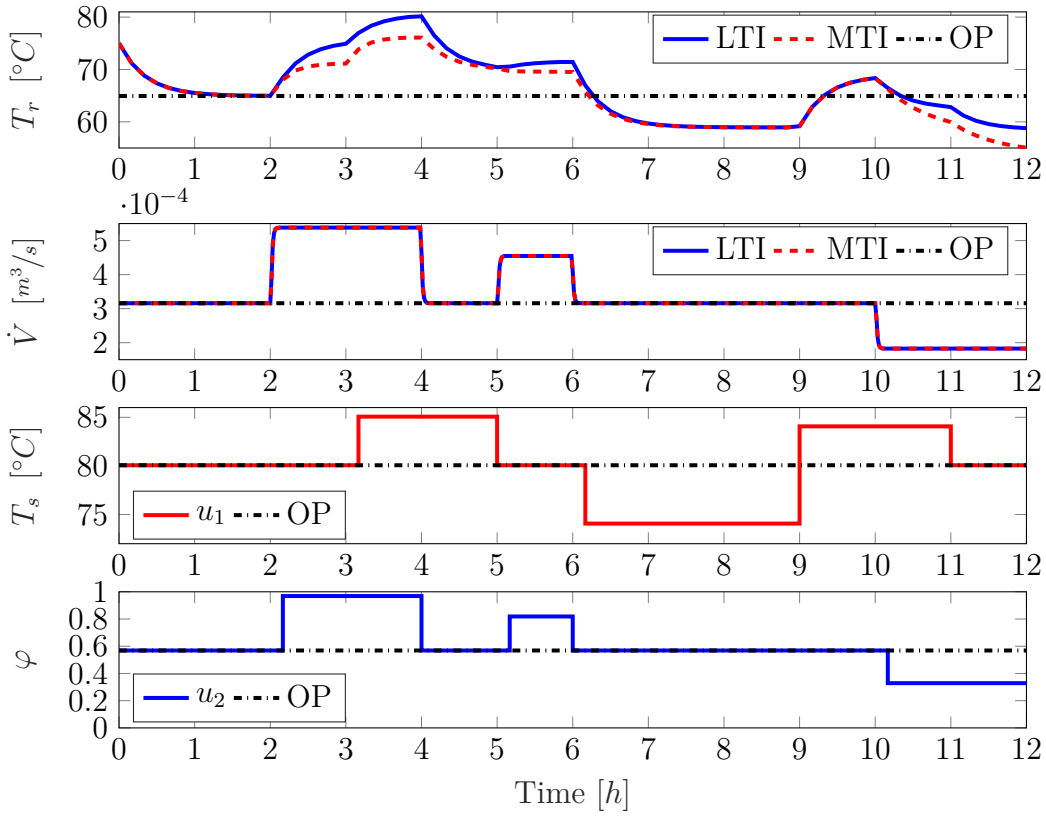


Figure 1.4: Simulation result of the radiator with a varying flow

1.5 Main contributions

This section briefly summarizes the main contributions of this thesis.

Tensor representation of polynomials and basic operators

From [63] or [82] it is known, that polynomials of multilinear structure can be represented in a tensor framework with one tensor containing the parameters and a second tensor containing the monomials. This concept is extended to polynomials of arbitrary order here.

With this generalized formulation it is possible to compute analytically exact solutions of standard polynomial operations like multiplication or partial differentiation just by numerical computations on the parameter tensor. This means e.g. for differentiation, that the parameter tensor of the derivative can be computed simply by using the parameter tensor of the given function.
Furthermore formulations of tensor operations, that are necessary in this thesis, are given, such that they are computed in decomposed format. No full tensors have to be built, if the operands have a decomposed tensor representation. The result is in decomposed form too, which allows an efficient computation on the factors.

Tensor methods for MTI systems

State space models of multilinear systems can be represented in a tensor framework. Since the number of parameters for MTI systems increases exponentially with the number of states and inputs, they are computationally not manageable for large-scale system representation. Tensor decomposition methods break this curse of dimensionality. Therefore representations of MTI models in the four most common tensor decomposition methods, CP, Tucker, TT and HT decomposition are derived. Evaluation methods of the right hand sides of the models based on the decomposed structure are provided. Several tools for work with MTI models are given like linearization or discritization, that work with the decomposed tensor structure. Additionally, a representation of distributed models as well as the serial, parallel and feedback connection, where the subsystems are in MTI representation, are investigated.

Central and distributed controller design for MTI systems

Controller design methods for linear or nonlinear systems are known in the literature. To derive design techniques for MTI systems two approaches are investigated. It is possible to adapt nonlinear controller synthesis methods like feedback linearization to the special model structure of MTI systems, resulting in a controller of fixed structure, that is parameterized by decomposed tensors and is computed analytically exact without any symbolic computations. It is also possible to apply linear methods like linear MPC to MTI systems. By using the multilinear model to adapt the linearization inside the controller, the good modeling properties of the MTI models are connected to the good algorithmic properties of the linear models. Since MTI systems allow to represent large-scale systems, when described in decomposed format, a distributed controller structure can be determined and the control task can be splitted to several controller nodes also when the subsystems are multilinear.

Application of MTI systems to heating systems in modeling and control

Models of large-scale heating and HVAC systems can be represented efficiently as MTI models by different decomposed tensor decomposition method. Using the decomposed multilinear model structure, different controller design methods are tested successfully with heating systems examples. To apply a predictive controller, that uses an MTI model, to a real world

plant, the controller as well as a plant simulator are implemented on real-time hardware to build a HIL environment. After passing the tests, the controller is implemented successfully to a heating circuit of an office building.

1.6 Outline

The thesis is structured as follows. Chapter 2 gives basics on tensors, tensor operations and tensor decomposition methods. Special operations for CP decomposed tensors are derived and tensor representations of polynomials and polynomials operations, like multiplication and differentiation are given. The tensor methods are used to represent state space models of MTI systems in decomposed format. Methods like linearization or discretization and a distributed model representation are developed for MTI systems in Chapter 3 in this framework. In Chapter 4 model-based controller synthesis methods are introduced, that use the decomposed structure of MTI models for central and distributed designs. The developed methods for modeling and controller design of MTI systems are applied to different examples in the field of building automation systems in Chapter 5. Efficient model representations are derived. The models are used to design different controllers, which are tested in closed loop simulations. Measurement results of an implementation of a predictive controller with MTI models at a real world heating circuit are provided. The main results of the thesis are summarized in Chapter 6 and an outlook is given.

Readers with different interest may follow different paths through the thesis. Chapter 2 gives a mathematical background. The first sections focus on tensors and tensor calculus, whereas the last Section 2.3 gives the mathematical base for the operations derived for MTI models and design in the following. For the reader, who is interested in modeling, Chapter 3 is the most important, where decomposed MTI model representations, simulation and model operations are introduced, such that Chapter 4 can be skipped and continued with the application to heating systems examples in Section 5.1. An implementation of a model as plant simulator on real-time hardware is described in Section 5.3.5. Another possible reading path focuses on model-based controller design for MTI systems. As MTI models are the base for the design methods, Sections 3.1 and 3.2 should be read first before going to the design methods in Chapter 4. Central design methods for MTI systems are described in Sections 4.2, 4.4 and 4.5, whereas decentralized methods are given in Sections 4.3 and 4.6. Applications for the controller design methods in simulation and in real-time are provided in Section 5.3.

2 Tensor operations

In mathematics, tensor calculus is an active field of research. Tensors are applied in the areas of signal processing, neuroscience, data analysis or machine learning, [19, 31]. A multi-dimensional structure can be found in many data sets as shown in the following example.

Example 2.1 *Consider a dataset of a temperature measurement of one year, that is sampled with one minute time step. In a simple time series notation this data is stored in a vector* $\mathbf{t} \in \mathbb{R}^{525\,600}$*. But it is possible to rearrange the data according to the time structure of minutes, hours and days, resulting in a three dimensional tensor* $\mathsf{T} \in \mathbb{R}^{60\times 24\times 365}$*, that is depicted in Figure 2.1. This is one of many possible tensorizations of the data.*

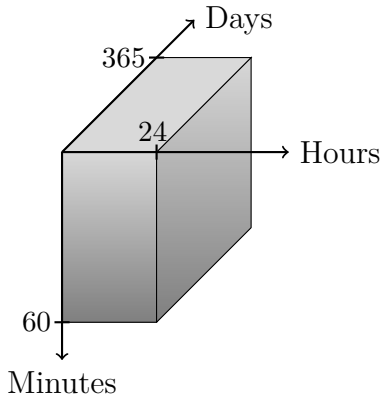


Figure 2.1: Tensor $\mathsf{T} \in \mathbb{R}^{60\times 24\times 365}$ containing one year data with one minute sampling time

The first index denotes the minute, the second index the hour and the third index represents the day. E.g. the temperature of the fifth day at 11:30 AM can be easily accessed by $t(30, 11, 5)$*. Compared to the vector notation this makes it e.g. much easier to get the temperature profile of a certain day. The temperature profile of day* 32 *is accessed by indexing the slice* $\mathbf{T}(:, :, 32)$*. Here the colon notation used like MATLAB provides access to submatrices, see page 14. In the vector notation some index computations would be necessary to extract the same information. If e.g. another year of data is available, the tensor structure is simply extended by adding an additional dimension depicting the year and leading to a four dimensional tensor* $\mathsf{T} \in \mathbb{R}^{60\times 24\times 365\times 2}$*.*

The basic concept of tensors is introduced in Section 2.1. To perform calculations on the data stored in a tensor, different operations are necessary that are shortly described here. When dealing with big data sets tensor calculus offers enormous data compression by tensor decomposition techniques, [31]. The four most common methods, CP, Tucker, TT and HT

decomposition, are shown here. When the operands of tensor operations are available in decomposed form, the operations can be computed very efficiently such that the result is directly given in decomposed form too, which is presented in Section 2.2. Finally, Section 2.3 investigates how tensors can be used to represent polynomial functions by describing them with a parameter tensor and a monomial tensor. In the following chapters these concepts, operations and methods will be used to describe state space models of MTI systems in an efficient way and to design controllers for this special class of systems.

2.1 Tensor basics

Data is often represented by scalar values, vectors or matrices. But in many cases the data has a multidimensional structure. This structure cannot be used when it is described by scalars, vectors or matrices. Therefore, it is desirable to generalize these concepts and store the values in multidimensional arrays, i.e. tensors. The following definitions are necessary for the application of tensors to MTI systems and can be found for example in [18] or [44].

Definition 2.1 (Tensor) *A tensor*

$$\mathsf{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_n}$$

of order n is an n-way array. Elements $x(i_1, i_2, \ldots, i_n)$ are indexed by $i_j \in \{1, 2, \ldots, I_j\}$ in each dimension $j = 1, \ldots, n$.

Example 2.2 *A three dimensional tensor $\mathsf{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ with $I_1 = 6$, $I_2 = 8$ and $I_3 = 5$ is depicted in Figure 2.2, [18].*

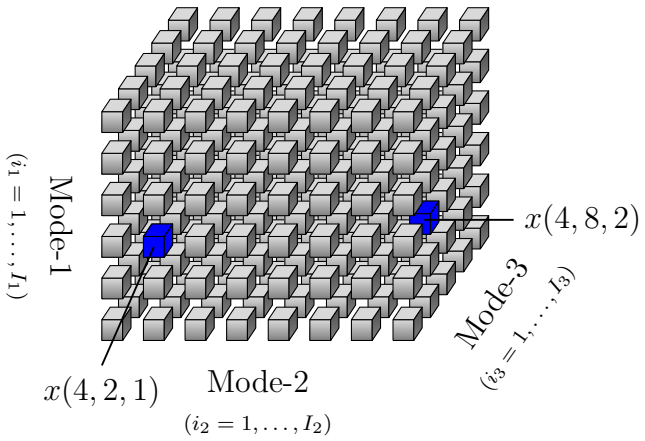


Figure 2.2: Elements of a three dimensional tensor $\mathsf{X} \in \mathbb{R}^{6 \times 8 \times 5}$, [18]

Although this framework has been developed for complex domain $\mathbb{C}$, here only real domains $\mathbb{R}$ are assumed because the numbers stored inside the tensor result from real physical parameters.

Definition 2.2 (Multiple order notation) *The set* $\mathbb{R}^{\overbrace{I \times I \times \cdots \times I}^{N \text{ times}}}$ *where each dimension has the same size* $I_1 = \cdots = I_N = I$ *is alternatively described by* $\mathbb{R}^{\times^{(N)} I}$.

To extract parts of a tensor $\mathsf{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ by fixing part of the indices, a MATLAB or Scilab like notation is used. The resulting tensor is called subtensor of X. A scalar element is selected by an index vector $\mathbf{i} = \begin{pmatrix} i_1 & i_2 & \cdots & i_N \end{pmatrix}$ or by giving the indices directly as in Definition 2.1

$$x(\mathbf{i}) = x(i_1, i_2, \ldots, i_N) \in \mathbb{R}.$$

If all elements of a certain dimension are selected, the symbol ":" is used according to the MATLAB or Scilab notation.

Example 2.3 *A subtensor of a tensor* $\mathsf{X} \in \mathbb{R}^{5 \times 2 \times 4 \times 6 \times 3}$ *is given by* $\mathsf{X}(2, :, 3, :, :)$*, that is of dimension* $\mathbb{R}^{2 \times 6 \times 3}$*. The subtensor contains the second, fourth and fifth dimension of* X.

Definition 2.3 (Tensor fiber and slice) *Fixing all but one index of a tensor results in a one-dimensional subtensor, that is called a tensor fiber. If all indices except for two are fixed, a two dimensional subtensor is selected, i.e. a tensor slice.*

Example 2.4 *Figures 2.3 and 2.4 show tensor fibers and slices of a three dimensional tensor.*

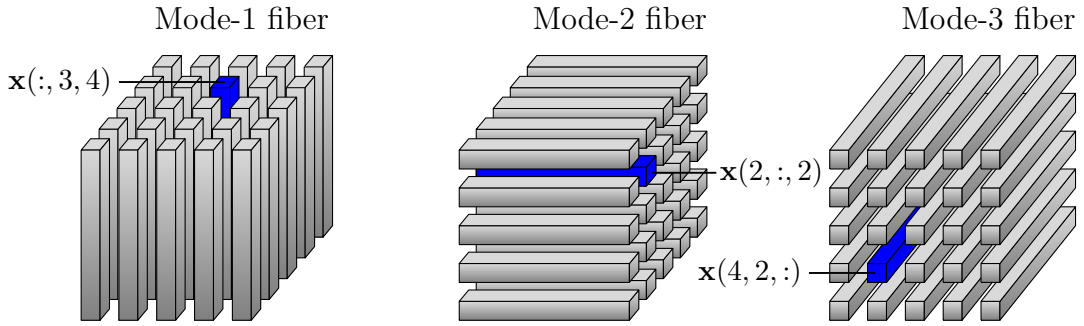


Figure 2.3: Three dimensional tensor X with fibers, [44]

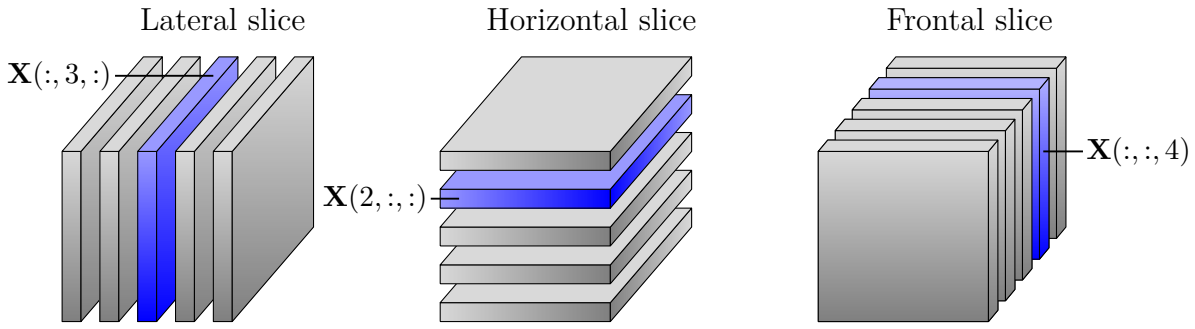


Figure 2.4: Three dimensional tensor X with slices, [44]

Based on the introduced definitions the notations of scalars, matrices and tensors are summarized in Table 2.1.

Table 2.1: Basic notations of elements in the real domain with different dimensions

Scalar	x	$\in$	$\mathbb{R}$
Vector	$\mathbf{x}$	$\in$	$\mathbb{R}^{I_1}$
Vector element	$x(i_1)$	$\in$	$\mathbb{R}$
Matrix	$\mathbf{X}$	$\in$	$\mathbb{R}^{I_1 \times I_2}$
Matrix row	$\mathbf{x}(i_1, :)$	$\in$	$\mathbb{R}^{I_2}$
Matrix column	$\mathbf{x}(:, i_2)$	$\in$	$\mathbb{R}^{I_1}$
Matrix element	$x(i_1, i_2)$	$\in$	$\mathbb{R}$
Tensor	X	$\in$	$\mathbb{R}^{I_1 \times I_2 \times I_3}$
Tensor slice	$\mathbf{X}(:, :, i_3)$	$\in$	$\mathbb{R}^{I_1 \times I_2}$
Tensor fiber	$\mathbf{x}(i_1, :, i_3)$	$\in$	$\mathbb{R}^{I_2}$
Tensor element	$x(i_1, i_2, i_3)$	$\in$	$\mathbb{R}$

For some applications it is useful to rearrange the elements of a tensor in a vector or matrix. This is called unfolding, matricization or vectorization, respectively. The elements of the tensor have to be stored at specific positions in a matrix or a vector. Therefore, it is necessary to map the indices of the tensor to scalar values.

Definition 2.4 (Multi-indices) *All possible combinations of values $i_1, i_2, \ldots, i_N$ of the indices for $i_n = 1, 2, \ldots, I_n,\ n = 1, 2, \ldots, N$ are mapped in a specific order to a scalar value by*

$$i = \overline{i_1 i_2 \cdots i_N} = i_1 + (i_2 - 1)I_1 + (i_3 - 1)I_1 I_2 + \cdots + (i_N - 1)I_1 \cdots I_{N-1}.$$

Different orders are possible. The little-endian convention (reverse lexicographic ordering) known from MATLAB or Scilab is used here, [20].

For a tensor different matricizations are possible, depending on the assignment of the dimensions of the tensor to the rows and columns of the matrix. The following definition describes the mapping of the tensor elements to its positions in the matrix.

Definition 2.5 (Matricization) *Consider an N-dimensional tensor $\mathsf{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and a splitting of its modes into two disjoint sets $\{1, \ldots, N\} = t \cup s$ with $t = \{t_1, \ldots, t_k\}$ and $s = \{s_1, \ldots, s_{N-k}\}$. The matricization*

$$\mathbf{X}^{(t)} \in \mathbb{R} \text{ with } \mathbf{X}^{(t)}(\overline{i_{t_1} \ldots i_{t_k}}, \overline{i_{s_1} \ldots i_{s_{N-k}}}) = x(i_1, \ldots, i_N) \tag{2.1}$$

is constructed by merging the modes contained in t to the row indices and the modes in s to the column indices, [45].

A special case is the mode-n matricization

$$\mathbf{X}^{(n)} \in \mathbb{R}^{I_n \times I_1 I_2 \cdots I_{n-1} I_{n+1} \cdots I_N}$$

where the set $t = \{n\}$ contains one mode only. The indices of this mode are mapped to the rows of the matrix, i.e. all mode-n fibers of the tensor are arranged into the columns of the matrix.

Example 2.5 *Different matricizations of a third order tensor* $\mathsf{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ *are shown in Figure 2.5.*

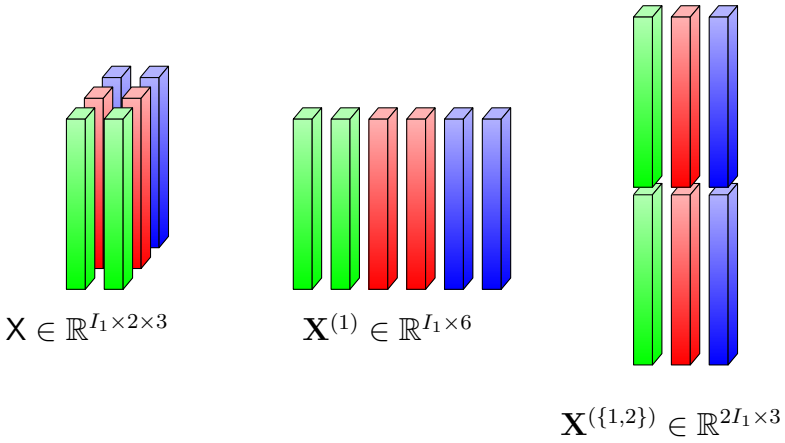


Figure 2.5: Matricizations of a three dimensional tensor X, [45]

Definition 2.6 (Vectorization) *When all indices are mapped to the rows of the matrix, the elements of the tensor are rearranged as vector. This vectorization is denoted by*

$$\mathrm{vec}(\mathsf{X}) = \mathbf{X}^{(\{1,\ldots,N\})} \in \mathbb{R}^{I_1 I_2 \cdots I_N}.$$

The reverse operation, i.e. rearranging the elements of a vector or a matrix in a tensor is called tensorization.

2.1.1 Operations

In linear algebra, operations for multiplication like matrix products or inner porducts are defined. These operations have to be adapted to tensors to deal with the multidimensional structure. Some new operations have to be defined, too. These operations are necessary to perform computations with data in tensor format and to define tensor decomposition methods. Definitions of operations used in this thesis are summarized here and can be found in more detail in [18] or [44]. The notations are more complicated than in the matrix case. But they are also based on standard operations as summation or multiplication.

Definition 2.7 (Kronecker product) *The Kronecker product of two matrices* $\mathbf{X} \in \mathbb{R}^{I \times J}$ *and* $\mathbf{Y} \in \mathbb{R}^{K \times L}$ *is a matrix*

$$\mathbf{Z} = \mathbf{X} \otimes \mathbf{Y} = \begin{pmatrix} x(1,1)\mathbf{Y} & x(1,2)\mathbf{Y} & \cdots & x(1,J)\mathbf{Y} \\ x(2,1)\mathbf{Y} & x(2,2)\mathbf{Y} & \cdots & x(2,J)\mathbf{Y} \\ \vdots & \vdots & \ddots & \vdots \\ x(I,1)\mathbf{Y} & x(I,2)\mathbf{Y} & \cdots & x(I,J)\mathbf{Y} \end{pmatrix}$$

of dimension $\mathbb{R}^{IK \times JL}$.

Definition 2.8 (Outer product) *The outer product*

$$\mathsf{Z} = \mathsf{X} \circ \mathsf{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N \times J_1 \times J_2 \times \cdots \times J_M}, \tag{2.2}$$

of two tensors $\mathsf{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ *and* $\mathsf{Y} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_M}$ *is a tensor of order* $N + M$ *with elements*

$$z(i_1, \ldots, i_N, j_1, \ldots, j_M) = x(i_1, \ldots, i_N) y(j_1, \ldots, j_M). \tag{2.3}$$

As the outer product is associative, it can be applied in sequence denoted by

$$\bigcirc_{i=1}^{N} \mathsf{X}_i = \mathsf{X}_1 \circ \mathsf{X}_2 \circ \cdots \circ \mathsf{X}_N .$$

Example 2.6 *The outer product of two first order tensors* $\mathsf{X} \in \mathbb{R}^I$ *and* $\mathsf{Y} \in \mathbb{R}^J$ *written as vectors is a tensor of order two, which can be unfolded as matrix*

$$\mathsf{X} \circ \mathsf{Y} = \begin{pmatrix} x_1 \\ \vdots \\ x_I \end{pmatrix} \begin{pmatrix} y_1 & \cdots & y_J \end{pmatrix} = \begin{pmatrix} x_1 y_1 & \cdots & x_1 y_J \\ x_2 y_1 & \cdots & x_2 y_J \\ \vdots & \vdots & \vdots \\ x_I y_1 & \cdots & x_I y_J \end{pmatrix} \in \mathbb{R}^{I \times J}.$$

For multiplications of tensors and matrices or vectors a special product is defined that performs the standard matrix-matrix or matrix-vector multiplication with one mode of the tensor.

Definition 2.9 (Mode-k tensor matrix product) *The mode-k tensor matrix product of a tensor* $\mathsf{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ *and a matrix* $\mathbf{W} \in \mathbb{R}^{J \times I_k}$ *is a tensor*

$$\mathsf{Y} = \mathsf{X} \times_k \mathbf{W} \in \mathbb{R}^{I_1 \times \cdots \times I_{k-1} \times J \times I_{k+1} \times \cdots \times I_N}.$$

The resulting tensor is given elementwise by

$$y(i_1, \ldots, i_{k-1}, j, i_{k+1}, \ldots, i_N) = \sum_{i_k=1}^{I_k} x(i_1, \ldots, i_N) w(j, i_k), \ j = 1, \ldots, J,$$

by the multiplication of the fibers of X *with the matrix* $\mathbf{W}$

$$\mathbf{y}(i_1, \ldots, i_{k-1}, :, i_{k+1}, \ldots, i_N) = \mathbf{W}\mathbf{x}(i_1, \ldots, i_{k-1}, :, i_{k+1}, \ldots, i_N).$$

Example 2.7 *The mode-2 product of a three dimensional tensor* X *and a matrix* $\mathbf{W}$ *is illustrated in Figure 2.6. The second mode fibers of the resulting tensor* $\mathsf{Y} = \mathsf{X} \times_2 \mathbf{W}$ *are computed by*

$$\mathbf{y}(i_1, :, i_3) = \mathbf{W}\mathbf{x}(i_1, :, i_3).$$

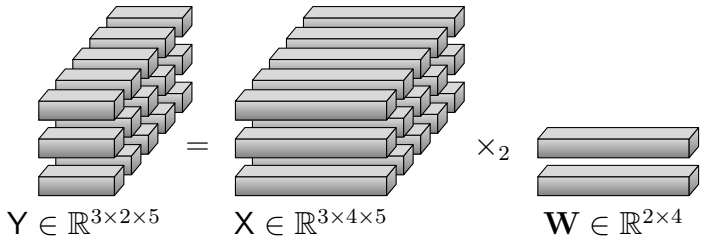


Figure 2.6: Mode-2 tensor matrix product

The mode-k product of a tensor and a vector gives a tensor of dimension $\mathbb{R}^{I_1 \times \cdots \times I_{k-1} \times 1 \times I_{k+1} \times I_N}$, where the singleton dimension can be removed. The result is a tensor with $(N-1)$ dimensions $\mathbb{R}^{I_1 \times \cdots \times I_{k-1} \times I_{k+1} \times I_N}$.

Definition 2.10 (Mode-k tensor vector product) *The product of a* N^{th} *order tensor* $\mathsf{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ *and a vector* $\mathbf{w} \in \mathbb{R}^{I_k}$ *in the* k^{th} *mode is a tensor*

$$\mathsf{Y} = \mathsf{X} \bar{\times}_k \mathbf{w} \in \mathbb{R}^{I_1 \times \cdots \times I_{k-1} \times I_{k+1} \times \cdots \times I_N}$$

of order $N-1$*. The elements of the resulting tensor are computed by multiplying the mode-k fiber of* X *by* $\mathbf{w}$

$$\begin{aligned} y(i_1, \ldots, i_{k-1}, i_{k+1}, \ldots, i_N) &= \mathbf{w}^T \mathbf{x}(i_1, \ldots, i_{k-1}, :, i_{k+1}, \ldots, i_N) \\ &= \sum_{i_k=1}^{I_k} x(i_1, \ldots, i_N) w(i_k). \end{aligned}$$

For the representation of the right hand sides of MTI systems in tensor format the contracted product of two tensors is very important.

Definition 2.11 (Contracted product) *The contracted product along the first* N *dimensions of two tensors* $\mathsf{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N \times I_{N+1} \times \cdots \times I_{N+M}}$ *and* $\mathsf{Y} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$

$$\begin{aligned} z(k_1, \ldots, k_m) &= \langle \mathsf{X} \mid \mathsf{Y} \rangle_{1,\ldots,N;1,\ldots,N} (k_1, \ldots, k_m) \\ &= \sum_{i_1=1}^{I_1} \cdots \sum_{i_n=1}^{I_n} x(i_1, \ldots, i_n, k_1, \ldots, k_m) y(i_1 \ldots i_n), \end{aligned} \tag{2.4}$$

with $k_i \in \{1, 2, \ldots, I_{N+i}\}$*,* $i = 1, \ldots, M$*, is a tensor* Z *of dimension* $\mathbb{R}^{I_{N+1} \times \cdots \times I_{N+M}}$ *.*

Example 2.8 *The contracted product of a tensor* $\mathsf{X} \in \mathbb{R}^{I_1 \times I_2 \times 3}$ *and a matrix* $\mathbf{Y} \in \mathbb{R}^{I_1 \times I_2}$ *is depicted in Figure 2.7. The elements in* $z(k)$, $k = 1, 2, 3$, *are computed by multiplying a slice* $\mathbf{X}(:, :, k)$ *of* X *elementwise with* $\mathbf{Y}$ *and sum up all the elements of the result*

$$z(k) = \langle \mathsf{X} \mid \mathbf{Y} \rangle_{1:2;1:2}(k) = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} x(i_1, i_2, k) y(i_1, i_2).$$

The colors in Figure 2.7 show which slice of X *belongs to which element in* $\mathbf{z}$.

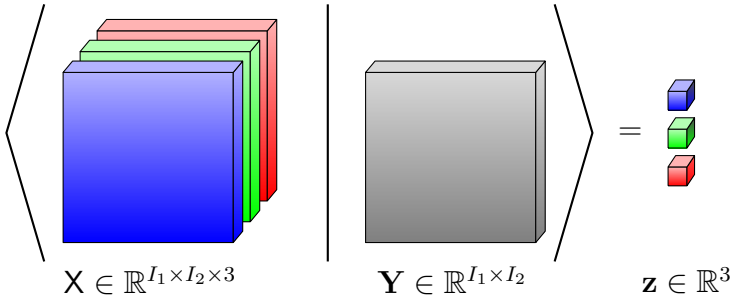


Figure 2.7: Contracted product of a three dimensional tensor X and a matrix $\mathbf{Y}$

The contracted product can be computed along arbitrary dimensions of both tensors. But it is necessary that the corresponding dimensions have the same sizes.

Example 2.9 *The contracted product of two tensors* $\mathsf{X} \in \mathbb{R}^{5 \times 2 \times 3}$ *and* $\mathsf{Y} \in \mathbb{R}^{2 \times 4 \times 3 \times 5}$ *along the modes* $1, 2$ *and* $4, 1$ *respectively is a tensor* $\langle \mathsf{X} \mid \mathsf{Y} \rangle_{1,2;4,1} \in \mathbb{R}^{3 \times 4 \times 3}$.

For ease of notation the subscript will be neglected in the following, if the contracted product is computed along the first N same modes of two tenors

$$\langle \mathsf{X} \mid \mathsf{Y} \rangle_{1,\ldots,N;1,\ldots,N} = \langle \mathsf{X} \mid \mathsf{Y} \rangle,$$

as in (2.4). If two tensors $\mathsf{X}, \mathsf{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ have the same sizes of each dimension, their contracted product along all modes is called inner product

$$\langle \mathsf{X} \mid \mathsf{Y} \rangle = \sum_{i_1=1}^{I_1} \cdots \sum_{i_N=1}^{I_N} x(i_1, \ldots, i_N) y(i_1, \ldots, i_N) \in \mathbb{R}.$$

and the result is a scalar. To simplify the notation the contracted product of two tensors in the last mode of the first tensor and the first mode of the second tensors gets a special symbol, [59].

Definition 2.12 (Mode-$(M, 1)$ contracted product) *The mode-(M,1) contracted product of two tensors* $\mathsf{X} \in \mathbb{R}^{I_1 \times \cdots \times I_M}$ *and* $\mathsf{Y} \in \mathbb{R}^{I_M \times J_2 \times \cdots \times J_N}$ *is a tensor*

$$\mathsf{Z} = \langle \mathsf{X} \mid \mathsf{Y} \rangle_{M;1} = \mathsf{X} \bullet \mathsf{Y} \in \mathbb{R}^{I_1 \times \cdots \times I_{M-1} \times J_2 \times \cdots \times J_N},$$

with elements

$$z(i_1, \ldots, i_{M-1}, j_2, \ldots, j_N) = \sum_{i_M=1}^{I_M} x(i_1, \ldots, i_M) y(i_M, j_2, \ldots, j_N).$$

Two tensors can be merged in different modes by operations like the concatenation.

Definition 2.13 (Concatenation) *Consider two N^{th} order tensors $\mathsf{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ as well as $\mathsf{Y} \in \mathbb{R}^{J_1 \times \cdots \times J_N}$ with same number of elements in each dimension except the k^{th}, i.e. $I_m = J_m$ for all $m \neq k$. The concatenation along mode-k is a tensor*

$$\mathsf{Z} = \mathsf{X} \boxplus_k \mathsf{Y} \in \mathbb{R}^{I_1 \times \cdots \times I_{k-1} \times (I_k + J_k) \times I_{k+1} \times \cdots \times I_N}, \tag{2.5}$$

where the fibers of Z are composed by the fibers of X and Y

$$\mathbf{z}(i_1, \ldots, i_{k-1}, :, i_{k+1}, \ldots, i_N) = \begin{pmatrix} \mathbf{x}(i_1, \ldots, i_{k-1}, :, i_{k+1}, \ldots, i_N) \\ \mathbf{y}(i_1, \ldots, i_{k-1}, :, i_{k+1}, \ldots, i_N) \end{pmatrix}. \tag{2.6}$$

Example 2.10 *The concatenation of two 3^{rd} order tensors $\mathsf{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ and $\mathsf{Y} \in \mathbb{R}^{J_1 \times J_2 \times J_3}$ with $I_1 = J_1$ and $I_3 = J_3$ in the second mode is a tensor*

$$\mathsf{Z} = \mathsf{X} \boxplus_2 \mathsf{Y} \in \mathbb{R}^{I_1 \times (I_2 + J_2) \times I_3}$$

and depicted in Figure 2.8.

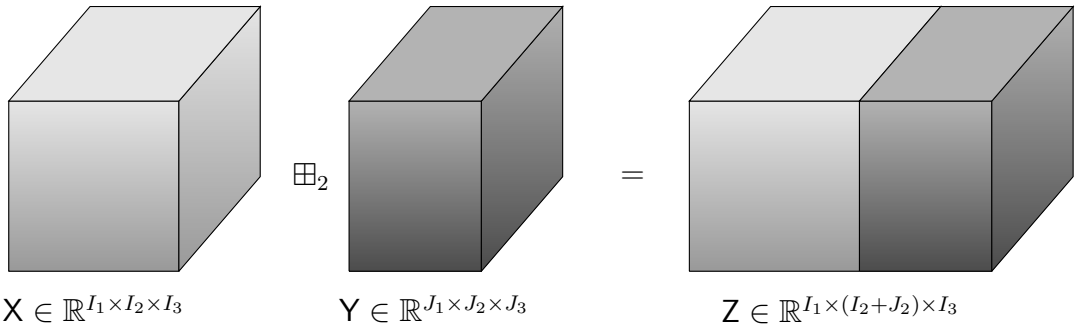


Figure 2.8: Concatenation of two tensors in the second mode

If the concatenation operation is applied in series to concatenate several tensors $\mathsf{X}_1, \ldots, \mathsf{X}_n$ in the same dimension, this is abbreviated by

$$\mathsf{Z} = \mathsf{X}_1 \boxplus_k \mathsf{X}_2 \boxplus_k \cdots \boxplus_k \mathsf{X}_n = \bigboxplus_{i=1}^{n}{}_k \mathsf{X}_i.$$

Another way to connect two tensors is the direct sum that is introduced in the following.

Definition 2.14 (Direct sum) *The direct sum of two N^{th} order tensors X and Y of dimensions $\mathbb{R}^{I_1 \times \cdots \times I_N}$ and $\mathbb{R}^{J_1 \times \cdots \times J_N}$, respectively is a tensor*

$$\mathsf{Z} = \mathsf{X} \oplus \mathsf{Y} \in \mathbb{R}^{(I_1 + J_1) \times \cdots \times (I_N + J_N)}, \tag{2.7}$$

where the elements are given by

$$z(k_1, \ldots, k_N) = \begin{cases} x(k_1, \ldots, k_N) & \text{, for } 1 \leq k_n \leq I_n, \ \forall n = 1, \ldots, N, \\ y(k_1 - I_1, \ldots, k_N - I_N) & \text{, for } I_n < k_n \leq I_n + J_n, \ \forall n = 1, \ldots, N, \\ 0 & \text{, else.} \end{cases} \tag{2.8}$$

Example 2.11 *Consider two three dimensional tensors* $\mathsf{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ *and* $\mathsf{Y} \in \mathbb{R}^{J_1 \times J_2 \times J_3}$. *The direct sum of both tensors results in a tensor*

$$\mathsf{Z} = \mathsf{X} \oplus \mathsf{Y} \in \mathbb{R}^{(I_1+J_1)\times(I_2+J_2)\times(I_3+J_3)},$$

that is depicted in Figure 2.9.

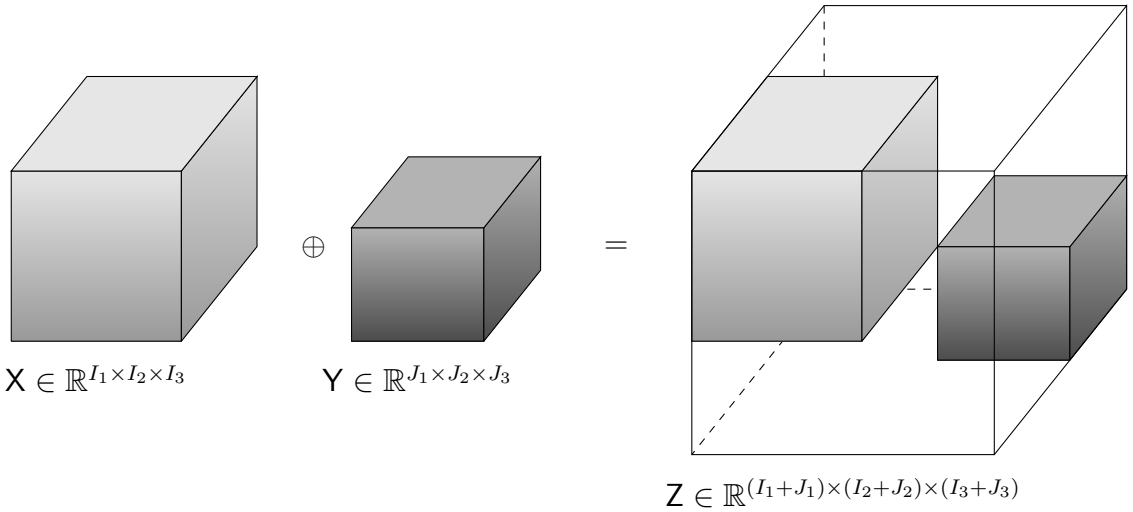


Figure 2.9: Direct sum of two three dimensional tensors X and Y

2.1.2 Decomposition methods

For tensors with a large number of dimensions the memory demand increases immensely, since the number of elements to be stored in a tensor rises exponentially with the number of dimensions. For a tensor K of dimension $\mathbb{R}^{\times^{(N)}I}$,

$$\xi_{Full}(\mathsf{K}) = I^N$$

elements have to be stored. The memory demand is measured by the number of scalar elements, that have to be stored for the tensor representation, [17].

Example 2.12 *For a* 20^{th} *order tensor with* $I = 5$ *entries in each dimension, the storage amount leads to approximately* $\xi_{Full}(\mathsf{K}) \approx 9 \cdot 10^{13}$ *values to be stored.*

To reduce the memory effort, linear algebra provides several methods for matrices, like SVD to approximate the data by low-rank representations, [14]. In the last decades several decomposition techniques were developed in the field of mathematics to reduce the storage amount of tensors. Four of the most common methods are CP, Tucker, TT and HT decomposition, [31]. Therefore these techniques are introduced here for the decomposition of a N-dimensional tensor $\mathsf{K} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and will be used in Section 3 to represent state space models of MTI systems. More details on the decompositions can be found in the referenced literature, e.g. [20, 21, 31].

Canonic polyadic decomposition

In CP decomposition a tensor is factorized to a sum of rank-1 components, [36].

Definition 2.15 (Rank-1 tensor) *A N^{th} order tensor $\mathsf{K} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is a rank-1 tensor, if it can be computed by the outer product of N vectors $\mathbf{x}_i \in \mathbb{R}^{I_i}$*

$$\mathsf{X} = \mathbf{x}_1 \circ \mathbf{x}_2 \circ \cdots \circ \mathbf{x}_N = \bigcirc_{i=1}^{N} \mathbf{x}_i. \tag{2.9}$$

Example 2.13 *A 3^{rd} order rank-1 tensor $\mathsf{K} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ is depicted in Figure 2.10 as outer product of three vectors $\mathbf{x}_1 \in \mathbb{R}^{I_1}$, $\mathbf{x}_2 \in \mathbb{R}^{I_2}$ and $\mathbf{x}_3 \in \mathbb{R}^{I_3}$.*

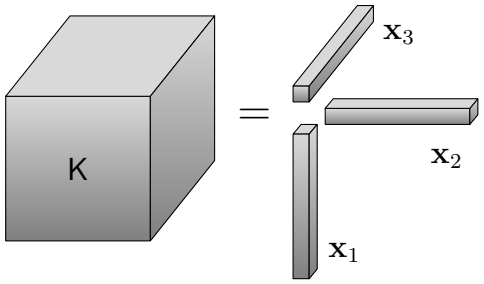


Figure 2.10: Rank-1 tensor

Definition 2.16 (CP tensor) *A CP tensor of dimension $I_1 \times \cdots \times I_N$ is given by*

$$\mathsf{K} = [\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N] \cdot \boldsymbol{\lambda} = \sum_{l=1}^{r_{cp}(\mathsf{K})} \lambda(l) \mathbf{x}_1(:, l) \circ \cdots \circ \mathbf{x}_N(:, l), \tag{2.10}$$

where elements are computed by the sums of the outer products of the column vectors of factor matrices $\mathbf{X}_i \in \mathbb{R}^{I_i \times r_{cp}(\mathsf{K})}$, weighted by the elements of the weighting (or parameter) vector $\boldsymbol{\lambda} \in \mathbb{R}^{r_{cp}(\mathsf{K})}$, [36]. The l^{th} column of the matrix $\mathbf{X}_i$ is denoted by $\mathbf{x}_i(:, l)$. An element of the tensor K is given by

$$k(i_1, \ldots, i_n) = \sum_{l=1}^{r_{cp}(\mathsf{K})} \lambda(l) \cdot x_1(i_1, l) \cdots x_n(i_n, l), \tag{2.11}$$

with the entries $\lambda(l)$ and $x_j(i_j, l),\ j = 1, \ldots, n,\ l = 1, \ldots, r_{cp}(\mathsf{K})$ of the weighting vector and the factor matrices, respectively.

Example 2.14 *A 3^{rd} order CP tensor is written as*

$$\mathsf{K} = [\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3] \cdot \boldsymbol{\lambda} = \sum_{l=1}^{r_{cp}(\mathsf{K})} \lambda(l) \mathbf{x}_1(:, l) \circ \mathbf{x}_2(:, l) \circ \mathbf{x}_3(:, l).$$

Figure 2.11 shows the tensor as the sum of outer products of the column vectors $\mathbf{X}_i(:, l)$ of the factor matrices. The tensor K is constructed by the sum of $r_{cp}(\mathsf{K})$ rank-1 elements.

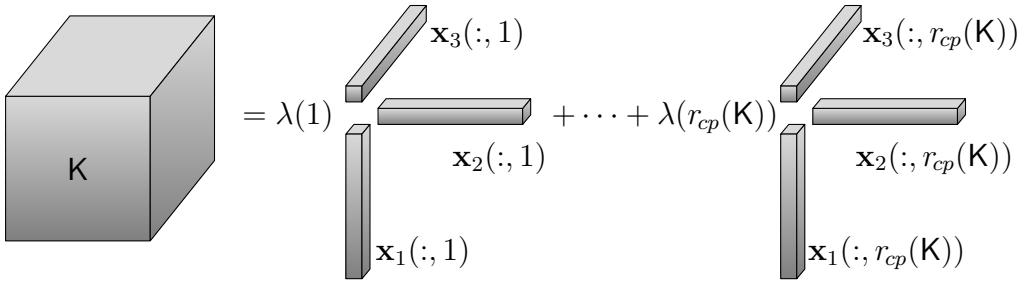


Figure 2.11: Third order CP tensor

Weighting the different outer products by a factor $\lambda(l)$ allows to normalize the column vectors $\mathbf{X}_i(:, l)$ of the factor matrices to length of one, which is useful in some applications.

Definition 2.17 (CP rank of a tensor) *The minimal number of rank-1 tensors, that are summed up in (2.10) to get* K *exactly is defined as CP rank of the tensor, [19]. An exact decompostion with a minimal number of rank-1 elements is called rank decomposition, [44].*

Focusing on the storage effort in CP decomposition, the number of elements to be stored does not depend exponentially on the number of dimensions N anymore, as it is in the full case. Each rank-1 component contains of $\sum_{i=1}^{N} I_i$ values. To represent the tensor K, $r_{cp}(\mathsf{K})$ rank-1 elements are summed up. Together with the weighting vector of length $r_{cp}(\mathsf{K})$ the storage effort is given by

$$\xi_{cp}(\mathsf{K}) = r_{cp}(\mathsf{K}) \left(1 + \sum_{i=1}^{N} I_i\right), \tag{2.12}$$

which depends only linear on the number of dimensions and sizes of each dimension. This leads to an enormous reduction in storage complexity, especially for big data as shown in the following example.

Example 2.15 *For the* 20^{th} *order tensor with* $I = 5$ *entries in each dimension of Example 2.12, a CP decomposition leads to* 100 *elements to be stored for each rank-1 component. Therefore a rank-3 approximation results in* $\xi_{cp}(\mathsf{K}) = 303$ *values to be stored. Thus, if a representation with a low number of rank-1 terms can be found, an immense reduction of storage demand is achieved, compared to* $\xi_{Full}(\mathsf{K}) = 9 \cdot 10^{13}$ *elements in the full tensor representation.*

The CP representation of a tensor can be computed by finding an approximation of the original tensor by a CP tensor of a fixed rank $r_{cp}(\mathsf{K})$. This is called low-rank approximation or rank-$r_{cp}(\mathsf{K})$ approximation, [18]. The CP decomposition of a tensor $\mathsf{K} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is computed by solving the optimization problem

$$\min_{\boldsymbol{\lambda}, \mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N} \left\| \mathsf{K} - [\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N] \cdot \boldsymbol{\lambda} \right\|_F, \tag{2.13}$$

with fixed rank $r_{cp}(\mathsf{K})$, where $\|\cdot\|_F$ denotes the Frobenius norm of a tensor, [14, 44]. The problem of finding the best low-rank approximation is ill-posed, [31]. Computing the rank of a tensor is an NP-hard problem, [19]. Since the optimization problem is not convex, the solution depends on the initial value for the factor matrices. Thus, the standard procedure is to compute several decompositions with different ranks and initial values until an approximation is found that fulfills the desired accuracy. Choosing the rank of the CP approximation is a trade-off between the accuracy of the approximation and the number of parameters to be stored. There are several different algorithms available to solve the approximation problem (2.13). Very common is the alternating least squares (ALS) algorithm, [44].

Finding a CP represenation of a tensor is not advantegous for the storing effort only. The decomposed structure can be used to compute operations as introduced in Section 2.1.1 very efficiently. Once the operands are given as decomposed tensors, it does not make sense to construct the full tensor and to compute the operation in the full format. The operations should be defined on the decomposition factors only such that the result is in decomposed representation, too. For CP tensors some of these operations are defined and implemented in [5] like the mode-k tensor matrix product.

Proposition 2.1 (Mode-k tensor matrix product in CP form) *The mode-k tensor matrix product*

$$\mathsf{Y} = \mathsf{X} \times_k \mathbf{W}$$

of a N^{th} order CP tensor

$$\mathsf{X} = [\mathbf{U}_1, \ldots, \mathbf{U}_N] \cdot \boldsymbol{\lambda}_{\mathsf{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$$

and a matrix $\mathbf{W} \in \mathbb{R}^{J \times I_k}$ is a CP tensor

$$\mathsf{Y} = [\mathbf{V}_1, \ldots, \mathbf{V}_N] \cdot \boldsymbol{\lambda}_{\mathsf{Y}}$$

with weighting vector and factor matrices given by, [5]

$$\boldsymbol{\lambda}_{\mathsf{Y}} = \boldsymbol{\lambda}_{\mathsf{X}}, \tag{2.14}$$

$$\mathbf{V}_i = \begin{cases} \mathbf{W}\mathbf{U}_i & , \textit{for } i = k, \\ \mathbf{U}_i & , \textit{else.} \end{cases} \tag{2.15}$$

This shows that no full tensor is computed. The CP factors of the result are computed, by using the factors of the operands. Other operations especially introduced for CP tensors, that are not available in the literature but required for MTI systems, are derived in Section 2.2.

Tucker decomposition

The second decomposition method considered is the Tucker decomposition, where the tensor is decomposed to N factor matrices and a core tensor, [18].

Definition 2.18 (Tucker) *A Tucker tensor is given by*

$$\mathsf{K} = [\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N] \cdot \mathsf{\Lambda}, \tag{2.16}$$

with core tensor $\mathsf{\Lambda} \in \mathbb{R}^{r_{t,1}(\mathsf{K}) \times \cdots \times r_{t,N}(\mathsf{K})}$ *with ranks* $r_{t,i}(\mathsf{K})$ *and factor matrices* $\mathbf{X}_i \in \mathbb{R}^{I_i \times r_{t,i}(\mathsf{K})}$. *The full tensor is created from the Tucker representation by multiplying the core tensor by one factor matrix along each mode*

$$\mathsf{K} = \mathsf{\Lambda} \times_1 \mathbf{X}_1 \times_2 \mathbf{X}_2 \times_3 \cdots \times_N \mathbf{X}_N = \sum_{j_1=1}^{r_{t,1}(\mathsf{K})} \cdots \sum_{j_N=1}^{r_{t,N}(\mathsf{K})} \lambda(j_1, \ldots, j_N) \mathbf{x}(:, j_1) \circ \cdots \circ \mathbf{x}(:, j_N).$$

An element of K is computed by using the factor matrices and core tensor by

$$k(i_1, \ldots, i_N) = \sum_{j_1=1}^{r_{t,1}(\mathsf{K})} \cdots \sum_{j_N=1}^{r_{t,N}(\mathsf{K})} \lambda(j_1, \ldots, j_N) \mathbf{x}(i_1, j_1) \cdots \mathbf{x}(i_N, j_N),$$

with $i_l = 1, \ldots, I_l \; \forall l = 1, \ldots, N$, *[18].*

Example 2.16 *A* 3^{rd} *order Tucker tensor reads*

$$\mathsf{K} = [\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3] \cdot \mathsf{\Lambda} = \mathsf{\Lambda} \times_1 \mathbf{X}_1 \times_2 \mathbf{X}_2 \times_3 \mathbf{X}_3.$$

The Tucker tensor is computed by tensor matrix products of the core tensor and one factor matrix in each dimension as shown in Figure 2.12.

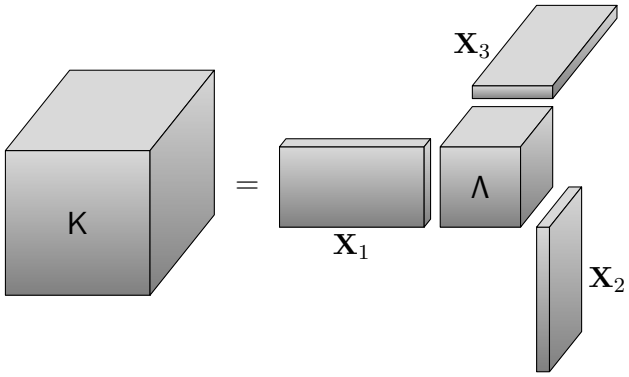


Figure 2.12: Third order Tucker tensor, [63]

Definition 2.19 (Multilinear rank) *The N-tuple* $(r_{t,1}(\mathsf{K}), r_{t,2}(\mathsf{K}), \ldots, r_{t,N}(\mathsf{K}))$ *is called the multilinear rank of* K *if the Tucker decomposition (2.16) holds exactly. The ranks for an exact representation are computed by*

$$r_{t,i}(\mathsf{K}) = rank(\mathbf{K}^{(i)}), \; i = 1, \ldots, N, \tag{2.17}$$

where $rank(\mathbf{K}^{(i)})$ *is the matrix rank of the mode-i matricization of* K, *[44].*

The existence of different tensor ranks, like the rank of a CP tensor (2.10) of Definition 2.17 and the multilinear rank (2.17) in Definition 2.19 is a big difference to the matrix case with its unique rank definition.
The storage reduction for Tucker tensors can be achieved by reducing the size of the core tensor, i.e. by setting $r_{t,i}(\mathsf{K}) < I_i,\ i = 1, \ldots, N$. The storage demand is composed of the size of the core tensor and the factor matrices

$$\xi_t(\mathsf{K}) = \prod_{i=1}^{N} r_{t,i}(\mathsf{K}) + \sum_{i=1}^{N} r_{t,i}(\mathsf{K}) I_i. \tag{2.18}$$

Since the storage effort still depends exponentially on the number of dimensions, it is necessary to find a small core tensor for a memory efficient representation. Since the number of dimensions is equal to the number of dimensions of the original tensor, it is desirable to get low ranks $r_{t,i}(\mathsf{K})$.

Example 2.17 *For the 20^{th} dimensional tensor $\mathsf{K} \in \mathbb{R}^{\times^{20}5}$ of Example 2.12 it is hard to find a Tucker tensor of low memory demand, because of the high number of dimensions. For a core tensor with multilinear rank of $(3, \ldots, 3)$ still more than $3 \cdot 10^9$ elements have to be stored. A reshape of the tensor to get a lower number of dimensions might help in this case.*

Tucker representations of tensors are often computed by using the higher-order SVD, which is an extension of the matrix SVD to the multidimensional case, [55]. Therefore, truncated SVDs of the matricizations $\mathbf{K}^{(\mathbf{i})}$ are computed to get low-rank approximations of (2.17). This yields a sub-optimal approximation. Based on that, iterative algorithms like ALS are used to refine the approximation with fixed multilinear rank $(r_{t,1}(\mathsf{K}), r_{t,2}(\mathsf{K}), \ldots, r_{t,N}(\mathsf{K}))$ by solving the optimization problem, [44]

$$\min_{\Lambda, \mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n} \left\| \mathsf{K} - [\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n] \cdot \Lambda \right\|_F. \tag{2.19}$$

Operations like the tensor matrix product as described in Section 2.1.1 are implemented in [5] based on the Tucker structure.

Tensor Train decomposition

Different to the methods introduced before, in an TT decomposition, the N-dimensional tensor is approximated by a product of N third order tensors, [83]. A TT has the advantage of an SVD based truncation like the Tucker decomposition but avoids the exponential growth in memory demand.

Definition 2.20 (Tensor Train) *A tensor K in TT format*

$$\mathsf{K} = [\mathsf{G}_1, \mathsf{G}_2, \ldots, \mathsf{G}_{N-1}, \mathsf{G}_N], \tag{2.20}$$

with three dimensional TT-core tensors $\mathsf{G}_j \in \mathbb{R}^{r_{tt,j-1}(\mathsf{K}) \times I_j \times r_{tt,j}(\mathsf{K})},\ j = 1, \ldots, N$ and TT-ranks $r_{tt,j}(\mathsf{K})$ is given by

$$\mathsf{K} = \mathsf{G}_1 \bullet \mathsf{G}_2 \bullet \cdots \bullet \mathsf{G}_N.$$

By construction, the first and the last ranks are equal to one $r_{tt,0}(\mathsf{K}) = r_{tt,N}(\mathsf{K}) = 1$*, such that the first and the last core element can be represented by matrices. An element of the tensor is computed by*

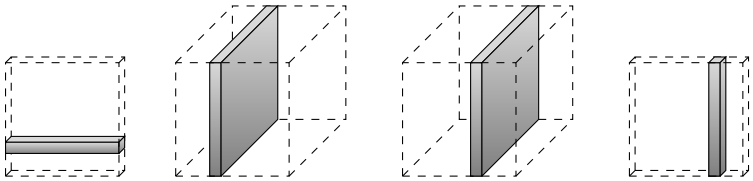
$$\begin{aligned} k(i_1, \ldots, i_N) &= \mathbf{g}_1(1, i_1, :)\mathbf{G}_2(:, i_2, :) \cdots \mathbf{G}_{N-1}(:, i_{N-1}, :)\mathbf{g}_N(:, i_N, 1) \\ &= \sum_{j_1=1}^{r_{tt,1}(\mathsf{K})} \cdots \sum_{j_N=1}^{r_{tt,N}(\mathsf{K})} g_1(1, i_1, j_1) g_2(j_1, i_2, j_2) \cdots g_{N-1}(j_{N-2}, i_{N-1}, j_{N-1}) g_N(j_{N-1}, i_N, 1), \end{aligned} \tag{2.21}$$

where $\mathbf{G}_j(:, i_j, :) \in \mathbb{R}^{R_{j-1} \times R_j}$ *are slices of the core tensors* G_j *[83].*

Example 2.18 *The elementwise description of a fourth order tensor in TT format gives*

$$k(i_1, i_2, i_3, i_4) = \mathbf{g}_1(1, i_1, :)\mathbf{G}_2(:, i_2, :)\mathbf{G}_3(:, i_3, :)\mathbf{g}_4(:, i_4, 1).$$

The cores are given by two tensors G_1 *and* G_4 *with singleton dimensions, that can be interpreted as matrices and two three dimensional tensors* G_2, G_3*. The elements of* K *are computed by the product of a column vector, two matrices and a row vector as given in Figure 2.13.*



$k(i_1, i_2, i_3, i_4) = \mathbf{g}_1(1, i_1, :) \cdot \quad \mathbf{G}_2(:, i_2, :) \cdot \qquad \mathbf{G}_3(:, i_3, :) \cdot \qquad \mathbf{g}_4(:, i_4, 1)$

Figure 2.13: Fourth order Tensor Train

Definition 2.21 (TT rank) *The TT rank, written as* N*-tuple* $(r_{tt,1}(\mathsf{K}), \ldots, r_{tt,N}(\mathsf{K}))$*, is determined for an exact representation by computing the ranks of matricizations* $\mathbf{K}^{(\{1,\ldots,i\})}$ *with* $i = 1, \ldots, N-1$ *of the original tensor*

$$r_{tt,i}(\mathsf{K}) = \operatorname{rank}(\mathbf{K}^{(\{1,\ldots,i\})}).$$

The storage demand of the TT representation depends on the ranks, but not in an exponential way as in the Tucker decomposition. For a TT tensor the elements of the three dimensional TT cores have to be stored, leading to

$$\xi_{tt}(\mathsf{K}) = \sum_{i=1}^{N} r_{tt,i-1}(\mathsf{K}) I_i r_{tt,i}(\mathsf{K})$$

elements to be stored.

Example 2.19 *Considering the tensor from Example 2.12 with* $N = 20$ *and* $I = 5$ *this leads to* 885 *elements in TT representation for TT ranks of* $r_{tt,0}(\mathsf{K}) = r_{tt,20}(\mathsf{K}) = 1$ *and* $r_{tt,i}(\mathsf{K}) = 3$*,* $i = 1, \ldots, 19$*, which is a reduction by* 10 *orders of magnitude compared to the full tensor.*

To find low-rank approximations in TT representation, SVD based algorithms are used. The SVDs of the matricizations $\mathbf{K}^{(\{1,\dots,i\})},\ i = 1,\dots,N$ of the tensor are truncated, such that less significant singular values are neglected. This makes it possible to determine an upper bound of the approximation error

$$\|\mathsf{K} - [\mathsf{G}_1, \mathsf{G}_2, \dots, \mathsf{G}_{N-1}, \mathsf{G}_N]\|_F \leq \epsilon \|\mathsf{K}\|_F , \tag{2.22}$$

where ϵ is the desired accuracy of the decomposition, [83]. The ranks are determined by the SVD based algorithm, such that the desired accuracy is met. This is an advantage e.g. compared with CP, where several decompositions with different ranks have to be computed to check, which number of rank-1 components gives a sufficient accuracy. The computation of many tensor operations with operands given as TT, were introduced in [59] and [60].

Hierarchical Tucker decomposition

The HT decomposition is an alternative way to reduce the memory demand of the Tucker decomposition and generalizes the TT decomposition, [31]. The method is based on a splitting of the modes of the tensor K resulting in a binary tree $\mathcal{T}$. Each node contains a subset of the modes $t \subset \{1, 2, \dots, N\}$ as shown in Figure 2.14 for a 6$^{\text{th}}$ order tensor.

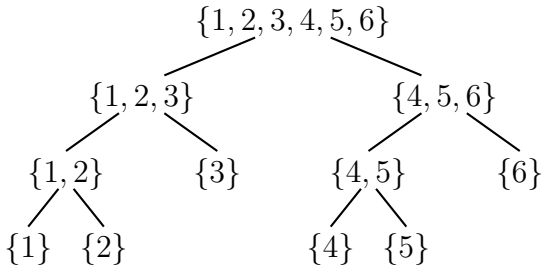


Figure 2.14: Tree of a 6$^{\text{th}}$ order tensor

Here balanced trees are considered with the additional assumption that left children nodes contain always elements that are smaller than any element in the right children nodes, [45]. According to this splitting of the modes, matricizations $\mathbf{K}^{(t)}$ of tensor K are determined. Base matrices $\mathbf{U}_t$ are computed, such that their columns span the image of the matricization $\mathbf{K}^{(t)}$ for each $t \in \mathcal{T}$. Thus, the base matrices have

$$r_{ht,t}(\mathsf{K}) = \operatorname{rank}(\mathbf{K}^{(t)})$$

columns for an exact representation. For each parent node there exists a so-called transfer matrix $\mathbf{B}_t \in \mathbb{R}^{r_{ht,t_l}(\mathsf{K}) r_{ht,t_r}(\mathsf{K}) \times r_{ht,t}(\mathsf{K})}$, such that the base matrix $\mathbf{U}_t$ is computed by its left $\mathbf{U}_{t_l}$ and right $\mathbf{U}_{t_r}$ children

$$\mathbf{U}_t = (\mathbf{U}_{t_r} \otimes \mathbf{U}_{t_l})\, \mathbf{B}_t, \tag{2.23}$$

with ranks of the corresponding matricizations $r_{ht,t}(\mathsf{K})$, $r_{ht,t_l}(\mathsf{K})$, $r_{ht,t_r}(\mathsf{K})$ and $t = t_l \cup t_r$ as well as $t_l \cap t_r = \emptyset$. Figure 2.15 shows the considered subtree of a parent node with its left and right children.

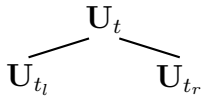


Figure 2.15: Parent node with left and right children

For some applications it is advantageous to reshape the transfer matrices

$$\mathbf{B}_t \in \mathbb{R}^{r_{ht,t_l}(\mathsf{K}) r_{ht,t_r}(\mathsf{K}) \times r_{ht,t}(\mathsf{K})} \Rightarrow \mathsf{B}_t \in \mathbb{R}^{r_{ht,t_l}(\mathsf{K}) \times r_{ht,t_r}(\mathsf{K}) \times r_{ht,t}(\mathsf{K})} \ \forall t \in \mathcal{T}.$$

to three dimensional transfer tensors. The parent node of the subtree depicted in Figure 2.15 is computed from the child nodes equivalently to the matrix case (2.23) by

$$\mathbf{u}_t(:,q) = \sum_{i=1}^{r_{ht,t_l}(\mathsf{K})} \sum_{j=1}^{r_{ht,t_r}(\mathsf{K})} \left(\mathbf{u}_{t_r}(:,j) \otimes \mathbf{u}_{t_l}(:,i)\right) b_t(i,j,q), \ q = 1, \ldots, r_{ht,t}(\mathsf{K}). \tag{2.24}$$

This simplifies the notation in some case, [45]. In the definition of the HT decomposition the matrix notation will be used here. But later for the representation of MTI systems the tensor notation leads to a simpler formulation of the factors. Thus, the splitting as shown in Figure 2.14 describes a hierarchy of matrices $\mathbf{U}_t$. Because of (2.23) not all base matrices $(\mathbf{U}_t)_{t\in\mathcal{T}}$ have to be stored explicitly, since $\mathbf{U}_t$ of a parent node can be computed by its left $\mathbf{U}_{t_l}$ and right $\mathbf{U}_{t_r}$ children. The recursive application of (2.23) leads to the HT decomposition. Thus, the base matrices $\mathbf{U}_t \in \mathbb{R}^{I_t \times r_{ht,t}(\mathsf{K})}$ of the leaf nodes $t = \{1\}, \ldots, \{N\}$ and the transfer matrices $\mathbf{B}_t$ for all other nodes have to be stored only, to reconstruct the entries of the full tensor. For the six dimensional tensor the components to be stored are depicted in Figure 2.16 in a tree structure.

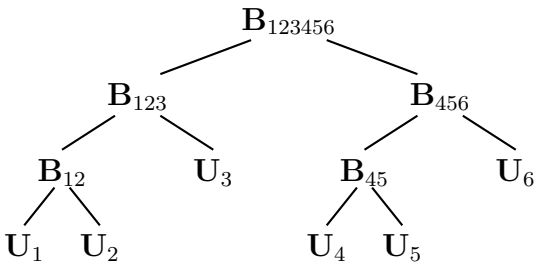


Figure 2.16: Tree of a 6^{th} order tensor in HT format

Example 2.20 *A tree of a third order HT tensor is shown in Figure 2.17.*

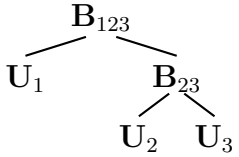


Figure 2.17: Tree of a 3^{rd} order tensor in HT format

The recursive application of (2.23) leads to the vectorized reconstruction of K

$$\begin{aligned} vec(\mathsf{K}) = \mathsf{K}^{(\{1,2,3\})} &= (\mathbf{U}_{23} \otimes \mathbf{U}_1)\, \mathbf{B}_{123} = (((\mathbf{U}_3 \otimes \mathbf{U}_2)\, \mathbf{B}_{23}) \otimes \mathbf{U}_1)\, \mathbf{B}_{123} \\ &= (\mathbf{U}_3 \otimes \mathbf{U}_2 \otimes \mathbf{U}_1)\, (\mathbf{B}_{23} \otimes \mathbf{I})\, \mathbf{B}_{123}, \end{aligned}$$

where $\mathbf{I}$ *denotes an identity matrix of suitable dimensions.*

Definition 2.22 (HT rank) *The tuple* $(r_{tt,t}(\mathsf{K}))_{t \in \mathcal{T}}$ *is defined as the HT rank of the tensor, [45].*

The number of elements for the storage requirement of the HT representation is

$$\xi_{ht}(\mathsf{K}) = \sum_{i=1}^{N} I_i r_{ht,\{i\}}(\mathsf{K}) + \sum_{t \in \mathcal{T} \setminus \{1\},\ldots,\{N\}} r_{ht,t}(\mathsf{K}) r_{ht,t_l}(\mathsf{K}) r_{ht,t_r}(\mathsf{K}),$$

resulting from the base matrices of the leaves and the transfer matrices of the other nodes.

Example 2.21 *The tensor given in Example 2.12 of order* 20 *and* $I = 5$ *leads to a storage requirement of* 795 *elements, with the assumption* $r_{tt,t}(\mathsf{K}) = 3\ \forall t \in \mathcal{T}$ *on the HT-rank.*

The memory demand depends cubic on the HT-rank. Thus, it is desirable to find low-rank approximations here, too. The HT-rank is computed by SVDs of the matricizations $\mathbf{K}^{(t)}\ \forall t \in \mathcal{T}$. For low-rank approximations small singular values are neglected. With that it is possible to compute a decomposition of desired accuracy as for the TT. The computation of operations like the mode-k tensor matrix product considering the HT tensor structure are given in [45].

Comparison of the complexity

In full representation a N^{th} order tensor suffers from the curse of dimensionality, because the number of elements scales exponentially with the order. This makes storage and computation unmanageable for large N. The introduced decomposition techniques should help to overcome this curse of dimensionality. The storage complexities of the decompositions are summarized in Table 2.2.

Table 2.2: Storage complexity of different tensor representations, [17]

Representation	**Storage complexity** $\xi_i(\mathsf{K})$
Full	$\mathcal{O}(I^N)$
CP	$\mathcal{O}(NIR + R)$
Tucker	$\mathcal{O}(NIR + R^N)$
TT	$\mathcal{O}(NIR^2)$
HT	$\mathcal{O}(NIR + (N-2)R^3 + R^2)$

The storage complexity is measured by the number of elements that have to be stored. To compare the decompositions maximum values for the sizes of the dimensions and ranks

$$I = \max(I_1, I_2, \ldots, I_N),$$
$$R = \begin{cases} \max(r_{i,1}(\mathsf{K}), \ldots, r_{i,N}(\mathsf{K})) & \text{for } i = \text{Tucker, TT, HT} \\ r_{cp}(\mathsf{K}) & \text{for CP} \end{cases}$$

were chosen, to get a worst case estimate of the complexity, [17]. With the Tucker decomposition, the number of elements can be reduced for $R \ll I$, but for large N Tucker tensors still suffer from the curse of dimensionality, because of the exponential growth with N. This can be bypassed with TT and HT decomposed tensors, with their linear dependence on the number of dimensions, only. Here the rank has a squared or cubic influence on the complexity, which makes it desirable to find good approximations of low ranks. CP tensors allow a very memory efficient representation, since the number of elements scales linearly with the number of dimensions as well as with the rank.
In this investigation of the complexity it seems as if the number of dimensions and the rank scale independently. But e.g. a linear dependence on N holds only for the assumption that R stays constant. In many applications, for larger N a higher rank is necessary either to get a good approximation result, such that R has to be increased. Thus, N and R are not completely independent, which has to be considered when interpreting Table 2.2.

Example 2.22 *The general results for the storage complexity are depicted in Table 2.2. In the previously introduced decomposition techniques, the storage demands for low-rank approximations of a high dimensional tensor* $\mathsf{K} \in \mathbb{R}^{\times(20)5}$ *was investigated. The results of Examples 2.15, 2.17, 2.19 and 2.21 are summarized in Table 2.3. The storage complexities match with the general results of Table 2.2. The table gives some numbers for the complexity to emphasize the enormous compression rates that are possible with the different decomposition techniques when low-rank approximations are found.*

Table 2.3: Storage complexity of different tensor representations of a tensor $\mathsf{K} \in \mathbb{R}^{\times(20)5}$ with ranks of 3 in the particular representations

Representation	**Storage complexity** $\xi_i(\mathsf{K})$
Full	$9 \cdot 10^{13}$
CP	303
Tucker	$3 \cdot 10^{9}$
TT	885
HT	795

2.1.3 Guiding questions

The previous sections showed, how tensors are used to store data. In contrast to the matrix representation of data, the number of dimensions is not limited to two. Also higher orders

are possible, which allows to capture multidimensional structures in the data. It was presented, how decomposition methods allow an efficient data representation with an enormous reduction in storage effort. The tensor framework is used in many application areas, [31]. In many of those applications data, e.g. from measurements, is stored in tensors. After collecting the data in a tensor suitable decompositions are searched. A multi-index structure can be found often as shown in Example 2.1. This is just one example for the benefits of the tensor notation. Using decomposition techniques offers the possibility to overcome the curse of dimensionality, when working with large datasets. In contrast to the established approach to store measurement data in decomposed tensors, here the representations of polynomials and state space models in tensor notation is focused on.

In [63] and [85] CP tensors were used to describe state space models, where the right hand sides are given by multilinear functions. The question now is, how different decomposition techniques can be used to store the model information and to find memory efficient representations. As decomposed tensors offer the possibility to deal with large measurement datasets or so called big data, the question arises if it is also possible to represent large-scale systems with state space models in decomposed tensor notation. Since four decomposition methods are proposed, it has to be determined, which one fits best and has benefits in the application to multilinear systems.

Not only the model representation is of interest, but also the system analysis and controller synthesis. Thus, computations with tensors are necessary, when the state space models are given in this framework. Several operations are defined e.g. in [18] or [44]. Once the tensor is available in decomposed form, no full tensor should be constructed again, even in intermediate computation steps. This leads to the question, if all operations necessary for analysis and design are defined for decomposed tensors. This means, if the operands are given in decomposed format, the operations should work with the decomposition factors only without building the full tensor again. The result should be given as decomposed tensor, too.

Since [63] and [85] described the multilinear functions in tensor format, the third question is, if this concept can be extended to arbitrary polynomials and if it is possible to compute standard polynomial operations in the tensor framework, to use them for analysis and controller design for MTI systems.

2.2 Operations for canonical polyadic (CP) tensors

As introduced in Section 2.1.1 many arithmetic operations for tensors are available. Tensors can be represented in a very efficient way with respect to the memory demand by using decomposition techniques. This efficient representation should be used for the computation of the arithmetic operations if the operands are given in a decomposed way as well. It does not make sense to rebuild the full representation of the tensors again to evaluate the operation. The decomposition factors of the operands are used only to compute the decomposition factors of the result, such that the result is given as decomposed tensor directly. Many of these operations are already introduced in the literature [44, 45, 59] and implemented in the different toolboxes for decomposed tensors, [5, 45, 84, 109]. But some of the operations

necessary here for calculations in the field of MTI systems are not stated for CP tensors yet. In the following Section the definitions for the outer and contracted product as well as the concatenation are given, for the case that the operands are given in CP format.

2.2.1 Outer product

In Definition 2.8 the general form of the outer product was introduced. If both operands are given in a CP structure the following Proposition shows, how to compute the outer product, such that the result is also in CP.

Proposition 2.2 (Outer product in CP form) *The outer product* $\mathsf{Z} = \mathsf{X} \circ \mathsf{Y}$ *of two CP tensors*

$$\mathsf{X} = [\mathbf{U}_1, \dots, \mathbf{U}_n] \cdot \boldsymbol{\lambda}_X \in \mathbb{R}^{I_1 \times \cdots \times I_n}, \tag{2.25}$$

$$\mathsf{Y} = [\mathbf{V}_1, \dots, \mathbf{V}_m] \cdot \boldsymbol{\lambda}_Y \in \mathbb{R}^{J_1 \times \cdots \times J_m}, \tag{2.26}$$

with $r_{cp}(\mathsf{X})$ *and* $r_{cp}(\mathsf{Y})$ *rank-1 components respectively results in a tensor*

$$\mathsf{Z} = [\mathbf{W}_1, \dots, \mathbf{W}_{n+m}] \cdot \boldsymbol{\lambda}_Z \in \mathbb{R}^{I_1 \times \cdots \times I_n \times J_1 \times \cdots \times J_m}, \tag{2.27}$$

that can be represented in a CP format. The factor matrices and weighting vector are given by

$$\mathbf{W}_i = \mathbf{U}_i \otimes \mathbf{1}^T_{r_{cp}(\mathsf{Y})} \in \mathbb{R}^{I_i \times r_{cp}(\mathsf{X}) r_{cp}(\mathsf{Y})}, \ \forall\, i = 1, \dots, n, \tag{2.28}$$

$$\mathbf{W}_{n+i} = \mathbf{1}^T_{r_{cp}(\mathsf{X})} \otimes \mathbf{V}_i \in \mathbb{R}^{J_i \times r_{cp}(\mathsf{X}) r_{cp}(\mathsf{Y})}, \ \forall\, i = 1, \dots, m, \tag{2.29}$$

$$\boldsymbol{\lambda}_Z = \boldsymbol{\lambda}_X \otimes \boldsymbol{\lambda}_Y, \tag{2.30}$$

with $\mathbf{1}_k \in \mathbb{R}^k$ *denoting a column vector full of ones of length* k*. Thus, the CP format of* Z *follows directly from the factors of* X *and* Y*. The number of rank-1 components of the resulting tensor* Z *is* $r_{cp}(\mathsf{Z}) = r_{cp}(\mathsf{X}) r_{cp}(\mathsf{Y})$*.*

The proof of this proposition can be found in the appendix A.1.

Remark 2.1 *Although the CP decomposition (2.27) is the exact result of the outer product of (2.25) and (2.26), it is possible that a CP decomposition with a lower number of rank-1 elements* $r_{cp}(\mathsf{Z}) < r_{cp}(\mathsf{X}) r_{cp}(\mathsf{Y})$ *exist. Finding this memory saving representation is an NP-hard problem as discussed in Section 2.1.2, [44].*

Example 2.23 *The outer product of two tensors in CP representation is illustrated in this example. Figure 2.18 shows the CP representation and the structure of the factor matrices of the three dimensional tensor* $\mathsf{X} = [\mathbf{U}_1, \mathbf{U}_2, \mathbf{U}_3] \cdot \boldsymbol{\lambda}_X \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ *with* $r_{cp}(\mathsf{X}) = 2$ *rank-1 components.*

The CP form of the second operand $\mathsf{Y} = [\mathbf{V}_1, \mathbf{V}_2] \cdot \boldsymbol{\lambda}_Y \in \mathbb{R}^{J_1 \times J_2}$ *of order two and* $r_{cp}(\mathsf{Y}) = 2$ *rank-1 components is depicted in Figure 2.19.*

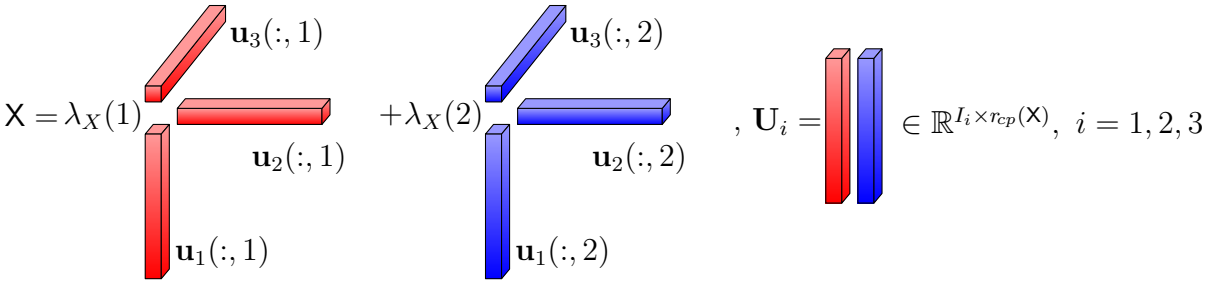


Figure 2.18: CP representation and structure of the factor matrices of X

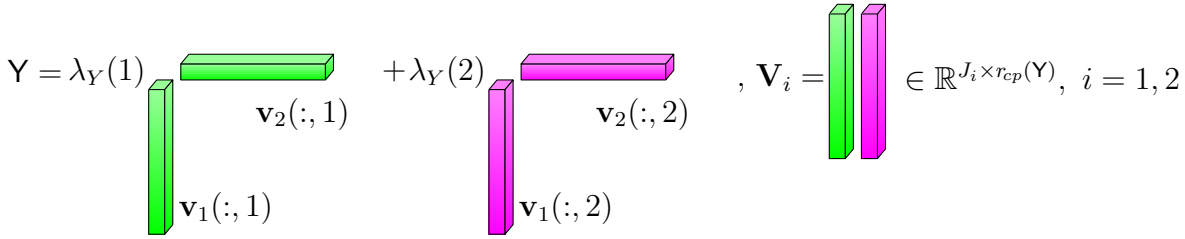


Figure 2.19: CP representation and structure of the factor matrices of Y

The factors of the resulting tensor $\mathsf{Z} = \mathsf{X} \circ \mathsf{Y} = [\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3, \mathbf{W}_4, \mathbf{W}_5] \cdot \boldsymbol{\lambda}_Z$ *of dimension* $\mathbb{R}^{I_1 \times I_2 \times I_3 \times J_1 \times J_2}$ *are written as*

$$\begin{aligned}
\boldsymbol{\lambda}_Z &= \begin{pmatrix} \lambda_X(1)\lambda_Y(1) & \lambda_X(1)\lambda_Y(2) & \lambda_X(2)\lambda_Y(1) & \lambda_X(2)\lambda_Y(2) \end{pmatrix}, \\
\mathbf{W}_r &= \begin{pmatrix} \mathbf{u}_r(:,1) & \mathbf{u}_r(:,1) & \mathbf{u}_r(:,2) & \mathbf{u}_r(:,2) \end{pmatrix}, \ r = 1,2,3, \\
\mathbf{W}_{3+r} &= \begin{pmatrix} \mathbf{v}_r(:,1) & \mathbf{v}_r(:,2) & \mathbf{v}_r(:,1) & \mathbf{v}_r(:,2) \end{pmatrix}, \ r = 1,2,
\end{aligned}$$

according to (2.28) to (2.30). The structures of the factor matrices of Z *are illustrated in Figure 2.20. The figure shows how the matrices* $\mathbf{W}_j, \ j = 1, \ldots, 5$ *are constructed using column vectors of the factor matrices of* X *and* Y *as they are depicted in Figures 2.18 and 2.19.*

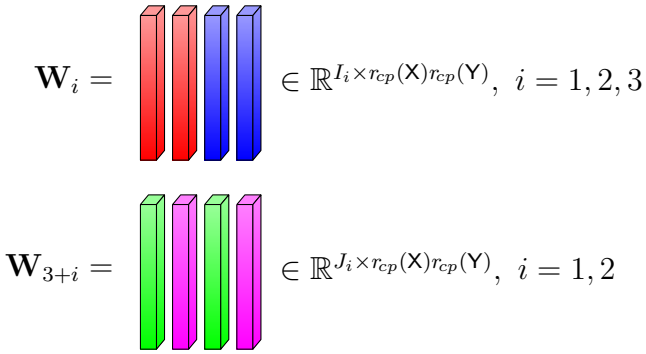


Figure 2.20: Structure of the CP factor matrices of $\mathsf{Z} = \mathsf{X} \circ \mathsf{Y}$

2.2.2 Contracted product

For many calculations in the field of MTI systems, the evaluation of the contracted product is necessary. When the operand tensors are given in CP format, the contracted product is evaluated efficiently on the decomposition factors.

Proposition 2.3 (Contracted product in CP format) *Consider two CP tensors*

$$\mathsf{X} = [\mathbf{U}_1, \ldots, \mathbf{U}_P, \mathbf{U}_{P+1}, \ldots, \mathbf{U}_{P+N}] \cdot \boldsymbol{\lambda}_X \in \mathbb{R}^{I_1 \times \cdots \times I_P \times J_1 \times \cdots \times J_N}, \tag{2.31}$$

$$\mathsf{Y} = [\mathbf{V}_1, \ldots, \mathbf{V}_P, \mathbf{V}_{P+1}, \ldots, \mathbf{V}_{P+M}] \cdot \boldsymbol{\lambda}_Y \in \mathbb{R}^{I_1 \times \cdots \times I_P \times K_1 \times \cdots \times K_M}, \tag{2.32}$$

with same sizes in the first P dimensions and $r_{cp}(\mathsf{X})$ and $r_{cp}(\mathsf{Y})$ rank-1 components, respectively. The contracted product along the first P dimensions results in a tensor

$$\mathsf{Z} = \langle \mathsf{X} \,|\, \mathsf{Y} \rangle \in \mathbb{R}^{J_1 \times \cdots \times J_N \times K_1 \times \cdots \times K_M}.$$

The CP representation of Z is computed using the factors of X (2.31) and Y (2.32)

$$\mathsf{Z} = [\mathbf{W}_1, \ldots, \mathbf{W}_N, \mathbf{W}_{N+1}, \ldots, \mathbf{W}_{N+M}] \cdot \boldsymbol{\lambda}_Z \in \mathbb{R}^{J_1 \times \cdots \times J_N \times K_1 \times \cdots \times K_M},$$

where the factor matrices and weighting vector are given by

$$\boldsymbol{\lambda}_Z = \boldsymbol{\lambda}_X \otimes \boldsymbol{\lambda}_Y \circledast \operatorname{vec}\left(\left(\mathbf{U}_1^T \mathbf{V}_1\right) \circledast \cdots \circledast \left(\mathbf{U}_P^T \mathbf{V}_P\right)\right), \tag{2.33}$$

$$\mathbf{W}_i = \mathbf{U}_{P+i} \otimes \mathbf{1}_{r_{cp}(\mathsf{Y})}^T, \; i = 1, \ldots, N, \tag{2.34}$$

$$\mathbf{W}_{N+i} = \mathbf{1}_{r_{cp}(\mathsf{X})}^T \otimes \mathbf{V}_{P+i}, \; i = 1, \ldots, M. \tag{2.35}$$

The proof of the proposition is given in the appendix A.2.

Example 2.24 *Consider two tensors in CP representation $\mathsf{X} = [\mathbf{U}_1, \mathbf{U}_2, \mathbf{U}_3] \cdot \boldsymbol{\lambda}_X$ of dimension $\mathbb{R}^{I_1 \times I_2 \times I_3}$ and $\mathsf{Y} = [\mathbf{V}_1, \mathbf{V}_2] \cdot \boldsymbol{\lambda}_Y \in \mathbb{R}^{J_1 \times J_2}$ with $r_{cp}(\mathsf{X}) = 2$ rank-1 components. Their structures are depicted in Figures 2.18 and 2.19 in Example 2.23. To compute the contracted product*

$$\mathsf{Z} = \langle \mathsf{X} \,|\, \mathsf{Y} \rangle,$$

along the first dimension, it has to be assumed, that the first dimensions of X and Y have the same sizes $I_1 = J_1$. Using (2.33) to (2.35) the parameter matrices and the weighting vector of the result $\mathsf{Z} = [\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3] \cdot \boldsymbol{\lambda}_Z \in \mathbb{R}^{I_2 \times I_3 \times J_2}$ are given by

$$\begin{aligned}
\mathbf{W}_i &= \begin{pmatrix} \mathbf{u}_{i+1}(:,1) & \mathbf{u}_{i+1}(:,1) & \mathbf{u}_{i+1}(:,2) & \mathbf{u}_{i+1}(:,2) \end{pmatrix}, \; i = 1, 2, \\
\mathbf{W}_3 &= \begin{pmatrix} \mathbf{v}_2(:,1) & \mathbf{v}_2(:,2) & \mathbf{v}_2(:,1) & \mathbf{v}_2(:,2) \end{pmatrix}, \\
\lambda_Z &= \Big(\; \lambda_X(1)\lambda_Y(1)\mathbf{u}_1(:,1)^T\mathbf{v}_1(:,1) \quad \lambda_X(1)\lambda_Y(2)\mathbf{u}_1(:,1)^T\mathbf{v}_1(:,2) \quad \cdots \\
&\qquad \lambda_X(2)\lambda_Y(1)\mathbf{u}_1(:,2)^T\mathbf{v}_1(:,1) \quad \lambda_X(2)\lambda_Y(2)\mathbf{u}_1(:,2)^T\mathbf{v}_1(:,2) \; \Big).
\end{aligned}$$

Figure 2.21 illustrates how the factor matrices of the result Z are composed of the factor matrices of the operands. The rearrangement of the columns of the factor matrices of X and Y to get the CP representation of Z is indicated by the different colors, that were introduced for the factors X and Y in the Figures 2.18 and 2.19.

$$\mathbf{W}_i = \quad \in \mathbb{R}^{I_{i+1} \times r_{cp}(\mathsf{X}) r c p Y}, \; i = 1, 2$$

$$\mathbf{W}_3 = \quad \in \mathbb{R}^{J_2 \times r_{cp}(\mathsf{X}) r_{cp}(\mathsf{Y})}$$

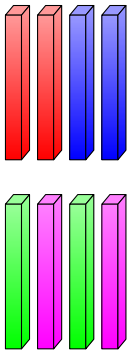


Figure 2.21: Structure of the factor matrices of the result $\mathsf{Z} = \langle \mathsf{X} \,|\, \mathsf{Y} \rangle$

2.2.3 Tensor concatenation

The concatenation of two tensors is described for general tensors in Definition 2.13. If the operands are given in CP form, the resulting tensor can be computed as a CP tensor directly.

Proposition 2.4 (Concatenation of tensors in CP format) *The concatenation in dimension k of two tensors*

$$\mathsf{X} = [\mathbf{U}_1, \ldots, \mathbf{U}_N] \cdot \boldsymbol{\lambda}_X \in \mathbb{R}^{I_1 \times \cdots \times I_N},$$
$$\mathsf{Y} = [\mathbf{V}_1, \ldots, \mathbf{V}_N] \cdot \boldsymbol{\lambda}_Y \in \mathbb{R}^{J_1 \times \cdots \times J_N},$$

with $I_m = J_m, \forall m \neq k$ yields in a tensor

$$\mathsf{Z} = \mathsf{X} \boxplus_k \mathsf{Y} \in \mathbb{R}^{I_1 \times \cdots \times (I_k + J_k) \times \cdots \times I_N}.$$

The CP representation of Z is given by

$$\mathsf{Z} = [\mathbf{W}_1, \ldots, \mathbf{W}_N] \cdot \boldsymbol{\lambda}_Z,$$

with decomposition factors

$$\mathsf{W}_m = \mathbf{U}_m \boxplus_2 \mathbf{V}_m, \; \forall m = 1, \ldots, N, \; m \neq k, \tag{2.36}$$
$$\mathsf{W}_k = \mathbf{U}_k \oplus \mathbf{V}_k, \tag{2.37}$$
$$\boldsymbol{\lambda}_Z = \boldsymbol{\lambda}_X \boxplus_1 \boldsymbol{\lambda}_Y. \tag{2.38}$$

The prof of the proposition is shown in the Appendix A.3.

Example 2.25 *Consider two three dimensional tensors $\mathsf{X} = [\mathbf{U}_1, \mathbf{U}_2, \mathbf{U}_3] \cdot \boldsymbol{\lambda}_X \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ and $\mathsf{Y} = [\mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3] \cdot \boldsymbol{\lambda}_Y \in \mathbb{R}^{J_1 \times J_2 \times J_3}$ in CP representation, where each tensor has two rank-1 components. The first and the third dimension of the two tensors have the same sizes $I_1 = J_1$ and $I_3 = J_3$. The concatenation in the second dimension gives a tensor*

$$\mathsf{Z} = \mathsf{X} \boxplus_2 \mathsf{Y} = [\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3] \cdot \boldsymbol{\lambda}_Z$$

of dimension $\mathbb{R}^{I_1 \times (I_2+J_2) \times I_3}$. *The resulting tensor is constructed by using the factor matrices of the operands. Applying (2.36) to (2.38) gives the CP factors of the resulting tensor* Z

$$\begin{aligned} \mathbf{W}_m &= \mathbf{U}_m \boxplus_2 \mathbf{V}_m, \ m = 1, 3, \\ \mathbf{W}_2 &= \mathbf{U}_2 \oplus \mathbf{V}_2, \\ \boldsymbol{\lambda}_Z &= \boldsymbol{\lambda}_X \boxplus_1 \boldsymbol{\lambda}_Y. \end{aligned}$$

This shows that the factor matrices of the result are built by concatenation and direct sum of the factor matrices of X *and* Y. *Figure 2.22 shows the block structure of the factor matrices of* Z *to clarify how they are constructed. The factor matrices of* X *and* Y *are highlighted in blue and red respectively. Zero matrices of suitable dimensions* $\mathbf{0}_{I_2 \times 2}$ *or* $\mathbf{0}_{J_2 \times 2}$ *are indicated in white.*

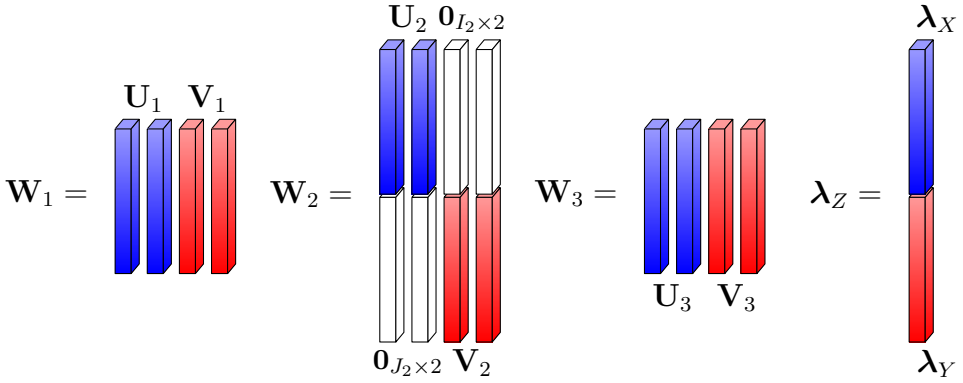


Figure 2.22: Structure of the factor matrices of $\mathsf{Z} = \mathsf{X} \boxplus_2 \mathsf{Y}$

*These factor matrices lead to the CP representation illustrated in Figure 2.23. The figure shows nicely, that the first two rank-*1 *components belong to the first tensor* X. *The third and fourth rank-*1 *components contain the factors belonging to* Y. *Because of the concatenation in the second mode, the vectors of the rank-*1 *terms of the second modes are partly zero depending on their relation to* X *or* Y.

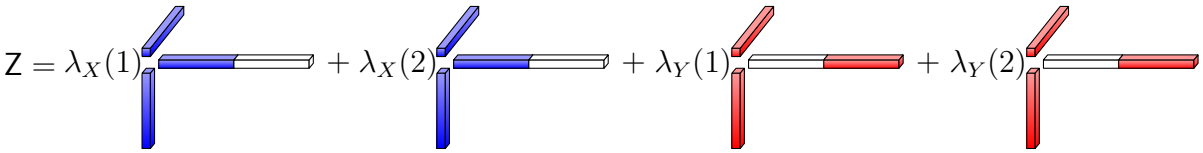


Figure 2.23: CP representation of Z

2.3 Polynomial calculus by operational tensors

The usage of tensors for representations of multilinear polynomials and state space models of multilinear systems were described in [63] and [85]. In the following Section 2.3.1 it will

be shown, how the tensor framework can be used to represent polynomials in a structured way. The tensors used here do not contain measurement information or sampled function evaluations as in many applications in the literature, [31]. In contrast to that, parameters and monomials of the polynomials are stored by tensors. After introducing the tensor representation of polynomials, Sections 2.3.2 to 2.3.5 describe several polynomial operations in this representation like multiplication or differentiation. An analytically exact result is computed by simple numerical computations without symbolic calculations.

2.3.1 Tensor form of polynomials

The tensor notation of polynomials is based on the tensor representation of multilinear functions that was introduced in [63] and [85] for the description of state space models of MTI systems. To derive the representation of arbitrary polynomials a brief summary of the description of multilinear polynomials, which are a subclass of general polynomials, is given first. The concept for multilinear functions is extended to the general polynomial case.
Before introducing multilinear functions, the meaning of term *multilinear* has to be clarified for this work. Hackbusch in [32] defined a mapping $h : \mathcal{X} \to \mathcal{H}$ with vector spaces $\mathcal{X}_i,\ i = 1, \ldots, n$ of each variable. The overall vector space of the variables is given by

$$\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n.$$

The mapping is called multilinear, if the vector space $\mathcal{H}$ is linear in each argument, i.e. it fulfills the condition

$$h(x_1, \ldots, a \cdot \tilde{x}_j + b \cdot \hat{x}_j, \ldots, x_n) = a \cdot h(x_1, \ldots, \tilde{x}_j, \ldots, x_n) + b \cdot h(x_1, \ldots, \hat{x}_j, \ldots, x_n), \quad (2.39)$$

for all $x_i \in \mathcal{X}_i,\ \tilde{x}_j, \hat{x}_j \in \mathcal{X}_j,\ 1 \leq j \leq n$ and coefficients $a, b \in \mathbb{R}$. Next, coordinates $\mathbf{x} \in \mathcal{X}$ are defined to describe multilinear functions. In other words the multilinearity condition (2.39) means, that the mapping is linear if all but one variable is held constant. This allows multiplications of variables but no squares or higher order exponents in a multilinear function.

Definition 2.23 (Multilinear function) *A multilinear function* $h : \mathbb{R}^n \to \mathbb{R}$

$$h(\mathbf{x}) = \boldsymbol{\alpha}^T \mathbf{m}(\mathbf{x}) \quad (2.40)$$

with variables $\mathbf{x} \in \mathbb{R}^n$*, coefficient vector* $\boldsymbol{\alpha} = \begin{pmatrix} \alpha_1 & \cdots & \alpha_{2^n} \end{pmatrix}^T \in \mathbb{R}^{2^n}$ *and monomial vector*

$$\mathbf{m}(\mathbf{x}) = \begin{pmatrix} 1 \\ x_n \end{pmatrix} \otimes \cdots \otimes \begin{pmatrix} 1 \\ x_1 \end{pmatrix} \in \mathbb{R}^{2^n} \quad (2.41)$$

is a polynomial in n variables, which is linear, when all but one variable are held constant.

Example 2.26 *A multilinear function with 2 variables* x_1 *and* x_2 *is given by*

$$h(x_1, x_2) = \boldsymbol{\alpha}^T\ \mathbf{m}(x_1, x_2) = \begin{pmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 \end{pmatrix} \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ x_1 x_2 \end{pmatrix}$$
$$= \alpha_1 \cdot 1 + \alpha_2 \cdot x_1 + \alpha_3 \cdot x_2 + \alpha_4 \cdot x_1 x_2.$$

Fixing one variable, e.g. $x_2 = 2$ *leads to a linear function in* x_1

$$h(x_1, 2) = (\alpha_1 + 2\alpha_3) + (\alpha_2 + 2\alpha_4)\, x_1.$$

This representation leads to a separation of the parameters of the function and the variables. The monomial vector contains all the multiplicative combinations of the variables that are allowed in the multilinear class of functions. Instead of the vector representation of the monomials with the Kronecker product, the multilinear monomials can be computed by an outer product and thus arranged in a tensor framework.

Definition 2.24 (Monomial tensor for multilinear polynomials) *The multilinear monomials of a function in* n *variables* $\mathbf{x} \in \mathbb{R}^n$ *are computed by the sequence of outer products*

$$\mathsf{M}(\mathbf{x}) = \begin{pmatrix} 1 \\ x_n \end{pmatrix} \circ \begin{pmatrix} 1 \\ x_{n-1} \end{pmatrix} \circ \cdots \circ \begin{pmatrix} 1 \\ x_1 \end{pmatrix} = \bigcirc_{i=1}^{n} \begin{pmatrix} 1 \\ x_{n-i+1} \end{pmatrix}, \tag{2.42}$$

leading to a tensor of dimension $\mathbb{R}^{\times^{(n)}2}$*. By construction, the monomial tensor is a rank-*1 *tensor*

$$\mathsf{M}(\mathbf{x}) = \left[\begin{pmatrix} 1 \\ x_n \end{pmatrix}, \ldots, \begin{pmatrix} 1 \\ x_1 \end{pmatrix} \right], \tag{2.43}$$

because it is computed by the outer product of vectors $\begin{pmatrix} 1 & x_i \end{pmatrix}^T$ *as in (2.9).*

Example 2.27 *Figure 2.24 shows how the multilinear monomials of a function in three variables* x_1*,* x_2 *and* x_3 *are arranged in the monomial tensor* $\mathsf{M}(x_1, x_2, x_3) \in \mathbb{R}^{2\times2\times2}$ *as rank-*1 *tensor and in full representation.*

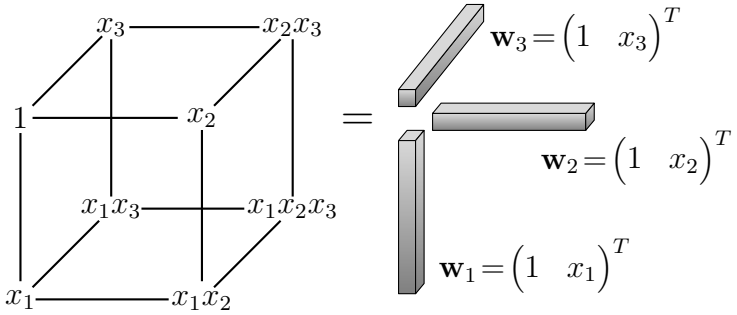


Figure 2.24: Monomial tensor $\mathsf{M}(x_1, x_2, x_3)$ of a function in three variables

With the tensorization of the monomials, the parameters of a multilinear function can be arranged in the same way in a tensor, leading to the tensor form of a multilinear function.

Definition 2.25 (Multilinear function in tensor form) *Using the monomial rank-*1 *tensor (2.43), the tensor representation of a multilinear function with* n *variables* $\mathbf{x} \in \mathbb{R}^n$ *reads*

$$h(\mathbf{x}) = \langle\, \mathsf{H} \mid \mathsf{M}(\mathbf{x}) \,\rangle \ , \tag{2.44}$$

with the parameter tensor $\mathsf{H} \in \mathbb{R}^{\times^{(n)}2}$*.*

Example 2.28 *The tensor representation of a function*

$$h(\mathbf{x}) = \langle\, \mathsf{H} \mid \mathsf{M}\,(x_1, x_2, x_3)\, \rangle = 2 + 5x_1 - 2x_2 - 7x_1x_2 + 6x_3 + 3x_1x_3 - 4x_2x_3 + x_1x_2x_3$$

in 3 *variables* x_1, x_2 *and* x_3 *is depicted in Figure 2.25.*

$$h(x_1, x_2, x_3) = \langle\, \mathsf{H} \mid \mathsf{M}\,(x_1, x_2, x_3)\, \rangle = \left\langle\, \mathsf{H} \,\middle|\, \left[\begin{pmatrix} 1 \\ x_3 \end{pmatrix}, \begin{pmatrix} 1 \\ x_2 \end{pmatrix}, \begin{pmatrix} 1 \\ x_1 \end{pmatrix} \right] \,\right\rangle$$

$$= \left\langle \begin{array}{ccc} & 6 & \text{———} & 3 \\ & \vert & & \vert \\ 2 & \text{———} & 5 & \vert \\ \vert & & \vert & \vert \\ \vert & -4 & \text{———} & 1 \\ \vert & \vert & & \vert \\ -2 & \text{———} & -7 & \end{array} \middle| \begin{array}{ccc} & x_3 & \text{———} & x_1x_3 \\ & \vert & & \vert \\ 1 & \text{———} & x_1 & \vert \\ \vert & & \vert & \vert \\ \vert & x_2x_3 & \text{———} & x_1x_2x_3 \\ \vert & \vert & & \vert \\ x_2 & \text{———} & x_1x_2 & \end{array} \right\rangle$$

$$= 2 + 5x_1 - 2x_2 - 7x_1x_2 + 6x_3 + 3x_1x_3 - 4x_2x_3 + x_1x_2x_3$$

Figure 2.25: Multilinear polynomial in tensor representation

The example shows the separation of constants and variables, since all parameters are stored in H *and all monomials are stored in* $\mathsf{M}(x_1, x_2, x_3)$*. The evaluation of the contracted product (2.4) gives exactly the given polynomial, which shows the correctness of the tensor representation here.*

So far, as introduced in [63] or [87], this polynomial tensor representation is limited to the class of multilinear polynomials. In modeling, multilinear polynomials are used as right hand sides to represent state space models of MTI systems, [63]. In the following chapters controller design methods, like feedback linearization, for MTI systems are derived in the tensor structure. Therefore, operations like the multiplication have to be defined in this framework. With the limitation to multilinear polynomials this is not possible as the following example illustrates.

Example 2.29 *Consider the multiplication of two multilinear functions with two variables* $h_1(x_1, x_2) = 1 + 2x_1x_2$ *and* $h_2(x_1, x_2) = x_1 - 3x_1x_2$

$$h_1(x_1, x_2) \cdot h_2(x_1, x_2) = (1 + 2x_1x_2)(x_1 - 3x_1x_2) = x_1 - 3x_1x_2 + 2x_1^2x_2 - 6x_1^2x_2^2. \qquad (2.45)$$

The result is not multilinear anymore, because it contains also higher order terms, like $2x_1^2x_2$ *or* $-6x_1^2x_2^2$*. Thus, the resulting polynomial cannot be represented in the tensor structure proposed until this point.*

For the reason illustrated by Example 2.29 the tensor framework has to be extended such that also higher order terms are captured. This is necessary to formulate controller design algorithms like feedback linearization in tensor structure in the following chapters.

To generalize the concept from multilinear to arbitrary polynomials, the monomial tensor of Definition 2.24 has to be extended, such that it does not contain multilinear combinations of the variables only, but also higher order combinations.

Definition 2.26 (Monomial tensor for higher order polynomials) *The monomials of a polynomial with maximal order N of the monomials in n variables $\mathbf{x} \in \mathbb{R}^n$ are computed, analogous to (2.42) by a sequence of outer products*

$$\mathsf{M}_p^N(\mathbf{x}) = \begin{pmatrix} 1 \\ x_n \end{pmatrix} \circ \cdots \circ \begin{pmatrix} 1 \\ x_1 \end{pmatrix} \circ \cdots\cdots \circ \begin{pmatrix} 1 \\ x_n \end{pmatrix} \circ \cdots \circ \begin{pmatrix} 1 \\ x_1 \end{pmatrix} = \bigcirc_{j=1}^{N} \bigcirc_{i=1}^{n} \begin{pmatrix} 1 \\ x_{n-i+1} \end{pmatrix},$$

leading to a tensor of dimension $\mathbb{R}^{\times^{(nN)}2}$. Since the monomial tensor for higher order polynomials is constructed by the outer product of vectors, it is a rank-1 tensor

$$\mathsf{M}_p^N(\mathbf{x}) = \underbrace{\left[\begin{pmatrix} 1 \\ x_n \end{pmatrix}, \ldots, \begin{pmatrix} 1 \\ x_1 \end{pmatrix}, \ldots\ldots, \begin{pmatrix} 1 \\ x_n \end{pmatrix}, \ldots, \begin{pmatrix} 1 \\ x_1 \end{pmatrix} \right]}_{N \text{ times}}. \tag{2.46}$$

A maximal order N of the monomials means, that variables with exponents up to N, i.e. x_i^N could occur.

Thus, the monomial tensor of polynomials of higher order is computed by an N times outer product of the multilinear monomial tensor with itself

$$\mathsf{M}_p^N(\mathbf{x}) = \bigcirc_{j=1}^{N} \mathsf{M}(\mathbf{x}).$$

With $N = 1$ one gets the multilinear case as introduced before in (2.42).

Example 2.30 *The monomial tensor of maximal order $N = 2$ in two variables*

$$\mathsf{M}_p^2(x_1, x_2) = \begin{pmatrix} 1 \\ x_2 \end{pmatrix} \circ \begin{pmatrix} 1 \\ x_1 \end{pmatrix} \circ \begin{pmatrix} 1 \\ x_2 \end{pmatrix} \circ \begin{pmatrix} 1 \\ x_1 \end{pmatrix} = \left[\begin{pmatrix} 1 \\ x_2 \end{pmatrix}, \begin{pmatrix} 1 \\ x_1 \end{pmatrix}, \begin{pmatrix} 1 \\ x_2 \end{pmatrix}, \begin{pmatrix} 1 \\ x_1 \end{pmatrix} \right]$$

is a tensor of dimension $\mathbb{R}^{2\times2\times2\times2}$ containing all the multiplicative combinations of x_1 and x_2 up to order 2. Figure 2.26 shows the two 3-dimensional subtensors of $\mathsf{M}_p^2(x_1, x_2)$.

$$\mathsf{M}_p^2(:,:,:,1) = \begin{array}{ccc} x_2 & - & x_1x_2 \\ | & \diagdown & | \\ 1 & - & x_1 \\ | & \diagdown & | \\ x_2^2 & - & x_1x_2^2 \\ | & \diagdown & | \\ x_2 & - & x_1x_2 \end{array} \qquad , \mathsf{M}_p^2(:,:,:,2) = \begin{array}{ccc} x_1x_2 & - & x_1^2x_2 \\ | & \diagdown & | \\ x_1 & - & x_1^2 \\ | & \diagdown & | \\ x_1x_2^2 & - & x_1^2x_2^2 \\ | & \diagdown & | \\ x_1x_2 & - & x_1^2x_2 \end{array}$$

Figure 2.26: Subtensors of the monomial tensor $\mathsf{M}_p^2(x_1, x_2)$ for a polynomial of maximal order 2

Example 2.30 shows, that obviously by construction some monomials appear multiple times inside the monomial tensor. The choice (2.46) of the monomial tensor, which is not unique, is not optimal regarding the storage effort but helps developing the algorithms. This redundancy is still acceptable, since the monomial tensor is a rank-1 tensor. By arranging the parameters of the polynomial in the same way, the tensor description of polynomials of higher order is defined.

Definition 2.27 (Higher order polynomial in tensor form) *A polynomial with maximal order N of the monomials in n variables $\mathbf{x} \in \mathbb{R}^n$ is given by*

$$h(\mathbf{x}) = \left\langle \mathsf{H} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle, \tag{2.47}$$

with parameter tensor $\mathsf{H} \in \mathbb{R}^{\times^{(nN)}2}$ and monomial tensor $\mathsf{M}_p^N(\mathbf{x}) \in \mathbb{R}^{\times^{(nN)}2}$, that is constructed as rank-1 tensor (2.46).

Example 2.31 *The result of the multiplication of two multilinear functions in Example 2.29 is a polynomial of maximal order 2*

$$h(x_1, x_2) = x_1 - 3x_1x_2 + 2x_1^2x_2 - 6x_1^2x_2^2.$$

To represent the function in the tensor framework a monomial tensor $\mathsf{M}_p^2(x_1, x_2)$ of order 2 is necessary. The monomial tensor was illustrated in Figure 2.26 and has the slices

$$\begin{aligned} \mathbf{M}_p^2(:,:,1,1) &= \begin{pmatrix} 1 & x_1 \\ x_2 & x_1x_2 \end{pmatrix}, \quad \mathbf{M}_p^2(:,:,1,2) = \begin{pmatrix} x_1 & x_1^2 \\ x_1x_2 & x_1^2x_2 \end{pmatrix}, \\ \mathbf{M}_p^2(:,:,2,1) &= \begin{pmatrix} x_2 & x_1x_2 \\ x_2^2 & x_1x_2^2 \end{pmatrix}, \quad \mathbf{M}_p^2(:,:,2,2) = \begin{pmatrix} x_1x_2 & x_1^2x_2 \\ x_1x_2^2 & x_1^2x_2^2 \end{pmatrix}. \end{aligned} \tag{2.48}$$

The function is represented by the contracted product of the parameter tensor and the monomial tensor

$$h(x_1, x_2) = \left\langle \mathsf{H} \,\middle|\, \mathsf{M}_p^2(x_1, x_2) \right\rangle.$$

The contracted product here is computed in the first step by an elementwise multiplication of the parameter and monomial tensor. Afterwards the elements of the result are summed up. This description of the contracted product helps to construct the parameter tensor

$$\begin{aligned} \mathbf{H}(:,:,1,1) &= \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad \mathbf{H}(:,:,1,2) = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}, \\ \mathbf{H}(:,:,2,1) &= \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad \mathbf{H}(:,:,2,2) = \begin{pmatrix} -3 & 0 \\ 0 & -6 \end{pmatrix}. \end{aligned} \tag{2.49}$$

Because of that, the parameters in H are in the same positions as the corresponding monomials in $\mathsf{M}_p^N(x_1, x_2)$ as one can see when comparing (2.49) with (2.48).

So far the tensor approach was derived for scalar polynomials $h(\mathbf{x}) \in \mathbb{R}$. The concept is simply extended to vector functions $\mathbf{h} : \mathbb{R}^n \to \mathbb{R}^m$ with components

$$\mathbf{h}(\mathbf{x}) = \begin{pmatrix} h_1(\mathbf{x}) & \cdots & h_m(\mathbf{x}) \end{pmatrix}^T \in \mathbb{R}^m.$$

The monomials of the function are still the same, such that the monomial tensor remains unchanged to the scalar case. The different scalar functions $h_i(\mathbf{x}),\ i = 1, \ldots, m$ of the vector function $\mathbf{h}(\mathbf{x})$ are considered by an additional dimension of the parameter tensor for the multilinear case

$$\mathbf{h}(\mathbf{x}) = \langle\, \mathsf{H} \mid \mathsf{M}\,(\mathbf{x}) \,\rangle\,,$$

with parameter tensor $\mathsf{H} \in \mathbb{R}^{\times^{(n)}2 \times m}$ and monomial tensor $\mathsf{M}\,(\mathbf{x}) \in \mathbb{R}^{\times^{(n)}2}$ as well as for higher order polynomials

$$\mathbf{h}(\mathbf{x}) = \left\langle\, \mathsf{H} \,\middle|\, \mathsf{M}_p^N\,(\mathbf{x}) \,\right\rangle,$$

with parameter tensor $\mathsf{H} \in \mathbb{R}^{\times^{(nN)}2 \times m}$ and monomial tensor $\mathsf{M}_p^N\,(\mathbf{x}) \in \mathbb{R}^{\times^{(nN)}2}$. The parameter tensor H of the vector function $\mathbf{h}(\mathbf{x})$ is constructed by concatenating the parameter tensors $\mathsf{H}_i \in \mathbb{R}^{\times^{(nN)}2}$ of the scalar elements $h_i(\mathbf{x}) = \left\langle\, \mathsf{H}_i \,\middle|\, \mathsf{M}_p^N\,(\mathbf{x}) \,\right\rangle,\ i = 1, \ldots, m$ in the new $(nN+1)^{\text{th}}$ dimension

$$\mathsf{H} = \mathsf{H}_1 \boxplus_{nN+1} \mathsf{H}_2 \boxplus_{nN+1} \cdots \boxplus_{nN+1} \mathsf{H}_m = \bigboxplus_{i=1}^{m}{}_{nN+1}\, \mathsf{H}_i,$$

which gives $\mathsf{H}(:, \ldots, :, i) = \mathsf{H}_i,\ i = 1, \ldots, m$.

Example 2.32 *The tensor description of the two multilinear functions used in Example 2.29 in tensor representation is given by*

$$h_1(x_1, x_2) = 1 + 2x_1x_2 = \langle\, \mathsf{H}_1 \mid \mathsf{M}\,(\mathbf{x}) \,\rangle = \left\langle \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix} \middle| \begin{pmatrix} 1 & x_1 \\ x_2 & x_1x_2 \end{pmatrix} \right\rangle, \tag{2.50}$$

$$h_2(x_1, x_2) = x_1 - 3x_1x_2 = \langle\, \mathsf{H}_2 \mid \mathsf{M}\,(\mathbf{x}) \,\rangle = \left\langle \begin{pmatrix} 0 & 1 \\ 0 & -3 \end{pmatrix} \middle| \begin{pmatrix} 1 & x_1 \\ x_2 & x_1x_2 \end{pmatrix} \right\rangle. \tag{2.51}$$

Combining these two functions to a vector function with the tensor representation

$$\mathbf{h}(\mathbf{x}) = \begin{pmatrix} h_1(\mathbf{x}) \\ h_2(\mathbf{x}) \end{pmatrix} = \langle\, \mathsf{H} \mid \mathsf{M}\,(\mathbf{x}) \,\rangle\,, \tag{2.52}$$

leads to the parameter tensor $\mathsf{H} \in \mathbb{R}^{\times^{(2)}2 \times 2}$ *with slices*

$$\mathbf{H}(:, :, 1) = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}, \quad \mathbf{H}(:, :, 2) = \begin{pmatrix} 0 & 1 \\ 0 & -3 \end{pmatrix}.$$

This representation of polynomials shows that in this context not data, like samples of measurements, is stored inside the tensors as shown in Example 2.1 but the values of the parameters. The monomials are arranged in a tensor, too. With the fixed structure of the monomials given by the monomial tensor, the function is completely characterized by its parameter tensor, when the number of variables and maximal order of the function is known. Using this tensor representation of multilinear and polynomial functions, where the parameters and monomials are separately stored in different tensors, the aim in the next sections is to compute the result of basic polynomial operations like multiplication or differentiation based on the parameter tensors. This means that the parameter tensor of the result of those operations is calculated by using the parameter tensors of the operands, without evaluating the contracted product with the monomial tensor. The result is computed by so-called operational tensors. This gives an analytically exact solution of these operations by simple numerical computations. No symbolic calculation is necessary.

2.3.2 Multiplication

Example 2.29 showed, that the result of the multiplication of two multilinear polynomials cannot be described in the multilinear tensor framework (2.44) since higher order terms occur. With Definition 2.27 it is possible to represent higher order polynomials and thus allows to derive the multiplication of polynomials based on the parameter tensors.

Theorem 2.1 (Multiplication in tensor form) *The multiplication of two polynomials in n variables of maximal orders N_1 and N_2 in tensor representation*

$$h_1(\mathbf{x}) = \left\langle \mathsf{H}_1 \,\middle|\, \mathsf{M}_p^{N_1}(\mathbf{x}) \right\rangle,$$
$$h_2(\mathbf{x}) = \left\langle \mathsf{H}_2 \,\middle|\, \mathsf{M}_p^{N_2}(\mathbf{x}) \right\rangle,$$

with parameter tensors $\mathsf{H}_i \in \mathbb{R}^{\times^{(nN_i)}2}$ and monomial tensors $\mathsf{M}_p^{N_i}(\mathbf{x}) \in \mathbb{R}^{\times^{(nN_i)}2}$, $i = 1, 2$,

$$h_1(\mathbf{x}) \cdot h_2(\mathbf{x}) = \left\langle \mathsf{H}_1 \circ \mathsf{H}_2 \,\middle|\, \mathsf{M}_p^{N_1+N_2}(\mathbf{x}) \right\rangle, \tag{2.53}$$

is a polynomial of order $N_1 + N_2$.

The Appendix A.4 shows the proof of this theorem.

Example 2.33 *The multiplication of the two multilinear functions of Example 2.29, that have tensor representations (2.50) and (2.51) is given by*

$$h(x_1, x_2) = h_1(x_1, x_2) h_2(x_1, x_2) = \langle \mathsf{H}_1 \mid \mathsf{M}(x_1, x_2) \rangle \langle \mathsf{H}_2 \mid \mathsf{M}(x_1, x_2) \rangle = \left\langle \mathsf{H}_1 \circ \mathsf{H}_2 \,\middle|\, \mathsf{M}_p^2(x_1, x_2) \right\rangle.$$

The result of the multiplication computed directly in (2.45) is a polynomial of maximal order 2, which corresponds to the monomial tensor $\mathsf{M}_p^2(x_1, x_2)$ of the result. The monomial

tensor was illustrated in Figure 2.26 and has the slices (2.48). Computing the outer product $\mathsf{H} = \mathsf{H}_1 \circ \mathsf{H}_2$ *gives the parameter tensor of the result with slices*

$$\begin{aligned} \mathbf{H}(:,:,1,1) &= \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad \mathbf{H}(:,:,1,2) = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}, \\ \mathbf{H}(:,:,2,1) &= \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad \mathbf{H}(:,:,2,2) = \begin{pmatrix} -3 & 0 \\ 0 & -6 \end{pmatrix}. \end{aligned} \tag{2.54}$$

The result of the outer product (2.54) is equal to the parameter tensor of $h(x_1, x_2)$ *that was derived in (2.49). Thus, the parameter tensor of the multiplication of* $h_1(x_1, x_2)$ *and* $h_2(x_1, x_2)$ *is computed correctly by the outer product* $\mathsf{H}_1 \circ \mathsf{H}_2$ *leading to*

$$h(x_1, x_2) = \left\langle \mathsf{H}_1 \circ \mathsf{H}_2 \;\middle|\; \mathsf{M}_p^2(x_1, x_2) \right\rangle = x_1 - 3x_1x_2 + 2x_1^2x_2 - 6x_1^2x_2^2.$$

If the parameter tensors H_1 and H_2 are given as decomposed tensors, e.g. in CP format, the parameter tensor of their product is computed efficiently since the outer product is defined on the decomposition factors like for CP tensors by Theorem 2.1. Thus building the full tensors of H_1 and H_2 is not required in this case.

2.3.3 Differentiation

In this section the computation of the partial derivative of a polynomial of order N in tensor representation (2.47) with respect to one variable $x_j,\ j = 1, \ldots, n$

$$\frac{\partial}{\partial x_j} h(\mathbf{x}) = \frac{\partial}{\partial x_j} \left\langle \mathsf{H} \;\middle|\; \mathsf{M}_p^N(\mathbf{x}) \right\rangle$$

is investigated. One possibility to determine the exact derivative of a polynomial is a symbolical computation. The differentiation approach introduced here should take advantage of the tensor structure of the polynomial representation. The fact is used, that the derivative of a polynomial with respect to one variable is still a polynomial, that can be represented by a parameter tensor and a monomial tensor. When differentiating a polynomial, the maximal order N of the monomials is not increased. The exponents of the variables x_i with $i \neq j$ remain unchanged and the order of the variables x_j are reduced by one

$$\frac{\partial}{\partial x_j} x_j^n = n x_j^{n-1}.$$

Thus, the derivative $\frac{\partial}{\partial x_j} h(\mathbf{x})$ can be represented with a monomial tensor of the same maximal order than the function $h(\mathbf{x})$, i.e. the monomial tensor of the derivative is $\mathsf{M}_p^N(\mathbf{x})$ too. The question, how the parameter tensor H_{x_j} of the derivative

$$\frac{\partial}{\partial x_j} h(\mathbf{x}) = \left\langle \mathsf{H}_{x_j} \;\middle|\; \mathsf{M}_p^N(\mathbf{x}) \right\rangle$$

is computed by using the parameter tensor H of the original function $h(\mathbf{x})$ only, is investigated in the following Theorem.

Theorem 2.2 (Differentiation in tensor form) *The partial derivative of a polynomial in n variables of maximal monomial order N with respect to one variable x_j with $j = 1, \dots, n$ in tensor representation is given by*

$$\frac{\partial}{\partial x_j} h(\mathbf{x}) = \frac{\partial}{\partial x_j} \left\langle \mathsf{H} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle = \left\langle \mathsf{H}_{x_j} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle, \tag{2.55}$$

with the parameter tensor of the differentiated function

$$\mathsf{H}_{x_j} = \sum_{k=1}^{N} \mathsf{H} \times_{kn-j+1} \boldsymbol{\Theta} \in \mathbb{R}^{\times (nN)2}, \tag{2.56}$$

and a matrix $\boldsymbol{\Theta} = \left(\begin{smallmatrix} 0 & 1 \\ 0 & 0 \end{smallmatrix}\right)$.

The theorem is proven in the Appendix A.5. With Theorem 2.2 a simple numerical computation of the partial derivative of polynomials based on the tensor representation is found, that computes an analytically exact solution and not a numerical approximation of the derivative. This realizes an automatic differentiation algorithm without the disadvantages of symbolical or numerical differentiation, like high computational costs or inaccuracies, [92]. A simple tensor matrix product in one mode has to be computed only to get the parameter tensor of the derivative. This allows a simple computation even for large polynomials with a lot of variables or a high maximal order. Independently of the number of variables or the maximal order, the parameter tensor has to be multiplied by a small scale, i.e. 2×2, matrix $\boldsymbol{\Theta}$ only. To compute the derivative with respect to different variables x_j, only the mode $\times_{kn-j+1}$ of the tensor matrix product has to be changed.
Especially in the case, where the parameter tensor H is given as decomposed tensor, the computation is simplified, because the tensor matrix product is defined for the decomposition methods as proposed in Section 2.1.2. Assuming as an example that the parameter tensor H is in CP form, the CP representation of H_{x_j} is constructed using the decomposition factors of H only, without building the full tensors. This is illustrated in the following example.

Example 2.34 *Consider a multilinear polynomial with two variables*

$$h(\mathbf{x}) = 1 + 2x_1 + 3x_2 + 6x_1x_2 = \langle \mathsf{H} \mid \mathsf{M}(\mathbf{x}) \rangle = \left\langle \begin{pmatrix} 1 & 2 \\ 3 & 6 \end{pmatrix} \middle| \begin{pmatrix} 1 & x_1 \\ x_2 & x_1x_2 \end{pmatrix} \right\rangle.$$

By applying (2.56) the parameter tensor of the differentiated function with respect to x_1 results in

$$\mathsf{H}_1 = \mathsf{H} \times_2 \boldsymbol{\Theta} = \begin{pmatrix} 2 & 0 \\ 6 & 0 \end{pmatrix},$$

which describes the function

$$\frac{\partial}{\partial x_1} h(\mathbf{x}) = \langle \mathsf{H}_1 \mid \mathsf{M}(\mathbf{x}) \rangle = \left\langle \begin{pmatrix} 2 & 0 \\ 6 & 0 \end{pmatrix} \middle| \begin{pmatrix} 1 & x_1 \\ x_2 & x_1x_2 \end{pmatrix} \right\rangle = 2 + 6x_2,$$

i.e. the correct derivative. The mode-2 multiplication with $\boldsymbol{\Theta}$ maps the elements of H belonging to the monomials x_1 and x_1x_2 to the monomials 1 and x_2. All others are zero, which is

exactly the case when differentiating this function with respect to x_1.
The parameter tensor of $h(\mathbf{x})$ can be given in a CP tensor structure

$$\mathsf{H} = \begin{pmatrix} 1 & 2 \\ 3 & 6 \end{pmatrix} = \left[\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \right] \cdot \begin{pmatrix} 1 & 2 & 3 & 6 \end{pmatrix}^T.$$

How to construct the CP form of a parameter tensor will be described in Section 3.2.1. To compute the tensor matrix product by using the decomposed form the second factor matrix is multiplied from the right by $\mathbf{\Theta}$. Thus, the CP representation of the differentiated function is given by

$$\mathsf{H}_1 = \left[\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \right] \cdot \begin{pmatrix} 1 & 2 & 3 & 6 \end{pmatrix}^T = \left[\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \right] \cdot \begin{pmatrix} 2 & 6 \end{pmatrix}^T,$$

where the number of rank-1 components can be reduced by removing the zero columns of the factor matrices to get a more efficient representation of the CP tensor. To verify the result the full tensor is reconstructed from the CP representation leading to

$$\mathsf{H}_1 = \left[\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \right] \cdot \begin{pmatrix} 2 & 6 \end{pmatrix}^T = \begin{pmatrix} 2 & 0 \\ 6 & 0 \end{pmatrix},$$

which shows the same result as before with the full tensor.

Remark 2.2 *The CP representation of the parameter tensors given in Example 2.34 needs a larger storage effort, than the full representation as a 2×2 matrix. This could be misleading because it was stated before that a tensor decomposition gives a memory efficient representation of the tensor. The reason for this contradicting case here is the low dimensionality of the parameter tensor. A polynomial with two variables and maximal order $N = 1$ is considered only, to have a simple example for the illustration of the differentiation concept. For a polynomial with a larger number of variables the benefits of the decomposition comes into play. For example a polynomial with 20 variables has $2^{20} = 10^6$ elements in full representation, i.e. approximately a million. If it is possible to construct a CP form with 4 rank-1 elements, like here in this example, 164 elements would have to be stored only, which shows the advantage of the decomposition.*

2.3.4 Lie derivative

Two basic polynomial operations were introduced in the previous two sections. To design a feedback linearizing controller, which will be introduced for MTI systems in Section 4.2, an additional operation is necessary. The Lie derivative of a scalar function $g(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ along a vector field $\mathbf{h} : \mathbb{R}^n \to \mathbb{R}^n$ with components $\mathbf{h}(\mathbf{x}) = \begin{pmatrix} h_1(\mathbf{x}) & \cdots & h_n(\mathbf{x}) \end{pmatrix}^T$ is defined as, [39]

$$L_{\mathbf{h}} g(\mathbf{x}) = \sum_{i=1}^{n} h_i(\mathbf{x}) \frac{\partial g(\mathbf{x})}{\partial x_i} \in \mathbb{R}. \tag{2.57}$$

The Lie derivative is known from differential geometry and gives the derivative of a function along a vector field, [39]. An i-times application of the Lie derivative to a function is denoted by $L_{\mathbf{h}} \cdots L_{\mathbf{h}} g(\mathbf{x}) = L_{\mathbf{h}}^i g(\mathbf{x})$. Equation (2.57) shows that multiplications and partial derivatives have to be computed to determine the Lie derivative. Consider now that the functions $\mathbf{h}(\mathbf{x})$ and $g(\mathbf{x})$ are given as polynomials in a tensor format

$$\mathbf{h}(\mathbf{x}) = \left\langle \mathsf{H} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle, \tag{2.58}$$

$$g(\mathbf{x}) = \left\langle \mathsf{G} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle, \tag{2.59}$$

with parameter tensors $\mathsf{H} \in \mathbb{R}^{\times(nN)2\times n}$ and $\mathsf{G} \in \mathbb{R}^{\times(nN)2}$. In this case the multiplication and differentiation methods (2.53) and (2.55) are used to compute the Lie derivative (2.57) based on the parameter tensors.

Theorem 2.3 (Lie derivative in tensor form) *The tensor representation of the Lie derivative (2.57) of the scalar polynomial (2.59) along the polynomial vector field (2.58) yields*

$$L_{\mathbf{h}}^l g(\mathbf{x}) = \sum_{i=1}^n h_i(\mathbf{x}) \frac{\partial}{\partial x_i} L_{\mathbf{h}}^{l-1} g(\mathbf{x}) = \left\langle \mathsf{L}_{\mathsf{H},\mathsf{G},l} \,\middle|\, \mathsf{M}_p^{N(l+1)}(\mathbf{x}) \right\rangle, \tag{2.60}$$

with the parameter tensor

$$\mathsf{L}_{\mathsf{H},\mathsf{G},l} = \begin{cases} \mathsf{G} & , \text{for } l = 0, \\ \sum\limits_{i=1}^n \mathsf{H}_i \circ \left(\sum\limits_{k=1}^N \mathsf{G} \times_{kn-i+1} \boldsymbol{\Theta} \right) & , \text{for } l = 1, \\ \sum\limits_{i=1}^n \mathsf{H}_i \circ \left(\sum\limits_{k=1}^{lN} \mathsf{L}_{\mathsf{H},\mathsf{G},l-1} \times_{kn-i+1} \boldsymbol{\Theta} \right) & , \text{else}, \end{cases} \tag{2.61}$$

with subtensor $\mathsf{H}_i = \mathsf{H}(:,\ldots,:,i)$*, where the last dimension of* H *is fixed, which means*

$$\mathsf{H} = \mathsf{H}_1 \boxplus_{n+1} \mathsf{H}_2 \boxplus_{n+1} \cdots \boxplus_{n+1} \mathsf{H}_n = \bigboxplus_{i=1}^n{}_{n+1} \mathsf{H}_i.$$

The theorem is proven in the Appendix A.6.

Example 2.35 *The vector function (2.52)*

$$\mathbf{h}(x_1, x_2) = \langle \mathsf{H} \mid \mathsf{M}(x_1, x_2) \rangle$$

from Example 2.32 has a parameter tensor

$$\mathsf{H}_1 = \mathbf{H}(:,:,1) = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}, \quad \mathsf{H}_2 = \mathbf{H}(:,:,2) = \begin{pmatrix} 0 & 1 \\ 0 & -3 \end{pmatrix}.$$

The first Lie derivative $L_{\mathbf{h}} g(\mathbf{x})$*, i.e.* $l = 1$*, of the scalar function*

$$g(\mathbf{x}) = x_2 - x_1 x_2 = \langle \mathsf{G} \mid \mathsf{M}(\mathbf{x}) \rangle = \left\langle \begin{pmatrix} 0 & 0 \\ 1 & -1 \end{pmatrix} \middle| \begin{pmatrix} 1 & x_1 \\ x_2 & x_1 x_2 \end{pmatrix} \right\rangle,$$

along the vector field $\mathbf{h}(\mathbf{x})$ *should be computed by operational tensors. For comparison the computation of the Lie derivative with the standard approach (2.57) gives*

$$L_{\mathbf{h}}g(\mathbf{x}) = h_1(\mathbf{x})\frac{\partial}{\partial x_1}g(\mathbf{x}) + h_2(\mathbf{x})\frac{\partial}{\partial x_2}g(\mathbf{x}) = x_1 - x_2 - 3x_1x_2 - x_1^2 + 3x_1^1x_2 - 2x_1x_2^2. \quad (2.62)$$

With the method introduced in Theorem 2.3 the parameter tensor of the Lie derivative

$$\mathsf{L}_{\mathsf{H},\mathsf{G},1} = \mathsf{H}_1 \circ (\mathsf{G} \times_2 \mathbf{\Theta}) + \mathsf{H}_2 \circ (\mathsf{G} \times_1 \mathbf{\Theta}),$$

results in a fourth order tensor

$$\mathbf{L}_{\mathsf{H},\mathsf{G},1}(:,:,1,1) = \begin{pmatrix} 0 & 1 \\ 0 & -3 \end{pmatrix}, \quad \mathbf{L}_{\mathsf{H},\mathsf{G},1}(:,:,1,2) = \begin{pmatrix} 0 & -1 \\ 0 & 3 \end{pmatrix},$$
$$\mathbf{L}_{\mathsf{H},\mathsf{G},1}(:,:,2,1) = \begin{pmatrix} -1 & 0 \\ 0 & -2 \end{pmatrix}, \quad \mathbf{L}_{\mathsf{H},\mathsf{G},1}(:,:,2,2) = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

From (2.60) follows that the monomial tensor $\mathsf{M}_p^2(x_1, x_2)$ *for the Lie derivative is of maximal order* 2*, which was computed in full representation in (2.48). Evaluating the contracted product of the parameter tensor with the monomial tensor* $\left\langle \mathsf{L}_{\mathsf{H},\mathsf{G},1} \,\middle|\, \mathsf{M}_p^2(x_1, x_2) \right\rangle$ *shows that the result with the operational tensor is equal to (2.62). Thus, the parameter tensor is computed correctly.*

With the definition of the Lie derivative (2.60) in tensor representation, a related operation, the Lie brackets, can be described in a similar way, that is also important in the design of feedback linearizing controllers. The Lie bracket of two vector functions $\mathbf{h}(\mathbf{x})$ and $\mathbf{g}(\mathbf{x})$ is given by, [39]

$$[\mathbf{h}, \mathbf{g}] = \frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}}\mathbf{h}(\mathbf{x}) - \frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}}\mathbf{g}(\mathbf{x}) = L_{\mathbf{h}}\mathbf{g}(\mathbf{x}) - L_{\mathbf{g}}\mathbf{h}(\mathbf{x}), \quad (2.63)$$

where $\frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}}$ and $\frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}}$ are the Jacobian matrices of $\mathbf{h}(\mathbf{x})$ and $\mathbf{g}(\mathbf{x})$ respectively.

Lemma 2.1 (Lie bracket) *The Lie bracket of two vector functions* $\mathbf{h}, \mathbf{g} : \mathbb{R}^n \to \mathbb{R}^n$ *in* n *variables*

$$\mathbf{h}(\mathbf{x}) = \left\langle \mathsf{H} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle,$$
$$\mathbf{g}(\mathbf{x}) = \left\langle \mathsf{G} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle,$$

with parameter tensors $\mathsf{H} \in \mathbb{R}^{\times^{(nN)}2\times n}$ *and* $\mathsf{G} \in \mathbb{R}^{\times^{(nN)}2\times n}$*, is given by*

$$[\mathbf{h}, \mathbf{g}] = \left\langle \mathsf{L}_{\mathsf{H},\mathsf{G},1} - \mathsf{L}_{\mathsf{G},\mathsf{H},1} \,\middle|\, \mathsf{M}_p^{2N}(\mathbf{x}) \right\rangle.$$

The proof of the lemma is given in the Appendix A.6. This can be extended to the multiple application of the Lie bracket. For the repeated Lie bracket $[\mathbf{h}, [\mathbf{h}, \dots [\mathbf{h}, \mathbf{g}]]]$ of the vector field $\mathbf{g}(\mathbf{x})$ with the same vector field $\mathbf{h}(\mathbf{x})$ the recursive notation

$$ad_{\mathbf{h}}^k\mathbf{g}(\mathbf{x}) = \left[\mathbf{h}, ad_{\mathbf{h}}^{k-1}\mathbf{g}\right](\mathbf{x}) = \left\langle \mathsf{L}_{ad_{\mathbf{h}}\mathbf{g}}^k \,\middle|\, \mathsf{M}_p^{k+1}(\mathbf{x}) \right\rangle, \quad (2.64)$$

is used with $k \geq 1$. The term $ad_{\mathbf{h}}^0\mathbf{g}(\mathbf{x})$ gives $\mathbf{g}(\mathbf{x})$.

2.3.5 Jacobian and Hessian matrix

The partial differentiation of a polynomial in tensor format by operational tensors was described in Section 2.3.3. This operation will be used in this section to compute the Jacobian and the Hessian of a polynomial in tensor format. This allows e.g. to approximate higher order polynomials with lower order Taylor series, to reduce the storage effort for the polynomials.

Jacobian

The Jacobian matrix of a vector function $\mathbf{h} : \mathbb{R}^n \to \mathbb{R}^m$ with n variables contains all first partial derivatives of the function, [14]. Each component of $\mathbf{h}(\mathbf{x})^T = \begin{pmatrix} h_1(\mathbf{x}) & \cdots & h_m(\mathbf{x}) \end{pmatrix}$ is derived with respect to all variables $x_j,\ j = 1, \ldots, n$. This results in a matrix of dimension $\mathbb{R}^{m \times n}$

$$\mathbf{J}_\mathbf{h}(\mathbf{x}) = \left(\frac{\partial h_i(\mathbf{x})}{x_j} \right) = \frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial h_1(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial h_1(\mathbf{x})}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_m(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial h_m(\mathbf{x})}{\partial x_n} \end{pmatrix}. \tag{2.65}$$

Assuming now that the vector function

$$\mathbf{h}(\mathbf{x}) = \left\langle \mathsf{H} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle,$$

is given in a tensor format. This allows to compute the partial derivatives of $\mathbf{h}(\mathbf{x})$ by operational tensors as derived in Section 2.3.3. With (2.55) the columns of the Jacobian matrix (2.65) are described by

$$\mathbf{j}_\mathbf{h}(\mathbf{x})(:, j) = \begin{pmatrix} \frac{\partial h_1(\mathbf{x})}{\partial x_j} \\ \vdots \\ \frac{\partial h_m(\mathbf{x})}{\partial x_j} \end{pmatrix} = \left\langle \mathsf{H}_{x_j} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle,\ j = 1, \ldots, n.$$

Each column of the Jacobian is represented by a parameter tensor and the monomial tensor. The monomial tensor of all columns is the same, since the orders of the derivatives do not change. Because of that, the parameter tensors of the derivatives can be stored in a tensor $\mathsf{J}_\mathbf{h} \in \mathbb{R}^{\times^{(nN)} 2 \times m \times n}$ to describe the whole Jacobian matrix directly

$$\mathbf{J}_\mathbf{h}(\mathbf{x}) = \left\langle \mathsf{J}_\mathbf{h} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle. \tag{2.66}$$

The first (nN) dimensions $\mathbf{i}_x = \begin{pmatrix} i_1 & \cdots & i_{nN} \end{pmatrix}^T \in \mathbb{R}^{nN}$ of $\mathsf{J}_\mathbf{h}$ belong to the variables $\mathbf{x}$, the last two dimensions $i_{nN+1} = m$ and $i_{nN+2} = n$ are the dimensions of the resulting Jacobian matrix. The parameter tensor of the Jacobian is constructed from the parameter tensors of the partial derivatives of $\mathbf{h}(\mathbf{x})$ by

$$\mathsf{J}_\mathbf{h}(\mathbf{i}_x, :, j) = \mathsf{H}_{x_j}(\mathbf{i}_x, :),\ j = 1, \ldots, n. \tag{2.67}$$

Having all the information on the derivatives stored in the parameter tensor $\mathsf{J}_\mathbf{h}$, the Jacobian matrix at some point $\mathbf{x}$ is computed by a simple evaluation of the contracted product (2.66) with the monomial tensor. The result of the derivation of the Jacobian matrix for polynomials in tensor format can be summarized in the following lemma.

Lemma 2.2 (Jacobian matrix in tensor form) *Consider a polynomial of maximal order N in n variables in tensor format*

$$\mathbf{h}(\mathbf{x}) = \left\langle \mathsf{H} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle.$$

The Jacobian matrix of $\mathbf{h}(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^m$ can be computed by

$$\mathbf{J}_{\mathbf{h}}(\mathbf{x}) = \left\langle \mathsf{J}_{\mathbf{h}} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle, \tag{2.68}$$

with the parameter tensor $\mathsf{J}_{\mathbf{h}} \in \mathbb{R}^{\times^{(nN)}2\times m\times n}$ constructed by

$$\mathsf{J}_{\mathbf{h}}(\mathbf{i}_x, :, j) = \mathsf{H}_{x_j}(\mathbf{i}_x, :),\ j = 1, \ldots, n, \tag{2.69}$$

where the parameter tensor H_{x_j} follows from (2.56).

Example 2.36 *Consider the example function*

$$h(\mathbf{x}) = x_1 - 3x_1x_2.$$

Since $h(\mathbf{x})$ is a scalar function, the Jacobian is a vector

$$\mathbf{J}_h(x_1, x_2) = \begin{pmatrix} 1 - 3x_2 \\ -3x_1 \end{pmatrix} = \langle\, \mathsf{J}_h \mid \mathsf{M}(x_1, x_2) \,\rangle,$$

where the parameter tensor is computed from the parameter tensor of $h(\mathbf{x})$ given in (2.51) and the differentiation approach (2.56) to

$$\mathsf{J}_h(:,:,1) = \begin{pmatrix} 0 & 1 \\ 0 & -3 \end{pmatrix} \times_2 \boldsymbol{\Theta} = \begin{pmatrix} 1 & 0 \\ -3 & 0 \end{pmatrix},$$
$$\mathsf{J}_h(:,:,2) = \begin{pmatrix} 0 & 1 \\ 0 & -3 \end{pmatrix} \times_1 \boldsymbol{\Theta} = \begin{pmatrix} 0 & -3 \\ 0 & 0 \end{pmatrix},$$

which gives the correct solution as shown in Figure 2.27.

$$\mathbf{J}_h(x_1, x_2) = \langle\, \mathsf{J}_h \mid \mathsf{M}(x_1, x_2) \,\rangle = \left\langle \begin{array}{ccc} 0 & & -3 \\ & \diagup\diagdown & \\ 1 & & 0 \\ & \diagdown\diagup & \\ 0 & & 0 \\ & \diagup\diagdown & \\ -3 & & 0 \end{array} \,\middle|\, \begin{array}{ccc} 1 & & x_1 \\ & \diagdown\diagup & \\ & & \\ & \diagup\diagdown & \\ x_2 & & x_1x_2 \end{array} \right\rangle = \begin{pmatrix} 1 - 3x_2 \\ -3x_1 \end{pmatrix}$$

Figure 2.27: Tensor representation with full tensors

Hessian

So far, the first partial derivatives $\partial h(\mathbf{x})/\partial x_j$ of polynomials were investigated with the tensor approach by computing the Jacobian matrix (2.68). This concept is extended now to second order derivatives in the Hessian matrix. Therefore consider a scalar function $h(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$. The Hessian matrix stores all second order partial derivatives of $h(\mathbf{x})$, [14]

$$\mathbf{H}_h(\mathbf{x}) = \left(\frac{\partial^2 h(\mathbf{x})}{\partial x_i \partial x_j} \right) = \begin{pmatrix} \frac{\partial^2 h(\mathbf{x})}{\partial x_1^2} & \frac{\partial^2 h(\mathbf{x})}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 h(\mathbf{x})}{\partial x_1 \partial x_n} \\ \frac{\partial^2 h(\mathbf{x})}{\partial x_2 \partial x_1} & \frac{\partial^2 h(\mathbf{x})}{\partial x_2^2} & \cdots & \frac{\partial^2 h(\mathbf{x})}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 h(\mathbf{x})}{\partial x_n \partial x_1} & \frac{\partial^2 h(\mathbf{x})}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 h(\mathbf{x})}{\partial x_n^2} \end{pmatrix} \in \mathbb{R}^{n \times n}.$$

If the polynomial is given in terms of the tensor description

$$h(\mathbf{x}) = \left\langle \, \mathsf{H} \, \middle| \, \mathsf{M}_p^N \left(\mathbf{x} \right) \, \right\rangle,$$

the derivative with respect to one variable, e.g. x_j, is written by (2.55). The result is again a polynomial in tensor format with the same maximal order, such that the derivation operation (2.55) can be simply applied one more time to the derivative to get the second order derivative with respect to two variables x_i and x_j

$$\frac{\partial^2 h(\mathbf{x})}{\partial x_i \partial x_j} = \frac{\partial}{\partial x_i} \left\langle \, \mathsf{H}_{x_j} \, \middle| \, \mathsf{M}_p^N \left(\mathbf{x} \right) \, \right\rangle = \left\langle \, \mathsf{H}_{x_i, x_j} \, \middle| \, \mathsf{M}_p^N \left(\mathbf{x} \right) \, \right\rangle. \tag{2.70}$$

Using (2.56) twice, the parameter tensor of the second order derivative is given by

$$\begin{aligned} \mathsf{H}_{x_i, x_j} &= \sum_{l=1}^{N} \mathsf{H}_{x_j} \times_{ln-i+1} \boldsymbol{\Theta} = \sum_{l=1}^{N} \left(\sum_{k=1}^{N} \mathsf{H} \times_{kn-j+1} \boldsymbol{\Theta} \right) \times_{ln-i+1} \boldsymbol{\Theta} \\ &= \sum_{l=1}^{N} \sum_{k=1}^{N} \mathsf{H} \times_{kn-j+1} \boldsymbol{\Theta} \times_{ln-i+1} \boldsymbol{\Theta}. \end{aligned} \tag{2.71}$$

With (2.70) and (2.71) all second order derivatives are described in the tensor format, such that all elements of the Hessian matrix can be computed with the same monomial tensor. The complete Hessian is expressed in this form by

$$\mathbf{H}_h(\mathbf{x}) = \left\langle \, \mathsf{\Gamma}_h \, \middle| \, \mathsf{M}_p^N \left(\mathbf{x} \right) \, \right\rangle, \tag{2.72}$$

where the parameter tensor $\mathsf{\Gamma}_h \in \mathbb{R}^{\times^{(nN)} 2 \times n \times n}$ contains the parameter tensors of the derivatives (2.71)

$$\Gamma_h(\mathbf{i}_x, i, j) = \mathsf{H}_{x_i, x_j}(\mathbf{i}_x), \forall i, \;\; j = 1, \ldots, n. \tag{2.73}$$

Again the first nN dimensions of H_h belong to the indices $\mathbf{i}_x \in \mathbb{R}^{nN}$ of the variables. The last two dimensions $i_{nN+1} = n$ and $i_{nN+2} = n$ belong to the dimensions of the Hessian matrix. To get the Hessian matrix at some point $\mathbf{x}$ the contracted product product (2.72) simply has to be evaluated. No further computations are necessary once the parameter tensor (2.71) is found. A brief summary of the derived results on the Hessian matrix is given in the following lemma.

Lemma 2.3 (Hessian matrix in tensor form) *The Hessian matrix of a scalar polynomial of maximal order N in n variables $\mathbf{x} \in \mathbb{R}^n$*

$$h(\mathbf{x}) = \left\langle \mathsf{H} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle,$$

is given in tensor format by

$$\mathbf{H}_h(\mathbf{x}) = \left\langle \mathsf{\Gamma}_h \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle. \tag{2.74}$$

The parameter tensor $\mathsf{\Gamma}_h \in \mathbb{R}^{\times^{(nN)}2\times n\times n}$ is described elementwise by

$$\Gamma_h(\mathbf{i}_x, i, j) = \mathsf{H}_{x_i,x_j}(\mathbf{i}_x),\ \forall i,\ j = 1,\ldots,n, \tag{2.75}$$

with the index vector $\mathbf{i}_x \in \mathbb{R}^{\times^{(nN)}2}$ of the variables and parameter tensors H_{x_i,x_j} of the second derivative (2.71).

Example 2.37 *The same function as in Example 2.36 is used to compute the Hessian, which gives*

$$\mathbf{H}_h(x_1,x_2) = \begin{pmatrix} 0 & -3 \\ -3 & 0 \end{pmatrix} = \langle\, \mathsf{\Gamma}_h \mid \mathsf{M}(x_1,x_2) \,\rangle.$$

With (2.73) parameter tensor H_h results in

$$\begin{aligned}
\mathbf{\Gamma}_h(:,:,1,1) &= \begin{pmatrix} 0 & 1 \\ 0 & -3 \end{pmatrix} \times_2 \mathbf{\Theta} \times_2 \mathbf{\Theta} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \\
\mathbf{\Gamma}_h(:,:,2,2) &= \begin{pmatrix} 0 & 1 \\ 0 & -3 \end{pmatrix} \times_1 \mathbf{\Theta} \times_1 \mathbf{\Theta} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \\
\mathbf{\Gamma}_h(:,:,1,2) &= \begin{pmatrix} 0 & 1 \\ 0 & -3 \end{pmatrix} \times_1 \mathbf{\Theta} \times_2 \mathbf{\Theta} = \begin{pmatrix} -3 & 0 \\ 0 & 0 \end{pmatrix}, \\
\mathbf{\Gamma}_h(:,:,2,1) &= \begin{pmatrix} 0 & 1 \\ 0 & -3 \end{pmatrix} \times_2 \mathbf{\Theta} \times_1 \mathbf{\Theta} = \begin{pmatrix} -3 & 0 \\ 0 & 0 \end{pmatrix}.
\end{aligned}$$

The evaluation of the contracted product of the parameter tensor and the monomial tensor $\mathsf{M}(x_1,x_2) = \begin{pmatrix} 1 & x_2 \\ x_2 & x_1x_2 \end{pmatrix}$ gives the correct result $\begin{pmatrix} 0 & -3 \\ -3 & 0 \end{pmatrix}$.

Taylor series

In some applications it is beneficial, that the maximal order of a polynomial does not increase too much. In this case it may help to approximate a polynomial with a high maximal order by a polynomial of lower order. A common approach is the Taylor series expansion, to approximate a function by a power series, [14]. Here it is investigated how a polynomial $h(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ in tensor format can be approximated by a Taylor series of order 2. In

general a function $h(\mathbf{x})$ is approximated around a point $\mathbf{x}_0$ by its Taylor series of order 2, [14]

$$h(\mathbf{x}) \approx t(\mathbf{x}) = h(\mathbf{x}_0) + \mathbf{J}_h(\mathbf{x}_0)\,(\mathbf{x}-\mathbf{x}_0) + \frac{1}{2}\,(\mathbf{x}-\mathbf{x}_0)^T\,\mathbf{H}_h(\mathbf{x_0})\,(\mathbf{x}-\mathbf{x}_0) \tag{2.76}$$

$$= \underbrace{h(\mathbf{x}_0) - \mathbf{J}_h(\mathbf{x}_0)\mathbf{x}_0 + \frac{1}{2}\mathbf{x}_0^T\mathbf{H}_h(\mathbf{x}_0)\mathbf{x}_0}_{\text{Constant terms}} + \underbrace{\left(\mathbf{J}_h(\mathbf{x}_0) - \mathbf{x}_0^T\mathbf{H}_h\right)\mathbf{x}}_{\text{Linear terms}} + \underbrace{\frac{1}{2}\mathbf{x}^T\mathbf{H}_h(\mathbf{x}_0)\mathbf{x}}_{\substack{\text{Multilinear and}\\ \text{quadratic terms}}} .$$

The 2nd order Taylor series consists of constant, linear, mutlilinear and quadratic terms. If the polynomial

$$h(\mathbf{x}) = \left\langle\, \mathsf{H} \,\middle|\, \mathsf{M}_p^N\,(\mathbf{x}) \,\right\rangle$$

is given in a tensor structure, the Jacobian $\mathbf{J}_h(\mathbf{x}) \in \mathbb{R}^{1\times n}$ and the Hessian $\mathbf{H}_h(\mathbf{x}) \in \mathbb{R}^{n\times n}$ are computed by (2.68) and (2.74). With that the Taylor series can be expressed in tensor format too by

$$t(\mathbf{x}) = \left\langle\, \mathsf{T}_h \,\middle|\, \mathsf{M}_p^2\,(\mathbf{x}) \,\right\rangle, \tag{2.77}$$

because it is a polynomial of maximal order 2. The parameters of corresponding terms have to be arranged in the right positions of the parameter tensor $\mathsf{T}_h \in \mathbb{R}^{\times^{(2n)}2}$. The constant term is stored at

$$t_h(1,\ldots,1) = h(\mathbf{x}_0) - \mathbf{J}_h(\mathbf{x}_0)\mathbf{x}_0 + \frac{1}{2}\mathbf{x}_0^T\mathbf{H}_h(\mathbf{x}_0)\mathbf{x}_0. \tag{2.78}$$

The index vector $\mathbf{i}_{x,j} \in \mathbb{R}^{2n},\ j = 1,\ldots,n$ of the parameters belonging to the linear monomials x_j are

$$\mathbf{i}_{x,j}(k) = \begin{cases} 2 & \text{, for } k = n-j+1, \\ 1 & \text{, else.} \end{cases}$$

The parameters of the linear terms are stored in the corresponding entries of the parameter tensor

$$t_h(\mathbf{i}_{x,j}) = -\mathbf{h}_h(\mathbf{x}_0)(j,:)\mathbf{x}_0 + j_h(\mathbf{x}_0)(1,j). \tag{2.79}$$

The parameters of the multilinear and quadratic terms $x_j x_l, \forall\ j,l = 1,\ldots,n$, are stored at positions

$$\mathbf{i}_{x,j,l} = \begin{cases} 2 & \text{, for } k = n-j+1, \\ 2 & \text{, for } k = n-l+1, j \neq l, \\ 2 & \text{, for } k = 2n-l+1, j = l, \\ 1 & \text{, else,} \end{cases}$$

of the parameter tensor T_h of the Taylor series

$$t_h(\mathbf{i}_{x,j,l}) = \begin{cases} h_h(\mathbf{x}_0)(j,l) & \text{, for } j \neq l, \\ \frac{1}{2}h_h(\mathbf{x}_0)(j,l) & \text{, for } j = l. \end{cases} \tag{2.80}$$

This arrangement of terms of (2.76) in the parameter tensor T_h, results in the Taylor series of second order in tensor form (2.77). To summarize the derivation, the main result is given in the next lemma.

Lemma 2.4 (Second order taylor series in tensor form) *Consider a scalar polynomial in n variables and maximal order N*

$$h(\mathbf{x}) = \left\langle \mathsf{H} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle.$$

The second order taylor series approximation of $h(\mathbf{x}) \approx t(\mathbf{x})$ at the point $\mathbf{x}_0$ is written as

$$t(\mathbf{x}) = \left\langle \mathsf{T}_h \,\middle|\, \mathsf{M}_p^2(\mathbf{x}) \right\rangle$$

where the elements of the parameter tensor $\mathsf{T}_h \in \mathbb{R}^{\times^{(2n)}2}$ are given by (2.78) to (2.80).

Example 2.38 *Consider a polynomial of maximal order 3 in 2 variables, where the tensor notation of the function, its Jacobian and its Hessian are given by (2.47), (2.68) and (2.74)*

$$\begin{aligned}
h(\mathbf{x}) &= \left\langle \mathsf{H} \,\middle|\, \mathsf{M}_p^3(x_1, x_2) \right\rangle = x_2 + 2x_1x_2^2 - 3x_1^3x_2, \\
\mathbf{J}_h(\mathbf{x}) &= \left\langle \mathsf{J}_h \,\middle|\, \mathsf{M}_p^3(x_1, x_2) \right\rangle = \begin{pmatrix} 2x_2^2 - 9x_1^2x_2 & 1 + 4x_1x_2 - 3x_1^3 \end{pmatrix}, \\
\mathbf{H}_h(\mathbf{x}) &= \left\langle \mathsf{H}_h \,\middle|\, \mathsf{M}_p^3(x_1, x_2) \right\rangle = \begin{pmatrix} -18x_1x_2 & 4x_2 - 9x_1^2 \\ 4x_2 - 9x_1^2 & 4x_1 \end{pmatrix}.
\end{aligned}$$

The Taylor approximation of $h(\mathbf{x})$ should be computed around $\mathbf{x}_0 = \begin{pmatrix} 1 & 0 \end{pmatrix}^T$ and is written as

$$t(x_1, x_2) = \left\langle \mathsf{T}_h \,\middle|\, \mathsf{M}_p^2(x_1, x_2) \right\rangle.$$

To compute the parameters of the Taylor series, i.e. the elements of $\mathsf{T}_h \in \mathbb{R}^{2\times2\times2\times2}$ the function, the Jacobian and the Hessian have to be evaluated at $\mathbf{x}_0$

$$\begin{aligned}
h(\mathbf{x}_0) &= \left\langle \mathsf{H} \,\middle|\, \mathsf{M}_p^3(1, 0) \right\rangle = 0, \\
\mathbf{J}_h(\mathbf{x}_0) &= \left\langle \mathsf{J}_h \,\middle|\, \mathsf{M}_p^3(1, 0) \right\rangle = \begin{pmatrix} 0 & -2 \end{pmatrix}, \\
\mathbf{H}_h(\mathbf{x}_0) &= \left\langle \mathsf{H}_h \,\middle|\, \mathsf{M}_p^3(1, 0) \right\rangle = \begin{pmatrix} 0 & -9 \\ -9 & 4 \end{pmatrix}.
\end{aligned}$$

Using these evaluations the elements of T_h are computed and arranged by (2.78) to (2.80), such that the parameters belong to the right monomials

$$\begin{aligned}
1 \to t_h(1,1,1,1) &= h(\mathbf{x_0}) - \mathbf{J}_h(\mathbf{x}_0)\mathbf{x}_0 + \frac{1}{2}\mathbf{x}_0^T\mathbf{H}_h(\mathbf{x}_0)\mathbf{x}_0 = 0, \\
x_1 \to t_h(2,1,1,1) &= -\mathbf{h}_h(\mathbf{x}_0)(1,:)\mathbf{x}_0 + j_h(\mathbf{x}_0)(1,1) = 0, \\
x_2 \to t_h(1,2,1,1) &= -\mathbf{h}_h(\mathbf{x}_0)(2,:)\mathbf{x}_0 + j_h(\mathbf{x}_0)(1,2) = 7, \\
x_1x_2 \to t_h(2,2,1,1) &= h_h(\mathbf{x}_0)(1,2) = -9 \\
x_1^2 \to t_h(2,1,2,1) &= \frac{1}{2}h_h(\mathbf{x}_0)(1,1) = 0, \\
x_2^2 \to t_h(1,2,1,2) &= \frac{1}{2}h_h(\mathbf{x}_0)(2,2) = 2.
\end{aligned}$$

For clarification the monomials corresponding to the particular parameters are shown in the left column. All other elements of the parameter tensor are equal to zero. With the evaluation of the defined parameter tensor and the monomial tensor, the taylor approximation yields

$$t(\mathbf{x}) = \left\langle \mathsf{T}_h \;\middle|\; \mathsf{M}_p^2\left(x_1, x_2\right) \right\rangle = 7x_2 - 9x_1x_2 + 2x_2^2.$$

2.4 Open and guiding questions

In this chapter basics on tensors and decomposition techniques have been introduced. It has been shown, how tensor operations are computed efficiently by CP tensors. Furthermore tensors have been used to describe polynomials of arbitrary order. Approaches have been derived to compute polynomial standard operations based on this tensor format.

But the chapter leaves open, which decomposition method is suited best to represent polynomials. The storage effort of the parameter tensor gets very large for polynomials with many variables and of high order. It would be interesting to see, how the storage effort of the parameter tensors can be reduced by decomposition methods. For modeling it is of interest to use the tensor representation as right hand sides of differential equation systems. It has to be investigated how low-rank approximations, that approximate the function parameters, affect the dynamical behavior of these models. Since the chapter derives, that polynomial operations can be expressed in the tensor format, this poses the question, how these methods can be applied for system analysis or controller design. Here just some operations are formulated in the tensor framework. It is still open how other polynomial operations like the integration can be expressed in a similar way.

In the previous chapter one possible representation of polynomials by tensors has been introduced based on the notation of multilinear functions with some nice properties for algorithm development, which allowed to describe different polynomial operations like multiplication or differentiation based on the tensor framework. The monomial tensor has been constructed as rank-1 tensor extending the concept of the monomial tensor for multilinear functions. This leads to redundancies in the monomial tensor for higher order polynomials as can be seen in Example 2.31. This representation is not memory efficient, since the redundancies are not necessary for the polynomial description. It results from the basic multilinear monomial tensor and helps for the first development of the introduced operations. The number of dimensions of the monomial tensor also prescribes the number of dimensions of the parameter tensor. For systems with a large number of variables and a very high order, this could lead to problems in memory demand, even though the parameter tensor is in decomposed structure too. It is left open here, how a monomial tensor is constructed with less or without redundancies, e.g. by a variable transformation. This would lead to an even more memory efficient polynomial representation. The polynomial operations, given in this chapter, would have to be adapted to this structure as well. If such a representation is found, this would allow together with the proposed decomposition strategies to represent even larger polynomials and thus opens a wider range of applications.

3 Modeling for multilinear systems

Models of multilinear time-invariant (MTI) systems were introduced in [63] and [85] and its applicability to heating systems was shown. A model representation with decomposed tensors was proposed by using the CP decompositions of the parameter tensors. Section 3.1 gives an overview on the basic concept of the tensor representation of MTI models. As proposed in the previous chapter from the field of mathematics different decomposition techniques are available. Up to now in the literature the focus was on the CP decomposition for MTI model representation. Thus, Section 3.2 investigates how it is possible to represent and simulate MTI system in the four proposed decomposition techniques and compares the different formats. Since no widespread theory for MTI systems is available, as e.g. for linear systems, [4], the Sections 3.3 to 3.6 introduce different tools for MTI systems analysis, like linearization or discretization, based on the decomposed tensor format. For large-scale systems, that are composed of several MTI subsystems a notation is introduced in Section 3.7 and serial, parallel and feedback connections between a special subclass of MTI subsystems are investigated. Section 3.8 summarizes open questions in this area of MTI system representation.

3.1 Multilinear time-invariant (MTI) systems

The basic concept of multilinear models as it is introduced in the literature by [63] or [85] is briefly summarized in the following section. At first the model class is classified in the model hierarchy in Subsection 3.1.1. The model classes focused here are restricted to time-invariant, deterministic, lumped-parameter systems. After that Subsection 3.1.2 creates a link to the previous chapter by describing the tensor representation of multilinear systems. Guiding questions focusing on open points in the theory of multilinear systems, that will be answered in the following chapters, are posed in Subsection 3.1.3.

3.1.1 Model classes

Models of real systems help to predict the future behavior of these dynamical processes. This could be beneficial e.g. for tests of new system components, fault detection, controller design and many other applications. Figure 3.1 shows a general system with its input and output signals, [43].

A system is manipulated from the outside by inputs $\mathbf{u} \in \mathbb{R}^m$. Output signals $\mathbf{y} \in \mathbb{R}^p$ are measured. All relevant information on the system at the current time step is given by its states $\mathbf{x} \in \mathbb{R}^n$. All signals are supposed to be in the real domain here, since no switching or hybrid systems are considered, where signals in the Boolean domain could occur.

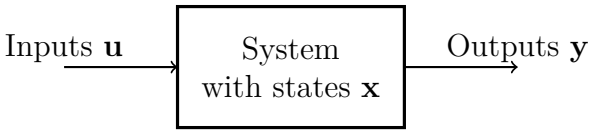


Figure 3.1: System with inputs, outputs and states

At first, it is assumed, that the dynamical behavior of the system does not change over time, i.e. it can be described by a time-invariant model. Thus, the parameters are constant and not time depended. The second assumption is, that the dynamical behavior of the system in continuous-time can be described by ordinary differential equations. Systems, e.g. modeled by partial differential equations, are not considered in this work. As a common system representation state space models are used here, where the relationship between $\mathbf{x}$, $\mathbf{u}$ and $\mathbf{y}$ is described by a system of first order ODEs. The state vector is used to transform also higher order ODEs in a system of first order differential equations, i.e. a state space model. The output is computed as a function of states and inputs. State space models of different model classes will be considered here, [43].

Nonlinear models

The most general class is the class of nonlinear models, that contains all the other model classes, that will be introduced in the following, [43]. A nonlinear continuous-time state space model is written as system of first order ODEs by

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \tag{3.1}$$
$$\mathbf{y}(t) = \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t)), \tag{3.2}$$

with time $t \in \mathbb{R}$, state equation $\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ and output equation $\mathbf{g} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^p$. The inital state is given by $\mathbf{x}_0 = \mathbf{x}(0)$. In discrete-time the system is sampled at fixed time steps T_{sample} leading to the description of the state space model by difference equations

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k)), \tag{3.3}$$
$$\mathbf{y}(k) = \mathbf{g}(\mathbf{x}(k), \mathbf{u}(k)), \tag{3.4}$$

where $\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ and $\mathbf{g} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^p$ are the next state and output function respectively and $k = 0, 1, 2, \ldots \in \mathbb{N}$ denotes the time index for the time steps kT_{sample}. Take into account, that the function $\mathbf{f}$ of the continous system is different to the function $\mathbf{f}$ of its discrete-time version. The basic structure of the continuous- and the discrete-time models are similar except the left hand sides of the equation. Therefore, many methods are applicable for continuous- as well as for discrete-time models. Thus, it is beneficial to summarize their descriptions by introducing the operator $\Phi(\mathbf{x})$, [63]. In the continuous case $\Phi(\mathbf{x})$ describes the time derivative $\dot{\mathbf{x}}$ of $\mathbf{x}$ and in the discrete setting the next state $\mathbf{x}(k+1)$. For ease of notation the time t or k will not be be mentioned explicitly in each equation.

With a given initial value $\mathbf{x}_0 = \mathbf{x}(0)$ and input trajectory the evolution of the state and the output can be uniquely determined by (3.1) and (3.2) in continuous-time and (3.3) and (3.4) in discrete-time. This computation of the future behavior is called simulation of

the model and is calculated by intgeration routines for ODEs for the continuous-time case or recursicvely for the discrete-time case, [43].

Polynomial models

The previously introduced class of nonlinear models allows arbitrary functions as right hand sides of the differential or difference equations. For polynomial systems the right hand side functions are limited to polynomials of arbitrary order. Exponential or sinusoidal functions for example are not included. The right hand sides of polynomial systems can be described by a linear combination of the polynomial monomials with a maximal order N. The monomials contain all the multiplicative combinations of states $\mathbf{x}$ and inputs $\mathbf{u}$ up to the maximal order. By using (2.41) the monomials are constructed by

$$\mathbf{m}_p^N(\mathbf{x},\mathbf{u}) = \bigotimes_{j=1}^{N}\left[\begin{pmatrix}1\\u_m\end{pmatrix}\otimes\cdots\otimes\begin{pmatrix}1\\u_1\end{pmatrix}\otimes\begin{pmatrix}1\\x_n\end{pmatrix}\otimes\cdots\otimes\begin{pmatrix}1\\x_1\end{pmatrix}\right], \tag{3.5}$$

resulting in a vector of dimension $\mathbb{R}^{2^{(n+m)N}}$. A linear combination of the monomials with the parameters of the system results in the right hand sides of the state space description, which is computed by the product of a matrix containing all the parameters of the system and the polynomial monomial vector (3.5). Thus, the polynomial state space model in matrix representation is given by

$$\Phi(\mathbf{x}) = \mathbf{F}\mathbf{m}_p^N(\mathbf{x},\mathbf{u}), \tag{3.6}$$

$$\mathbf{y} = \mathbf{G}\mathbf{m}_p^N(\mathbf{x},\mathbf{u}), \tag{3.7}$$

with the transition matrix $\mathbf{F} \in \mathbb{R}^{n\times 2^{(n+m)N}}$ and the output matrix $\mathbf{G} \in \mathbb{R}^{p\times 2^{(n+m)N}}$. This approach for polynomials of arbitrary order can be used to describe multilinear state space models by limiting the maximal order to $N = 1$.

Multilinear models

State space models of multilinear systems are described by multilinear functions depending on states and inputs as right hand sides as introduced in [64], [85] and [86]. This means that the function is linear if all but one state or input is held constant. Thus, no quadratic or higher order terms occur in the monomials, which is achieved by setting the maximal order of the polynomial monomial vector (3.5) to $N = 1$. The multilinear multiplicative combinations of states $\mathbf{x}$ and inputs $\mathbf{u}$ are contained in the monomial vector

$$\mathbf{m}(\mathbf{x},\mathbf{u}) = \begin{pmatrix}1\\u_m\end{pmatrix}\otimes\ldots\otimes\begin{pmatrix}1\\u_1\end{pmatrix}\otimes\begin{pmatrix}1\\x_n\end{pmatrix}\otimes\cdots\otimes\begin{pmatrix}1\\x_1\end{pmatrix} \in \mathbb{R}^{2^{n+m}}.$$

With that the matrix representation of a multilinear state space model is given by

$$\Phi(\mathbf{x}) = \mathbf{F}\mathbf{m}(\mathbf{x},\mathbf{u}), \tag{3.8}$$

$$\mathbf{y} = \mathbf{G}\mathbf{m}(\mathbf{x},\mathbf{u}), \tag{3.9}$$

where $\mathbf{F} \in \mathbb{R}^{n\times 2^{n+m}}$ is the transition matrix and $\mathbf{G} \in \mathbb{R}^{m\times 2^{n+m}}$ is the output matrix.

Example 3.1 *The state equation of a multilinear state space model with one input and two states in matrix representation is*

$$\begin{aligned}
\Phi\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) &= \mathbf{Fm}\left(x_1, x_2, u_1\right) \\
&= \begin{pmatrix} f(1,1) & f(1,2) & f(1,3) & f(1,4) & f(1,5) & f(1,6) & f(1,7) & f(1,8) \\ f(2,1) & f(2,2) & f(2,3) & f(2,4) & f(2,5) & f(2,6) & f(2,7) & f(2,8) \end{pmatrix} \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ x_1x_2 \\ u_1 \\ x_1u_1 \\ x_2u_1 \\ x_1x_2u_1 \end{pmatrix} \\
&= \begin{pmatrix} f(1,1) + f(1,2)x_1 + f(1,3)x_2 + f(1,4)x_1x_2 + \\ f(2,1) + f(2,2)x_1 + f(2,3)x_2 + f(2,4)x_1x_2 + \end{pmatrix} \dots \\
&\qquad \begin{pmatrix} f(1,5)u_1 + f(1,6)x_1u_1 + f(1,7)x_2u_1 + f(1,8)x_1x_2u_1 \\ f(2,5)u_1 + f(2,6)x_1u_1 + f(2,7)x_2u_1 + f(2,8)x_1x_2u_1 \end{pmatrix}.
\end{aligned}$$

Bilinear models

A subclass of multilinear models are bilinear models. Models of this class are linear in the states and inputs but have a nonlinear extension. In the nonlinear part products of state and input signals are involved. This is a limitation compared to multilinear models, where products of states and states or inputs and inputs are included as well. A bilinear state space model is written as

$$\begin{aligned}
\Phi\left(\mathbf{x}\right) &= \mathbf{Ax} + \mathbf{E}\left(\mathbf{x} \otimes \mathbf{u}\right) + \mathbf{Bu}, \\
\mathbf{y} &= \mathbf{Cx} + \mathbf{Du},
\end{aligned}$$

with system matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, input matrix $\mathbf{B} \in \mathbb{R}^{n \times m}$, output matrix $\mathbf{C} \in \mathbb{R}^{p \times n}$ and feedthrough matrix $\mathbf{D} \in \mathbb{R}^{p \times m}$ of the linear part and the matrix $\mathbf{E} \in \mathbb{R}^{n \times nm}$ belonging to the bilinear terms $x_i u_j,\ i = 1, \dots, n,\ j = 1, \dots, m$. The following example shows that bilinear models can be formulated as multilinear models, i.e. the class of bilinear models is contained in the multilinear class.

Example 3.2 *A state space model of a bilinear system with two states and one input can be formulated as a multilinear system by*

$$\begin{aligned}
\Phi\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) &= \begin{pmatrix} 0 & a(1,1) & a(1,2) & 0 & b(1) & e(1,1) & e(1,2) & 0 \\ 0 & a(2,1) & a(2,2) & 0 & b(2) & e(2,1) & e(2,2) & 0 \end{pmatrix} \mathbf{m}\left(x_1, x_2, u_1\right) \\
&= \begin{pmatrix} a(1,1)x_1 + a(1,2)x_2 + b(1)u_1 + e(1,1)x_1u_1 + e(1,2)x_2u_1 \\ a(2,1)x_1 + a(2,2)x_2 + b(2)u_1 + e(2,1)x_1u_1 + e(2,2)x_2u_1 \end{pmatrix}.
\end{aligned}$$

Comparing the parameter matrix of the bilinear model with the parameter matrix of the general multilinear model of same dimensions of Example 3.1 shows, that the parameters of the multilinear and not bilinear monomials 1, x_1x_2 *and* x_1x_2u *are set to zero in the bilinear case. Thus, the multilinear class extends the bilinear.*

Linear models

Linear models are very common in control engineering, because of their simple handling e.g. for system analysis or controller synthesis. But they have the drawback, that the class is very restrictive since only linear combinations of states and inputs are involved leading to the state space representation

$$\Phi(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}, \tag{3.10}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}, \tag{3.11}$$

with system matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, input matrix $\mathbf{B} \in \mathbb{R}^{n\times m}$, output matrix $\mathbf{C} \in \mathbb{R}^{p\times n}$ and feedthrough matrix $\mathbf{D} \in \mathbb{R}^{p\times m}$. Compared to multilinear models, the set of monomials for linear models is reduced. No multiplicative combinations of signals are allowed at all. Because of these restrictions linear systems are a subclass of multilinear systems and can be expressed in the multilinear framework, too.

Example 3.3 *A linear system with two states and one input has a state equation rearranged in the MTI framework of*

$$\Phi\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) = \begin{pmatrix} 0 & a(1,1) & a(1,2) & 0 & b(1) & 0 & 0 & 0 \\ 0 & a(2,1) & a(2,2) & 0 & b(2) & 0 & 0 & 0 \end{pmatrix} \mathbf{m}(x_1, x_2, u_1) = \begin{pmatrix} a(1,1)x_1 + a(1,2)x_2 + b(1)u_1 \\ a(2,1)x_1 + a(2,2)x_2 + b(2)u_1 \end{pmatrix}.$$

The restrictions of the linear class gets obvious, when focusing on the zeros in the parameter matrix. All the parameters of monomials except x_1, x_2 and u are set to zero.

Overview

An overview of the different introduced classes is depicted in Figure 3.2.

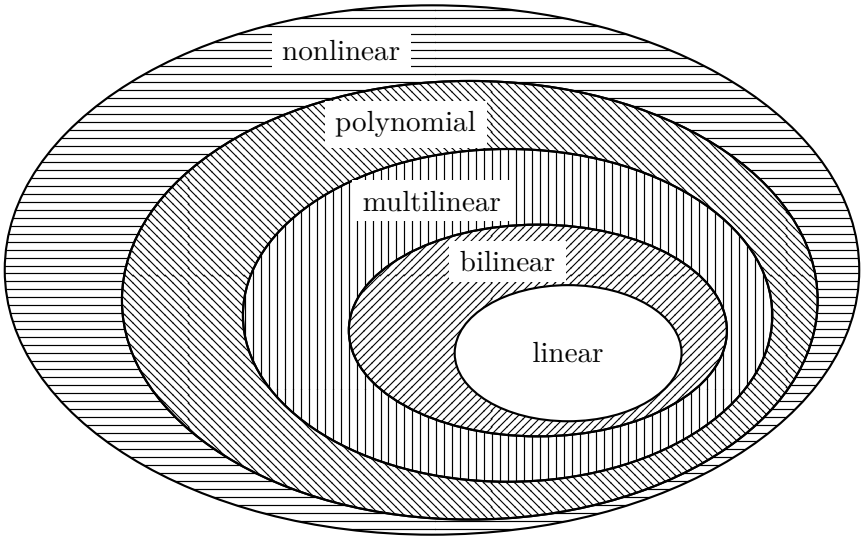


Figure 3.2: System classes

The most general class are nonlinear models including arbitrary polynomial terms and other nonlinear functions like exponential or sinusoidal functions. All other previously introduced classes are subclasses of the nonlinear one. This very general formulation allows high flexibility in modeling but leads to a high complexity, when focusing on system analysis or controller design since no structure is imposed on the model at all. Thus, tools for analysis and design are often mathematically complex and limited to certain categories of nonlinear systems, [43].
The most restrictive class are linear models. This class is well established and has a widespread theory. The linearity property of these systems helps to develop algorithms for modeling and design. But in cases where nonlinear effects are essential, methods for linear models may fail, because linear models cannot capture the system dynamics in a sufficient accuracy. That is the drawback of their simplicity.
That is why systems between the complex nonlinear and the simple linear systems are of interest. Bilinear, multilinear or polynomial systems can model more complex dynamics than linear systems, since they include more monomials on the right hand sides of their ODE descriptions. But these three classes still impose a structure on their model equations and are not as general as in the nonlinear case. This is an advantage, when developing algorithms for modeling or controller synthesis. Usually specialization on a specific class with a given system structure lowers complexity and improves robustness of the design process. One of the main guidelines in modeling is to make the model as simple as possible but as complex as necessary to get a good model accuracy, [68].
From an application point of view the focus in this work is on heating systems. As shown in the very simple example in Section 1.4 already for small subsystems of heating systems the bilinear class is not sufficient to capture the dynamics. Thus, the linear as well as the bilinear class are too restrictive for heating systems, [85, 86]. The multilinear terms follow from the multiplication of temperature and flow in the heat balances of the heating systems model. In general, one can say that systems modeled by heat or mass balances like heating systems or chemical systems often show such structures, [50, 86]. Polynomial systems can also capture these dynamics since they build a superclass of multilinear systems. But these applications do not show higher order behavior very often. To keep the model as simple as possible it is assumed that the higher order terms do not have a major affect on the system dynamics. Therefore, multilinear systems are focused on in this work for modeling and controller design. As illustrated in Examples 3.2 and 3.3 linear and bilinear models are contained in this class, which shows that simpler dynamics can be captured too by MTI systems. Even though the considered systems might have no inherent multilinear structure, it was shown that their system behavior can be approximated adequately by an MTI model, [50].

3.1.2 Tensor representation

In the previous section a state space model of an MTI system (3.8) and (3.9) was introduced in matrix notation. The right hand sides of the state space model are given by multilinear functions that are computed by a product of parameter matrices and monomial vectors. Besides the matrix format (2.40), multilinear functions can be described in a tensor framework (2.44). This tensor concept can be applied here to describe the right hand sides of

the MTI systems as introduced in [64] or [86]. The tensor description offers the possibility to use methods from the tensor calculus presented in Section 2, e.g. to reduce the complexity of large-scale systems by tensor decompositions. The tensor representation of MTI systems will be investigated in the following sections. The monomial tensor (2.43) arranges the monomials in a tensor. It depends on the states and the inputs in the state space setting

$$\begin{aligned}\mathsf{M}(\mathbf{x},\mathbf{u}) &= \begin{pmatrix}1\\u_m\end{pmatrix}\circ\begin{pmatrix}1\\u_{m-1}\end{pmatrix}\circ\cdots\circ\begin{pmatrix}1\\u_1\end{pmatrix}\circ\begin{pmatrix}1\\x_n\end{pmatrix}\circ\begin{pmatrix}1\\x_{n-1}\end{pmatrix}\circ\cdots\circ\begin{pmatrix}1\\x_1\end{pmatrix}\\ &= \left[\begin{pmatrix}1\\u_m\end{pmatrix},\ldots,\begin{pmatrix}1\\u_1\end{pmatrix},\begin{pmatrix}1\\x_n\end{pmatrix},\ldots,\begin{pmatrix}1\\x_1\end{pmatrix}\right],\end{aligned} \tag{3.12}$$

and is still a rank-1 tensor of dimension $\mathbb{R}^{\times^{(n+m)}2}$. Thus, the right hand sides of a multilinear state space model is computed by a contracted product of a parameter and a monomial tensor leading to the MTI system in tensor format

$$\Phi(\mathbf{x}) = \langle\, \mathsf{F} \mid \mathsf{M}(\mathbf{x},\mathbf{u}) \,\rangle, \tag{3.13}$$

$$\mathbf{y} = \langle\, \mathsf{G} \mid \mathsf{M}(\mathbf{x},\mathbf{u}) \,\rangle, \tag{3.14}$$

with transition tensor $\mathsf{F} \in \mathbb{R}^{\times^{(n+m)}2\times n}$ and output tensor $\mathsf{G} \in \mathbb{R}^{\times^{(n+m)}2\times p}$. The parameter tensors F and G contain the same parameters than the matrices $\mathbf{F}$ and $\mathbf{G}$ of (3.8) and (3.9), but they are arranged now in a tensor according to the rearrangement of the monomials in the monomial tensor $\mathsf{M}(\mathbf{x},\mathbf{u})$.

Example 3.4 *The state equation of an MTI system with two states is given in the tensor framework by*

$$\Phi\left(\begin{pmatrix}\dot{x}_1\\\dot{x}_2\end{pmatrix}\right) = \langle\, \mathsf{F} \mid \mathsf{M}(x_1,x_2) \,\rangle = \left\langle \mathsf{F} \,\middle|\, \left[\begin{pmatrix}1\\x_2\end{pmatrix},\begin{pmatrix}1\\x_1\end{pmatrix}\right] \right\rangle$$

$$= \left\langle \begin{array}{ccc} f(1,1,2) & \text{——} & f(1,2,2) \\ | & & | \\ f(1,1,1) & \text{——} & f(1,2,1) \\ | & & | \\ f(2,1,2) & \text{——} & f(2,2,2) \\ | & & | \\ f(2,1,1) & \text{——} & f(2,2,1) \end{array} \,\middle|\, \begin{array}{ccc} 1 & \text{——} & x_1 \\ | & & | \\ x_2 & \text{——} & x_1x_2 \end{array} \right\rangle$$

$$= \begin{pmatrix} f(1,1,1) + f(1,2,1)x_1 + f(2,1,1)x_2 + f(2,2,1)x_1x_2 \\ f(1,1,2) + f(1,2,2)x_1 + f(2,1,2)x_2 + f(2,2,2)x_1x_2 \end{pmatrix}.$$

The description of the tensor notation of MTI systems follows from [63], [85] and [86]. Since the concept of the tensor representation for multilinear functions was extended to general polynomials in this work (2.47), the tensor description of MTI systems can be augmented too

to represent polynomial models. Analogous to (2.46) the monomial tensor of a polynomial system of maximal monomial order N with states $\mathbf{x}$ and inputs $\mathbf{u}$ is written as tensor of dimension $\mathbb{R}^{\times^{N(n+m)}2}$

$$\begin{aligned}\mathsf{M}_p^N(\mathbf{x},\mathbf{u}) &= \bigcirc_{i=1}^{N} \mathsf{M}(\mathbf{x},\mathbf{u}) = \bigcirc_{i=1}^{N} \begin{pmatrix}1\\u_m\end{pmatrix} \circ \cdots \circ \begin{pmatrix}1\\u_1\end{pmatrix} \circ \begin{pmatrix}1\\x_n\end{pmatrix} \circ \cdots \circ \begin{pmatrix}1\\x_1\end{pmatrix} \\ &= \underbrace{\left[\begin{pmatrix}1\\u_m\end{pmatrix}, \ldots, \begin{pmatrix}1\\x_1\end{pmatrix}, \ldots\ldots, \begin{pmatrix}1\\u_m\end{pmatrix}, \ldots, \begin{pmatrix}1\\x_1\end{pmatrix}\right]}_{N \text{ times}}.\end{aligned}$$

As mentioned before, this derivation of the monomials is not unique but the representation as rank-1 tensor is beneficial for the development of the algorithms. By rearranging the parameters of the polynomial system in matrix representation (3.6) and (3.7) in the same way than the monomials in $\mathsf{M}_p^N(\mathbf{x},\mathbf{u})$ one gets the tensor representation of polynomial models of maximal order N

$$\Phi(\mathbf{x}) = \left\langle \mathsf{F} \,\middle|\, \mathsf{M}_p^N(\mathbf{x},\mathbf{u}) \right\rangle, \tag{3.15}$$

$$\mathbf{y} = \left\langle \mathsf{G} \,\middle|\, \mathsf{M}_p^N(\mathbf{x},\mathbf{u}) \right\rangle, \tag{3.16}$$

that is computed by the contracted product of a transition tensor $\mathsf{F} \in \mathbb{R}^{\times^{N(n+m)}2\times n}$ and output tensor $\mathsf{G} \in \mathbb{R}^{\times^{N(n+m)}2\times p}$ with the monomial tensor $\mathsf{M}_p^N(\mathbf{x},\mathbf{u})$. The number of dimensions of the tensors increases with the product of order and number of states and inputs.

Example 3.5 *Due to the limitations of the visualization of tensors to three dimensions, a polynomial model with one state of maximal order two of the polynomials is considered here as an example, that has a monomial tensor*

$$\mathsf{M}_p^2(x_1) = \begin{pmatrix}1\\x_1\end{pmatrix} \circ \begin{pmatrix}1\\x_1\end{pmatrix} = \begin{pmatrix}1 & x_1\\ x_1 & x_1^2\end{pmatrix}.$$

With this monomial tensor the state equation of the polynomial system in tensor representation is given as

$$\begin{aligned}\Phi(x_1) = \left\langle \mathsf{F} \,\middle|\, \mathsf{M}_p^N(x_1) \right\rangle &= \left\langle \begin{pmatrix}f(1,1) & f(1,2)\\ f(2,1) & f(2,2)\end{pmatrix} \,\middle|\, \begin{pmatrix}1 & x_1\\ x_1 & x_1^2\end{pmatrix} \right\rangle \\ &= f(1,1) + (f(1,2) + f(2,1))\, x_1 + f(2,2)x_1^2.\end{aligned}$$

3.1.3 Guiding questions

The following sections deal with modeling and controller synthesis of MTI systems. In today's applications systems get more and more complex, like in smart grids or heating systems. Plants in these application areas can be modeled by MTI systems with a large number of states. With dense parameter tensors these models can capture very complex dynamics. But a large order n leads to a very large number of system parameters in full representation. The number of parameters of a MTI system depends on the numbers of

elements in the tensors F and G and increases exponentially with the numbers of states and inputs, i.e. the transition tensor F contains $n \cdot 2^{n+m}$ elements. Systems can also be represented in a symbolical way. But with the assumption of a system with non-sparse parameter tensors the number of terms also in a symbolic representation increases exponentially with the number of states and inputs. Thus, for large-scale systems a system description is impossible, because of the curse of dimensionality and leads to problems in memory demand. Figure 3.3 shows in a logarithmic scale, how the number of parameters increases with the number of inputs and states of the system, for a full representation.

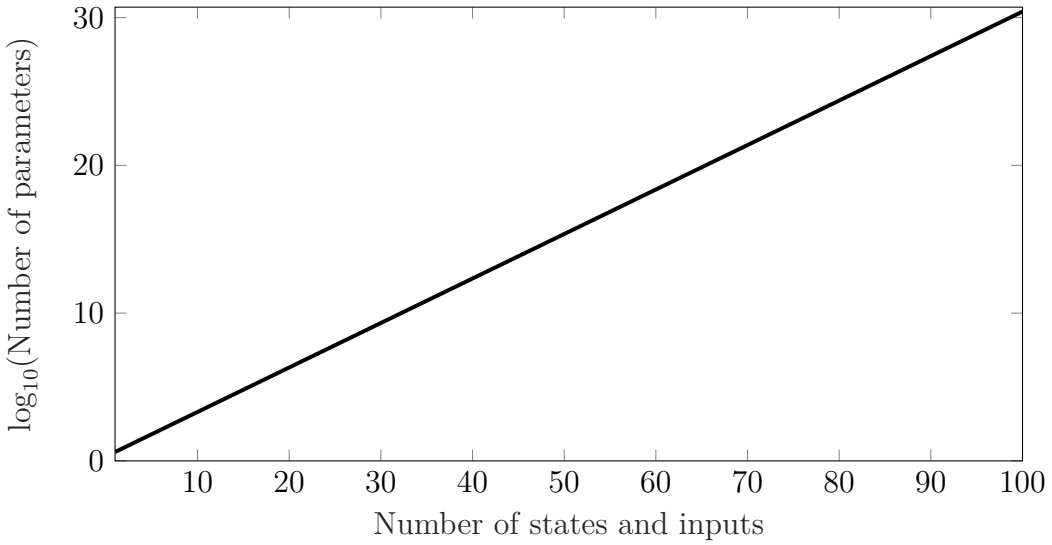


Figure 3.3: Number of parameters for MTI systems in full representation

A system with 100 states and inputs has more than 10^{32} terms in full representation, which does not fit in memory anymore. The state space representation and thus the computation of a controller of such big systems is not possible in the proposed representations by full tensors or in a symbolical way, if the parameter tensor is dense. Tensor decomposition methods as introduced in Section 2.1.2 offer the possibility to compute low-rank approximations of the parameter tensors, [64]. This could break the curse of dimensionality and maybe give an approximative representation of large-scale systems with a number of parameters, that is by orders of magnitude lower.

Low-rank approximation techniques for parameter tensors could make it possible to represent large systems and offer the possibility to compute controllers for such systems. This raises the question which decomposition methods are suitable to represent MTI systems and how to determine decomposed representations from the state equations directly in a structured way. The next Section 3.2 investigates the application of the four decomposition methods from Section 2.1.2 to MTI systems.

As already mentioned for linear systems many methods in modeling are available, [63]. For MTI systems this is not the case. In [63] the whole class of multilinear systems was introduced for the first time. In [85] the representation of hybrid systems by multilinear models and the application of MTI models to heating systems was investigated. The focus

was more on system representation in the tensor framework and controller design. Thus, since MTI systems are a current field of research, standard tools for this class of systems like for linear systems are not available. With (3.13) and (3.14) a structured representation for MTI systems is available. Here general tools for MTI systems are derived like linearization, discretization or scaling of the model. If it is possible to represent the model in decomposed structure, such tools should work with the decomposition factors to get efficient algorithms. The development of tools that are necessary for synthesis of controllers for MTI systems is investigated in Sections 3.3 to 3.7.

3.2 Decomposed MTI system representation and simulation

As shown in Figure 3.3, the number of parameters in the parameter tensors F and G of an MTI system in full representation increases exponentially with the number of states n and inputs m. This leads to a very high memory demand to store the parameter tensors of large-scale systems, as they occur often in today's applications. Section 3.1.2 introduced that the parameters of MTI systems can be arranged in a tensor format resulting in the state space formulation (3.13) and (3.14). From mathematics, tools are offered to reduce the storage effort of tensors by tensor decomposition methods enormously as suggested in Section 2.1.2. Since the parameters are given in a tensor framework, these decomposition techniques are used here to reduce the memory demand and make the MTI system representation computationally manageable in particular for large-scale systems. A decomposed system representation is used for computation either. Thus, the application of the decomposition methods leads besides an efficient model representation to efficient algorithms for system analysis and controller design. As introduced in (3.12) the monomial tensor is already a rank-1 tensor by construction, which is already minimal. In the following parts the application of the four introduced decomposition techniques, i.e. CP, Tucker, TT and HT decomposition, to the parameter tensors of an MTI system is investigated to reduce their memory demands. Therefore, the decomposition of the parameter tensor F of the state equation is shown here. The decomposition approaches can be applied to the output function and its parameter tensor G in the same way. Besides the system representation by decomposed tensors, the simulation of the decomposed systems is investigated in the following parts. Once the system is available with a decomposed parameter tensor, this tensor format can be used for simulation. During simulation the right hand sides of the state space system (3.13) and (3.14) have to be evaluated for the actual values of the states and inputs. In case of a constinuous-time model they are integrated numerically over time to calculate the state trajectories. Here, the efficient evaluation based on the decomposed structure of the right hand side is focused on. Therefore the contracted product of a decomposed parameter tensor and the rank-1 monomial tensor has to be computed. Standard algorithms are used for integration. The next subsections introduce the representation and simulation of MTI systems by using the four proposed decomposition techniques, [46].

3.2.1 Canonical polyadic decomposition

The parameter tensor F of the state equation is considered here to represent the parameters in CP format. The same approach is possible for the output tensor G. In [63] and [85] it was introduced that the parameters can be transformed directly term by term in a CP representation. This is illustrated by an example here.

Example 3.6 *A second order MTI system with one input*

$$\Phi(x_1) = a_1 + a_2 x_1 + a_3 u_1 x_1, \tag{3.17}$$
$$\Phi(x_2) = b_1 x_2 + b_2 x_2 x_1 + b_3 u_1 x_2 x_1, \tag{3.18}$$

has a CP transition tensor $\mathsf{F} = [\mathbf{F}_{u_1}, \mathbf{F}_{x_2}, \mathbf{F}_{x_1}.\mathbf{F}_\Phi] \cdot \boldsymbol{\lambda}_F$*, [47]. The construction of the factor matrices from the state equation is illustrated by*

$$\begin{array}{lccccccccccc}
\Phi(x_1) = & a_1 & + & a_2 x_1 & + & a_3 u_1 x_1, & & & & & & \\
\Phi(x_2) = & & & & & & b_1 x_2 & + & b_2 x_2 x_1 & + & b_3 u_1 x_2 x_1, & \\
\mathbf{F}_{x_1} = & \left(\begin{array}{c} 1 \\ 0 \end{array}\right. & & \begin{array}{c} 0 \\ 1 \end{array} & & \begin{array}{c} 0 \\ 1 \end{array} & \begin{array}{c} 1 \\ 0 \end{array} & & \begin{array}{c} 0 \\ 1 \end{array} & & \left.\begin{array}{c} 0 \\ 1 \end{array}\right), & \\
\mathbf{F}_{x_2} = & \left(\begin{array}{c} 1 \\ 0 \end{array}\right. & & \begin{array}{c} 1 \\ 0 \end{array} & & \begin{array}{c} 1 \\ 0 \end{array} & \begin{array}{c} 0 \\ 1 \end{array} & & \begin{array}{c} 0 \\ 1 \end{array} & & \left.\begin{array}{c} 0 \\ 1 \end{array}\right), & \\
\mathbf{F}_{u_1} = & \left(\begin{array}{c} 1 \\ 0 \end{array}\right. & & \begin{array}{c} 1 \\ 0 \end{array} & & \begin{array}{c} 0 \\ 1 \end{array} & \begin{array}{c} 1 \\ 0 \end{array} & & \begin{array}{c} 1 \\ 0 \end{array} & & \left.\begin{array}{c} 0 \\ 1 \end{array}\right), & \\
\mathbf{F}_{\Phi} = & \left(\begin{array}{c} 1 \\ 0 \end{array}\right. & & \begin{array}{c} 1 \\ 0 \end{array} & & \begin{array}{c} 1 \\ 0 \end{array} & \begin{array}{c} 0 \\ 1 \end{array} & & \begin{array}{c} 0 \\ 1 \end{array} & & \left.\begin{array}{c} 0 \\ 1 \end{array}\right), & \\
\boldsymbol{\lambda}_F = & \big(& a_1 & & a_2 & & a_3 & b_1 & & b_2 & & b_3 \big). &
\end{array}$$

The example shows, that the CP parameter tensor has one factor matrix for each state $\mathbf{F}_{x_i}$ with $i = 1, \ldots, n$ and input $\mathbf{F}_{u_j}$ with $j = 1, \ldots, m$. The factor matrices are build up columnwise as highlighted in Example 3.6. Each coefficient a_i or b_i belongs to one column of the factor matrices. The number of terms in the state equation is equal to the number of nonzero elements in F

$$\Psi_F = \operatorname{card}(\mathsf{F}),$$

where $\operatorname{card}(\cdot)$ is the cardinality function returning the number of nonzero entries in a tensor. Therefore each of the $n + m + 1$ factor matrices has Ψ_F columns. Each term of the state equation results in a column in each factor matrix. If the state x_i or input u_j is used in the term, the corresponding column of $\mathbf{F}_{x_i}$ or $\mathbf{F}_{u_j}$ is set to $\begin{pmatrix} 0 & 1 \end{pmatrix}^T$ otherwise it is set to $\begin{pmatrix} 1 & 0 \end{pmatrix}^T$. The corresponding coefficient is inserted to the weighting vector $\boldsymbol{\lambda}_F$. This leads to a weighting vector $\boldsymbol{\lambda}_F \in \mathbb{R}^{\Psi_F}$ and factor matrices $\mathbf{F}_{x_i}$ and $\mathbf{F}_{u_j}$ of dimension $\mathbb{R}^{2 \times \Psi_F}$. The last factor matrix $\mathbf{F}_\Phi$ indicates to which state equation $\Phi(x_i)$, $i = 1, \ldots, n$ the term belongs, i.e. $\Phi(x_1)$ or $\Phi(x_2)$ in this example. Since in the general case the state equation consists

of n scalar state equations the factor matrix $\mathbf{F}_\Phi$ has the dimension $\mathbb{R}^{n\times\Psi_F}$. Because of this direct translation every MTI system can be written in CP structure

$$\mathsf{F} = [\mathbf{F}_{u_m}, \ldots, \mathbf{F}_{u_1}, \mathbf{F}_{x_n}, \ldots, \mathbf{F}_{x_1}, \mathbf{F}_\Phi] \cdot \boldsymbol{\lambda}_F. \tag{3.19}$$

This direct translation of the system parameters to a CP representation is a big advantage here, [85]. In standard applications of tensor decompositions in mathematics the decomposed version is very often an approximation of the data, that has been stored in a tensor, [20]. Here the decomposed tensor represents exactly the parameters in the full tensor. But it is not guaranteed that the number Ψ_F of rank-1 components is the rank of the tensor, i.e. the minimal number of rank-1 components. Here an exact decomposition of F is found but it is not necessarily a minimal decomposition. With the CP representation (3.19) of the parameter tensor and the CP storage demand (2.12) an upper bound for the storage effort of an MTI system can be given

$$\begin{aligned}\xi_{cp}(\mathsf{F}) &\leq \Psi_F + (n+m)2\Psi_F + n\Psi_F \\ &= \Psi_F\left(1 + 3n + 2m\right),\end{aligned} \tag{3.20}$$

that results from the summation of the storage effort of the weighting vector and the factor matrices. By using low-rank approximation techniques the complexity of the parameter tensor may be further reduced depending on the application by finding approximations with less rank-1 components, but sufficient accuracy.

By having a CP representation of the parameter tensor the evaluation of the right hand sides of the state equation (3.13) can be computed very efficiently based on the decomposition factors, since the monomial tensor is rank-1. The contracted product of the CP parameter tensor and the monmial tensor is computed by a sequence of matrix products, [44, 63]

$$\dot{\mathbf{x}} = \mathbf{F}_\Phi\left(\boldsymbol{\lambda}_F \circledast \left(\mathbf{F}_{u_m}^T\begin{pmatrix}1\\u_m\end{pmatrix}\right)\circledast\cdots\circledast\left(\mathbf{F}_{u_1}^T\begin{pmatrix}1\\u_1\end{pmatrix}\right)\circledast\left(\mathbf{F}_{x_n}^T\begin{pmatrix}1\\x_n\end{pmatrix}\right)\circledast\cdots\circledast\left(\mathbf{F}_{x_1}^T\begin{pmatrix}1\\x_1\end{pmatrix}\right)\right). \tag{3.21}$$

Example 3.7 *With the decomposition factors of Example 3.6 the right hand side of the state equation is evaluated by*

$$\begin{aligned}\Phi\left(\begin{pmatrix}x_1\\x_2\end{pmatrix}\right) &= \langle\, \mathsf{F} \mid \mathsf{M}\left(x_1, x_2, u_1\right)\rangle = \mathbf{F}_\Phi\left(\boldsymbol{\lambda}_F \circledast \left(\mathbf{F}_{u_1}^T\begin{pmatrix}1\\u_1\end{pmatrix}\right)\circledast\left(\mathbf{F}_{x_2}^T\begin{pmatrix}1\\x_2\end{pmatrix}\right)\circledast\left(\mathbf{F}_{x_1}^T\begin{pmatrix}1\\x_1\end{pmatrix}\right)\right)\\
&= \begin{pmatrix}1&0\\1&0\\1&0\\0&1\\0&1\\0&1\end{pmatrix}\left(\begin{pmatrix}a_1\\a_2\\a_3\\b_1\\b_2\\b_3\end{pmatrix}\circledast\left(\begin{pmatrix}1&0\\1&0\\0&1\\1&0\\1&0\\0&1\end{pmatrix}\begin{pmatrix}1\\u_1\end{pmatrix}\right)\circledast\left(\begin{pmatrix}1&0\\1&0\\1&0\\0&1\\0&1\\0&1\end{pmatrix}\begin{pmatrix}1\\x_2\end{pmatrix}\right)\circledast\left(\begin{pmatrix}1&0\\0&1\\0&1\\1&0\\0&1\\0&1\end{pmatrix}\begin{pmatrix}1\\x_1\end{pmatrix}\right)\right)\\
&= \begin{pmatrix}a_1 + a_2x_1 + a_3u_1x_1\\ b_1x_2 + b_2x_2x_1 + b_3u_1x_2x_1\end{pmatrix}.\end{aligned}$$

The example shows, that the result is calculated by simple standard matrix products only. No complicated operations have to be computed. Low dimensional decomposition factors are used only. No full tensors have to be build, resulting in a computationally less demanding computation of the right hand sides than in the case of full tensors.

This CP form helps to create model representations with Tucker tensors, HT tensors or TTs.

3.2.2 Tucker decomposition

The CP representation of the parameter tensor can be used to construct one possible Tucker format. The Tucker tensor has a structure that is comparable to the CP one. The Tucker decomposition of an N dimensional tensor K (2.16) can be represented as, [44]

$$\mathsf{K} = \sum_{i_1=1}^{r_{t,1}(\mathsf{K})} \cdots \sum_{i_N=1}^{r_{t,N}(\mathsf{K})} \lambda(i_1,\ldots,i_N)\mathbf{x}_1(:,i_1)\circ\cdots\circ\mathbf{x}_N(:,i_N). \tag{3.22}$$

By choosing a diagonal core tensor $\Lambda \in \mathbb{R}^{\times^{(N)} r_{cp}(\mathsf{K})}$, i.e. only the elements $\lambda(i_1,\ldots,i_N)$ on the superdiagonal with $i_1 = i_2 = \cdots = i_N$ are nonzero, (3.22) can be reduced by eliminating the zero terms of the core tensor, leading to

$$\mathsf{K} = \sum_{i_1=1}^{r_{cp}(\mathsf{K})} \cdots \sum_{i_N=1}^{r_{cp}(\mathsf{K})} \lambda(i_1,\ldots,i_N)\mathbf{x}_1(:,i_1)\circ\cdots\circ\mathbf{x}_N(:,i_N) = \sum_{i=1}^{r_{cp}(\mathsf{K})} \lambda(i,\ldots,i)\mathbf{x}_1(:,i)\circ\cdots\circ\mathbf{x}_N(:,i).$$

Comparing the reduced equation with (2.10) shows the CP structure of the representation. Thus, if a CP representation is available, a Tucker tensor with same factor matrices can be constructed by taking the elements of the weighting vector as diagonal of the core tensor. This can be applied to the parameter tensor here too. To get one possible Tucker representation of F, that directly translates the CP factors to Tucker, the entries of the weighting vector $\boldsymbol{\lambda}_F$ of the CP decomposition have to be written as diagonal of the core tensor Λ_F. Its elements are given by

$$\Lambda_F(i_1,\ldots,i_{n+m+1}) = \begin{cases} \lambda_F(j) & ,\ \forall j = i_1 = \cdots = i_{n+m+1}, \\ 0 & ,\ \text{otherwise}, \end{cases} \tag{3.23}$$

where $j = 1,\ldots,\Psi_F$. This leads to a core size $\mathbb{R}^{\times^{(n+m+1)}\Psi_F}$, which depends on the number of terms Ψ_F of the state equation. As shown, by the choice of this core tensor the CP factor matrices can be used for the Tucker representation too, leading to

$$\mathsf{F} = [\mathbf{F}_{u_m},\ldots,\mathbf{F}_{u_1},\mathbf{F}_{x_n},\ldots,\mathbf{F}_{x_1},\mathbf{F}_\Phi]\cdot\Lambda_F.$$

Example 3.8 *The parameter tensor of the system introduced in Example 3.6 is given as Tucker tensor by*

$$\mathsf{F} = [\mathbf{F}_{u_1},\mathbf{F}_{x_2},\mathbf{F}_{x_1},\mathbf{F}_\Phi]\cdot\Lambda_F,$$

with the same factor matrices as derived for the CP form and the core tensor $\mathsf{\Lambda}_F \in \mathbb{R}^{6\times6\times6\times6}$ *with elements*

$$\Lambda_F(i_1,\ldots,i_4) = \begin{cases} a_j & \text{, for } j = i_1 = \cdots = i_4,\ \forall j = 1,2,3, \\ b_{j-3} & \text{, for } j = i_1 = \cdots = i_4,\ \forall j = 4,5,6, \\ 0 & \text{, else.} \end{cases}$$

With (2.18), the storage effort of the Tucker representation of the parameter tensor is given by the sum of the number of elements in the core tensor and the factor matrices

$$\begin{aligned} \xi_t(\mathsf{F}) &\leq \Psi_F^{n+m+1} + 2(n+m)\Psi_F + n\Psi_F \\ &= \Psi_F^{n+m+1} + \Psi_F(3n+2m). \end{aligned}$$

This representation of the parameter tensor is not manageable for large system since the storage effort still depends exponentially on the number of states and inputs, because of the construction of the core tensor. But from (3.23) follows that there are many zero entries in the proposed core tensor. Thus, less memory consuming representations than the full tensor representation can be used. By using a sparse format for the core tensor the memory effort of the Tucker tensor can be reduced to

$$\xi_t(\mathsf{F}) \leq \Psi_F(n+m+2) + 2(n+m)\Psi_F + n\Psi_F. \tag{3.24}$$

Another possibility to reduce the storage effort for the core tensor is to translate the core to a CP format $\mathsf{\Lambda} = [\mathbf{U}_{\Lambda,1},\ldots,\mathbf{U}_{\Lambda,n+m+1}]\cdot\boldsymbol{\lambda}_\Lambda$, with the same weighting vector than the CP representation $\boldsymbol{\lambda}_\Lambda = \boldsymbol{\lambda}_F$ and factor matrices

$$\mathbf{U}_{\Lambda,i} = \mathbf{I}_{\Psi_F},\ i = 1,\ldots,n+m+1,$$

where $\mathbf{I}_{\Psi_F}$ is a $\Psi_F \times \Psi_F$ identity matrix. This core representation leads to an upper limit on the overall memory demand of the Tucker parameter tensor of

$$\xi_t(\mathsf{F}) \leq \Psi_F + (n+m+1)\Psi_F^2 + 2(n+m)\Psi_F + n\Psi_F \tag{3.25}$$

elements. With the sparse and the CP versions of the core tensor also larger systems can be represented with the Tucker approach, since the storage effort depends only linear and not exponential on number of states and inputs in these cases. To reduce the memory demand further, truncation techniques should be used to reduce the core size further.

To simulate the system in the Tucker format the right hand side can be evaluated based on the factor matrices and the core tensor. A sequence of mode-k tensor vector products has to be computed, [18]

$$\Phi(\mathbf{x}) = \mathbf{F}_\Phi\left(\mathsf{\Lambda}_F \bar{\times}_1 \left(\mathbf{F}_{u_m}^T \begin{pmatrix}1\\u_m\end{pmatrix}\right) \bar{\times}_2 \cdots \bar{\times}_m \left(\mathbf{F}_{u_1}^T \begin{pmatrix}1\\u_1\end{pmatrix}\right) \bar{\times}_{m+1} \left(\mathbf{F}_{x_n}^T \begin{pmatrix}1\\x_n\end{pmatrix}\right) \bar{\times}_{m+2} \cdots \bar{\times}_{m+n} \left(\mathbf{F}_{x_1}^T \begin{pmatrix}1\\x_1\end{pmatrix}\right)\right).$$

Example 3.9 *The contracted product of the state equation with the parameter tensor in Tucker decomposition as derived in Example 3.8 and the monomial tensor is computed by*

$$\begin{aligned}\Phi\left(\begin{pmatrix}x_1\\x_2\end{pmatrix}\right) &= \langle\, \mathsf{F} \mid \mathsf{M}\,(x_1,x_2,u_1)\,\rangle = \mathbf{F}_\Phi\left(\Lambda_F \bar{\times}_1 \left(\mathbf{F}_{u_1}^T\begin{pmatrix}1\\u_1\end{pmatrix}\right)\bar{\times}_2\left(\mathbf{F}_{x_2}^T\begin{pmatrix}1\\x_2\end{pmatrix}\right)\bar{\times}_3\left(\mathbf{F}_{x_1}^T\begin{pmatrix}1\\x_1\end{pmatrix}\right)\right)\\ &= \mathbf{F}_\Phi\left(\Lambda_F\bar{\times}_1\begin{pmatrix}1\\1\\u_1\\1\\1\\u_1\end{pmatrix}\bar{\times}_2\begin{pmatrix}1\\1\\1\\x_2\\x_2\\x_2\end{pmatrix}\bar{\times}_3\begin{pmatrix}1\\x_1\\x_1\\1\\x_1\\x_1\end{pmatrix}\right) = \begin{pmatrix}1&1&1&0&0&0\\0&0&0&1&1&1\end{pmatrix}\begin{pmatrix}a_1\\a_2x_1\\a_3x_1u_1\\b_1x_2\\b_2x_1x_2\\b_3x_1x_2u_1\end{pmatrix}\\ &= \begin{pmatrix}a_1+a_2x_1+a_3u_1x_1\\b_1x_2+b_2x_2x_1+b_3u_1x_2x_1\end{pmatrix}.\end{aligned}$$

3.2.3 Tensor Trains

The third decomposition method considered to represent the parameter tensor of a MTI system is the TT decomposition. If a tensor is available in CP format, the TT cores can be derived directly from the CP factors, [83]. To illustrate this conversion, consider a four dimensional tensor with CP representation $\mathsf{K} = [\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4]\cdot\boldsymbol{\lambda}$ given elementwise by

$$k(i_1,i_2,i_3,i_4) = \sum_{j=1}^{r_{cp}(\mathsf{K})} x_1(i_1,j)x_2(i_2,j)x_3(i_3,j)x_4(i_4,j).$$

To simplify the notation in this first example it is assumed that the weighting vector of the CP form is a vector full of ones. The elements of a four dimensional TT tensor $\mathsf{K} = [\mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_3, \mathsf{G}_4]$ are written as (2.21)

$$k(i_1,i_2,i_3,i_4) = \sum_{j_1=1}^{r_{tt,1}(\mathsf{K})}\sum_{j_2=1}^{r_{tt,2}(\mathsf{K})}\sum_{j_3=1}^{r_{tt,3}(\mathsf{K})} g_1(1,i_1,j_1)g_2(j_1,i_2,j_2)g_3(j_2,i_3,j_3)g_4(j_3,i_4,1).$$

Let the first and the last core tensor be equal to the first and the last factor matrix of the CP representation respectively, i.e.

$$\mathbf{G}_1(1,:,:) = \mathbf{X}_1, \tag{3.26}$$
$$\mathbf{G}_4(:,:,1) = \mathbf{X}_4^T, \tag{3.27}$$

and set all other core tensor elements to

$$g_l(j_{l-1},i_l,j_l) = \begin{cases} x_l(i_l,j_l) & \text{, for } j_{l-1}=j_l, \forall j_{l-1}, j_l = 1,\dots,r_{cp}(\mathsf{K}),\\ 0 & \text{, otherwise.}\end{cases} \tag{3.28}$$

This translates the TT format to a CP one, which gets obvious when inserting the definitions of the core tensors to the elementwise description and eliminating the zeros

$$\begin{aligned} k(i_1,i_2,i_3,i_4) &= \sum_{j_1=1}^{r_{cp}(\mathsf{K})} \sum_{j_2=1}^{r_{cp}(\mathsf{K})} \sum_{j_3=1}^{r_{cp}(\mathsf{K})} g_1(1,i_1,j_1) g_2(j_1,i_2,j_2) g_3(j_2,i_3,j_3) g_4(j_3,i_4,1) \\ &= \sum_{j=1}^{r_{cp}(\mathsf{K})} g_1(1,i_1,j) g_2(j,i_2,j) g_3(j,i_3,j) g_4(j,i_4,1) \\ &= \sum_{j=1}^{r_{cp}(\mathsf{K})} x_1(i_1,j) x_2(i_2,j) x_3(i_3,j) x_4(i_4,j), \end{aligned}$$

which results in the CP representation of the tensor. This shows that by arranging the CP factors as given in (3.26) to (3.28) in the core tensors a TT tensor is found that is equal to the CP one.

This conversion from a CP tensor to a TT can be applied to the parameter tensor of a MTI system now. In other words, the introduced construction of the core tensors (3.28) means that the rows of the CP factor matrices $\mathbf{F}_{x_i}$ and $\mathbf{F}_{u_i}$ are written as diagonals of the corresponding core slices $\mathsf{F}_{x_i}(:, j_{n+m-i+1}, :)$ and $\mathsf{F}_{u_i}(:, j_{m-i+1}, :)$

$$\mathsf{F}_{u_m}(1,:,:) = \mathbf{F}_{u_m}, \tag{3.29}$$

$$\mathsf{F}_{u_k}(:, i_{m-k+1}, :) = \operatorname{diag}\left(\mathbf{f}_{u_k}(i_{m-k+1},:)\right),\ k = 1,\dots,m-1, \tag{3.30}$$

$$\mathsf{F}_{x_k}(:, i_{n+m-k+1}, :) = \operatorname{diag}\left(\mathbf{f}_{x_k}(i_{n+m-k+1},:)\right),\ k = 1,\dots,n, \tag{3.31}$$

$$\mathsf{F}_{\Phi} = \mathbf{F}_{\Phi}^T \circledast \left(\boldsymbol{\lambda}_F \mathbf{1}_n^T\right), \tag{3.32}$$

where $\mathbf{1}_n \in \mathbb{R}^{n\times 1}$ denotes a vector full of ones. In contrast to the example shown before the weighting vector of the CP format of F is not limited to a vector full of ones. The TT format has no weighting vector. Thus, the weighting vector is integrated here by an elementwise product to the last factor F_{Φ} in (3.32). The sizes of the cores of the TT depend on the number Ψ_F of rank-1 elements of the CP tensor resulting in

$$\begin{aligned} \mathsf{F}_{u_m} &\in \mathbb{R}^{1\times 2\times \Psi_F}, \\ \mathsf{F}_{u_k},\ \mathsf{F}_{x_k} &\in \mathbb{R}^{\Psi_F\times 2\times \Psi_F}, \\ \mathsf{F}_{\Phi} &\in \mathbb{R}^{\Psi_F\times 2\times 1}. \end{aligned}$$

The parameter tensor is given in TT format with core tensors (3.29) to (3.32) by

$$\mathsf{F} = \left[\mathsf{F}_{u_m}, \dots, \mathsf{F}_{u_1}, \mathsf{F}_{x_n}, \dots, \mathsf{F}_{x_1}, \mathsf{F}_{\Phi}\right].$$

Example 3.10 *The parameter tensor of the system with two states and one input introduced in Example 3.6 can be represented as TT tensor by*

$$\mathsf{F} = \left[\mathsf{F}_{u_1}, \mathsf{F}_{x_2}, \mathsf{F}_{x_1}, \mathsf{F}_{\Phi}\right],$$

where the cores are constructed by using the factor matrices of the CP representation derived in Example 3.6. The first and the last factor matrix can be expressed by matrices

$$\mathbf{F}_{u_1}(1,:,:) = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}, \ \mathbf{F}_{\Phi}(:,:,1) = \begin{pmatrix} a_1 & 0 \\ a_2 & 0 \\ a_3 & 0 \\ 0 & b_1 \\ 0 & b_2 \\ 0 & b_3 \end{pmatrix}.$$

The TT-cores F_{x_1} *and* F_{x_2} *are given by three dimensional tensors, that are depicted in Figure 3.4. All tensor entries, that are not shown in the figure are equal to zero. The lateral slices are diagonal matrices, where the diagonals are equal to the rows of the corresponding factor matrices of the CP format*

$$\begin{aligned} \mathsf{F}_{x_1}(:,i,:) &= \operatorname{diag}(\mathbf{F}_{x_1}(i,:)), i = 1,2, \\ \mathsf{F}_{x_2}(:,i,:) &= \operatorname{diag}(\mathbf{F}_{x_2}(i,:)), i = 1,2. \end{aligned}$$

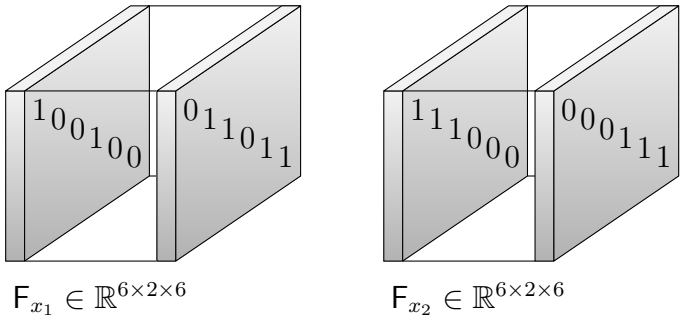


Figure 3.4: Core tensors F_{x_1} and F_{x_2}

This direct translation from CP to TT gives an exact representation of the parameters by the TT. The storage effort for this tensor gives an upper limit for the storage demand of the parameter tensor when represented as TT. To determine the storage demand, the number of elements of the core tensors have to be summed up

$$\begin{aligned} \xi_{tt}(\mathsf{F}) &\leq 2\Psi_F + 2(n+m-1)\Psi_F^2 + n\Psi_F \\ &= (n+2)\Psi_F + 2(n+m-1)\Psi_F^2. \end{aligned} \tag{3.33}$$

The direct conversion from CP to TT is not optimal regarding the complexity of the TT. Thus, truncation methods for TT, that are based on singular value decompositions (SVD), should be used to approximate the tensor by a TT with lower ranks, [83].

For simulation the right hand side of the state equation is evaluated by the computation of the contracted product of parameter tensor and the rank-1 monomial tensor, using the core tensors of the TT format. The contracted product is calculated by a sequence of

multiplications of mode-2 tensor vector products of the core tensors and the factors of the monomial tensor, [59]

$$\Phi(\mathbf{x}) = \mathsf{F}_{\Phi}^{T}\left(\mathsf{F}_{x_1} \bar{\times}_2 \begin{pmatrix}1\\x_1\end{pmatrix}\right)^{T} \cdots \left(\mathsf{F}_{x_n} \bar{\times}_2 \begin{pmatrix}1\\x_n\end{pmatrix}\right)^{T} \left(\mathsf{F}_{u_1} \bar{\times}_2 \begin{pmatrix}1\\u_1\end{pmatrix}\right)^{T} \cdots \left(\mathsf{F}_{u_m} \bar{\times}_2 \begin{pmatrix}1\\u_m\end{pmatrix}\right)^{T}.$$

Example 3.11 *To simulate the state equation with the TT parameter tensor introduced in Example 3.10 its right hand side is evaluated by*

$$\Phi\left(\begin{pmatrix}x_1\\x_2\end{pmatrix}\right) = \langle\, \mathsf{F} \mid \mathsf{M}(x_1, x_2, u_1) \,\rangle = \mathsf{F}_{\Phi}^{T}\left(\mathsf{F}_{x_1} \bar{\times}_2 \begin{pmatrix}1\\x_1\end{pmatrix}\right)^{T} \left(\mathsf{F}_{x_2} \bar{\times}_2 \begin{pmatrix}1\\x_2\end{pmatrix}\right)^{T} \left(\mathsf{F}_{u_1} \bar{\times}_2 \begin{pmatrix}1\\u_1\end{pmatrix}\right)^{T}$$

$$= \begin{pmatrix} a_1 & a_2 & a_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & b_1 & b_2 & b_3 \end{pmatrix} \operatorname{diag}\left(\begin{pmatrix}1\\x_1\\x_1\\1\\x_1\\x_1\end{pmatrix}\right) \operatorname{diag}\left(\begin{pmatrix}1\\1\\1\\x_2\\x_2\\x_2\end{pmatrix}\right) \begin{pmatrix}1\\1\\u_1\\1\\1\\u_1\end{pmatrix} = \begin{pmatrix} a_1 + a_2 x_1 + a_3 u_1 x_1 \\ b_1 x_2 + b_2 x_2 x_1 + b_3 u_1 x_2 x_1 \end{pmatrix}.$$

3.2.4 Hierarchical Tucker

The last decomposition method that will be applied to the parameter tensors of MTI systems here is the HT decomposition. As for the Tucker and TT decompositions there is a constructive method to generate a HT from a CP decomposition, which is - in general - not minimal. This will be illustrated by a small example. Consider therefore a two dimensional tensor, i.e. a matrix $\mathbf{K} \in \mathbb{R}^{I_1 \times I_2}$, that can be represented as HT tensor by a subtree as shown in Figure 2.15. It is assumed, that in CP representation the tensor is given by $\mathbf{K} = [\mathbf{X}_1, \mathbf{X}_2] \cdot \boldsymbol{\lambda}_K$ and is constructed by $r_{cp}(\mathsf{K})$ rank-1 components. The values of the original tensor are computed with the HT leaves $\mathbf{U}_{t_l} \in \mathbb{R}^{I_1 \times r_{ht,t_l}(\mathsf{K})}$ and $\mathbf{U}_{t_r} \in \mathbb{R}^{I_2 \times r_{ht,t_r}(\mathsf{K})}$ and node $\mathsf{B}_t \in \mathbb{R}^{r_{ht,t_l}(\mathsf{K}) \times r_{ht,t_r}(\mathsf{K}) \times 1}$ factors according to (2.24) by

$$\mathbf{K}^{(\{1,2\})} = \sum_{i=1}^{r_{ht,t_l}(\mathsf{K})} \sum_{j=1}^{r_{ht,t_r}(\mathsf{K})} \left(\mathbf{u}_{t_r}(:,j) \otimes \mathbf{u}_{t_l}(:,i)\right) b_t(i,j,1). \tag{3.34}$$

To get the relation to the CP format the leaf matrices are set equal to the factor matrices of the CP representation $\mathbf{U}_{t_l} = \mathbf{X}_1$ and $\mathbf{U}_{t_r} = \mathbf{X}_2$, [45]. With this choice the ranks are given by $r_{ht,t_l}(\mathsf{K}) = r_{ht,t_r}(\mathsf{K}) = r_{cp}(\mathsf{K})$. A diagonal transfer matrix is chosen

$$\mathbf{B}_t(i,j,1) = \begin{cases} \lambda_K(i) & \text{, for } i = j, \forall i, j = 1, \ldots, r_{cp}(\mathsf{K}), \\ 0 & \text{, otherwise,} \end{cases} \tag{3.35}$$

containing the entries of the weighting vector $\boldsymbol{\lambda}_K$. Inserting these definitions for leaves and transfer matrices to (3.34) and taking into account the zero entries of (3.35) leads to

$$\mathbf{K}^{(\{1,2\})} = \sum_{i=1}^{r_{cp}(\mathsf{K})} \sum_{j=1}^{r_{cp}(\mathsf{K})} \left(\mathbf{x}_2(:,j) \otimes \mathbf{x}_1(:,i)\right) b_t(i,j,1) = \sum_{j=1}^{r_{cp}(\mathsf{K})} \left(\mathbf{x}_2(:,j) \otimes \mathbf{x}_1(:,j)\right) \lambda_K(j),$$

which is exactly the mode-($\{1,2\}$) matricization of the CP representation

$$\mathbf{K} = \sum_{j=1}^{r_{cp}(\mathsf{K})} \lambda_K(j)\mathbf{x}_1(:,j) \circ \mathbf{x}_2(:,j),$$

showing the equivalence with the CP format with the given choice of leaves and nodes.

This approach is extended to higher order tensors, such that the HT presentation of the parameter tensor F is derived from the CP factors. For the direct translation from CP structure of the MTI system to HT first of all a tree representing the tensor F has to be constructed, as shown in Figure 3.5 as an example for a system with 2 states and 1 input. This is one way to construct the HT tree of F, that should be used here in the following. Also other tree layouts would be possible.

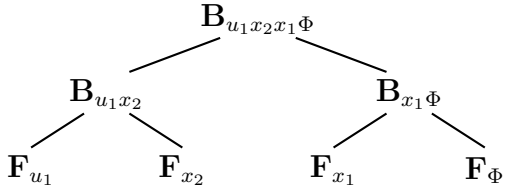


Figure 3.5: Tree of a parameter tensor F of a system with two states and one input

The leaf node matrices $\mathbf{F}_\Phi$, $\mathbf{F}_{x_i}$ with $i = 1,\dots,n$ and $\mathbf{F}_{u_i}$ with $i = 1,\dots,m$ are equal to the factor matrices of the CP representation (3.19). The transfer matrix of the top node

$$\mathbf{B}_{u_m\cdots u_1x_n\cdots x_1\Phi} = \operatorname{diag}(\boldsymbol{\lambda}_F)$$

is a diagonal matrix with the entries of the CP weighting vector $\boldsymbol{\lambda}_F$ on the diagonal. All other transfer nodes are given in tensor form by

$$\mathsf{B}_t = \mathbf{I}_{\Psi_F,3} \in \mathbb{R}^{\Psi_F\times\Psi_F\times\Psi_F},$$

where $\mathbf{I}_{\Psi_F,3}$ is a three dimensional diagonal tensor with elements

$$i_{\Psi_F,3}(j_1,j_2,j_3) = \begin{cases} 1 & \text{, for } \forall j_1 = j_2 = j_3 = 1,\dots,\Psi_F, \\ 0 & \text{, otherwise,} \end{cases}$$

with ones on the superdiagonal, [45].

Example 3.12 *Consider the parameter tensor of the system introduced in Example 3.6 with its parameter tensor* F *in CP representation. Based on that the parameter tensor can be expressed as HT tensor with a tree as depicted in Figure 3.5. The leaf nodes are equal to the factor matrices of the CP tensor* $\mathbf{F}_\Phi$, $\mathbf{F}_{x_1}$, $\mathbf{F}_{x_2}$ *and* $\mathbf{F}_{u_1}$ *given in Example 3.6. The transfer tensor of the top node contains the elements of the weighting vector, i.e. the parameters of the system by*

$$\mathbf{B}_{u_1x_2x_1\Phi}(:,:,1) = \operatorname{diag}\begin{pmatrix} a_1 & a_2 & a_3 & b_1 & b_2 & b_3 \end{pmatrix}.$$

The other transfer tensors are diagonal tensors with ones on the diagonal

$$\mathsf{B}_{u_1x_2} = \mathsf{B}_{x_1\Phi} = \mathbf{I}_{6,3} \in \mathbb{R}^{6\times6\times6}.$$

To store the parameter tensor in HT representation, the leaf and the transfer matrices have to be considered. Each state and input belongs to a leaf node. The additional factor $\mathbf{F}_\Phi$ leads to $n+m+1$ leaf nodes. With the proposed tree structure a tensor of order N has $N-1$ transfer matrices resulting in $n+m$ transfer matrices here, [45]. An upper bound for the memory demand of the HT parameter tensor is estimated by

$$\begin{aligned}\xi_{ht}(\mathsf{F}) &\leq \Psi_F^2 + (n+m-1)\Psi_F^3 + 2(n+m)\Psi_F + n\Psi_F \\ &= \Psi_F^2 + (n+m-1)\Psi_F^3 + (3n+2m)\Psi_F. \end{aligned} \tag{3.36}$$

Since the direct translation from CP to HT is very inefficient regarding the memory demand, i.e. all ranks are equal to the number of rank-1 components of the CP tensor, a truncation should be applied that is based on the SVD of the matricizations of the original tensor to find approximations of lower rank.

If the parameter tensor F is given in HT form, the right hand side of the state equation can be efficiently evaluated based on the tree of the HT tensor. The contracted product of parameter and monomial tensor is computed by a sequence of mode-k tensor vector products

$$\Phi(\mathbf{x}) = \mathsf{F} \bar{\times}_1 \begin{pmatrix} 1 \\ u_m \end{pmatrix} \bar{\times}_2 \begin{pmatrix} 1 \\ u_{m-1} \end{pmatrix} \bar{\times}_3 \cdots \bar{\times}_{n+m} \begin{pmatrix} 1 \\ x_1 \end{pmatrix}. \tag{3.37}$$

Having the tensor F in HT representation the mode-k tensor vector products are defined by simply replacing the leaf nodes $\mathbf{F}_{u_i}$ and $\mathbf{F}_{x_i}$ of F by $\begin{pmatrix} 1 & u_i \end{pmatrix} \mathbf{F}_{u_i}$ and $\begin{pmatrix} 1 & x_i \end{pmatrix} \mathbf{F}_{x_i}$, respectively. The state derivative or next state $\Phi(\mathbf{x})$ is computed by solving the nodes of this tree recursively using (2.23).

Example 3.13 *The right hand side of the state equation for the example system (3.17) and (3.18) is evaluated by*

$$\Phi\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) = \mathsf{F} \bar{\times}_1 \begin{pmatrix} 1 \\ u_1 \end{pmatrix} \bar{\times}_2 \begin{pmatrix} 1 \\ x_2 \end{pmatrix} \bar{\times}_3 \begin{pmatrix} 1 \\ x_1 \end{pmatrix},$$

which is computed with the HT representation of F *from Example 3.12 by the tree shown in Figure 3.6.*

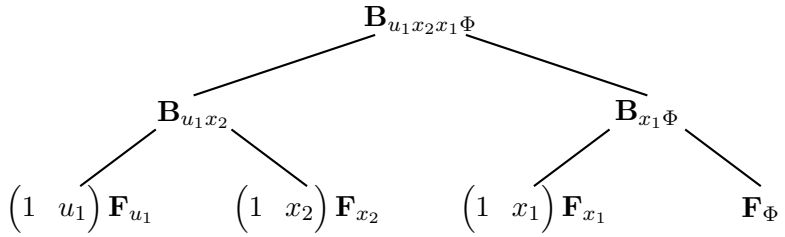


Figure 3.6: Evaluation of the right hand side of the state equation with a HT tensor

Applying (2.23) recursively to the tree depicted in Figure 3.6 gives the desired evaluation of

the right hand side of the state equation by

$$\begin{aligned}
\Phi\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) &= \langle\, \mathsf{F} \mid \mathsf{M}\,(x_1, x_2, u_1) \,\rangle = (\mathbf{U}_{x_1\Phi} \otimes \mathbf{U}_{u_1x_2})\, \mathbf{B}_{u_1x_2x_1\Phi} \\
&= \left(\left(\mathbf{F}_\Phi \otimes \begin{pmatrix} 1 & x_1 \end{pmatrix} \mathbf{F}_{x_1}\right) \mathbf{B}_{x_1\Phi} \otimes \left(\begin{pmatrix} 1 & x_2 \end{pmatrix} \mathbf{F}_{x_2} \otimes \begin{pmatrix} 1 & u_1 \end{pmatrix} \mathbf{F}_{u_1}\right) \mathbf{B}_{u_1x_2}\right) \mathbf{B}_{u_1x_2x_1\Phi} \\
&= \left(\mathbf{F}_\Phi \otimes \begin{pmatrix} 1 & x_1 \end{pmatrix} \mathbf{F}_{x_1} \otimes \begin{pmatrix} 1 & x_2 \end{pmatrix} \mathbf{F}_{x_2} \otimes \begin{pmatrix} 1 & u_1 \end{pmatrix} \mathbf{F}_{u_1}\right) (\mathbf{B}_{x_1\Phi} \otimes \mathbf{B}_{u_1x_2})\, \mathbf{B}_{u_1x_2x_1\Phi} \\
&= \left(\begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}^T \otimes \begin{pmatrix} 1 \\ x_1 \\ x_1 \\ 1 \\ x_1 \\ x_1 \end{pmatrix}^T \otimes \begin{pmatrix} 1 \\ 1 \\ 1 \\ x_2 \\ x_2 \\ x_2 \end{pmatrix}^T \otimes \begin{pmatrix} 1 \\ 1 \\ u_1 \\ 1 \\ 1 \\ u_1 \end{pmatrix}^T\right) (\mathbf{B}_{x_1\Phi} \otimes \mathbf{B}_{u_1x_2})\, \mathbf{B}_{u_1x_2x_1\Phi} \\
&= \begin{pmatrix} a_1 + a_2x_1 + a_3u_1x_1 \\ b_1x_2 + b_2x_2x_1 + b_3u_1x_2x_1 \end{pmatrix}.
\end{aligned}$$

The matrices

$$\begin{aligned}
\mathbf{U}_{x_1\Phi} &= \left(\mathbf{F}_\Phi \otimes \begin{pmatrix} 1 & x_1 \end{pmatrix} \mathbf{F}_{x_1}\right) \mathbf{B}_{x_1\Phi}, \\
\mathbf{U}_{u_1x_2} &= \left(\begin{pmatrix} 1 & x_2 \end{pmatrix} \mathbf{F}_{x_2} \otimes \begin{pmatrix} 1 & u_1 \end{pmatrix} \mathbf{F}_{u_1}\right) \mathbf{B}_{u_1x_2}
\end{aligned}$$

belong to the right and the left subtree, respectively.

3.2.5 Comparison of the decomposed representations

The previous subsections show, that all proposed decomposition methods allow to represent the parameter tensors of MTI systems exactly. The symbolic description of the right hand sides of the state equations are translated directly term by term to a CP format of the parameter tensor. The other decomposed representations are not constructed directly from the equations but follow from the CP version. No approximations have to be done in the first step such that the decomposed tensors describe the system parameters exactly, i.e. the dynamical behavior of the system in the standard symbolic representation of the state equations and with a decomposed one is exactly the same. In the tensor form the state space model is represented in a structured way, where all specific parameters of a plant are stored in the parameter tensor and the model structure is given by the monomial tensor. Thus, tools for polynomials like multiplication or differentiation as introduced in Section 2.3 can be applied to the right hand side functions of the state space models of MTI systems. This allows to develop algorithms especially for MTI systems for system analysis and controller synthesis. With the decomposed representations more efficient algorithms can be developed, when the calculations are performed on the low dimensional decomposition factors.

As already said, all four decomposition formats lead to the exact parameter data. But the direct translation from the state equations to CP or from CP to the other decomposed forms does not give an efficient or optimal solution regarding the storage effort. When building the CP tensor the number of rank-1 terms is not necessarily minimal by construction. It

depends on the number of terms Ψ_F of the state equations. The focus when constructing the decompositions is at first on the exact representation of the full tensor and not on finding an optimal representation, i.e. with minimal number of terms. The other formats result from a conversion from the CP tensor as shown in Figure 3.7.

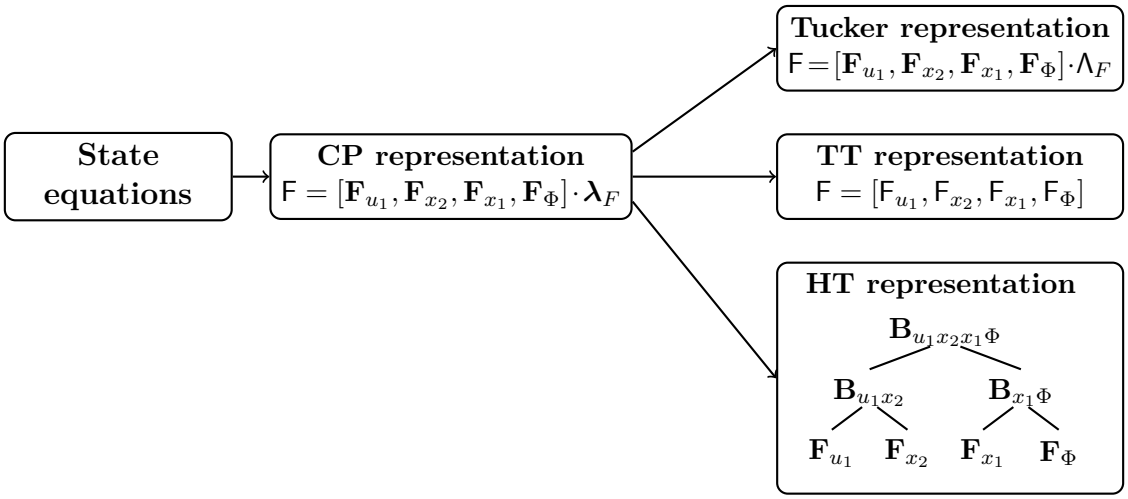


Figure 3.7: Construction of the parameter tensor with different decomposition methods

Because of that, the storage effort of all formats in the other decomposition approaches depends on the number of rank-1 elements of the CP tensor. The Tucker, TT and HT representations are also not optimal regarding the storage demand. There are two reasons for that. First, these formats result from a CP representation that might not have the minimal number of rank-1 terms. Second, the direct translation from CP to the other decompositions is not optimal. It gives exact representations but it might be possible to get representations with less components. But it is ensured, that not more elements are necessary with the particular decomposition techniques. Thus, an upper bound on the storage demand is computed for each decomposition method.

For the direct translation without any truncation an upper limit of the number of elements that have to be stored can be computed as derived in (3.20), (3.24), (3.25) , (3.33) and (3.36). The decomposed parameter tensors are built in a similar way in each of the four decomposition methods since always each state x_i and each input u_j gets its own factor matrix. Figure 3.8 shows the comparison of the storage complexities ξ_i of the different decompositions. In addition to that one more factor matrix $\mathbf{F}_\Phi$ is necessary that has the same memory demand with $n\Psi_F$ elements for all decompositions. This is not considered in the graph since it is the same for all methods.

Figure 3.8 compares the upper bound on the memory demand of the different decomposition techniques in a log-scale. The CP decomposition has the lowest demand. The Tucker decomposition shows the second lowest demand, where the sparse core representation is less demanding than the CP core representation. But for large-scale systems the storage demand of the Tucker representation cannot be further reduced since truncation is impossible. This further reduction is possible when focusing on TT and HT decomposition, where the HT

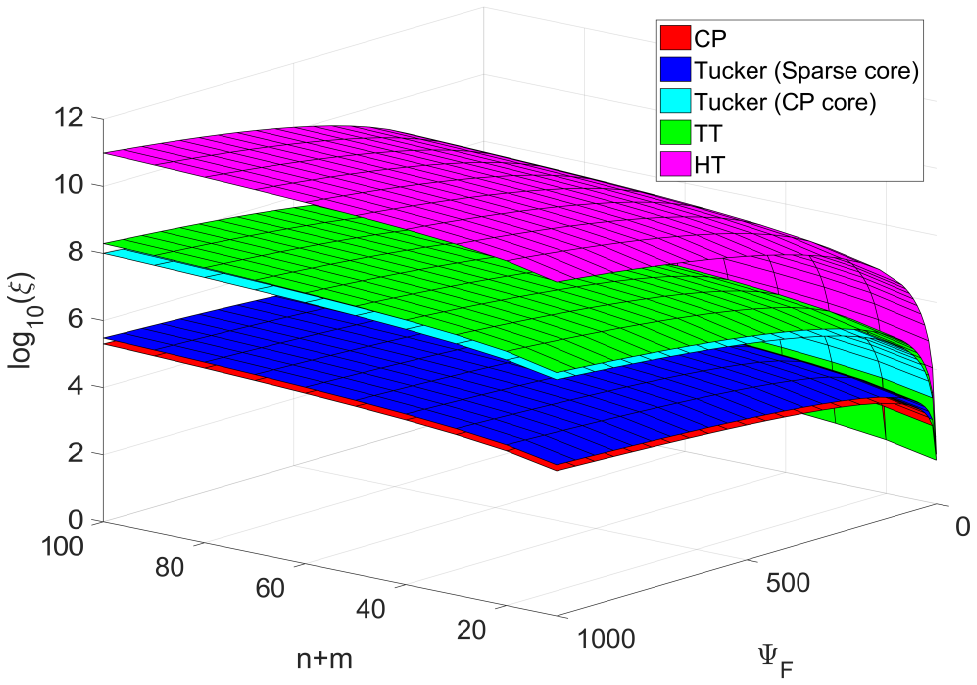


Figure 3.8: Storage complexities ξ_i of the different decomposition techniques

decomposition has the highest upper bound on the storage effort. When comparing the logarithmic change of the storage effort of all decomposition techniques, the main dependence is on the number of terms of the state equation. How good truncation techniques can reduce this effort, will be shown in Section 5.2 where the decompositions are applied to heating systems applications.

Assuming the values are stored in a double precision format, i.e. 8 byte per value, the upper bounds for CP, Tucker and TT stay in the megabyte storage range, which is computationally manageable. Only HT has a higher demand but the decomposition has some very good reduction properties, such that this maximal possible storage effort that is shown here is not the complexity that it necessary when getting to application. The storage complexities are very low, when comparing them with the demand of the full representation that was shown in Figure 3.3. For small scale systems with many parameters the full representation might be beneficial to the direct translation to a decomposed format, because each term adds a rank-1 element there. But after truncation and especially for higher order systems the full representation has no advantage. Since the number of elements in full representation increases exponentially the number of terms explodes and it is not possible to store the parameters anymore. No system analysis and controller design is possible then. Already for systems with 28 states and inputs more than 1 gigabyte of memory is necessary to store the parameters. This shows that the full representation is not suitable for large-scale systems and highlights the advantage of the decomposed representations.

To reduce the storage effort, toolboxes in different programming environments like MATLAB are available that offer different algorithms for the truncation of the decompositions to find low-rank approximations, [5, 45, 84, 109]. But the number of elements can not be reduced arbitrary low, since the approximation has an effect on the dynamics of the corresponding MTI system, which is shown in the following example.

Example 3.14 *Consider the MTI system*

$$\begin{aligned}\dot{x}_1 &= -2x_1 + 3x_2 - 4x_1x_2,\\ \dot{x}_2 &= -0.1x_1x_2 + x_1u_1,\\ \dot{x}_3 &= -x_3 - 0.2x_2u_2,\end{aligned}$$

that can be represented in all four decomposed formats. Approximating the original tensor with a CP tensor with 6 *rank-*1 *elements or with a Tucker core of dimension* $\mathbb{R}^{2\times 6 2\times 3}$ *or with approximation accuracy of* 0.01 *for TT or HT, results in an MTI system, where the system response to a sawtooth and a pulse input signal for the first and the second input respectively is shown in Figure 3.9.*

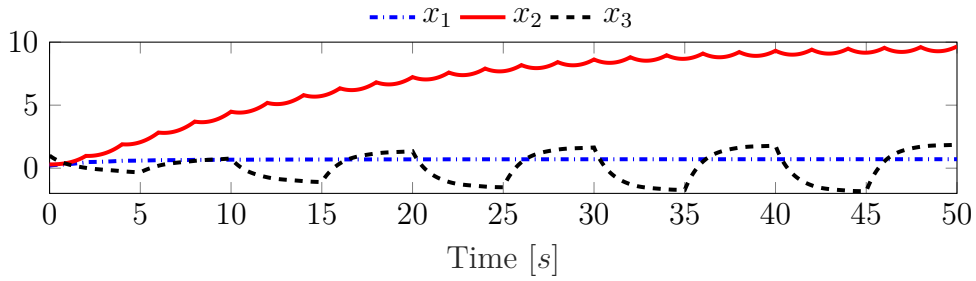


Figure 3.9: Simulation result of the truncated system without loss of dynamics

*With this parameter tensor approximation all system dynamics are still captured. If now the decompositions are further truncated, by reducing the number of rank-*1 *components of the CP tensor to* 5 *or reducing the size of the Tucker core to* $\mathbb{R}^{1\times 6 2\times 3}$ *or approximating the TT or HT tensors with accuracy* 0.1*, this influences the dynamical behavior. The system response of these approximations to the same input signal is depicted in Figure 3.10.*

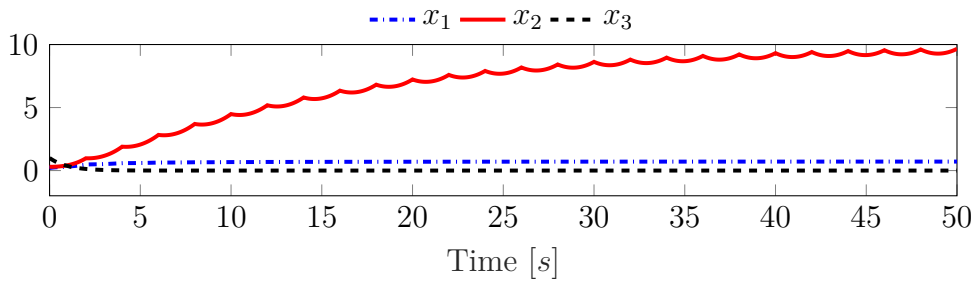


Figure 3.10: Simulation result of the truncated system with loss of dynamics

It is obvious that the dynamics of the third state are not captured any longer by this approximation. This example shows that the tensor decomposition cannot be truncated arbitrary, because truncation can lead to a loss of captured dynamics.

As mentioned in Section 2.1.2 the different decomposition methods have different decomposition algorithms. To compare the methods, they can be sorted in two categories.

On the one hand there are the CP and Tucker decompositions. With these approaches low-rank approximations are computed by setting a desired rank. A decomposition is determined, that has the desired rank and fits the data of the full tensor as good as possible. Therefore a nonlinear optimization problem (2.13) or (2.19) has to be solved with techniques like the ALS algorithm. The advantage of this approach is, that by setting a desired rank the decomposed tensor has a predictable size. But it is not possible to predict the accuracy of the result. The optimal CP rank of a tensor cannot be determined, because it is an NP hard problem, [44]. For small scale tensors in [44] typical ranks and maximal ranks for some specific tensors are given. But results are available for tensors of orders up to three only. Since in the field of MTI systems in most cases higher order tensors are investigated, these results cannot be applied here. The accuracy has to be determined after the decomposition, which leads to an trial and error procedure, [44]. The decomposition with a selected number of rank-1 elements is computed and the accuracy is determined afterwards. If the accuracy is in the desired range, the decomposition can be used further. If not, the number of rank-1 elements has to be adjusted. Again a decomposition is computed and the accuracy is investigated. This procedure is repeated until a good approximation is found. Another disadvantage of the computation of CP and Tucker decomposition is, that the optimization problems that have to be solved are non-convex, i.e. their solutions depend on the initial values of the decomposition factors. Techniques for finding good initial values are available for CP by a generalized eigenvalue decomposition as well as for Tucker by a HOSVD but in many cases the optimization problem has to be solved several times with different initial values to get a good solution, which again leads to a trail and error computation strategy, [44, 109]. Because of the curse of dimensionality the parameter tensors of large-scale systems cannot be represented as full tensors. But translations to a decomposed representation can overcome this curse of dimensionality. As already mentioned for further reduction of complexity, low-rank approximation techniques should be applied. That makes it necessary that the approximation algorithms work with the decomposed representations to reduce the rank. Tucker decomposition has the big drawback, that no algorithm in the standard toolboxes is available that allows a further reduction based on the decomposed tensor. The algorithm always needs a full tensor as starting value. Since for large-scale systems a full representation is not manageable, a truncation of a Tucker tensor is impossible.

The second group of decomposition techniques is formed by the TT and HT decomposition. The advantages of these decompositions are their SVD based decomposition and truncation algorithms. These algorithms compute SVDs of different matricizations of the tensors and cut less significant singular values. This provides a quasi-optimal solution and it represents the best approximation with an adjustable accuracy (2.22). It has to be pointed out that no initial values have to be set for that in contrast to the CP or Tucker case. When applying the decomposition to MTI systems, one can estimate the accuracy bound by the knowledge of the sensitivity of the parameters of the system. One has to check how much additional inaccuracy from the decomposition can be allowed without influencing the main dynamical characteristics of the system. The user does not have to set the TT or HT ranks. The desired accuracy is set directly and during the computation of the decomposition, the ranks

are determined such that the resulting decomposed tensor fulfills the accuracy constraint. This has the disadvantage, that it is not clear how much reduction can be achieved. It depends on the structure of the parameter data, how much compression is possible with a desired accuracy. In contrast to the Tucker decomposition, algorithms are available that can truncate already decomposed tensors to find a representation of lower rank without building the full tensor as well for TT as for HT. This makes these decompositions suitable for representations of parameter tensors of large-scale systems whose parameter tensors are high dimensional.

When focusing on the simulation of the systems in the different decomposed forms, Sections 3.2.1 to 3.2.4 described that the contracted product of parameter tensor and monomial tensor can be computed based on the decomposition factors for all four methods. For CP and HT representation only standard matrix products are necessary that are available in many mathematical programming tools. For Tucker and TT, tensor vector or tensor matrix products have to be computed. This needs tensor toolboxes like [5] or [84]. The complexity of these operations depends on the sizes of the factor matrices or core tensors. Again the ranks of the decompositions play an important role here, since they determine the sizes of the factors. This shows the importance of low-rank approximations also for an efficient evaluation of the state and output equations.

To summarize this part the investigation has shown that MTI systems can be represented in all four proposed decomposition methods, but there are significant differences in the complexity. The general results are summarized in Table 3.1. In Section 5.2 the methods are applied to heating systems examples for further comparison.

Table 3.1: Comparison of general properties of the decomposed formats of MTI systems (Ranking starting with the best: $+ \circ -$; ✓: possible, ×: not possible)

	CP	Tucker	TT	HT
Representation of MTI systems	✓	✓	✓	✓
Predictable size of decomposition	✓	✓	×	×
Truncation with guaranteed accuracy	×	×	✓	✓
Truncation independent of initial value	×	×	✓	✓
Truncation of decomposed tensors	✓	×	✓	✓
Memory demand of the direct translation	$+$	$\circ$	$-$	$-$
Simulation based on decomposition factors	✓	✓	✓	✓

3.3 Linearization

In control engineering many tools and methods are available for linear state space models. The dynamics of real world plants are in general nonlinear. To use the methods from linear control, nonlinear state space models are approximated around an operating point by a linear model. An algorithm for linearization of MTI systems is derived here that is based on the

differentiation in the tensor framework introduced in Section 2.3. The task is to approximate an MTI system

$$\Phi(\mathbf{x}) = \mathbf{f}(\mathbf{x},\mathbf{u}) = \langle\, \mathsf{F} \mid \mathsf{M}(\mathbf{x},\mathbf{u}) \,\rangle, \tag{3.38}$$
$$\mathbf{y} = \mathbf{g}(\mathbf{x},\mathbf{u}) = \langle\, \mathsf{G} \mid \mathsf{M}(\mathbf{x},\mathbf{u}) \,\rangle, \tag{3.39}$$

by a linear system

$$\Phi(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}, \tag{3.40}$$
$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}, \tag{3.41}$$

around an operating point $(\bar{\mathbf{x}},\bar{\mathbf{u}})$. The system matrices of the linear approximation follow from partial differentiation of the state and output equations of the MTI system with respect to the states and the inputs, [4]. The computation of the partial derivatives with the operational tensor approach of Theorem 2.2 is adapted here to consider the separation of the variables in states and inputs. The partial derivatives with respect to the states and inputs of the transition tensor is given by

$$\mathsf{F}_{x_j} = \mathsf{F} \times_{n+m-j+1} \mathbf{\Theta},\ j = 1,\dots,n,$$
$$\mathsf{F}_{u_j} = \mathsf{F} \times_{m-j+1} \mathbf{\Theta},\ j = 1,\dots,m.$$

The approach can be used as well for the output tensor G. Evaluation at the operating point results in

$$\mathbf{A}(:,j) = \left.\frac{\partial \mathbf{f}(\mathbf{x},\mathbf{u})}{\partial x_j}\right|_{\substack{\mathbf{x}=\bar{\mathbf{x}},\\ \mathbf{u}=\bar{\mathbf{u}}}} = \left\langle\, \mathsf{F}_{x_j} \mid \mathsf{M}(\bar{\mathbf{x}},\bar{\mathbf{u}}) \,\right\rangle,\ j = 1,\dots,n,$$
$$\mathbf{B}(:,j) = \left.\frac{\partial \mathbf{f}(\mathbf{x},\mathbf{u})}{\partial u_j}\right|_{\substack{\mathbf{x}=\bar{\mathbf{x}},\\ \mathbf{u}=\bar{\mathbf{u}}}} = \left\langle\, \mathsf{F}_{u_j} \mid \mathsf{M}(\bar{\mathbf{x}},\bar{\mathbf{u}}) \,\right\rangle,\ j = 1,\dots,m,$$
$$\mathbf{C}(:,j) = \left.\frac{\partial \mathbf{g}(\mathbf{x},\mathbf{u})}{\partial x_j}\right|_{\substack{\mathbf{x}=\bar{\mathbf{x}},\\ \mathbf{u}=\bar{\mathbf{u}}}} = \left\langle\, \mathsf{G}_{x_j} \mid \mathsf{M}(\bar{\mathbf{x}},\bar{\mathbf{u}}) \,\right\rangle,\ j = 1,\dots,n,$$
$$\mathbf{D}(:,j) = \left.\frac{\partial \mathbf{g}(\mathbf{x},\mathbf{u})}{\partial u_j}\right|_{\substack{\mathbf{x}=\bar{\mathbf{x}},\\ \mathbf{u}=\bar{\mathbf{u}}}} = \left\langle\, \mathsf{G}_{u_j} \mid \mathsf{M}(\bar{\mathbf{x}},\bar{\mathbf{u}}) \,\right\rangle,\ j = 1,\dots,m.$$

Thus, the columns of the system matrices of the linear state space model are the columns of the Jacobians

$$\mathbf{J}_{\mathbf{f}}(\bar{\mathbf{x}},\bar{\mathbf{u}}) = \langle\, \mathsf{J}_{\mathbf{f}} \mid \mathsf{M}(\bar{\mathbf{x}},\bar{\mathbf{u}}) \,\rangle = \left.\begin{pmatrix} \frac{\partial f_1(\mathbf{x},\mathbf{u})}{\partial x_1} & \dots & \frac{\partial f_1(\mathbf{x},\mathbf{u})}{\partial x_n} & \frac{\partial f_1(\mathbf{x},\mathbf{u})}{\partial u_1} & \dots & \frac{\partial f_1(\mathbf{x},\mathbf{u})}{\partial u_m} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n(\mathbf{x},\mathbf{u})}{\partial x_1} & \dots & \frac{\partial f_n(\mathbf{x},\mathbf{u})}{\partial x_n} & \frac{\partial f_n(\mathbf{x},\mathbf{u})}{\partial u_1} & \dots & \frac{\partial f_n(\mathbf{x},\mathbf{u})}{\partial u_m} \end{pmatrix}\right|_{\substack{\mathbf{x}=\bar{\mathbf{x}},\\ \mathbf{u}=\bar{\mathbf{u}}}},$$

of the state function $\mathbf{f}(\mathbf{x},\mathbf{u})$ and $\mathbf{J}_{\mathbf{g}}(\bar{\mathbf{x}},\bar{\mathbf{u}}) = \langle\, \mathsf{J}_{\mathbf{g}} \mid \mathsf{M}(\bar{\mathbf{x}},\bar{\mathbf{u}}) \,\rangle$ of the output function $\mathbf{g}(\mathbf{x},\mathbf{u})$ at the operating point. This allows to derive a tensor description of the linear system matrices

directly as for the Jacobian, such that the matrices can be computed by a simple evaluation of a contracted product with a monomial tensor at the operating point like

$$\mathbf{A}(\bar{\mathbf{x}}, \bar{\mathbf{u}}) = \langle\, \mathsf{A}_{lin} \mid \mathsf{M}\,(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \,\rangle\,.$$

The same holds for the other system matrices with parameter tensors B_{lin}, C_{lin} and D_{lin}. The Jacobians of state and output function are computed as introduced in Lemma 2.2. To extract the parameters that belong to the different system matrices, subtensors of the parameter tensors $\mathsf{J}_{\mathbf{f}}$ and $\mathsf{J}_{\mathbf{g}}$ of the Jacobians have to be selected to get the parameter tensors of the system matrices, since e.g. the first n columns of $\mathbf{J}_{\mathbf{f}}(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ give the state matrix $\mathbf{A}$ and the following m columns give the input matrix $\mathbf{B}$. Therefore with (2.69) the parameter tensors $\mathsf{A}_{lin} \in \mathbb{R}^{\times^{(n+m)}2\times n\times n}$ and $\mathsf{B}_{lin} \in \mathbb{R}^{\times^{(n+m)}2\times n\times m}$ of the system matrices of the linear state equation have fibers

$$\mathbf{a}_{lin}(\mathbf{i}_u, \mathbf{i}_x, :, j) = \mathbf{j}_{\mathbf{f}}(\mathbf{i}_u, \mathbf{i}_x, :, j) = \mathbf{f}_{x_j}(\mathbf{i}_u, \mathbf{i}_x, :),\ j = 1, \ldots, n, \tag{3.42}$$
$$\mathbf{b}_{lin}(\mathbf{i}_u, \mathbf{i}_x, :, j) = \mathbf{j}_{\mathbf{f}}(\mathbf{i}_u, \mathbf{i}_x, :, n+j) = \mathbf{f}_{u_j}(\mathbf{i}_u, \mathbf{i}_x, :),\ j = 1, \ldots, m. \tag{3.43}$$

The fibers of the parameter tensors $\mathsf{C}_{lin} \in \mathbb{R}^{\times^{(n+m)}2\times p\times n}$ and $\mathsf{D}_{lin} \in \mathbb{R}^{\times^{(n+m)}2\times p\times m}$ of the output equation are given by

$$\mathbf{c}_{lin}(\mathbf{i}_u, \mathbf{i}_x, :, j) = \mathbf{j}_{\mathbf{g}}(\mathbf{i}_u, \mathbf{i}_x, :, j) = \mathbf{g}_{x_j}(\mathbf{i}_u, \mathbf{i}_x, :),\ j = 1, \ldots, n, \tag{3.44}$$
$$\mathbf{d}_{lin}(\mathbf{i}_u, \mathbf{i}_x, :, j) = \mathbf{j}_{\mathbf{g}}(\mathbf{i}_u, \mathbf{i}_x, :, n+j) = \mathbf{g}_{u_j}(\mathbf{i}_u, \mathbf{i}_x, :),\ j = 1, \ldots, m. \tag{3.45}$$

The sizes of the first $n+m$ dimensions of the parameter tensors A_{lin} to D_{lin} is equal to the size of the same dimensions of F and G respectively, indicated by the index vectors $\mathbf{i}_x \in \mathbb{R}^n$ an $\mathbf{i}_u \in \mathbb{R}^m$. The last two dimensions, i.e. i_{n+m+1} and i_{n+m+2} are the dimensions of the linear system matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{D}$ respectively. With this derivation of the parameter tensors the system matrices are computed as given in the following lemma.

Lemma 3.1 (Linearization of MTI systems in tensor format) *Using (2.68) the system matrices of the linearization (3.40) and (3.41) of a MTI system (3.38) and (3.39) around the operating point $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ are given analytically depending on the operating point by*

$$\mathbf{A}(\bar{\mathbf{x}}, \bar{\mathbf{u}}) = \langle\, \mathsf{A}_{lin} \mid \mathsf{M}\,(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \,\rangle \in \mathbb{R}^{n\times n}, \tag{3.46}$$
$$\mathbf{B}(\bar{\mathbf{x}}, \bar{\mathbf{u}}) = \langle\, \mathsf{B}_{lin} \mid \mathsf{M}\,(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \,\rangle \in \mathbb{R}^{n\times m}, \tag{3.47}$$
$$\mathbf{C}(\bar{\mathbf{x}}, \bar{\mathbf{u}}) = \langle\, \mathsf{C}_{lin} \mid \mathsf{M}\,(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \,\rangle \in \mathbb{R}^{p\times n}, \tag{3.48}$$
$$\mathbf{D}(\bar{\mathbf{x}}, \bar{\mathbf{u}}) = \langle\, \mathsf{D}_{lin} \mid \mathsf{M}\,(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \,\rangle \in \mathbb{R}^{p\times m}, \tag{3.49}$$

with parameter tensors (3.42) to (3.45).

Remark 3.1 *The system matrices can be computed directly from the parameter tensors of the MTI system by simple operations like the mode-k tensor matrix product to compute the parameter tensors of the partial derivatives (2.56) or the contracted product to evaluate the parameter tensors at the operating points. If the parameter tensors of the MTI system (3.40) and (3.41) are available in a decomposed format, these operations can be applied directly to the decomposed versions of the tensors without recomputing the full tensors, leading to an*

efficient computation. The parameter tensors A_{lin}, B_{lin}, C_{lin} *and* D_{lin} *have to be computed only once for an MTI system. The linearization can be very easily adapted to the desired operating point, since the linear system matrices result from an evaluation of the contracted product of parameter tensors of the linearization and the monomial tensor at the operating point (3.46) to (3.49). With this structured computation of the linearization local properties around the operating point of the MTI system can be simply determined, e.g. by investigating the eigenvalues of the state matrix* $\mathbf{A}(\bar{\mathbf{x}}, \bar{\mathbf{u}})$. *This gives an idea on the dynamical behavior of the system or the stability in the neighborhood of the operating point, [4]. This is an advantage since e.g. no stability theory especially for MTI systems besides the general nonlinear theory is available yet.*

Example 3.15 *Consider an MTI system with one state and one input with state equation*

$$\begin{aligned}\Phi(x_1) &= f(1,1) + f(1,2)x_1 + f(2,1)u_1 + f(2,2)x_1u_1 \\ &= \langle\, \mathsf{F} \mid \mathsf{M}(x_1, u_1) \,\rangle = \left\langle \begin{pmatrix} f(1,1) & f(1,2) \\ f(2,1) & f(2,2) \end{pmatrix} \middle| \begin{pmatrix} 1 & x_1 \\ u_1 & x_1u_1 \end{pmatrix} \right\rangle.\end{aligned}$$

The linearization of the state equation of the MTI system around the operating point $(\bar{x}_1, \bar{u}_1)$

$$\Phi(x_1) = a(\bar{x}_1, \bar{u}_1)x_1 + b(\bar{x}_1, \bar{u}_1)u_1$$

is computed by the partial derivatives of the state equation with respect to x_1 *and* u_1 *by (2.56). The parameter tensors of the system matrices are thus given by*

$$\begin{aligned}\mathbf{A}_{lin}(:,:,1,1) &= \mathsf{F}_{x_1} = \mathsf{F} \times_2 \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} f(1,2) & 0 \\ f(2,2) & 0 \end{pmatrix}, \\ \mathbf{B}_{lin}(:,:,1,1) &= \mathsf{F}_{u_1} = \mathsf{F} \times_1 \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} f(2,1) & f(2,2) \\ 0 & 0 \end{pmatrix},\end{aligned}$$

with dimensions $\mathsf{A}_{lin}, \mathsf{B}_{lin} \in \mathbb{R}^{2\times2\times1\times1}$. *This results in the system matrices of the linear approximation of the MTI state equation*

$$\begin{aligned}a(\bar{x}_1, \bar{u}_1) &= \langle\, \mathsf{A}_{lin} \mid \mathsf{M}(\bar{x}_1, \bar{u}_1) \,\rangle = \left\langle \begin{pmatrix} f(1,2) & 0 \\ f(2,2) & 0 \end{pmatrix} \middle| \begin{pmatrix} 1 & \bar{x}_1 \\ \bar{u}_1 & \bar{x}_1\bar{u}_1 \end{pmatrix} \right\rangle = f(1,2) + f(2,2)\bar{u}_1, \\ b(\bar{x}_1, \bar{u}_1) &= \langle\, \mathsf{B}_{lin} \mid \mathsf{M}(\bar{x}_1, \bar{u}_1) \,\rangle = \left\langle \begin{pmatrix} f(2,1) & f(2,2) \\ 0 & 0 \end{pmatrix} \middle| \begin{pmatrix} 1 & \bar{x}_1 \\ \bar{u}_1 & \bar{x}_1\bar{u}_1 \end{pmatrix} \right\rangle = f(2,1) + f(2,2)\bar{x}_1.\end{aligned}$$

With that an operating point depended description of the linearization of the MTI system is found

$$\Phi(x_1) = a(\bar{x}_1, \bar{u}_1)x_1 + b(\bar{x}_1, \bar{u}_1)u_1 = \left(f(1,2) + f(2,2)\bar{u}_1\right)x_1 + \left(f(2,1) + f(2,2)\bar{x}_1\right)u_1.$$

The approach can be applied in the same way to the output equation.

3.4 Discretization

When using continuous-time state space models, it is assumed that the signals are available at each time $t \in \mathbb{R}$. When focusing on e.g. implementation of controllers, algorithms are computed on digital computers with a fixed sampling time. The signals are not available at arbitrary times t but at fixed time steps $t = kT_{sample},\ k \in \mathbb{N}$, where T_{sample} is the sampling time. To get a system representation in discrete-time the continuous-time system is extended by a hold and a sample element as shown in Figure 3.11.

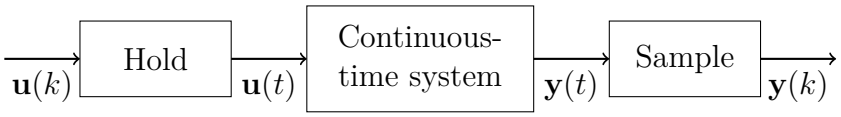


Figure 3.11: Discrete-time system

By the hold block the discrete input value $\mathbf{u}(k)$ is held constant until the next input value $\mathbf{u}(k+1)$ arrives, which results in a constant signal $\mathbf{u}(t)$. The continuous-time system has an output $\mathbf{y}(t)$ that is sampled with sampling time T_{sample} to get the discrete-time output signal $\mathbf{y}(k)$, [23]. Assume that the continuous dynamics are given by a continuous-time MTI system with state equation

$$\dot{\mathbf{x}}(t) = \langle\, \mathsf{F}_c \mid \mathsf{M}\left(\mathbf{x}(t), \mathbf{u}(t)\right) \rangle\,. \tag{3.50}$$

The parameter tensor has an index c here to distinguish between the parameter tensors of the continuous- and discrete-time systems. The parameter tensor of the discrete-time model is denoted by F_d. The discrete-time model is computed by difference equations but has the same structure of the right hand side than in the continuous case regarding the separation of parameters and multilinear monomials. The next state of the corresponding discrete-time system is

$$\mathbf{x}(k+1) = \langle\, \mathsf{F}_d \mid \mathsf{M}\left(\mathbf{x}(k), \mathbf{u}(k)\right) \rangle\,, \tag{3.51}$$

where the parameters in F_d are not equal to the ones in F_c and depend on the sampling time T_{sample}. A conversion from the continuos-time model to the discrete-time model is available, e.g. for linear systems, [23]. This is investigated here now for MTI systems. One discretization method is the Euler forward technique, where the time derivative of the states $\dot{\mathbf{x}}$ is approximated by a first order approach

$$\dot{\mathbf{x}}(kT_{sample}) \approx \frac{\mathbf{x}(k+1) - \mathbf{x}(k)}{T_{sample}}.$$

Rearranging and inserting the right hand side of the continuous-time state equation (3.50) gives

$$\mathbf{x}(k+1) = \mathbf{x}(k) + T_{sample}\dot{\mathbf{x}}(kT_{sample}) = \mathbf{x}(k) + T_{sample}\,\langle\, \mathsf{F}_c \mid \mathsf{M}\left(\mathbf{x}(k), \mathbf{u}(k)\right) \rangle\,. \tag{3.52}$$

The task is to find a description of the parameter tensor F_d, such that (3.52) is fulfilled. Therefore, the state vector $\mathbf{x}(k)$ can be described in the tensor framework

$$\mathbf{x}(k) = \langle\, \mathsf{F}_{\mathbf{x}} \mid \mathsf{M}\left(\mathbf{x}(k), \mathbf{u}(k)\right) \rangle\,, \tag{3.53}$$

by simply selecting the elements $x_j,\ j = 1, \ldots, n$ of the monomial tensor with the parameter tensor $\mathsf{F}_{\mathbf{x}}$. The parameter tensor $\mathsf{F}_{\mathbf{x}}$ can be constructed directly as CP tensor too by

$$\mathsf{F}_{\mathbf{x}} = [\mathbf{F}_{\mathbf{x},u_m}, \ldots, \mathbf{F}_{\mathbf{x},u_1}, \mathbf{F}_{\mathbf{x},x_n}, \ldots, \mathbf{F}_{\mathbf{x},x_1}, \mathbf{F}_{\mathbf{x},\Phi}] \cdot \boldsymbol{\lambda}_{F_x}.$$

Therefore, the columns of the factor matrices $\mathbf{F}_{\mathbf{x},u_j}$ belonging to the inputs are set to

$$\mathbf{f}_{\mathbf{x},u_j}(:,k) = \begin{pmatrix} 1 & 0 \end{pmatrix}^T,$$

with $j = 1, \ldots, m$ and $k = 1, \ldots, n$ because only the states should be selected. The different states are selected by

$$\mathbf{f}_{\mathbf{x},x_j}(:,k) = \begin{cases} \begin{pmatrix} 0 & 1 \end{pmatrix}^T & \text{, for } k = j, \\ \begin{pmatrix} 1 & 0 \end{pmatrix}^T & \text{, otherwise,} \end{cases}$$

with $j = 1, \ldots, n$ and $k = 1, \ldots, n$ and ordered in the resulting vector by

$$\mathbf{F}_{\mathbf{x},\Phi} = \mathbf{I}_n,$$

where $\mathbf{I}_n$ is an n by n identity matrix. The weighting vector $\boldsymbol{\lambda}_{F_x}$ is a vector full of ones. With this tensor the states are selected from the monomial tensor, such that(3.53) is fulfilled. Using the derived description of the state vector (3.53) in (3.52) gives

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{x}(k) + T_{sample} \langle\, \mathsf{F}_c \mid \mathsf{M}\left(\mathbf{x}(k), \mathbf{u}(k)\right) \rangle \\ &= \langle\, \mathsf{F}_{\mathbf{x}} \mid \mathsf{M}\left(\mathbf{x}(k), \mathbf{u}(k)\right) \rangle + T_{sample} \langle\, \mathsf{F}_c \mid \mathsf{M}\left(\mathbf{x}(k), \mathbf{u}(k)\right) \rangle \\ &= \langle\, \mathsf{F}_{\mathbf{x}} + T_{sample}\mathsf{F}_c \mid \mathsf{M}\left(\mathbf{x}(k), \mathbf{u}(k)\right) \rangle\,. \end{aligned}$$

Proposition 3.1 (Discretization of MTI systems in tensor format) *The parameter tensor of the state equation F_d of the Euler discretization (3.51) with sampling time T_{sample} of a continuous-time MTI system (3.50) with parameter tensor F_c is computed by*

$$\mathsf{F}_d = \mathsf{F}_{\mathbf{x}} + T_{sample}\mathsf{F}_c. \tag{3.54}$$

Until now the state equation was considered only. For the output equation no transformation is necessary. The output equation can be used as well in the continuous- as in the discrete-time case with the same parameter tensor G.

Remark 3.2 *It was shown, that the parameter tensor $\mathsf{F}_{\mathbf{x}}$ can be constructed in a CP format. As introduced in the Subsections 3.2.1 to 3.2.4, the CP format can be converted to other decomposed forms. Thus, if the parameter tensor is given in a decomposed format, $\mathsf{F}_{\mathbf{x}}$ can be given in the same format and the parameter tensor of the discrete-time system is in decomposed form, too, because addition and multiplication with a scalar are defined for the decompositions. The decomposed structure is not lost by the discretization. No full tensor has to be build, if the continuous-time system is given in decomposed form.*

Example 3.16 *Assume that the parameter tensor of the state equation of a continuous-time MTI system*

$$\begin{pmatrix}\dot{x}_1\\ \dot{x}_2\end{pmatrix} = \begin{pmatrix}-2x_2 + x_1x_2u_1\\ -x_1u_1\end{pmatrix} = \langle\, \mathsf{F}_c \mid \mathsf{M}\,(x_1, x_2, u_1)\,\rangle\,,$$

is given in CP decomposition

$$\mathsf{F}_c = [\mathbf{F}_{c,u_1}, \mathbf{F}_{c,x_2}, \mathbf{F}_{c,x_1}] \cdot \boldsymbol{\lambda}_{F_c} = \left[\begin{pmatrix}1&0&0\\0&1&1\end{pmatrix}, \begin{pmatrix}0&0&1\\1&1&0\end{pmatrix}, \begin{pmatrix}1&0&0\\0&1&1\end{pmatrix}, \begin{pmatrix}1&1&0\\0&0&1\end{pmatrix}\right] \cdot \begin{pmatrix}-2\\1\\-1\end{pmatrix}.$$

The system should be discretized with sampling time T_{sample}. The parameter tensor of the discrete-time system is computed by (3.54) with the tensor

$$\mathsf{F}_{\mathbf{x}} = \left[\begin{pmatrix}1&1\\0&0\end{pmatrix}, \begin{pmatrix}1&0\\0&1\end{pmatrix}, \begin{pmatrix}0&1\\1&0\end{pmatrix}, \begin{pmatrix}1&0\\0&1\end{pmatrix}\right] \cdot \begin{pmatrix}1\\1\end{pmatrix},$$

and results in the CP description by simply concatenating the factor matrices and weighting vector

$$\begin{aligned}\mathsf{F}_d &= \mathsf{F}_{\mathbf{x}} + T_{sample}\mathsf{F}_c = [\mathbf{F}_{d,u_1}, \mathbf{F}_{d,x_2}, \mathbf{F}_{d,x_1}] \cdot \boldsymbol{\lambda}_{F_d}\\ &= \left[\begin{pmatrix}1&1&1&0&0\\0&0&0&1&1\end{pmatrix}, \begin{pmatrix}1&0&0&0&1\\0&1&1&1&0\end{pmatrix}, \begin{pmatrix}0&1&1&0&0\\1&0&0&1&1\end{pmatrix}, \begin{pmatrix}1&0&1&1&0\\0&1&0&0&1\end{pmatrix}\right] \cdot \begin{pmatrix}1\\1\\-2T_{sample}\\T_{sample}\\-T_{sample}\end{pmatrix}.\end{aligned}$$

This gives the discretized state space model of the continuous-time MTI system

$$\begin{pmatrix}x_1(k+1)\\ x_2(k+1)\end{pmatrix} = \langle\, \mathsf{F}_d \mid \mathsf{M}\,(x_1(k), x_2(k), u_1(k))\,\rangle$$

in a CP tensor framework.

3.5 Scaling

In most applications dynamical systems have a typical operating range. For a continuous-time MTI system

$$\dot{\mathbf{x}} = \langle\, \mathsf{F} \mid \mathsf{M}\,(\mathbf{x}, \mathbf{u})\,\rangle\,, \tag{3.55}$$
$$\mathbf{y} = \langle\, \mathsf{G} \mid \mathsf{M}\,(\mathbf{x}, \mathbf{u})\,\rangle\,, \tag{3.56}$$

the operating area is e.g. bounded by

$$x_i \in [x_{i,l}, x_{i,u}]\,,\ i = 1, \ldots, n, \tag{3.57}$$
$$u_i \in [u_{i,l}, u_{i,u}]\,,\ i = 1, \ldots, m, \tag{3.58}$$
$$y_i \in [y_{i,l}, y_{i,u}]\,,\ i = 1, \ldots, p. \tag{3.59}$$

When using the state space models, e.g. for an optimization algorithm as in MPC, numerical problems may arise, if the signals are of different magnitude. Thus, it makes sense to scale the signals, such that they are in fixed intervals of same magnitude, e.g. by scaling all signals to the interval $[0, 1]$. The desired intervals for the scaled states and inputs are given by

$$\tilde{x}_i \in [\tilde{x}_{i,l}, \tilde{x}_{i,u}], \ i = 1, \ldots, n, \tag{3.60}$$
$$\tilde{u}_i \in [\tilde{u}_{i,l}, \tilde{u}_{i,u}], \ i = 1, \ldots, m, \tag{3.61}$$
$$\tilde{y}_i \in [\tilde{y}_{i,l}, \tilde{y}_{i,u}], \ i = 1, \ldots, p. \tag{3.62}$$

This numerical preconditioning is computed by a linear transformation of the states and inputs

$$x_i = \tilde{a}_i \tilde{x}_i + \tilde{b}_i, \ i = 1, \ldots, n, \tag{3.63}$$
$$u_i = \tilde{a}_{n+i} \tilde{u}_i + \tilde{b}_{n+i}, \ i = 1, \ldots, m, \tag{3.64}$$
$$y_i = \tilde{c}_i \tilde{y}_i + \tilde{e}_i, \ i = 1, \ldots, p, \tag{3.65}$$

where the scaling factors result from the operating range (3.57) to (3.59) and the chosen intervals (3.60) to (3.62) of the states, inputs and outputs, which is shown for the states, $i = 1, \ldots, n$ here

$$\tilde{a}_i = \frac{x_{i,l} - x_{i,u}}{\tilde{x}_{i,l} - \tilde{x}_{i,u}}, \tag{3.66}$$
$$\tilde{b}_i = x_{i,l} - a_i \tilde{x}_{i,l}. \tag{3.67}$$

The scaling factors of inputs and outputs are computed in a similar way. The approach of scaling of MTI systems in matrix representation was shown in [50] and [85]. Here the linear transformation of MTI systems in tensor representation is introduced.

Lemma 3.2 (Scaling of MTI systems in tensor format) *The operating range (3.57) to (3.59) of a continuous-time MTI system with state space representation (3.55) and (3.56) is scaled to an operating range (3.60) to (3.62) by a linear transformation of the states, inputs and outputs as in (3.66) and (3.67). The transformed state equation is given by*

$$\dot{\tilde{\mathbf{x}}} = \left\langle \tilde{\mathsf{F}} \mid \mathsf{M}\left(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}\right) \right\rangle,$$

with the parameter tensor

$$\tilde{\mathsf{F}} = \mathsf{F} \times_1 \begin{pmatrix} \tilde{b}_{n+m} & 0 \\ \tilde{a}_{n+m} & 1 \end{pmatrix} \times_2 \begin{pmatrix} \tilde{b}_{n+m-1} & 0 \\ \tilde{a}_{n+m-1} & 1 \end{pmatrix} \times_3 \cdots \times_{n+m} \begin{pmatrix} \tilde{b}_1 & 0 \\ \tilde{a}_1 & 1 \end{pmatrix} \times_{n+m+1} \underset{i=1,\ldots,n}{\operatorname{diag}} \left(\tilde{a}_i^{-1}\right). \tag{3.68}$$

The scaled output equation

$$\begin{aligned} \tilde{\mathbf{y}} &= \left\langle \hat{\mathsf{G}} - \mathsf{E} \mid \mathsf{M}\left(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}\right) \right\rangle \\ &= \left\langle \tilde{\mathsf{G}} \mid \mathsf{M}\left(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}\right) \right\rangle, \end{aligned}$$

has a parameter tensor that is composed of two tensors

$$\hat{\mathsf{G}} = \mathsf{G} \times_1 \begin{pmatrix} \tilde{b}_{n+m} & 0 \\ \tilde{a}_{n+m} & 1 \end{pmatrix} \times_2 \begin{pmatrix} \tilde{b}_{n+m-1} & 0 \\ \tilde{a}_{n+m-1} & 1 \end{pmatrix} \times_3 \cdots \times_{n+m} \begin{pmatrix} \tilde{b}_1 & 0 \\ \tilde{a}_1 & 1 \end{pmatrix} \times_{n+m+1} \underset{i=1,\ldots,n}{\operatorname{diag}} \left(\tilde{c}_i^{-1} \right),$$
$$\mathsf{E} = [\mathbf{E}_{u_m}, \ldots, \mathbf{E}_{u_1}, \mathbf{E}_{x_n}, \ldots, \mathbf{E}_{x_1}, \mathbf{E}_\Phi] \cdot \boldsymbol{\lambda}_E.$$

The CP factors of E *are*

$$\mathbf{E}_{x_i} = \mathbf{E}_{u_j} = \begin{pmatrix} 1 & \cdots & 1 \\ 0 & \cdots & 0 \end{pmatrix} \in \mathbb{R}^{2 \times p}, \ i = 1, \ldots, n, \ j = 1, \ldots, m, \tag{3.69}$$

$$\mathbf{E}_\Phi = \mathbf{I}_p, \tag{3.70}$$

$$\boldsymbol{\lambda}_E = \underset{i=1,\ldots,n}{\operatorname{diag}} \left(\tilde{c}_i \right)^{-1} \begin{pmatrix} \tilde{e}_1 & \cdots & \tilde{e}_p \end{pmatrix}^T. \tag{3.71}$$

The proof of the lemma is given in the Appendix A.7.

Remark 3.3 *For the transformation of the parameter tensors a sequence of mode-k tensor matrix products has to be computed only, which is defined for all four proposed decomposition techniques. Since the tensor* E *is given in CP format it can be translated to the other decomposed representations as well. Thus, this tool can be be used for all decomposed representations of MTI systems. The resulting parameter tensors of the scaled system are also decomposed in the same format.*

Example 3.17 *The MTI model introduced in Example 3.14 is used for scaling here, where the state equation is given by*

$$\dot{\mathbf{x}} = \langle \, \mathsf{F} \mid \mathsf{M} \left(x_1, x_2, x_3, u_1, u_2 \right) \rangle \, .$$

The operating range is defined according to the range of the signals in Figure 3.9 resulting in

$$x_1 \in [0, 1] \, , \ x_2 \in [0, 10] \, , \ x_3 \in [-2, 2] \, , \ u_1 \in [0, 2] \, , \ u_2 \in [-1, 1] \, .$$

All scaled signals $\tilde{x}_i$, $\tilde{u}_i$ *should stay in the interval* $[0, 1]$*. Using (3.66) and (3.67), this leads e.g. for the state* x_2 *to the scaling factors*

$$\tilde{a}_2 = \frac{0 - 10}{0 - 1} = 10,$$
$$\tilde{b}_2 = 0 - 10 \cdot 0 = 0,$$

such that the scaling from x_2 *to* $\tilde{x}_2$ *follows from*

$$\tilde{x}_2 = \frac{1}{\tilde{a}_2} \left(x_2 - \tilde{b}_2 \right) = \frac{1}{10} x_2.$$

With the other scaling factors and (3.68) the parameter tensor of the state equation of the scaled system is given by

$$\tilde{\mathsf{F}} = \mathsf{F} \times_1 \begin{pmatrix} \tilde{b}_5 & 0 \\ \tilde{a}_5 & 1 \end{pmatrix} \times_2 \begin{pmatrix} \tilde{b}_4 & 0 \\ \tilde{a}_4 & 1 \end{pmatrix} \times_3 \begin{pmatrix} \tilde{b}_3 & 0 \\ \tilde{a}_3 & 1 \end{pmatrix} \times_4 \begin{pmatrix} \tilde{b}_2 & 0 \\ \tilde{a}_2 & 1 \end{pmatrix} \times_5 \begin{pmatrix} \tilde{b}_1 & 0 \\ \tilde{a}_1 & 1 \end{pmatrix} \times_6 \begin{pmatrix} \frac{1}{\tilde{a}_1} & 0 & 0 \\ 0 & \frac{1}{\tilde{a}_2} & 0 \\ 0 & 0 & \frac{1}{\tilde{a}_3} \end{pmatrix}.$$

The simulation of the scaled model with state equation

$$\dot{\tilde{\mathbf{x}}} = \left\langle \tilde{\mathsf{F}} \mid \mathsf{M}\left(\tilde{x}_1, \tilde{x}_2, \tilde{x}_3, \tilde{u}_1, \tilde{u}_2\right) \right\rangle,$$

with the same but scaled input than in Figure 3.9 results in the state trajectories depicted in Figure 3.12. The figure shows, that the basic state dynamics are the same compared to the original system shown in Figure 3.9. But because of the scaling of the model all states are in the interval $[0, 1]$ *here as desired. The states of the original model are retrieved by (3.63).*

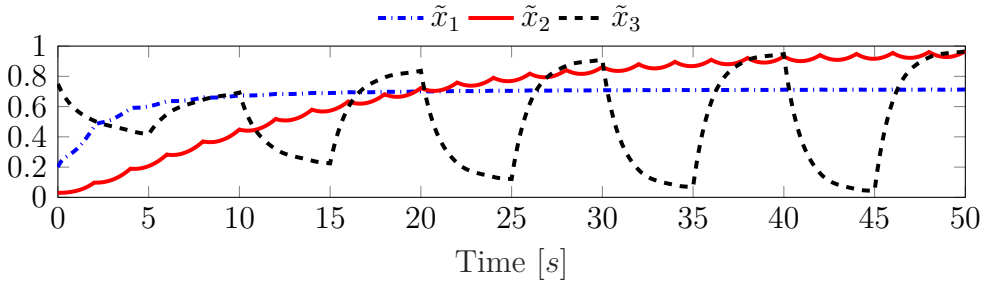


Figure 3.12: Simulation result of the scaled system with parameter tensor $\tilde{\mathsf{F}}$

3.6 Multi-step transitions of discrete-time MTI models

In this subsection the description of multi-step transitions of discrete-time MTI models is investigated, i.e. the computation of future states by the actual state and a given future input trajectory. When simulating the response of a discrete-time model, the state trajectory is computed by iterating the model equations. In the case of linear systems this results in

$$\begin{aligned}
\mathbf{x}(k+1) &= \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k), && (3.72)\\
\mathbf{x}(k+2) &= \mathbf{A}\mathbf{x}(k+1) + \mathbf{B}\mathbf{u}(k+1) && (3.73)\\
&= \mathbf{A}^2\mathbf{x}(k) + \mathbf{A}\mathbf{B}\mathbf{u}(k) + \mathbf{B}\mathbf{u}(k+1), && (3.74)\\
\mathbf{x}(k+3) &= \mathbf{A}\mathbf{x}(k+2) + \mathbf{B}\mathbf{u}(k+2) && (3.75)\\
&= \mathbf{A}^3\mathbf{x}(k) + \mathbf{A}^2\mathbf{B}\mathbf{u}(k) + \mathbf{A}\mathbf{B}\mathbf{u}(k+1) + \mathbf{B}\mathbf{u}(k+2), && (3.76)\\
&\vdots\\
\mathbf{x}(k+N) &= \mathbf{A}\mathbf{x}(k+N-1) + \mathbf{B}\mathbf{u}(k+N-1) && (3.77)\\
&= \mathbf{A}^N\mathbf{x}(k) + \mathbf{A}^{N-1}\mathbf{B}\mathbf{u}(k) + \mathbf{A}^{N-2}\mathbf{B}\mathbf{u}(k+1) + \cdots + \mathbf{B}\mathbf{u}(k+N-1). && (3.78)
\end{aligned}$$

This shows, that with linear systems, it is possible to describe the simulated future states explicitly in closed form as a linear function depending on the initial state $\mathbf{x}(k)$ and the inputs $\mathbf{u}(k+i)$, $i = 0, \ldots, N-1$. This description is called lifted system and is useful, e.g. for the formulation of the cost function for the optimization problem of MPC, which is described in Section 4, [73]. This poses the question, if an explicit formulation of the future states $\mathbf{x}(k+i)$ is possible too, when MTI models are used. With an MTI system given as discrete-time state space model in tensor framework

$$\mathbf{x}(k+1) = \left\langle \mathsf{F} \mid \mathsf{M}\left(\mathbf{x}(k), \mathbf{u}(k)\right) \right\rangle,$$

the goal is to describe the future states $\mathbf{x}(k+i)$ by the initial state $\mathbf{x}(k)$ and the input trajectory $\mathbf{u}(k)$ to $\mathbf{u}(k+i-1)$. With a linear system it was shown that the future states are computed by linear functions too, i.e. the multi-step transitions stay in the class of linear systems. The following example shows the situation for multilinear models.

Example 3.18 *Consider the state equations of a discrete-time multilinear system with two states and one input*

$$\begin{aligned} x_1(k+1) &= -x_1(k)x_2(k), \\ x_2(k+1) &= -x_2(k) + u(k). \end{aligned}$$

The states of the next time step k+2 are computed by iterating the state equation resulting in

$$\begin{aligned} x_1(k+2) &= -x_1(k+1)x_2(k+1) = -x_1(k)x_2(k)^2 + x_1(k)x_2(k)u(k), \\ x_2(k+2) &= -x_2(k+1) + u(k+1) = x_2(k) - u(k) + u(k+1). \end{aligned}$$

The example shows that already with one iteration, higher order terms like $x_1(k)x_2(k)^2$ in this example, could occur that are no longer multilinear. Therefore, the multi-step transitions have to be described by polynomials of higher order using the tensor framework (2.47). At first, the approach is described here for a general multilinear model with two states and one input

$$\begin{pmatrix} x_1(k+1) \\ x_2(k+1) \end{pmatrix} = \langle\, \mathsf{F} \mid \mathsf{M}\,(x_1(k), x_2(k), u(k))\, \rangle\,,$$

to introduce the idea. Afterwards it is generalized to MTI models with n states and m inputs. The state at time instant $k+2$ is given by

$$\mathbf{x}(k+2) = \langle\, \mathsf{F} \mid \mathsf{M}\,(x_1(k+1), x_2(k+1), u(k+1))\, \rangle\,. \tag{3.79}$$

At first, the monomial tensor is investigated. The particular states read

$$x_i(k+1) = \langle\, \mathsf{F}(:,\ldots,:,i) \mid \mathsf{M}\,(\mathbf{x}(k), \mathbf{u}(k))\, \rangle = \langle\, \mathsf{F} \bar{\times}_{n+m+1} \mathbf{e}_{i,n} \mid \mathsf{M}\,(\mathbf{x}(k), \mathbf{u}(k))\, \rangle\,, \tag{3.80}$$

where $\mathbf{e}_{n,i}$ is the i^{th} unit vector of length n and can be inserted to the state equation (3.79) yielding

$$\begin{aligned} &\mathsf{M}\,(x_1(k+1), x_2(k+1), u(k+1)) = \\ &\mathsf{M}\left(\langle\, \mathsf{F} \bar{\times}_{n+m+1} \mathbf{e}_{1,n} \mid \mathsf{M}\,(\mathbf{x}(k), \mathbf{u}(k))\, \rangle, \langle\, \mathsf{F} \bar{\times}_{n+m+1} \mathbf{e}_{2,n} \mid \mathsf{M}\,(\mathbf{x}(k), \mathbf{u}(k))\, \rangle\,, u(k+1)\right). \end{aligned} \tag{3.81}$$

To summarize the terms of the monomial tensor their descriptions have to depend on the same variables, i.e. $x_1(k)$, $x_2(k)$, $u(k)$ and $u(k+1)$. Thus, the parameter tensor is extended, leading to

$$\mathbf{x}(k+1) = \langle\, \mathsf{F} \mid \mathsf{M}\,(x_1(k), x_2(k), u(k), u(k+1))\, \rangle\,,$$

with elements

$$x_i(k+1) = \langle\, \mathsf{F} \bar{\times}_{n+2m+1} \mathbf{e}_{i,n} \mid \mathsf{M}\,(x_1(k), x_2(k), u(k), u(k+1))\, \rangle\,.$$

This extension is easily constructed by adding zeros in the parameter tensor in each position, that belongs to a monomial that contains the input $u(k+1)$. Also the input at time $k+1$ can be expressed in a tensor framework

$$u(k+1) = \langle \mathsf{F}_u \mid \mathsf{M}\left(x_1(k), x_2(k), u(k), u(k+1)\right) \rangle$$

with the parameter tensor, e.g. in CP representation

$$\mathsf{F}_u = \left[\begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right].$$

The monomial tensor is a rank one tensor, that is written as outer product of its factor vectors by

$$\mathsf{M}\left(x_1(k+1), x_2(k+1), u(k+1)\right) = \begin{pmatrix} 1 \\ u(k+1) \end{pmatrix} \circ \begin{pmatrix} 1 \\ x_2(k+1) \end{pmatrix} \circ \begin{pmatrix} 1 \\ x_1(k+1) \end{pmatrix}.$$

The factor vectors belonging to the states $x_1(k+1)$ and $x_2(k+1)$ are expressed with respect to the initial state and the inputs by inserting (3.80)

$$\begin{aligned} \begin{pmatrix} 1 \\ x_j(k+1) \end{pmatrix} &= \begin{pmatrix} 1 \\ \langle \mathsf{F} \bar{\times}_5 \mathbf{e}_{j,2} \mid \mathsf{M}\left(x_1(k), x_2(k), u(k), u(k+1)\right) \rangle \end{pmatrix} \\ &= \langle \mathsf{1}_4 \boxplus_5 \left(\mathsf{F} \bar{\times}_5 \mathbf{e}_{j,2}\right) \mid \mathsf{M}\left(x_1(k), x_2(k), u(k), u(k+1)\right) \rangle \\ &= \left\langle \bar{\mathsf{F}}_{x_j} \mid \mathsf{M}\left(x_1(k), x_2(k), u(k), u(k+1)\right) \right\rangle, \ j = 1, 2. \end{aligned}$$

The 1 in the first component of the factor vector $\begin{pmatrix} 1 & x_i(k+1) \end{pmatrix}^T$ is added by concatenation to the parameter tensor of the state with a one tensor that is defined by

$$1 = \langle \mathsf{1}_{n+m} \mid \mathsf{M}\left(\mathbf{x}, \mathbf{u}\right) \rangle. \tag{3.82}$$

The parameter tensor is given, e.g. in CP representation, as rank-1 tensor by

$$\mathsf{1}_{n+m} = \left[\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \ldots, \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right] \in \mathbb{R}^{\times^{(n+m)}2},$$

such that it has the property

$$1_{n+m}(i_1, \ldots, i_{n+m}) = \begin{cases} 1 & \text{, for } i_1 = \cdots = i_{n+m} = 1, \\ 0 & \text{, otherwise.} \end{cases}$$

A one tensor is constructed for polynomials of higher order in the same way, such that

$$1 = \left\langle \mathsf{1}^N_{n+m} \,\middle|\, \mathsf{M}^N_p\left(\mathbf{x}, \mathbf{u}\right) \right\rangle$$

holds, with the one tensor $\mathsf{1}^N_{n+m} \in \mathbb{R}^{\times^{(N(n+m))}2}$ for polynomials of maximal order N. The approach is applied to the factor vector of the input in the same way resulting in

$$\begin{aligned} \begin{pmatrix} 1 \\ u(k+1) \end{pmatrix} &= \langle \mathsf{1}_4 \boxplus_5 \mathsf{F}_u \mid \mathsf{M}\left(x_1(k), x_2(k), u(k), u(k+1)\right) \rangle \\ &= \left\langle \bar{\mathsf{F}}_u \mid \mathsf{M}\left(x_1(k), x_2(k), u(k), u(k+1)\right) \right\rangle. \end{aligned}$$

All factor vectors are described with respect to the initial state and the inputs. Using these definitions, the monomial tensor (3.81) for the state at time $k+2$ is written with respect to the desired variables by

$$\begin{aligned}
&\mathsf{M}\left(x_1(k+1), x_2(k+1), u(k+1)\right) \\
&= \left\langle \bar{\mathsf{F}}_u \mid \mathsf{M}\left(x_1(k), x_2(k), u(k), u(k+1)\right) \right\rangle \circ \left\langle \bar{\mathsf{F}}_{x_2} \mid \mathsf{M}\left(x_1(k), x_2(k), u(k), u(k+1)\right) \right\rangle \\
&\circ \left\langle \bar{\mathsf{F}}_{x_1} \mid \mathsf{M}\left(x_1(k), x_2(k), u(k), u(k+1)\right) \right\rangle \\
&= \left\langle \mathsf{F}_M^2 \,\middle|\, \mathsf{M}_p^3\left(x_1(k), x_2(k), u(k), u(k+1)\right) \right\rangle_{4:15,1:12},
\end{aligned}$$

where the parameter tensor $\mathsf{F}_M^2 \in \mathbb{R}^{\times_{(15)}2}$ is constructed by

$$\mathsf{F}_M^2(i_u, i_{x_2}, i_{x_1}, :, \ldots, :) = \bar{\mathsf{F}}_u(:, \ldots, :, i_u) \circ \bar{\mathsf{F}}_{x_2}(:, \ldots, :, , i_{x_2}) \circ \bar{\mathsf{F}}_{x_1}(:, \ldots, :, , i_{x_1}).$$

The contracted product is computed over the dimensions 4 to 15 of the parameter tensor F_M^2 and all twelve dimensions of the monomial tensor $\mathsf{M}_p^3\left(x_1(k), x_2(k), u(k), u(k+1)\right)$, because the first three dimensions of F_M^2 belong to the three indices i_u, i_{x_2} and i_{x_1} of the resulting multilinear monomial tensor $\mathsf{M}\left(x_1(k+1), x_2(k+1), u(k+1)\right)$. With this description of the monomial tensor, the states at time $k+2$ are given by

$$\begin{aligned}
\mathbf{x}(k+2) &= \left\langle \mathsf{F} \mid \mathsf{M}\left(x_1(k+1), x_2(k+1), u(k+1)\right) \right\rangle \\
&= \left\langle \mathsf{F} \,\middle|\, \left\langle \mathsf{F}_M^2 \,\middle|\, \mathsf{M}_p^3\left(x_1(k), x_2(k), u(k), u(k+1)\right) \right\rangle_{4:15,1:12} \right\rangle. \qquad (3.83)
\end{aligned}$$

In this description the monomial tensor contains parameters, such that the parameters and the monomials are not separated yet, as it should be for the tensor framework introduced in Section 2.3.1. From the elementwise definition of the contracted product (2.4) the separation of monomials and parameters follows by rearranging

$$\begin{aligned}
x_j(k+2) &= \sum_{i_u=1}^{2} \sum_{i_{x_2}=1}^{2} \sum_{i_{x_1}=1}^{2} f(i_u, i_{x_2}, i_{x_1}, j) \\
&\cdot \sum_{k_1=1}^{2} \cdots \sum_{k_{12}=1}^{2} f_M^2(i_u, i_{x_2}, i_{x_1}, k_1, \ldots, k_{12}) m_p^3\left(x_1(k), x_2(k), u(k), u(k+1)\right)(k_1, \ldots, k_{12}) \\
&= \sum_{k_1=1}^{2} \cdots \sum_{k_{12}=1}^{2} \underbrace{\sum_{i_u=1}^{2} \sum_{i_{x_1}=1}^{2} \sum_{i_{x_2}=1}^{2} f_M^2(i_u, i_{x_2}, i_{x_1}, k_1, \ldots, k_{12}) f(i_u, i_{x_2}, i_{x_1}, j)}_{\left\langle \mathsf{F}_M^2 \mid \mathsf{F} \right\rangle_{1:3}} \\
&\cdot m_p^3\left(x_1(k), x_2(k), u(k), u(k+1)\right)(k_1, \ldots, k_{12}). \qquad (3.84)
\end{aligned}$$

Thus, the states at time $k+2$ are given as a polynomial system of order 3 by

$$\begin{aligned}
\mathbf{x}(k+2) &= \left\langle \left\langle \mathsf{F}_M^2 \mid \mathsf{F} \right\rangle_{1:3} \,\middle|\, \mathsf{M}_p^3\left(x_1(k), x_2(k), u(k), u(k+1)\right) \right\rangle \\
&= \left\langle \mathsf{F}^2 \,\middle|\, \mathsf{M}_p^3\left(x_1(k), x_2(k), u(k), u(k+1)\right) \right\rangle.
\end{aligned}$$

This shows that an explicit description of the two step transition of an MTI system can be found in the tensor framework resulting in a polynomial system. This result is used in the following for the next step to compute $\mathbf{x}(k+3)$. From the state equations, the states at time $k+3$ are given using the states of the previous time step by

$$\mathbf{x}(k+3) = \langle\, \mathsf{F} \mid \mathsf{M}\left(x_1(k+2), x_2(k+2), u(k+2)\right) \rangle\,.$$

Again the state evaluations (3.83) from the previous time step, i.e. $k+2$, are inserted. At first the parameter tensors are extended, such that their monomial tensors depend on the same variables. Here the dependence on $u(k+2)$ is added

$$\begin{aligned}
\mathbf{x}(k+2) &= \left\langle\, \mathsf{F}^2 \;\middle|\; \mathsf{M}_p^3\left(x_1(k), x_2(k), u(k), u(k+1), u(k+2)\right) \right\rangle, \\
u(k+2) &= \left\langle\, \mathsf{F}_u \;\middle|\; \mathsf{M}_p^3\left(x_1(k), x_2(k), u(k), u(k+1), u(k+2)\right) \right\rangle.
\end{aligned}$$

Putting the states and inputs into the monomial vector, it is written as

$$\begin{aligned}
\mathsf{M}\left(x_1(k+2), x_2(k+2), u(k+2)\right) = \mathsf{M}\Big(&\Big\langle \mathsf{F}^2 \bar{\times}_{16} \mathbf{e}_{1,2} \;\Big|\; \mathsf{M}_p^3\left(x_1(k), x_2(k), u(k), u(k+1), u(k+2)\right)\Big\rangle, \\
&\left\langle \mathsf{F}^2 \bar{\times}_{16} \mathbf{e}_{2,2} \;\middle|\; \mathsf{M}_p^3\left(x_1(k), x_2(k), u(k), u(k+1), u(k+2)\right) \right\rangle, \\
&\left\langle \mathsf{F}_{u(k+2)} \;\middle|\; \mathsf{M}_p^3\left(x_1(k), x_2(k), u(k), u(k+1), u(k+2)\right) \right\rangle\Big)
\end{aligned}$$

and its rank-1 factors for the states read

$$\begin{aligned}
\begin{pmatrix} 1 \\ x_j(k+2) \end{pmatrix} &= \left(\left\langle\, \mathsf{F}^2 \bar{\times}_{16} \mathbf{e}_{j,2} \;\middle|\; \mathsf{M}_p^3\left(x_1(k), x_2(k), u(k), u(k+1), u(k+2)\right) \right\rangle \right) \\
&= \left\langle\, 1_5^3 \boxplus_{16} \left(\mathsf{F}^2 \bar{\times}_{16} \mathbf{e}_{j,2} \right) \;\middle|\; \mathsf{M}_p^3\left(x_1(k), x_2(k), u(k), u(k+1), u(k+2)\right) \right\rangle \\
&= \left\langle\, \bar{\mathsf{F}}_{x_j}^2 \;\middle|\; \mathsf{M}_p^3\left(x_1(k), x_2(k), u(k), u(k+1), u(k+2)\right) \right\rangle, \; j = 1, 2.
\end{aligned}$$

The same holds for the factor matrix of the input $u(k+2)$

$$\begin{pmatrix} 1 \\ u(k+2) \end{pmatrix} = \left\langle\, \bar{\mathsf{F}}_u^2 \;\middle|\; \mathsf{M}_p^3\left(x_1(k), x_2(k), u(k), u(k+1), u(k+2)\right) \right\rangle.$$

Using the rank-1 factors the monomial tensor given by

$$\begin{aligned}
\mathsf{M}\left(x_1(k+2), x_2(k+2), u(k+2)\right) = &\left\langle\, \bar{\mathsf{F}}_u^2 \;\middle|\; \mathsf{M}_p^3\left(x_1(k), x_2(k), u(k), u(k+1), u(k+2)\right) \right\rangle \\
&\circ \left\langle\, \bar{\mathsf{F}}_{x_2}^2 \;\middle|\; \mathsf{M}_p^3\left(x_1(k), x_2(k), u(k), u(k+1), u(k+2)\right) \right\rangle \\
&\circ \left\langle\, \bar{\mathsf{F}}_{x_1}^2 \;\middle|\; \mathsf{M}_p^3\left(x_1(k), x_2(k), u(k), u(k+1), u(k+2)\right) \right\rangle \\
= &\left\langle\, \mathsf{F}_M^3 \;\middle|\; \mathsf{M}_p^9\left(x_1(k), x_2(k), u(k), u(k+1), u(k+2)\right) \right\rangle_{4:48,1:45}
\end{aligned}$$

with the parameter tensor $\mathsf{F}_M^3 \in \mathbb{R}^{\times^{(48)}2}$ constructed by subtensors

$$\mathsf{F}_M^3(i_u, i_{x_2}, i_{x_1}, :, \ldots, :) = \bar{\mathsf{F}}_u(:, \ldots, :, i_u) \circ \bar{\mathsf{F}}_{x_2}(:, \ldots, :, , i_{x_2}) \circ \bar{\mathsf{F}}_{x_1}(:, \ldots, :, , i_{x_1}).$$

As before, this description of the monomial tensor gives the states at time $k+3$

$$\begin{aligned}\mathbf{x}(k+2) &= \left\langle \mathsf{F} \,\middle|\, \left\langle \mathsf{F}_M^3 \,\middle|\, \mathsf{M}_p^9\left(x_1(k), x_2(k), u(k), u(k+1), u(k+2)\right) \right\rangle_{4:48,1:45} \right\rangle \\ &= \left\langle \left\langle \mathsf{F}_M^2 \mid \mathsf{F} \right\rangle_{1:3} \,\middle|\, \mathsf{M}_p^9\left(x_1(k), x_2(k), u(k), u(k+1), u(k+2)\right) \right\rangle \\ &= \left\langle \mathsf{F}^3 \,\middle|\, \mathsf{M}_p^9\left(x_1(k), x_2(k), u(k), u(k+1), u(k+2)\right) \right\rangle.\end{aligned}$$

To illustrate the main idea of the approach, an MTI system with two states and one input was considered up to this point to avoid an even more complex notation. The results are generalized in the following to MTI systems with n states and m inputs up to $k+i$ time steps. By using the state vector of the previous time step the states at time $k+i$ are given by

$$\mathbf{x}(k+i) = \left\langle \mathsf{F} \mid \mathsf{M}\left(\mathbf{x}(k+i-1), \mathbf{u}(k+i-1)\right) \right\rangle, \; i = 1, \ldots, N. \tag{3.85}$$

The approach introduced before is generalized such that the states $\mathbf{x}(k+i)$ are computed with respect to the initial states $\mathbf{x}(k)$ and the input trajectory $\mathbf{u}(k+j),\ j = 1, \ldots, i-1$ by

$$\mathbf{x}(k+i) = \left\langle \mathsf{F}^i \,\middle|\, \mathsf{M}_p^{(n+m)^{i-1}}\left(\mathbf{x}(k), \mathbf{u}(k), \ldots, \mathbf{u}(k+i-1)\right) \right\rangle.$$

The goal is to determine the parameter tensor F^i. The maximal order of the monomials $(n+m)^{i-1}$ follows from the multilinearity of the state equation (3.85) of the model. This allows multiplications of states of the system. If now the states of the previous time step of order $(n+m)^{i-2}$ are inserted in the monomial tensor, monomials of maximal order $(n+m)^{i-1}$ can occur at time $k+i$. To compute the parameter tensor F^i, it is assumed that the parameter tensor F^{i-1} of the previous time step

$$\mathbf{x}(k+i-1) = \left\langle \mathsf{F}^{i-1} \,\middle|\, \mathsf{M}_p^{(n+m)^{i-2}}\left(\mathbf{x}(k), \mathbf{u}(k), \ldots, \mathbf{u}(k+i-2)\right) \right\rangle$$

is known. To calculate the parameter tensor F^i, at first, the parameter tensor F^{i-1} is extended by adding zeros to get a dependence on $\mathbf{u}(k+i-1)$

$$\mathbf{x}(k+i-1) = \left\langle \mathsf{F}^{i-1} \,\middle|\, \mathsf{M}_p^{(n+m)^{i-2}}\left(\mathbf{x}(k), \mathbf{u}(k), \ldots, \mathbf{u}(k+i-2), \mathbf{u}(k+i-1)\right) \right\rangle.$$

The inputs at time $k+i-1$ are expressed in a tensor framework with the same monomial tensor by

$$\mathbf{u}(k+i-1) = \left\langle \mathsf{F}_u \,\middle|\, \mathsf{M}_p^{(n+m)^{i-2}}\left(\mathbf{x}(k), \mathbf{u}(k), \ldots, \mathbf{u}(k+i-2), \mathbf{u}(k+i-1)\right) \right\rangle.$$

The state and input representations are inserted to the monomial tensor of (3.85). The monomial tensor is a rank-1 tensor, where the elements belonging to the states are

$$\begin{aligned}&\begin{pmatrix} 1 \\ x_j(k+i-1) \end{pmatrix} = \begin{pmatrix} 1 \\ \left\langle \mathsf{F}^{i-1} \bar{\times}_\beta \mathbf{e}_{j,n} \,\middle|\, \mathsf{M}_p^{(n+m)^{i-2}}\left(\mathbf{x}(k), \mathbf{u}(k), \ldots, \mathbf{u}(k+i-1)\right) \right\rangle \end{pmatrix} \\ &= \left\langle \mathbf{1}_{n+im}^{(n+m)^{i-2}} \boxplus_\beta \left(\mathsf{F}^{i-1} \bar{\times}_\beta \mathbf{e}_{j,n}\right) \,\middle|\, \mathsf{M}_p^{(n+m)^{i-2}}\left(\mathbf{x}(k), \mathbf{u}(k), \ldots, \mathbf{u}(k+i-1)\right) \right\rangle \\ &= \left\langle \bar{\mathsf{F}}_{x_j}^i \,\middle|\, \mathsf{M}_p^{(n+m)^{i-2}}\left(\mathbf{x}(k), \mathbf{u}(k), \ldots, \mathbf{u}(k+i-1)\right) \right\rangle, \; j = 1, \ldots, n,\end{aligned} \tag{3.86}$$

where $\beta = (n+m)^{i-2}(n+im)+1$ gives the last dimension of the tensor F^{i-1}, which is of order $(n+m)^{i-2}$ and depends on $(n+im)$ variables. The same approach is applied to the rank-1 factors of the inputs

$$\begin{aligned}\begin{pmatrix}1\\ u_j(k+i-1)\end{pmatrix} &= \begin{pmatrix}1\\ \left\langle\, \mathsf{F}_u \,\bar{\times}_\beta \mathbf{e}_{j,m} \;\middle|\; \mathsf{M}_p^{(n+m)^{i-2}}(\mathbf{x}(k),\mathbf{u}(k),\dots,\mathbf{u}(k+i-1)) \,\right\rangle\end{pmatrix}\\ &= \left\langle\, 1_{n+im}^{(n+m)^{i-2}} \boxplus_\beta (\mathsf{F}_u \,\bar{\times}_\beta \mathbf{e}_{j,m}) \;\middle|\; \mathsf{M}_p^{(n+m)^{i-2}}(\mathbf{x}(k),\mathbf{u}(k),\dots,\mathbf{u}(k+i-1)) \,\right\rangle\\ &= \left\langle\, \bar{\mathsf{F}}^i_{u_j} \;\middle|\; \mathsf{M}_p^{(n+m)^{i-2}}(\mathbf{x}(k),\mathbf{u}(k),\dots,\mathbf{u}(k+i-1)) \,\right\rangle,\ j=1,\dots,m. \end{aligned} \tag{3.87}$$

The monomial tensor is computed by the outer product of the rank-1 factors (3.86) and (3.87)

$$\begin{aligned}&\mathsf{M}(\mathbf{x}(k+i-1),\mathbf{u}(k+i-1)) =\\ &\left\langle \bar{\mathsf{F}}^i_{u_m} \middle| \mathsf{M}_p^{(n+m)^{i-2}}(\mathbf{x}(k),\dots,\mathbf{u}(k+i-1)) \right\rangle \circ\cdots\circ \left\langle \bar{\mathsf{F}}^i_{u_1} \middle| \mathsf{M}_p^{(n+m)^{i-2}}(\mathbf{x}(k),\dots,\mathbf{u}(k+i-1)) \right\rangle\\ &\circ \left\langle \bar{\mathsf{F}}^i_{x_n} \middle| \mathsf{M}_p^{(n+m)^{i-2}}(\mathbf{x}(k),\dots,\mathbf{u}(k+i-1)) \right\rangle \circ\cdots\circ \left\langle \bar{\mathsf{F}}^i_{x_1} \middle| \mathsf{M}_p^{(n+m)^{i-2}}(\mathbf{x}(k),\dots,\mathbf{u}(k+i-1)) \right\rangle\\ &= \left\langle\, \mathsf{F}^i_M \;\middle|\; \mathsf{M}_p^{(n+m)^{i-2}}(\mathbf{x}(k),\mathbf{u}(k),\dots,\mathbf{u}(k+i-1)) \,\right\rangle_{n+m+1:\beta+n+m,1:\beta}, \end{aligned}$$

which allows to describe the monomial tensor in a tensor framework with the parameter tensor

$$\begin{aligned}&\mathsf{F}^i_M(i_{u_m},\dots,i_{u_1},i_{x_n}\dots,i_{u_{x_1}},:,\dots,:)\\ &= \bar{\mathsf{F}}^i_{u_m}(:,\dots,:,i_{u_m})\circ\cdots\circ\bar{\mathsf{F}}^i_{u_1}(:,\dots,:,i_{u_1})\circ\bar{\mathsf{F}}^i_{x_n}(:,\dots,:,i_{x_n})\circ\cdots\circ\bar{\mathsf{F}}^i_{x_1}(:,\dots,:,i_{x_1}), \end{aligned} \tag{3.88}$$

for all $i_{u_j}, i_{x_k} \in [1,2]$. As shown before in (3.84) the state at time $k+i$ is given depending on the initial state $\mathbf{x}(k)$ and the inputs $\mathbf{u}(k)$ to $\mathbf{u}(k+i-1)$ as given by the following proposition.

Proposition 3.2 (Multi-step transition of MTI models) *The multi-step transition of a discrete-time MTI system*

$$\mathbf{x}(k+1) = \langle\, \mathsf{F} \mid \mathsf{M}(\mathbf{x}(k),\mathbf{u}(k)) \,\rangle$$

is written in closed form to determine the states at time $k+1$ by

$$\begin{aligned}\mathbf{x}(k+i) &= \left\langle \left\langle\, \mathsf{F}^i_M \mid \mathsf{F} \,\right\rangle_{1:n+m} \;\middle|\; \mathsf{M}_p^{(n+m)^{i-1}}(\mathbf{x}(k),\mathbf{u}(k),\dots,\mathbf{u}(k+i-1)) \right\rangle\\ &= \left\langle\, \mathsf{F}^i \;\middle|\; \mathsf{M}_p^{(n+m)^{i-1}}(\mathbf{x}(k),\mathbf{u}(k),\dots,\mathbf{u}(k+i-1)) \,\right\rangle, \end{aligned} \tag{3.89}$$

where the parameter tensor is constructed by (3.86) to (3.88).

Example 3.19 *Consider a first order system with one input*

$$x(k+1) = -x(k)+x(k)u(k) = \langle\, \mathsf{F} \mid \mathsf{M}(x(k),u(k)) \,\rangle = \left\langle \begin{pmatrix}0 & -1\\ 0 & 1\end{pmatrix} \middle| \begin{pmatrix}1 & x(k)\\ u(k) & x(k)u(k)\end{pmatrix} \right\rangle. \tag{3.90}$$

At first the parameter tensor F *is extended to add a dependence on* $u(k+1)$*. The extension is done by adding a second slice to the parameter tensor full of zeros, as depicted in Figure 3.13. The first slice contains the parameters of (3.90) leading to*

$$x(k+1) = \langle\, \mathsf{F} \mid \mathsf{M}\left(x(k), u(k), u(k+1)\right) \rangle\,.$$

The input is expressed in a tensor framework by

$$u(k+1) = \langle\, \mathsf{F}_u \mid \mathsf{M}\left(x(k), u(k), u(k+1)\right) \rangle\,,$$

with parameter tensor shown in Figure 3.13.

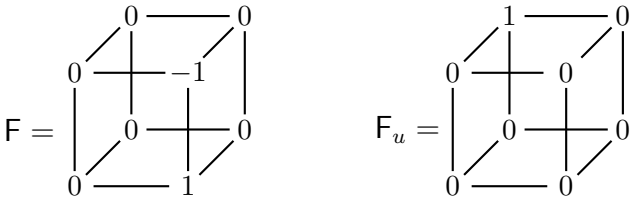


Figure 3.13: Extended tensors F and F_u

Using this description of states and inputs the monomial tensor of

$$x(k+1) = \langle\, \mathsf{F} \mid \mathsf{M}\left(x(k+1), u(k+1)\right) \rangle$$

is given by

$$\begin{aligned}
\mathsf{M}\left(x(k+1), u(k+1)\right) &= \begin{pmatrix} 1 \\ u(k+1) \end{pmatrix} \circ \begin{pmatrix} 1 \\ x(k+1) \end{pmatrix} \\
&= \begin{pmatrix} 1 \\ \langle\, \mathsf{F}_u \mid \mathsf{M}\left(x(k), u(k), u(k+1)\right) \rangle \end{pmatrix} \circ \begin{pmatrix} 1 \\ \langle\, \mathsf{F} \mid \mathsf{M}\left(x(k), u(k), u(k+1)\right) \rangle \end{pmatrix} \\
&= \langle\, \mathsf{1}_3 \boxplus_4 \mathsf{F}_u \mid \mathsf{M}\left(x(k), u(k), u(k+1)\right) \rangle \circ \langle\, \mathsf{1}_3 \boxplus_4 \mathsf{F} \mid \mathsf{M}\left(x(k), u(k), u(k+1)\right) \rangle \\
&= \left\langle\, \bar{\mathsf{F}}^2_u \mid \mathsf{M}\left(x(k), u(k), u(k+1)\right) \right\rangle \circ \left\langle\, \bar{\mathsf{F}}^2_x \mid \mathsf{M}\left(x(k), u(k), u(k+1)\right) \right\rangle \\
&= \left\langle\, \mathsf{F}^2_M \;\middle|\; \mathsf{M}^2_p\left(x(k), u(k), u(k+1)\right) \right\rangle_{3:8,1:6},
\end{aligned}$$

where the parameter tensor reads

$$\mathsf{F}^2_M(i_u, i_x, :, :, :, :, :, :) = \bar{\mathsf{F}}^2_u(:, :, :, i_u) \circ \bar{\mathsf{F}}^2_x(:, :, :, i_x).$$

With this description of the monomial tensor, the monomials are constructed in terms of $x(k), u(k), u(k+1)$ *by*

$$\begin{aligned}
&\mathsf{M}\left(x(k+1), u(k+1)\right) = \left\langle\, \mathsf{F}^2_M \;\middle|\; \mathsf{M}^2_p\left(x(k), u(k), u(k+1)\right) \right\rangle_{3:8,1:6} \\
&= \begin{pmatrix} \left\langle \mathsf{F}^2_M(1,1,:,\cdots,:) \middle| \mathsf{M}^2_p\left(x(k), u(k), u(k+1)\right) \right\rangle & \left\langle \mathsf{F}^2_M(1,2,:,\cdots,:) \middle| \mathsf{M}^2_p\left(x(k), u(k), u(k+1)\right) \right\rangle \\ \left\langle \mathsf{F}^2_M(2,1,:,\cdots,:) \middle| \mathsf{M}^2_p\left(x(k), u(k), u(k+1)\right) \right\rangle & \left\langle \mathsf{F}^2_M(2,2,:,\cdots,:) \middle| \mathsf{M}^2_p\left(x(k), u(k), u(k+1)\right) \right\rangle \end{pmatrix} \\
&= \begin{pmatrix} 1 & \langle\, \mathsf{F} \,|\mathsf{M}\left(x(k), u(k), u(k+1)\right)\rangle \\ \langle\, \mathsf{F}_u \,|\mathsf{M}\left(x(k), u(k), u(k+1)\right)\rangle & \langle\, \mathsf{F} \,|\mathsf{M}\left(x(k), u(k), u(k+1)\right)\rangle \langle\, \mathsf{F}_u \,|\mathsf{M}\left(x(k), u(k), u(k+1)\right)\rangle \end{pmatrix} \\
&= \begin{pmatrix} 1 & x(k+1) \\ u(k+1) & x(k+1)u(k+1) \end{pmatrix} = \begin{pmatrix} 1 & -x(k) + x(k)u(k) \\ u(k+1) & (-x(k) + x(k)u(k))u(k+1) \end{pmatrix}.
\end{aligned}$$

With this representation the parameters and monomials are separated by

$$\begin{aligned} x(k+2) &= \left\langle \mathsf{F} \,\middle|\, \left\langle \mathsf{F}_M^2 \,\middle|\, \mathsf{M}_p^2\left(x(k), u(k), u(k+1)\right) \right\rangle_{3:8,1:6} \right\rangle \\ &= \left\langle \left\langle \mathsf{F}_M^2 \mid \mathsf{F} \right\rangle_{1:2} \,\middle|\, \mathsf{M}_p^2\left(x(k), u(k), u(k+1)\right) \right\rangle = \left\langle \mathsf{F}^2 \,\middle|\, \mathsf{M}_p^2\left(x(k), u(k), u(k+1)\right) \right\rangle \\ &= -x(k) + u(k)x(k) + (-x(k) + u(k)x(k))u(k+1), \end{aligned}$$

which results in the desired representation of $x(k+2)$.

This shows that the multi-step transition of a MTI system leads to a polynomial function in tensor representation, whose parameter tensor follows from the parameter tensor F of the MTI system. But with the given tensor representation of polynomials as introduced in Section 2.3.1, the memory demand of the parameter tensor increases fast even though they are represented in decomposed way, which is possible here since all operations used are available, e.g. for CP tensors. Because of the multilinearity of the state equations, multiplications of e.g. states could occur, which leads to a polynomial in the multi-step transitions as illustrated by the Example 3.18. The multilinear state functions are applied iteratively during simulation resulting in a maximal possible order of the monomials that increases with the number of states and inputs and with the time horizon by $(n+m)^{i-1}$ for $\mathbf{x}(k+i)$. The number of variables increases with $n+im$, since the initial state and an input vector for each time step has to be provided. Thus, the monomial vector of (3.89) has $(n+im)(n+m)^{i-1}$ dimensions. Because of that, the memory demand of the rank-1 monomial tensor is already high with $2(n+im)(n+m)^{i-1}$ elements. This could lead to problems in the representation of the multi-step transition, when large-scale systems and a large number of time steps are investigated.

To get rid of this large storage effort, approximate solutions can be used, e.g. by applying the Taylor approximation as introduced in Subsection 2.3.5. A quadratic Taylor approximation in each step limits the maximal possible order of the polynomials to 2. Thus the memory demand is reduced and it is possible to represent also larger time horizons. The drawback is, that it does not give the exact solution as (3.89). The application of the Taylor approach results in an approximation, where the sufficient accuracy depends on the application.

3.7 Distributed systems with MTI subsystems

In many applications plants are composed of several subsystems, like a heating system contains e.g. of several boilers, consumers or storage tanks. Even though it was shown that also very large systems can be represented in the MTI framework by using tensor decomposition techniques, it is sometimes easier to divide the overall plant model into several parts and model and identify the parameters of the smaller subsystems first. This distributes a large modeling task to several smaller ones, which are easier to solve. After modeling and identification, all the subsystems are connected to the overall model. Modelling a plant by subsystems also allows a modular construction. This supports on the one hand the reusability of submodels for other plants. It is also easier to maintain the model. If a certain subsystem of the overall plant changes, not the whole model has to be investigated but only

the affected submodel has to be changed. This can be done with less effort and prevents also errors, because it is guaranteed that all other subsystems remain unchanged. The structure of the plant often gives already an idea, how the plant can be divided into subsystems. Another approach, how to find a structure when focusing on control will be shown in Section 4.3. A representation of distributed systems is introduced here, if the subsystems are modeled as MTI systems. In [85] connections of MTI subsystems were already investigated with the outcome that the resulting system is not necessarily multilinear. Serial, parallel and feedback connections were considered and conditions for the subsystems given, such that the connection of the system still remains in the multilinear class. In the following subsections the class of affine MTI systems is introduced that fulfills these conditions and it is shown how the parameter tensor of the resulting system can be computed from the parameter tensors of the subsystems, if they are connected in series, parallel or by feedback.

3.7.1 MTI subsystem representation

By construction in many applications plants can be divided into several subsystems, like e.g. boilers or consumers in heating systems, that can be modeled independently in the first step and connected afterwards. Here plants are focused on, where the particular subsystems are modeled by an MTI system. The description of the subsystem and its connections is adapted to the distributed system description of linear systems in [70]. In the distributed representation by MTI subsystems, the plant is described by a collection of N_{sub} MTI models

$$\Phi\left(\mathbf{x}^{(i)}\right) = \left\langle \mathsf{F}^{(i)} \;\middle|\; \mathsf{M}\left(\mathbf{x}^{(i)}, \mathbf{u}^{(i)}, \mathbf{s}^{(i)}\right) \right\rangle, \tag{3.91}$$

$$\mathbf{y}^{(i)} = \left\langle \mathsf{G}^{(i)} \;\middle|\; \mathsf{M}\left(\mathbf{x}^{(i)}, \mathbf{u}^{(i)}, \mathbf{s}^{(i)}\right) \right\rangle, \tag{3.92}$$

with $i = 1, \ldots, N_{sub}$. The superscript (i) indicates, that e.g. the state $\mathbf{x}^{(i)}$ belongs to the i^{th} subsystem. The same holds for the other signals $\mathbf{u}^{(i)}$ and $\mathbf{s}^{(i)}$ as well as for the parameter tensors $\mathsf{F}^{(i)}$ and $\mathsf{G}^{(i)}$. The effects of other subsystems on the i^{th} one are handled as a disturbance coming from inside the overall plant and described by the signals $\mathbf{s}^{(i)} \in \mathbb{R}^{m_{s,i}}$, where $m_{s,i}$ is the number of internal disturbances from other subsystems on subsystem i. The connections between the subsystems, i.e. the description of the internal disturbances $\mathbf{s}^{(i)}$, are given by the coupling function

$$\begin{pmatrix} \mathbf{s}^{(1)} \\ \vdots \\ \mathbf{s}^{(N_{sub})} \end{pmatrix} = \mathbf{h}(\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(N_{sub})}, \mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(N_{sub})}), \tag{3.93}$$

that depends on the inputs and outputs, of all subsystems and is of dimension $\sum_{i=1}^{N_{sub}} m_{s,i}$. These signals can influence the other subsystems. Thus, the whole plant is described by the collection of N_{sub} MTI systems (3.91) and (3.92), describing the dynamics of the subsystems, and the static coupling function (3.93).

Example 3.20 *A plant is given that consists of $N_{sub} = 3$ subsystems, that are connected as shown in Figure 3.14.*

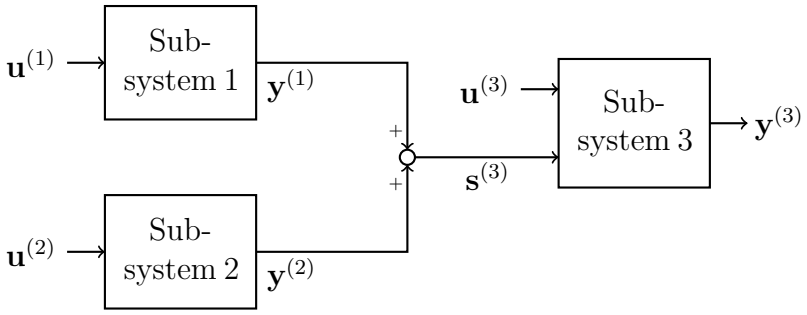


Figure 3.14: Plant decomposed into three subsystems

It is assumed that the dynamics of all three subsystems can be described by MTI models. The first and the second subsystem both have an external input but are not influenced by other subsystems. Thus, the number of internal disturbances is equal to zero for both systems $m_{s,1} = m_{s,2} = 0$ *and* $\mathbf{s}^{(1)}$ *and* $\mathbf{s}^{(2)}$ *can be neglected in their system descriptions. This results in the MTI systems*

$$\begin{aligned} \Phi\left(\mathbf{x}^{(1)}\right) &= \left\langle \mathsf{F}^{(1)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{u}^{(1)}\right) \right\rangle, \\ \mathbf{y}^{(1)} &= \left\langle \mathsf{G}^{(1)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{u}^{(1)}\right) \right\rangle, \end{aligned}$$

for subsystem 1 *and*

$$\begin{aligned} \Phi\left(\mathbf{x}^{(2)}\right) &= \left\langle \mathsf{F}^{(2)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(2)}, \mathbf{u}^{(2)}\right) \right\rangle, \\ \mathbf{y}^{(2)} &= \left\langle \mathsf{G}^{(2)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(2)}, \mathbf{u}^{(2)}\right) \right\rangle, \end{aligned}$$

for subsystem 2*. The first two subsystems act independently, but they both have an effect on the third subsystem, such that the input* $\mathbf{s}^{(3)}$ *has to be considered here for the MTI description of the third subsystem*

$$\begin{aligned} \Phi\left(\mathbf{x}^{(3)}\right) &= \left\langle \mathsf{F}^{(3)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(3)}, \mathbf{u}^{(3)}, \mathbf{s}^{(3)}\right) \right\rangle, \\ \mathbf{y}^{(3)} &= \left\langle \mathsf{G}^{(3)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(3)}, \mathbf{u}^{(3)}, \mathbf{s}^{(3)}\right) \right\rangle. \end{aligned}$$

The systems are coupled by the sum of the outputs of the first two subsystems that influences the third subsystem. This is given by the coupling equation

$$\mathbf{s}^{(3)} = \mathbf{h}(\mathbf{y}^{(1)}, \mathbf{y}^{(2)}) = \mathbf{y}^{(1)} + \mathbf{y}^{(2)}.$$

With that the whole system is described. If now e.g. the second subsystem changes, the parameter tensor $\mathsf{F}^{(2)}$ *has to be adapted only, which makes maintenance much easier and less sensitive to errors compared to adaption of the full system description. This has an even larger effect for a plant with more subsystems.*

3.7.2 Affine MTI systems

In the previous section plants that are composed of several MTI subsystems were introduced. In the next sections the serial, parallel and feedback connections of two subsystems are

investigated. The focus here is on systems, that stay in the class of multilinear systems, i.e. that fulfill the conditions stated in [85]. Because of that, a special class of MTI systems, i.e. affine MTI systems, will be considered here. State space models of affine systems are of the following type

$$\Phi(\mathbf{x}) = \mathbf{a}(\mathbf{x}) + \sum_{i=1}^{m} \mathbf{b}_i(\mathbf{x}) u_i, \tag{3.94}$$

$$\mathbf{y} = \mathbf{c}(\mathbf{x}), \tag{3.95}$$

with nonlinear functions $\mathbf{a}(\mathbf{x}), \mathbf{b}_i(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^n,\ i = 1, \ldots, m$ and the nonlinear output function $\mathbf{c}(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^p$, [39]. From [85] follows that the series, parallel and feedback connections remain in the multilinear class, if the subsystems, that are connected, belong to the class of affine MTI systems.

Definition 3.1 (Affine MTI system) *Affine MTI systems with n states, m inputs and p outputs can be written in state space format within the tensor framework as*

$$\begin{aligned} \Phi(\mathbf{x}) = \mathbf{a}(\mathbf{x}) + \sum_{i=1}^{m} \mathbf{b}_i(\mathbf{x}) u_i &= \langle\, \mathsf{F} \mid \mathsf{M}(\mathbf{x}, \mathbf{u}) \,\rangle \\ &= \langle\, \mathsf{A} \mid \mathsf{M}(\mathbf{x}) \,\rangle + \sum_{i=1}^{m} \langle\, \mathsf{B}_i \mid \mathsf{M}(\mathbf{x}) \,\rangle\, u_i, \end{aligned} \tag{3.96}$$

$$\begin{aligned} \mathbf{y} = \mathbf{c}(\mathbf{x}) &= \langle\, \mathsf{G} \mid \mathsf{M}(\mathbf{x}, \mathbf{u}) \,\rangle \\ &= \langle\, \mathsf{C} \mid \mathsf{M}(\mathbf{x}) \,\rangle, \end{aligned} \tag{3.97}$$

with parameter tensors $\mathsf{A} \in \mathbb{R}^{\times^{(n)}2\times n}$, $\mathsf{B}_i \in \mathbb{R}^{\times^{(n)}2\times n}$ *of the state equation and the parameter tensor* $\mathsf{C} \in \mathbb{R}^{\times^{(n)}2\times p}$ *of the output equation. It is assumed, that the system has no direct feedthrough, i.e. the current outputs do not depend on the current input variables but only on the states.*

The difference to general MTI systems is, that in the class of affine MTI systems no multiplications between inputs are allowed. Multiplicative combinations between all states and all states and one input are still possible, such that this class is a subsystem of general MTI systems but still extends the class of bilinear systems. The parameter tensors A, B_i and C of the affine MTI system can be extracted as subtensors from the parameter tensors F and G of general MTI systems. They are selected by fixing the first m indices belonging to the inputs of the system by

$$\mathsf{F}(i_1, \ldots, i_m, :, \ldots, :) = \begin{cases} \mathsf{A} & \text{, for } i_k = 1,\ k = 1, \ldots, m, \\ \mathsf{B}_j & \text{, for } i_j = 2, i_k = 1,\ k = 1, \ldots, m,\ k \neq j, \\ \mathsf{0}_A & \text{, otherwise,} \end{cases} \tag{3.98}$$

$$\mathsf{G}(i_1, \ldots, i_m, :, \ldots, :) = \begin{cases} \mathsf{C} & \text{, for } i_k = 1,\ k = 1, \ldots, m, \\ \mathsf{0}_C & \text{, otherwise,} \end{cases} \tag{3.99}$$

where $\mathsf{0}_A$ and $\mathsf{0}_C$ are tensors of zeros of same dimensions as $\mathsf{A} \in \mathbb{R}^{\times^{(n)}2}$ and $\mathsf{C} \in \mathbb{R}^{\times^{(n)}2}$, respectively. All other parameters belonging to multiplicative combinations of inputs have to be equal to zero.

Example 3.21 *An affine MTI system with one state x and two inputs u_1 and u_2 is given generally by*

$$\Phi(x) = a(x) + b_1(x)u_1 + b_2(x)u_2 = \alpha(1) + \alpha(2)x + (\beta_1(1) + \beta_1(2)x)\,u_1 + (\beta_2(1) + \beta_2(2)x)\,u_2,$$
$$y = c(x) = \gamma(1) + \gamma(2)x.$$

The state space model can be written as general MTI system with state equation

$$\Phi(x) = \langle\, \mathsf{F} \mid \mathsf{M}(x, u_1, u_2) \,\rangle$$
$$= \left\langle \begin{array}{ccc} & \beta_2(1) & \beta_2(2) \\ \alpha(1) & & \alpha(2) \\ & 0 & 0 \\ \beta_1(1) & & \beta_1(2) \end{array} \middle| \begin{array}{ccc} & u_2 & xu_2 \\ 1 & & x \\ & u_1u_2 & xu_1u_2 \\ u_1 & & xu_1 \end{array} \right\rangle,$$

and output equation

$$y = \langle\, \mathsf{G} \mid \mathsf{M}(x, u_1, u_2) \,\rangle$$
$$= \left\langle \begin{array}{ccc} & 0 & 0 \\ \gamma(1) & & \gamma(2) \\ & 0 & 0 \\ 0 & & 0 \end{array} \middle| \begin{array}{ccc} & u_2 & xu_2 \\ 1 & & x \\ & u_1u_2 & xu_1u_2 \\ u_1 & & xu_1 \end{array} \right\rangle.$$

In the example model no multiplications of the inputs u_1 and u_2 occur. Because of that, the model belongs to the class of affine MTI systems. This gets also clear when looking at the parameter tensor F of the system. The parameters belonging to the monomials u_1u_2 and xu_1u_2 are equal to zero. The parameter tensors of the affine MTI system representation can be extracted from the parameter tensors F and G by (3.98) and (3.99) or directly from the differential equations leading to the affine MTI system

$$\begin{aligned} \Phi(x) &= \langle\, \mathsf{A} \mid \mathsf{M}(x) \,\rangle + \langle\, \mathsf{B}_1 \mid \mathsf{M}(x) \,\rangle\, u_1 + \langle\, \mathsf{B}_2 \mid \mathsf{M}(x) \,\rangle\, u_2 \\ &= \left\langle \begin{pmatrix} \alpha(1) \\ \alpha(2) \end{pmatrix} \middle| \begin{pmatrix} 1 \\ x \end{pmatrix} \right\rangle + \left\langle \begin{pmatrix} \beta_1(1) \\ \beta_1(2) \end{pmatrix} \middle| \begin{pmatrix} 1 \\ x \end{pmatrix} \right\rangle u_1 + \left\langle \begin{pmatrix} \beta_2(1) \\ \beta_2(2) \end{pmatrix} \middle| \begin{pmatrix} 1 \\ x \end{pmatrix} \right\rangle u_2, \\ y = \langle\, \mathsf{C} \mid \mathsf{M}(x) \,\rangle &= \left\langle \begin{pmatrix} \gamma(1) \\ \gamma(2) \end{pmatrix} \middle| \begin{pmatrix} 1 \\ x \end{pmatrix} \right\rangle. \end{aligned}$$

Affine systems were defined for the general MIMO case with m inputs and p outputs. In the next sections connections of subsystems that are modeled by affine MTI systems are investigated. Thus, the description of the affine MTI systems have to be extended by internal

disturbance inputs $\mathbf{s}^{(i)}$ as shown in Section 3.7.1 for general MTI systems. This is done in the same way than for the other inputs, resulting in the MTI representation of the j^{th} subsystem

$$\begin{aligned}\Phi\left(\mathbf{x}^{(j)}\right) &= \mathbf{a}^{(j)}(\mathbf{x}^{(j)}) + \sum_{i=1}^{m_j} \mathbf{b}_i^{(j)}(\mathbf{x}^{(j)}) u_i^{(j)} + \sum_{i=1}^{m_{s,j}} \mathbf{b}_{s,i}^{(j)}(\mathbf{x}^{(j)}) s_i^{(j)} \qquad (3.100)\\ &= \left\langle \mathsf{A}^{(j)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(j)}\right) \right\rangle + \sum_{i=1}^{m_j} \left\langle \mathsf{B}_i^{(j)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(j)}\right) \right\rangle u_i^{(j)} + \sum_{i=1}^{m_{s,j}} \left\langle \mathsf{B}_{s,i}^{(j)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(j)}\right) \right\rangle s_i^{(j)},\\ \mathbf{y}^{(j)} &= \left\langle \mathsf{C}^{(j)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(j)}\right) \right\rangle. \qquad (3.101)\end{aligned}$$

3.7.3 Augmentation of affine MTI systems

In this and the following subsections serial, parallel and feedback connections of two affine MTI systems given by (3.100) and (3.101) are investigated, where $\mathbf{x}^{(j)},\ j = 1, 2$ are the state vectors of the two systems, respectively. The goal is to describe the connected system as an affine state space model

$$\Phi(\mathbf{x}) = \Phi\left(\begin{pmatrix}\mathbf{x}^{(1)}\\ \mathbf{x}^{(2)}\end{pmatrix}\right) = \left\langle \mathsf{A} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right) \right\rangle + \sum_{i=1}^{m} \left\langle \mathsf{B}_i \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right) \right\rangle u_i, \qquad (3.102)$$

$$\mathbf{y} = \left\langle \mathsf{C} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right) \right\rangle, \qquad (3.103)$$

where the state vector consists of the states of the two subsystems. The parameter tensors are composed of the parameter tensors of the subsystems. The resulting system has external inputs $u_i,\ i = 1, \ldots, m$ only. All internal connections $\mathbf{s}^{(j)}$ are resolved and included in the overall model structure. To merge the state equations according to the different types of connections at first the submodel functions have to be extended, such that their monomial tensors does not depend on $\mathbf{x}^{(1)}$ or $\mathbf{x}^{(2)}$ only but on both states, i.e. the parameter tensor has to be extended such that

$$\left\langle \mathsf{A}^{(j)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(j)}\right) \right\rangle = \left\langle \tilde{\mathsf{A}}^{(j)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right) \right\rangle, \qquad (3.104)$$

$$\left\langle \mathsf{B}_i^{(j)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(j)}\right) \right\rangle = \left\langle \tilde{\mathsf{B}}_i^{(j)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right) \right\rangle,\ i = 1, \ldots, m_j, \qquad (3.105)$$

$$\left\langle \mathsf{B}_{s,i}^{(j)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(j)}\right) \right\rangle = \left\langle \tilde{\mathsf{B}}_{s,i}^{(j)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right) \right\rangle,\ i = 1, \ldots, m_{s,j} \qquad (3.106)$$

$$\left\langle \mathsf{C}^{(j)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(j)}\right) \right\rangle = \left\langle \tilde{\mathsf{C}}^{(j)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right) \right\rangle, \qquad (3.107)$$

with $j = 1, 2$. At first this extension of the parameter tensor is described for scalar functions and afterwards generalized to vector functions like the system function used here. Therefore, consider a scalar multilinear function, that depends on n_1 variables

$$h(\mathbf{x}^{(1)}) = \left\langle \mathsf{H} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}\right) \right\rangle.$$

To add a dependence on variables $\mathbf{x}^{(2)} \in \mathbb{R}^{n_2}$ the function $h(\mathbf{x}^{(1)})$ is multiplied by 1, where the 1 is expressed in the tensor framework. With the multiplication approach of polynomials by operational tensors introduced in Section 2.3.2 the polynomial $h(\mathbf{x}^{(1)})$ is extended by

$$\begin{aligned}h(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) &= 1 \cdot \left\langle \mathsf{H} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}\right) \right\rangle = \left\langle 1_{n_2} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(2)}\right) \right\rangle \left\langle \mathsf{H} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}\right) \right\rangle\\ &= \left\langle 1_{n_2} \circ \mathsf{H} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(2)}\right) \circ \mathsf{M}\left(\mathbf{x}^{(1)}\right) \right\rangle = \left\langle 1_{n_2} \circ \mathsf{H} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right) \right\rangle.\end{aligned}$$

The tensor $\mathsf{1}_{n_2}$ has to be multiplied from the left to the function to get a monomial tensor that depends on both states like $\mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right)$, since for the monomial tensor holds

$$\mathsf{M}\left(\mathbf{x}^{(2)}\right) \circ \mathsf{M}\left(\mathbf{x}^{(1)}\right) = \left[\begin{pmatrix} 1 \\ x_{n_2}^{(2)} \end{pmatrix}, \ldots, \begin{pmatrix} 1 \\ x_1^{(2)} \end{pmatrix}, \begin{pmatrix} 1 \\ x_{n_1}^{(1)} \end{pmatrix}, \ldots, \begin{pmatrix} 1 \\ x_1^{(1)} \end{pmatrix}\right] = \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right).$$

Thus $\left\langle \mathsf{H} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}\right) \right\rangle$ is equal to $\left\langle \tilde{\mathsf{H}} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right) \right\rangle$ with the parameter tensor

$$\tilde{\mathsf{H}} = \mathsf{1}_{n_2} \circ \mathsf{H}.$$

If the function depends on $\mathbf{x}^{(2)}$ and the parameter tensor should be expressed in terms of a monomial tensor $\mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right)$ this is done by

$$\begin{aligned} \left\langle \mathsf{H} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(2)}\right) \right\rangle &= \left\langle \mathsf{H} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(2)}\right) \right\rangle \left\langle \mathsf{1}_{n_1} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}\right) \right\rangle = \left\langle \mathsf{H} \circ \mathsf{1}_{n_1} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right) \right\rangle \\ &= \left\langle \tilde{\mathsf{H}} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right) \right\rangle. \end{aligned} \tag{3.108}$$

This approach can be applied to compute the parameter tensors $\tilde{\mathsf{A}}^{(j)}$, $\tilde{\mathsf{B}}_i^{(j)}$, $\tilde{\mathsf{B}}_{s,i}^{(j)}$ and $\tilde{\mathsf{C}}^{(j)}$ in (3.104) to (3.107). The difference is, that in the case of affine systems, vector functions, not scalar functions are used. The extension is applied to each component of the vector function, e.g. to each component $a^{(1)}(i)(\mathbf{x}^{(1)})$ of the vector function

$$\mathbf{a}^{(1)}(\mathbf{x}^{(1)}) = \begin{pmatrix} a^{(1)}(1)(\mathbf{x}^{(1)}) & \cdots & a^{(n_1)}(i)(\mathbf{x}^{(1)}) \end{pmatrix}^T.$$

The parameter tensor of one component of the vector function is given by selecting a subtensor of $\mathbf{a}^{(1)}(\mathbf{x}^{(1)})$ by fixing its last dimension and is computed as

$$a^{(1)}(i)(\mathbf{x}^{(1)}) = \left\langle \mathsf{A}^{(1)}(:, \ldots, :, i) \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}\right) \right\rangle = \left\langle \mathsf{A}^{(1)} \bar{\times}_{n_1+1} \mathbf{e}_{n_1,i} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}\right) \right\rangle.$$

where $\mathbf{e}_{n_1,i}$ is the i^{th} unit vector of length n_1. Each component of the vector function is a scalar function and thus it can be extended as described before by

$$\left\langle \mathsf{A}^{(1)} \bar{\times}_{n_1+1} \mathbf{e}_{n_1,i} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}\right) \right\rangle = \left\langle \mathsf{1}_{n_2} \circ \left(\mathsf{A}^{(1)} \bar{\times}_{n_1+1} \mathbf{e}_{n_1,i}\right) \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right) \right\rangle,$$

with $i = 1, \ldots, n_1$. The extensions of each component are concatenated in the last dimension again, which gives a closed expression of the parameter tensor of the extended function

$$\tilde{\mathsf{A}}^{(1)} = \coprod_{j=1}^{n_1}{}_{n_1+n_2+1} \mathsf{1}_{n_2} \circ \left(\mathsf{A}^{(1)} \bar{\times}_{n_1+1} \mathbf{e}_{n_1,j}\right). \tag{3.109}$$

The dependence on $\mathbf{x}^{(2)}$ is added to the other functions of the first system in the same way by

$$\tilde{\mathsf{B}}_i^{(1)} = \coprod_{j=1}^{n_1}{}_{n_1+n_2+1} \mathsf{1}_{n_2} \circ \left(\mathsf{B}_i^{(1)} \bar{\times}_{n_1+1} \mathbf{e}_{n_1,j}\right), \; i = 1, \ldots, m_1, \tag{3.110}$$

$$\tilde{\mathsf{B}}_{s,i}^{(1)} = \coprod_{j=1}^{n_1}{}_{n_1+n_2+1} \mathsf{1}_{n_2} \circ \left(\mathsf{B}_{s,i}^{(1)} \bar{\times}_{n_1+1} \mathbf{e}_{n_1,j}\right) \; i = 1, \ldots, m_{s,1}, \tag{3.111}$$

$$\tilde{\mathsf{C}}^{(1)} = \coprod_{j=1}^{n_1}{}_{n_1+n_2+1} \mathsf{1}_{n_2} \circ \left(\mathsf{C}^{(1)} \bar{\times}_{n_1+1} \mathbf{e}_{n_1,j}\right). \tag{3.112}$$

The approach is also applied for the second system. Here the tensor (3.82) has to be multiplied from the right to the original parameter tensor as in (3.108) leading to

$$\tilde{\mathsf{A}}^{(2)} = \coprod_{j=1}^{n_2}{}_{n_1+n_2+1} \left(\mathsf{A}^{(2)} \bar{\times}_{n_2+1} \mathbf{e}_{n_2,j}\right) \circ \mathbf{1}_{n_1}, \tag{3.113}$$

$$\tilde{\mathsf{B}}_i^{(2)} = \coprod_{j=1}^{n_2}{}_{n_1+n_2+1} \left(\mathsf{B}_i^{(2)} \bar{\times}_{n_2+1} \mathbf{e}_{n_2,j}\right) \circ \mathbf{1}_{n_1}, \; i = 1, \ldots, m_2, \tag{3.114}$$

$$\tilde{\mathsf{B}}_{s,i}^{(2)} = \coprod_{j=1}^{n_2}{}_{n_1+n_2+1} \left(\mathsf{B}_{s,i}^{(2)} \bar{\times}_{n_2+1} \mathbf{e}_{n_2,j}\right) \circ \mathbf{1}_{n_1}, \; i = 1, \ldots, m_{s,2}, \tag{3.115}$$

$$\tilde{\mathsf{C}}^{(2)} = \coprod_{j=1}^{n_2}{}_{n_1+n_2+1} \left(\mathsf{C}^{(2)} \bar{\times}_{n_2+1} \mathbf{e}_{n_2,j}\right) \circ \mathbf{1}_{n_1}. \tag{3.116}$$

With these extensions the serial, parallel and feedback connection of affine MTI systems can be described.

3.7.4 Serial connection

Two affine MTI subsystems are connected in series as depicted in Figure 3.15.

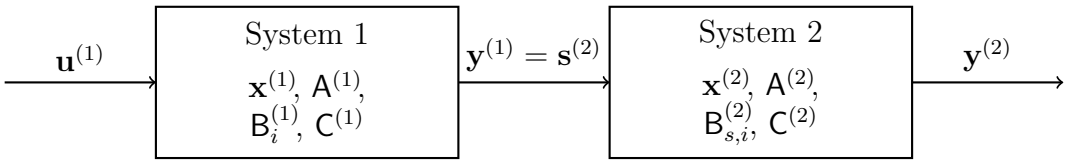


Figure 3.15: Serial connection

The first system has m_1 external inputs $\mathbf{u}^{(1)}$, that are also the inputs of the overall system

$$\mathbf{u} = \mathbf{u}^{(1)},$$

and is not influenced by other subsystems $m_{s,1} = 0$. Because of the serial connection

$$\mathbf{s}^{(2)} = \mathbf{y}^{(1)},$$

the second system has no external inputs ($m_2 = 0$), but the number p_1 of outputs of the first system has to be equal to the number $m_{s,2}$ of internal disturbances of the second system. The overall output is equal to the output of system 2

$$\mathbf{y} = \mathbf{y}^{(2)}.$$

For the derivation of the connected subsystem it is easier to consider an affine SISO MTI systems first. For the two subsystems, the state space model (3.100) and (3.101) simplifies to the state equations

$$\Phi\left(\mathbf{x}^{(1)}\right) = \left\langle \mathsf{A}^{(1)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}\right) \right\rangle + \left\langle \mathsf{B}_1^{(1)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}\right) \right\rangle u_1^{(1)}, \tag{3.117}$$

$$\Phi\left(\mathbf{x}^{(2)}\right) = \left\langle \mathsf{A}^{(2)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(2)}\right) \right\rangle + \left\langle \mathsf{B}_{s,1}^{(2)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(2)}\right) \right\rangle s_2^{(1)}, \tag{3.118}$$

and the output equations

$$\mathbf{y}^{(j)} = \left\langle \mathsf{C}^{(j)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(j)}\right) \right\rangle, \; j = 1, 2, \tag{3.119}$$

of the two subsystems, where the output functions are scalar with parameter tensors $\mathsf{C}^{(j)}$ of dimension $\mathbb{R}^{\times(n)2\times 1}$. Inserting the coupling function in the state equations of the two systems gives

$$\Phi\left(\mathbf{x}^{(1)}\right) = \left\langle \mathsf{A}^{(1)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}\right) \right\rangle + \left\langle \mathsf{B}_1^{(1)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}\right) \right\rangle u_1^{(1)}, \tag{3.120}$$

$$\Phi\left(\mathbf{x}^{(2)}\right) = \left\langle \mathsf{A}^{(2)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(2)}\right) \right\rangle + \left\langle \mathsf{B}_{s,1}^{(2)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(2)}\right) \right\rangle \left\langle \mathsf{C}^{(1)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}\right) \right\rangle, \tag{3.121}$$

$$\mathbf{y}^{(2)} = \left\langle \mathsf{C}^{(2)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(2)}\right) \right\rangle. \tag{3.122}$$

To get an overall description of the system (3.102) and (3.103) with states $\mathbf{x} = \begin{pmatrix}\mathbf{x}^{(1)} & \mathbf{x}^{(2)}\end{pmatrix}^T$ the extended versions (3.109), (3.113) and (3.110) of parameter tensors $\mathsf{A}^{(j)}$ and $\mathsf{B}_1^{(1)}$ have to be used. For the calculation of the state equation of the second subsystem the multiplication of a scalar function $\left\langle \mathsf{C}^{(1)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}\right) \right\rangle \in \mathbb{R}^1$ and a vector function $\left\langle \mathsf{B}_{s,1}^{(2)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(2)}\right) \right\rangle$ of dimension $\mathbb{R}^{n_2}$ is computed by

$$\left\langle \mathsf{B}_{s,1}^{(2)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(2)}\right) \right\rangle \left\langle \mathsf{C}^{(1)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}\right) \right\rangle = \left\langle \mathsf{D} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right) \right\rangle. \tag{3.123}$$

The components

$$b_{s,1}^{(2)}(i)(\mathbf{x}^{(2)}) = \left\langle \mathsf{B}_{s,1}^{(2)} \bar{\times}_{n_2+1} \mathbf{e}_i \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(2)}\right) \right\rangle,$$

with $i = 1, \ldots, n_2$ of

$$\mathbf{b}_{s,1}^{(2)}(\mathbf{x}^{(2)}) = \begin{pmatrix} b_{s,1}^{(2)}(1)(\mathbf{x}^{(2)}) & \cdots & b_{s,1}^{(2)}(n_2)(\mathbf{x}^{(2)}) \end{pmatrix}^T = \left\langle \mathsf{B}_{s,1}^{(2)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(2)}\right) \right\rangle$$

are multiplied by $\left\langle \mathsf{C}^{(1)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}\right) \right\rangle$ leading to

$$\left\langle \mathsf{B}_{s,1}^{(2)} \bar{\times}_{n_2+1} \mathbf{e}_i \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(2)}\right) \right\rangle \left\langle \mathsf{C}^{(1)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}\right) \right\rangle = \left\langle \left(\mathsf{B}_{s,1}^{(2)} \bar{\times}_{n_2+1} \mathbf{e}_i\right) \circ \mathsf{C}^{(1)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right) \right\rangle.$$

To get the parameter tensor of (3.123) the results of the elementwise multiplications are concatenated in the last dimension

$$\mathsf{D} = \coprod_{j=1}^{n_2}{}_{n_1+n_2+1} \left(\mathsf{B}_{s,1}^{(2)} \bar{\times}_{n_2+1} \mathbf{e}_j\right) \circ \mathsf{C}^{(1)}. \tag{3.124}$$

With that, the second state equation (3.121) is written as

$$\begin{aligned} \Phi\left(\mathbf{x}^{(2)}\right) &= \left\langle \tilde{\mathsf{A}}^{(2)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right) \right\rangle + \left\langle \mathsf{D} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right) \right\rangle \\ &= \left\langle \tilde{\mathsf{A}}^{(2)} + \mathsf{D} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right) \right\rangle. \end{aligned}$$

The parameter tensor A of the resulting system (3.102) and (3.103) is computed by concatenating the extended tensors $\tilde{\mathsf{A}}^{(1)}$ and $\tilde{\mathsf{A}}^{(2)} + \mathsf{D}$ in the last dimension

$$\mathsf{A} = \tilde{\mathsf{A}}^{(1)} \boxplus_{n_1+n_2+1} \left(\tilde{\mathsf{A}}^{(2)} + \mathsf{D} \right).$$

With the definition of the concatenation operation this gives the desired result

$$\begin{aligned} \mathsf{A}(:, \ldots, :, 1 : n_1) &= \tilde{\mathsf{A}}^{(1)}, \\ \mathsf{A}(:, \ldots, :, n_1 + 1 : n_1 + n_2) &= \tilde{\mathsf{A}}^{(2)} + \mathsf{D}. \end{aligned}$$

The state equations (3.120) and (3.121) show, that the states of the first system are influenced by the input of the overall system only. The second subsystem is not influenced from the outside. Because of that zeros have be added to the extended parameter tensor of the first system

$$\mathsf{B}_1 = \tilde{\mathsf{B}}_1^{(1)} \boxplus_{n_1+n_2+1} \mathsf{0}_{\tilde{\mathsf{B}}_{s,1}^{(2)}} \in \mathbb{R}^{\times^{(n_1+n_2)} 2 \times n_1 + n_2},$$

with a tensor full of zeros $\mathsf{0}_{\tilde{\mathsf{B}}_{s,1}^{(2)}}$, that has the same dimension $\mathbb{R}^{\times^{(n_1+n_2)} 2 \times n_2}$ than $\tilde{\mathsf{B}}_{s,1}^{(2)}$, to get the parameter tensor of the overall system. The output of the connected system is equal to the output of the second subsystem, such that the parameter tensor of the output equation is given by

$$\mathsf{C} = \tilde{\mathsf{C}}^{(2)}.$$

This defines the parameter tensors of the connected overall affine MTI system for the SISO case. The concept is adapted to systems with multiple inputs and outputs as described in the following.

The serial connection of two affine MIMO MTI systems results in the state space model

$$\Phi\left(\mathbf{x}^{(1)}\right) = \left\langle \mathsf{A}^{(1)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}\right) \right\rangle + \sum_{i=1}^{m_1} \left\langle \mathsf{B}_i^{(1)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}\right) \right\rangle u_i^{(1)}, \tag{3.125}$$

$$\Phi\left(\mathbf{x}^{(2)}\right) = \left\langle \mathsf{A}^{(2)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(2)}\right) \right\rangle + \sum_{i=1}^{m_{s,2}} \left\langle \mathsf{B}_{s,i}^{(2)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(2)}\right) \right\rangle \left\langle \mathsf{C}^{(1)} \bar{\times}_{p_1+1} \mathbf{e}_{p_1,i} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}\right) \right\rangle, \tag{3.126}$$

$$\mathbf{y}^{(2)} = \left\langle \mathsf{C}^{(2)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(2)}\right) \right\rangle, \tag{3.127}$$

where $\mathsf{C}^{(1)} \bar{\times}_{p_1+1} \mathbf{e}_{p_1,i}$ describes the parameter tensor of the i^{th} component

$$y_i^{(1)} = \left\langle \mathsf{C}^{(1)}(:, \ldots, :, i) \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}\right) \right\rangle, i = 1, \ldots, p_1,$$

of the output equation of system 1. The multiplication

$$\left\langle \mathsf{B}_{s,i}^{(2)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(2)}\right) \right\rangle \left\langle \mathsf{C}^{(1)} \bar{\times}_{p_1+1} \mathbf{e}_{p_1,i} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}\right) \right\rangle = \left\langle \mathsf{D}_i \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right) \right\rangle$$

is computed analogous to (3.124) with parameter tensor

$$\mathsf{D}_i = \boxplus_{j=1}^{n_2}{}_{n_1+n_2+1} \left(\left(\mathsf{B}_i^{(2)} \bar{\times}_{n_2+1} \mathbf{e}_{n_2,j} \right) \circ \left(\mathsf{C}^{(1)} \bar{\times}_{p_1+1} \mathbf{e}_{p_1,i} \right) \right),$$

for $i = 1, \ldots, m_{s,2}$, such that the second state equation is written as

$$\begin{aligned}\Phi\left(\mathbf{x}^{(2)}\right) &= \left\langle \tilde{\mathsf{A}}^{(2)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right) \right\rangle + \sum_{i=1}^{m_{s,2}} \left\langle \mathsf{D}_i \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right) \right\rangle \\ &= \left\langle \tilde{\mathsf{A}}^{(2)} + \sum_{i=1}^{m_2} \mathsf{D}_i \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right) \right\rangle.\end{aligned}$$

As in the SISO case the parameter tensor A of the overall system follows from the concatenation

$$\mathsf{A} = \tilde{\mathsf{A}}^{(1)} \boxplus_{n_1+n_2+1} \left(\tilde{\mathsf{A}}^{(2)} + \sum_{i=1}^{m_{s,2}} \mathsf{D}_i\right), \tag{3.128}$$

in the last dimension. The tensors $\mathsf{B}_i^{(1)}$ are extended, such that they depend on both state variables and zeros are added, because the inputs do not influence the second state

$$\mathsf{B}_i = \tilde{\mathsf{B}}_i^{(1)} \boxplus_{n_1+n_2+1} \mathsf{0}_{\tilde{\mathsf{B}}_{s,i}^{(2)}}, \; i = 1, \ldots, m_1. \tag{3.129}$$

The output of the overall system is the output of the second system and thus the parameter tensor of the output equation is

$$\mathsf{C} = \tilde{\mathsf{C}}^{(2)}. \tag{3.130}$$

By (3.128) to (3.130) all parameter tensors are derived to describe the state space model (3.102) and (3.103) of the serial connection of two affine MTI subsystems, that are summarized in the following Proposition.

Proposition 3.3 (Serial connection of two affine MTI systems) *The serial connection of two affine MTI systems given in a tensor structure (3.100) and (3.101) results in an affine MTI system (3.102) and (3.103), where the parameter tensors of the connected system follow from the parameter tensors of the subsystems by*

$$\begin{aligned}\mathsf{A} =& \tilde{\mathsf{A}}^{(1)} \boxplus_{n_1+n_2+1} \left(\tilde{\mathsf{A}}^{(2)} + \sum_{i=1}^{m_{s,2}} \boxplus\!\!\!\!\!\!\boxplus_{j=1}^{n_2}{}_{n_1+n_2+1} \left(\left(\mathsf{B}_i^{(2)} \bar{\times}_{n_2+1} \mathbf{e}_{n_2,j}\right) \circ \left(\mathsf{C}^{(1)} \bar{\times}_{p_1+1} \mathbf{e}_{p_1,i}\right)\right)\right) \\ \mathsf{B}_i =& \tilde{\mathsf{B}}_i^{(1)} \boxplus_{n_1+n_2+1} \mathsf{0}_{\tilde{\mathsf{B}}_{s,i}^{(2)}}, \; i = 1, \ldots, m_1, \\ \mathsf{C} =& \tilde{\mathsf{C}}^{(2)}.\end{aligned}$$

Example 3.22 *The approach of connecting two MTI systems in series should be illustrated*

by an example of two affine SISO MTI systems of first order with state equations

$$\begin{aligned}
\Phi\left(x^{(1)}\right) &= \left\langle \mathsf{A}^{(1)} \,\middle|\, \mathsf{M}\left(x^{(1)}\right) \right\rangle + \left\langle \mathsf{B}_1^{(1)} \,\middle|\, \mathsf{M}\left(x^{(1)}\right) \right\rangle u_1^{(1)}, \\
&= \left\langle \begin{pmatrix} a^{(1)}(1) \\ a^{(1)}(2) \end{pmatrix} \middle| \begin{pmatrix} 1 \\ x^{(1)} \end{pmatrix} \right\rangle + \left\langle \begin{pmatrix} b^{(1)}(1) \\ b^{(1)}(2) \end{pmatrix} \middle| \begin{pmatrix} 1 \\ x^{(1)} \end{pmatrix} \right\rangle u_1^{(1)} \\
&= a^{(1)}(1) + a^{(1)}(2)x^{(1)} + b^{(1)}(1)u_1^{(1)} + b^{(1)}(2)x^{(1)}u_1^{(1)}, \qquad (3.131) \\
\Phi\left(x^{(2)}\right) &= \left\langle \mathsf{A}^{(2)} \,\middle|\, \mathsf{M}\left(x^{(2)}\right) \right\rangle + \left\langle \mathsf{B}_{s,1}^{(2)} \,\middle|\, \mathsf{M}\left(x^{(2)}\right) \right\rangle s_2^{(1)}, \\
&= \left\langle \begin{pmatrix} a^{(2)}(1) \\ a^{(2)}(2) \end{pmatrix} \middle| \begin{pmatrix} 1 \\ x^{(2)} \end{pmatrix} \right\rangle + \left\langle \begin{pmatrix} b_s^{(2)}(1) \\ b_s^{(2)}(2) \end{pmatrix} \middle| \begin{pmatrix} 1 \\ x^{(2)} \end{pmatrix} \right\rangle s_2^{(1)} \\
&= a^{(2)}(1) + a^{(2)}(2)x^{(2)} + b_s^{(2)}(1)s_2^{(1)} + b_s^{(2)}(2)x^{(2)}s_2^{(1)}, \qquad (3.132)
\end{aligned}$$

and output equations

$$y^{(j)} = \left\langle \mathsf{C}^{(j)} \,\middle|\, \mathsf{M}\left(x^{(j)}\right) \right\rangle = \left\langle \begin{pmatrix} c^{(j)}(1) \\ c^{(j)}(2) \end{pmatrix} \middle| \begin{pmatrix} 1 \\ x^{(j)} \end{pmatrix} \right\rangle = c^{(j)}(1) + c^{(j)}(2)x^{(j)}, \; j = 1, 2. \qquad (3.133)$$

As first step the parameter tensors $\mathsf{A}^{(1)}$, $\mathsf{B}_1^{(1)}$ *are extended to add the dependence on* $x^{(2)}$ *by (3.109) and (3.110)*

$$\begin{aligned}
&\Phi\left(x^{(1)}\right) = \left\langle \tilde{\mathsf{A}}^{(1)} \,\middle|\, \mathsf{M}\left(x^{(1)}, x^{(2)}\right) \right\rangle + \left\langle \tilde{\mathsf{B}}_1^{(1)} \,\middle|\, \mathsf{M}\left(x^{(1)}, x^{(2)}\right) \right\rangle u_1^{(1)}, \\
&= \left\langle \begin{pmatrix} a^{(1)}(1) & a^{(1)}(2) \\ 0 & 0 \end{pmatrix} \middle| \begin{pmatrix} 1 & x^{(1)} \\ x^{(2)} & x^{(1)}x^{(2)} \end{pmatrix} \right\rangle + \left\langle \begin{pmatrix} b^{(1)}(1) & b^{(1)}(2) \\ 0 & 0 \end{pmatrix} \middle| \begin{pmatrix} 1 & x^{(1)} \\ x^{(2)} & x^{(1)}x^{(2)} \end{pmatrix} \right\rangle u_1^{(1)}.
\end{aligned}$$

Due to the serial connection the output of the first system is used as input to the second system. With the extension of $\mathsf{A}^{(2)}$ *by (3.113) and the multiplication (3.124) by operational tensors, the second state equation reads*

$$\begin{aligned}
&\Phi\left(x^{(2)}\right) = \left\langle \tilde{\mathsf{A}}^{(2)} \,\middle|\, \mathsf{M}\left(x^{(1)}, x^{(2)}\right) \right\rangle + \left\langle \mathsf{B}_{s,1}^{(2)} \,\middle|\, \mathsf{M}\left(x^{(1)}\right) \right\rangle \left\langle \mathsf{C}^{(1)} \,\middle|\, \mathsf{M}\left(x^{(2)}\right) \right\rangle \\
&= \left\langle \begin{pmatrix} a^{(2)}(1) & 0 \\ a^{(2)}(2) & 0 \end{pmatrix} \middle| \begin{pmatrix} 1 & x^{(1)} \\ x^{(2)} & x^{(1)}x^{(2)} \end{pmatrix} \right\rangle + \left\langle \begin{pmatrix} b_s^{(2)}(1) \\ b_s^{(2)}(2) \end{pmatrix} \middle| \begin{pmatrix} 1 \\ x^{(2)} \end{pmatrix} \right\rangle \left\langle \begin{pmatrix} c^{(1)}(1) \\ c^{(1)}(2) \end{pmatrix} \middle| \begin{pmatrix} 1 \\ x^{(1)} \end{pmatrix} \right\rangle \\
&= \left\langle \begin{pmatrix} a^{(2)}(1) & 0 \\ a^{(2)}(2) & 0 \end{pmatrix} + \begin{pmatrix} b_s^{(2)}(1)c^{(1)}(1) & b_s^{(2)}(1)c^{(1)}(2) \\ b_s^{(2)}(2)c^{(1)}(1) & b_s^{(2)}(2)c^{(1)}(2) \end{pmatrix} \middle| \begin{pmatrix} 1 & x^{(1)} \\ x^{(2)} & x^{(1)}x^{(2)} \end{pmatrix} \right\rangle. \qquad (3.134)
\end{aligned}$$

The state equations are merged by concatenating the parameter tensors (3.128) and (3.129). This gives the overall state equation of the affine MTI systems of the serial connection by

$$\Phi\left(\begin{pmatrix} x^{(1)} \\ x^{(2)} \end{pmatrix}\right) = \left\langle \begin{array}{c} \begin{array}{ccc} & a^{(2)}(1)+ & \\ & b_s^{(2)}(1)c^{(1)}(1) & b_s^{(2)}(1)c^{(1)}(2) \\ a^{(1)}(1) & & a^{(1)}(2) \\ & a^{(2)}(2)+ & \\ & b_s^{(2)}(2)c^{(1)}(1) & b_s^{(2)}(2)c^{(1)}(2) \\ 0 & & 0 \end{array} \\ \Big| \\ \begin{array}{cc} 1 & x^{(1)} \\ x^{(2)} & x^{(1)}x^{(2)} \end{array} \end{array} \right\rangle$$

$$+ \left\langle \begin{array}{c} \begin{array}{ccc} & 0 & \\ b^{(1)}(1) & & b^{(1)}(2) \\ & 0 & \\ 0 & & 0 \end{array} \\ \Big| \\ \begin{array}{cc} 1 & x^{(1)} \\ x^{(2)} & x^{(1)}x^{(2)} \end{array} \end{array} \right\rangle u.$$

The example shows nicely that the first frontal slice of A *is equal to* $\tilde{\mathsf{A}}^{(1)}$ *and the second frontal slice is equal to the parameter tensor of the second state equation* $\tilde{\mathsf{A}}^{(2)} + \mathsf{D}$*. Considering the tensor* B_1 *the first frontal slice is* $\mathsf{B}_1^{(1)}$*, the second slice is full of zeros resulting from (3.129). The output of the connected system is the output of the second system resulting in the output equation with parameter tensor (3.130)*

$$y = \left\langle \begin{pmatrix} c^{(2)}(1) & 0 \\ c^{(2)}(2) & 0 \end{pmatrix} \middle| \begin{pmatrix} 1 & x^{(1)} \\ x^{(2)} & x^{(1)}x^{(2)} \end{pmatrix} \right\rangle.$$

3.7.5 Parallel connection

When connecting the two subsystems (3.100) and (3.101) in parallel, both systems are fed with the same external input. The output is the sum of the outputs of the subsystems

$$\mathbf{u} = \mathbf{u}^{(1)} = \mathbf{u}^{(2)}, \tag{3.135}$$

$$\mathbf{y} = \mathbf{y}^{(1)} + \mathbf{y}^{(2)}, \tag{3.136}$$

as shown in Figure 3.16. There are no internal connections between the systems, such that $m_{s,1} = m_{s,2} = 0$.

Here MIMO systems are considered directly. The SISO case is included and can be derived by setting the number of inputs and outputs of the systems to 1. In the general MIMO case investigated here, it has to be ensured for the parallel connection, that the number of inputs and the number of outputs are equal, i.e. $m_1 = m_2$ and $p_1 = p_2$. Inserting the

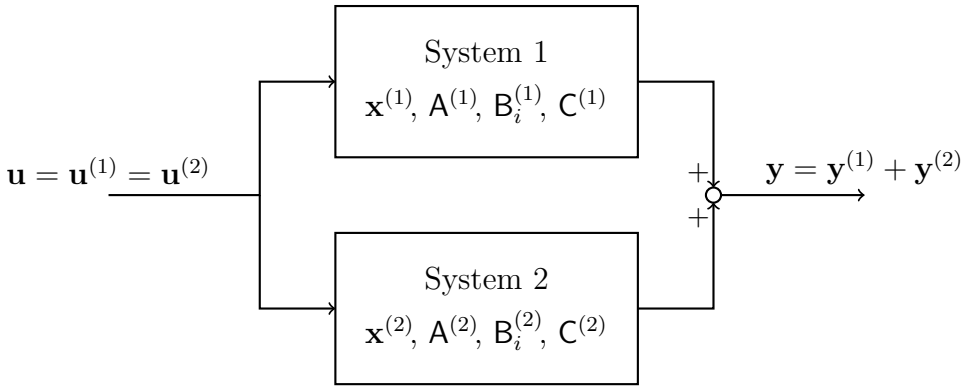


Figure 3.16: Parallel connection

constraints (3.135) and (3.136) in the state space models of both systems (3.100) and (3.101) yields

$$\Phi\left(\mathbf{x}^{(1)}\right) = \left\langle \mathsf{A}^{(1)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}\right) \right\rangle + \sum_{i=1}^{m_1} \left\langle \mathsf{B}_i^{(1)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}\right) \right\rangle u_i^{(1)}, \tag{3.137}$$

$$\Phi\left(\mathbf{x}^{(2)}\right) = \left\langle \mathsf{A}^{(2)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(2)}\right) \right\rangle + \sum_{i=1}^{m_2} \left\langle \mathsf{B}_i^{(2)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(2)}\right) \right\rangle u_i^{(2)}, \tag{3.138}$$

$$\mathbf{y} = \left\langle \mathsf{C}^{(1)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}\right) \right\rangle + \left\langle \mathsf{C}^{(2)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(2)}\right) \right\rangle. \tag{3.139}$$

To determine the parameter tensors of the overall state space model (3.102) and (3.103) the extended parameter tensors (3.104) to (3.107) depending on both state vectors are used. The state equations (3.137) and (3.138) are simply connected by concatenating their parameter tensors

$$\mathsf{A} = \tilde{\mathsf{A}}^{(1)} \boxplus_{n_1+n_2+1} \tilde{\mathsf{A}}^{(2)}, \tag{3.140}$$

$$\mathsf{B}_i = \tilde{\mathsf{B}}_i^{(1)} \boxplus_{n_1+n_2+1} \tilde{\mathsf{B}}_i^{(2)}, \; i = 1, \ldots, m_1. \tag{3.141}$$

The output equation can be simplified, when inserting the parameter tensors depending on $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ leading to

$$\begin{aligned} \mathbf{y} &= \left\langle \tilde{\mathsf{C}}^{(1)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right) \right\rangle + \left\langle \tilde{\mathsf{C}}^{(2)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right) \right\rangle \\ &= \left\langle \tilde{\mathsf{C}}^{(1)} + \tilde{\mathsf{C}}^{(2)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right) \right\rangle, \end{aligned}$$

which gives the parameter tensor of the output equation of the connected system as

$$\mathsf{C} = \tilde{\mathsf{C}}^{(1)} + \tilde{\mathsf{C}}^{(2)}. \tag{3.142}$$

Thus, the resulting parameter tensors can be summarized in the following proposition.

Proposition 3.4 (Parallel connection of affine MTI systems) *The parallel connection of two affine MTI systems given in a tensor structure (3.100) and (3.101) is still an affine*

MTI system (3.102) and (3.103), where the parameter tensors are given by

$$\begin{aligned}\mathsf{A} =& \tilde{\mathsf{A}}^{(1)} \boxplus_{n_1+n_2+1} \tilde{\mathsf{A}}^{(2)},\\ \mathsf{B}_i =& \tilde{\mathsf{B}}_i^{(1)} \boxplus_{n_1+n_2+1} \tilde{\mathsf{B}}_i^{(2)},\ i = 1, \ldots, m_1,\\ \mathsf{C} =& \tilde{\mathsf{C}}^{(1)} + \tilde{\mathsf{C}}^{(2)}.\end{aligned}$$

Example 3.23 *Two first order affine MTI systems (3.131) and (3.132) introduced in Example 3.22 are connected in parallel in this example. To derive the state space model of the connected system the tensors $\mathsf{A}^{(j)}$, $\mathsf{B}_1^{(j)}$ and $\mathsf{C}^{(j)}$ have to be extended according to (3.109) to (3.116), which is shown here exemplarily for the tensors $\mathsf{A}^{(j)}$*

$$\tilde{\mathsf{A}}^{(1)} = \begin{pmatrix} a^{(1)}(1) & a^{(1)}(2) \\ 0 & 0 \end{pmatrix},\ \tilde{\mathsf{A}}^{(2)} = \begin{pmatrix} a^{(2)}(1) & 0 \\ a^{(2)}(2) & 0 \end{pmatrix}.$$

The tensors $\mathsf{B}_1^{(j)}$ and $\mathsf{C}^{(j)}$ are extended in the same way. To get the parallel connection the extended tensors of the state equations of the two systems are concatenated (3.140) and (3.141) leading to the overall state equation

$$\Phi\left(\begin{pmatrix} x^{(1)} \\ x^{(2)} \end{pmatrix}\right) = \left\langle \begin{array}{c|c} \begin{array}{ccc} & a^{(2)}(1) & \\ a^{(1)}(1) & & a^{(1)}(2) \\ & a^{(2)}(2) & \\ 0 & & 0 \end{array} & \begin{array}{ccc} 1 & & x^{(1)} \\ & & \\ x^{(2)} & & x^{(1)}x^{(2)} \end{array} \end{array} \right\rangle$$

$$+ \left\langle \begin{array}{c|c} \begin{array}{ccc} & b^{(2)}(1) & \\ b^{(1)}(1) & & b^{(1)}(2) \\ & b^{(2)}(2) & \\ 0 & & 0 \end{array} & \begin{array}{ccc} 1 & & x^{(1)} \\ & & \\ x^{(2)} & & x^{(1)}x^{(2)} \end{array} \end{array} \right\rangle u.$$

The outputs of the two subsystems are added (3.142), which results in the output equation

$$\begin{aligned} y &= \left\langle \tilde{\mathsf{C}}^{(1)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right) \right\rangle + \left\langle \tilde{\mathsf{C}}^{(2)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right) \right\rangle \\ &= \left\langle \begin{pmatrix} c^{(1)}(1) & c^{(1)}(2) \\ 0 & 0 \end{pmatrix} \middle| \begin{pmatrix} 1 & x^{(1)} \\ x^{(2)} & x^{(1)}x^{(2)} \end{pmatrix} \right\rangle + \left\langle \begin{pmatrix} c^{(2)}(1) & 0 \\ c^{(2)}(2) & 0 \end{pmatrix} \middle| \begin{pmatrix} 1 & x^{(1)} \\ x^{(2)} & x^{(1)}x^{(2)} \end{pmatrix} \right\rangle \\ &= \left\langle \begin{pmatrix} c^{(1)}(1) + c^{(2)}(1) & c^{(1)}(2) \\ c^{(2)}(2) & 0 \end{pmatrix} \middle| \begin{pmatrix} 1 & x^{(1)} \\ x^{(2)} & x^{(1)}x^{(2)} \end{pmatrix} \right\rangle. \end{aligned}$$

3.7.6 Feedback connection

The third connection investigated is a feedback between two affine MTI systems. The feedback structure is depicted in Figure 3.17, such that the subsystems have no external inputs, which gives $m_1 = m_2 = 0$. The inputs of the subsystems follow from the outputs of the other subsystem respectively, leading to the coupling equations

$$\mathbf{s}^{(1)} = \mathbf{y}^{(2)}, \tag{3.143}$$

$$\mathbf{s}^{(2)} = \mathbf{y}^{(1)}. \tag{3.144}$$

Thus, the number of inputs of the first system must be equal to the number of outputs of the second system ($m_{s,1} = p_2$) and the number of outputs of the first system must be equal to number of inputs of the second system ($p_1 = m_{s,2}$). The output of the first system should be the output of the overall system

$$\mathbf{y} = \mathbf{y}^{(1)}. \tag{3.145}$$

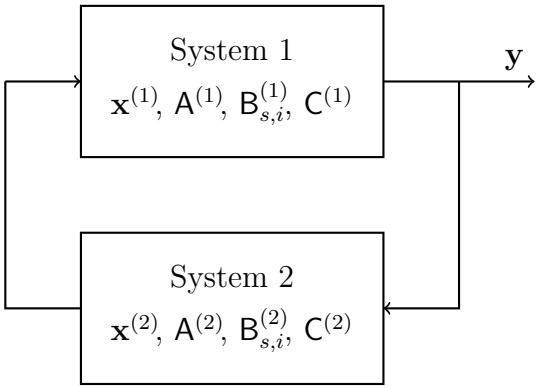


Figure 3.17: Feedback connection

With the feedback connection (3.143) to (3.145) the model is given by

$$\begin{aligned}
\Phi\left(\mathbf{x}^{(1)}\right) &= \left\langle \mathsf{A}^{(1)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}\right) \right\rangle + \sum_{i=1}^{m_{s,1}} \left\langle \mathsf{B}^{(1)}_{s,i} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}\right) \right\rangle \left\langle \mathsf{C}^{(2)} \bar{\times}_{p_2+1} \mathbf{e}_{p_2,i} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(2)}\right) \right\rangle, \\
\Phi\left(\mathbf{x}^{(2)}\right) &= \left\langle \mathsf{A}^{(2)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(2)}\right) \right\rangle + \sum_{i=1}^{m_{s,2}} \left\langle \mathsf{B}^{(2)}_{s,i} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(2)}\right) \right\rangle \left\langle \mathsf{C}^{(1)} \bar{\times}_{p_1+1} \mathbf{e}_{p_1,i} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}\right) \right\rangle, \\
\mathbf{y} &= \left\langle \mathsf{C}^{(1)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}\right) \right\rangle.
\end{aligned}$$

As in Subsection 3.7.4 with (3.123) the multiplications of the vector functions

$$\left\langle \mathsf{B}^{(1)}_{s,i} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}\right) \right\rangle \left\langle \mathsf{C}^{(2)} \bar{\times}_{p_2+1} \mathbf{e}_{p_2,i} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(2)}\right) \right\rangle = \left\langle \mathsf{D}^{(1)}_{i} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right) \right\rangle, \tag{3.146}$$

$$\left\langle \mathsf{B}^{(2)}_{s,i} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(2)}\right) \right\rangle \left\langle \mathsf{C}^{(1)} \bar{\times}_{p_1+1} \mathbf{e}_{p_1,i} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}\right) \right\rangle = \left\langle \mathsf{D}^{(2)}_{i} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right) \right\rangle, \tag{3.147}$$

with $i = 1, \ldots, m_{s,j},\ j = 1, 2$ respectively are computed elementwise by operational tensors leading to

$$\mathsf{D}_i^{(1)} = \coprod_{j=1}^{n_1} {}_{n_1+n_2+1} \left(\left(\mathsf{C}^{(2)} \bar{\times}_{p_2+1} \mathbf{e}_{p_2,i} \right) \circ \left(\mathsf{B}_{s,i}^{(1)} \bar{\times}_{n_1+1} \mathbf{e}_{n_1,j} \right) \right), \tag{3.148}$$

$$\mathsf{D}_i^{(2)} = \coprod_{j=1}^{n_2} {}_{n_1+n_2+1} \left(\left(\mathsf{B}_{s,i}^{(2)} \bar{\times}_{n_2+1} \mathbf{e}_{n_2,j} \right) \circ \left(\mathsf{C}^{(1)} \bar{\times}_{p_1+1} \mathbf{e}_{p_1,i} \right) \right). \tag{3.149}$$

Using that as well as the extended parameter tensors (3.104) and (3.105), the state equations are simplified to

$$\Phi\left(\mathbf{x}^{(1)}\right) = \left\langle \tilde{\mathsf{A}}^{(1)} + \sum_{i=1}^{m_{s,1}} \mathsf{D}_i^{(1)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right) \right\rangle, \tag{3.150}$$

$$\Phi\left(\mathbf{x}^{(2)}\right) = \left\langle \tilde{\mathsf{A}}^{(2)} + \sum_{i=1}^{m_{s,2}} \mathsf{D}_i^{(2)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right) \right\rangle. \tag{3.151}$$

To merge the state equations to get the description with respect to the common state vector $\mathbf{x} = \begin{pmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} \end{pmatrix}^T$, the tensor A is constructed by the concatenation

$$\mathsf{A} = \left(\tilde{\mathsf{A}}^{(1)} + \sum_{i=1}^{m_1} \mathsf{D}_i^{(1)} \right) \boxplus_{n_1+n_2+1} \left(\tilde{\mathsf{A}}^{(2)} + \sum_{i=1}^{m_2} \mathsf{D}_i^{(2)} \right). \tag{3.152}$$

The overall system has no external inputs, i.e. $m = 0$. Because of that no parameter tensor B_i has to be constructed for the system description. The output of the overall system is equal to the output of the first system, resulting in the parameter tensor of the output equation

$$\mathsf{C} = \tilde{\mathsf{C}}^{(1)}. \tag{3.153}$$

The following proposition summarizes the results presented before.

Proposition 3.5 (Feedback connection of two affine MTI system) *The feedback connection of two affine MTI systems in a tensor structure (3.100) and (3.101) results in an affine MTI system with no external inputs*

$$\begin{aligned} \Phi\left(\begin{pmatrix} \mathbf{x}^{(1)} \\ \mathbf{x}^{(2)} \end{pmatrix} \right) &= \left\langle \mathsf{A} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right) \right\rangle, \\ \mathbf{y} &= \left\langle \mathsf{C} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right) \right\rangle. \end{aligned}$$

The parameter tensors of the connected system are computed from the subsystems by

$$\begin{aligned} \mathsf{A} &= \left(\tilde{\mathsf{A}}^{(1)} + \sum_{i=1}^{m_1} \mathsf{D}_i^{(1)} \right) \boxplus_{n_1+n_2+1} \left(\tilde{\mathsf{A}}^{(2)} + \sum_{i=1}^{m_2} \mathsf{D}_i^{(2)} \right), \\ \mathsf{C} &= \tilde{\mathsf{C}}^{(1)} \end{aligned}$$

with tensors $\mathsf{D}_i^{(1)}$ *and* $\mathsf{D}_i^{(2)}$ *given by (3.148) and (3.149).*

Example 3.24 *Example 3.22 introduced two first order affine MTI systems. Inserting the constraints of the feedback connection, give the state equations that can be summarized by (3.146) to (3.152). The equation of the first state results in*

$$\begin{aligned}
\Phi\left(x^{(1)}\right) &= \left\langle \tilde{\mathsf{A}}^{(1)} \,\middle|\, \mathsf{M}\left(x^{(1)}, x^{(2)}\right) \right\rangle + \left\langle \mathsf{B}_{s,1}^{(1)} \,\middle|\, \mathsf{M}\left(x^{(1)}\right) \right\rangle \left\langle \mathsf{C}^{(1)} \,\middle|\, \mathsf{M}\left(x^{(2)}\right) \right\rangle \\
&= \left\langle \begin{pmatrix} a^{(1)}(1) & a^{(1)}(2) \\ 0 & 0 \end{pmatrix} \,\middle|\, \begin{pmatrix} 1 & x^{(1)} \\ x^{(2)} & x^{(1)}x^{(2)} \end{pmatrix} \right\rangle + \left\langle \begin{pmatrix} b_s^{(1)}(1) \\ b_s^{(1)}(2) \end{pmatrix} \,\middle|\, \begin{pmatrix} 1 \\ x^{(1)} \end{pmatrix} \right\rangle \left\langle \begin{pmatrix} c^{(2)}(1) \\ c^{(2)}(2) \end{pmatrix} \,\middle|\, \begin{pmatrix} 1 \\ x^{(2)} \end{pmatrix} \right\rangle \\
&= \left\langle \begin{pmatrix} a^{(1)}(1) & a^{(1)}(2) \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} c^{(2)}(1)b_s^{(1)}(1) & c^{(2)}(1)b_s^{(1)}(2) \\ c^{(2)}(2)b_s^{(1)}(1) & c^{(2)}(2)b_s^{(1)}(2) \end{pmatrix} \,\middle|\, \begin{pmatrix} 1 & x^{(1)} \\ x^{(2)} & x^{(1)}x^{(2)} \end{pmatrix} \right\rangle.
\end{aligned}$$

The description of the second state is the same than in the serial case (3.134), because the subsystem has no external inputs and the output of the first system is the input of the second system here, too. To get the parameter tensors of the overall feedback system the parameter tensors of the two state equations are concatenated by (3.152) resulting in

$$\Phi\left(\begin{pmatrix} x^{(1)} \\ x^{(2)} \end{pmatrix}\right) = \left\langle \begin{array}{cc|cc}
 & a^{(2)}(1)+ & & b_s^{(2)}(1)c^{(1)}(2) \\
 & b_s^{(2)}(1)c^{(1)}(1) & & \\
a^{(1)}(1)+ & & a^{(1)}(2)+ & \\
c^{(2)}(1)b_s^{(1)}(1) & & c^{(2)}(1)b_s^{(1)}(2) & \\
 & a^{(2)}(2)+ & & b_s^{(2)}(2)c^{(1)}(2) \\
 & b_s^{(2)}(2)c^{(1)}(1) & & \\
c^{(2)}(2)b_s^{(1)}(1) & & c^{(2)}(2)b_s^{(1)}(2) & \\
\hline
1 & & x^{(1)} & \\
 & & & \\
x^{(2)} & & x^{(1)}x^{(2)} & \\
 & & &
\end{array} \right\rangle.$$

The output of the feedback system is equal to the output of the first subsystem, such that the output equation is written with (3.153) as

$$y = \left\langle \tilde{\mathsf{C}}^{(1)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right) \right\rangle = \left\langle \begin{pmatrix} c^{(1)}(1) & c^{(1)}(2) \\ 0 & 0 \end{pmatrix} \,\middle|\, \begin{pmatrix} 1 & x^{(1)} \\ x^{(2)} & x^{(1)}x^{(2)} \end{pmatrix} \right\rangle.$$

3.8 Open questions

The chapter showed, how the four proposed decomposition methods are applied to represent MTI models. The advantages and drawbacks of the particular methods are highlighted. Different tools for the work with MTI models have been introduced, that use the decomposed MTI model structure. But with the developed methods also some new research questions arise, that are left open.

On the one hand the future developments in the mathematical field of tensor calculus have to be observed. The tensor methods are currently an active field of research, where it is expected, that new methods and operations are published in the future. The applicability of these methods to MTI systems has to be checked to get even more efficient model representations. Especially of interest are decomposition methods that preserve the sparsity structure

of the original tensor. The positions with zero entries in the original tensor of the MTI model should also contain zeros after a low-rank approximation. This setting guarantees that by low-rank approximation no monomials can influence the model dynamics that are not contained in the original model. This could lead to errors in the model dynamics. Structural constraints on tensors are possible by structured data fusion introduced by [104]. But this does not allow the desired operations yet. Furthermore since the translation from CP to the other decomposition techniques is not optimal regarding the storage effort, conversion from the state equations to the other decomposition techniques are interesting without using the CP decomposition as intermediate step.

Another open point, that follows from this chapter is the simulation of MTI models. It has been shown in Section 3.2, that the right hand sides of an MTI state space model are evaluated efficiently based in the decomposition factors, if the parameter tensors are given in one of the four decomposition formats. But during simulation of continuous-time models standard integration routines are used. Here the question arises, if it possible to use the decomposed model structure also inside the integration routines to get a fast and robust simulation. Therefore algorithms like the symbolic Runge-Kutta should be investigated and reformulated by using the tensor approach, [103]. Tools like the multi-step transition of Section 3.6 may help for that, since state evaluations at different time instants are necessary for the integration routines.

As pointed out before, a more efficient polynomial representation would have a positive effect here e.g. on the multi-step transitions. For large-scale systems and many time steps the number of dimensions for the parameter tensor gets very large, which could lead to problems in memory demand even though the tensors are given in a decomposed form. Another tensor representation of polynomials would lead to improvements in this area.

A property of MTI systems is, that they are not closed regarding different connections or the multi-step transitions. This is a huge difference to linear models, where connections or multi-step transitions are still given by linear functions. As shown in this chapter for MTI models this holds e.g. for feedback connections only for subclasses like the affine MTI systems or results in higher order polynomials like in the multi-step transition. An open question is, if other MTI model representations e.g. by similarity transformations are possible, such that e.g. also the feedback connection of general MTI systems stays in the multilinear class. This would be beneficial, since no other concepts like polynomials would have to be considered resulting in simpler computations with MTI systems.

In this thesis MTI models are derived by a grey-box modeling approach, [68]. The dynamical behavior of the plants are described by first physical principles as e.g. in Section 5.1 and rearranged in the MTI model structure with decomposed tensors afterwards. The model parameters are identified by a parameter estimation with measurement data, i.e. the parameters in the fixed model structure given by the physical equations are estimated, such that the simulated model outputs fit well to the output measurements of the plant, if they are fed with the same inputs. Here standard optimization routines for nonlinear systems are used for that. It is open to adapt these optimization routines to the multilinear model structure and the representation by decomposed tenors to improve the algorithms e.g. regarding speed and convergence.

Another open point concerning system identification is the black-box identification for MTI models. In black-box identification input-output measurements are used to estimate a state space model. The system is taken as black-box without any knowledge on the underlying physical relationships. For linear models very efficient algorithms are available, [108]. For MTI models no black-box identification algorithm is available. First results were presented in [7, 8, 24], such that it is still an open point. It would be very beneficial, because before applying a model-based controller method, a model of the plant has to be identified, which can be complicated without a black-box approach. If an MTI model could be identified just by using measurement data, this would simplify the whole modeling process and thus widen the application area of the MTI models and the presented tools and design methods.

Additionally further analysis of the class of MTI models is open. Properties like stability, controllability or observability are not investigated yet. Furthermore the design of observers of MTI systems is interesting. All these concepts are defined for linear and nonlinear systems. But with a special focus on MTI systems they are still open and would help for system analysis or controller design if the MTI subclass assumption brings more efficient algorithms or better analysis methods compared to the general nonlinear ones. With the linearization approach of Section 3.3 only local assumptions are possible by applying the concepts of linear system theory.

All decomposition methods investigated here are applied to continuous- or discrete-time models. No e.g. hybrid models are considered as in [85]. The introduced concepts should be applicable to these class of models too, but the validity has to be checked. Furthermore, also other application areas are possible like in diagnosis with qualitative models, where the simulation of stochastic automata has a similar structure than the MTI models, such that it might be expressed in the tensor framework, [65, 77, 79]. The reusability of state space models in different application areas like control and diagnosis was shown for the linear case in [52]. For multilinear state space models this is a current field of research.

4 Controller design for MTI systems

The previous sections described mathematical operations with multilinear functions in the tensor framework and their application to model dynamical systems by MTI state space models. This chapter focuses on model-based controller design techniques for MTI systems, i.e., the model of the system is used in the design phase of the controller or inside the controller during operation. Many model-based design methods are available that work with linear models, that approximate the nonlinear system behavior, [4, 23]. On the other hand general nonlinear design techniques exist for nonlinear systems or special subclasses of nonlinear systems, [26, 43]. But for MTI systems no general design method has been introduced so far. Chapter 3 showed the good modeling properties of MTI systems, especially for large-scale, complex systems. Because of that, MTI systems are in the focus here to take advantage from their modeling properties also in controller synthesis and operation.

The chapter is structured as follows. Section 4.1 introduces the general concept of the considered controller design techniques, i.e. state feedback control, feedback linearization and MPC, as they are known in the literature. The following sections describe the applications and adaption of these model-based design methods to state space models of MTI systems. Section 4.2 shows the feedback linearization of MTI systems in tensor framework. A decentral system structure for a state feedback controller is determined in Section 4.3. The three following sections deal with MPC, where in Section 4.4 the properties of the MPC optimization problem are investigated using MTI systems directly. Section 4.5 shows how a linear MPC can be used in combination with a multilinear state space model. Ways to divide the resulting MPC optimization problem to several nodes are investigated in Section 4.6. The chapter closes with a summary of the resulting open questions in the field of controller design for MTI systems in Section 4.7.

4.1 Controller design basics

This section introduces the basic concepts of the three model-based controller design methods considered in this thesis. Therefore, Subsection 4.1.2 describes, how a general affine nonlinear system is feedback linearized by a state feedback. The two other methods, state feedback design in Subsection 4.1.1 and MPC in Subsection 4.1.3 are introduced for the use with linear state space models basically. Additionally, it is highlighted, how these two methods are used in a decentralized context. The basic concepts introduced here are used with MTI models in the next sections, according to the guiding questions given in Subsection 4.1.4.

4.1.1 State feedback control

LQR design is a well known state feedback control design method, [61]. In standard design this leads to a centralized controller that needs the full system information, i.e. information on all system states to compute the control inputs for the plant. Therefore, consider a state feedback control loop with feedback gain $\mathbf{K}$ as depicted in Figure 4.1.

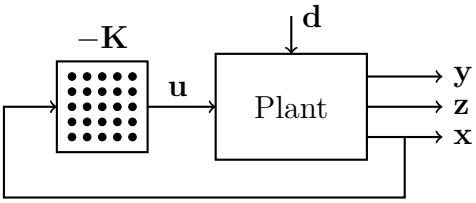


Figure 4.1: State feedback control with central controller

In standard LQR design the plant is described by a continuous-time linear state space model as (3.10) and (3.11)

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}_u\mathbf{u} + \mathbf{B}_d\mathbf{d}, \tag{4.1}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x}, \tag{4.2}$$

$$z = \mathbf{C}_z\mathbf{x} + \mathbf{D}_z\mathbf{u}. \tag{4.3}$$

The inputs are divided into control inputs $\mathbf{u} \in \mathbb{R}^m$ and disturbance inputs $\mathbf{d} \in \mathbb{R}^{m_d}$ with the corresponding control and disturbance input matrices $\mathbf{B}_u \in \mathbb{R}^{n\times m}$ and $\mathbf{B}_d \in \mathbb{R}^{n\times m_d}$ respectively. The loop in Figure 4.1 is closed by feeding back the states of the system with a static gain matrix $\mathbf{K} \in \mathbb{R}^{m\times n}$ leading to the control input of the plant

$$\mathbf{u} = -\mathbf{K}\mathbf{x}, \tag{4.4}$$

where in general all states $\mathbf{x} \in \mathbb{R}^n$ are used to compute the input signal $\mathbf{u} \in \mathbb{R}^m$ of the plant. In addition to the measured output $\mathbf{y}$ a performance output $z \in \mathbb{R}$ is introduced, with matrices $\mathbf{C}_z = \begin{pmatrix}\mathbf{Q}^{1/2} & \mathbf{0}\end{pmatrix}^T$ and $\mathbf{D}_z = \begin{pmatrix}\mathbf{0} & \mathbf{R}^{1/2}\end{pmatrix}^T$. The matrices $\mathbf{Q} = \mathbf{Q}^T \geq 0$ and $\mathbf{R} = \mathbf{R}^T > 0$ are weights for state and input performance. It is assumed, that $(\mathbf{A}, \mathbf{B}_u)$ is stabilizable and $(\mathbf{A}, \mathbf{Q}^{1/2})$ is detectable. In [67] for the centralized design the design problem is formulated as an $\mathcal{H}_2$ optimization problem. A feedback gain $\mathbf{K}$ has to be found, such that the $\mathcal{H}_2$ norm of the transfer function from $\mathbf{d}$ to z of the closed loop system is minimized

$$\min_{\mathbf{K}} J(\mathbf{K}), \tag{4.5}$$

where the objective function $J(\mathbf{K})$ is the $\mathcal{H}_2$ norm of the transfer function from $\mathbf{d}$ to z of the system (4.1) to (4.3). To solve the optimization problem (4.5) the objective function is written as

$$J(\mathbf{K}) = \begin{cases} \text{trace}\left(\mathbf{B}_d^T\mathbf{P}(\mathbf{K})\mathbf{B}_d\right) & \text{, for } \mathbf{K} \text{ stabilizing,} \\ \infty & \text{, else,} \end{cases} \tag{4.6}$$

where $\mathbf{P}(\mathbf{K})$ is the resulting closed loop observability Gramian

$$\mathbf{P}(\mathbf{K}) = \int_0^\infty e^{(\mathbf{A}-\mathbf{B}_u\mathbf{K})^T t} \left(\mathbf{Q} + \mathbf{K}^T\mathbf{R}\mathbf{K}\right) e^{(\mathbf{A}-\mathbf{B}_u\mathbf{K})t} \mathrm{d}t,$$

which is the solution of the Lyapunov equation

$$(\mathbf{A} - \mathbf{B}_u\mathbf{K})^T \mathbf{P} + \mathbf{P}(\mathbf{A} - \mathbf{B}_u\mathbf{K}) = -\left(\mathbf{Q} + \mathbf{K}^T\mathbf{R}\mathbf{K}\right).$$

With the solution of the Lyapunov equation the optimal feedback gain is given by

$$\mathbf{K} = -\mathbf{R}^{-1}\mathbf{B}_u^T\mathbf{P},$$

which results in a centralized controller gain, i.e. all communication links between the states and the inputs are necessary, since the system structure has no influence during controller synthesis, [66]. In general $k(i,j) \neq 0$ holds for all elements of the gain matrix, which means that all communication links have to be established. The performance of the controller is the only optimization criterion.

Since nowadays the plants get more and more complex, in [66] a method to impose a certain structure constraint on the feedback gain is proposed, in contrast to the centralized design. This results in zero entries of the feedback matrix $\mathbf{K}$ and reduces the communication effort, because not all signal connections between the states $\mathbf{x}$ and inputs $\mathbf{u}$ have to be established. Only those connections between x_j and u_i are necessary, where the relation $k(i,j) \neq 0$ is fulfilled. The algorithm developed in [66] computes a sparsity structure of $\mathbf{K}$, such that there is a trade-off between sparsity and overall control performance. It is investigated, which communication links can be cut with a slight decrease in the overall control performance only, which leads to benefits in the communication effort. An overview of the algorithm for linear systems is given here as described in [28, 67]. In decentralized design the feedback gain should have a given sparsity structure leading to the optimization problem

$$\begin{aligned} &\min_{\mathbf{K}} J(\mathbf{K}), \\ &\text{subject to } \mathbf{K} \in \mathcal{S}, \end{aligned} \tag{4.7}$$

where $\mathcal{S}$ is the structure of $\mathbf{K}$, [66]. To determine the sparsity structure $\mathcal{S}$ the objective function in (4.5) is extended to

$$\min_{\mathbf{K}} J(\mathbf{K}) + \gamma g(\mathbf{K}), \tag{4.8}$$

where the function $g(\mathbf{K})$ measures the sparsity of $\mathbf{K}$, [67]. The factor γ is a tuning parameter to find a trade-off between control performance and sparsity. Setting $\gamma = 0$ leads to the centralized problem (4.5). A larger value for $\gamma > 0$ means that sparsity is more promoted. In Section 4.3 a way to choose γ for a good trade-off is described. If $g(\mathbf{K})$ is chosen to be the cardinality function

$$g(\mathbf{K}) = \mathrm{card}(\mathbf{K}),$$

that measures the number of nonzero entries of $\mathbf{K}$, the optimization problem (4.8) is non-convex. Solving this optimization becomes a combinatorial problem, which is very complex. For a convex relaxation the l_1-norm of $\mathbf{K}$

$$\|\mathbf{K}\|_1 = \sum_{i=1}^{m} \sum_{j=1}^{n} |k(i,j)|$$

is often used to approximate the cardinality function, [12]. But there is a difference between the cardinality function and the l_1-norm, since with the cardinality function all nonzero entries are weighted the same. With the l_1-norm more weight is put on elements with larger magnitude. To bridge this gap between cardinality function and l_1-norm, a weighted l_1-norm is chosen here

$$g(\mathbf{K}) = \sum_{i=1}^{m} \sum_{j=1}^{n} w(i,j)\,|k(i,j)|\,,$$

to measure the sparsity of $\mathbf{K}$, [15]. The weights should balance the varying influences of elements of different magnitude in the result. E.g. by choosing the weights to

$$w(i,j) = \begin{cases} \frac{1}{|k(i,j)|} & \text{, for } k(i,j) \neq 0, \\ \infty & \text{, otherwise,} \end{cases}$$

the weighted l_1-norm is equal to the cardinality function. But this choice of the weights is not possible for implementation, since the weights depend on the resulting elements of the solution $k(i,j)$. Therefore, the weights are determined by an iterative process, where the weights of the next iteration are computed by using the result of the previous iteration, as proposed in [15]. The optimization problem (4.8) is solved by an ADMM algorithm to determine the sparsity structure, [67]. Finally, if a sparsity structure is found, a polishing step is done to compute an optimal feedback gain for the determined structure by solving (4.7). As shown in [66] this problem is solved by using Newton's method in conjunction with a conjugate gradient scheme. This results in a decentralized feedback controller with less communication effort compared to the central case and little loss in the control performance only. The control loop with the sparse controller is depicted in Figure 4.2. The sparsity pattern of the gain matrix is illustrated inside the controller block.

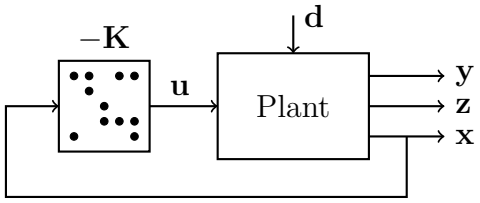


Figure 4.2: State feedback control with sparse controller

4.1.2 Feedback linearization

This subsection highlights the most important features of the feedback linearization as described in [39] or [43], which is a controller design method for scalar nonlinear systems. The main idea of this method is to design an affine nonlinear state feedback controller such that the reference behavior in closed loop from reference input r to output y is linear. The main setup of feedback linearization is shown in Figure 4.3.

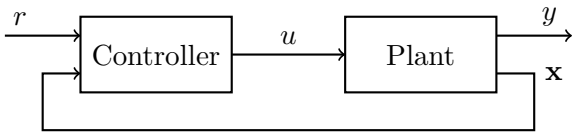


Figure 4.3: Closed loop state feedback

Consider a continuous-time, nonlinear, affine, single-input single-output (SISO) system

$$\dot{\mathbf{x}} = \mathbf{a}(\mathbf{x}) + \mathbf{b}(\mathbf{x})u, \tag{4.9}$$

$$y = c(\mathbf{x}), \tag{4.10}$$

with nonlinear functions $\mathbf{a}, \mathbf{b} : \mathbb{R}^n \to \mathbb{R}^n$ of the state and output $c : \mathbb{R}^n \to \mathbb{R}$ that is a specialization of the general MIMO case (4.9) and (4.10). It is assumed that the functions are sufficiently smooth, which means that all later appearing partial derivatives of the functions are defined and continuous, [43]. The task is to find a nonlinear state feedback controller of the form

$$u = k(\mathbf{x}) + v(\mathbf{x})r, \tag{4.11}$$

with the closed loop reference r, controller function $k(\mathbf{x})$ and prefilter $v(\mathbf{x})$, such that the closed loop shows a linear behavior from r to y prescribed by the linear differential equation

$$\mu_{\rho_{sys}} \frac{\partial^{\rho_{sys}}}{\partial t^{\rho_{sys}}} y + \mu_{\rho_{sys}-1} \frac{\partial^{\rho_{sys}-1}}{\partial t^{\rho_{sys}-1}} y + \cdots + \mu_1 \dot{y} + \mu_0 y = \mu_0 r, \tag{4.12}$$

where ρ_{sys} is the relative degree or index of the plant that is defined in the following. Without loss of generality the factor $\mu_{\rho_{sys}}$ is set to one. The factors are determined, e.g. by a pole placement using an LQR design for a linearization of the model, [23]. It is easy to find such controller if the system is in nonlinear controller canonical form, [39]. Since nonlinear systems like (4.9) and (4.10) do not show the structure of this normal form in general, a state space transformation has to be found. Therefore, the definition (2.57) of Lie derivatives is useful, [39]. With help of Lie derivatives a state space transformation of (4.9) and (4.10) to controller canonical form can be found. Using (2.57), the time derivatives of the output of the system are given by

$$\frac{\partial^i}{\partial t^i} y = L_{\mathbf{a}}^i c(\mathbf{x}) + L_{\mathbf{b}} L_{\mathbf{a}}^{i-1} c(\mathbf{x}) \cdot u \ , \ i = 1, \ldots, \rho_{sys}. \tag{4.13}$$

The relative degree ρ_{sys} describes the lowest derivative of the output y that is directly influenced by the input u, which means that there is a direct feedthrough of the input to the ρ_{sys}^{th} derivative of the output. Thus, the relative degree is defined by

$$L_{\mathbf{b}} L_{\mathbf{a}}^{i} c(\mathbf{x}) = 0 \;\forall\; \mathbf{x}, \quad 0 \leq i \leq \rho_{sys} - 2, \tag{4.14}$$

$$L_{\mathbf{b}} L_{\mathbf{a}}^{\rho_{sys}-1} c(\mathbf{x}) \neq 0 \;\forall\; \mathbf{x}. \tag{4.15}$$

It is assumed that the system (4.9) and (4.10) has a well defined relative degree $\rho_{sys} \leq n$, [34]. In this case a state transformation

$$\tilde{\mathbf{x}} = \mathbf{t}(\mathbf{x}) = \begin{pmatrix} c(\mathbf{x}) & L_{\mathbf{a}} c(\mathbf{x}) & \cdots & L_{\mathbf{a}}^{\rho-1} c(\mathbf{x}) & t_{\rho+1}(\mathbf{x}) & \cdots & t_n(\mathbf{x}) \end{pmatrix}^T$$

can be found, that leads to the controller canonical form

$$\begin{pmatrix} \dot{\tilde{x}}_1 \\ \vdots \\ \dot{\tilde{x}}_{\rho_{sys}-1} \\ \dot{\tilde{x}}_{\rho_{sys}} \\ \dot{\tilde{x}}_{\rho_{sys}+1} \\ \vdots \\ \dot{\tilde{x}}_{\rho_{sys}} \end{pmatrix} = \begin{pmatrix} \tilde{x}_2 \\ \vdots \\ \tilde{x}_{\rho_{sys}} \\ L_{\mathbf{a}}^{\rho_{sys}} c(\mathbf{x}) + L_{\mathbf{b}} L_{\mathbf{a}}^{\rho_{sys}-1} c(\mathbf{x}) u \\ q_{\rho_{sys}+1}(\tilde{\mathbf{x}}) \\ \vdots \\ q_n(\tilde{\mathbf{x}}) \end{pmatrix},$$

$$y = \tilde{x}_1,$$

where $q_i(\tilde{\mathbf{x}})$, $i = \rho_{sys}+1, \ldots, n$ describe the right hand sides of the zero dynamics. If the relative degree is smaller than the number of states $\rho_{sys} < n$, the stability of the zero dynamics has to be checked additionally, [39]. Since the system is transformed to controller canonical form, the controller (4.11) is easily derived by setting

$$u = k(\mathbf{x}) + v(\mathbf{x}) r = \frac{-\sum_{i=0}^{\rho_{sys}} \mu_i L_{\mathbf{a}}^{i} c(\mathbf{x}) + \mu_0 \, r}{L_{\mathbf{b}} L_{\mathbf{a}}^{\rho_{sys}-1} c(\mathbf{x})}. \tag{4.16}$$

Application of the controller in closed loop

$$\dot{\mathbf{x}} = \mathbf{a}(\mathbf{x}) + \mathbf{b}(\mathbf{x}) k(\mathbf{x}) + \mathbf{b}(\mathbf{x}) v(\mathbf{x}) r,$$
$$y = c(\mathbf{x}),$$

results in a state space model in the transformed variables

$$\begin{pmatrix} \dot{\tilde{x}}_1 \\ \dot{\tilde{x}}_2 \\ \vdots \\ \dot{\tilde{x}}_{\rho_{sys}-1} \\ \dot{\tilde{x}}_{\rho_{sys}} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -\mu_0 & -\mu_1 & -\mu_2 & \cdots & -\mu_{\rho_{sys}-1} \end{pmatrix} \begin{pmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \vdots \\ \tilde{x}_{\rho_{sys}-1} \\ \tilde{x}_{\rho_{sys}} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \mu_0 \end{pmatrix} r,$$

$$\begin{pmatrix} \dot{\tilde{x}}_{\rho_{sys}+1} \\ \vdots \\ \dot{\tilde{x}}_n \end{pmatrix} = \begin{pmatrix} q_{\rho_{sys}+1}(\tilde{\mathbf{x}}) \\ \vdots \\ q_{n(\tilde{\mathbf{x}})} \end{pmatrix},$$

$$y = \tilde{x}_1,$$

which gives the desired linear input-output behavior (4.12), since the output is not influenced by the zero dynamics, [43].

4.1.3 Model predictive control

This subsection introduces the main aspects of MPC following the notation of [73]. MPC is a model-based controller design technique. Inside the controller, a model of the plant is used to predict the future plant behavior to upcoming inputs. Using the model the controller computes the optimal future control input sequence at each sampling instant with respect to a cost or objective function describing the control goal, i.e. the desired behavior of the plant, e.g. reference tracking. The controller loop is depicted in Figure 4.4.

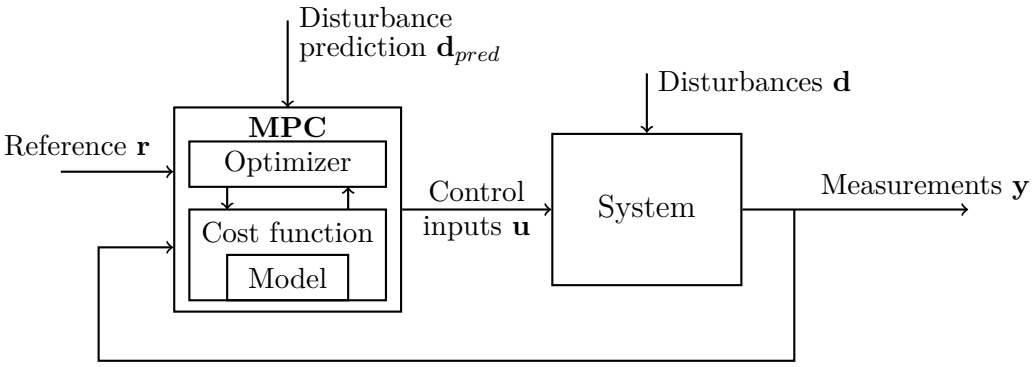


Figure 4.4: Closed loop structure of MPC

As shown in the figure, the plant is influenced by control inputs $\mathbf{u}$ and disturbance inputs $\mathbf{d}$. In contrast to the control inputs from the controller, the disturbances represent the ambient conditions of the plant, that cannot be influenced by the controller. The plant is equipped with sensors, such that measurement information is available by signals $\mathbf{y}$. The controller receives the measurement information of the plant to initialize the model. In this thesis, it is assumed that all states of the system are measured, i.e. the output vector is equal to the state vector. Thus, no observer to estimate the state information from the output measurements is necessary. An optimal control sequence is computed with a length of H_u time steps. This is called the control horizon. The optimal control sequence is determined by minimizing the cost function $J(k)$ by an optimizer. The future plant behavior resulting from a certain input sequence is predicted by a forward simulation of the plant model over the so called prediction horizon of H_p time steps. The prediction horizon is always larger than or equal to the control horizon $H_p \geq H_u$. The simulation of the model with different control inputs considers predictions of the disturbances $\mathbf{d}_{pred}(k+i),\ i = 1, \ldots, H_p$ for the whole prediction horizon. If the plant should follow a reference, the signal $\mathbf{r}(k+i),\ i = 1, \ldots, H_p$ should be available too for the whole prediction horizon to benefit from the predictive behavior of the controller. The result of the optimization problem

$$\min_{\substack{\mathbf{u}(k+i),\\ i=0,\ldots,H_u-1}} J(k)$$

is the control sequence $\hat{\mathbf{u}}(k+i),\ i=0,\ldots,H_u-1$, that is optimal with respect to the cost function $J(k)$. Only the control input $\hat{\mathbf{u}}(k)$ of the first time step of this sequence is given to the plant. At the next time instant the model is initialized again with current measurement data and the optimization is solved again. This is called receding horizon principle, [73].

Different control goals are possible in MPC. A very common desired closed loop behavior is reference tracking, where the cost function penalizes the quadratic difference of the outputs to their reference values. Since not an arbitrary large control effort should be used to follow the reference, the control effort, which is measured by the change of the control signal from one time step to the next one, is considered in the cost function too.
To show an alternative cost function in contrast to that, in economic model predictive control (EMPC) no reference for the outputs is used. In EMPC the cost function contains higher level factors like the overall economic operation costs or the overall energy consumption. An example for a cost function of an EMPC is given by

$$J(k) = c_1 \dot{Q}_{therm}(k) + c_2 P_{el}(k),$$

where $\dot{Q}_{therm}(k)$ and $P_{el}(k)$ are the thermal and electrical powers, respectively. The coefficients c_1 and c_2 in $€/W$ represent the economical costs, [27]. As for reference tracking the operating range of the system has to be defined by constraints on states, inputs and outputs. But no reference trajectory is given here. During optimization the costs defined in the objective function are minimized, such that the system stays in the prescribed operating range. These are only two examples of cost functions. Depending on the application, other cost functions are possible.

As described before, a model is used to predict the future plant behavior. The complexity of the MPC optimization problem depends on the choice of the model class. Very common in MPC control is the use of a linear model, [73]. With a linear model the optimization problem has nice properties, especially if a quadratic cost function is used. Depending on the quadratic term the optimization problem is convex, i.e. a unique, globally optimal solution can be found and standard solvers are available, [12]. The drawback of a linear model is, that the model describes the plant behavior, that is in general nonlinear, only in a neighborhood of the operating point with a sufficient accuracy, especially when nonlinear effects are essential. This modeling error leads to a control error of the MPC. To avoid this modeling error from linear modeling, nonlinear models are used in nonlinear model predictive control (NMPC). This gives the advantage, that the plant dynamics can be captured better by the model that is used inside the controller. But in general this leads to a more complex optimization problem. The convexity of the optimization problem cannot be guaranteed in this nonlinear case, such that during optimization e.g. local minima are found only. In Sections 4.4 to 4.6 different model classes are investigated for use with MPC.

In the following, the standard formulation of MPC in the literature should be described, that uses a linear model and a quadratic cost function to realize a reference tracking, [73]. This approach is taken as basis for the application of the MPC control structure to MTI systems in Sections 4.4 to 4.6. Linear model predictive control is based on a linear discrete-time

model of the plant

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}_u\mathbf{u}(k) + \mathbf{B}_d\mathbf{d}(k), \tag{4.17}$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k). \tag{4.18}$$

At time k the state $\mathbf{x}(k)$ of the plant is determined to initialize the model. Starting from this initial value the controller predicts the response of the system for several input sequences. The controller optimizes the control sequence, such that the cost function

$$J(k) = \sum_{i=1}^{H_p} \|\hat{\mathbf{x}}(k+i) - \mathbf{r}(k+i)\|_{\mathbf{Q}}^2 + \sum_{i=0}^{H_u-1} \|\Delta\mathbf{u}(k+i)\|_{\mathbf{R}}^2 \tag{4.19}$$

is minimized with input changes $\Delta\mathbf{u}(k+i) = \mathbf{u}(k+i) - \mathbf{u}(k+i-1)$ and a output reference $\mathbf{r}(k+i) \in \mathbb{R}^{p\times 1}$. The outputs $\hat{\mathbf{y}}(k+i),\ i = 1,\ldots,H_p$ are predicted with the plant model. The states should follow a reference trajectory $\mathbf{r}$ with a certain control effort $\Delta\mathbf{u}$. The used norm, e.g. for the first term, is given by

$$\|\hat{\mathbf{x}}(k+i) - \mathbf{r}(k+i)\|_{\mathbf{Q}} = \left(\hat{\mathbf{x}}(k+i) - \mathbf{r}(k+i)\right)^T \mathbf{Q} \left(\hat{\mathbf{x}}(k+i) - \mathbf{r}(k+i)\right).$$

The weighting matrices $\mathbf{Q} \in \mathbb{R}^{p\times p}$ and $\mathbf{R} \in \mathbb{R}^{m\times m}$ with $\mathbf{Q} \geq 0$ and $\mathbf{R} > 0$ are used to weight the difference of the outputs from the reference and the input changes, respectively. An increase in $\mathbf{Q}$ puts more weight on the output reference tracking, which in general results in a larger control effort. Increasing $\mathbf{R}$ leads to less change in the control signal, which yields in general in a slower system. Thus, tuning the weights of the controller is a tradeoff between control effort and tracking performance. The principle of MPC is illustrated in Figure 4.5.

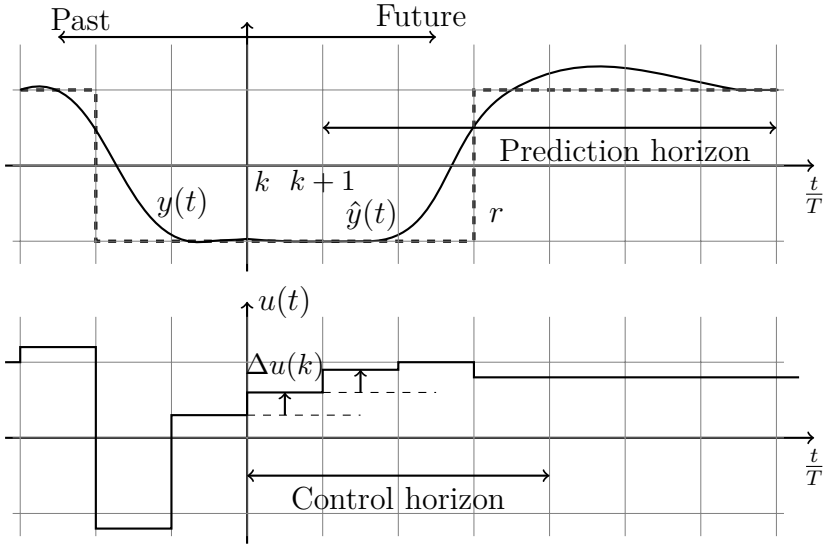


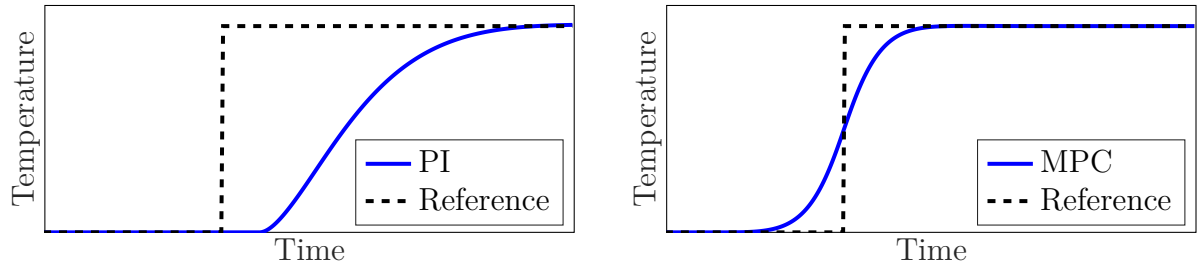
Figure 4.5: Principle of MPC

In each sampling step the cost function (4.19) is minimized. The optimization problem

$$\begin{aligned} \min_{\substack{\mathbf{u}(k+i),\\ i=0,\dots,H_u-1}} \quad & J(k) \qquad (4.20)\\ \text{subject to} \quad \hat{\mathbf{x}}(k+i+1) &= \mathbf{A}\hat{\mathbf{x}}(k+i) + \mathbf{B}_u\mathbf{u}(k+i) + \mathbf{B}_d\mathbf{d}_{pred}(k+i),\ i=0,\dots,H_p-1,\\ \hat{\mathbf{y}}(k+i) &= \mathbf{C}\hat{\mathbf{x}}(k+i),\ i=1,\dots,H_p-1,\\ \mathbf{u}_{min} &\le \mathbf{u}(k+i) \le \mathbf{u}_{max},\ i=0,\dots,H_u-1,\\ \mathbf{x}_{min} &\le \hat{\mathbf{x}}(k+i) \le \mathbf{x}_{max},\ i=1,\dots,H_p. \end{aligned}$$

is solved with respect to several equality and inequality constraints. The initial value of the state $\hat{\mathbf{x}}(k)$ at time k is supposed to be known from measurements directly or via an observer estimated. With the equality constraints the states and ouputs are forced to admit the dynamics of the linear model of the plant. The disturbances are estimated by the disturbance predictions $\mathbf{d}_{pred}$. The operating contraints on inputs and states are described by the inequalities. Here box constraints for inputs and states are assumed. This use of a linear model and quadratic cost function leads to an quadratic programming optimization problem, that can be solved efficiently by standard optimization solvers, [73]. The predictive action of the controller is illustrated in Example 4.1.

Example 4.1 *As example control task a temperature of a room should follow a reference here by controlling a radiator. The plant can be approximated by a first order system with time delay. When controlling this plant with a standard controller like a PI controller the time delay can be clearly identified in the resulting closed loop output and has a negative effect on the control error as shown in Figure 4.6a. The output reacts delayed on the change in the reference signal. Controlling the plant by an MPC controller leads to an improved closed loop behavior. The controller has information on the time constant and the time delay of the plant by the plant model. Thus the controller acts predictively and increases the temperature even before the reference step, as illustrated in Figure 4.6b. This leads to an optimal closed loop behavior with respect to the cost function (4.19) and the time delay has less effects on the control result.*



(a) PI controller

(b) MPC controller

Figure 4.6: Exemplary comparison between PI and MPC controller

Here the online computation of the optimization problem is considered only. Other concepts like explicit MPC, where the optimization is computed offline before operation are described

in the literature, [2, 53]. In this thesis online MPC is considered since MPC problems are investigated for the application area of heating systems which have slow dynamics such that sampling rates in the order of minutes are sufficient. Enough computation time for online optimization is available. Also complex systems are considered here - which would lead to large data sets to be stored for explicit MPC.

4.1.4 Guiding questions

The previous chapter showed the good modelling properties of MTI systems. These models are used in this chapter for model-based controller design with the controller methods that were introduced in the previous sections for applications with linear or nonlinear models but not multilinear ones. Two different approaches are chosen here. On the one hand it is of interest how nonlinear design methods like feedback linearization can be adapted to the special structure of MTI systems, resulting in a controller design method especially for MTI systems. The second question focuses on preservation of the nice properties of controller algorithms for linear models when multilinear systems are used. The design methods considered here are feedback design as a standard design method and MPC as an advanced design method, that is a current field of research in the application to heating systems, [35]. In the design phase or during operation of these controllers, optimization problems have to be solved, that are convex when linear systems are used. The question is, if the linear methods can be adapted by using multilinear models, such that the model accuracy is improved to make the models applicable to a wider range of operation than in the purely linear case. Good properties of the optimization problems in the linear case should be preserved as much as possible also for application to multilinear models. All methods should work with the decomposed tensor structure to get efficient algorithms based on the operations introduced in Chapter 2.

As shown in Chapter 3, MTI systems are well suited to model complex large-scale systems in a systematic way with decomposed tensors. When focusing on control of such large-scale systems a complexity problem could arise. A centralized controller leads to a high communication effort in practice, since all relevant sensors and actuators have to be connected to the controller. Furthermore, the control problem itself gets complex for large-scale systems leading to a high computational complexity. Because of that the application of decentralized or distributed controller concepts to MTI systems is of interest too. Two questions are posed. At first, what is the best decentralized controller structure? This is investigated in combination with a state feedback controller. The second question here is, how the control problem can be split into different system components for a given system structure. Therefore, the control problem of an MPC controller is divided into several MPC nodes leading to a distributed MPC controller.

4.2 Feedback linearization

The controller design by feedback linearization introduced in Section 4.1.2 is investigated in this section for MTI systems with CP decomposed parameter tensors, see [49]. According

to (4.9) and (4.10) an affine SISO MTI system in continuous-time is given by

$$\dot{\mathbf{x}} = \mathbf{a}(\mathbf{x}) + \mathbf{b}(\mathbf{x})u = \langle\, \mathsf{A} \mid \mathsf{M}(\mathbf{x}) \,\rangle + \langle\, \mathsf{B} \mid \mathsf{M}(\mathbf{x}) \,\rangle\, u, \tag{4.21}$$

$$y = c(\mathbf{x}) = \langle\, \mathsf{C} \mid \mathsf{M}(\mathbf{x}) \,\rangle\,. \tag{4.22}$$

It is assumed that the system is controllable, observable and feedback linearizable with well-defined relative degree ρ_{sys}. Theorem 2.3 is used to compute the Lie derivatives of the multilinear model functions in tensor form. This leads to the following Lemma.

Lemma 4.1 (MTI system Lie derivatives) *The Lie derivative (2.60) of the multilinear output function given by tensor* C *along the multilinear vector field* $\mathbf{a}(\mathbf{x})$ *given by tensor* A *yields*

$$L_{\mathbf{a}}^{l} c(\mathbf{x}) = \sum_{i=1}^{n} a_i(\mathbf{x}) \frac{\partial}{\partial x_i} L_{\mathbf{a}}^{l-1} c(\mathbf{x}) = \left\langle\, \mathsf{L}_{\mathsf{A},\mathsf{C},l} \;\middle|\; \mathsf{M}_p^{l+1}(\mathbf{x}) \,\right\rangle, \tag{4.23}$$

with parameter tensor

$$\mathsf{L}_{\mathsf{A},\mathsf{C},l} = \begin{cases} \mathsf{C} & \text{for } l = 0, \\ \sum\limits_{i=1}^{n} \mathsf{A}_i \circ (\mathsf{C} \times_{n-i+1} \boldsymbol{\Theta}) & \text{for } l = 1, \\ \sum\limits_{i=1}^{n} \mathsf{A}_i \circ \left(\sum\limits_{k=1}^{l} \mathsf{L}_{\mathsf{A},\mathsf{C},l-1} \times_{kn-i+1} \boldsymbol{\Theta} \right) & \text{else}, \end{cases} \tag{4.24}$$

with $l = 0, \ldots, \rho_{sys}$ *and subtensor* $\mathsf{A}_i = \mathsf{A}(:, \ldots, :, i),\ i = 1, \ldots, n$ *of* A*, such that*

$$\mathsf{A} = \mathsf{A}_1 \boxplus_{n+1} \mathsf{A}_2 \boxplus_{n+1} \cdots \boxplus_{n+1} \mathsf{A}_n.$$

and $\boldsymbol{\Theta} = \left(\begin{smallmatrix} 0 & 1 \\ 0 & 0 \end{smallmatrix}\right)$ *as in (2.61). Using the parameter tensor*

$$\mathsf{L}_{\mathsf{B},\mathsf{A},\mathsf{C},l} = \sum_{i=1}^{n} \mathsf{B}_i \circ \left(\sum_{k=1}^{l+1} \mathsf{L}_{\mathsf{A},\mathsf{C},l} \times_{kn-i+1} \boldsymbol{\Theta} \right), \tag{4.25}$$

with subtensors $\mathsf{B}_i = \mathsf{B}(:, \ldots, :, i),\ i = 1, \ldots, n$ *of* B*, the Lie derivatives along* $\mathbf{b}(\mathbf{x})$ *are given by*

$$L_{\mathbf{b}} L_{\mathbf{a}}^{l} c(\mathbf{x}) = \sum_{i=1}^{n} b_i(\mathbf{x}) \frac{\partial}{\partial x_i} L_{\mathbf{a}}^{l} c(\mathbf{x}) = \left\langle\, \mathsf{L}_{\mathsf{B},\mathsf{A},\mathsf{C},l} \;\middle|\; \mathsf{M}_p^{l+2}(\mathbf{x}) \,\right\rangle. \tag{4.26}$$

Since all operations, i.e. summation, outer product and mode-k tensor matrix product, used to get the parameter tensors of the Lie derivatives are introduced for decomposed tensors, no full tensor has to be build during computation of the Lie derivatives. This is important especially for large-scale systems. With the tensor description of the Lie derivatives given in Lemma 4.1, the controller law for feedback linearization is constructed as shown in the following lemma.

Lemma 4.2 (MTI feedback linearization) *The feedback linearizing controller for an affine SISO MTI system given by (4.21) and (4.22) reads*

$$u = \frac{-\left\langle \sum_{i=0}^{\rho_{sys}} \mu_i \mathsf{L}_{\mathsf{A},\mathsf{C},i} \,\middle|\, \mathsf{M}_p^{\rho_{sys}+1}(\mathbf{x}) \right\rangle + \mu_0\, r}{\left\langle \mathsf{L}_{\mathsf{B},\mathsf{A},\mathsf{C},\rho_{sys}-1} \,\middle|\, \mathsf{M}_p^{\rho_{sys}+1}(\mathbf{x}) \right\rangle}, \tag{4.27}$$

when the Lie derivatives are defined by parameter tensors as in Lemma 4.1.

Thus, the feedback linearizing controller has a fixed structure. The factors μ_i and tensors $\mathsf{L}_{\mathsf{A},\mathsf{C},i}$ and $\mathsf{L}_{\mathsf{B},\mathsf{A},\mathsf{C},\rho-1}$ have to be provided, to adapt the controller for a given plant. Setting these parameters is like setting a static state feedback gain matrix for pole placement in the linear case. The structure of the controller for MTI systems is fixed and does not depend on the plant. This structural invariance is an advantage for application of the controller. Compared to the general nonlinear case it is not necessary to provide an arbitrary set of mathematical operations, that depends on the plant model. Here it is limited to the given structure and the operations can be defined efficiently on the coefficient spaces of the decomposed factors.

The parameter tensors A, B and C of the system can be used to check if the system is feedback linearizable with relative degree $\rho_{sys} = n$. The conditions for feedback linearizability of [39] are adapted to the MTI case here.

Lemma 4.3 (Feedback linearizability of MTI systems) *An affine MTI system is feedback linearizable with relative degree n at the point $\mathbf{x}_0$, if the following conditions hold*

i. Rank condition

$$\operatorname{rank}\left(\left\langle \mathsf{T} \,\middle|\, \mathsf{M}_p^n(\mathbf{x}_0) \right\rangle\right) = n$$

with tensor $\mathsf{T}(:,\ldots,:,k) = \mathsf{L}_{ad_{\mathbf{a}}\mathbf{b}}^{k-1}$ and $k = 1,\ldots,n$ containing the parameters of the Lie brackets (2.64) with respect to the monomial tensor $\mathsf{M}_p^n(\mathbf{x}_0)$,

ii. Involution condition for the distribution $\operatorname{span}\{\mathbf{b}, ad_{\mathbf{a}}\mathbf{b}, \ldots, ad_{\mathbf{a}}^{n-2}\mathbf{b}\}$

$$\operatorname{rank}\left(\left\langle \mathsf{T}_2(:,\ldots,:,i,j) \,\middle|\, \mathsf{M}_p^{2(n-1)}(\mathbf{x}_0) \right\rangle\right) = \operatorname{rank}\left(\left\langle \mathsf{T}_1 \,\middle|\, \mathsf{M}_p^{n-1}(\mathbf{x}_0) \right\rangle\right),$$

for all i and $j = 0,\ldots,n-2$ with tensors

$$\mathsf{T}_1(:,\ldots,:,k) = \mathsf{L}_{ad_{\mathbf{a}}\mathbf{b}}^{k-1},\ k = 1,\ldots,n-1,$$

$$\mathsf{T}_2(:,\ldots,:,k,i,j) = \begin{cases} \mathsf{L}_{ad_{\mathbf{a}}\mathbf{b}}^{k-1}, & k = 1,\ldots,n-1, \\ \mathsf{L}_{i,j}, & k = n, \end{cases}$$

with $\left\langle \mathsf{L}_{i,j} \,\middle|\, \mathsf{M}_p^{2(n-1)}(\mathbf{x}_0) \right\rangle$ denoting the Lie bracket $[ad_{\mathbf{a}}^i\mathbf{b}, ad_{\mathbf{a}}^j\mathbf{b}](\mathbf{x}_0)$.

If these conditions are met, there exists an output function, such that the system has relative degree n.

The proofs of the Lemmas 4.1 to 4.3 are given in the Appendix A.8.

Example 4.2 *The proposed method for feedback linearization is applied here to a SISO, MTI system with 2 states to show all steps of the design procedure explicitly. Consider the system*

$$\begin{aligned}\begin{pmatrix}\dot{x}_1\\ \dot{x}_2\end{pmatrix} &= \begin{pmatrix}2x_2\\ -x_1+0.2x_1x_2\end{pmatrix} + \begin{pmatrix}0\\1\end{pmatrix} u,\\ y &= 1+2x_1.\end{aligned}$$

The system belongs to the class of affine MTI systems (4.21) and (4.22), where the factor matrices are represented as CP tensors by

$$\begin{aligned}\mathsf{A} &= \left[\begin{pmatrix}0&1&0\\1&0&1\end{pmatrix}, \begin{pmatrix}1&0&0\\0&1&1\end{pmatrix}, \begin{pmatrix}1&0&0\\0&1&1\end{pmatrix}\right] \cdot \begin{pmatrix}2\\-1\\0.2\end{pmatrix},\\ \mathsf{B} &= \left[\begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}\right] \cdot 1,\\ \mathsf{C} &= \left[\begin{pmatrix}1&1\\0&0\end{pmatrix}, \begin{pmatrix}0&1\\1&0\end{pmatrix}\right] \cdot \begin{pmatrix}2\\1\end{pmatrix}.\end{aligned}$$

To design the controller, the Lie derivatives of the system must be computed. Using (4.24) the parameter tensors of the Lie derivatives along $\mathbf{a}(\mathbf{x})$ *in CP representation are given by*

$$\begin{aligned}\mathsf{L}_{\mathsf{A},\mathsf{C},1} &= \left[\begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}\right] \cdot 4,\\ \mathsf{L}_{\mathsf{A},\mathsf{C},2} &= \left[\begin{pmatrix}1&0\\0&1\end{pmatrix}, \begin{pmatrix}0&0\\1&1\end{pmatrix}, \begin{pmatrix}1&1\\0&0\end{pmatrix}, \begin{pmatrix}1&1\\0&0\end{pmatrix}, \begin{pmatrix}1&1\\0&0\end{pmatrix}, \begin{pmatrix}1&1\\0&0\end{pmatrix}\right] \cdot \begin{pmatrix}-4\\0.8\end{pmatrix},\end{aligned}$$

resulting in the derivatives

$$\begin{aligned}L_{\mathbf{a}}c(\mathbf{x}) &= \left\langle \mathsf{L}_{\mathsf{A},\mathsf{C},1} \,\middle|\, \mathsf{M}_p^2(x_1,x_2) \right\rangle = 4x_2,\\ L_{\mathbf{a}}^2c(\mathbf{x}) &= \left\langle \mathsf{L}_{\mathsf{A},\mathsf{C},2} \,\middle|\, \mathsf{M}_p^3(x_1,x_2) \right\rangle = -4x_1+0.8x_1x_2.\end{aligned}$$

The Lie derivatives along $\mathbf{b}(\mathbf{x})$ *leads to* $L_{\mathbf{b}}c(\mathbf{x}) = 0$ *and* $L_{\mathbf{b}}L_{\mathbf{a}}c(\mathbf{x}) = 4$ *by the parameter tensors (4.25)*

$$\begin{aligned}\mathsf{L}_{\mathsf{B},\mathsf{A},\mathsf{C},0} &= \left[\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}0\\0\end{pmatrix}\right] \cdot 0,\\ \mathsf{L}_{\mathsf{B},\mathsf{A},\mathsf{C},1} &= \left[\begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}\right] \cdot 4.\end{aligned}$$

Because of $L_{\mathbf{b}}L_{\mathbf{a}}c(\mathbf{x}) = 4 \neq 0\ \forall\mathbf{x}$ *the relative degree* ρ_{sys} *of the system is two, which is equal to the number of states. Thus, the system has full degree and the zero dynamics do not need*

to be checked. Additionally the system is controllable and observable. In closed loop, the linear behavior

$$\mu_2 \ddot{y} + \mu_1 \dot{y} + \mu_0 y = \mu_0 r$$

is desired. As an example $\mu_1 = \mu_0 = 10$ *and* $\mu_2 = 1$ *are chosen. Using the parameter tensors of the Lie derivatives, the feedback linearizing controller (4.27) is given by*

$$u = \frac{-\left\langle \mu_0 \mathsf{C} + \mu_1 \mathsf{L}_{\mathsf{A},\mathsf{C},1} + \mu_2 \mathsf{L}_{\mathsf{A},\mathsf{C},2} \,\middle|\, \mathsf{M}_p^3(\mathbf{x}) \right\rangle + \mu_0 \, r}{\left\langle \mathsf{L}_{\mathsf{B},\mathsf{A},\mathsf{C},1} \,\middle|\, \mathsf{M}_p^3(\mathbf{x}) \right\rangle} = \frac{-10 - 16x_1 - 40x_2 - 0.8x_1x_2 + 10r}{4}.$$

Figure 4.7 shows the closed loop simulation result with an initial value $\mathbf{x}_0 = \begin{pmatrix} -0.5 & 0 \end{pmatrix}^T$ *for a step change in reference signal* r*. The system behaves as the specified* 2^{nd} *order linear system.*

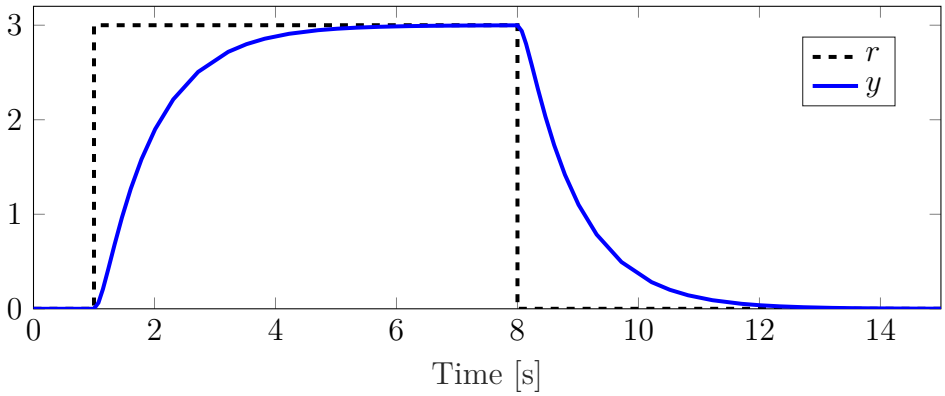


Figure 4.7: Closed loop simulation

Using low-rank approximation techniques for parameter tensors, makes it possible to represent large systems as shown in Section 3.2 and to compute controllers as shown in this section. The proposed feedback linearization approach works with the decomposed representations of the system and all operations are defined for decomposed tensors such that no full tensor representation has to be constructed during the whole controller design process leading to a controller in a decomposed structure. The storage demand is still computationally manageable also for large systems as shown in Figure 4.8, where an upper bound for the number of elements to be stored for the parameter tensors of the controller are depicted, that are represented in CP decomposition. The figure considers different orders and different ranks of the system tensors A, B and C for a system with relative degree of one. Often the results can be represented by tensors of lower rank, which reduces the storage effort further.

During application the controller law is evaluated based on the decomposed factor matrices. This makes it possible to use this feedback linearizing scheme also for large-scale MTI systems, where the full representations cannot be processed but low-rank approximations exist. This possible handling of the dimensionality problem for large-scale systems is the big advantage of the introduced feedback linearization design approach by decomposed tensors.

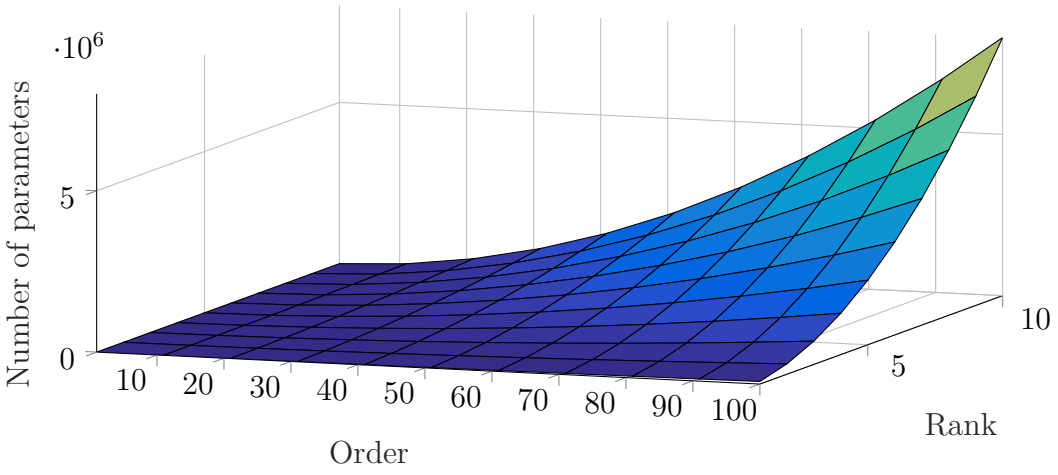


Figure 4.8: Upper bound for number of parameters for a feedback linearizing controller of decomposed MTI systems

4.3 Decentralized state feedback design

The decentralized state feedback design method for linear models introduced by [66] and described in Section 4.1.1 is applied to an MTI system here, see [48].

Control problem

A decentralized feedback gain should be found for an MTI system with a disturbance input $\mathbf{d} \in \mathbb{R}^{m_d}$

$$\dot{\mathbf{x}} = \langle \mathsf{F} \mid \mathsf{M}(\mathbf{x}, \mathbf{u}, \mathbf{d}) \rangle, \tag{4.28}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x}, \tag{4.29}$$

$$z = \mathbf{C}_z\mathbf{x} + \mathbf{D}_z\mathbf{u}. \tag{4.30}$$

The state equation (4.28) is in MTI tensor form. The measured output and the performance output z are the same as in (4.2) and in (4.3) with weighting matrices $\mathbf{Q}$ and $\mathbf{R}$. It is assumed, that all states are measured for the feedback loop. The outputs $\mathbf{y}$ should be driven to a given, feasible reference $\mathbf{r} \in \mathbb{R}^p$ by the controller. The matrix $\mathbf{C} \in \mathbb{R}^{p \times n}$ chooses the states (4.29), that should track a reference value. The controller is computed for a linearization of (4.28) around an operating point $(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}})$. If the reference or the disturbance changes, this results in a change in the operating point. Thus, a new linearization has to be computed and the control law should be updated according to the current operating conditions.

For the application of the sparse LQR method, the MTI system has to be linearized around an operating point. The operating point is determined by the reference values $\mathbf{r}$ and the actual disturbance $\mathbf{d}$. The operating point $(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}})$ is computed by setting the right hand side of (4.28) to zero

$$\left\langle \mathsf{F} \,\middle|\, \mathsf{M}\left(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}}\right) \right\rangle = 0. \tag{4.31}$$

To consider the reference value, the disturbance and input constraints, the equation (4.31) has to be solved with constraints on $(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}})$. The operating point of states is computed, such that the corresponding output is equal to the current reference

$$\mathbf{C}\bar{\mathbf{x}} = \mathbf{r}. \tag{4.32}$$

To take into account the current value $\mathbf{d}(t)$ of the disturbance, the operating point of the disturbance is set to

$$\bar{\mathbf{d}} = \mathbf{d}(t). \tag{4.33}$$

In many applications the inputs are saturated, such that the operating point should satisfy

$$\mathbf{u}_{min} \leq \bar{\mathbf{u}} \leq \mathbf{u}_{max}. \tag{4.34}$$

Thus, the system of multilinear equations (4.31) has to be solved for the variables $\mathbf{x}$, $\mathbf{u}$ and $\mathbf{d}$ with equality and inequality constraints (4.32) to (4.34). Using this operating point $(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}})$, the linearization of the multilinear system (4.28) can be computed based on the parameter tensors as introduced in Section 3.3. This results in the operating point depended description of the system matrix

$$\mathbf{A}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}}) = \left\langle \mathsf{A}_{lin} \,\middle|\, \mathsf{M}\left(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}}\right) \right\rangle. \tag{4.35}$$

The inputs are divided into control and disturbance inputs, such that the parameter tensor $\mathsf{A}_{lin} \in \mathbb{R}^{\times^{(n+m+m_d)}2\times n\times n}$ is build up by fibers

$$\mathbf{a}_{lin}(\mathbf{i}_d, \mathbf{i}_u, \mathbf{i}_x, :, j) = \mathbf{j}_{\mathbf{f}}(\mathbf{i}_d, \mathbf{i}_u, \mathbf{i}_x, :, j) = \mathbf{f}_{x_j}(\mathbf{i}_d, \mathbf{i}_u, \mathbf{i}_x, :),\ j = 1, \ldots, n,$$

with index vectors $\mathbf{i}_x$, $\mathbf{i}_u$, $\mathbf{i}_d$ for state, control and disturbance inputs and the Jacobian matrix $\mathbf{J}_{\mathbf{f}}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}}) = \left\langle \mathsf{J}_{\mathbf{f}} \,\middle|\, \mathsf{M}\left(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}}\right) \right\rangle$ with parameter tensor $\mathsf{J}_{\mathbf{f}} \in \mathbb{R}^{\times^{(n+m+m_d)}2\times n\times(n+m+m_d)}$ of (4.28). The overall input matrix results from

$$\mathbf{B}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}}) = \left\langle \mathsf{B}_{lin} \,\middle|\, \mathsf{M}\left(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}}\right) \right\rangle. \tag{4.36}$$

The parameter tensor $\mathsf{B}_{lin} \in \mathbb{R}^{\times^{(n+m+m_d)}2\times n\times(m+m_d)}$ follows from (3.43) under consideration of the separation of the inputs to control and disturbance signals by

$$\begin{aligned}
\mathbf{b}_{lin}(\mathbf{i}_d, \mathbf{i}_u, \mathbf{i}_x, :, j) &= \mathbf{j}_{\mathbf{f}}(\mathbf{i}_d, \mathbf{i}_u, \mathbf{i}_x, :, n+j) = \mathbf{f}_{u_j}(\mathbf{i}_d, \mathbf{i}_u, \mathbf{i}_x, :),\ j = 1, \ldots, m, \\
\mathbf{b}_{lin}(\mathbf{i}_d, \mathbf{i}_u, \mathbf{i}_x, :, m+j) &= \mathbf{j}_{\mathbf{f}}(\mathbf{i}_d, \mathbf{i}_u, \mathbf{i}_x, :, n+m+j) = \mathbf{f}_{d_j}(\mathbf{i}_d, \mathbf{i}_u, \mathbf{i}_x, :),\ j = 1, \ldots, m_d,
\end{aligned}$$

where F_{u_j} and F_{d_j} are parameter tensors of the derivatives of the right hand side of the state equation (4.28) with respect to u_j and d_j, respectively. The parameter tensors A_{lin} and B_{lin} are computed by the partial derivatives of the state equation of the MTI system with respect to x_j, u_j and d_j by operational tensors from simple matrix tensor products of the parameter tensor F as introduced in Section 2.3.3. The input matrix is split to get the input and disturbance related parts

$$\mathbf{B}_u(:, i) = \mathbf{B}(:, i),\ i = 1, \ldots, m, \tag{4.37}$$

$$\mathbf{B}_d(:, i) = \mathbf{B}(:, m+i),\ i = 1, \ldots, m_d. \tag{4.38}$$

To design the controller the matrices $\mathbf{Q}$ and $\mathbf{R}$ have to be tuned. The weighting $\mathbf{Q}$ of the states is chosen initially as

$$\mathbf{Q} = \mathbf{C}^T\mathbf{C},$$

such that the outputs $\mathbf{y}$, which should reach its reference value are weighted. The control effort has to be penalized by $\mathbf{R}$ such that the signals do not exceed the input constraints. With the system matrices $\mathbf{A}$, $\mathbf{B}_u$, $\mathbf{B}_d$ and the weighting matrices $\mathbf{Q}$ and $\mathbf{R}$ the decentralized controller is computed by solving (4.8) for a given value of γ.

Preprocessing

The structure of the feedback controller should be fixed during the operation, i.e. the communication structure should not change permanently. Thus, before the operation a structure analysis is needed. This is done in the preprocessing step. A sparsity structure of $\mathbf{K}$ should be found such that the performance loss compared to the central controller (4.5) is low. The performance degradation is measured by

$$\Delta J = \frac{|J_d - J_c|}{J_c} \cdot 100\%, \tag{4.39}$$

which compares the performance of the decentralized controller J_d with the performance of the central one J_c. The central performance J_c is the value of the objective function (4.6) for the solution of the central feedback design (4.5). The performance factor J_d of the decentralized design is the evaluation of the same cost function (4.6) evaluated with the result of the optimization problem (4.7) with sparsity structure constraint, where the structure was determined by solving (4.8). In general the performance loss increases when the controller structure gets more sparse, as shown by an example in Figure 4.9. When γ is increased the performance loss increases too, because the focus is more and more on sparsity and not controller performance in the optimization problem (4.8).

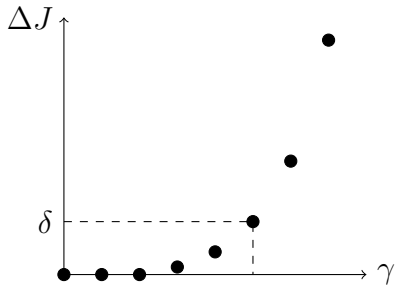


Figure 4.9: Performance loss depending on tuning factor γ

A structure with minimal communication effort should be found here, such that the performance loss is still acceptable $\Delta J < \delta$, i.e. below a threshold δ, as depicted in Figure 4.9. The value of γ belonging to this performance degradation has to be determined as described in the following. Since not a linear but a multilinear system is considered here, the performance

constraint of the structure of minimal complexity must hold for every operating point in the operation area given by the application.
To determine the structure, the optimization problem (4.8) has to be solved. The structure cannot be set directly but is influenced by changing the sparsity promoting weighting factor γ. The choice of γ results in a certain structure, where the controller complexity decreases for an increasing γ. Thus, to find a structure, that fulfills the performance constraint for all operating points, a heuristic approach is chosen to iterate over the factor γ for a number of feasible operating points. The sparsity structure of $\mathbf{K}$ with minimal communication effort and a performance loss smaller than δ for all investigated operating points is chosen as controller structure $\mathcal{S}$. This heuristic approach does not find the global optimum of the multicriteria optimization, but it approximates the solution by a lower bound on the performance loss. For applications where the structure strongly depends on the operating point, the structure changes with different operating points for a constant value of γ. In this case the communication effort cannot be reduced that much since an overall controller structure has to be chosen that satisfies all operating points. But application shows that, e.g. for a heating system example in Section 5.3.2, the tuning factor γ is the main influencing factor on the structure. The preprocessing step can be summarized in the flow chart depicted in Figure 4.10.

Operation

During the operation of the state feedback controller the determined structure $\mathcal{S}$ should not be changed anymore. The controller gain $\mathbf{K}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}})$ depends on the operating point and is valid in a neighborhood around the operating point only. If the state reference $\mathbf{r}$ changes, the operating point changes too. A new operating point with the actual reference has to be computed by (4.31). Also a change in the disturbance signal leads to an update of the operating point. A new operating point with new value for $\bar{\mathbf{d}}$ is computed, if the disturbance signal changes more than $\Delta\mathbf{d} \in \mathbb{R}^{m_d}$ which is a tuning parameter

$$\bar{d}_i = \begin{cases} d_i(t) & \text{, if } \left| d_i(t) - \bar{d}_i \right| > \Delta d_i, \\ \bar{d}_i & \text{, else,} \end{cases}$$

with $i = 1, \ldots, m_d$. With the value $\Delta\mathbf{d}$ the rate of the computation of new linearizations can be influenced. A small value of $\Delta\mathbf{d}$ leads to a frequent change in the operating point, i.e. the operating point is adapted very often to the current ambient conditions. But it results in a larger computation effort. A large value of $\Delta\mathbf{d}$ decreases the rate of adaptions to the operating point. The choice of the factor depends on the application and the nonlinearities of the plant with respect to the disturbances. With a high sensitivity the operating point, i.e. the linear approximation and thus the controller gain should be updated with a smaller value of $\Delta\mathbf{d}$. If the sensitivity is not that large, $\Delta\mathbf{d}$ can be chosen larger, leading to a less computational demanding controller since the feedback gain will not be updated that often.

With a new operating point the linearization of the MTI system is determined by (4.35) and (4.36). The linearization can be computed very efficiently by evaluating one contracted product only, which is very simple, if the tensors are available in a decomposed structure as it is the case here. This leads to a gain update by solving (4.7) with the structure $\mathcal{S}$

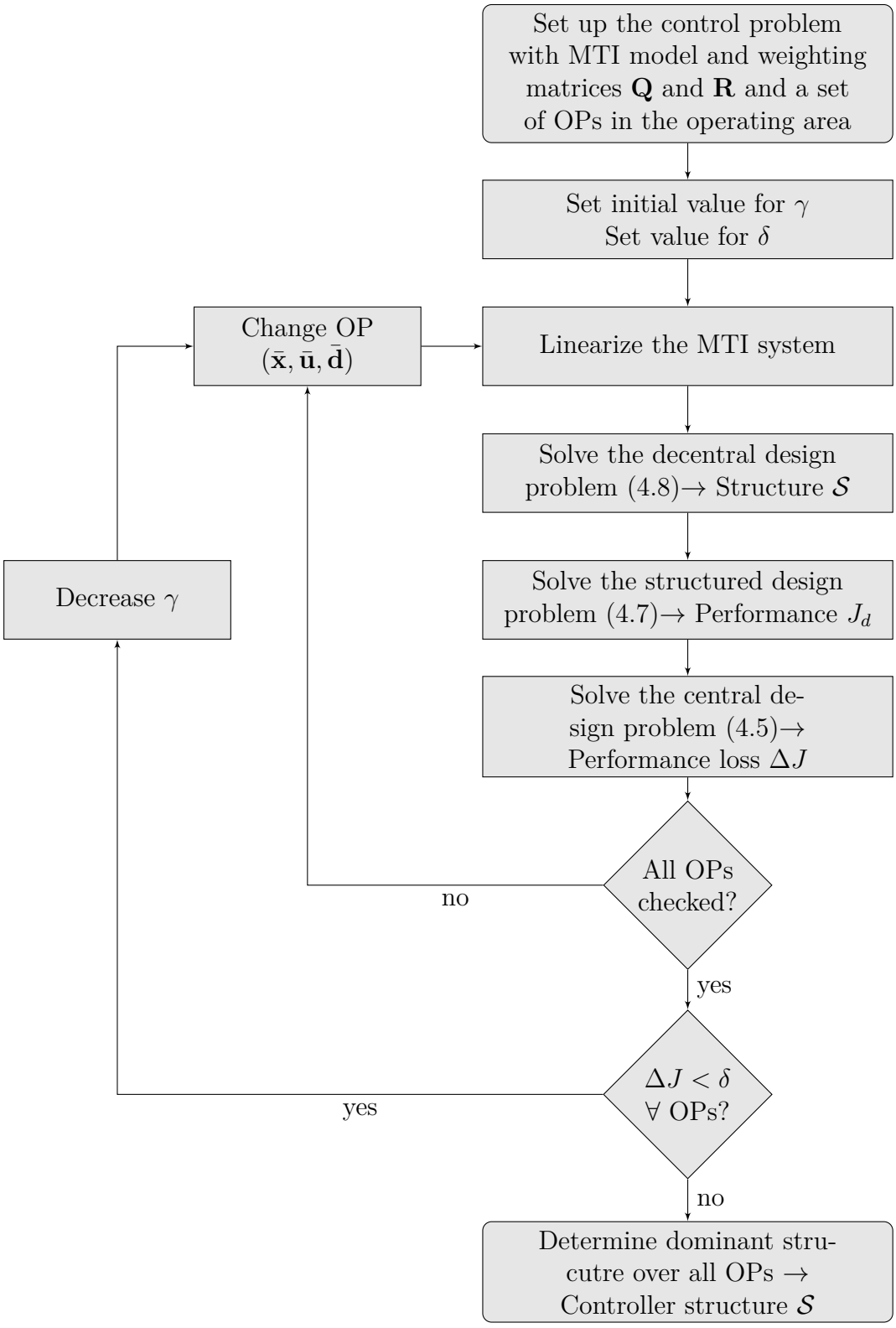


Figure 4.10: Flow chart of the preprocessing step

determined in the preprocessing step. Having computed the feedback gain $\mathbf{K}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}})$ as a function of the operating point, the input

$$\mathbf{u} = -\mathbf{K}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}})(\mathbf{x} - \bar{\mathbf{x}}) + \bar{\mathbf{u}}, \tag{4.40}$$

is given to the plant. At each sampling instant,the controller computes the steps summarized in the flow chart depicted in Figure 4.11.

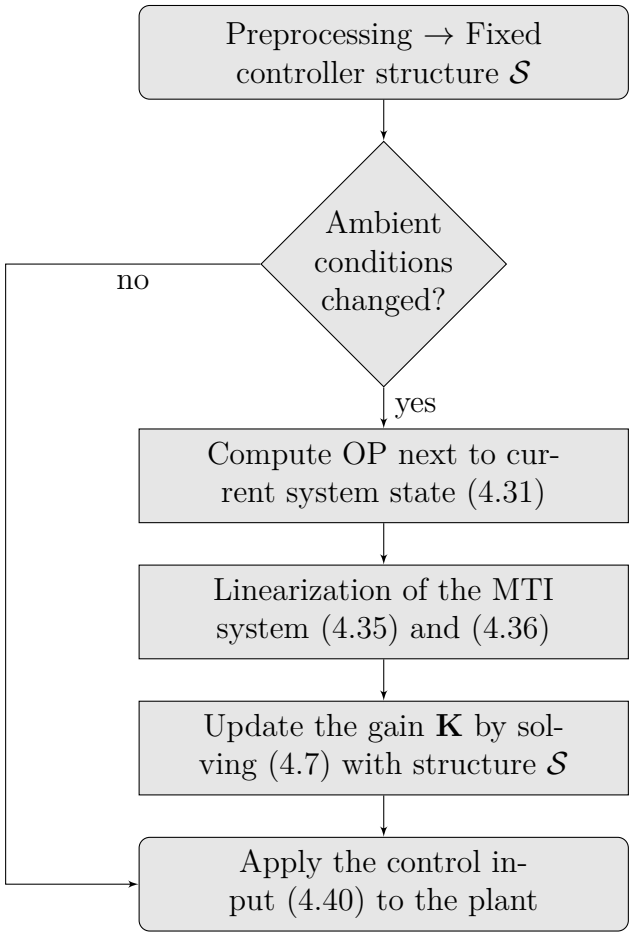


Figure 4.11: Flow chart of the decentralized feedback controller during operation

The closed loop with sparse controller and the MTI system is depicted in Figure 4.12.

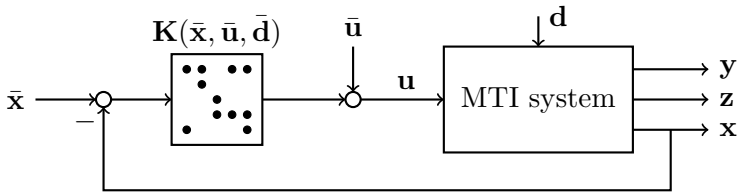


Figure 4.12: Decentralized state feedback with sparse gain $\mathbf{K}$

Example 4.3 *As an example an MTI system with six states, three controlled inputs and one disturbance input with state equations*

$$\begin{aligned}
\dot{x}_1 &= x_1 + u_1, \\
\dot{x}_2 &= x_1 - x_2 d, \\
\dot{x}_3 &= x_2 + u_2, \\
\dot{x}_4 &= x_3 - x_4 d, \\
\dot{x}_5 &= x_4 + u_3, \\
\dot{x}_6 &= x_5 - x_6 d,
\end{aligned}$$

is considered, that can be represented by (4.28). It is assumed that all states are measured. Figure 4.13 shows how the system can be represented as a serial connection of 3 subsystems. This representation will help to verify the decentralized controller design.

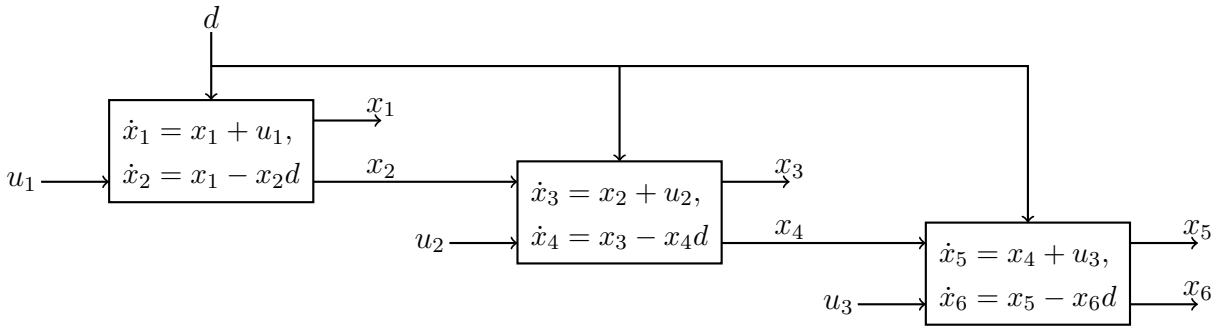


Figure 4.13: System representation as serial connection

The controller should drive the states x_1, x_3 and x_5 to desired references r_1, r_3 and r_5, respectively, which leads to output matrix

$$\mathbf{C} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix},$$

of (4.29). According to this, the weighting matrix for the state performance is chosen as

$$\mathbf{Q} = \mathbf{C}^T\mathbf{C} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

The input performance should be weighted the same here by

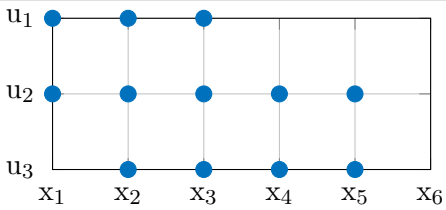
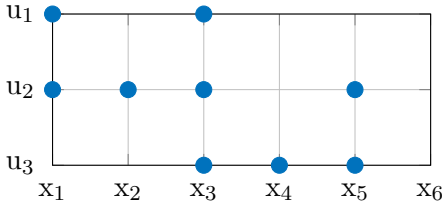
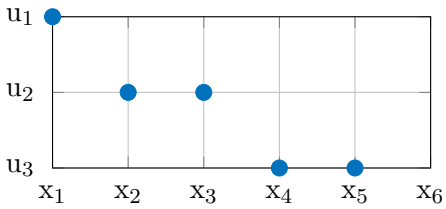
$$\mathbf{R} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

The overall system is now used in the preprocessing step to find a suitable sparsity structure for the introduced controller configuration. To analyze the system structure, operating points are determined for combinations of reference r_i and disturbance d values in the operating areas

$$r_i \in \{1, 2, 3, 4\}\,, \;\; i = 1, 3, 5,$$
$$d \in \{1, 2, 3\}\,.$$

With these specifications of references and disturbances the corresponding operating points are computed by (4.31) to (4.34). No constraints on the inputs are assumed here. For each operating point the MTI system is linearized and the decentralized state feedback gain is computed for this linearizations with different values of γ in the interval $[10^{-5},\ 10^{-1}]$ by solving the sparsity promoting optimization problem (4.8). For this operating interval and the interval of γ, three sparsity structures are determined with different sparsity degree. The performance losses are determined by (4.39). The three structures are summarized in Table 4.1, where the nonzero values are highlighted in the sparsity plot. The values of γ, the percentage of zeros and the maximal performance loss over all operating points are also shown.

Table 4.1: Controller structures with different degree of sparsity

Structure	γ	**Max. ΔJ**	**Sparsity**
	$3 \cdot 10^{-5}$	$2 \cdot 10^{-3}\%$	33%
	$1 \cdot 10^{-3}$	0.17%	50%
	$1 \cdot 10^{-2}$	0.85%	72%

The table shows, that with increasing value of γ the structure gets more and more sparse. For all three structures the performance loss is acceptably low even for the third controller of high sparsity, where the lowest number of communication links is necessary. To verify the results of the decentralized controller design process, the structure of the system and the

control problem is investigated. The states x_1, x_3 and x_5 should follow its references and are influenced by the inputs u_1 to u_3 directly. Because of that, the inputs need the information on the respective states, e.g. u_1 needs the information on x_1, as can be seen in the first row of the gain matrices. In addition to that, the states x_3 and x_5 are influenced by the state x_2 and x_4 of the neighboring subsystems, such that communication links between these states and u_2 and u_3 have to be established too. This gives the minimal set of communication channels in the last row of Table 4.1.
A closer look on the example system shows, that e.g. the state x_3 can be influenced by input u_1 via the state x_2. Because of that, the performance of the controller can be improved by closing the communication link between u_1 and x_3 and between u_2 and the states of the first system. The same dependencies hold for the connections between the second on the third system, which explains the controller structure of medium sparsity. With the controller of low sparsity even more information is exchanged, but still some communication links are cut. For example, the first subsystem does not need the full system information of the third system and vice versa in this configuration. The sixth state is not influenced by any reference tracking variable, such that this state is not considered in all controller structures.

Since the performance loss of the sparsest controller is still acceptable, this structure should be used for a simulation of the system. The system with the controller structure that results from this sparsity pattern is illustrated in Figure 4.14, which shows that with a decentralized controller only local information on the states is used and not the whole system information.

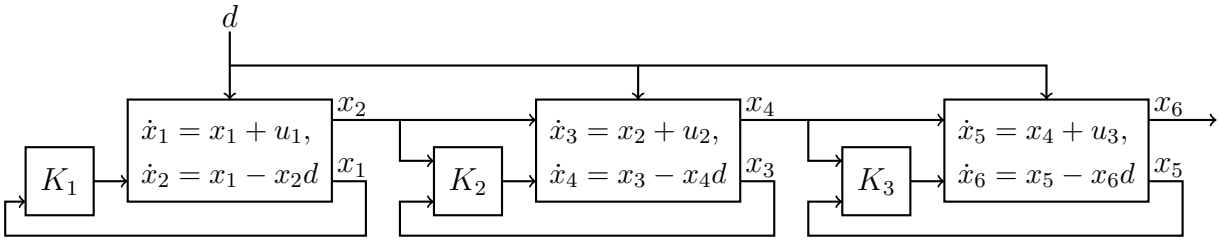


Figure 4.14: Example system with decentralized controller structure

To simulate the closed loop the reaction on step changes in each reference is investigated. In addition to that, the simulation scenario contains a step change in the disturbance signal. The reference and disturbance signal are depicted in Figure 4.15, together with the closed loop simulation results of the states x_1, x_3 and x_5 with a central and the decentralized controller. During simulation the structure of the controller determined in the preprocessing phase is fixed. During operation each change in the reference and disturbance leads to a change in the operating point by computing (4.31) and thus an update of the controller gain with the linearizations (4.35) and (4.36) around the new operating point by solving the structured optimization problem (4.7). The simulation shows that the decentralized controller has a good tracking and disturbance rejection behavior that is comparable to the central design but with 5 established communication links only instead of 18 in the general central case. Thus, the communication effort can be reduced without significant loss in controller performance.

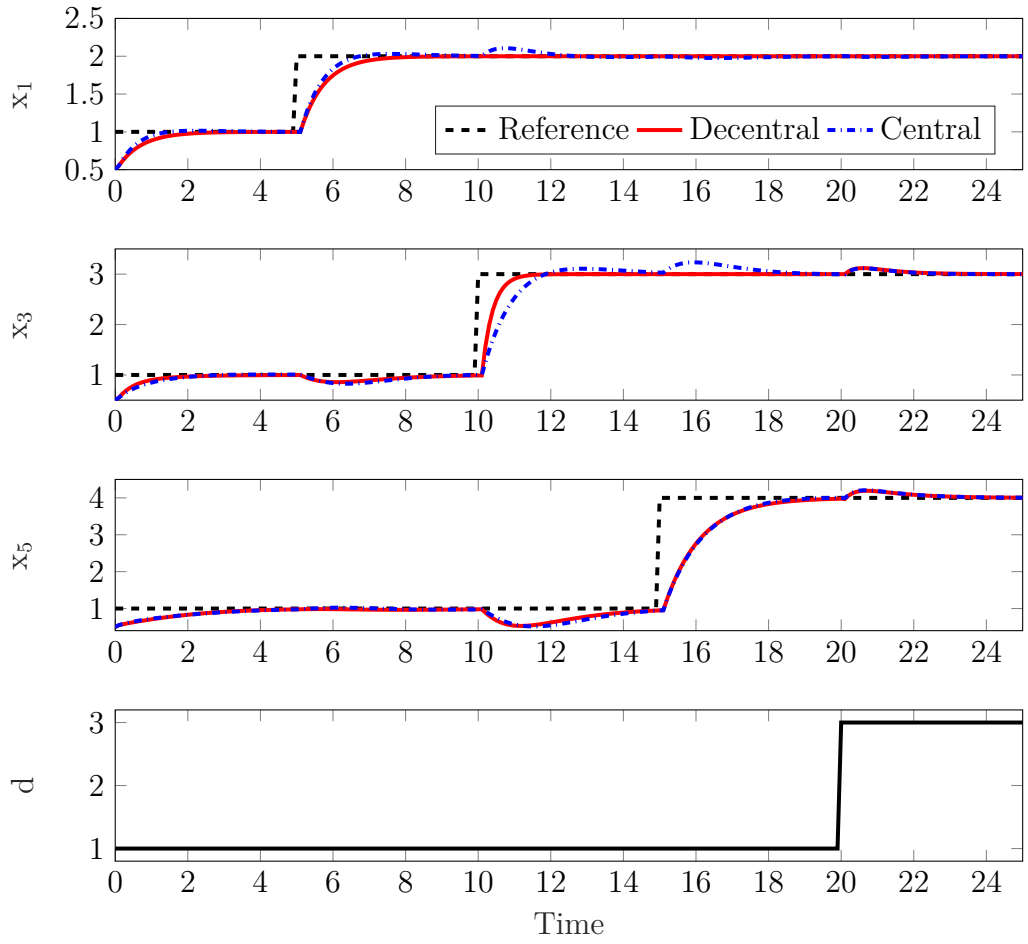


Figure 4.15: Closed loop simulation results of the example system with central and decentralized controller

4.4 Model predictive control

In the standard formulation in the literature, e.g. in [73], MPC is well established for linear systems with a quadratic cost function as described in Section 4.1.3. Inserting the lifted system (3.72) to (3.78) into the cost function (4.19) leads to a closed description of the cost function and gives a quadratic programming optimization problem in the linear case. The following subsections investigate the MPC optimization problem with a quadratic cost function, when MTI models are used. At first a closed description of the cost function is derived in Subsection 4.4.1 by applying the multi-step transition of MTI systems of Section 3.6. For linear systems the MPC optimization problem is a quadratic problem and thus convex, if the constraints are convex too, [12]. This is an important property because with a convex problem a unique global optimum is found and standard algorithms like the interior point method are available, [12]. The convexity of the optimization problem for MTI systems is analyzed in Subsection 4.4.2. It is investigated, if the convexity holds for the whole class of

MTI systems or if conditions can be derived for a subclass of MTI systems that lead to a convex optimization problem.

4.4.1 Optimization problem for MTI systems

In this section the optimization problem of NMPC is investigated, when the system dynamics are described by a discrete-time MTI state space model

$$\mathbf{x}(k+1) = \langle\, \mathsf{F} \mid \mathsf{M}\left(\mathbf{x}(k), \mathbf{u}(k)\right) \rangle\,,$$

that is derived directly or follows from discretization from a continuous-time MTI model, e.g. by the approach introduced in Subsection 3.4. For the design of an MPC controller with an MTI model a quadratic cost function is used

$$J(k) = \sum_{i=1}^{H_p} \|\hat{\mathbf{x}}(k+i) - \mathbf{r}(k+i)\|_{\mathbf{Q}}^2 + \sum_{i=0}^{H_u-1} \|\Delta\mathbf{u}(k+i)\|_{\mathbf{R}}^2. \tag{4.41}$$

For ease of notation it is assumed, that all states are measured $\mathbf{y} = \mathbf{x}$ and that the control horizon is equal to the prediction horizon $H_p = H_u$. To evaluate the cost function, the states $\hat{\mathbf{x}}(k+i),\ i = 1, \ldots, H_p$ have to be computed for the prediction horizon. When using an MTI model the state trajectory is available from the multi-step transition (3.89) derived in Section 3.6. To get a closed expression for the cost function the parameter tensors of all time instants $k+i,\ i = 1, \ldots, H_p$ are extended by zeros, such that the monomial vector describes a dependence on the inputs $\mathbf{u}(k+i),\ i = 0, \ldots, H_p - 1$ and reference values $\mathbf{r}(k+i),\ i = 1, \ldots, H_p$ for the whole prediction horizon

$$\hat{\mathbf{x}}(k+i) = \left\langle \mathsf{F}^i \,\middle|\, \mathsf{M}_p^{(n+m)^{i-1}}\left(\mathbf{x}(k), \mathbf{u}(k-1), \ldots, \mathbf{u}(k+H_p-1), \mathbf{r}(k+1), \ldots, \mathbf{r}(k+H_p)\right)\right\rangle, \tag{4.42}$$

with $\mathsf{F}^i \in \mathbb{R}^{\times^{(n+m)^{i-1}((H_p+3)n+H_pm)}2}$. The reference trajectory for the prediction horizon is known and can be expressed with respect to the same monomial tensor by constructing a parameter tensor F_r^i such that

$$\mathbf{r}(k+i) = \left\langle \mathsf{F}_r^i \,\middle|\, \mathsf{M}_p^{(n+m)^{i-1}}\left(\mathbf{x}(k), \mathbf{u}(k-1), \ldots, \mathbf{u}(k+H_p-1), \mathbf{r}(k+1), \ldots, \mathbf{r}(k+H_p)\right)\right\rangle, \tag{4.43}$$

for all $i \in \{1, \ldots, H_p\}$. All entries in the tensor F_r^i belonging to the states and inputs are set to zeros. Only the linear terms of the references are selected. The first term of the cost function (4.41) $\sum_{i=1}^{H_p} \|\hat{\mathbf{x}}(k+i) - \mathbf{r}(k+i)\|_{\mathbf{Q}}^2$ penalizes the deviation of the states from their reference values. The difference between states and their references is computed in the tensor framework with above definitions (4.42) and (4.43) by

$$\begin{aligned}&\hat{\mathbf{x}}(k+i) - \mathbf{r}(k+i)\\ &= \left\langle\, \mathsf{F}^i - \mathsf{F}_r^i \,\middle|\, \mathsf{M}_p^{(n+m)^{i-1}}\left(\mathbf{x}(k), \mathbf{u}(k-1), \ldots, \mathbf{u}(k+H_p-1), \mathbf{r}(k+1), \ldots, \mathbf{r}(k+H_p)\right)\,\right\rangle.\end{aligned}$$

The deviation of states and references is computed by a weighted 2-norm

$$\begin{aligned}&\|\hat{\mathbf{x}}(k+i)-\mathbf{r}(k+i)\|_{\mathbf{Q}}^2\\&=\left\langle \mathsf{F}^i-\mathsf{F}_r^i\left|\mathsf{M}_p^{(n+m)^{i-1}}\left(\mathbf{x}(k),\ldots,\mathbf{r}(k+H_p)\right)\right.\right\rangle^T\mathbf{Q}\left\langle \mathsf{F}^i-\mathsf{F}_r^i\left|\mathsf{M}_p^{(n+m)^{i-1}}\left(\mathbf{x}(k),\ldots,\mathbf{r}(k+H_p)\right)\right.\right\rangle.\end{aligned}$$

In many applications a diagonal weighting matrix

$$\mathbf{Q}=\begin{pmatrix}q(1)&0&\cdots&0\\0&q(2)&\cdots&0\\\vdots&\vdots&\ddots&\vdots\\0&\cdots&0&q(n)\end{pmatrix}\in\mathbb{R}^{n\times n},$$

is used, which is assumed here, too. The norm of the state difference of the reference is given by the particular states

$$\begin{aligned}&\|\hat{\mathbf{x}}(k+i)-\mathbf{r}(k+i)\|_{\mathbf{Q}}^2=\sum_{j=1}^{n}q(j)\left(\hat{x}_j(k+i)-r_j(k+i)\right)^2\\&=\sum_{j=1}^{n}q(j)\left\langle\left(\mathsf{F}^i-\mathsf{F}_r^i\right)\bar{\times}_\beta\mathbf{e}_{j,n}\left|\mathsf{M}_p^{(n+m)^{i-1}}\left(\mathbf{x}(k),\mathbf{u}(k-1),\ldots,\mathbf{r}(k+H_p)\right)\right.\right\rangle^2\\&=\sum_{j=1}^{n}q(j)\left\langle\left(\left(\mathsf{F}^i-\mathsf{F}_r^i\right)\bar{\times}_\beta\mathbf{e}_{j,n}\right)\circ\left(\left(\mathsf{F}^i-\mathsf{F}_r^i\right)\bar{\times}_\beta\mathbf{e}_{j,n}\right)\left|\mathsf{M}_p^{2(n+m)^{i-1}}\left(\mathbf{x}(k),\ldots,\mathbf{r}(k+H_p)\right)\right.\right\rangle\\&=\left\langle\sum_{j=1}^{n}q(j)\left(\left(\mathsf{F}^i-\mathsf{F}_r^i\right)\bar{\times}_\beta\mathbf{e}_{j,n}\right)\circ\left(\left(\mathsf{F}^i-\mathsf{F}_r^i\right)\bar{\times}_\beta\mathbf{e}_{j,n}\right)\left|\mathsf{M}_p^{2(n+m)^{i-1}}\left(\mathbf{x}(k),\ldots,\mathbf{r}(k+H_p)\right)\right.\right\rangle\\&=\left\langle\mathsf{F}_{cost,x}^i\left|\mathsf{M}_p^{2(n+m)^{i-1}}\left(\mathbf{x}(k),\mathbf{u}(k-1),\ldots,\mathbf{r}(k+H_p)\right)\right.\right\rangle,\end{aligned}\tag{4.44}$$

where $\beta=(n+m)^{i-1}(H_p+1)(n+m)+1$ is the index of the last dimension of F^i or F_r^i. The tensor vector product with the unit vector is used to select the particular states. This describes the first sum of the quadratic cost function (4.41) in a tensor framework. It results in a polynomial of maximal order $2(n+m)^{i-1}$. The second part of the cost function $\sum_{i=0}^{H_u-1}\|\Delta\mathbf{u}(k+i)\|_{\mathbf{R}}^2$ is expressed in a tensor framework too. With $\Delta\mathbf{u}(k+i)$ the change of the input signals from one time step to the next one is computed. A parameter tensor is simply constructed that selects the corresponding input from the monomial tensor and computes the difference

$$\Delta\mathbf{u}(k+i)=\mathbf{u}(k+i)-\mathbf{u}(k+i-1)=\left\langle\mathsf{F}_{\Delta\mathbf{u}}^i\left|\mathsf{M}_p^{(n+m)^{i-1}}\left(\mathbf{x}(k),\mathbf{u}(k-1),\ldots,\mathbf{r}(k+H_p)\right)\right.\right\rangle.$$

Example 4.4 *Assume the input difference $\Delta u(k+1)=u(k+1)-u(k)$ should be computed from a monomial tensor $\mathsf{M}\left(u(k),u(k+1)\right)$. The parameter tensor is constructed, such that*

$$\Delta u(k+1)=\left\langle\begin{pmatrix}0&-1\\1&0\end{pmatrix}\left|\begin{pmatrix}1&u(k)\\u(k+1)&u(k)u(k+1)\end{pmatrix}\right.\right\rangle=u(k+1)-u(k).$$

The example shows, that the parameter tensor $\mathsf{F}^i_{\Delta\mathbf{u}}$ is simply constructed. This is done in a similar way for higher order monomial tensors. Again it is assumed here as it is the case often in the applications, that the weighting matrix $\mathbf{R}$ of the inputs is a diagonal matrix

$$\mathbf{R} = \begin{pmatrix} r(1) & 0 & \cdots & 0 \\ 0 & r(2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & r(m) \end{pmatrix} \in \mathbb{R}^{m\times m}.$$

Using the tensor representation of the input difference, the second part of the quadratic MPC cost function is derived analogous to the first part by

$$\begin{aligned}
&\|\Delta\mathbf{u}(k+i)\|^2_{\mathbf{R}} = \sum_{j=1}^{m} r(j)\Delta u_j(k+i)^2 \\
&= \sum_{j=1}^{m} r(j) \left\langle \mathsf{F}^i_{\Delta\mathbf{u}} \bar{\times}_\beta \mathbf{e}_{j,n} \;\middle|\; \mathsf{M}_p^{(n+m)^{i-1}} (\mathbf{x}(k), \mathbf{u}(k-1), \ldots, \mathbf{r}(k+H_p)) \right\rangle^2 \\
&= \sum_{j=1}^{m} r(j) \left\langle \left(\mathsf{F}^i_{\Delta\mathbf{u}} \bar{\times}_\beta \mathbf{e}_{j,n}\right) \circ \left(\mathsf{F}^i_{\Delta\mathbf{u}} \bar{\times}_\beta \mathbf{e}_{j,n}\right) \;\middle|\; \mathsf{M}_p^{2(n+m)^{i-1}} (\mathbf{x}(k), \mathbf{u}(k-1), \ldots, \mathbf{r}(k+H_p)) \right\rangle \\
&= \left\langle \sum_{j=1}^{m} r(j) \left(\mathsf{F}^i_{\Delta\mathbf{u}} \bar{\times}_\beta \mathbf{e}_{j,n}\right) \circ \left(\mathsf{F}^i_{\Delta\mathbf{u}} \bar{\times}_\beta \mathbf{e}_{j,n}\right) \;\middle|\; \mathsf{M}_p^{2(n+m)^{i-1}} (\mathbf{x}(k), \mathbf{u}(k-1), \ldots, \mathbf{r}(k+H_p)) \right\rangle \\
&= \left\langle \mathsf{F}^i_{cost,u} \;\middle|\; \mathsf{M}_p^{2(n+m)^{i-1}} (\mathbf{x}(k), \mathbf{u}(k-1), \ldots, \mathbf{r}(k+H_p)) \right\rangle
\end{aligned}$$

Since both parts of the cost function (4.41) are described in a tensor framework now, the cost function itself can be represented in closed form too for MTI systems. Expressing all parameter tensors $\mathsf{F}^i_{cost,x}$ and $\mathsf{F}^i_{cost,u}$ with respect to the monomial tensor

$$\mathsf{M}_p^{(n+m)^{H_p-1}} (\mathbf{x}(k), \mathbf{u}(k-1), \ldots, \mathbf{u}(k+H_p-1), \mathbf{r}(k), \ldots, \mathbf{r}(k+H_p)),$$

that contains the reference and input values for the whole prediction horizon gives

$$\begin{aligned}
J(k) &= \left\langle \sum_{i=1}^{H_p} \mathsf{F}^i_{cost,x} + \sum_{i=0}^{H_u-1} \mathsf{F}^i_{cost,u} \;\middle|\; \mathsf{M}_p^{2(n+m)^{H_p-1}} (\mathbf{x}(k), \mathbf{u}(k-1), \ldots, \mathbf{r}(k+H_p)) \right\rangle \\
&= \left\langle \mathsf{F}_J \;\middle|\; \mathsf{M}_p^{2(n+m)^{H_p-1}} (\mathbf{x}(k), \mathbf{u}(k-1), \ldots, \mathbf{r}(k+H_p)) \right\rangle
\end{aligned} \tag{4.45}$$

with the parameter tensor F_J, that describes a polynomial of maximal order $2(n+m)^{H_p-1}$ with $(H_p+1)(n+m)$ variables leading to a dimensionality of $\mathbb{R}^{\times\left(\left(2(n+m)^{H_p-1}\right)(H_p+1)(n+m)\right)_2}$. If approximate solutions are sufficient the order might be reduced by applying a Taylor approach as described in Section 2.3.5 to limit the order to 2. With this explicit description of the cost function at time k for the whole prediction horizon, no forward simulation by iterative application of the system model has to be executed as it is necessary in standard nonlinear MPC. The cost function that is minimized, when solving the problem

$$\min_{\substack{\mathbf{u}(k+i),\\ i=0,\ldots,H_u-1}} J(k), \tag{4.46}$$

is computed directly by inserting the actual state, the reference trajectory for the prediction horizon and the input trajectory for the prediction horizon in the monomial tensor. Another advantage of this closed description of the cost function is, that it is possible with (2.68) and (2.74) to calculate e.g. the gradient of the cost function analytically in the tensor framework. This leads to a faster and more accurate solution of the optimization, e.g. when using an interior point algorithm, [12]. Without an analytic solution, the gradient is determined approximately by the algorithm by a finite differences approach, leading to longer computation times and possible inaccuracies.

Example 4.5 *The plant is given as discrete-time MTI state space model with two states and one input*

$$\begin{pmatrix} x_1(k+1) \\ x_2(k+1) \end{pmatrix} = \begin{pmatrix} -0.02x_2 - 0.1x_1x_2 + 0.1x_1u, \\ -0.2 + 0.3x_1 + 0.3x_1x_2 - 0.1x_2u \end{pmatrix} = \langle\, \mathsf{F} \mid \mathsf{M}\left(x_1(k), x_2(k), u(k)\right) \rangle\,,$$

and a sampling time of $T_{sample} = 1s$ to illustrate the approach and the dimensionality of the parameter tensors. The MPC controller in this example should drive the second state to a reference value. The first state should not follow a reference. Because of that, the weighting matrix of the states is written as

$$\mathbf{Q} = \begin{pmatrix} q(1) & 0 \\ 0 & q(2) \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 100 \end{pmatrix},$$

such that the difference of the first state is weighted by $q(1) = 0$. The system has one input only and input changes are penalized with a weighting factor $r = 0.1$. A reference $\mathbf{r}(k+i)$ with $i = 1, \ldots, H_p$ is given as depicted in the simulation results in Figure 4.16 for the whole simulation interval. The prediction and the input horizon are both chosen to three time steps. At first the multiple state transition representations are computed as shown in Subsection 3.6 leading to

$$\begin{aligned}
\mathbf{x}(k+1) &= \left\langle\, \mathsf{F}^1 \mid \mathsf{M}\left(\mathbf{x}(k), u(k)\right) \right\rangle, \\
\mathbf{x}(k+2) &= \left\langle\, \mathsf{F}^2 \,\middle|\, \mathsf{M}_p^3\left(\mathbf{x}(k), u(k), u(k+1)\right) \right\rangle, \\
\mathbf{x}(k+3) &= \left\langle\, \mathsf{F}^3 \,\middle|\, \mathsf{M}_p^9\left(\mathbf{x}(k), u(k), u(k+1), u(k+2)\right) \right\rangle,
\end{aligned}$$

with parameter tensors of dimension $\mathsf{F}^1 \in \mathbb{R}^{\times(4)2}$, $\mathsf{F}^2 \in \mathbb{R}^{\times(13)2}$ and $\mathsf{F}^3 \in \mathbb{R}^{\times(46)2}$. The state deviations from the references are computed for the prediction horizon by

$$\begin{aligned}
&\|\hat{\mathbf{x}}(k+1) - \mathbf{r}(k+1)\|_{\mathbf{Q}}^2 \\
&= \left\langle \mathsf{F}_{cost,x}^1 \,\middle|\, \mathsf{M}_p^2\left(\mathbf{x}(k), u(k-1), u(k), u(k+1), u(k+2), \mathbf{r}(k+1), \mathbf{r}(k+2), \mathbf{r}(k+3)\right) \right\rangle \\
&\|\hat{\mathbf{x}}(k+2) - \mathbf{r}(k+2)\|_{\mathbf{Q}}^2 \\
&= \left\langle \mathsf{F}_{cost,x}^2 \,\middle|\, \mathsf{M}_p^6\left(\mathbf{x}(k), u(k-1), u(k), u(k+1), u(k+2), \mathbf{r}(k+1), \mathbf{r}(k+2), \mathbf{r}(k+3)\right) \right\rangle \\
&\|\hat{\mathbf{x}}(k+3) - \mathbf{r}(k+3)\|_{\mathbf{Q}}^2 \\
&= \left\langle \mathsf{F}_{cost,x}^3 \,\middle|\, \mathsf{M}_p^{18}\left(\mathbf{x}(k), u(k-1), u(k), u(k+1), u(k+2), \mathbf{r}(k+1), \mathbf{r}(k+2), \mathbf{r}(k+3)\right) \right\rangle
\end{aligned}$$

and represented with respect to the actual state, the input variables and the reference for the whole prediction horizon, which is necessary to summarize all the terms to compute the overall cost function. This results in the parameter tensors of the first $\mathsf{F}^1_{cost,x} \in \mathbb{R}^{\times^{(24)}2}$, *second* $\mathsf{F}^2_{cost,x} \in \mathbb{R}^{\times^{(72)}2}$ *and third* $\mathsf{F}^3_{cost,x} \in \mathbb{R}^{\times^{(216)}2}$ *time step. The input changes are given by*

$$\begin{aligned}
\|\Delta\mathbf{u}(k)\|^2_{\mathbf{R}} &= \left\langle \mathsf{F}^1_{cost,u} \,\middle|\, \mathsf{M}^2_p\left(\mathbf{x}(k), u(k-1), u(k), u(k+1), u(k+2), \mathbf{r}(k+1), \mathbf{r}(k+2), \mathbf{r}(k+3)\right)\right\rangle \\
\|\Delta\mathbf{u}(k+1)\|^2_{\mathbf{R}} &= \left\langle \mathsf{F}^2_{cost,u} \,\middle|\, \mathsf{M}^6_p\left(\mathbf{x}(k), u(k-1), u(k), u(k+1), u(k+2), \mathbf{r}(k+1), \mathbf{r}(k+2), \mathbf{r}(k+3)\right)\right\rangle \\
\|\Delta\mathbf{u}(k+2)\|^2_{\mathbf{R}} &= \left\langle \mathsf{F}^3_{cost,u} \,\middle|\, \mathsf{M}^{18}_p\left(\mathbf{x}(k), u(k-1), u(k), u(k+1), u(k+2), \mathbf{r}(k+1), \mathbf{r}(k+2), \mathbf{r}(k+3)\right)\right\rangle,
\end{aligned}$$

with tensors of same dimensionality as for the states. The cost function of the MPC optimization problem (4.41) results in a tensor notation

$$\begin{aligned}
J(k) &= \left\langle \sum_{i=1}^{3} \mathsf{F}^i_{cost,x} + \sum_{i=0}^{2} \mathsf{F}^i_{cost,u} \,\middle|\, \mathsf{M}^{18}_p\left(\mathbf{x}(k), u(k-1), \ldots, u(k+2), \mathbf{r}(k+1), \ldots, \mathbf{r}(k+3)\right)\right\rangle \\
&= \left\langle \mathsf{F}_J \,\middle|\, \mathsf{M}^{18}_p\left(\mathbf{x}(k), u(k-1), u(k), u(k+1), u(k+2), \mathbf{r}(k+1), \mathbf{r}(k+2), \mathbf{r}(k+3)\right)\right\rangle, \quad (4.47)
\end{aligned}$$

with parameter tensor $\mathsf{F}_J \in \mathbb{R}^{\times^{(198)}2}$ *of the cost function. This representation of the cost function is used now to solve the MPC optimization problem. Therefore, the gradient of* $J(k)$ *by operational tensors as described in Subsection 2.3.5 is computed and given as additional parameters to the solver in Matlab, that uses an interior point algorithm, to improve the optimization performance, [12].*
This example shows, that the parameter tensors of the cost function gets large even though a small system with a short prediction horizon is investigated. The size of F_J *is* $\mathbb{R}^{\times^{(198)}2}$ *and depends on the number of states and the prediction horizon. The number of dimensions increases exponentially with the prediction horizon. As mentioned in Subsection 3.6, this can be avoided by approximating the multi-step transitions by a Taylor approximation of order* 2, *as derived in Section 2.3.5. In this case the parameter tensor* F_J *has* 44 *dimensions which is better manageable. Figure 4.16 compares the closed loop simulation results of the second state for an MPC which uses the exact cost function (4.47) and for an MPC, that uses an approximated version of the parameter tensor.*

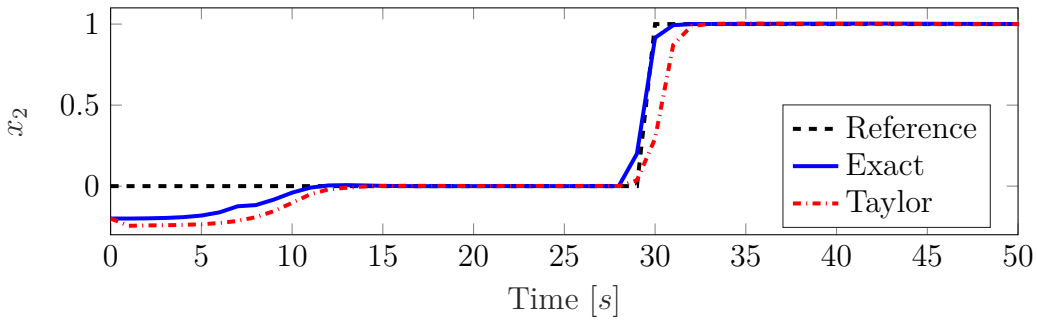


Figure 4.16: Closed loop simulation result result for state x_2

The results show that the approximated version has a slightly worse tracking performance but still shows good results. By approximating the parameters of the multi-step transitions the

complexity of the parameter tensor F_J of the cost function is reduced. This leads to less computation effort during optimization, which is shown in Table 4.2 by the average computation times for solving the optimization problem (4.46) in one time step. The table shows also the benefits of the use of an analytically computed gradient by (2.68). The computation time is reduced significantly, when a gradient is provided for the optimization algorithm. But the computation times show also that with the exact representation the computation of the optimization is too complex, since the computation takes longer than the sampling time, which is not acceptable, when the controller should be applied in real-time. With the approximation the computation time is below the sampling time of one second and still gives good results.

Table 4.2: Average computation times for solving the MPC optimization problem

	Analytic gradient	Approximated gradient
Exact	3.04s	20.32s
Taylor approximation	0.11s	0.76s

4.4.2 Convexity analysis

The previous subsection derived a closed description of the cost function of the MPC optimization problem in tensor notation. The MPC control strategy is applied very often by using linear models inside the controller, leading to a convex optimization problem in the proposed configuration, [75]. It was shown in Chapter 3, that MTI state space models have nice modeling properties and thus may give better results than a linear model when used inside an MPC. Therefore, in this subsection the convexity property of the MPC optimization problem is investigated when MTI models are used by summarizing the results given in [56]. An optimization problem is convex, if and only if, the cost function and the constraints are convex, [12]. Here it is assumed, that the constraints are convex, which is the case, e.g. if linear constraints or box constraints are used, which is the case often in application. Thus the focus on convex constraints is not a strong restriction in general. To check the convexity of the cost function $J(k)$ the second order derivatives of $J(k)$ have to be investigated. The Hessian matrix of $J(k)$ with respect to the optimization variables $\mathbf{u}(k+i),\ i=0,\dots,H_u-1$ has to be positive semi-definite

$$\mathbf{H}_J = \left(\frac{\partial^2 J(k)}{\partial u_i(k+l)\partial u_j(k+q)} \right) \geq 0, \forall i,j = 1,\dots,m,\ \forall l,q = 0,\dots,H_u-1.$$

The Hessian $\mathbf{H}_J \in \mathbb{R}^{mH_u \times mH_u}$ is positive semi-definite, if its eigenvalues are greater than or equal to zero. Because of the fact, that a sum of convex functions is a convex function, the convexity of the first and the second sum of the cost function (4.41) can be investigated independently. The second part $\sum_{i=0}^{H_u-1} \|\Delta\mathbf{u}(k+i)\|^2_{\mathbf{R}}$ is obviously convex, since it is a sum of

squared input differences. This reduces the convexity analysis of $J(k)$ to the first term

$$J_x(k) = \sum_{i=1}^{H_p} \|\hat{\mathbf{x}}(k+i) - \mathbf{r}(k+i)\|_{\mathbf{Q}}^2.$$

In [56] the convexity of the optimization problem for different classes of MTI systems and different prediction horizons were investigated. For a one step prediction horizon $H_p = 1$ the optimization problem (4.41) of MTI systems is convex. When extending the prediction horizon to two time steps $H_p = 2$, it turned out, that the convexity is not given for all MTI systems, which can be shown by the following example.

Example 4.6 *Consider the state equation of a second order discrete-time MTI system*

$$\begin{pmatrix} x_1(k+1) \\ x_2(k+1) \end{pmatrix} = \begin{pmatrix} x_1 x_2 + u \\ u \end{pmatrix}$$

and weighing matrices chosen as identity. The Hessian matrix of J_x for $H_p = 2$ gives

$$\mathbf{H}_J = \begin{pmatrix} 12u(k)^2 + 4u(k+1) & 4u(k) \\ 4u(k) & 4 \end{pmatrix},$$

when the states and the reference vectors were chosen as $x_1 = 0$, $x_2 = 0$, $\mathbf{r}(k+1) = \begin{bmatrix} 1 & -1 \end{bmatrix}^T$ and $\mathbf{r}(k+2) = \begin{bmatrix} 1 & -1 \end{bmatrix}^T$. At the point $u(k) = 0$ and $u(k+1) = -1$ the eigenvalues of $\mathbf{H}$ are $\lambda_{1,2} = \pm 4$. It follows that $\mathbf{H}$ is indefinite. Thus, the cost function is not a convex function in $\mathbf{u} \in \mathbb{R}^2$.

The example shows that MTI systems in general do not result in a convex optimization problem (4.46). Because of that, in [56] a subclass of MTI models is defined for which the optimization problem is convex with $H_p = 2$. This subclass is based on input linear MTI systems with one input, that are defined in matrix notation by

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{m}\left(\mathbf{x}(k)\right) + \mathbf{B}u$$

with parameter matrices $\mathbf{F} \in \mathbb{R}^{2^n \times n}$ and $\mathbf{B} \in \mathbb{R}^{n \times 1}$. It was shown that certain constraints on the parameter matrix have to be imposed, such that the optimization problem is convex. With this structural constraint a subclass of MTI systems can be defined for which the MPC optimization problem is convex for a prediction horizon of 2.

Now this concept is extended to answer the question, how to check the convexity property for general MTI systems with arbitrary prediction horizons. This would give an idea of the complexity of the optimization problem, which is an interesting information. It is important for e.g. the choice of the solver, the sampling time or the hardware for the controller. To check the convexity of the cost function $J(k)$ it is sufficient to show the convexity of the first part $J_x(k)$ of the cost function. The function is convex if and only if its Hessian with respect to the optimization variables

$$\mathbf{H}_{J_x}(\mathbf{u}(k), \ldots, \mathbf{u}(k+H_u-1)) = \left(\frac{\partial^2 J_x(\mathbf{u}(k), \ldots, \mathbf{u}(k+H_u-1))}{\partial u_i(k+l) \partial u_j(k+q)} \right) \in \mathbb{R}^{mH_u \times mH_u} \tag{4.48}$$

is positive semi-definite $\mathbf{H}_{J_x}(\mathbf{u}(k),\ldots,\mathbf{u}(k+H_u-1)) \geq 0$. In Subsection 4.4.1 the first part of the cost function was derived in tensor notation (4.44). At a given time step k the values of the current state $\mathbf{x}(k)$ and the reference values $\mathbf{r}(k+i),\ i=1,\ldots,H_p$ are known and fixed. The optimization variables are the inputs $\mathbf{u}(k+i),\ i=0,\ldots,H_u-1$. To get a description of the cost function with respect to the optimization variables the contracted product of (4.44) has to be evaluated along the dimensions belonging to the state and references. With the evaluation of the contracted product the factors of the states and references are included to the parameter tensor

$$\begin{aligned}&\bar{\mathsf{F}}_{cost,x}(\mathbf{x}(k),\mathbf{r}(k),\ldots,\mathbf{r}(k+H_p))\\&=\left\langle \mathsf{F}_{cost,x} \,\middle|\, \mathsf{M}_p^{2(n+m)^{H_p-1}}(\mathbf{x}(k),\mathbf{r}(k),\ldots,\mathbf{r}(k+H_p)) \right\rangle_{\mathbf{i}_{\mathbf{x}(k),\mathbf{r}(k+i)},1:2(n+m)^{(H_p-1)}n(H_p+1)}\end{aligned}$$

with the vector

$$\begin{aligned}\mathbf{i}_{\mathbf{x}(k),\mathbf{r}(k+i)} = \boxplus_{i=1}^{2(n+m)^{H_p-1}} \Big(\ &(i-1)n+(i-1)mH_u+(i-1)nH_p+1 \ \cdots\\ &(i-1)n+(i-1)mH_u+inH_p \quad (i-1)n+imH_u+inH_p+1 \ \cdots \ in+imH_u+inH_p\ \Big),\end{aligned}$$

containing the indices belonging to the state and references in $\mathsf{F}_{cost,x}$. Using this parameter tensor the cost function $J_x(k)$ is described by

$$J_x(k)=\left\langle \bar{\mathsf{F}}_{cost,x}(\mathbf{x}(k),\mathbf{r}(k),\ldots,\mathbf{r}(k+H_p)) \,\middle|\, \mathsf{M}_p^{2(n+m)^{H_p-1}}(\mathbf{u}(k),\ldots,\mathbf{u}(k+H_u-1)) \right\rangle,$$

such that the monomials depend on the input values only. With the representation of the cost function in the tensor framework the Hessian (4.48) is computed by (2.74) with operational tensors resulting in

$$\mathbf{H}_{J_x}(\mathbf{u}(k),\ldots,\mathbf{u}(k+H_u-1))=\left\langle \mathsf{H}_{J_x} \,\middle|\, \mathsf{M}_p^{2(n+m)^{H_p-1}}(\mathbf{u}(k),\ldots,\mathbf{u}(k+H_u-1)) \right\rangle, \tag{4.49}$$

where the parameter tensor H_{J_x} is computed by (2.75). This shows that the Hessian matrix of the part J_x of the cost function that is significant for the convexity is computed from the cost function parameter tensor representation (4.45). This goes beyond the scope of [56] because it allows to compute the Hessian also for general MTI systems with larger prediction horizons. In the last step the semi-definiteness of the Hessian (4.49) has to be checked. If the matrix is positive semi-definite for all states, references and inputs, the MPC optimization problem is convex. The semi-definiteness of the Hessian is quite hard to check in general. It depends on the monomials that are used in the cost function. But it is simple to check the property for several typical points in the operating range by evaluting (4.49) for these points and computing the eigenvalues. If the Hessian has a negative eigenvalue for one of these points, the cost function is non-convex. But one has to take care, if all checked points result in positive eigenvalues only, this does not guarantee convexity. By checking some points only non-convexity can be shown. To show convexity the semi-definiteness of (4.49) has to be shown in general, which can be complex, [12, 14].

Example 4.7 *To illustrate the approach with the Hessian computed by operational tensors the system of Example 4.6 is considered. At first the convexity for a prediction horizon*

of $H_p = 1$ is investigated. With that the cost function J_x is given by

$$J_x(k) = \left\langle \mathsf{F}_{cost,x} \,\middle|\, \mathsf{M}_p^2\left(\mathbf{x}(k), \mathbf{u}(k), \mathbf{r}(k+1)\right) \right\rangle.$$

The cost function is described by the optimization variable $\mathbf{u}(k)$ by

$$J_x(\mathbf{u}(k)) = \left\langle \bar{\mathsf{F}}_{cost,x}(\mathbf{x}(k), \mathbf{r}(k+1)) \,\middle|\, \mathsf{M}_p^2\left(\mathbf{u}(k)\right) \right\rangle$$

with the parameter tensor

$$\bar{\mathsf{F}}_{cost,x}(\mathbf{x}(k), \mathbf{r}(k+1)) = \left\langle \mathsf{F}_{cost,x} \,\middle|\, \mathsf{M}_p^2\left(\mathbf{x}(k), \mathbf{r}(k+1)\right) \right\rangle_{\mathbf{i}_{\mathbf{x}(k),\mathbf{r}(k+i)},1:8},$$

where the index vector is written as $\mathbf{i}_{\mathbf{x}(k),\mathbf{r}(k+i)} = \begin{pmatrix}1 & 2 & 4 & 5 & 6 & 7 & 9 & 10\end{pmatrix}^T$. Computing the Hessian of $J_x(\mathbf{u}(k))$ results in the tensor notation

$$\mathbf{H}_{J_x}(\mathbf{u}(k)) = \left\langle \mathsf{H}_{J_x}(\mathbf{x}(k), \mathbf{r}(k+1)) \,\middle|\, \mathsf{M}_p^2\left(\mathbf{u}(k)\right) \right\rangle = 4.$$

Thus, the Hessian is positive for all values of $\mathbf{x}(k)$, $\mathbf{u}(k)$ and $\mathbf{r}(k+1)$ and therefore the optimization problem is convex, as it is the case for all MTI system with $H_p = 1$.

For a prediction horizon of two $H_p = 2$ the same steps are executed to compute the Hessian of

$$J_x(k) = \left\langle \mathsf{F}_{cost,x} \,\middle|\, \mathsf{M}_p^6\left(\mathbf{x}(k), \mathbf{u}(k), \mathbf{u}(k+1), \mathbf{r}(k+1), \mathbf{r}(k+2)\right) \right\rangle.$$

Here an additional optimization variable is necessary leading to

$$J_x(\mathbf{u}(k), \mathbf{u}(k+1)) = \left\langle \bar{\mathsf{F}}_{cost,x}(\mathbf{x}(k), \mathbf{r}(k+1), \mathbf{r}(k+2)) \,\middle|\, \mathsf{M}_p^6\left(\mathbf{u}(k), \mathbf{u}(k+1)\right) \right\rangle,$$

where the parameter tensor is computed with index vector $\mathbf{i}_{\mathbf{x}(k),\mathbf{r}(k+i)} \in \mathbb{R}^{36}$ as

$$\bar{\mathsf{F}}_{cost,x}(\mathbf{x}(k), \mathbf{r}(k+1), \mathbf{r}(k+2)) = \left\langle \mathsf{F}_{cost,x} \,\middle|\, \mathsf{M}_p^6\left(\mathbf{x}(k), \mathbf{r}(k+1), \mathbf{r}(k+2)\right) \right\rangle_{\mathbf{i}_{\mathbf{x}(k),\mathbf{r}(k+i)},1:36}.$$

With the tensor approach (4.49) the Hessian yields

$$\mathbf{H}_{J_x}(\mathbf{u}(k), \mathbf{u}(k+1)) = \left\langle \mathsf{H}_{J_x}(\mathbf{x}(k), \mathbf{r}(k+1), \mathbf{r}(k+2)) \,\middle|\, \mathsf{M}_p^6\left(\mathbf{u}(k), \mathbf{u}(k+1)\right) \right\rangle.$$

Now it has to be checked if the Hessian is positive semi-definite. Here different combinations of states, reference and inputs can be checked. Evaluating the Hessian at the point given in Example 4.6, gives the Hessian matrix

$$\mathbf{H}_{J_x}(0,-1) = \left\langle \mathsf{H}_{J_x}\left(\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}1\\-1\end{pmatrix}, \begin{pmatrix}1\\-1\end{pmatrix}\right) \,\middle|\, \mathsf{M}_p^2\left(0,-1\right) \right\rangle = \begin{pmatrix}-4 & 0\\ 0 & 4\end{pmatrix},$$

that has eigenvalues ± 4. This shows that the Hessian is not positive semi-definite for the whole operating range and thus the optimization problem is non-convex. This goes together with the results of Example 4.6, where the Hessian was computed symbolically directly. To check a larger horizon does not make sense, since this cannot restore convexity. But this example shows that with the proposed approach the Hessian could have been computed also for larger horizons and is not limited to a certain subclass.

If convexity can be shown for small prediction horizons only, this can have positive effects also on MPC optimization problems with larger prediction horizon. If the optimization problem (4.41) is not convex the optimization result depends on the choice of the initial values. To improve the choice of initial values, the solution of the optimization problem with smaller prediction horizon that leads to a convex cost function and a global optimum may help. This solution can be used for the initialization for the nonconvex optimization problem of larger prediction horizon. In [56] an example was shown where a solution next to the global optimum is found with fast convergence with this initialization.

4.5 Adaptive model predictive control with successive linearization

Because of its ability to optimally control multiple-input multiple-output systems with consideration of constraints and disturbances, MPC is a popular control method especially for systems with time delays or large time constants. Inside the controller a model is used to predict the future plant behavior. In many applications linear models are used for that, [73]. This has the advantage, that the MPC optimization problem is convex and can thus be solved very efficiently by standard solvers like the interior point method, which makes it well suitable for real-time implementation, [12]. But using linear models leads to the drawback that, depending on the application, linear models describe the behavior of a nonlinear system in a sufficient accuracy in the neighborhood of the operating point only. When nonlinear effects are essential, the prediction of the system behavior by a linear model might not be sufficient. In this case nonlinear models can be used. Since MTI systems showed some good modeling properties as shown in Chapter 3, in this section multilinear state space model should be used to describe the behavior of the plant. But the previous Section 4.4 showed, that the good properties of the optimization problems get lost in general when MTI systems are used. Only for some subclasses the MPC optimization problem is convex, when MTI models are applied. In general the optimization problem is non-convex in this case. This could lead to long computation times when solving the optimization problem and it is not guaranteed that a global optimal solution is found. This large complexity could also lead to problems within implementation, since in this case it has to be guaranteed that the result of the optimization is computed during one time step.
To combine the advantages of both MPC approaches, i.e. the good modeling properties and the convex optimization problem, in [57] or [76] different concepts are given that adapt the linear system to the current operating conditions of the plant, e.g. by online linearization of a nonlinear model or an LPV model. By adapting the linear model to the current plant conditions this leads to improved modeling properties compared to the application of one single linearized model. Since the model used during optimization is linear in this approach, the convexity of the optimization problem is guaranteed. Here the idea is to successively linearize the MTI system in each sampling step around actual operating conditions, as it is done for general nonlinear systems, e.g. in [58]. This MPC approach is called adaptive MPC, since the model used during optimization is adapted to the plant. To avoid conflicts with other approaches, where the linear model is e.g. adapted to the plant by an online black-box

identification with measurement data, the concept used here is called adaptive model predictive control with successive linearization (AMPC-SL). Therefore, a continuous-time MTI model

$$\begin{aligned}\dot{\mathbf{x}} &= \langle\, \mathsf{F} \mid \mathsf{M}\,(\mathbf{x}, \mathbf{u}, \mathbf{d})\, \rangle\,,\\ \mathbf{y} &= \langle\, \mathsf{G} \mid \mathsf{M}\,(\mathbf{x}, \mathbf{u})\, \rangle\,,\end{aligned}$$

of the plant has to be identified, where the inputs are divided into control and disturbance inputs. In each time step, the MTI model is linearized around an operating point. Thus in the first step an operating point next to the actual operating conditions is computed by (4.31) with constraints (4.32) to (4.34). The values of the operating point are set to the actually measured disturbances and the outputs are set to the reference values. The constraints on the inputs are given by the formulation of the MPC problem (4.20). This leads to the operating point

$$\begin{pmatrix}\bar{\mathbf{x}} & \bar{\mathbf{u}} & \bar{\mathbf{d}} & \bar{\mathbf{y}}\end{pmatrix}. \tag{4.50}$$

Since the plant is described by an MTI model, the online linearization is computed very efficiently without any symbolical computations by the linearization approach introduced in Section 3.3. An analytically correct linearization can be computed here without much computational effort and by standard numerical operations such that it is also suitable for real-time implementations. The linear approximation of the state equation is computed as a function of the operating point by (4.35) to (4.38) resulting in the linear system matrices $\mathbf{A}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}})$, $\mathbf{B}_u(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}})$ and $\mathbf{B}_d(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}})$. It is assumed that the output is not disturbed, such that the parameter matrices of the output equation $\mathbf{C}(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ and $\mathbf{D}(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ follow from (3.48) and (3.49). This linear model is discretized by standard discretization techniques for linear systems, [12]. The discrete-time linear model approximates the system behavior around the operating point by

$$\begin{aligned}\mathbf{x}(k{+}1){-}\bar{\mathbf{x}} &= \mathbf{A}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}})\,(\mathbf{x}(k){-}\bar{\mathbf{x}}) + \mathbf{B}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}})\,(\mathbf{u}(k){-}\bar{\mathbf{u}}){+}\mathbf{B}_d(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}})\,\left(\mathbf{d}(k){-}\bar{\mathbf{d}}\right),\\ \mathbf{y}(k){-}\bar{\mathbf{y}} &= \mathbf{C}(\bar{\mathbf{x}}, \bar{\mathbf{u}})\,(\mathbf{x}(k){-}\bar{\mathbf{x}}){+}\mathbf{D}(\bar{\mathbf{x}}, \bar{\mathbf{u}})\,(\mathbf{u}(k){-}\bar{\mathbf{u}}),\end{aligned}$$

This linearization of the MTI system is used to solve the optimization problem

$$\min_{\substack{\mathbf{u}(k+i),\\ i=0,\ldots,H_u-1}} \quad J(k) \tag{4.51}$$

$$\begin{aligned}\text{subject to}\quad \hat{\mathbf{x}}(k+i+1) - \bar{\mathbf{x}} &= \mathbf{A}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}})\,(\hat{\mathbf{x}}(k+i) - \bar{\mathbf{x}}) + \mathbf{B}_u(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}})\,(\mathbf{u}(k+i) - \bar{\mathbf{u}}) \cdots\\ &\quad + \mathbf{B}_d(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}})\,\left(\mathbf{d}_{pred}(k+i) - \bar{\mathbf{d}}\right),\ i = 0, \ldots, H_p - 1,\\ \hat{\mathbf{y}}(k+i) + \bar{\mathbf{y}} &= \mathbf{C}\,(\hat{\mathbf{x}}(k+i) - \bar{\mathbf{x}})\,,\ i = 1, \ldots, H_p - 1,\\ \mathbf{u}_{min} &\le \mathbf{u}(k+i) \le \mathbf{u}_{max},\ i = 0, \ldots, H_u - 1,\\ \mathbf{y}_{min} &\le \hat{\mathbf{y}}(k+i) \le \mathbf{y}_{max},\ i = 1, \ldots, H_p.\end{aligned}$$

With a quadratic cost function as (4.41) the optimization problem is convex. The solution of the optimization problem is the input sequence $\hat{\mathbf{u}}(k+i)$ with $i = 0, \ldots, H_u - 1$ for the input horizon. The inputs $\hat{\mathbf{u}}(k)$ of the first time step are applied to the plant. At the next time instant again an operating point of the MTI system is calculated, the system is linearized

around this operating point and the MPC optimization problem is solved with this linear model.

Up to this point the main idea of the proposed algorithm for AMPC-SL for MTI systems is described. The performance can be further improved with tools introduced in Section 3. From a numerical point of view problems may arise for the computation of the operating point (4.50) and the MPC optimization problem (4.51), when the signals or variables are of different magnitude, which is e.g. the case in the application area of heating systems with flows in $\sim 10^{-3} m^3/s$ and temperatures around $300K$. A numerical preconditioning is done by scaling of all signals $\mathbf{x}$, $\mathbf{u}$, $\mathbf{d}$ and $\mathbf{y}$. The block diagram of the resulting closed loop system with the scaling is depicted in Figure 4.17.

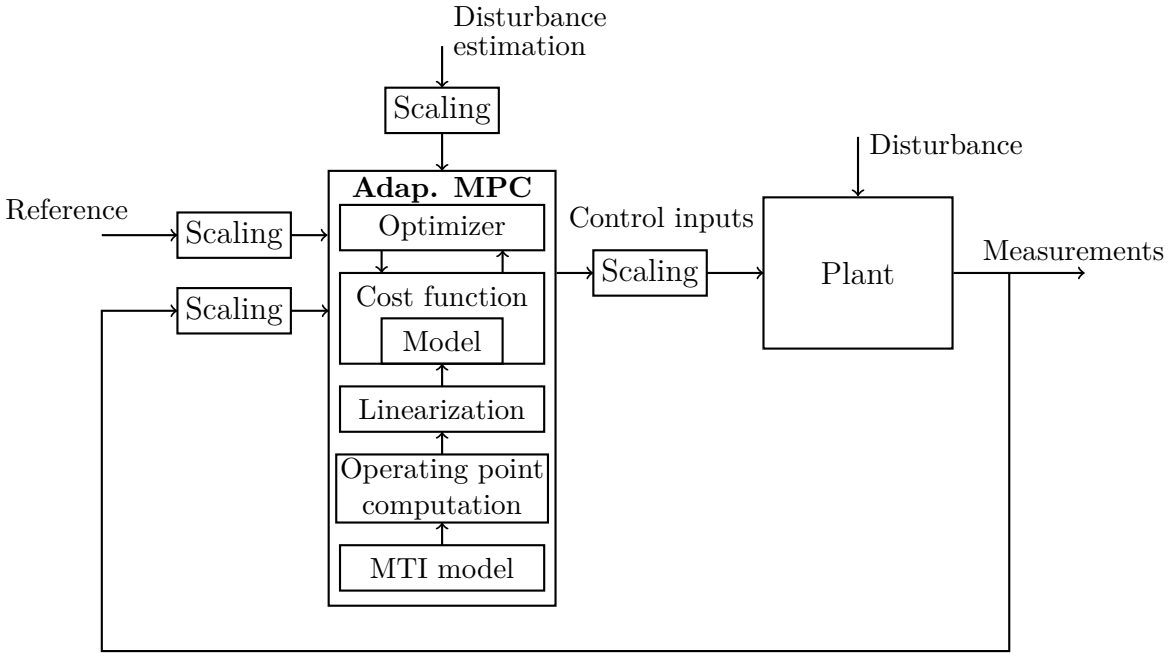


Figure 4.17: Closed loop of the AMPC-SL controller with scaled signals and the plant

The operating region

$$\mathbf{x} \in [\mathbf{x}_{min}, \mathbf{x}_{max}]\,,\ \mathbf{u} \in [\mathbf{u}_{min}, \mathbf{u}_{max}]\,,\ \mathbf{d} \in [\mathbf{d}_{min}, \mathbf{d}_{max}]\,,\ \mathbf{y} \in [\mathbf{y}_{min}, \mathbf{y}_{max}]\,. \tag{4.52}$$

of each signal that has to be specified by the user is scaled to the intervals $\tilde{\mathbf{x}}$, $\tilde{\mathbf{u}}$, $\tilde{\mathbf{d}}$ and $\tilde{\mathbf{y}} \in [0, 1]$. This has the advantage that all signals have the same magnitude and that multiplications of the signals as they occur in multilinear models, stay in this interval and do not get larger as it is the case for the multiplication of numbers larger than 1. Thus, the MTI system is scaled as described in Lemma 3.2. The model is scaled before controller design, such that the algorithm works with the scaled model of the plant, resulting in a scaled operating point, a scaled linear system and a scaled result of the optimization problem. Therefore, the measured signals and disturbance predictions have to be scaled before the AMPC-SL

algorithm according to (3.63) to (3.65) by

$$\begin{aligned}
\tilde{x}_i &= \frac{1}{\tilde{a}_i}\left(x_i - \tilde{b}_i\right),\ i = 1,\dots,n,\\
\tilde{d}_i &= \frac{1}{\tilde{a}_{n+m+i}}\left(d_i - \tilde{b}_{n+m+i}\right),\ i = 1,\dots,m_d,\\
\tilde{y}_i &= \frac{1}{\tilde{c}_i}\left(y_i - \tilde{e}_i\right),\ i = 1,\dots,n.
\end{aligned}$$

The control input determined by the MPC has to be transformed back to the original operating area by

$$u_i = \tilde{a}_{n+i}\tilde{u}_i + \tilde{b}_{n+i},\ i = 1,\dots,m,$$

before being applied to the plant. This avoids numerical problems caused by the different magnitudes of the signals.

When focusing on real-time implementation, the use of linear systems avoids the online solution of nonlinear optimization problems. A quadratic problem has to be solved only. The linearization is efficient since it is computed based on decomposed parameter tensors of the MTI model. But for the computation of the operating point still a nonlinear optimization or root finding problem (4.31) has to be solved. Solving this optimization problem online could lead to timing problems. That is why an offline computation approach is proposed here. The MTI model and the operating region (4.52) have to be known in advance to apply the controller. The operating points of the system can be computed for different points in the operating region. A grid for the operating region is constructed by dividing the intervals for each signal in $N_{x,i}$, $N_{u,i}$ and $N_{d,i}$ parts, respectively. As an example the grid vector for the state x_i is given by

$$\mathbf{X}_j = \begin{pmatrix} x_{j,min} & x_{j,min} + \kappa_{x,j} & \cdots & x_{j,min} + (N_{x,j}-1)\,\kappa_{x,j} & x_{j,max} \end{pmatrix} \in \mathbb{R}^{N_{x,j}+1},\ j = 1,\dots,n,$$

with the step size $\kappa_{x,j} = \frac{x_{j,max} - x_{j,min}}{N_{x,j}}$. The grid vectors $\mathbf{U}_j$ and $\mathbf{D}_j$ of the other signals are constructed analogously. For each combination of the elements of the grid vectors $\mathbf{X}_j(i_{x,j})$, $\mathbf{U}_j(i_{u,j})$ and $\mathbf{D}_j(i_{d,j})$ the optimization problem (4.31) is solved to compute an operating point of the plant for these operating conditions. The resulting operating point $\begin{pmatrix} \bar{\mathbf{x}} & \bar{\mathbf{u}} & \bar{\mathbf{d}} & \bar{\mathbf{y}} \end{pmatrix}$ is stored in a multidimensional array, i.e. a tensor Θ_{OP} with elements

$$\Theta_{OP}(i_{x,1},\dots,i_{x,n},i_{u,1},\dots,i_{u,m},i_{d,1},\dots,i_{d,m_d},:) = \begin{pmatrix} \bar{\mathbf{x}} & \bar{\mathbf{u}} & \bar{\mathbf{d}} & \bar{\mathbf{y}} \end{pmatrix},$$

of dimension $\mathbb{R}^{N_{x,1}+1\times\cdots\times N_{x,n}+1\times N_{u,1}+1\times\cdots\times N_{u,m}+1\times N_{d,1}+1\times\cdots\times N_{d,m_d}+1\times n+m+m_d+p}$. Each output, state, control and disturbance input belongs to one dimension of the tensor Θ_{OP}. In the last dimension the result of the corresponding operating point computation is stored. To reduce the storing effort, the tensor Θ_{OP} can be decomposed by standard decomposition routines as described in Subsection 2.1.2. The TT or HT are well suited here for application since a fixed desired accuracy can be set by the user for the decomposition. This gives a storage efficient representation of the field of operating points. During operation the closest grid point to the actual measurements is chosen. This means e.g. for the actual value of

disturbance d_j, that the grid point $\mathbf{D}_j(i_{d,j})$ with index $i_{d,j}$ is selected, such that the absolute difference $|d_j - \mathbf{D}_j(i_{d,j})|$ compared with all other grid points $\mathbf{D}_j(k),\ k = 1, \ldots, N_{d,j}, k \neq i_{d,j}$ is minimal. The indices for the other signals are determined in the same way. With the computed indices $i_{x,j}$, $i_{u,j}$ and $i_{d,j}$ the operating point can be restored from the field of operating points by

$$\begin{aligned}
\Theta_{OP}(i_{x,1}, \ldots, i_{x,n}, i_{u,1}, \ldots, i_{u,m}, i_{d,1}, \ldots, i_{x,m_d}, 1:n) &= \bar{\mathbf{x}}, \\
\Theta_{OP}(i_{x,1}, \ldots, i_{x,n}, i_{u,1}, \ldots, i_{u,m}, i_{d,1}, \ldots, i_{x,m_d}, n+1:n+m) &= \bar{\mathbf{u}}, \\
\Theta_{OP}(i_{x,1}, \ldots, i_{x,n}, i_{u,1}, \ldots, i_{u,m}, i_{d,1}, \ldots, i_{x,m_d}, n+m+1:n+m+m_d) &= \bar{\mathbf{d}}, \\
\Theta_{OP}(i_{x,1}, \ldots, i_{x,n}, i_{u,1}, \ldots, i_{u,m}, i_{d,1}, \ldots, i_{x,m_d}, n+m+m_d+1:n+m+m_d+p) &= \bar{\mathbf{y}}.
\end{aligned}$$

Thus, during operation the operating point is determined by simply selecting the elements from the field Θ_{OP} belonging to the operating conditions nearest to the actual ones. No optimization problem has to be solved during controller operation for the calculation of the operating point. Because of the discretization of the operating area, this leads to reduced accuracy, but it is acceptable, if the number of grid points is chosen sufficiently large for the given application. The reduced complexity for the operating point computation leads to less computational effort, which is important especially for real-time implementation. The algorithm for the AMPC-SL is summarized in Figure 4.18.

Example 4.8 *In this example an AMPC-SL controller is designed for a plant with one state, one input and no disturbance*

$$\begin{aligned}
\dot{x}_1 &= x_1 - x_1 u_1, \\
y_1 &= x_1,
\end{aligned}$$

where the state is measured and given as an output. To formulate the MPC optimization problem the standard quadratic cost function (4.41) for reference tracking is used. The operating region of the system is defined by

$$-5 \leq x_1 \leq 5,\ -5 \leq y_1 \leq 5,\ -5 \leq u_1 \leq 5.$$

The system has no disturbance input. With this operating region and a reference trajectory for the state, that is shown in the plot of the closed loop simulation results in Figure 4.19, the operating points of the system can be computed on the one hand online and on the other hand by precomputing a field of operating points. To precompute the field of operating points at first grid vectors for the signals of the system have to be specified. E.g. for the state a deviation to $N_{x,1} = 100$ parts is done, leading to

$$\begin{aligned}
\mathbf{X}_1 &= \begin{pmatrix} x_{1,min} & x_{1min} + \kappa_{x,1} & \cdots & x_{1,min} + (N_{x,1} - 1)\,\kappa_{x,1} & x_{1,max} \end{pmatrix} \\
&= \begin{pmatrix} -5 & -4.9 & \cdots & 4.9 & 5 \end{pmatrix} \in \mathbb{R}^{101 \times 1},
\end{aligned}$$

because the step size is given by $\kappa_{x,1} = \frac{x_{1,max} - x_{1,min}}{N_{x,1}} = 0.1$. The same discritization is applied to the input signal resulting in the grid vector $\mathbf{U}_1 \in \mathbb{R}^{101 \times 1}$. The resulting tensor with their operating points is of dimension $\Theta_{OP} \in \mathbb{R}^{101 \times 101 \times 2}$, since the output is equal to the state

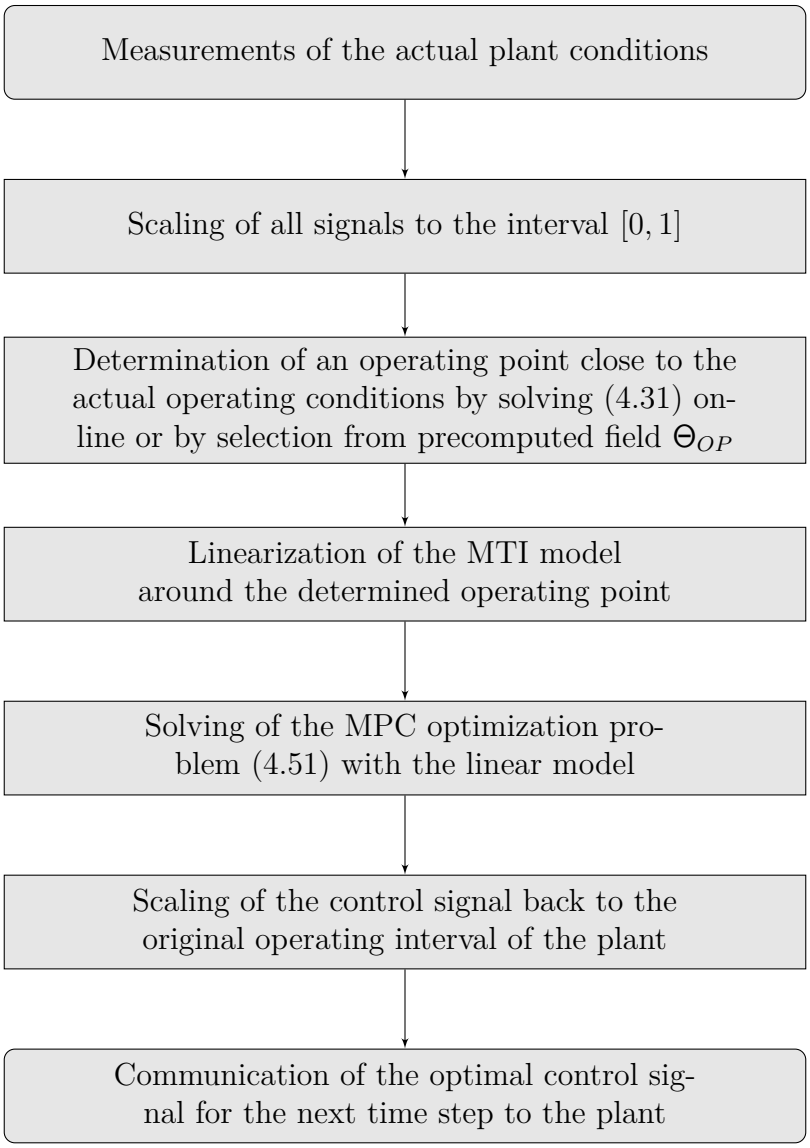


Figure 4.18: Flow chart of the AMPC-SL algorithm for each sampling step

here. In full representation this tensor has a storage effort of 20402 *elements. With a TT approximation of the data with an accuracy of* 10^{-15}*, i.e. nearly no loss in accuracy, the number of elements to be stored can be reduced to* 1822 *elements. Allowing a less accurate representation would lead to even less storage effort. By the constraint (4.32) the operating point of state* x_1 *should be equal to its reference value. Thus for a reference of* 2 *and a current input signal of* 0.5 *the corresponding operating point is selected from the tensor* Θ_{OP} *with indices* $i_{x,1} = 71$ *and* $i_{u,1} = 56$ *by*

$$\Theta_{OP}(71, 56, 1) = \bar{x}_1 = \bar{y}_1 = 2,$$
$$\Theta_{OP}(71, 56, 2) = \bar{u}_1 = 1.$$

Since all signals have a comparable magnitude, no scaling is used here in this Example. The

parameter tensors of the linarizations are precomputed, e.g. A_{lin} *by (4.35) for the state matrix* $\mathbf{A}(\bar{x}_1, \bar{u}_1, \bar{y}_1)$*, such that the parameter matrices of the linear model are simply computed by evaluating the contracted products (4.35) to (4.38) as well as (3.48) and (3.49) at the operating point in each sampling instant. Using this linearization the optimal control input is computed by solving the optimization problem (4.51). The controller performance of the AMPC-SL is compared to a standard MPC with a linear model that was computed for a fixed operating point. Both controllers are equipped with the same controller parameters, i.e. the same weights of* $Q = 1$ *and* $R = 1$ *as well as the same prediction* $H_p = 10$ *and input* $H_u = 5$ *horizons. The sampling time for both controllers is* $T_s = 0.1$*. The closed loop simulation results to a step in the reference for the state* x_1 *and the trajectories of the operating points is shown in Figure 4.19.*

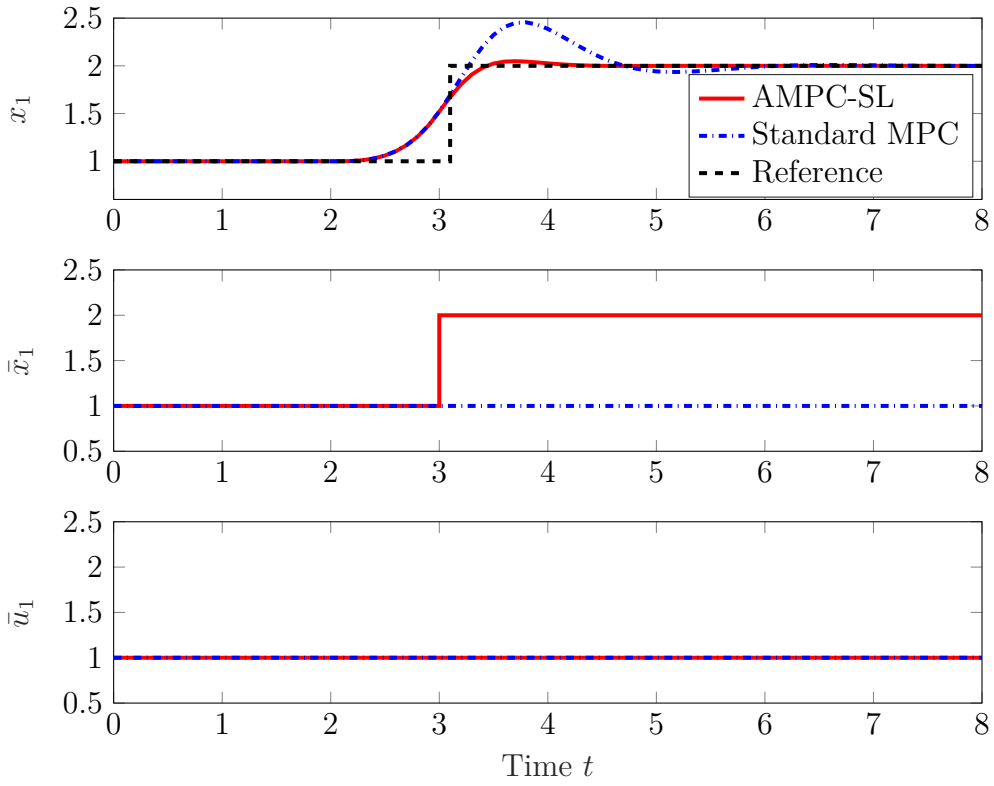


Figure 4.19: Closed loop simulation results of the example for AMPC-SL

For the standard MPC the MTI example system is linearized around the operating point $\bar{x}_1 = 1$ *and* $\bar{u}_1 = 1$*. For the standard MPC the resulting linear model is used for the whole simulation time. With the AMPC-SL the same operating point is used until a step change in the reference occurs. With the change in the reference also the operating point changes, such that it stays on the reference leading to the operating point* $\bar{x}_1 = 2$ *and* $\bar{u}_1 = 1$*. This change of the linear model used inside the controller has an effect on the controller performance. In the resulting trajectory for state* x_1 *the behavior until the reference changes is the same for both controllers. But the update of the linear model inside the AMPC-SL shows a significant advantage in the performance after the reference change. Because of the better description of the system*

dynamics with the linear model of the updated operating point after $t = 3$ the overshoot can be reduced enormously and thus the reference is reached much faster. This shows how the adaption of the model has a positive effect on the overall controller performance.

4.6 Distributed model predictive control

In today's applications the size of the plants increases more and more. This leads in general also to an increased complexity of the control problem. The application of a centralized controller for the whole plant can lead to very large computational effort, e.g. when predictive controllers are used. This leads to problems, e.g. for large heating systems, [35]. If the problem is too complex, i.e. its solution takes too much time, the centralized approach is not applicable. Because of that, an approach is derived here to split the predictive control task to several controller nodes. Each controller node has to solve smaller optimization problems resulting in a reduced complexity of the particular computations. But the global optimization goal should still be considered. Several structures for the design of predictive controllers are possible, [95].

So far in the previous chapters, centralized designs for MPC were focused on. The centralized predictive controller has access to all measurements and control signals, as depicted in Figure 4.20 for a plant that is composed of two subsystems.

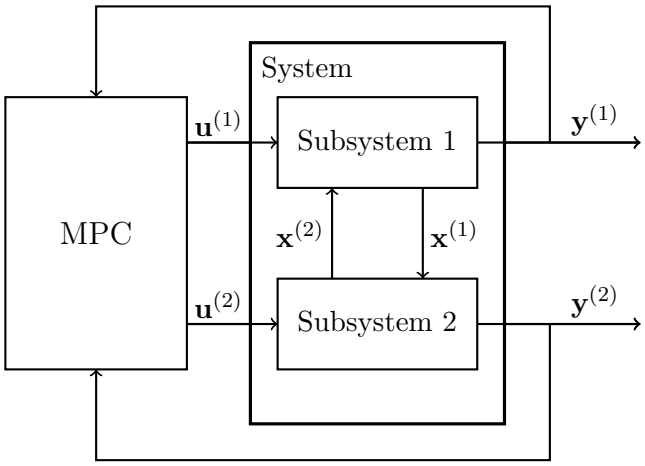


Figure 4.20: Centralized MPC, [16]

In centralized MPC the models of the plants can get very complex, which in general also leads to an increase in the complexity of the optimization problem of the controller. Several problems may arise in this case, if a central design is used. In predictive control an optimization has to be solved at each time instant. The decision variables are the inputs of the plant, that have to be computed for each input for the whole control horizon. Thus, the size of the search space of the optimization problem depends on the number of control inputs and the control horizon. If a central controller is used for a large-scale system with many control signals, this leads to a complex optimization problem, because besides other

influencing factors the search space for the optimization variables gets very large. This is a limiting factor for application, since for real-time implementation it has to be guaranteed that the result of the optimization problem is available at the end of each sampling time step. An arbitrary large increase of the sampling time is not possible, because the sampling time has to fit to the dynamics, i.e. the time constants, of the plant. Another solution could be the use of hardware with large computational power. But this is often not possible or leads to high economical costs, which is also not acceptable. Furthermore, a large amount of data has to be transmitted to a central location. This large communication effort could lead to delays and makes the structure sensitive to faults. If a change in the plant occurs, the whole controller has to be adapted to that in the central case. The model of the plant has to be reconstructed and the whole controller must be tuned again, e.g. its weighting factors for reference tracking and control effort $\mathbf{Q}$ and $\mathbf{R}$. With a decentralized or distributed design only the model and the controller of the changed subsystem has to be adapted, which results in less effort, [80]. This distribution of the controller task into several nodes solves some of the described problems for the central design and will be considered in the following.

The highest degree of distribution is achieved with a decentralized structure. Each subsystem has an own local controller. The controller nodes have access to the sensors and actors of the particular subsystems, but no information of other subsystems is available for them. The controllers of the different subsystems also do not exchange any information, e.g. on future control inputs. The controllers only take into account the influencing dynamics of the neighbouring systems by the measurement information on the own subsystems. This structure of independent controllers is reasonable for weakly interconnnected systems only. Because of the lack of information on the other subsystems, the decentralized design often leads to an significantly degraded performance or robustness compared to the central case, [70]. A decentralized structure for a plant with two subsystems is shown in Figure 4.21.

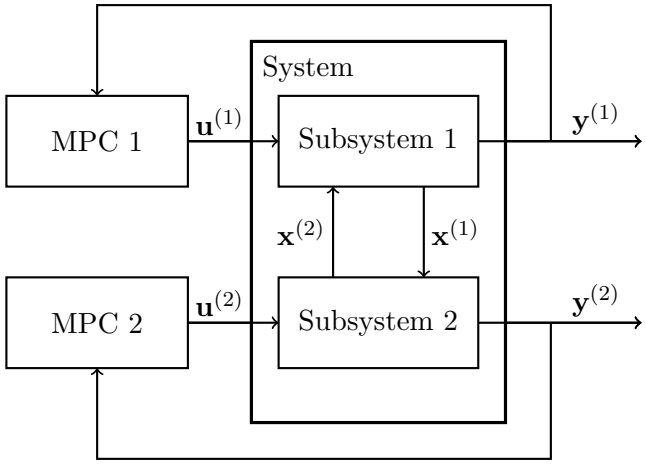


Figure 4.21: Decentralized MPC, [16]

If connections between subsystems are stronger, a fully decentralized design does not lead to a desired control performance. To combine the advantages of the previously introduced approaches, the distributed controller structure is described in the following. Like in the decentralized approach the control task is distributed to several controller nodes, but com-

munication between the nodes is allowed to improve the closed loop performance, [70]. This is called distributed MPC and abbreviated by DMPC in the literature, [16]. The controller gets information on the measurements and computes the control inputs of its own subsystem. Furthermore the controllers e.g. exchange information with other nodes on planned future control inputs and future output trajectories. This leads to a higher communication effort compared to the decentral design but improves the control performance especially for interconnected systems. In most cases it is not necessary that each controller gets information of all other subsystems. If a subsystem is connected to a few other subsystems only, the controller simply needs information on those neighboured subsystems and solves its small optimization problem considering these information. Thus, the communication effort is not increased enormously. But it depends on the structure of the system. In many cases the control structure follows from the underlying plant structure, [70]. A distributed controller setup is depicted in Figure 4.22.

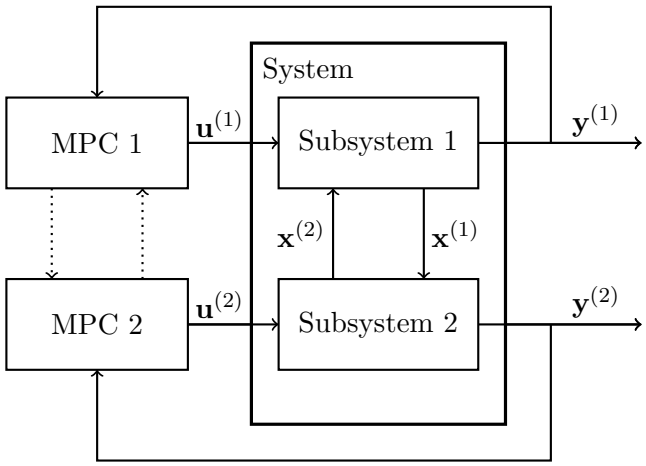


Figure 4.22: Distributed MPC, [16]

The focus in this thesis is on the application field of heating systems. Typical subsystems follow from the system structure with components like boilers, consumers or storage tanks. Especially for very large plants the design of a central predictive controller gets too complex, such that a decentralized or distributed design should be investigated here. Since the components of heating systems are interconnected, e.g. by a water flow through pipes, a distributed design is used here. The structure of the proposed DMPC approach with MTI submodels is described in the following.

It is assumed, that the plant model is divided into N_{sub} subsystems. As introduced in Section 3.7.1, the dynamics of each subsystem is described by an MTI model

$$\dot{\mathbf{x}}^{(i)} = \left\langle \mathsf{F}^{(i)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(i)}, \mathbf{u}^{(i)}, \mathbf{s}^{(i)}, \mathbf{d}^{(i)}\right) \right\rangle, \tag{4.53}$$

$$\mathbf{y}^{(i)} = \left\langle \mathsf{G}^{(i)} \,\middle|\, \mathsf{M}\left(\mathbf{x}^{(i)}, \mathbf{u}^{(i)}, \mathbf{s}^{(i)}, \mathbf{d}^{(i)}\right) \right\rangle, \tag{4.54}$$

with $i = 1, \ldots, N_{sub}$. The inputs of the i^{th} subsystem are dived into control inputs $\mathbf{u}^{(i)}$, external disturbances $\mathbf{d}^{(i)}$ and influences from other subsystems $\mathbf{s}^{(i)}$ that are handled as

internal disturbances for the subsystem i. The internal disturbances follow from the coupling equation (3.93)

$$\begin{pmatrix} \mathbf{s}^{(1)} \\ \vdots \\ \mathbf{s}^{(N_{sub})} \end{pmatrix} = \mathbf{h}(\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(N_{sub})}, \mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(N_{sub})}). \tag{4.55}$$

Each subsystem has its own local predictive controller node, that implements the AMPC-SL approach derived in Section 4.5. The controller node of subsystem i is constructed as follows. Inside the local controller the model of its own subsystem is available to them. The controller gets measurement information from the own subsystem and controls the actors of this subsystem. A quadratic cost function

$$J_i(k) = \sum_{j=1}^{H_p} \|\hat{\mathbf{x}}^{(i)}(k+j) - \mathbf{r}^{(i)}(k+j)\|^2_{\mathbf{Q}^{(i)}} + \sum_{j=0}^{H_u-1} \|\Delta\mathbf{u}^{(i)}(k+j)\|^2_{\mathbf{R}^{(i)}},$$

with weighting matrices $\mathbf{Q}^{(i)}$ and $\mathbf{R}^{(i)}$ of subsystem i is implemented, that depends on the input and state signals of subsystem i only. During operation of the MPC of subsystem i the optimization problem

$$\min_{\substack{\mathbf{u}^{(i)}(k+j),\\ j=0,\ldots,H_u-1}} J_i(k) \tag{4.56}$$

is solved with constraints. Inequality constraints are available to limit the input and state signals

$$\begin{aligned} \mathbf{u}^{(i)}_{min} &\leq \mathbf{u}^{(i)}(k+j) \leq \mathbf{u}^{(i)}_{max},\ j = 0, \ldots, H_u - 1, \\ \hat{\mathbf{x}}^{(i)}_{min} &\leq \hat{\mathbf{x}}^{(i)}(k+j) \leq \hat{\mathbf{x}}^{(i)}_{max},\ j = 1, \ldots, H_p. \end{aligned}$$

As introduced in Section 4.5 for the AMPC-SL approach a linear approximation

$$\begin{aligned} \hat{\mathbf{x}}^{(i)}(k+j+1) &= \mathbf{A}^{(i)}\hat{\mathbf{x}}^{(i)}(k+j) + \mathbf{B}^{(i)}_u\mathbf{u}^{(i)}(k+j) + \mathbf{B}^{(i)}_s\mathbf{s}^{(i)}(k+j) + \mathbf{B}^{(i)}_d\mathbf{d}^{(i)}(k+j) \\ \hat{\mathbf{y}}^{(i)}(k+j) &= \mathbf{C}^{(i)}\hat{\mathbf{x}}^{(i)}(k+j), \end{aligned}$$

of the MTI model (4.53) and (4.54) is used during optimization to predict the future behavior of the subsystem. The linear approximation is computed by the linearization algorithm for MTI systems derived in Section 3.3, resulting in the linear parameter matrices $\mathbf{A}^{(i)}$, $\mathbf{B}^{(i)}_u$, $\mathbf{B}^{(i)}_s$, $\mathbf{B}^{(i)}_d$ and $\mathbf{C}^{(i)}$ of subsystem i. Predictions on the internal disturbances $\hat{\mathbf{S}}^{(i)}(k) = \begin{pmatrix} \mathbf{s}^{(i)}(k) & \cdots & \mathbf{s}^{(i)}(k+H_p) \end{pmatrix}^T$ and external $\hat{\mathbf{D}}^{(i)}(k) = \begin{pmatrix} \mathbf{d}^{(i)}(k) & \cdots & \mathbf{d}^{(i)}(k+H_p) \end{pmatrix}$ disturbances are provided for the controller node for the whole prediction horizon, such that they can be considered during optimization. Additionally a reference trajectory

$$\hat{\mathbf{R}}^{(i)}(k) = \begin{pmatrix} \mathbf{r}^{(i)}(k+1) & \mathbf{r}^{(i)}(k+2) & \cdots & \mathbf{r}^{(i)}(k+H_p) \end{pmatrix}$$

for the state trajectory is given. The solution of the optimization problem (4.56) is the optimal input trajectory at time step k

$$\hat{\mathbf{U}}^{(i)} = \begin{pmatrix} \hat{\mathbf{u}}^{(i)}(k) & \hat{\mathbf{u}}^{(i)}(k+1) & \cdots & \hat{\mathbf{u}}^{(i)}(k+H_u-1) \end{pmatrix}.$$

As it is typical for MPC only the first input $\hat{\mathbf{u}}^{(i)}(k)$ is given to the plant, since in the next time step the optimization is computed again with updated plant conditions. But the other part of the input sequence is used here, too. The future subsystem behavior is predicted by a forward simulation of the MTI model of the subsystem with the input trajectory for the whole prediction horizon. After the end of the control horizon the input signal is held constant till the end of the prediction horizon

$$\hat{\mathbf{u}}^{(i)}(k+j) = \hat{\mathbf{u}}^{(i)}(k+H_u-1),\ j = H_u, \ldots, H_p.$$

This results in the predicted output trajectory at time k

$$\hat{\mathbf{Y}}^{(i)}(k) = \begin{pmatrix} \hat{\mathbf{y}}^{(i)}(k+1) & \hat{\mathbf{y}}^{(i)}(k+2) & \cdots & \hat{\mathbf{y}}^{(i)}(k+H_p) \end{pmatrix}.$$

To use this control strategy, the trajectories have to be initialized once with reasonable values. The input as well as the output trajectory are outputs of the controller node of subsystem i, such that information on the subsystem can be transmitted to other subsystems. The local control loop with controller node and subsystem is depicted in Figure 4.23.

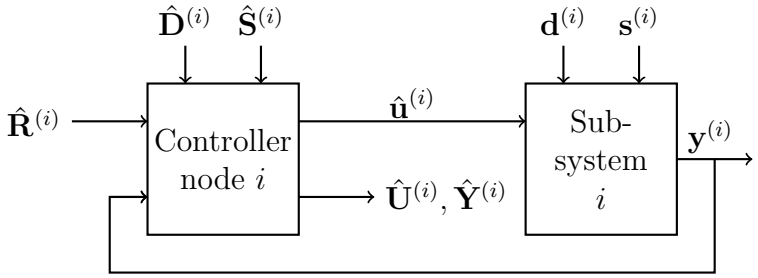


Figure 4.23: Local control loop of subsystem i

Each subsystem optimizes its own cost function J_i. Because of that, only the inputs of the i^{th} subsystem have to be determined, such that the search space and thus the complexity of the optimization problem is reduced significantly compared to the central problem. Furthermore it may be possible to solve some of the problems in parallel. The global control goal of the overall system is given by the overall cost function J that results from the sum of the cost functions of the subsystems

$$J(k) = \sum_{i=1}^{N_{sub}} J_i(k).$$

Thus, each controller node optimizes one part of the cost function $J_i,\ i = 1, \ldots, N_{sub}$, such that the global control goal is considered too. So far the connections between the subsystems are not considered. A network based communication is used here, since network protocols like BACnet are often used in buildings. The transmission of the future input and output trajectories has to be established only between subsystems that are connected. The necessary signals are send to the network. The data transmission is organized by a coordinator. The coordinator receives the necessary prediction of the inputs and outputs

from the subsystems. The coordinator knows the coupling equation (4.55) and computes the predictions of the internal disturbances

$$\begin{pmatrix} \mathbf{s}^{(1)}(k+j) \\ \vdots \\ \mathbf{s}^{(N_{sub})}(k+j) \end{pmatrix} = \mathbf{h}(\hat{\mathbf{y}}^{(1)}(k+j), \ldots, \hat{\mathbf{y}}^{(N_{sub})}(k+j), \mathbf{u}^{(1)}(k+j), \ldots, \mathbf{u}^{(N_{sub})}(k+j)),$$

with $i = 1, \ldots, N_{sub}$ and $j = 1, \ldots, H_p$, that are sent afterwards to the particular subsystems via the network. Additionally the coordinator also provides the reference trajectories $\hat{\mathbf{R}}^{(i)}$ for the subsystems. The interface of the coordinator is shown in Figure 4.24.

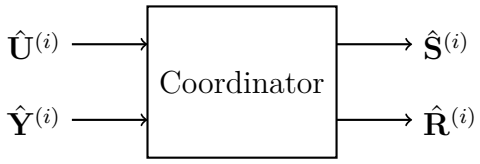


Figure 4.24: Coordinator for the DMPC

The overall structure of the control system is composed of the local controllers of the subsystems and the coordinator. Figure 4.25 shows the system setup containing all proposed components.

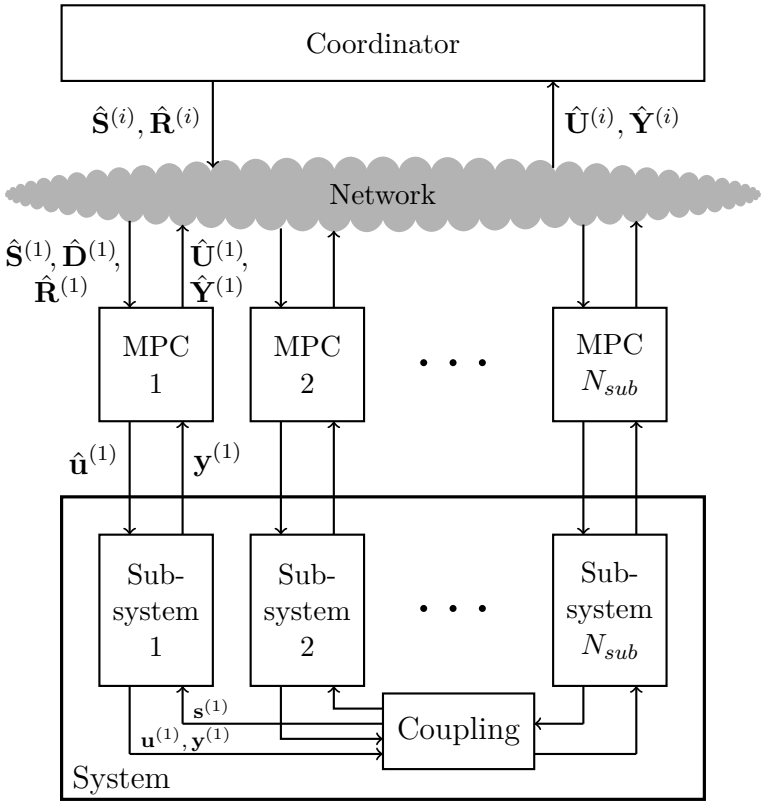


Figure 4.25: Overall controller setup for the DMPC

4.7 Open questions

This section showed, how model-based controller design methods are applied to multilinear systems in decomposed tensor representation. It has been derived, that nonlinear design methods, like the feedback linearization can be specialized to the multilinear structure. Furthermore, it has been described how the applicability of linear methods is extended by adapting the linear models using the MTI system descriptions. For large-scale systems distributed approaches have been investigated to find a sparse controller structure and to split the controller task to several nodes.

The feedback linearization method has been derived for SISO plants assuming no disturbances or model mismatches. If these assumptions are not fullfilled, as it is the case in applications this would lead to an additional control error. Therefore the controller approach would have to be extended e.g. for disturbance rejection as proposed in [90]. Additionally, in practice it is difficult to tune the controller, i.e. the desired linear behavior, such that all plant constraints are fulfilled, e.g. because of input saturations. Furthermore when thinking of real world heating systems, plants often have multiple inputs and multiple outputs, that have to be controlled. This has to be considered in the design method, too. Up to this point the feedback linearization method has been introduced for SISO MTI systems here. For general nonlinear systems this approach was extended to MIMO systems, [94]. The tensor feedback linearizaton technique for MTI systems has to be adapted to that, which is future work.

Two more design methods, the decentralized feedback design and MPC have been investigated here. For both methods optimization problems have to be solved; on the one hand for controller design and on the other hand during operation. These optimization problems are solved very efficiently, if linear models are used. Because of that, in this work the MTI models are linearized around current operating conditions and the optimization problems are solved with the adapted linear model to preserve these good properties. But by linearizing the multilinear models, an approximation of the model is used. Thus, it would be beneficial, if the MTI model is used directly without a linearization, since the plant dynamics are captured better by the MTI model. Therefore, optimization algorithms for multilinear equations have to be derived, that use the special multilinear structure of the right hand sides of the state space models and the memory efficient decomposed tensor representations.

Another open point is the solution of systems of multilinear equations as for the computation of operating point by (4.31), where the right hand side of the state equation is set to zero. Here standard nonlinear solvers have been used. For linear systems of equations well established algorithms exist, [14]. One possibility to solve a system of multilinear equations could be the alternating least squares (ALS) method, where iteratively all but one variable is fixed and the resulting linear problem is solved, [44]. This approach uses the special multilinear structure of the equations but further research is necessary.

If a sparse feedback controller structure is found with the approach of Section 4.3 this opens the question, how this structure can be used for other distributed design methods like DMPC. With the sparse feedback method, a control structure is determined automatically by solving an optimization problem. It has to be investigated, if the resulting structure is optimal for

static feedback only or if it can be transferred to other distributed controller methods. For the proposed DMPC approach the control structure is not determined automatically. It follows heuristically from the basic structure of the plant. But this choice may be not optimal. Thus it has to be investigated, if the results of the sparse feedback control may help here, too.

For the DMPC approach the central control task is split to several controller nodes according to the plant structure. The particular controllers solve their local control problems and communicate their results to adjacent nodes. This showed good results for a heating systems example which will be presented in Section 5.3.4.
A more sophisticated algorithm might improve the distributed solution of the optimization problem. A possible approach, that is used in many distributed predictive control applications is the ADMM algorithm, that allows to solve a decomposed optimizaton problem, [13]. The basic structure of the underlying control problem for the predictive control approach used here fits to the problem description of ADMM. The overall cost function is a sum of convex cost functions of the particular subsystems and the constraints are also a sum of linear constraints of the subsystems. This indicates, that the ADMM algorithm can be used to solve the proposed DMPC problem in a distributed way. The realization is further research.

5 Application of decomposed MTI systems in heating systems

The previous chapters presented several tools and controller design techniques for MTI state space models. To use these models also for large-scale systems the model representation with decomposed tensors was focused on. In this chapter the controller design methods introduced in Chapter 4 are applied to examples in the field of heating systems of non-residential buildings by using tools of Chapters 2 and 3. Therefore Section 5.1 derives models of two example systems in the area of buildings, i.e. a large non-residential heating system and an HVAC system. The model representations of both examples with the four decomposition techniques is investigated in Section 5.2. Section 5.3 applies the different controller design methods to control the room temperature of buildings. The methods are applied to single heating circuits by using the feedback linearization and the AMPC-SL design as well as to large-scale buildings with the decentralized feedback and the DMPC approach. Finally, the AMPC-SL controller is implemented on real-time hardware and applied to a real world office building.

5.1 Application systems

In the following two typical building technology examples are presented, that are inherently multilinear. The examples are used to apply the decomposed MTI model representation of Chapter 3 and the controller design methods of Chapter 4. The first example in Section 5.1.1 is a model of a large-scale heating system with typical structure, where the complexity can be adjusted by setting the number of heat generation units and consumers, i.e. boilers and heating circuits. Another typical example in the area of buildings is a model of a complex HVAC system given in Section 5.1.2. The implementation of the methods for modeling and controller design with MTI systems in the MTI toolbox are described in Section 5.1.3.

5.1.1 Large non-residential heating system

A structure that can be found in many heating systems for non-residential buildings is investigated here to derive a model that should be used to apply the methods for MTI system representation and controller design introduced in the Chapters 3 and 4. The structure of the model is developed according to a large office building, where measurement data is available. The heat is generated by two boilers, that supply the consumer, i.e. the building. The building is divided into several heating circuits, one for each floor. The considered

building consists of 7 circuits. Inside each circuit the supply temperature T_s is adapted to the individual needs of the particular heating circuit. It is assumed that the constructions of the circuits are similar. The cold return water with temperature T_r is fed into a collector and the mixed water gets back to the boilers. In the investigated building the heating system consists of 2 boilers and 7 heating circuits. In the following a more general model structure is developed, such that the number of boilers and consumers can be set by parameters N_B and N_{HC}. This more flexible structure helps to define models of different complexity for the following controller design tasks. Therefore, the single components are modeled individually first and are connected to the overall system afterwards. The structure of the overall heating system is shown in Figure 5.1.

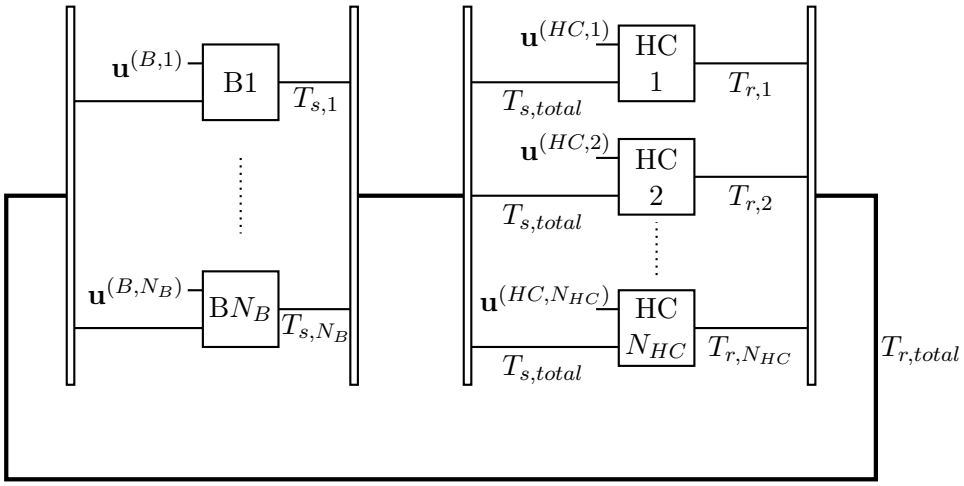


Figure 5.1: Scheme of the heating circuit with N_B boilers (B) and N_{HC} heating circuits (HC)

Consumer

Each heating circuit is modeled as individual consumer. The thermal behavior is modeled by heat balances as in [85]. All rooms in a heating circuit are summarized to one large zone, where it is assumed, that the air inside the zone is completely mixed with homogeneous room or building temperature $T_{b,i}$. This results in a one zone model for the consumer in each heating circuit. The heat demands of the consumers are covered by radiators. The heat transfer from the radiators to the zone air is assumed to be proportional to the difference between the room and the return temperatures $T_{r,i}$ of the radiators with the heat transfer coefficients $k_{rb,i}$. The thermal losses to the environment with the ambient temperature T_o are given proportional to the difference of the room and the ambient temperatures with heat transfer coefficients $k_{bo,i}$. The heat gain due to solar radiation $\dot{Q}_{solar}$ is considered with factor $k_{s,i}$. This leads to the heat balances

$$\dot{T}_{b,i} = \frac{k_{rb,i}}{C_{b,i}}(T_{r,i} - T_{b,i}) - \frac{k_{bo,i}}{C_{b,i}}(T_{b,i} - T_o) + \frac{k_{s,i}}{C_{b,i}}\dot{Q}_{solar}, \tag{5.1}$$

where $C_{b,i}$ are the heat capacities of the heating circuits $i = 1, \ldots, N_{HC}$. The heat transfer between the zones are neglected here, since it is assumed that effects on the room tem-

peratures of adjacent zones are small. The consumer is supplied with warm water with temperature $T_{s,HCi}$ and flow $\dot{V}_{HCi}$. Heat is transferred to the building, such that cooled water leaves the radiators at return temperatures $T_{r,i}$ with rates

$$\dot{T}_{r,i} = \frac{1}{V_{c,i}} \dot{V}_{HCi} \left(T_{s,HCi} - T_{r,i} \right) - \frac{k_{rb,i}}{c\rho V_{c,i}} \left(T_{r,i} - T_{b,i} \right), \tag{5.2}$$

where $V_{c,i}$ are the volumes of the water in the consumer with density ρ and specific heat capacity c. The consumer and the heat transfers are illustrated in Figure 5.2.

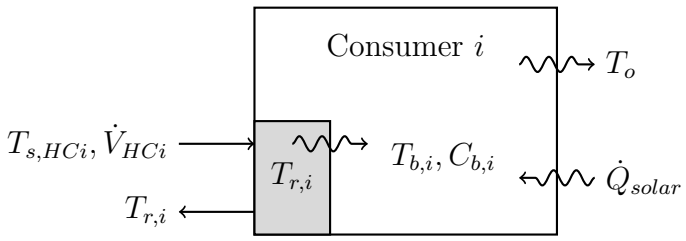


Figure 5.2: Consumer with heat transfers

Mixing circuit

To adapt the supply temperature $T_{s,total}$ of the total supply coming from the boilers to the particular heating circuit, each heating circuit has a three way valve and a pump. The valve allows a return flow addition to reduce the supply temperature for the individual heating circuit, which is shown in Figure 5.3.

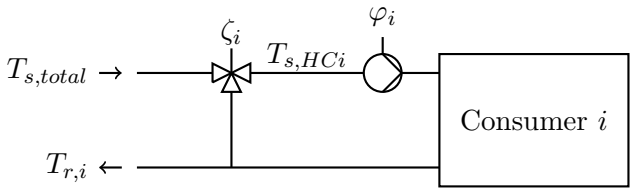


Figure 5.3: Mixing circuit of the consumer

The three way valves are controlled by a signal $\zeta_i \in [0, 1]$ such that the radiators are supplied by the mixing temperatures

$$T_{s,HCi} = \zeta_i T_{r,i} + (1 - \zeta_i) T_{s,total}. \tag{5.3}$$

The flows in the consumer circuit are determined by pumps

$$\dot{V}_{HCi} = \varphi_i \dot{V}_{max,i}, \tag{5.4}$$

depending on the input signals $\varphi_i \in [0, 1]$ with maximal flows $\dot{V}_{max,i}$. The flow $(1-\zeta_i)\dot{V}_{HCi}$ leaves the heating circuit with return temperature $T_{r,i}$.

Boiler

The boilers supply the consumers with heat. The mixed return flows of the consumers reach the boiler with temperature $T_{r,total}$. Each boiler has an input, the modulation signal $\alpha_i \in [0,1]$ with $i = 1\ldots, N_B$, that controls their thermal powers $P_{in,i} = \alpha_i P_{max,i}$, where $P_{max,i}$ are the maximal powers. Building the heat balances of the boilers with volumes $V_{b,i}$, their supply temperatures $T_{s,i}$ are given by the ODE

$$\dot{T}_{s,i} = \frac{1}{V_{b,i}}\dot{V}_{b,i}\left(T_{r,total} - T_{s,i}\right) - \frac{k_{b,i}}{c\rho V_{b,i}}\left(T_{s,i} - T_{amb}\right) + \alpha_i \frac{P_{max,i}}{c\rho V_{b,i}}, \tag{5.5}$$

with the assumption, that the water inside the boiler is perfectly mixed at temperature $T_{s,i}$. The flow through the boiler is denoted by $\dot{V}_{b,i}$. No flow losses should occur. The heat losses of the boiler to its ambient with temperature T_{amb} is assumed to be proportional to the temperature difference with factor $k_{b,i}$. The boiler with its inputs and outputs is depicted in Figure 5.4.

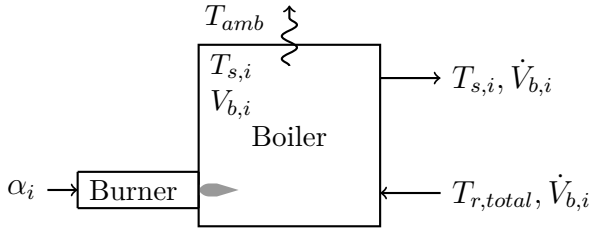


Figure 5.4: Boiler structure

Flow distribution

The flows on the generation side of the three way valves coming from the consumers are connected to a return collector resulting in

$$\dot{V}_{total} = \sum_{j=1}^{N_{HC}} (1-\zeta_j)\dot{V}_{HCj}, \tag{5.6}$$

reaching the boilers with the mixing temperature

$$T_{r,total} = \frac{\sum\limits_{j=1}^{N_{HC}} T_{r,j}(1-\zeta_j)\dot{V}_{HCj}}{\sum\limits_{j=1}^{N_{HC}} (1-\zeta_j)\dot{V}_{HCj}} = \frac{1}{\dot{V}_{total}} \sum_{j=1}^{N_{HC}} T_{r,j}(1-\zeta_j)\dot{V}_{HCj}. \tag{5.7}$$

It is assumed, that the volume flows through the boilers exactly fits to the flow demand by the consumer pumps and that the boilers are hydraulically balanced, such that they all have the same flow

$$\dot{V}_{b,i} = \frac{1}{N_B}\dot{V}_{total}, \tag{5.8}$$

with the total flow $\dot{V}_{total}$ coming from the consumers. The consumers are supplied by a volume flow with warm water coming from the boilers. Since all boilers have the same flow, the total supply temperature of the consumers is the mixing temperature of all boiler temperatures $T_{s,i}, i = 1, \ldots, N_B$

$$T_{s,total} = \frac{\sum_{i=1}^{N_B} \dot{V}_{b,i} T_{s,i}}{\sum_{i=1}^{N_B} \dot{V}_{b,i}} = \frac{1}{N_B} \sum_{i=1}^{N_B} T_{s,i}.$$

Overall heating system

In the previous parts of this section the particular components of a typical heating system of a non-residential building were introduced. Here the components should be connected now to build an overall heating system as depicted in Figure 5.1. The heat generation units are connected via the flow distributions with the consumer circuits, that consist of a mixing circuit for return flow addition and a one zone building model of the particular section of the building. This results in a generic model, where the number N_{HC} of heat generation units and the number N_B of heating circuits can be adjusted by parameters. Thus this model can be adapted to different buildings with this typical heating system structure.

The flow through the boilers follows from the pump and the valve position by inserting (5.4) and (5.6) into (5.8)

$$\dot{V}_{b,i} = \frac{1}{N_B} \sum_{j=1}^{N_{HC}} (1 - \zeta_j) \dot{V}_{max,j} \varphi_j.$$

With that and the definition of the overall return temperature (5.7) the supply temperature dynamics of the boiler are given by

$$\begin{aligned} \dot{T}_{s,i} =& \frac{1}{V_{b,i} N_B} \sum_{j=1}^{N_{HC}} \left(T_{r,j} (1 - \zeta_j) \dot{V}_{max,j} \varphi_j - (1 - \zeta_j) \dot{V}_{max,j} \varphi_j T_{s,j} \right) \\ & - \frac{k_{b,i}}{c \rho V_{b,i}} (T_{s,i} - T_{amb}) + \alpha_i \frac{P_{max,i}}{c \rho V_{b,i}}. \end{aligned} \tag{5.9}$$

The differential equation of the building temperature (5.1) is not effected directly by the connection of the subsystems. The supply temperature of the consumer is set by the three way valve in the mixing circuit, that is supplied by the total supply temperature coming from the boilers. Thus, the supply temperature (5.3) is written as

$$T_{s,HCi} = \zeta_i T_{r,i} + (1 - \zeta_i) \frac{1}{N_{HC}} \sum_{i=1}^{N_B} T_{s,i}$$

and inserted together with (5.4) into (5.2), leading to the differential equation of the return temperature of the consumer

$$\dot{T}_{r,i} = \frac{\dot{V}_{max,i}}{V_{c,i}} \varphi_i \left(\zeta_i T_{r,i} + (1 - \zeta_i) \frac{1}{N_{HC}} \sum_{i=1}^{N_B} T_{s,i} - T_{r,i} \right) - \frac{k_{rb,i}}{c \rho V_{c,i}} (T_{r,i} - T_{b,i}). \tag{5.10}$$

Thus the dynamics of the heating system are described by a system of $N_B + 2N_{HC}$ first order differential equations for the boilers (5.9) and the consumers (5.10) and (5.1).

5.1.2 Heating, ventilation and air-conditioning (HVAC) system

A component that is found in many heating systems is a HVAC unit. Here a model of an HVAC system is described, that was introduced in [106]. Mathematical descriptions of different parts of HVAC systems are derived and an example model is developed with the purpose to test and analyze new control strategies for energy efficient operation of the HVAC system, [106]. Here the model should be used to check the applicability of MTI models in decomposed tensor structure in the area of heating systems. Because of that, a system that can be found in many buildings is chosen here. The model consists of an AHU with a fan, a heater, a humidifier and ductwork. As load or consumer a room is connected to the AHU. The goal is to condition the room air, i.e. to control its temperature and humidity ratio, such that they stay in their comfort ranges. Thus, the temperature as well as the humidity of the HVAC system are modeled here. This is done by mass and energy balances. A HVAC unit can be used for heating or cooling purposes. As an example to show the applicability of MTI systems here, the winter operation of the HVAC system is focused on, which means heating and humidification. The setup of the system is depicted in Figure 5.5.

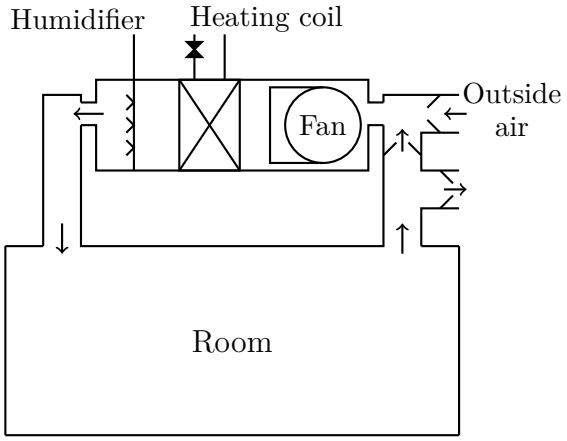


Figure 5.5: Setup of the HVAC system example,[106]

The figure shows the connection of the AHU and the room. The AHU supplies the room with warm air. A part of the room air is removed by the AHU and goes through a mixing box, where some fresh outside air is added. Inside the AHU a fan generates an air flow. The air is heated by a heating coil and moisture is added by a humidifier. After that the air gets back into the room. To simulate a load for the AHU, a one zone model of a room is used. The room with its heat transfers is shown in Figure 5.6.

It is assumed, that the air in the room is fully mixed at the same temperature T_b and same humidity ratio W_b in the whole room. The heat capacity of the room is composed of the room interior, the 4 walls and the roof. For the walls the assumption is made that

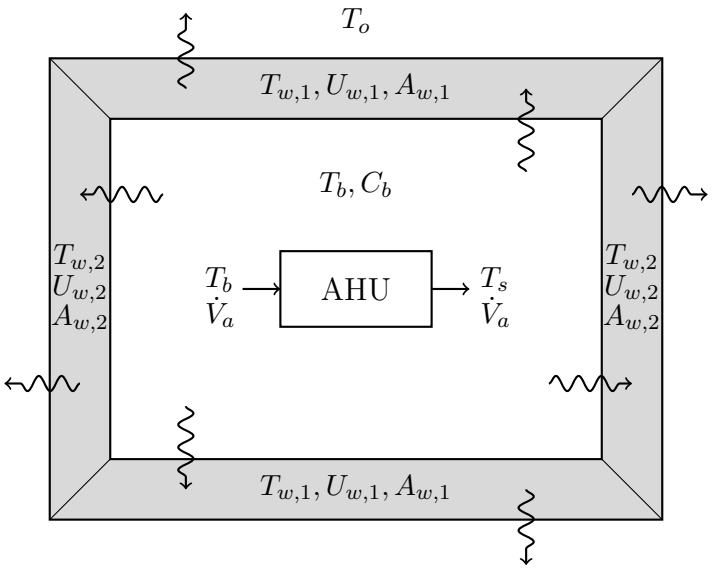


Figure 5.6: Top view of the room with its modeled heat transfers (Heat transfer through the roof is not depicted here)

the northern and southern walls have the same influence on the room temperature. The heat transfer by conduction or convection between the room and the wall with area $A_{w,1}$ is modeled proportionally to the difference of room and wall temperature $T_{w,1}$ with heat transfer coefficient $U_{w,1}$. The same holds for the eastern and western walls with temperatures $T_{w,2}$, areas $A_{w,2}$ and heat transfer coefficients $U_{w,2}$ as well as for the roof with temperature T_R, area A_R and heat transfer coefficient U_R. The ground has no effect on the room temperature. Coming from the AHU, warm air with temperature $T_{s,a}$ enters the room with flow $\dot{V}_a$. No flow losses should occur, such that air with same flow at room temperature T_b is removed by the AHU. The room interior with e.g. air or furniture has the heat capacity C_b. With these influencing factors the building or room temperature T_b is computed by the heat balance

$$\begin{aligned}\dot{T}_b =& \frac{\rho_a c_a}{C_b}\dot{V}_a(T_{s,a} - T_b) + 2\frac{U_{w,1}A_{w,1}}{C_b}(T_{w,1} - T_b) + \frac{U_R A_R}{C_b}(T_R - T_b) \\ &+ 2\frac{U_{w,2}A_{w,2}}{C_b}(T_{w,2} - T_b) + \frac{1}{C_b}\dot{Q}_{\text{occupant}},\end{aligned} \tag{5.11}$$

where the density and the heat capacity of the air are denoted by ρ_a and c_a, respectively. Heat gains resulting from e.g. occupants and lights are summarized in the disturbance input $\dot{Q}_{\text{occupant}}$. Due to the temperature difference between the room and the outside with temperature T_o, heat is removed through the walls and roof. The heat transfer to the outside is again assumed to be proportional to the temperature difference between walls or roof and the outside. The walls and the roof have a heat capacities $C_{w,1}$, $C_{w,2}$ and C_R, leading to the

heat balance for the wall and roof temperatures

$$\dot{T}_{w,1} = \frac{U_{w,1}A_{w,1}}{C_{w,1}}(T_b - T_{w,1}) + \frac{U_{w,1}A_{w,1}}{C_{w,1}}(T_o - T_{w,1}), \tag{5.12}$$

$$\dot{T}_{w,2} = \frac{U_{w,2}A_{w,2}}{C_{w,2}}(T_b - T_{w,2}) + \frac{U_{w,2}A_{w,2}}{C_{w,2}}(T_o - T_{w,2}), \tag{5.13}$$

$$\dot{T}_R = \frac{U_R A_R}{C_R}(T_b - T_R) + \frac{U_R A_R}{C_R}(T_o - T_R). \tag{5.14}$$

This describes the thermal behavior of the room. The humidity ratio W_b of the room air is computed by a mass balance

$$\dot{W}_b = \frac{1}{V_b}\dot{V}_a(W_s - W_b) + \frac{1}{\rho_a V_b}P_{\text{occupant}}. \tag{5.15}$$

The airflow with humidity ratio W_s enters the room which has the volume V_b. The same flow with humidity W_b is removed from the room by the AHU. Occupants add moisture, which is captured by the disturbance input P_{occupant}.

Air is removed from the room by the AHU, that consist of ducts, a mixing box, a fan, a heating coil and a humidifier. The components and their connection are shown in Figure 5.7 and described in the following.

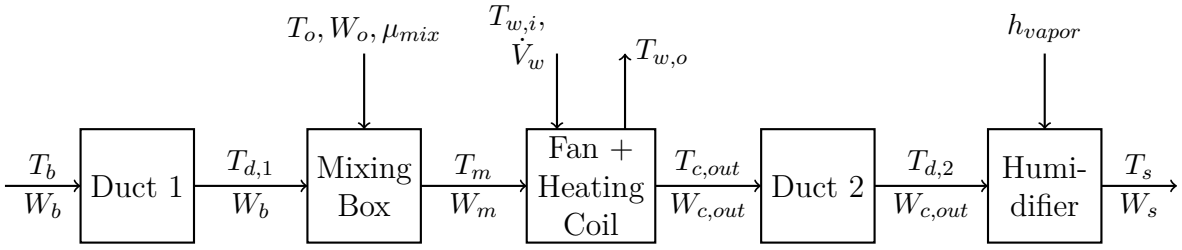


Figure 5.7: Air handling unit

At first, the room is connected with the AHU by a duct. The thermal behavior of the duct is modeled as first order system, such that the temperature $T_{d,1}$ of the flow leaving the duct is given by

$$\dot{T}_{d,1} = \frac{(h_i + h_o)\rho_a c_a}{h_i M_d C_d}\dot{V}_a(T_b - T_{d,1}), \tag{5.16}$$

with the heat transfer coefficients inside the duct h_i and to the ambient h_o, the duct mass M_d and the specific heat of its material C_d. In the mixing box fresh outside air is added to the air flow and part of the used air is removed with the same flow than fresh air is added. The ratio $\mu_{mix} \in [0, 1]$ describes how much of the air coming from the room is reused, i.e. $\mu_{mix} = 1$ means that no fresh air is added and $\mu_{mix} = 0$ means only fresh air with temperature T_o and humidity ratio W_o is used. Heat and mass balances lead to the temperature and humidity ratio

$$T_m = \mu_{mix}T_{d,1} + (1 - \mu_{mix})T_o, \tag{5.17}$$

$$W_m = \mu_{mix}W_b + (1 - \mu_{mix})W_o, \tag{5.18}$$

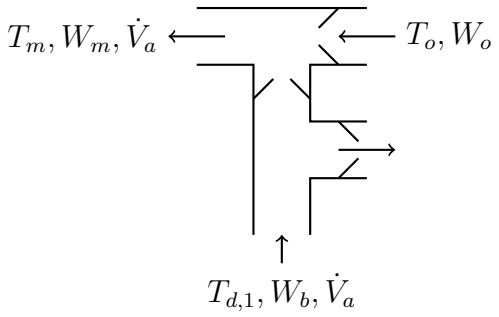


Figure 5.8: Mixing box

of the flow leaving the mixing box as shown in Figure 5.8.

A fan generates the air flow $\dot{V}_a$. To heat up the air, a heating coil is installed with a water to air heat exchanger, that is supplied on the water side with a flow $\dot{V}_w$ of warm water at temperature $T_{w,i}$. The water leaves the heating coil at temperature $T_{w,o}$. On the air side heat is added to the air flow, such that the temperature $T_{c,out}$ of the air leaving the heating coil is given by

$$\dot{T}_{c,out} = \frac{c\rho}{C_h}\dot{V}_w(T_{w,i} - T_{w,o}) + \frac{U_c A_c}{C_h}(T_o - T_{c,out}) + \frac{\rho_a c_a}{C_h}\dot{V}_a(T_m - T_{c,out}), \tag{5.19}$$

with coil heating capacity C_h, heat transfer coefficient U_c and area A_c to the outside. The dynamics of the water boiler and the heat exchanger are not modeled in detail here. It is assumed that the water temperatures can be set as desired by a controller, such that they are used as inputs for the model here. Since the heating coil has some volume V_c the humidity ratio $W_{c,out}$ of the air leaving the component is computed by the mass balance

$$\dot{W}_{c,out} = \frac{\dot{V}_a}{V_c}(W_m - W_{c,out}). \tag{5.20}$$

A schematic of the heating coil is shown in Figure 5.9.

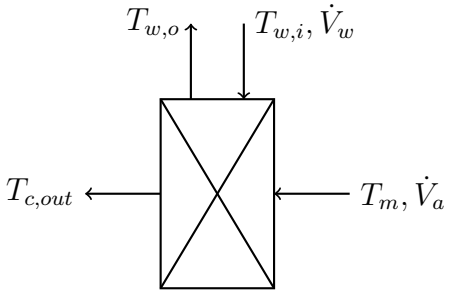


Figure 5.9: Heating coil

Next, the air passes a second duct, that is modeled as the one before as first order system

resulting in the heat balance

$$\dot{T}_{d,2} = \frac{(h_i + h_o)\rho_a c_a}{h_i M_d C_d} \dot{V}_a (T_{c,out} - T_{d,2}), \tag{5.21}$$

for the air temperature $T_{d,2}$ at the end of the duct. After the regulation of the temperature by the heating coil, the humidity ration of the air is increased by the humidifier. Inside this component a mass transfer of water vapor to the air occurs. Air with humidity ratio $W_{c,out}$ enters the humidifier and leaves the AHU with humidity W_s, that is described by the mass balance

$$\dot{W}_s = \frac{1}{V_h} \dot{V}_a (W_{c,out} - W_s) + \frac{1}{\rho_a V_h} h_{vapor}, \tag{5.22}$$

where V_h is the volume of the humidifier and h_{vapor} the rate of moisture air produced by the humidifier which is used as input. Inside the humidifier some temperature losses to the environment occur by the surface area A_h with heat transfer coefficient U_h, such that the temperature of the air entering the room is given by

$$\dot{T}_s = \frac{c_a \rho_a}{C_h} \dot{V}_a (T_{d,2} - T_s) + \frac{U_h A_h}{C_h} (T_o - T_s), \tag{5.23}$$

with the heat capacity C_h of the humidifier. It is assumed that the actors of the system do not react instantaneously to changes in the inputs. The user or controller set a desired reference value. It is assumed that the actors reach these setpoint values perfectly but with some dynamics that are modeled by first order systems. Therefore, the actor dynamics for the control signals $\dot{V}_a$, μ_{mix}, $\dot{V}_w$, $T_{w,i}$, $T_{w,o}$ and h_{vapor} are given by

$$\begin{aligned}
\ddot{V}_a &= \tfrac{\dot{V}_{a,set}}{\tau_{\dot{V},a}} - \tfrac{\dot{V}_a}{\tau_{\dot{V},a}}, & \dot{\mu}_{mix} &= \tfrac{\mu_{set}}{\tau_\mu} - \tfrac{\mu_{mix}}{\tau_\mu}, \\
\ddot{V}_w &= \tfrac{\dot{V}_{w,set}}{\tau_{\dot{V}w}} - \tfrac{\dot{V}_w}{\tau_{\dot{V}w}}, & \dot{T}_{w,i} &= \tfrac{T_{w,i,set}}{\tau_{Twi}} - \tfrac{T_{w,i}}{\tau_{Twi}}, \\
\dot{T}_{w,o} &= \tfrac{T_{w,o,set}}{\tau_{Two}} - \tfrac{T_{w,o}}{\tau_{Two}}, & \dot{h}_{vapor} &= \tfrac{h_{vapor,set}}{\tau_{hvapor}} - \tfrac{h_{vapor}}{\tau_{hvapor}}
\end{aligned} \tag{5.24}$$

with time constants $\tau_{\dot{V},a}$, τ_μ, $\tau_{\dot{V}w}$, τ_{Twi}, τ_{Two} and τ_{hvapor}. Combining the equations (5.11) to (5.24) leads to a system of first order differential equations with multilinear right hand sides.

5.1.3 MTI Toolbox

The previous sections introduced many methods for modeling and controller synthesis for decomposed MTI systems. To apply these methods in a user-friendly way, the MTI toolbox for MATLAB and Simulink was developed in the scope of this thesis. The developed toolbox offers the possibility to have a simple access to the application of the derived algorithms. The tensor decomposition methods can be applied to MTI systems easily with this toolbox, since the algorithms use the tensor methods of the popular tensor toolboxes from the mathematical community, [5, 45, 84, 109]. Thus this toolbox connects the tensors toolboxes developed in

mathematics with the methods for modeling and controller design introduced in this thesis. This significantly simplifies their application. The toolbox is implemented in an object oriented way. Different tasks in the work with MTI systems are structured in different classes with their attributes and methods. The main parts of the toolbox are:

- *Tensor operations*: Tensor operations like transformations between decomposed representations, i.e. from CP to Tucker or TT or special tensor operations for CP tensors like contracted product or outer product.
- *Polynomial operations*: Methods for representation of polynomials of order N and operations like multiplication and partial derivatives based on operational tensors.
- *Modeling*: Representation of decomposed MTI systems in continuous- and discrete-time.
- *Simulation*: Simulation methods for MTI systems using the decomposed structure of parameter tensors.
- *Controller design*: Design methods for MTI systems like feedback linearization or decentralized design.
- *Simulink*: Simulink library for simulation of MTI systems and their controllers in Simulink environment.
- *Code generation*: Some selected functions with adapted implementation, such that they are supported for code generation to use them in real-time applications.

Thus, the MTI Toolbox offers a lot of tools to work with MTI systems in model-based design from system representation over simulation to controller development, [46]. The main parts of the toolbox are summarized in Figure 5.10.

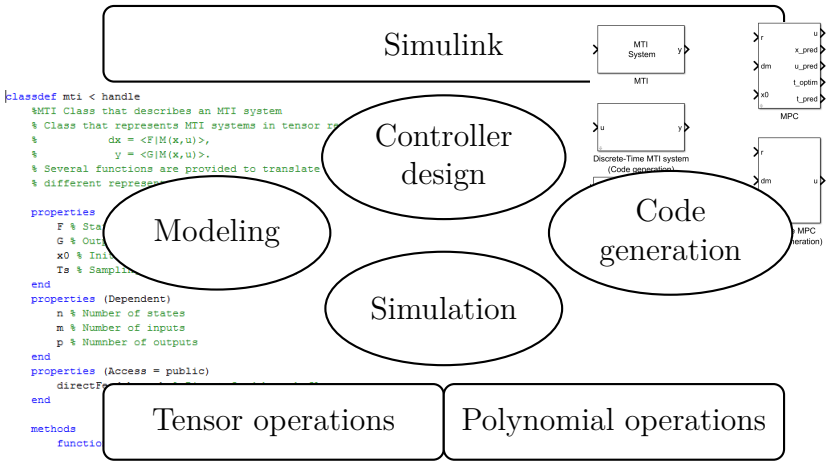


Figure 5.10: MTI toolbox overview

5.2 Representation as decomposed MTI systems

Section 3.2 showed that MTI systems can be represented in all four proposed decomposition techniques and simulation as well as linearization are possible, too. In the following the application of the representations to the two heating system examples of Section 5.1 are compared according to several criteria. Therefore as reference system the exact CP representation of the system

$$\dot{\mathbf{x}} = \langle\, \mathsf{F} \mid \mathsf{M}(\mathbf{x}, \mathbf{u}) \,\rangle$$

is chosen, build as shown in Example 3.6 resulting in the parameter tensor representation (3.19). The other representations

$$\dot{\mathbf{x}}_{decomp} = \langle\, \mathsf{F}_{decomp} \mid \mathsf{M}(\mathbf{x}_{decomp}, \mathbf{u}) \,\rangle\,,$$

will be compared to the exact one, i.e. the other decomposed parameter tensors and their low-rank approximations. The tensors are rated on the one hand by the Frobenius norm approximation measure

$$\frac{\|\mathsf{F} - \mathsf{F}_{decomp}\|_F^2}{\|\mathsf{F}\|_F^2}$$

of the parameter tensor. On the other hand the dynamical behavior of the systems are compared by investigating the simulation results of a typical system scenario by the measure

$$\int_0^{T_{end}} \sum_{i=1}^{n} \frac{\left(x_i(t) - x_{decomp,i}(t)\right)^2}{x_i(t)^2} dt$$

and by comparing the eigenvalues $\boldsymbol{\lambda}(\mathbf{A})$ of the linearization around a typical operating point $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ computed by (3.46). The memory demand is the last performance measure of the proposed techniques. In the following two subsections, the decomposition techniques are applied to the heating and the HVAC system and finally the general comparison of Section 3.2.5 is enlarged with the results from the application examples.

5.2.1 Heating system

The generic model of a standard heating system derived in Section 5.1.1 with N_B boilers and N_{HC} consumers will be represented here as multilinear state space model and decomposition techniques will be applied to get an efficient format for the model description. From (5.1), (5.9) and (5.10) follows, that each boiler has one state, the supply temperature, and each consumer has two states by the room and return temperatures. The states are arranged in the state vector as follows

$$\mathbf{x} = \begin{pmatrix} T_{s,1} & \cdots & T_{s,N_B} & T_{r,1} & \cdots & T_{r,N_{HC}} & T_{b,1} & \cdots & T_{b,N_{HC}} \end{pmatrix}^T \in \mathbb{R}^{N_B+2N_{HC}}. \tag{5.25}$$

The inputs of the boilers, i.e. their modulation signals, and the consumers, i.e. their pump and valve signals, are summarized in the input vector

$$\mathbf{u} = \begin{pmatrix} \alpha_1 & \cdots & \alpha_{N_B} & \zeta_1 & \cdots & \zeta_{N_{HC}} & \varphi_1 & \cdots & \varphi_{N_{HC}} \end{pmatrix}^T \in \mathbb{R}^{N_B+2N_{HC}}. \tag{5.26}$$

The plant is influenced externally by the weather conditions that are considered here with the outside temperature and the solar radiation. These two signals form the disturbance vector

$$\mathbf{d} = \begin{pmatrix} T_o & \dot{Q}_{solar} \end{pmatrix}^T \in \mathbb{R}^2. \tag{5.27}$$

Inserting this choice of the state, input and disturbance variables into (5.1), (5.9) and (5.10) leads to the the state space model described in the Appendix B.1. When investigating the state equations it gets obvious, that the model is not linear, since multiplications of states and inputs occur. Since no other polynomial or nonlinear terms are included in the state equations the model is multilinear. This means, it is described as MTI system in tensor representation

$$\begin{aligned} \dot{\mathbf{x}} &= \langle\, \mathsf{F} \mid \mathsf{M}\,(\mathbf{x}, \mathbf{u}, \mathbf{d})\, \rangle\,, \\ \mathbf{y} &= \langle\, \mathsf{G} \mid \mathsf{M}\,(\mathbf{x}, \mathbf{u}, \mathbf{d})\, \rangle\,. \end{aligned}$$

Since the model equations have got $2N_B + 4N_{HC} + 2$ variables by the states, inputs and disturbance variables this would lead to, e.g. $(N_B + 2N_{HC})\, 2^{2N_B+4N_{HC}+3}$ elements to be stored for the full representation of F. As example a heating system with one boiler and seven consumers is used. This would mean to store $17 \cdot 2^{37} \approx 2.3 \cdot 10^{12}$ elements, because of its 15 states, 15 inputs and 2 disturbances. Because of this large storing effort, it is not possible to use the full representation here. Thus, the state equations are translated to the CP representation presented in Section 3.2.1, which results the CP format of the parameter tensor

$$\begin{aligned} \mathsf{F} = \Big[&\mathbf{F}_{\dot{Q}_{solar}}, \mathbf{F}_{T_o}, \mathbf{F}_{\varphi_{N_{HC}}}, \ldots, \mathbf{F}_{\varphi_1}, \mathbf{F}_{\zeta_{N_{HC}}}, \ldots, \mathbf{F}_{\zeta_1}, \mathbf{F}_{\alpha_{N_B}}, \ldots, \mathbf{F}_{\alpha_1}, \ldots \\ &\mathbf{F}_{T_{b,N_{HC}}}, \ldots, \mathbf{F}_{T_{b,1}}, \mathbf{F}_{T_{r,N_{HC}}}, \ldots, \mathbf{F}_{T_{r,1}}, , \mathbf{F}_{T_{s,N_B}}, \ldots, \mathbf{F}_{T_{s,1}}, \mathbf{F}_{\Phi} \Big] \cdot \boldsymbol{\lambda}_F. \end{aligned}$$

The same approach is applied to the parameter tensor G of the output equation. Since the state space model consists of 115 terms, i.e. parameters, the factor matrices of all but the last factor matrix $\mathbf{F}_\Phi$ have the dimension $\mathbb{R}^{2\times 115}$. The sizes of last factor matrix $\mathbf{F}_\Phi$ and the weighting vector are given by $\mathbb{R}^{17\times 115}$ and $\mathbb{R}^{115}$. This direct translation of the state equation to the CP representation is used as reference system for the other approaches. At first the CP representation is translated to Tucker format as shown in 3.2.2 resulting in

$$\mathsf{F} = \Big[\mathbf{F}_{\dot{Q}_{solar}}, \mathbf{F}_{T_o}, \mathbf{F}_{\varphi_{N_{HC}}}, \cdots, \mathbf{F}_{\alpha_1}, \mathbf{F}_{\Phi} \Big] \cdot \Lambda_F,$$

with the same factor matrices than the CP format. When translating the CP representation to TT like in Section 3.2.3 the first and the last cores are of dimensions $\mathbb{R}^{1\times 2\times 115}$ and $\mathbb{R}^{115\times 2\times 1}$ respectively. The dimension of all other cores is given by $\mathbb{R}^{115\times 2\times 115}$, which gives the TT representation

$$\mathsf{F} = \Big[\mathsf{F}_{\dot{Q}_{solar}}, \mathsf{F}_{T_o}, \mathsf{F}_{\varphi_{N_{HC}}}, \cdots, \mathsf{F}_{\alpha_1}, \mathsf{F}_{\Phi} \Big].$$

The fourth format used here is the HT representation of Section 3.2.4, where for the direct translation the same factor matrices than for the CP tensor are used as leaf nodes. Furthermore the transfer tensors of the top node $\mathbf{B}_{\dot{Q}_{solar}\cdots\Phi} \in \mathbb{R}^{115\times115}$ and all other transfer tensors $\mathbf{B}_t \in \mathbb{R}^{115\times115\times115}$ have to be stored. This shows, that the MTI model of the heating system can be represented in all four decomposition techniques. They store the parameters exactly, i.e. the tensor error to the reference system is equal to zero. Because of that, the models also show the same dynamical behavior regarding the simulation of a typical scenario or the eigenvalues of a linearization around an operating point. The main difference between the different representations is their storage demand, as shown for the exact representation in Table 5.1. The table shows that especially for the TT and HT representations truncation techniques have to be applied to reduce the size of the factor matrices or cores and thus the memory demand. The goal is to approximate the exact tensor with less elements. But the main dynamics of the system should still be captured by the approximated parameter tensor. A problem in the truncation of the tensor is that the entries equal to zeros in the original tensor are not necessarily zero in the approximation. This could lead to problems when working with unscaled systems, which is illustrated by an example here. E.g. the monomial $T_{s,1}T_{r,2}T_{b,1}$ does not occur in the state equations. Thus, the corresponding parameter should be equal to zero. If after truncation the entry in the parameter tensor has a value slightly different to zero, this has not a great effect on the tensor error. But since temperatures are given in Kelvin the monomial $T_{s,1}T_{r,2}T_{b,1}$ can take very large values, such that the multiplication of the small parameter with this monomial can have a significant effect on the system dynamics, which is not desired. To avoid such numerical effects the decomposed system is scaled as introduced in Section 3.5 before truncation such that all signals are in the range $[0, 1]$. With this scaled model truncated representations are searched with the standard approaches [44, 83, 45] by setting a specific rank for CP or a desired accuracy for TT and HT. A reduction of the storage effort for the Tucker form is not possible since, as mentioned before, a full representation is necessary for the considered algorithms, which is not possible here. The storage effort of the representations should be reduced by using the truncation techniques. Table 5.1 shows the effect of the truncation procedure.

Table 5.1: Memory demand of F of the heating system

	Full	CP	Tucker	TT	HT
Exact	$2.3 \cdot 10^{12}$	10 235	14 605	927 935	53 254 085
Minimal	$2.3 \cdot 10^{12}$	2 581	14 605	17 411	12 499

These techniques focus on the tensor error, i.e. they try to find an approximation of the parameter tensor that fits best to the original one. Here it is not the main goal, that the MTI system with the resulting parameter tensor has a good tensor fit only. Also the desired dynamical behavior, which is tested by the eigenvalues of a linearization and the simulation results of a test scenario, should be maintained after truncation. With these guidelines the number of rank-1 elements could be reduced from 115 to 29 for the CP tensor. The TT and HT representation are approximated with an accuracy of 10^{-3}. The storage effort of the CP,

TT and HT could be reduced enormously and without significant losses in the dynamical behavior of the MTI system.

5.2.2 HVAC system

In Section 5.1.2 the behavior of the HVAC system was described by a system of 17 first order differential equations. As shown in [46] they form a state space model. The state vector of the system is defined as

$$\begin{aligned}\mathbf{x} = \Big(\; & T_b \quad T_{w,1} \quad T_{w,2} \quad T_R \quad W_b \quad T_{d,1} \quad T_{c,out} \quad W_{c,out} \quad T_{d,2} \quad T_s \quad W_s \quad \dot{V}_a \quad \mu_{mix} \quad \cdots \\ & \dot{V}_w \quad T_{w,i} \quad T_{w,o} \quad h_{vapor} \; \Big)^T \in \mathbb{R}^{17}. \end{aligned} \tag{5.28}$$

The control and disturbance inputs are arranged in the vectors

$$\mathbf{u} = \begin{pmatrix} \dot{V}_{a,set} & \mu_{set} & \dot{V}_{w,set} & T_{w,i,set} & T_{w,o,set} & h_{vapor,set} \end{pmatrix}^T, \tag{5.29}$$

$$\mathbf{d} = \begin{pmatrix} \dot{Q}_{\text{occupant}} & P_{\text{occupant}} & T_o & W_o \end{pmatrix}^T. \tag{5.30}$$

Multiplications between different states or states and inputs occur, e.g. because of the multiplication of temperatures and flows like $T_{w,i}\dot{V}_w$ on the water side or $T_b\dot{V}_a$ on the air side of the system. This gets obvious, when inserting the definition of states, inputs and disturbances (5.28) to (5.30) to the state equations (5.11) to (5.24) leading to the state space model shown in Appendix B.2. The room temperature and humidity ration are chosen as outputs

$$\mathbf{y} = \begin{pmatrix} T_b & W_b \end{pmatrix}^T = \begin{pmatrix} x_1 & x_5 \end{pmatrix}^T.$$

The right hand sides of the state space model in B.2 are inherently multilinear. No terms outside the multilinear class occur. Thus, the system is represented as MTI model in tensor format

$$\begin{aligned} \dot{\mathbf{x}} &= \langle\, \mathsf{F} \mid \mathsf{M}\,(\mathbf{x}, \mathbf{u}, \mathbf{d}) \,\rangle\,, \\ \mathbf{y} &= \langle\, \mathsf{G} \mid \mathsf{M}\,(\mathbf{x}, \mathbf{u}, \mathbf{d}) \,\rangle\,, \end{aligned}$$

where F contains all model parameters and the states, control and disturbance inputs are given by (5.28) to (5.30). The model has 17 states and 10 inputs and 54 parameters. In full tensor representation this would lead to $17 \cdot 2^{27} = 2.3 \cdot 10^9$ elements in the parameter tensor F. Thus, decomposition techniques have to be applied to represent this system. With the approach introduced in Section 3.2.1 the HVAC system model is translated to an MTI system in CP representation leading to parameter tensor

$$\begin{aligned} \mathsf{F} = \Big[& \mathbf{F}_{W_o}, \mathbf{F}_{T_o}, \mathbf{F}_{P_{\text{occupant}}}, \mathbf{F}_{\dot{Q}_{\text{occupant}}}, \mathbf{F}_h, \mathbf{F}_{T_{w,o,set}}, \mathbf{F}_{T_{w,i,set}}, \mathbf{F}_{\dot{V}_{w,set}}, \mathbf{F}_{\mu_{set}}, \mathbf{F}_{\dot{V}_{a,set}}, \mathbf{F}_{T_{w,o}}, \mathbf{F}_{T_{w,i}}, \cdots \\ & \mathbf{F}_{\dot{V}_w}, \mathbf{F}_{\mu_{mix}}, \mathbf{F}_{\dot{V}_a}, \mathbf{F}_{W_s}, \mathbf{F}_{T_s}, \mathbf{F}_{T_{d,2}}, \mathbf{F}_{W_{c,out}}, \mathbf{F}_{T_{c,out}}, \mathbf{F}_{T_{d,1}}, \mathbf{F}_{W_b}, \mathbf{F}_{T_R}, \mathbf{F}_{T_{w,2}}, \mathbf{F}_{T_{w,1}}, \mathbf{F}_{T_b}, \mathbf{F}_\Phi \Big] \cdot \boldsymbol{\lambda}_F, \end{aligned}$$

with 27 factor matrices of dimension $\mathbb{R}^{2\times 54}$, factor matrix $\mathbf{F}_\Phi \in \mathbb{R}^{17\times 54}$ and a weighting vector $\boldsymbol{\lambda}_F \in \mathbb{R}^{54}$.

The system equations are written in CP format (3.19), i.e. the reference system. This reference system is compared to the Tucker, TT and HT conversions. The Tucker format is constructed with the same factor matrices

$$\mathsf{F} = \left[\mathbf{F}_{W_o}, \mathbf{F}_{T_o}, \cdots, \mathbf{F}_{T_{w,1}}, \mathbf{F}_{T_b}, \mathbf{F}_{\Phi}\right] \cdot \Lambda_F.$$

The TT is written as

$$\mathsf{F} = \left[\mathsf{F}_{W_o}, \mathsf{F}_{T_o}, \cdots, \mathsf{F}_{T_{w,1}}, \mathsf{F}_{T_b}, \mathsf{F}_{\Phi}\right],$$

where the first and the last cores are of dimensions $\mathbb{R}^{1\times 2\times 54}$ and $\mathbb{R}^{54\times 2\times 1}$ respectively. All other cores have the size $\mathbb{R}^{54\times 2\times 54}$. The translation of the CP representation to HT leads to leave nodes $\mathbf{F}_{W_o}, \mathbf{F}_{T_o}, \ldots, \mathbf{F}_{T_{w,1}}, \mathbf{F}_{T_b}, \mathbf{F}_{\Phi}$, that are equal to the CP factor matrices and 27 transfer tensors $\mathsf{B}_t \in \mathbb{R}^{54\times 54\times 54}$ and the top node $\mathbf{B}_{W_o\cdots\Phi} \in \mathbb{R}^{54\times 54}$. The direct translation is not optimal regarding the storage effort. All decomposition techniques show good results according to system representation. The eigenvalue differences and simulation as well as the tensor error are zero for all systems in exact representation. A significant difference between the representations are their memory demands, i.e. the number of elements to be stored, as shown in Table 5.2.

Table 5.2: Memory demand of F for the HVAC system

	Full	CP	Tucker	TT	HT
Exact	$2.3 \cdot 10^9$	3 834	85 482	152 658	4 100 814
Minimal	$2.3 \cdot 10^9$	3 124	85 482	9 120	7 177

The memory demands of the exact versions show the necessity of truncating Tucker, TT and HT representations of the HVAC model. Truncations of the tensors are computed by standard techniques, [44, 45, 83]. For the CP representation, the number of rank-1 components is reduced from 54 to 44, such that the model still fulfills the tensor and dynamical accuracy. As mentioned in Section 3.2.5 a truncation of already decomposed tensors in Tucker representation is not possible, such that no truncation can be applied here, since the full representation of the parameter tensor is not manageable regarding the storage effort. The TT and HT format can be truncated by setting a desired approximation threshold, which is a great advantage for this application. Both representations could be truncated without any significant loss in the dynamical system behavior, i.e. regarding their eigenvalues or the simulation error. The resulting memory demands of the parameter tensor F after truncation are summarized in Table 5.2.

5.2.3 Comparison of the decomposition methods

The representation of models of a heating system and a HVAC system as MTI model with the four proposed decomposition techniques showed the applicability of the decomposition methods in the area of heating systems. A general comparison of the different decomposed

MTI systems was drawn in Section 3.2.5. Here, this comparison is extended by the experiences gained during application. The models are constructed as described in Section 3.2. At first the state equations are translated term by term to CP representation and converted to the other formats afterwards. For both examples considered here a direct translation is necessary, since no full representations of the parameter tensors are possible due to the system sizes. The full representations need too much memory. But the exact representations of the model are possible in all four decomposed formats, i.e. the tensor error is zero. Because of that, the model representations show the same good dynamical properties as well. The main difference between the methods is their storage effort. Due to the direct translation from CP, as expected, the memory demand of the other formats is very high, especially for TT and HT. With the help of truncation techniques low-rank approximations of the parameter tensors should be found to reduce the storage demand.

For the CP decomposition the ALS algorithm was used to find an approximation of the parameter tensor with less rank-1 elements than the exact representation. Therefore, a desired rank is set and the result is investigated concerning the tensor error and the dynamical properties of the resulting parameter tensor. With that, the number of rank-1 terms are reduced for both examples. But an interesting fact is, that the number of rank-1 components can be reduced for the heating system example from 115 to 29 by a factor of 3.97. For the HVAC system a reduction factor 1.23 from 54 to 44 could be realized only. This shows that the data reduction potential depends on the systems structure. The reason for the different reduction factors achieved, could be that the heating system consists of different subsystem that have the same structure, i.e. several consumers. Furthermore it is assumed, that the consumers show a comparable behavior and because of that their parameters are similar. The truncation results for the CP decomposition show that this allows a representation with much less rank-1 components and leads to good decomposition properties. In the HVAC example the overall model is constructed of subsystems with different structures. Only the duct occurs two times in the model. No more repeating components are included. This complicates the truncation and leads to less compression. But one can say that the CP format has nevertheless a low memory demand for both systems and shows good dynamical properties.

As described in Section 3.2.5 with the truncation algorithms in [5] for Tucker tensors, it is possible to compute Tucker decompositions of full tensors only. A truncation of an already Tucker decomposed tensor directly is not possible. Since the parameter tensors of the two examples are too large for full representation, a further reduction of the memory demand of the Tucker representation by truncation could not be achieved.

The TT and HT representations are constructed by direct translation from the exact CP tensor. Since the direct translation is not optimal regarding the storage effort, this results in a very large storage effort in the first step, which is also the case here for the two examples. But by setting a desired accuracy, truncations are computed. Thus no trial and error approach has to be applied as for the CP case. Setting a desired accuracy directly allows a better handling for finding good low-rank representations. Since the algorithms for truncation of TT and HT are based on SVD and do not work iteratively like for CP, the truncation result is computed much faster. With that, the very large memory demand is reduced significantly even though the memory demand is higher than in the CP case. Before truncation the

memory demand of HT is higher than for TT for both examples, because of the large transfer tensors of dimension $\mathbb{R}^{\Psi_F \times \Psi_F \times \Psi_F}$. The cores of the TT representation of dimension $\mathbb{R}^{\Psi_F \times 2 \times \Psi_F}$ are smaller. But the reduction for HT is enormous such that after truncation the HT representation shows a lower number of elements to be stored compared to the TT format. But their storage demands are both in the same order of magnitude. It was shown for the CP decomposition, that the reduction factor for the number of rank-1 elements, i.e. for the memory demand either, from the exact to the low-rank approximation is larger for the heating systems example than for the HVAC system. The same effect can be observed for TT and HT formats, which is illustrated in Table 5.3. The system structure has a similar effect on the reduction factor of the TT and HT decomposition than for the CP.

Table 5.3: Comparison of the memory demand of the heating and HVAC system examples

	CP		**Tucker**		**TT**		**HT**	
	Heating	HVAC	Heating	HVAC	Heating	HVAC	Heating	HVAC
Exact	10 235	3 834	14 605	85 482	927 935	152 658	53 254 085	4 100 814
Minimal	2 581	3 124	14 605	85 482	17 411	9 120	12 499	7 177
Factor	3.97	1.23	1	1	53.3	16.74	4 260	571.4

But especially the heating systems example shows a problem that could occur when constructing these formats in the proposed way. Even though the parameter tensor is in decomposed structure their memory demand is very high for the exact representation as indicated in Table 5.1. In the considered example the exact representation is possible but if one thinks about systems of larger scale, i.e. with more states and inputs, the example shows that the direct translation from exact CP especially to HT could lead to problems to store the decomposition factors. The size of the direct translation to TT and HT depends significantly on the number of rank-1 terms of the CP representation as derived in (3.33) and (3.36). It was shown that low-rank approximations of the CP representation can be found. Thus, to overcome this storage problem the TT and HT representations can be constructed from the low-rank CP representation and not by using the exact one. This causes a loss in the accuracy, that can be controlled by the CP tensor rank and the accuracy of TT and HT truncation. But the big advantage is that the direct translation from CP to the other formats has a lower memory demand, which allows to apply these decomposed formats also for systems of larger scale.

The two application examples confirm the general conclusions drawn in the comparison in Section 3.2.5. The representation of the MTI system examples is possible with all four methods. The reduction of the memory demand by truncation techniques is a big advantage of the tensor format, which is important especially for large-scale systems. The methods can be applied to the CP, TT and HT decompositions with good reduction results by maintaining the dynamical behavior of the models. The examples showed the big potential of these decomposition methods for large-scale systems that are constructed by subsystems, that have a similar structure, as it is the case e.g. for large heating systems or smart grid applications. For the Tucker approach the truncation is not possible, which lowers its applicability for large-

scale systems. The results of the comparison considering the example systems is summarized in Table 5.4, that extends the Table 3.1 with general results.

Table 5.4: Result of the comparison of the decomposition application (Ranking starting with the best: $+ \circ -$; ✓: possible, ×: not possible)

	CP	Tucker	TT	HT
Representation of example systems	✓	✓	✓	✓
Memory demand of the direct translation	+	∘	−	−
Memory demand of the truncation	+	−	∘	∘
Truncation possible	+	−	+	+
Tensor error	+	+	+	+
Simulation error	+	+	+	+
Eigenvalues	+	+	+	+

The CP representation showed the best results for the memory demand. But also TT and HT had a good memory effort properties. Their additional advantage is the SVD based truncation that is numerically more stable and has a more comfortable application than the truncation algorithm for CP, such that CP, TT and HT are suited best for MTI system representation. In the followings application examples for controller design the CP representation is used, because of its low number of elements that have to be stored and its nice properties for algorithm development.

5.3 Controller design for heating systems

Several controller design techniques for MTI systems in decomposed tensor representation were described in Chapter 4. These methods are applied in this section to the heating system example, whose model was derived in Section 5.1.1. In Section 5.2.1 decomposed representations of the heating system model were found with a low memory demand. It turned out, that CP, TT and HT tensors are suited best for the model representation. Since the CP model showed the lowest storage effort, this decomposed format of the heating system model is used in the following. To apply the feedback linearization method of Section 4.2, a single heating circuit is used in Subsection 5.3.1, where the plant is linearized by controlling the pump of the circuit via nonlinear state feedback given in a tensor structure. In Subsection 5.3.2 a heating system of larger scale is taken with several boilers and consumers. To determine a controller structure with reduced communication effort and little loss in the control performance only, a decentralized feedback controller is designed for this plant as described in Section 4.3. This leads to the application of MPC controllers for MTI systems. At first the AMPC-SL method for MTI systems derived in Section 4.5 is applied to a single heating circuit and tested in closed loop simulations in Section 5.3.3. Afterwards Section 5.3.4 extends the application to a distributed predictive controller design for an overall heating system with a boiler and several consumers, where each component gets its own AMPC-SL controller node as described in the DMPC method in Section 4.6. Finally, the AMPC-SL

method is implemented on a real-time hardware device to operate one controller node at a heating circuit of a real world office building. The results are presented in Section 5.3.5.

5.3.1 Feedback linearization

The method of feedback linearization for decomposed MTI system introduced in Section 4.2 is applied in this section to an example from heating systems. In the literature the general feedback linearization approach for nonlinear systems was used for heating or HVAC systems in several applications, [40, 91, 98, 107]. Here one component, a consumer, is linearized by the tensor based approach for MTI systems. This subsystem can then be used e.g. in a larger system context as linear one. The consumer model derived in Section 5.1.1 is a MIMO model. The feedback linearization approach was developed for SISO systems. Therefore several assumptions have to be made to apply this control method. It is assumed, that the consumer is supplied with a constant supply temperature of $T_s = 80\,°\mathrm{C}$. The disturbances are also set to constants with an outside temperature of $T_o = 5\,°\mathrm{C}$ and no solar radiation by $\dot{Q}_{solar} = 0\,{}^{W}/_{\mathrm{m}^2}$. The thermal dynamics of the consumer are described by the differential equations (5.1) and (5.2). The flow $\dot{V}_{HC}$ through the consumer is generated by a controllable pump. The input signal $\varphi \in [0, 1]$ of the pump sets the flow from $0\,{}^{m^3}/_{\mathrm{s}}$ to the maximal flow $\dot{V}_{max}$. Since the pump does not react instantaneously on a change in the control signal φ, it is modeled as first order system here

$$\ddot{V}_{HC} = -\frac{1}{\tau_{\dot{V}}}\dot{V}_{HC} + \frac{\dot{V}_{max}}{\tau_{\dot{V}}}\varphi, \tag{5.31}$$

with time constant $\tau_{\dot{V}}$. Thus, the states of the consumer model are given by

$$\mathbf{x} = \begin{pmatrix} T_r & \dot{V}_{HC} & T_b \end{pmatrix}^T.$$

The input is the pump signal $u = \varphi$. The room temperature is the output $y = T_b$ of the model. With these state variables the state space model of the consumer follows from (5.1), (5.2) and (5.31)

$$\begin{aligned}
\dot{x}_1 &= \frac{T_s}{V_c}x_2 - \frac{1}{V_c}x_1x_2 - \frac{k_{rb}}{c\rho V_c}x_1 + \frac{k_{rb}}{c\rho V_c}x_3,\\
\dot{x}_2 &= -\frac{1}{\tau_{\dot{V}}}x_2 + \frac{\dot{V}_{max}}{\tau_{\dot{V}}}u,\\
\dot{x}_3 &= \frac{k_{rb}}{C_b}x_1 + \frac{-k_{rb} - k_{bo}}{C_b}x_3 + \frac{k_{bo}}{C_b}T_o,\\
y &= x_3,
\end{aligned}$$

where the supply and outside temperatures T_s and T_o are assumed as constants here. Inside the state equations the flow is multiplied by the return temperature, which results in a multiplication of two states $\dot{V} \cdot T_r = x_2 \cdot x_1$. Because of that, the model is multilinear. All other terms are linear. Since the system has one input, one output and no multiplications

between inputs occur, the model belongs to the class of affine MTI models, that has a tensor representation

$$\begin{aligned}\dot{\mathbf{x}} &= \left\langle \mathsf{A} \,\middle|\, \mathsf{M}\left(T_r, \dot{V}_{HC}, T_b\right) \right\rangle + \left\langle \mathsf{B} \,\middle|\, \mathsf{M}\left(T_r, \dot{V}_{HC}, T_b\right) \right\rangle u,\\ y &= \left\langle \mathsf{C} \,\middle|\, \mathsf{M}\left(T_r, \dot{V}_{HC}, T_b\right) \right\rangle,\end{aligned}$$

with parameter tensors $\mathsf{A} \in \mathbb{R}^{2\times2\times2\times3}$, $\mathsf{B} \in \mathbb{R}^{2\times2\times2\times3}$ and $\mathsf{C} \in \mathbb{R}^{2\times2\times2}$. For the output, i.e. the building temperature, a reference temperature $T_{b,ref}$ is given. A feedback linearizing controller is designed, such that the closed loop from $T_{b,ref}$ to T_b is linear. The closed loop is depicted in Figure 5.11.

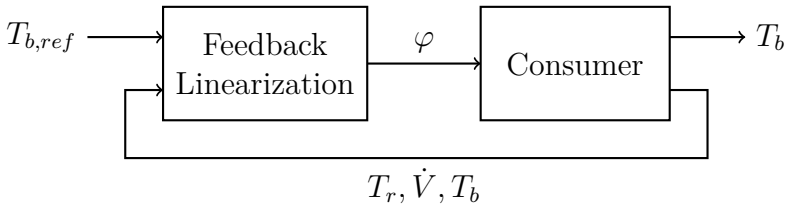


Figure 5.11: Closed loop of the consumer and feedback linearization controller

To design the controller at first the Lie derivatives along $\mathbf{a}(\mathbf{x})$ have to be determined with the tensor approach (4.23) by

$$L_{\mathbf{a}}^0 c(\mathbf{x}) = \left\langle \mathsf{C} \,\middle|\, \mathsf{M}\left(T_r, \dot{V}_{HC}, T_b\right) \right\rangle, \tag{5.32}$$

$$\begin{aligned}L_{\mathbf{a}}^1 c(\mathbf{x}) &= \left\langle \mathsf{A}_1 \circ (\mathsf{C} \times_3 \boldsymbol{\Theta}) + \mathsf{A}_2 \circ (\mathsf{C} \times_2 \boldsymbol{\Theta}) + \mathsf{A}_3 \circ (\mathsf{C} \times_1 \boldsymbol{\Theta}) \,\middle|\, \mathsf{M}_p^2\left(T_r, \dot{V}_{HC}, T_b\right) \right\rangle\\ &= \left\langle \mathsf{L}_{\mathsf{A},\mathsf{C},1} \,\middle|\, \mathsf{M}_p^2\left(T_r, \dot{V}_{HC}, T_b\right) \right\rangle,\end{aligned} \tag{5.33}$$

$$L_{\mathbf{a}}^2 c(\mathbf{x}) = \left\langle \mathsf{L}_{\mathsf{A},\mathsf{C},2} \,\middle|\, \mathsf{M}_p^3\left(T_r, \dot{V}_{HC}, T_b\right) \right\rangle, \tag{5.34}$$

$$L_{\mathbf{a}}^3 c(\mathbf{x}) = \left\langle \mathsf{L}_{\mathsf{A},\mathsf{C},3} \,\middle|\, \mathsf{M}_p^4\left(T_r, \dot{V}_{HC}, T_b\right) \right\rangle, \tag{5.35}$$

where the parameter tensors follow from (4.24). The detailed results for the Lie derivatives are given in the Appendix B.3. With these results, the Lie derivatives along $\mathbf{b}(\mathbf{x})$ are computed by operational tensors by (4.26) with parameter tensors (4.25) leading to

$$\begin{aligned}L_{\mathbf{b}} L_{\mathbf{a}}^0 c(\mathbf{x}) &= \left\langle \mathsf{L}_{\mathsf{B},\mathsf{A},\mathsf{C},0} \,\middle|\, \mathsf{M}_p^2\left(T_r, \dot{V}_{HC}, T_b\right) \right\rangle = 0,\\ L_{\mathbf{b}} L_{\mathbf{a}}^1 c(\mathbf{x}) &= \left\langle \mathsf{L}_{\mathsf{B},\mathsf{A},\mathsf{C},1} \,\middle|\, \mathsf{M}_p^3\left(T_r, \dot{V}_{HC}, T_b\right) \right\rangle = 0,\\ L_{\mathbf{b}} L_{\mathbf{a}}^2 c(\mathbf{x}) &= \left\langle \mathsf{L}_{\mathsf{B},\mathsf{A},\mathsf{C},2} \,\middle|\, \mathsf{M}_p^4\left(T_r, \dot{V}_{HC}, T_b\right) \right\rangle = \frac{\dot{V}_{max} k_{rb}}{C_b V_c \tau_{\dot{V}}} \left(T_s - x_1\right).\end{aligned} \tag{5.36}$$

As stated in (4.14) and (4.15) it is required for this controller approach, that $L_{\mathbf{b}} L_{\mathbf{a}}^2 c(\mathbf{x})$ is nonzero for this plant with three states. This is the case, if $T_s \neq x_1$, which means that the return temperature should not be equal to the supply temperature. That is fulfilled here, if the consumer is operated under nominal conditions, when water flows through the consumer

pipes and e.g. no bypasses occur. The consumer has a certain heat demand and thus takes heat from the supply flow, such that the return temperature is reduced compared to the supply temperature. When $L_{\mathbf{b}}L_{\mathbf{a}}^2 c(\mathbf{x}) \neq 0$ holds, the relative degree of the plant follows from (4.14) and (4.15) and is equal to $\rho_{sys} = 3$. The relative degree is equal to the number of states, such that no zero dynamics have to be checked. The desired linear closed loop behavior is given by

$$\dddot{T}_b + \mu_2 \ddot{T}_b + \mu_1 \dot{T}_b + \mu_0 T_b = \mu_0 T_{b,ref}. \tag{5.37}$$

To determine the pole locations of the linear transfer function following from (5.37), the MTI consumer model is linearized around an operating point by (3.46) and (3.47). In Section 4.1.1 it is described, how pole locations for pole placement are determined by a LQR design for linear systems, [23]. The same is done here for the linear approximation of the plant model. The pole locations determined by the LQR design are used then to compute the coefficients of the desired linear behavior (5.37).

With the linear behavior (5.37) and the parameter tensors of the Lie derivatives (5.32) to (5.36) all parameters of the feedback linearization controller are defined and can be expressed with respect to the monomial tensor $\mathsf{M}_p^3\left(T_r, \dot{V}, T_b\right)$ leading to the controller law

$$u = \varphi = \frac{-\left\langle \mu_0 \mathsf{C} + \mu_1 \mathsf{L}_{\mathsf{A},\mathsf{C},1} + \mu_2 \mathsf{L}_{\mathsf{A},\mathsf{C},2} + \mathsf{L}_{\mathsf{A},\mathsf{C},3} \,\middle|\, \mathsf{M}_p^3\left(T_r, \dot{V}, T_b\right) \right\rangle + \mu_0\, T_{b,ref}}{\left\langle \mathsf{L}_{\mathsf{B},\mathsf{A},\mathsf{C},2} \,\middle|\, \mathsf{M}_p^4\left(T_r, \dot{V}, T_b\right) \right\rangle} \tag{5.38}$$

The closed loop system is simulated for a step change in the room temperature reference. The simulation result is shown in Figure 5.12 with the input signal of the pump computed by the controller and the resulting room temperature.

The simulation shows, that the controller works as desired, such that the room temperature follows its reference with the desired closed loop dynamics as given by (5.37). This shows that the tensor based feedback linearization method derived in Section 4.2 is applicable for plants in the field of heating systems. The approach works fine in the simulation scenario here, where no disturbances occur and some assumptions, e.g. regarding the inputs, were made. A model mismatch or disturbances as they exist for real buildings or consumers would lead to inaccuracies in the controller result. Furthermore the tensor based design method has to be extended to MIMO systems. Because of these reasons this controller approach will not be used for implementation at a real building here. Other methods fit better to this application since they allow to consider MIMO systems, disturbances and plant constraints directly in the controller design like in MPC, focused on in Sections 5.3.3 to 5.3.5. But the derivation of the feedback linearization method for MTI systems based on decomposed tensors showed that it is possible to adapt general nonlinear control design methods to the special structure of MTI systems resulting in a fixed controller structure. The decomposed structure allows to apply the methods also to large-scale systems. The application to a heating system example showed the validity of the approach also for real world systems in simulation.

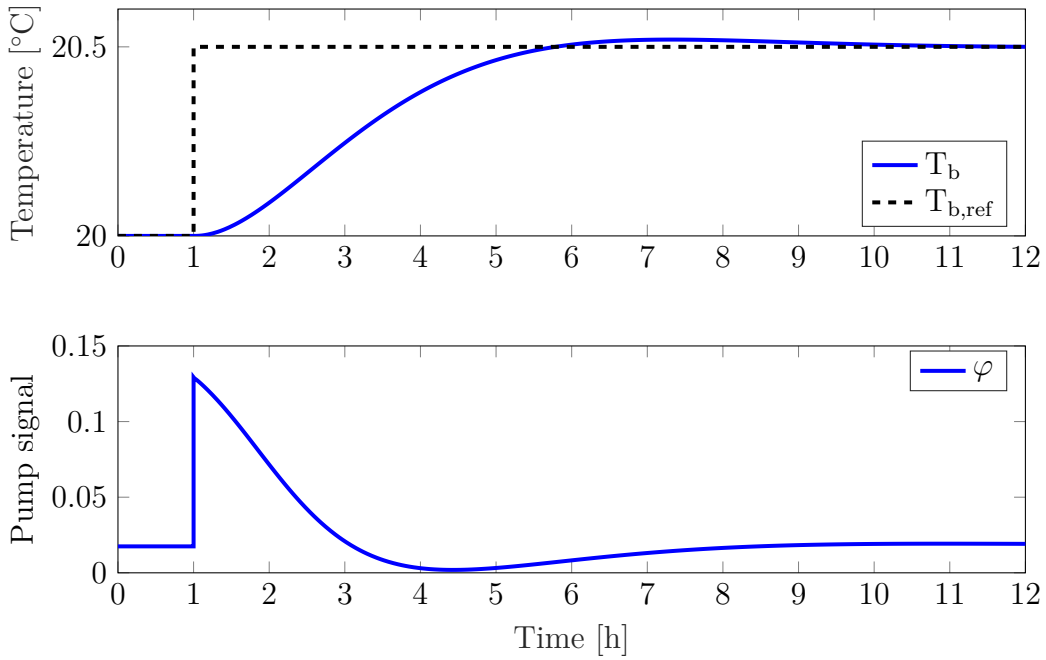


Figure 5.12: Closed loop simulation result of the consumer and feedback linerization controller

5.3.2 Decentralized feedback design

The LQR design was used for heating systems e.g. in [72]. For large-scale heating systems as introduced in Section 5.1.1 a central state feedback design, as it is the case in LQR controller synthesis, leads to a large communication effort since each state is necessary for the computation of each control signal signal of the plant. The communication gets even more complex as the consumers and the boilers are located at different positions, such that some distance has to be passed. This motivates the use of decentralized LQR design. Since the model of the plant is multilinear, as shown in Section 5.2.1, the decentralized feedback design for MTI systems should be applied here, see [48]. The considered plant is a heating system with $N_B = 2$ boilers and $N_{HC} = 7$ consumers, which results in a MTI state space model as derived in the Appendix B.1, that is represented as CP decomposed MTI system

$$\dot{\mathbf{x}} = \langle\, \mathsf{F} \mid \mathsf{M}\,(\mathbf{x}, \mathbf{u}, d)\, \rangle\,,$$

with state, (5.25) and input (5.26) vectors

$$\mathbf{x} = \begin{pmatrix} T_{s,1} & T_{s,2} & T_{r,1} & \cdots & T_{r,7} & T_{b,1} & \cdots & T_{b,7} \end{pmatrix}^T \in \mathbb{R}^{16},$$
$$\mathbf{u} = \begin{pmatrix} \alpha_1 & \alpha_2 & \zeta_1 & \cdots & \zeta_7 & \varphi_1 & \cdots & \varphi_7 \end{pmatrix}^T \in \mathbb{R}^{16}.$$

The disturbance of the plant is the outside temperature $d = T_o$. The solar radiation is not considered here and set to zero $\dot{Q}_{solar} = 0\,{}^{W}/_{\mathrm{m}^2}$.

The aim of the controller is that the supply temperatures of the boilers should track a reference temperature. In standard applications this reference temperature depends linearly

on the outside temperature by a heating curve

$$T_{s,i,ref} = \eta_{1,i} T_o + \eta_{0,i},\ i = 1, 2, \tag{5.39}$$

such that the supply temperature is high for cold outside temperatures and low for high outside temperatures as depicted in Figure 5.13.

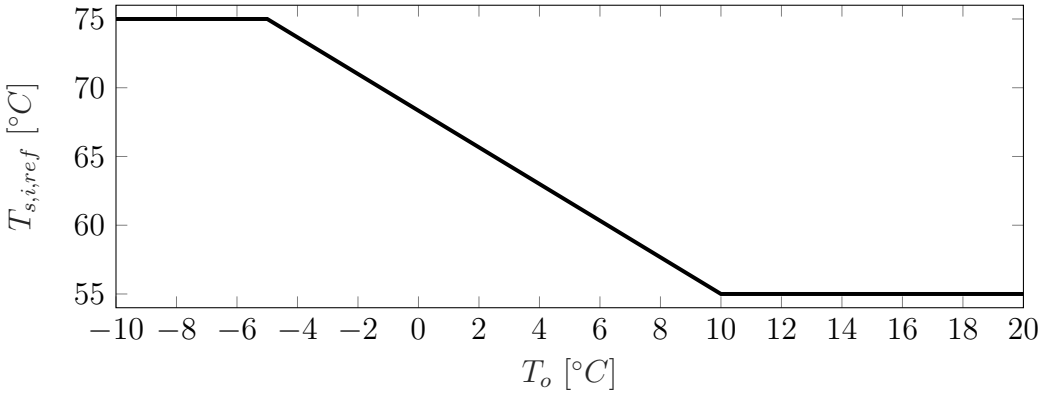


Figure 5.13: Heating curve

The valves and pumps are controlled to adapt the supply temperature to each heating circuit, such that the room temperatures are at a reference value $T_{b,i,ref}$. Thus, the reference vector reads

$$\mathbf{r} = \begin{pmatrix} T_{s,1,ref} & T_{s,2,ref} & T_{b,1,ref} & \cdots & T_{b,7,ref} \end{pmatrix}^T.$$

Here the reference is chosen for all rooms constant at $T_{b,i,ref} = 21\,^\circ\mathrm{C}$. The heating curves (5.39) are the same for both boilers. The inputs of the plant, i.e. the modulation signals on the generator side

$$\alpha_i \in [0, 1],\ i = 1, 2,$$

and the control signals of the pumps and the valves on the consumer side

$$\varphi_i \in [0, 1],\ \zeta_i \in [0, 1],\ i = 1, \ldots, 7,$$

are constrained. As first step in controller synthesis the structure of the feedback gain and thus the necessary communication structure has to be investigated. Therefore, the sparsity promoting design problem (4.8) is solved for different operating points in the preprocessing step, see Figure 4.10. The operating points considered here depend on the outside temperature only. For a fixed outside temperature the supply temperature reference is determined by (5.39). The outside temperature is the disturbance. The references for the building temperatures are fixed. All other states and inputs at the operating point are computed by (4.31). Thus, for computing the possible operating points the outside temperature is iterated over an interval $T_o \in [-10\,^\circ\mathrm{C}, 10\,^\circ\mathrm{C}]$. The decentralized controller is computed by (4.8) for linearizations (4.35) and (4.36) at different operating points and values of γ. The performance degradation ΔJ for different values of γ and T_o is shown in Figure 5.14.

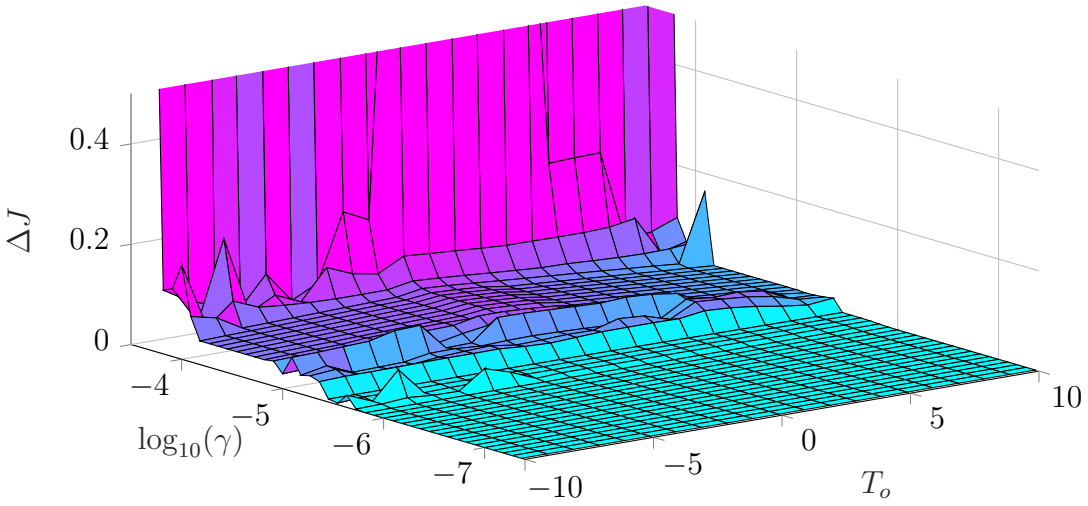


Figure 5.14: Performance loss ΔJ of the decentralized controller

The figure shows that the performance degradation increases for decreasing γ, because the focus of controller design is put on sparsity more and more. Up to a value for γ of $1 \cdot 10^{-5}$ the performance degradation is low and is rapidly increasing for larger γ, such that this value is chosen here. It can be seen that the performance does not depend significantly on the operating point, such that the same γ can be used for every operating point. The next question is, which structure results from the chosen γ. In this example the same sparsity structure is determined in every investigated operating point, which is depicted in Figure 5.15 and used for the controller operation.

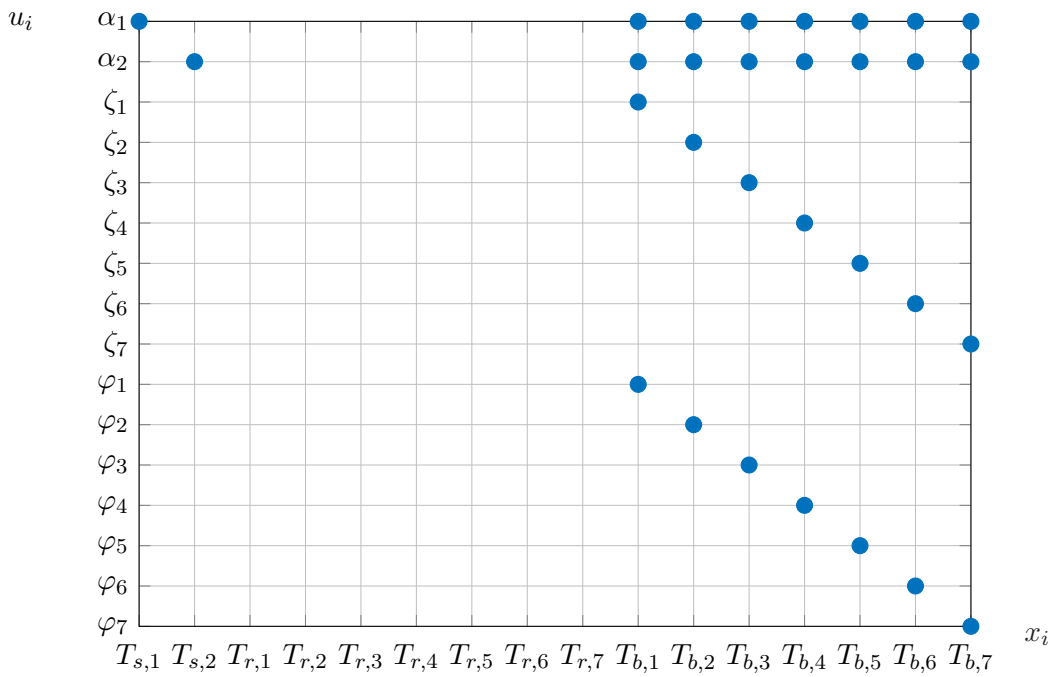


Figure 5.15: Sparsity structure of $\mathbf{K}$

In contrast to the central design with $16 \cdot 16 = 256$, only 30 non-zeros entries are necessary in the gain matrix to get a controller with small loss in control performance compared to the central case. This is an enormous reduction of communication effort. The structure, determined automatically, shows that only the controllers of the boilers need information of the overall system, i.e. the supply temperatures and the building temperatures $T_{b,i}$ of the consumers. The controllers of the pumps and the three way valves only need local information of the particular consumer, i.e. their building temperature $T_{b,i}$. The local controllers for the consumers do not need the information of the whole system, e.g. the building and return temperatures of the other consumers, which results in a much simpler communication infrastructure compared to the central case. In case of the central design the controller would have to access all measurement and control signals. This is not necessary in the proposed decentralized design, as the resulting structure with the distribution of the control task to several controller nodes shows, which is depicted in Figure 5.16.

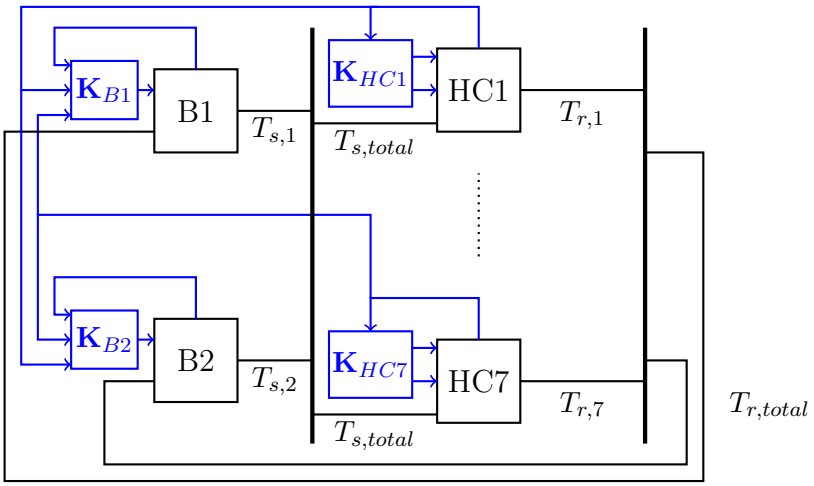


Figure 5.16: Decentralized controller structure for the heating system

Simulating the closed loop system depicted in Figure 4.12 for one day results in the supply temperature shown in Figure 5.17.

Even though many entries in the feedback gain matrix are set to zero, the simulation shows, that the system temperatures follow the references as shown for the supply temperatures of the first boiler in Figure 5.17. The temperature of the other boiler is controlled in the same way and shows the desired behavior as well. The room temperatures $T_{b,i}$ of the heating circuits reach the reference value with a small maximum deviation of $\pm 0.2\,\mathrm{K}$, which is much better than the acceptable limits. Thus, the control performance is good. Every change in the references and the disturbances of more that $0.5\,\mathrm{K}$ leads to an adaption of the controller gain such that the controller shows good results even though the plant is not linear but multilinear.

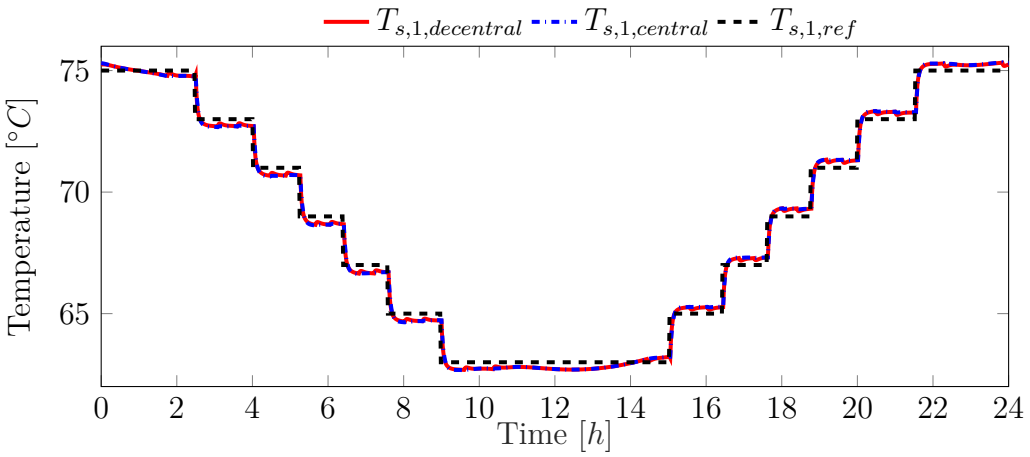


Figure 5.17: Comparison of the closed loop simulation results for central and decentral design

5.3.3 Adaptive model predictive control with successive linearization

The adaptive MPC approach with successive linearization introduced in Section 4.5 is applied here to a real heating system. The controller should work with one heating circuit of a real world office building. The setup of the plant with consumer and mixing circuit is described in Section 5.1.1 and shown in Figure 5.3. All heating circuits inside this building are supplied with a total supply temperature $T_{s,total}$. To adapt the overall supply temperature to the needs of the specific consumer, a mixing circuit is installed, that allows to lower the supply temperature $T_{s,HC}$ reaching the consumer compared to the total one by return flow addition. A pump is installed in each heating circuit to generate a flow through the consumer. In the building focused on here the pump is controlled manually only, such that the controller has no access to its control signal. The only input of the heating circuit plant is the three way valve. In the former used conventional control strategy the reference for the supply temperature of a heating circuit is determined by a heating curve. A heating curve gives the supply temperature reference by a saturated linear dependence on the outside temperature, such that the supply temperature is high for low outside temperatures and low for high outside temperatures. To realize a night reduction of the temperature in this plant the heating curve is reduced by 10 K for night operation. The heating curve for the considered heating circuit is shown later in this section in Figure 5.28.

In a conventional design the three way valve is controlled by a PI controller. The PI controller gets a reference for the supply temperature from the heating curve and sets the control signal of the valve, such that the reference temperature for the supply of the consumer is reached. This conventional closed loop setup is depicted in Figure 5.18.

In most cases the heating curve is designed conservatively, such that the consumer is supplied with enough heat in any case. Due to this fact situations occur, where the heating circuit is operated with supply temperatures that are too high from an energy efficiency perspective. Especially the night operation with a reduction of the temperatures offers a high energy saving potential. The idea here is to replace the heating curve by a AMPC-SL controller to

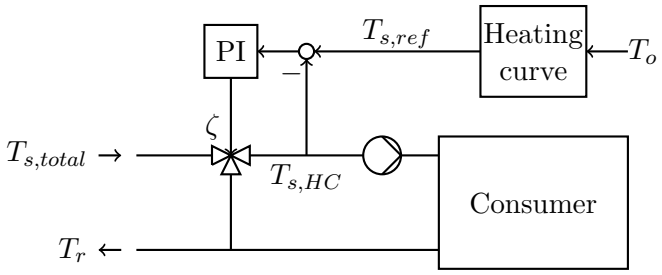


Figure 5.18: Setup of the consumer circuit with conventional controller design

compute a supply temperature reference, that is adapted better to the real ambient conditions. This should prevent the operation with too high supply temperatures. The controller computes a supply temperature, such that the room temperature follows a reference leading to a building operation in the comfort range. Therefore, the controller needs measurement data from the plant of the current room, return and outside temperatures as well as the volume flow. To act predictively the weather forecast on future outside temperatures and solar radiation is included. Using a model of the plant, the reference for the supply temperature of the consumer is determined, that is set by an underlying PI control loop afterwards. Figure 5.19 shows the controller setup with the plant and the predictive controller.

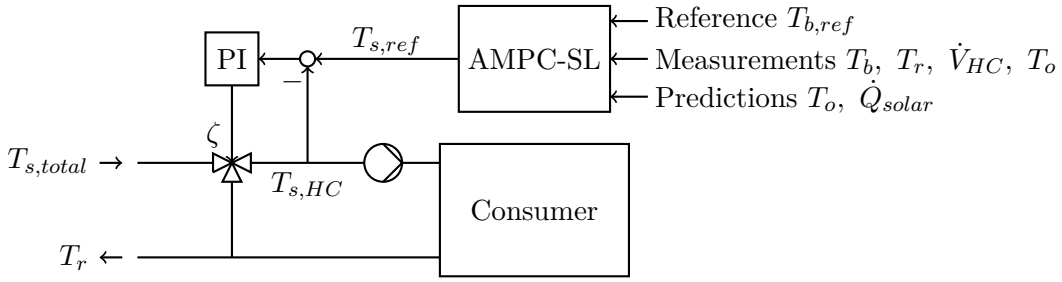


Figure 5.19: Setup of the consumer circuit with the AMPC-SL controller design

To determine the optimal supply temperatures, a model of the consumer is necessary to predict, which building temperatures would result in the future from particular supply temperatures, flows, outside temperatures and solar radiations. The model of the consumer was introduced in Section 5.1.1. Its thermal dynamics are described by heat balances and are given by (5.1) and (5.2). The consumer model with its inputs and outputs are shown in Figure 5.20.

The states

$$\mathbf{x} = \begin{pmatrix} T_r & T_b \end{pmatrix}^T,$$

of the model used inside the predictive controller are the return and the room temperatures. As input of the model it is assumed, that the consumer is supplied with a certain supply temperature, which is the control signal here. The disturbances of the consumer are on the one hand the weather conditions with the outside temperature and the solar radiation. One

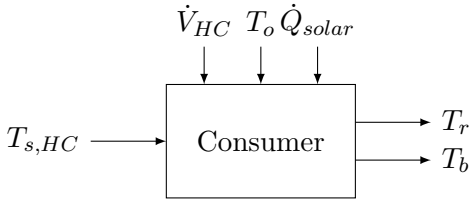


Figure 5.20: Inputs and outputs of the consumer model for the AMPC-SL

the other hand since the pump of the consumer circuit is not controllable, the volume flow is treated as additional disturbance here leading to the control and disturbance input vectors

$$\begin{aligned} u &= T_{s,HC}, \\ \mathbf{d} &= \begin{pmatrix} \dot{V}_{HC} & T_o & \dot{Q}_{solar} \end{pmatrix}^T. \end{aligned}$$

The outputs $\mathbf{y} = \mathbf{x}$ of the plant are equal to the states. Because of the multiplication of the disturbance variable $\dot{V}_{HC}$ with the control variable $T_{s,HC}$ as well as the state variable T_r in (5.2) the model belongs to the class of MTI systems, that can be written in tensor format by

$$\begin{aligned} \dot{\mathbf{x}} &= \langle\, \mathsf{F} \mid \mathsf{M}(\mathbf{x}, u, \mathbf{d}) \,\rangle\,, \\ \mathbf{y} &= \langle\, \mathsf{G} \mid \mathsf{M}(\mathbf{x}, u, \mathbf{d}) \,\rangle\,, \end{aligned}$$

with parameter tensors $\mathsf{F} \in \mathbb{R}^{\times(6)2\times 2}$ and $\mathsf{G} \in \mathbb{R}^{\times(6)2\times 2}$, that are assumed to be given in CP structure. In Section 4.5 it was proposed, that using a scaled version of the model avoids some numerical issues, when signals are of different magnitude, which is the case here with e.g. temperatures and volume flows. For scaling, operating ranges for all model signals

$$\begin{aligned} &15\,^\circ\mathrm{C} \le T_r \le 80\,^\circ\mathrm{C}, && 0\,\tfrac{\mathrm{m}^3}{\mathrm{s}} \le \dot{V}_{HC} \le 1.4 \cdot 10^{-3}\,\tfrac{\mathrm{m}^3}{\mathrm{s}}, \\ &18\,^\circ\mathrm{C} \le T_b \le 23.5\,^\circ\mathrm{C}, && -5\,^\circ\mathrm{C} \le T_o \le 10\,^\circ\mathrm{C}, \\ &30\,^\circ\mathrm{C} \le T_{s,HC} \le 80\,^\circ\mathrm{C}, && 0\,\tfrac{\mathrm{W}}{\mathrm{m}^2} \le \dot{V}_{HC} \le 500\,\tfrac{\mathrm{W}}{\mathrm{m}^2}, \end{aligned} \tag{5.40}$$

are set according to the typical operation of the plant, that were determined from measurement data. The decomposed MTI model is scaled by the approach introduced in Section 3.5, such that all signals are in the interval $[0, 1]$. The scaled model

$$\begin{aligned} \dot{\tilde{\mathbf{x}}} &= \left\langle\, \tilde{\mathsf{F}} \;\middle|\; \mathsf{M}(\tilde{\mathbf{x}}, \tilde{u}, \tilde{\mathbf{d}}) \,\right\rangle, \\ \tilde{\mathbf{y}} &= \left\langle\, \tilde{\mathsf{G}} \;\middle|\; \mathsf{M}(\tilde{\mathbf{x}}, \tilde{u}, \tilde{\mathbf{d}}) \,\right\rangle, \end{aligned}$$

is used inside the controller and is linearized in each sampling time step around an actual operating point by the linearization approach for MTI system from Section 3.3. Before operation of the controller, a field Θ_{OP} of operating points is computed for the scaled model. For a memory efficient representation of the field of operating points the TT decomposition is used with a high accuracy to avoid problems with a possible approximation error resulting from the decomposition.

Inside the MPC optimization problem the linear approximation of the scaled model is used to predict the future plant behavior. The cost function is given by

$$J = \sum_{i=1}^{H_p} \left\| \hat{\tilde{\mathbf{x}}}(k+i) - \tilde{\mathbf{x}}_{ref}(k+i) \right\|_{\mathbf{Q}}^2 + \sum_{i=0}^{H_u-1} \|\tilde{u}(k+i) - \tilde{u}(k+i-1)\|_{\mathbf{R}}^2 + \sum_{i=0}^{H_u-1} \|\tilde{u}(k+i) - \tilde{u}_{target}(k+i)\|_{\mathbf{S}}^2 + \rho_\epsilon \epsilon_k,$$

where $\tilde{u}_{target}(k+i)$ is a target value for the control input signal. By choosing this target value as minimal value $\tilde{T}_{s,min}$ of the supply temperature reference $T_{s,ref}$, this term supports the usage of low supply temperatures. The variable ϵ_k is a slack variable to soften the hard constraints (5.40) of the signals with weighting factor ρ_ϵ, which supports the feasibility of the optimization problem, [12]. In the given application the controller should realize a reference tracking of the room temperature, i.e. the second state. The first state, the return temperature, is not considered for reference tracking. Thus, the weighting matrix of the reference tracking has the following structure

$$\mathbf{Q} = \begin{pmatrix} 0 & 0 \\ 0 & q \end{pmatrix},$$

such that a weight is put on the tracking of the second state only. The weighting factor of the first state is set to zero. With that the cost function is given here by

$$J = \sum_{j=1}^{H_p} q \left(\tilde{T}_b(k+j) - \tilde{T}_{b,ref}(k+j) \right)^2 + \sum_{j=0}^{H_u-1} r \left(\tilde{T}_{s,ref}(k+j) - \tilde{T}_{s,ref}(k+j-1) \right)^2 + \sum_{j=0}^{H_u-1} s \left(\tilde{T}_{s,ref}(k+j) - \tilde{T}_{s,min}(k+j) \right)^2 + \rho_\epsilon \epsilon_k. \tag{5.41}$$

The first term evaluates the reference tracking of the room temperature with weighting factor q. The second term penalizes the rate of change of the supply temperature reference with factor r. This reduces the dynamics of the control signal and avoids fast changes in the supply temperature reference signal. The third terms puts costs on the difference between the supply temperature reference and its minimal value, such that low supply temperatures are used. For the focused office building an occupation time from 6:00 to 20:00 o'clock could be identified. To fulfill the comfort requirements of the users, the building temperature should be next to the reference during the day. The minimal and maximal constraints of the room temperature are close to the reference value at occupation time. During the night the room temperature can be lowered. Also the constraints on the room temperature are wider. The controller has to take care, that the building does not cool down too much at night, i.e. the room temperature should not fall below a minimal value. Furthermore the controller has to determine the time for heating up the building in the morning, such that the comfort range of the room temperature is reached at the beginning of the occupation time at 6:00 o'clock. As minimal value for the building temperature 18 °C are reasonable. During occupation time the users request a building temperature of 23 °C to fulfill their comfort demands. The simplest reference trajectory would be a switch between these two values resulting in a rectangular signal as shown in Figure 5.21.

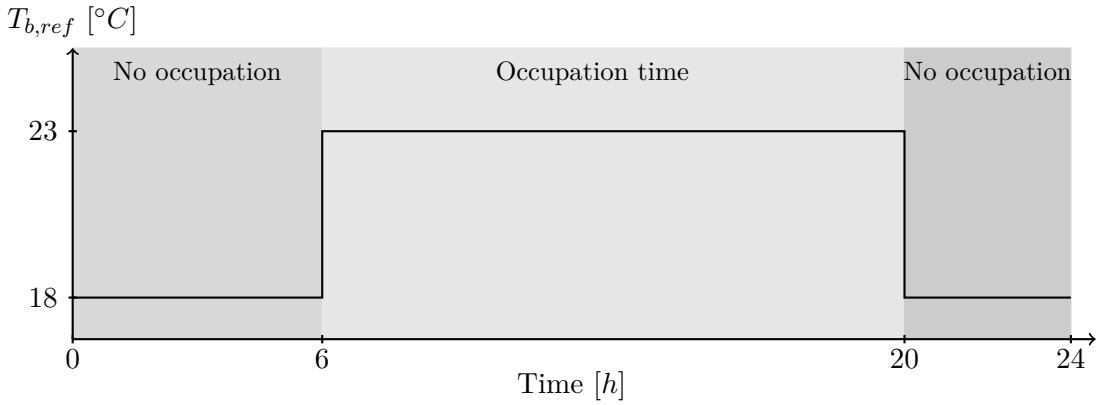


Figure 5.21: Rectangular reference signal for the room temperature

But this choice of the reference leads to some application specific problems, that will be discussed in the following. The first issue is the start of the occupation time with the simple reference in Figure 5.21: At 6:00 o'clock there is a step change in room temperature from 18 °C to 23 °C. Because of the slow building dynamics it is not possible with acceptable control effort, that the room temperature follows this fast increase in temperature. Since the reference is already known in advance for the whole prediction horizon, an optimal behavior in the sense of a quadratic cost function (5.41) would lead to a behavior that was illustrated in Figure 4.6b. The room temperature is already increased before the jump in the reference temperature as it is desired here since the controller should set the starting time for heat up. But it is also obvious that the reference value is just reached after the jump in the reference. This is the optimal behavior with respect to the quadratic cost function. But this is not the desired behavior from an application point of view. Since the occupation starts at 6:00 o'clock the room temperature should already reach its reference value at this time and not later. This is a requirement from practice. Thus, a rectangular reference trajectory does not make sense for this application with quadratic cost function. Another form of the reference has to be chosen that fits to the dynamics of the plant. To keep the reference signal simple a ramp is used, that has a slope that fits to the temperature dynamics of the building, such that it is possible, that the room temperature follows the reference signal.

Example 5.1 *Returning to the first order system of Example 4.1 here now a ramp is used as reference when the plant is controlled by a MPC controller. The ramp slope is chosen, such that it fits to the plant dynamics, i.e. the plant output can follow the reference. The closed loop output with a MPC controller and a ramp reference is depicted in Figure 5.22. When comparing Figure 5.22 to Figure 4.6b this exemplary shows that the control error is reduced enormously, when a ramp with a reasonable slope is used as reference instead of a step.*

For the heating system the ramp should reach the desired reference temperature of 23 °C at the beginning of the occupation time. Since the reference is adapted to the room dynamics, this allows that the room temperature is at the desired value at the beginning of the occupation. But it is only possible that the room temperature follows the reference, if the slope is chosen in the right way. The controller is designed for a floor of an office building, where

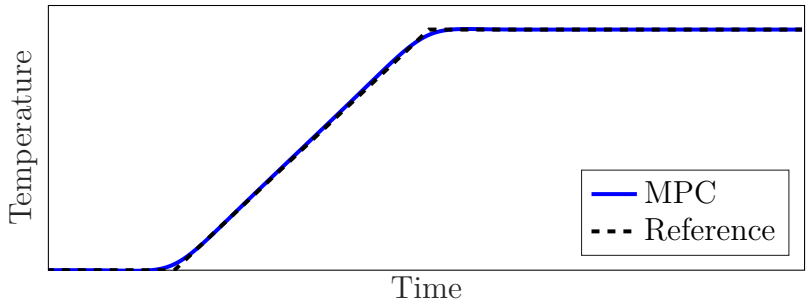


Figure 5.22: Exemplary closed loop simulation of a first order system with a MPC and ramp reference

measurement data is available. The building temperature is determined by a room temperature sensor in a reference room. Therefore different heat up processes are investigated to determine the temperature dynamics, that are illustrated in Figure 5.23.

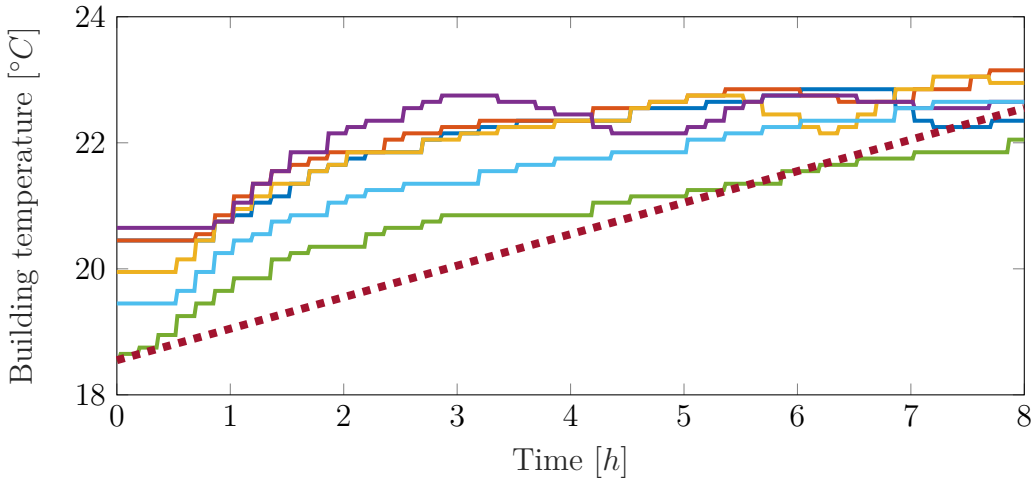


Figure 5.23: Estimation of the slope of the reference ramp (Solid: measurements, Dashed: Straight line with estimated slope)

The figure shows the different heat up processes of the room temperature as solid lines. The slope of the reference ramp is illustrated by a dashed straight line, that is chosen in a way, that the room temperatures are not slower than the chosen slope. The measurements show that the building dynamics are very slow, because the heat up process takes several hours. This can be caused by large masses or high losses due to large window surface areas of the building. This investigation results in a slope of 0.5 $^{K}/_{\mathrm{h}}$. The difference between the minimal value of 18 °C and the maximal value of 23 °C of the building temperature reference are 5 K. This means that the room needs 10 h to overcome this temperature difference. Because of that, the reference ramp starts at 20:00 o'clock at the day before to reach the 23 °C at 6:00 o'clock in the morning. During the weekend no occupation is assumed such that the reference temperature is set to its minimal value then. The constraints of the room temperature are determined according to the reference trajectory. The lower constraint is set parallel with a shift of 0.5 K below the reference. The upper constraint is held constant for the whole

time 0.5 K above the maximal value of the room temperature reference at occupation time. The reference and the constraints for the room temperature is depicted in Figure 5.24 for a whole week.

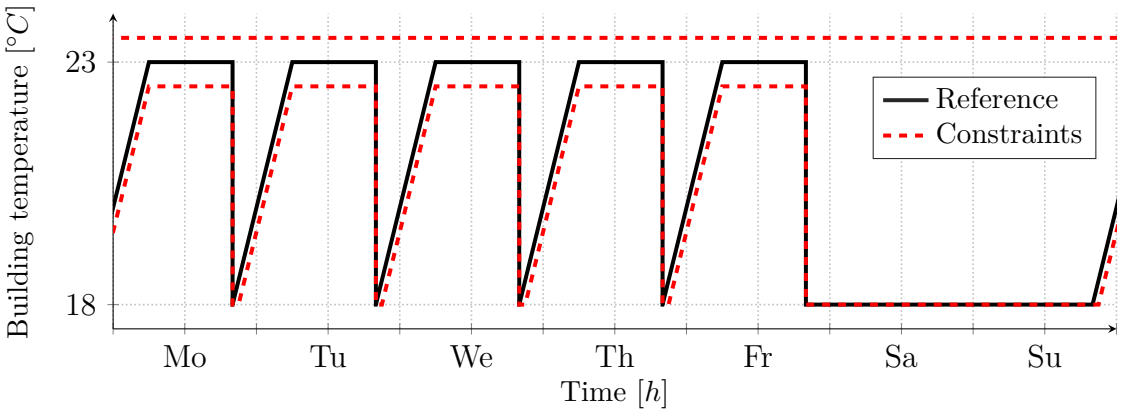


Figure 5.24: Reference trajectory and constraints of the room temperature for one week

The problem at the beginning of the occupation time is solved by the ramp reference. After the end of occupation time the focus is on a low supply temperature to save energy. The room temperature can be reduced. Because of that, the reference is set to its minimal value. But the same behavior as depicted in Figure 4.6b can be investigated also at the end of the occupation time in standard MPC configuration. The room temperature would be reduced already before the step change in the reference to minimize the cost function (5.41) since the controller knows the future reference values for the prediction horizon. This behavior can be explained by the MPC algorithm but is not desired here from an application point of view. In practice the room temperature should stay in the comfort range for the whole occupation time, i.e. until 20:00 o'clock. Because of that the prediction of the reference is changed and held constant during occupation time to a value of 23 °C. Figure 5.25 illustrates the prediction of the references for two different time instances on the one hand during the night and on the other hand at the end of occupation time, to illustrate the previously described procedure.

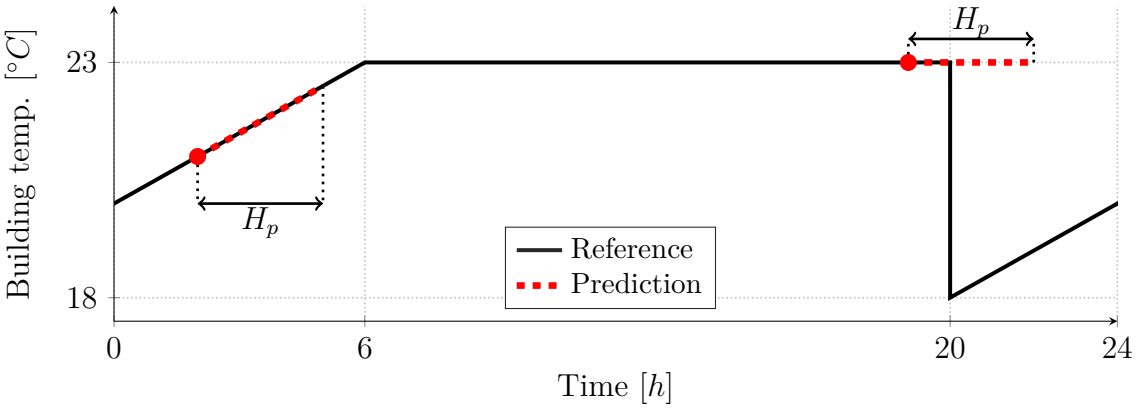


Figure 5.25: Reference of the room temperature and prediction for two time instants

Since the controller gets the information that the room temperature should be held at 23 °C the temperature is not reduced before 20:00 o'clock. After the end of occupation time the correct reference prediction according to the reference ramp is used inside the controller. Due to the system dynamics, the prediction and control horizon are set to 3 and 2 hours, respectively. The sampling time is one minute. The parameters q, r and s of the cost function are tuned by closed loop simulation with the MTI model of the plant. The simulation for one day of the tuned AMPC-SL controller for the heating circuit is depicted in Figure 5.26.

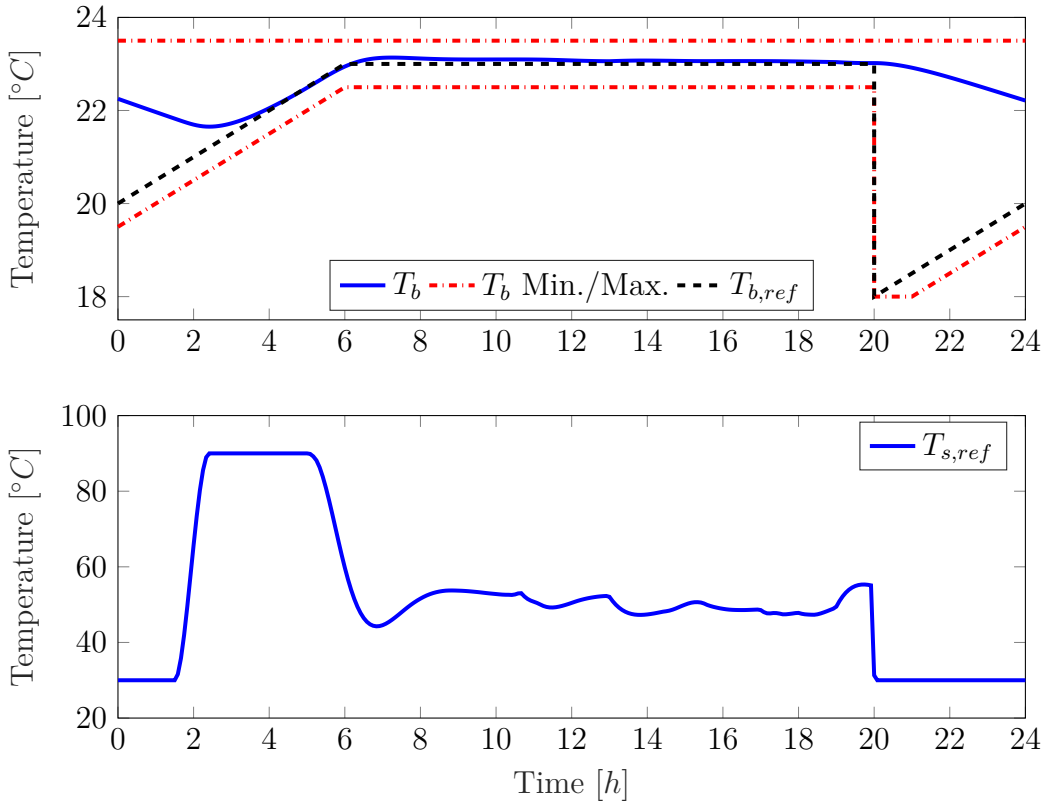


Figure 5.26: Closed loop simulation result with a consumer and AMPC-SL controller for one day

The simulation results show, how the controller reduces the supply temperature during the night. It is set to its minimal value. This shows one advantage of the controller. Due to that, the room cools down and is heated up again at the beginning of the day, which is done by an increase in the supply temperature. During the occupation time the room temperature is held constant at the comfort value. In the conventional control a fixed point in time is given, where the heat up should start. This time instant is chosen conservatively in many cases, such that the building is in the comfort zone even though it is very cold outside. If it is e.g. warmer outside the point in time is not adapted, i.e. it is not set later. Therefore the comfort range is reached earlier even before occupation time. This leads to a loss of energy since it is not necessary that the building is already in the comfort range when nobody is present. Here the second advantage of the controller comes into play. With the AMPC-SL the time instant of heating up is not fixed any longer. The necessary supply temperature

is computed with the help of the model in dependence of the current and future ambient conditions like outside temperature or solar radiation. With that the time where the heating starts is flexible and it is avoided that the building is heated up too early. This results in a more flexible operation, which allows also a reasonable cooling of the room at the weekend. The behavior for weekends is shown in Figure 5.27.

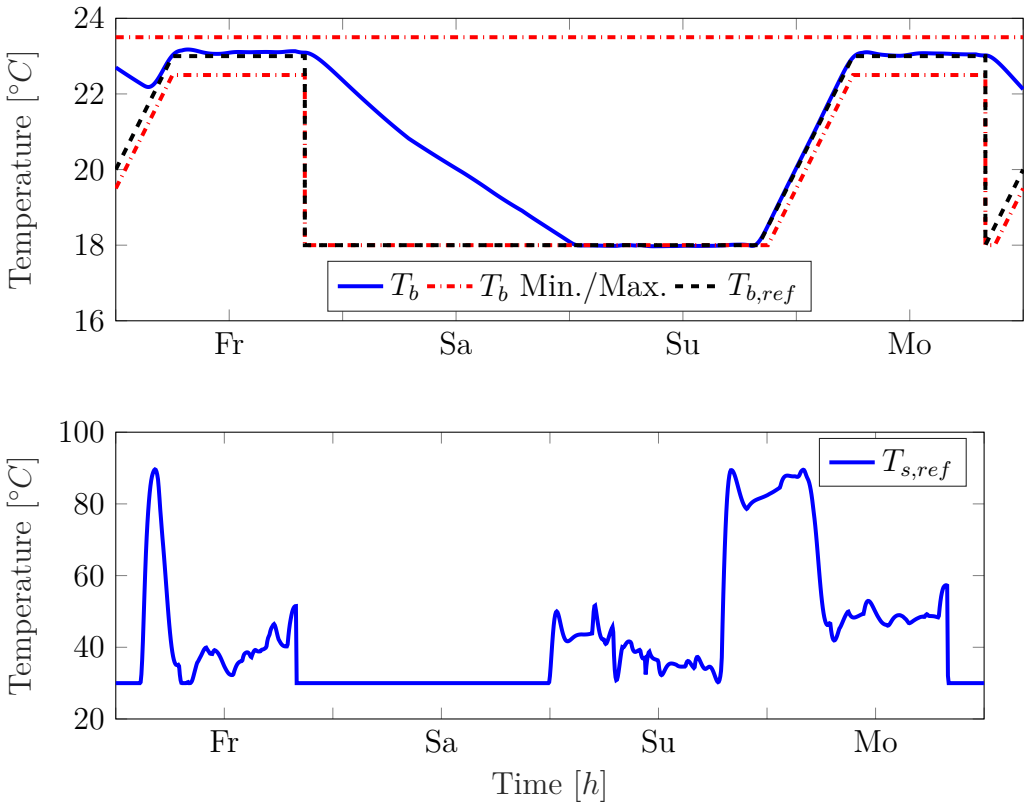


Figure 5.27: Closed loop simulation result with a consumer and AMPC-SL controller for a weekend and adjacent days

During the weekend the heating circuit is operated with minimum supply temperature, which saves energy. Because of that, the room cools down, but the room temperature does not fall below the minimal value of 18 °C. The controller starts heating with low supply temperatures to hold this minimal room temperature. With the model the controller determines the right time instant in the night between Sunday and Monday to increase the supply temperature again, such that the rooms are in the comfort range when the occupation time starts.

With the conventional control structure the supply temperature reference is determined by a heating curve. The heating curve used in this floor of the office building is estimated by plotting measurement data of the supply temperature over the outside temperature for all four seasons as depicted in Figure 5.28. The estimated heating curve can be fitted to the data by a saturated straight line. If the building is not occupied the heating curve is reduced by 10 K.

With the heating curve and the measurement data of the outside temperature used for the

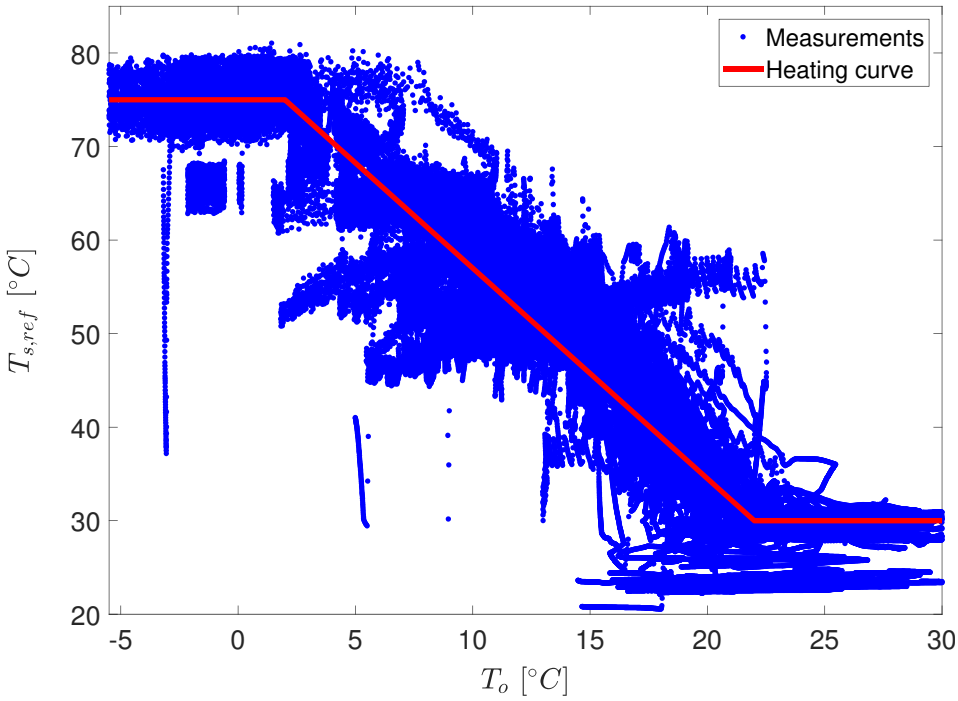


Figure 5.28: Estimation of the heating curve of the standard controller

closed loop simulation depicted in Figure 5.27 the supply temperature reference that would have been used with the conventional heating curve based control can be computed. When comparing this supply temperature reference with the one of the AMPC-SL approach, it is easy to see that the supply temperature during the night is much lower with the AMPC-SL, which is shown in Figure 5.29.

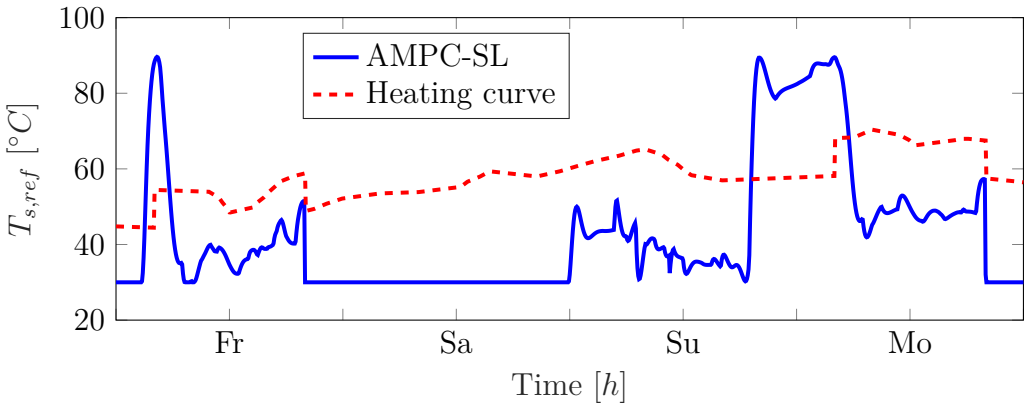


Figure 5.29: Comparison of supply temperature references of the heating curve and the AMPC-SL controller

Also during daytime lower supply temperatures are used to keep the heating circuit in the comfort range. Thus it is sufficient, if the boiler would provide lower supply temperatures

than the heating curve proposes, which shows the energy saving potential here. Only in the heat up interval before occupation time higher supply temperatures are necessary for the AMPC-SL, because the room temperature is decreased more during the night. The proposed AMPC-SL controller structure leads to a more flexible operation regarding the choice of lower supply temperatures and a flexible beginning of the heat up before occupation time, that still fulfills the comfort demands of the users.

5.3.4 Distributed model predictive control

The increasing complexity of systems can be also investigated in the field heating systems, e.g. by large-scale buildings. Since MPC has proven to be a control strategy that leads to good results in the temperature regulation of buildings like in [35] or in the previous Section 5.3.3 one has to investigate, how to apply such control concepts to large-scale buildings. To solve this problem the distribution of the predictive control task to several controller nodes to reduce the computational complexity was investigated for different building applications [54, 71, 88, 112]. In contrast to the approaches in the literature the focus here is on buildings, where subsystems are modeled by decomposed MTI systems and thus the AMPC-SL method introduced in Section 4.5 is applied to these subsystems.

Here the heating system introduced in Section 5.1.1 with one boiler and N_{HC} consumers is used as an application example. The decentralized feedback design in Section 5.3.2 for a similar plant already showed that the different subsystems, i.e. the boiler and the consumer, do not need the full system information. The subsystems can have its own local controllers with limited system information. The particular consumers do not need knowledge of the other consumers current situation. Only some information exchange has to take place between boiler and consumers. This controller structure that follows also from the system structure will be used here for a distributed MPC design. The models of the subsystems were derived in Section 5.1.1. It has been shown in Section 5.2.1 that the model of the overall plant belongs to the class of MTI systems. Thus the overall model is also described by an MTI state space model with decomposed parameter tensors

$$\begin{aligned}\dot{\mathbf{x}} &= \langle\, \mathsf{F} \mid \mathsf{M}\,(\mathbf{x}, \mathbf{u}, \mathbf{d}) \,\rangle\,, \\ \mathbf{y} &= \langle\, \mathsf{G} \mid \mathsf{M}\,(\mathbf{x}, \mathbf{u}, \mathbf{d}) \,\rangle\,.\end{aligned}$$

Each consumer has two states and two control signals. The boiler has one state and one control signal only. The outside temperature and the solar radiation are the disturbances to the plant. This results in $2N_{HC} + 1$ states, $2N_{HC} + 1$ control signals and 2 disturbances of the overall heating system. In case of a central MPC design, this model is used inside the controller. The central control loop is shown in Figure 5.30.

The central MPC controller sets the modulation signal of the boiler and the control signals of the pumps and valves of the consumers, such that the room temperatures in the particular heating circuits are on their reference values, i.e. they are in the comfort range. Therefore,

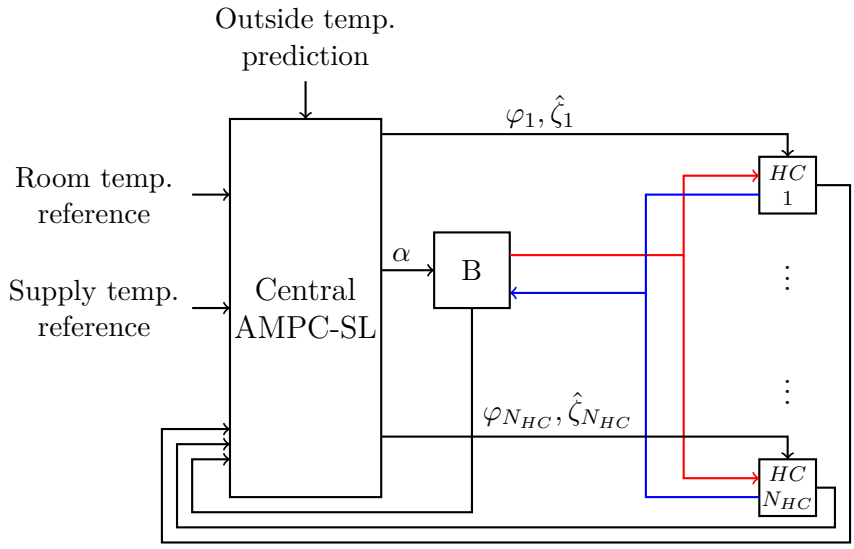


Figure 5.30: Central MPC for the heating system example

the cost function is defined as

$$J = \sum_{i=1}^{N_{HC}} \left(\sum_{j=1}^{H_p} \left\| T_{b,i}(k+j) - T_{b,i,ref}(k+j) \right\|_{q_{HC,i}} + \sum_{j=0}^{H_u-1} \left\| \Delta \mathbf{u}^{(HC,i)}(k+j) \right\|_{\mathbf{R}_{HC,i}} \right)$$
$$+ \sum_{j=1}^{H_p} \left\| T_s(k+j) - T_{s,ref}(k+j) \right\|_{q_B} + \sum_{j=0}^{H_u-1} \left\| \Delta u^{(B)}(k+j) \right\|_{r_B},$$

with the weighting factors q_B and r_B for the boiler as well as $q_{HC,i}$ and $\mathbf{R}_{HC,i}$ for the consumers $i = 1, \ldots, N_{HC}$. Predictions of the outside temperature and the solar radiation are used as weather forecast. In each sampling step the optimal input trajectory $\hat{\mathbf{U}}^{(i)}$ is computed for each input signal for the whole control horizon H_u by solving the optimization problem (4.51). The number of optimization variables follows from the product $(2N_{HC}+1)H_u$ of the number of control signals and the control horizon. The complexity of the optimization problem increases with the size of the search space. Therefore, methods should be found to reduce the complexity. The length of the control horizon depends on the dynamics of the plant. Thus, it should be chosen as short as possible to get still a good control result, but do not loose the benefits of the predictive controller actions. But it cannot be reduced arbitrary. The idea here is to split the overall optimization problem into several smaller subproblems with smaller search spaces. The global optimization goal for the whole plant should still be in focus. The boiler and each consumer gets its own controller node, that acts only locally. The global optimization problem is distributed to these controllers of the subsystems. Therefore, the number of control signals, that has to be computed by each controller is significantly smaller than in the central case such that also the computational effort is reduced enormously. The controller nodes of the consumer have to determine 2 control signals. This is a significant reduction compared to the central case. Furthermore the complexities of the particular optimizations do not scale with the size of the plant, i.e. the number of consumers. The single subproblems still have the same number of control

signals. For the central problem the number of control signals increases linearly with the number N_{HC} of consumers. To get a good optimization result for the overall plant, even though the problem is split to several subproblems, a communication between the controller nodes is necessary. The nodes of the heating circuits communicate the predictions of their future heat demands to the controller of the boiler. With this information the controller determines the reference of the supply temperature, such that the boiler can provide hot water with temperature in a reasonable range as requested from the consumers. The specific local controller nodes are presented in the following in detail.

Consumer with mixing circuit

The consumers are modeled as derived in Section 5.1.1 where the models have two states (5.1) and (5.2)

$$\mathbf{x}^{(HC,i)} = \begin{pmatrix} T_{r,i} & T_{b,i} \end{pmatrix}^T, \ i = 1, \ldots, N_{HC},$$

with the return and room temperature. The actuators are the pumps and the valve, such that the control signals are given by

$$\mathbf{u}^{(HC,i)} = \begin{pmatrix} \varphi_i & \zeta_i \end{pmatrix}^T.$$

The particular heating circuit is influenced by the supply temperature coming from the boiler and the weather conditions. Both cannot be influenced by the local controller of the consumer, such that they are treated as internal and external disturbances for this subproblem

$$\begin{aligned} s^{(HC,i)} &= T_{s,total}, \\ \mathbf{d}^{(HC,i)} &= \begin{pmatrix} T_o & \dot{Q}_{solar} \end{pmatrix}^T. \end{aligned}$$

Each consumer gets its own controller node that works with the MTI model of the specific consumer to determine, which supply temperature and which volume flow is necessary to realize a reference tracking of the room temperature to fulfill the comfort demands of the users. The reference is given e.g. by an occupation profile of the specific heating circuit. The reference tracking of the room temperature is described by the cost function

$$J_{HC,i} = \sum_{j=1}^{H_p} \left\| T_{b,i}(k+j) - T_{b,i,ref}(k+j) \right\|_{q_{HC,i}} + \sum_{j=0}^{H_u-1} \left\| \Delta \mathbf{u}^{(HC,i)}(k+j) \right\|_{\mathbf{R}_{HC,i}}.$$

The predictive controller computes the desired supply temperature $T_{s,HCi}$ of the heating circuit and the control signal of the pump. The three way valve is not considered inside the model for the controller. It is assumed that the valve position can be set afterwards such that the desired supply temperature is achieved as mixing product. Thus the model consists of the temperature dynamics (5.1) and (5.2) of the consumer and the pump equation (5.4) with states $\begin{pmatrix} T_{r,i} & T_{b,i} \end{pmatrix}^T$, inputs $\begin{pmatrix} \varphi_i & T_{s,HCi} \end{pmatrix}^T$ and disturbances $\begin{pmatrix} T_o & \dot{Q}_{solar} \end{pmatrix}^T$. The controller computes the desired supply temperature of the consumer, that is set by the three way valve

using a PI controller. It is assumed that this underlying control loop works perfectly and that this process is faster than the sampling time of one minute, such that the control signal of the valve is described by a static model here

$$\zeta_i = \frac{T_{s,HCi} - T_{s,total}}{T_{r,i} - T_{s,total}} \in [0,1]\,.$$

The local control loop of the consumer is depicted in Figure 5.31.

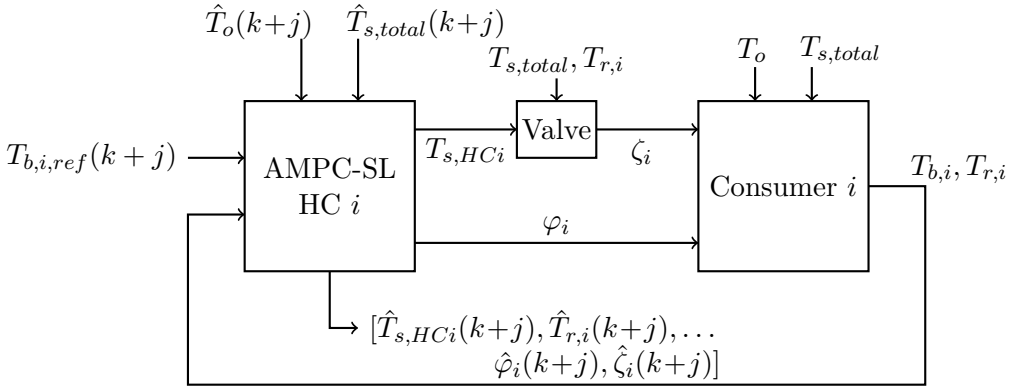


Figure 5.31: Local control loop of a consumer with predictions $j = 1, \ldots, H_p$ for the whole prediction horizon

The consumer receives information on the predicted future overall supply temperature from the boiler. By using this data together with the computed trajectories of the own control signals, the controller of the consumer computes predictions of its return temperature, its volume flow and its supply temperature demand by a forward simulation of the model (5.1), (5.2) and (5.4) for the prediction horizon.

Boiler

Inside the controller node of the boiler, the model (5.5) is used for the AMPC-SL algorithm. The model has one state, the supply temperature $x^{(B)} = T_{s,total}$ and one input, the modulation signal $u^{(B)} = \alpha$. The overall return temperature and the flow are internal disturbances

$$\mathbf{s}^{(B)} = \begin{pmatrix} T_{r,total} & \dot{V}_{total} \end{pmatrix}^T .$$

Based on the predictions of the heat demand of the consumers the controller receives a reference $T_{s,total,ref}$ for the supply temperature. The boiler should follow this reference resulting in the cost function

$$J_B = \sum_{j=1}^{H_p} \left\| T_{s,total}(k+j) - T_{s,total,ref}(k+j) \right\|_{q_K} + \sum_{j=0}^{H_u-1} \left\| \Delta\alpha(k+j) \right\|_{r_B} .$$

The disturbances of the boiler coming from the consumers are the overall return temperature and the overall flow, that are available as predictions from the consumer controller nodes.

Figure 5.32 shows the local control loop of the boiler. Predictions $\hat{T}_{s,total}(k+j)$ of the future generated supply temperatures are determined by a forward simulation of (5.5) for the prediction horizon.

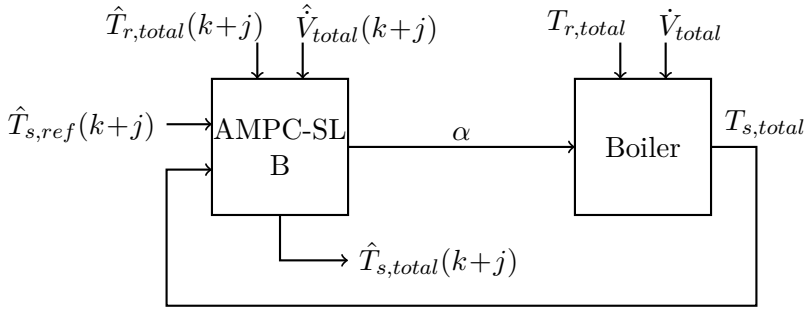


Figure 5.32: Local control loop of the boiler with predictions $j = 1, \ldots, H_p$ for the whole prediction horizon

Coordinator

As previously said, the controller of the boiler needs information on the future supply temperature reference, overall return temperature and the overall flow. The controllers of the consumers provide predictions for their own heating circuit respectively. These information is collected by the coordinator as shown in Figure 5.33.

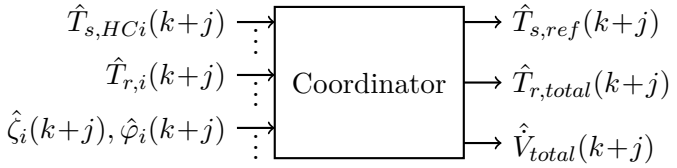


Figure 5.33: Coordinator of the boiler with predictions $j = 1, \ldots, H_p$ for the whole prediction horizon

The coordinator gets predictions on the future return temperatures and flows of the particular consumers. In the overall heating system, the flows coming from the consumers are merged by a return flow collector. The resulting mixing product has the flow (5.6) and temperature (5.7). Furthermore the coordinator determines a reference for the overall supply temperature from the supply temperature requests of the consumers. The maximal supply temperature demand from the consumer is selected for each time step of the prediction horizon

$$T_{s,total,ref}(k+j) = \max_{i=1,\ldots,N_{HC}} \left(T_{s,HCi}(k+j)\right) \quad \forall j = 1, \ldots, H_p.$$

Furthermore, the coordinator transmits predictions on the future supply temperatures generated from the boiler to the controller nodes of the consumer. The setup of the distributed controller with the nodes as described before is shown in Figure 5.34.

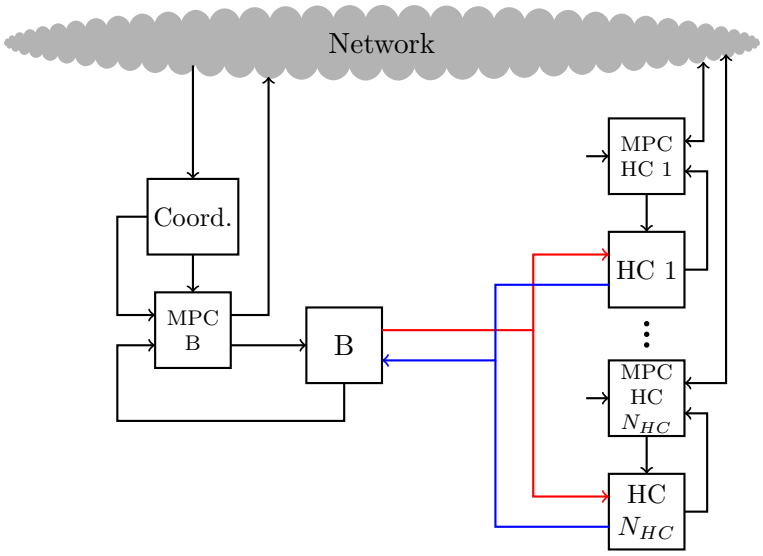


Figure 5.34: Distributed predictive controller for the overall system

As a simulation example a heating system with $N_{HC} = 3$ heating circuits is investigated. The structure is given as before, but here the controller was designed for a building with less masses, i.e. the room temperature reacts faster on changing supply. A central and a distributed MPC are designed for that plant. The controllers work with a sample time of 60 s and different prediction horizons were investigated. The reference for the building temperatures for the different heating circuits is implemented, such that a night reduction of the temperatures is realized. Since the time constants of the plant are smaller, a rectangular reference is used in contrast to the very slow building investigated in Section 5.3.3. The different controller nodes work with the AMPC-SL algorithm. The MTI models and cost functions of the subsystems are used as described before. Figure 5.35 shows the simulation results for the closed loop simulation with the DMPC for the 3 building temperatures and a prediction horizon of 60 min.

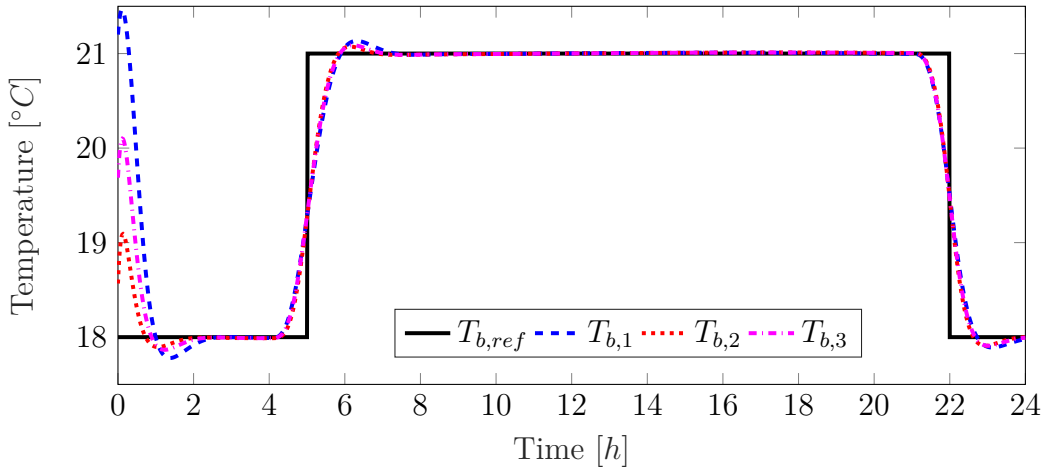


Figure 5.35: Simulation result of the building temperatures for one day

The simulation results show, that the room temperatures follow their references very well, even though the optimization task is distributed to the different controller nodes for each of the three consumers and the boiler. The different rooms start at different room temperatures. The controllers drive all rooms to the desired reference temperatures. Small undershoots and overshoots occur at the edges of the reference signal but they are in acceptable limits. The controllers for the different components compute their control signals independently by exchanging some information on their future behavior at the end of each time step. It was also investigated, whether the control result can be improved if the controllers exchange information several times during one time instant and performing the optimization iteratively with updates from the other nodes before going on to the next time step. This leads to more computational effort and did not show significant benefit in the control result. Even though a building with less masses, i.e. it reacts faster, was used here, the system dynamics are slow enough compared to the sampling time, that it is sufficient to compute the optimization only once during one time step.

This shows, that the closed loop with distributed predictive controllers leads to a desired system behavior. The big advantage compared to the central design gets clear when focusing on the computational complexity of the controllers. The complexity is measured here by the time that is necessary to solve the particular optimization problems. Therefore, t_{MPC} denotes the time to solve the optimization problem of the central MPC. The optimization problem of the DMPC is solved in time t_{DMPC}, that is composed of the times necessary to solve the optimization problems of the particular controller nodes of the boiler t_B and the consumers $t_{HC,i},\ i = 1, 2, 3$. Since there are no interconnections between the optimization problems of the heating circuits, they are computed in parallel. The MPC of the boiler is computed afterwards using the heat demand predictions of the consumers. Because of that, the computational time for the distributed MPC is given by the sum of the time for the solution of the optimization problem of the boiler and the maximum time for the solution of the consumer problems, since they are computed in parallel, leading to

$$t_{DMPC} = t_B + \max_{i=1,\dots,N_{HC}} t_{HC,i}.$$

The simulation was executed on a computer with Intel Core i7-3540M processor with 3.0 GHz and 16 GB working memory. The mean times for solving the optimization problems are summarized in Table 5.5.

Table 5.5: Comparison of the mean times for solving the optimization problems

$\mathbf{H_p}$	$\mathbf{t_{MPC}}$	$\mathbf{t_B}$	$\mathbf{t_{HC,1}}$	$\mathbf{t_{HC,2}}$	$\mathbf{t_{HC,3}}$	$\mathbf{t_{DMPC}}$
60 min	3.39 s	0.045 s	0.115 s	0.121 s	0.118 s	0.166 s

The prediction horizon was fixed to one hour and chosen equally to the control horizon, which leads to horizons of 60 time steps with the sampling time of $T_{sample} = 60\,\text{s}$. Therefore, the central MPC computes the control signal trajectory for 7 control signals, since each consumer has 2 actors and the boiler has 1 actor. This leads to a search space of $7 \cdot 60 = 420$ optimization variables. The controller nodes of the DMPC work with the actors of the

particular subsystems only. This results in $1 \cdot 60 = 60$ optimization variables for the boiler with 1 control signal and $2 \cdot 60 = 120$ optimization variables for each consumer with 2 control signals. The effect of the different complexities is displayed for this example in Table 5.5. By distributing the controller task to several nodes a significantly reduced computation time $t_{MPC} > t_{DMPC}$ can be achieved, because the particular subproblems of the controller nodes are computed much faster than the optimization problem of the central MPC. There is a strong dependence between the size of the search space and the computation time here. Already for the subsystems of the DMPC one can see that the subproblem of the boiler with one control signal is solved faster than the problems of the consumers with 2 control signals. Compared with the central problem this advantage in computational complexity is even more significant.

If the prediction and control horizons are increased, this leads also to an increase of the complexity of the optimization problem since the size of the search space gets bigger. Figure 5.36 shows, how the time for the solution of the optimization problem evolves for different prediction horizons for the central and the distributed MPC. The control horizon is chosen equally to the prediction horizon.

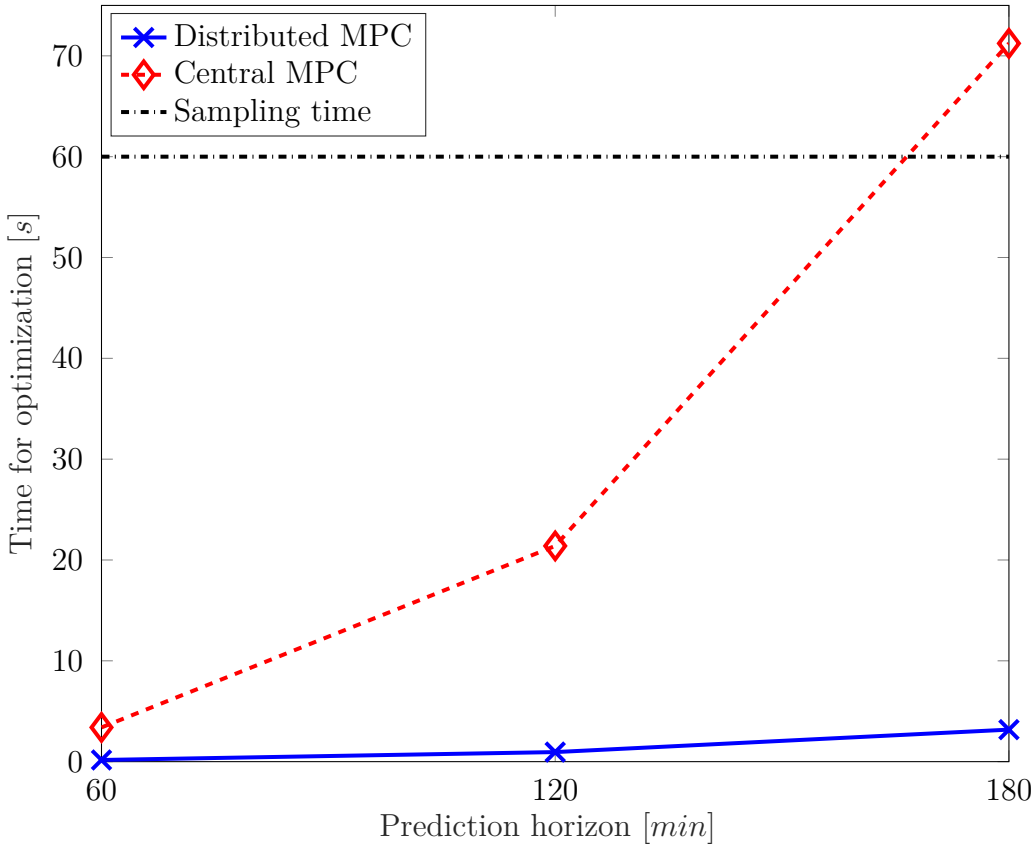


Figure 5.36: Time for the solution of the optimization problem for different prediction horizons ($H_p = H_u$)

It again gets obvious that the time for solving the optimization problem increases much faster, when using a central MPC. With an prediction horizon of 3 h, the optimization time in the central case is larger than the sampling time of 60 s, which is not acceptable for a real-time implementation, since in this case the solution of the optimization problem has to be available at the end of each sampling time step. The optimization time of the DMPC shows a significantly slower increase, such that in this case much longer prediction horizons would be possible until the optimization time gets next to the sampling time. To summarize one can say, that the application example showed, that the complexity of the MPC optimization problem can be reduced enormously, when the control task is distributed to several controller nodes. The subproblems are smaller and can be computed partly in parallel. Since the complexity increases much slower than in the central case, the DMPC allows also larger prediction and control horizons. Additionally one can assume, that also a larger number of consumers leads to an significant increase in complexity in the central case since the number of optimization variables increases. For the DMPC the increasing number of consumers has a smaller effect on the computational complexity since the search spaces of the particular subproblems do not change. Only the number of small subproblems increases but a parallel computation is possible. This results in great benefits for the application of a distributed approach especially for large-scale plants. The application also shows the validity of the combination of a DMPC approach with controller nodes working with the AMPC-SL algorithm introduced in Section 4.5.

5.3.5 Real-time implementation

In Section 4.5 an MPC method was adapted to the special model structure of MTI systems resulting in the AMPC-SL method. The method was successfully applied in simulation to a building example, i.e. a heating circuit, in Section 5.3.3. This section describes the real-time implementation of the approach and the application of the resulting controller at a real world office building with a heating circuit structure as considered before. To apply the controller, the algorithm has to be implemented on a real-time hardware. To test the real-time implementation a HIL platform is developed. After successful tests the controller is operated at the real building. The results will be presented in the following. At first the used hardware and software is described shortly.

Hardware and software

The hardware for controller implementation should have sufficient computational power, communication interfaces and is ideally of low cost. That is why the single-board computer Raspberry Pi was chosen, that is very popular in prototypical applications. This is a computer of small size of 93 mm × 63.5 mm × 20 mm, where all components like the processor, the working memory or the interfaces are installed on one board. No hard drive is provided. As storage device a SD card is used, that is bootable and contains the operating system, which is the Linux distribution "Raspbian" here, that is optimized for operation on Raspberry Pi.

Different versions of the Raspberry Pi are available. Here the version Raspberry Pi 3 Version B is used with a quad core processor with a clock rate of 1.2 GHz and 1 GB RAM working memory. Different interfaces are available, like USB or Ethernet ports or GPIO pins that are freely configurable. Due to this high flexibility and computational power the Raspberry Pi is chosen as implementation hardware.

Modeling and controller design is done with the MATLAB and Simulink software package. Besides the standard toolboxes additional toolboxes for work with Raspberry Pi and predictive controllers are used. To connect MATLAB with the Raspberry Pi environment, support packages are available. This allows e.g. to manipulate and read the communication interfaces of the Raspberry Pi like the GPIO or the Ethernet interface by UDP with MATLAB and Simulink software. Additionally an automatic C code generation is provided to deploy algorithms developed in Simulink to the Raspberry Pi and execute them on the hardware independently, without using a host PC, which is desired here. For that, it is necessary that functions and data types are used only, that are supported for code generation. For the design of the predictive controller the "MPC Toolbox" of MATLAB is used, because it offers efficient QP solvers, that are supported for code generation, to solve the standard MPC optimization problem, [9]. This functionality is extended with own code to implement the AMPC-SL method for decomposed MTI systems.

Real-time simulator for MTI systems

A plant simulator helps to test controllers before operation at the real plant. Therefore a simulator for MTI models is developed here, such that the model runs on a Raspberry Pi. The hardware component runs with a fixed sample time, such that a discrete time model

$$\begin{aligned} \mathbf{x}(k+1) &= \langle\, \mathsf{F} \mid \mathsf{M}\left(\mathbf{x}(k), \mathbf{u}(k)\right) \rangle\,, \\ \mathbf{y}(k) &= \langle\, \mathsf{G} \mid \mathsf{M}\left(\mathbf{x}(k), \mathbf{u}(k)\right) \rangle\,. \end{aligned}$$

is considered. To parameterize the MTI model on Raspberry Pi, the parameter tensors F and G have to be provided in decomposed form. In Section 5.2.3 the CP decomposition turned out to be well suited for MTI model representation of heating systems. But the standard toolboxes [5] and [109] for CP tensor representation work with the MATLAB data types `struct` and `cell`. Since these data types are not supported for code generation, alternative storage formats for CP parameter tensors of MTI systems have to be derived for that. The parameter tensor of the state equation of an MTI system with n states and m inputs is given by

$$\mathsf{F} = [\mathbf{F}_{u_m}, \ldots, \mathbf{F}_{u_1}, \mathbf{F}_{x_n}, \ldots, \mathbf{F}_{x_1}, \mathbf{F}_{\Phi}] \cdot \boldsymbol{\lambda}_F,$$

with $r_{cp}(\mathsf{F})$ rank-1 components, factor matrices $\mathbf{F}_{u_i}, \mathbf{F}_{x_i} \in \mathbb{R}^{2 \times r_{cp}(\mathsf{F})}$, $\mathbf{F}_{\Phi} \in \mathbb{R}^{n \times r_{cp}(\mathsf{F})}$ and weighting vector $\boldsymbol{\lambda}_F \in \mathbb{R}^{r_{cp}(\mathsf{F})}$. The parameter tensor of the output equation with p outputs reads

$$\mathsf{G} = [\mathbf{G}_{u_m}, \ldots, \mathbf{G}_{u_1}, \mathbf{G}_{x_n}, \ldots, \mathbf{G}_{x_1}, \mathbf{G}_{\Phi}] \cdot \boldsymbol{\lambda}_G$$

with $r_{cp}(\mathsf{G})$ rank-1 components, factor matrices $\mathbf{G}_{u_i}, \mathbf{G}_{x_i} \in \mathbb{R}^{2\times r_{cp}(\mathsf{G})}$, $\mathbf{G}_\Phi \in \mathbb{R}^{p\times r_{cp}(\mathsf{G})}$ and weighting vector $\boldsymbol{\lambda}_G \in \mathbb{R}^{r_{cp}(\mathsf{G})}$. Data formats supported for code generation are arrays of data type `double`. Here it is exploited, that the factor matrices $\mathbf{F}_{u_i}$ and $\mathbf{F}_{x_i}$ as well as $\mathbf{G}_{u_i}$ and $\mathbf{G}_{x_i}$ are all of same sizes $\mathbb{R}^{2\times r_{cp}(\mathsf{F})}$ or $\mathbb{R}^{2\times r_{cp}(\mathsf{G})}$, respectively. Because of that, the factor matrices can be concatenated in a third dimension

$$\begin{aligned}\mathsf{F}_{xu} &= \mathbf{F}_{u_m} \boxplus_3 \cdots \boxplus_3 \mathbf{F}_{u_1} \boxplus_3 \mathbf{F}_{x_n} \boxplus_3 \cdots \boxplus_3 \mathbf{F}_{x_1},\\ \mathsf{G}_{xu} &= \mathbf{G}_{u_m} \boxplus_3 \cdots \boxplus_3 \mathbf{G}_{u_1} \boxplus_3 \mathbf{G}_{x_n} \boxplus_3 \cdots \boxplus_3 \mathbf{G}_{x_1},\end{aligned}$$

resulting in tensors of dimensions $\mathbb{R}^{2\times r_{cp}(\mathsf{F})\times n+m}$ and $\mathbb{R}^{2\times r_{cp}(\mathsf{G})\times n+m}$. The particular factor matrices are selected by the last dimension, e.g. for F_{xu}

$$\begin{aligned}\mathbf{F}_{xu}(:,:,1) &= \mathbf{F}_{u_m},\\ \mathbf{F}_{xu}(:,:,n+m) &= \mathbf{F}_{x_1}.\end{aligned}$$

Furthermore the factor matrices $\mathbf{F}_\Phi$ and $\mathbf{G}_\Phi$ and the weighting vectors $\boldsymbol{\lambda}_F$ and $\boldsymbol{\lambda}_G$ have to be stored separately. With this storage format all arrays are of data type `double` and supported for code generation.

To simulate the discrete-time model, the model equations are evaluated iteratively. This means, that the contracted products of parameter tensors and monomial tensor have to be solved to determine $\mathbf{x}(k+1)$ and $\mathbf{y}(k)$. The contracted product is computed based on the factor matrices of the CP decomposition by (3.21). To consider the proposed storage format the algorithm has to be adapted resulting in

$$\begin{aligned}\mathbf{x}(k+1) =& \mathbf{F}_\Phi \left(\boldsymbol{\lambda}_F \circledast \left(\mathbf{F}_{xu}(:,:,1)^T \begin{pmatrix}1\\ u_m\end{pmatrix}\right) \circledast \cdots \circledast \left(\mathbf{F}_{xu}(:,:,m)^T \begin{pmatrix}1\\ u_1\end{pmatrix}\right) \cdots\right.\\ &\left. \circledast \left(\mathbf{F}_{xu}(:,:,m+1)^T \begin{pmatrix}1\\ x_n\end{pmatrix}\right) \circledast \cdots \circledast \left(\mathbf{F}_{xu}(:,:,n+m)^T \begin{pmatrix}1\\ x_1\end{pmatrix}\right)\right),\\ \mathbf{y}(k) =& \mathbf{G}_\Phi \left(\boldsymbol{\lambda}_G \circledast \left(\mathbf{G}_{xu}(:,:,1)^T \begin{pmatrix}1\\ u_m\end{pmatrix}\right) \circledast \cdots \circledast \left(\mathbf{G}_{xu}(:,:,m)^T \begin{pmatrix}1\\ u_1\end{pmatrix}\right) \cdots\right.\\ &\left. \circledast \left(\mathbf{G}_{xu}(:,:,m+1)^T \begin{pmatrix}1\\ x_n\end{pmatrix}\right) \circledast \cdots \circledast \left(\mathbf{G}_{xu}(:,:,n+m)^T \begin{pmatrix}1\\ x_1\end{pmatrix}\right)\right).\end{aligned}$$

Since in this approach standard operations like summation or multiplication are used and all components are stored as arrays of data type `double`, it is now possible to generate code and to implement the MTI plant simulator on hardware components like Raspberry Pi. The inputs $\mathbf{u}$ can be received e.g. from the GPIO pins or the UDP interface. The outputs can be communicated by the same ports. This allows e.g. the connection to a controller prototype.

Real-time implementation AMPC-SL

The AMPC-SL controller design for the heating circuit of Section 5.3.3 should be implemented for real-time operation on the Rapsberry Pi. The controller is implemented in Simulink on the host computer. Afterwards an automatic C code generation is executed, to deploy

the controller on the Raspberry Pi. The Simulink implementation is composed of two main parts. That is on the one hand the linearization as described in Section 3.3. For the standard MPC routine the "MPC Toolbox" of MATLAB is used with the extension, that the linear model is updated in each sampling time step, [9]. It is also possible to use time depended constraints or weights as necessary here for day and night operation. For the solution of the quadratic MPC optimization problem the KWIK algorithm, i.e. a multi-step Newton method, is used, [96]. The basic implementation of the AMPC-SL controller is illustrated in Figure 5.37.

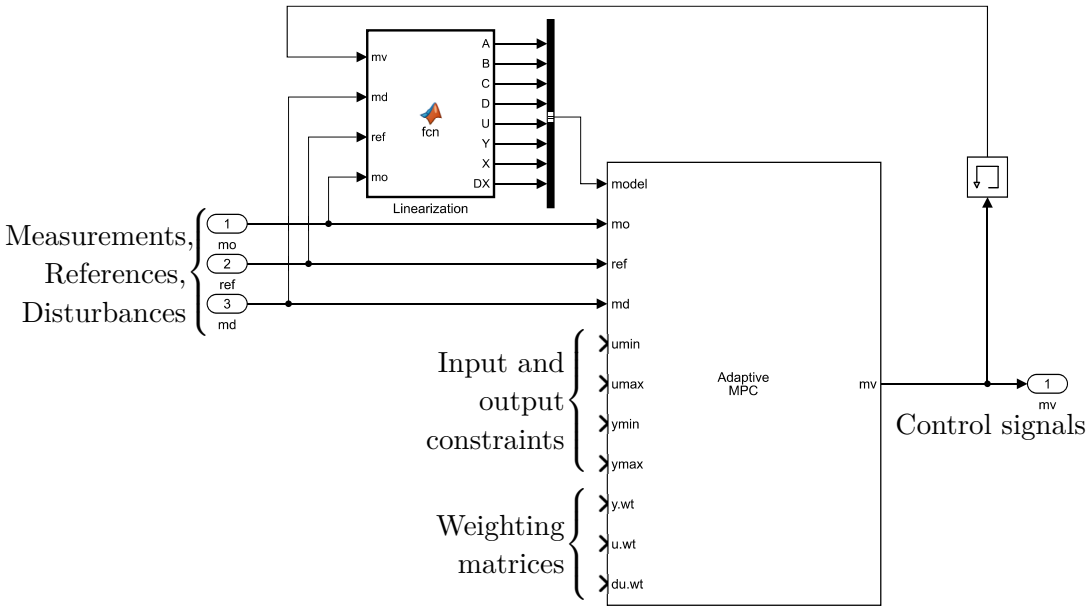


Figure 5.37: Basic Simulink implementation of the AMPC-SL for MTI systems

HIL test environment

It was described in the previous parts, how to implement an MTI model and the AMPC-SL controller on a Raspberry Pi for real-time operation. In a typical controller synthesis process, the controller is designed on the host PC and at first tested with a model of the plant in closed loop simulations. If the controller fulfills all requirements the hardware implementation follows. Before applying this implementation to the real plant it has to be tested, if the hardware implementation works as designed on the host computer before. To check the controller in closed loop it is operated with a plant simulator. This HIL environment is build up by two Raspberry Pis. On the first Raspberry the controller that should be tested is implemented. The second Raspberry implements the MTI plant model. Measurements and prediction values as well as the control signals are communicated between the components via the network protocol UDP. The setup is depicted in Figure 5.38.

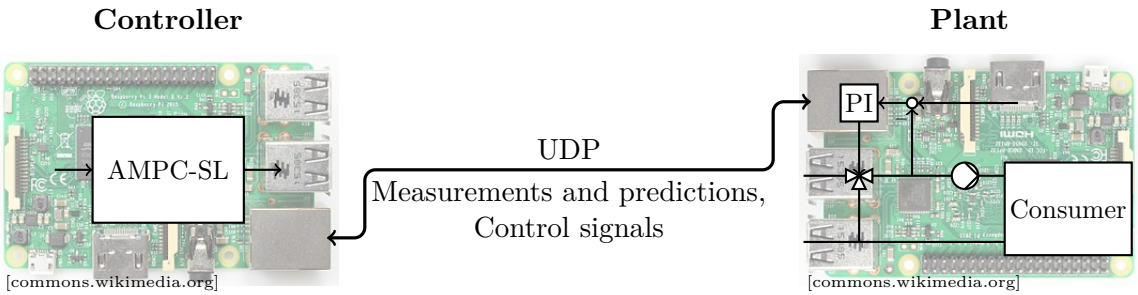


Figure 5.38: Hardware-in-the-loop environment

This leads to a test environment for the controller implementation in closed loop. It is important to test here, if the algorithm works in real-time on the proposed hardware with the desired sampling time. For predictive controllers it is important, that the result of the optimization is computed during one sampling time step, such that the control signal is available at the end of the sampling interval. It is not allowed that the computation takes longer than the sampling time for a proper operation. The closed loop HIL test of the AMPC-SL algorithm for the heating circuit proposed in Section 5.3.3 were successful. The controller works with a sampling time of one minute. During the tests with a prediction horizon of 3 hours and a control horizon of 2 hours no sampling time violations were observed. The controller results were as expected, i.e. like on the host computer as depicted e.g. in Figure 5.26. After passing these tests the controller is operated at the real building. The results are shown in the next part.

Controller operation with real building

After the successful test the AMPC-SL controller is operated at a heating circuit of a real office building, [81]. The setup is described in Section 5.3.3. The model parameters are identified with measurement data from this plant. The controller is implemented on a Raspberry Pi. Now the measurement signals are not computed by a plant model as it is the case during closed loop simulations or the HIL tests. They are received from the sensors of the real plant instead. In the heating circuit heat meters are installed measuring the supply and return temperature as well as the flow. The measurement information on the room temperature is generated by a temperature sensor inside a reference room on the floor belonging to the considered heating circuit. The predictions on the future outside temperature and the solar radiation are given by a weather station. The controller manipulates the real actor now, i.e. the supply temperature reference for the PI controller of the three way valve is set. The interface between the Raspberry Pi and the plant is the direct digital control (DDC) component. On the DDC the standard controllers for the whole building are implemented. Because of that, the DDC has access to all relevant sensors and actors of plant. Thus, it is straightforward to use the DDC as interface. The AMPC-SL controller reads measurement data from the DDC. The determined control signals are send to the DDC. On the DDC a plausibility check of the control signal is done. If the signals are fine, they are passed to the plant. If an implausible signal is detected, the DDC switches back to the conventional

control strategy with the heating curve. The communication infrastructure inside the plant between sensors, actuators and DDC is realized with the BACnet network protocol, a common communication protocol in building automation applications, [22, 41]. The controller is implemented in Simulink and has to be connected to this communication infrastructure. The controller implementation supports a network communication with the UDP protocol but not with the BACnet protocol directly. Because of that, a UDP to BACnet interface is necessary, [81]. The controller implementation sends and receives the data to and from its own component, i.e. the local host, by UDP. An interface is used on the Raspberry Pi, that converts the UDP packages to BACnet and the other way round. Control signals computed by the controller are send by UDP to the interface, that sends the signals to the DDC with the BACnet protocol. Measurement data is requested by the interface from the DDC via BACnet and converted, such that the data is provided by UDP for the controller. With that the communication outside the Raspberry Pi is performed with the BACnet protocol and no basic changes in the building communication infrastructure have to be made to apply the controller prototype. A schematic of the communication is illustrated in Figure 5.39.

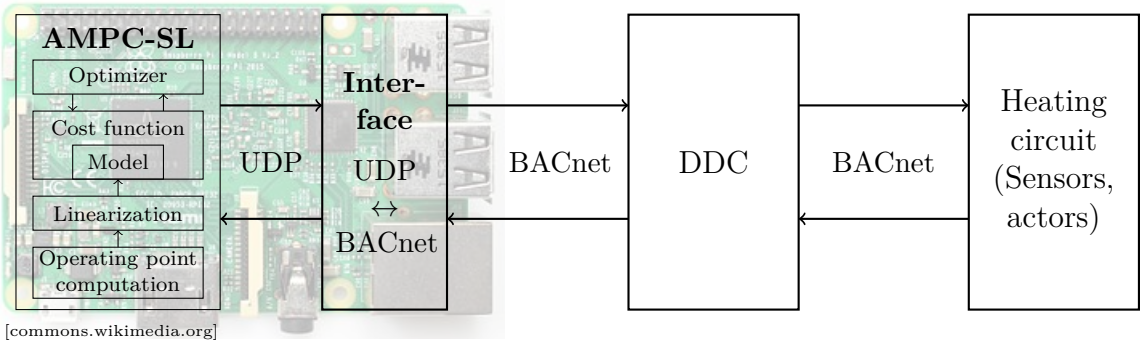


Figure 5.39: Communication setup

In this setup the communication interface and the controller algorithm are implemented on one hardware component. This avoids synchronization problems between different components. The controller is the master here that prescribes the sampling interval.

The controller was operated at the heating circuit of the considered office building for several weeks in the winter period. The hardware works very reliable over the whole period. The AMPC-SL algorithm runs very stable and the sampling time of one minute is held constantly. The controller shows a robust behavior against short-term network failures or errors in the measurement data acquisition. To detect such simple faults inside the controller plausibility intervals are defined for each signal to characterize nominal values. If e.g. an implausible measurement value arrives, i.e. it is outside the specified interval, this is detected and the control signal computed by the AMPC-SL is rejected. Instead a default value of 55 °C for the supply temperature reference is given to the plant. This is an empirical value. It is known from experience that the building does not cool down completely with this default supply temperature also for low outside temperatures. The result of the AMPC-SL would not be meaningful in this case, since e.g. with faulty measurement data the current state of the plant cannot be determined properly. If the controller receives plausible values again, an

automatic switch to the AMPC-SL algorithm follows and the control signals are send to the plant again. In the closed loop simulation in Section 5.3.3 a prediction horizon of three hours and a control horizon of two hours turned out as proper choices. Due to plant limitations the constraints on the supply temperature are set to

$$30\,^{\circ}\mathrm{C} \leq T_{s,ref} \leq 80\,^{\circ}\mathrm{C}.$$

The room temperature reference and constraints are used as described in Section 5.3.3. To optimize the controller, the parameters like the weighting factors were updated several times during the test period. The resulting closed loop behavior is illustrated by measurement data of one specific day at the end of the test period shown in Figure 5.40.

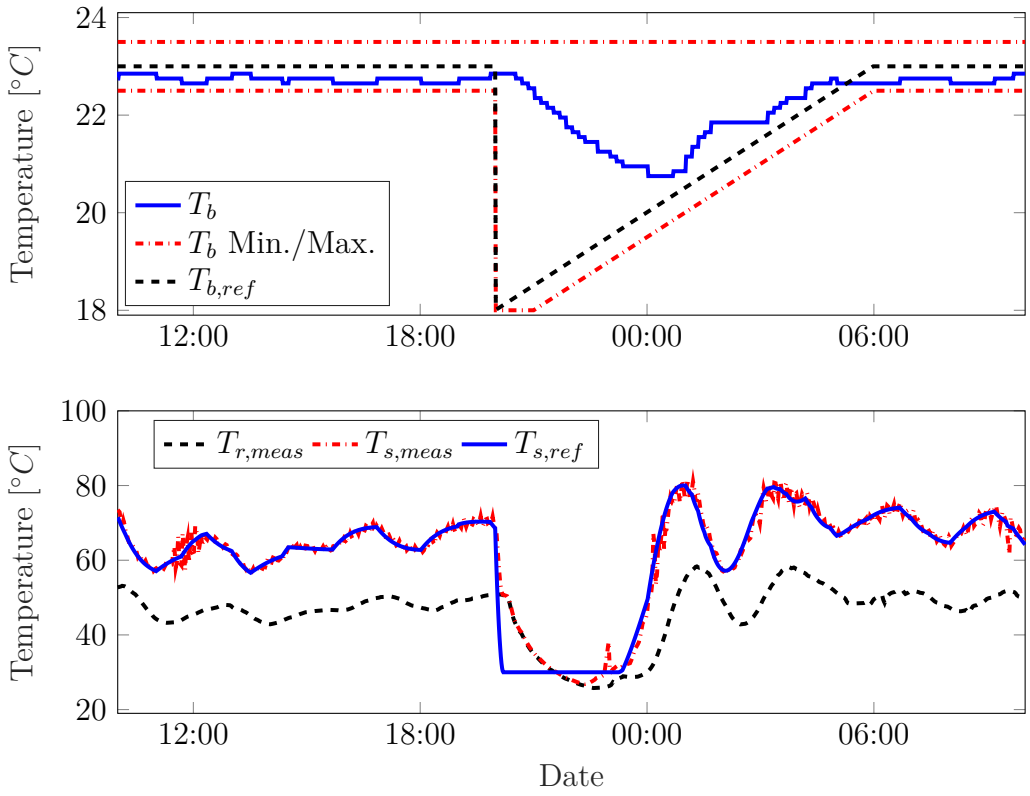


Figure 5.40: Measurements of the AMPC-SL tests from 13.12.2017, 10:00 o'clock to 14.12.2017, 10:00 o'clock

The measurement data shows, that during occupation hours the room temperature is inside its constraints. Only small deviations to the reference occur. This could follow from a model mismatch between the real plant and its MTI model. But the error is just small, such that it still fulfills the comfort requirements. The room temperature is held in the comfort range until the end of the occupation time at 20 : 00 o'clock. After that the room cools down. Energy is saved because the controller sets the supply temperature to its minimal value. In the heat up period at the beginning of the day the supply temperature is increased and the room temperature follows the reference during heat up. Here the measured room temperature is above the reference, which might be caused by differences between the time

constants for the heat up process between model and plant. The control signal, i.e. the supply temperature reference shows the expected trend. At night, a very low supply temperatures is necessary only. In the morning, high supply temperatures heat the building and again lower supply temperatures hold the room in the desired comfort range during the occupation time. The supply temperature reference shows some oscillations during the occupation phases when the room temperature is held constant. This might be reduced by model improvements or parameter adaptions, such that the controller reacts less aggressive, i.e. by increasing the factor r that penalizes the control effort. Experiences of the testes showed that a more conservative controller tuning compared to the closed loop simulations on the host computer is beneficial. Otherwise model mismatches or other disturbances like people inside the building or other internal loads would lead e.g. to oscillations, if the controller is too aggressive.

The test of the real-time implementation of the AMPC-SL controller at the heating circuit showed that the predictive control concept works well with a simple model at a real plant. The behavior of the heating circuit changed significantly compared to the conventional control strategy with a heating curve. One difference is the reduction of the temperature during the non occupation period. The heating curve is reduced by 10 K outside the time interval from 4:30 to 20:00 o'clock. The configuration of the heating curve and its reduction are based on experiences and a worst case estimation, such that the building fulfills the comfort requirements of the users surely at occupation time. But since the beginning of the heat up is fixed at 4:30 o'clock independently from the outside conditions, it may happen that too high supply temperatures are requested and the building is in the comfort range too early. The AMPC-SL is more flexible. Here the room temperature is controlled directly and the controller acts predictively. Because of that, more down cooling of the room temperature is allowed during the night or weekends leading to an operation with lower supply temperatures. The room temperature is measured the whole time and the controller would react if the room temperature falls below its minimal value. Furthermore, the starting time of heating up the building in the morning is not fixed any longer. It is determined with help of the model taking into consideration the environmental conditions like the outside temperature. Since the room temperature is reduced more, the supply temperatures in the morning are higher than in the case, where a heating curve is used. But the heating circuit is operated with significantly lower temperatures in the night or weekend. How much cool down is allowed and how the heat up process is characterized can be adapted by the AMPC-SL parameters. One possibility is shown here as proof of concept for the real-time implementation.

This leads to further research ideas that are beyond the scope of this thesis. With the AMPC-SL controller the temperature reduction in the heating circuit is realized taking into consideration many different factors of the plant. It is interesting to investigate the optimal setting for that. An open question is the optimal temperature reduction with respect to economical and energy efficiency criteria. If the room temperature is reduced very much this has the advantage that the heating circuit can be supplied with low temperatures in the night, but high temperatures are necessary in the morning. This can be adjusted by the parameters of the AMPC-SL. Economical and energy efficiency criteria have to be derived to optimize the parameters and thus the closed loop behavior. Up to now the controller was implemented on a prototypical hardware like the Raspberry Pi. The next step would be an

implementation on the building automation devices like the DDC or the BMS. To summarize this part, the real-time implementation showed that the AMPC-SL approach showed good closed loop results also at a real world plant. From this proof of concept different potentials are identified for further improvement of the closed loop behavior.

6 Conclusion

This chapter summarizes the main results of this thesis in Section 6.1 and draws conclusions. Section 6.2 highlights possible further directions of research.

6.1 Summary

The main result of the thesis are summarzed according to the research questions posed in Section 1.2.

- *Which tensor decomposition methods are suitable for the complexity reduction of MTI models and their controller design methods?*
  In mathematics developments like tensor decomposition methods offer very powerful tools. This thesis combines the methods of tensor calculus from the mathematics and informatics community with modeling and controller design methods from engineering. Tensor decomposition methods allow to represent state space models of MTI systems in a memory efficient way. Four decomposition methods were investigated and compared with the result, that CP, TT and HT are suited best, especially regarding the storage demand and the search for low-rank approximations.
- *How can tensor methods and the multilinear model structure improve the controller design process of known methods like feedback linearization or predictive control?*
  Many methods in modeling and control of MTI systems are based on polynomial operations. Therefore at first some mathematical tools are derived, that form a theoretical base for the upcoming methods in modeling and control. No full tensors should be used, such that all required tensor operations have to be defined for decomposed tensors. For computations with multilinear functions in tensor format a generalized tensor description for polynomials and different polynomial operations like multiplication or differentiation were derived, that work with the parameter tensors only. Using this memory efficient representation, tools like the linearization or discretization were developed, that can be used for model-based controller design with MTI models.
  With these methods controller design algorithms were derived especially for MTI models to get a less complex synthesis than in the general nonlinear case. The nonlinear control method of feedback linearization was specialized to multilinear models, such that it works with the decomposed parameter tensors of the model resulting in a controller of fixed tensor structure, that is computed by simple operations without any symbolical operations. This shows, that it is possible to infer design methods especially for MTI systems from general nonlinear methods with several benefits like a fixed structure and predictable complexity.

MPC is a promising advanced design method for control e.g. of heating systems. Often linear models are used with the advantage of a convex optimization problem, that has to be solved during operation to realize a reference tracking. But multilinear models capture the dynamics of many plants better than linear ones. Therefore the optimization problem of MPC with MTI models was of interest in this thesis. It was shown that in this case the convexity of the optimization problem is not guaranteed, but a test for convexity is provided. To combine the good modeling properties of the MTI models with the convex optimization with linear models, the AMPC-SL algorithm was developed, where the MTI model is linearized successively around an operating point next to the current operating conditions by using tensor methods. During optimization the adapted linear model is used to preserve the convexity. This approach based on the multilinear model shows advantages compared with the standard linear approach.

- *How to derive efficient distributed controller design methods for a multilinear model in tensor representation?*
  At first a notation was defined for large-scale systems, where the particular subsystems can be modeled by MTI systems, to describe the multilinear subsystems and their couplings. All three basis connections of two subsystems were investigated further to determine the parameters of the overall model from their single parts. To control large-scale plants two main approaches are presented. It was investigated how a controller structure of an MTI system is determined using a decentralized state feedback method. Based on a linear algorithm an approach was proposed, that finds a controller structure for an MTI system in a preprocessing step by solving an optimization problem. During operation the controller gain of fixed structure is adapted to the multilinear dynamics of the plant. This gives a state feedback controller with little less performance compared to the centralized design but significantly reduced communication effort.
  In the second approach the focus was on the distribution of the control task to several controller nodes in a given structure to reduce computational complexity of the particular controllers. It was shown, that the application of predictive control strategies to large-scale systems results in complex optimization problems. To avoid the increase in complexity the predictive control task is distributed to different controller nodes here. Each node works with the AMPC-SL technique controlling the own subsystem only. This leads to optimization problems of reduced complexity compared the the central case. Couplings with other subsystems are considered as predictable disturbances. This makes it possible to use advanced methods like AMPC-SL also for large-scale plants.
- *What are the benefits for the application of decomposed models and design methods to heating systems?*
  The developed methods were applied to two example systems, i.e. an HVAC system and a heating system of a large non-residential building. Multilinear models of the systems were derived and represented in decomposed format. Tensor decomposition methods were applied to reduce the storage complexity of the models leading to a memory efficient representation. The decomposed MTI models were used for controller design for the heating systems applications. This results in a feedback linearization controller and a AMPC-SL for a particular heating circuit and decentralized controllers

for the whole non-residential building. Closed loop simulations showed good results to maintain the comfort demand of the occupants and to realize an energy efficient operation, e.g. by an operation with lower supply temperatures compared to the conventional control with a heating curve.

- *Is it possible to implement a multilinear model-based controller on real-time hardware?* Because of the possibilities to control MIMO systems as well as to consider disturbance predictions by the weather forecast and the dynamics of the plant by an MTI model, the AMPC-SL algorithm turned out to be suited best for a real-time implementation. The controller was implemented on a Raspberry Pi single-board computer. After passing tests in a HIL environment, the controller was successfully applied to a heating circuit of an office building, which brought further insight into the plant operation with a predictive controller.

Thus, the thesis shows, that MTI models are represented very efficiently by different tensor decomposition methods, that can be used well for model analysis and controller design. This offers the possibility to optimize the operation of large-scale heating systems, whose thermal dynamics are very well captured by MTI models. An implementation example shows the applicability of the methods also for real plants.

6.2 Outlook

The outlook shows the most promising future research directions by summarizing and rating the open questions of Sections 2.4, 3.8 and 4.7 according to impact and effort.

As it is the mathematical base for many of the introduced methods an improvement of the tensor representation of polynomials would have a great effect on the presented methods. By reducing the redundancies in the monomial tensor for higher order functions, the description offers a memory saving potential. For this and all other tensor approaches in the thesis new developments in the tensor community have to be studied and their applicability to MTI systems has to be checked.

From a scientific point of view the investigation of the closedness of the class of MTI models is very interesting, e.g. regarding the feedback connection. Since no approaches for e.g. a variable transformation of MTI systems is available for this application yet more research effort has to be spent here.

Different steps of the model-based design process for MTI systems were investigated in this thesis from model representation and analysis to controller synthesis. But to adapt the model to a specific plant, values of the parameters are required. No parameter estimation method specifically for MTI models is available yet. Especially black-box identification techniques would simplify the model creation step enormously. Since an MTI model is required to apply the presented design methods, a simplification of the modeling process, that is often regarded as complex, would have a great impact on the model-based design application for MTI models. The first approaches in [24] showed, that the development of algorithms for that is complex and further research is necessary. First results are given in [7, 8]. The same holds for simulation algorithms specialized for MTI models.

In the field of controller design the development of specific optimization algorithms, when multilinear models are used, would have an impact on many optimal control methods like MPC or the decentralized feedback design in this thesis. MPC is a very powerful design method for building applications. Heating systems are modeled very well in a multilinear way, such that this is an important research topic. For plants of larger scale a DMPC method was introduced for an example here. It is interesting, if controller structures for a distributed MPC can be derived from the proposed optimal decentralized state feedback scheme. Other properties of the DMPC like stability and convergence have to be investigated further too. In contrast to these complex further research directions other improvements of the control methods e.g. the extension of the feedback linearization to MIMO systems are straight forward.

For practical applications predictive methods are of high interest. Future work should extend the prototypical implementation of the AMPC-SL approach for one heating circuit to the distributed design for the whole plant. The next steps would be the implementation of the algorithms on on-site hardware like the DDC or BMS level to integrate the approach completely in the building automation to realize an energy efficient operation.

Bibliography

[1] Afram, Abdul and Janabi-Sharifi, Farrokh: "Theory and applications of HVAC control systems–A review of model predictive control (MPC)". In: *Building and Environment* 72 (2014), pp. 343–355.

[2] Alessio, Alessandro and Bemporad, Alberto: "A Survey on Explicit Model Predictive Control". In: *Nonlinear Model Predictive Control: Towards New Challenging Applications*. Ed. by Magni, Lalo; Raimondo, Davide Martino, and Allgöwer, Frank. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 345–369.

[3] Aste, Niccolò; Manfren, Massimiliano, and Marenzi, Giorgia: "Building Automation and Control Systems and Performance Optimization: A Framework for Analysis". In: *Renewable and Sustainable Energy Reviews* 75 (2017), pp. 313 –330.

[4] Astrom, Karl Johan and Murray, Richard M.: *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton, NJ, USA: Princeton University Press, 2008.

[5] Bader, Brett W.; Kolda, Tamara G., et al.: *MATLAB Tensor Toolbox Version 2.6*. Available online. 2015. url: `http://www.sandia.gov/~tgkolda/TensorToolbox/`.

[6] Bakule, Lubomir: "Decentralized control: An overview". In: *Annual Reviews in Control* 32.1 (2008), pp. 87 –98.

[7] Batselier, Kim; Ko, Ching - Yun, and Wong, Ngai: "Tensor Network Subspace Identification of Polynomial State Space Models". In: *Automatica* 95 (2018), pp. 187 –196.

[8] Batselier, Kim; Ko, Ching-Yun; Phan, Anh-Huy; Cichocki, Andrzej, and Wong, Ngai: "Multilinear State Space System Identification with Matrix Product Operators". In: *IFAC-PapersOnLine* 51.15 (2018). 18th IFAC Symposium on System Identification SYSID 2018, pp. 640 –645.

[9] Bemporad, Alberto; Morari, Manfred, and Ricker, N. Lawrence: *Model Predictive Control Toolbox User's Guide*. The Mathworks Inc., 2016.

[10] Bigalke, Uwe; Armbruster, Aline; Lukas, Franziska; Krieger, Oliver; Schuch, Cornelia, and Kunde, Jan: *Der dena-Gebäudereport 2016: Statistiken und Analysen zur Energieeffizienz im Gebäudebestand*. Deutsche Energie - Agentur GmbH (dena), 2016.

[11] Bloemen, H. H. J.; Boom, T. J. van den, and Verbruggen, H. B.: "Optimization Algorithms for Bilinear Model-Based Predictive Control Problems". In: *AIChE Journal* 50.7 (2004), pp. 1453–1461.

[12] Boyd, Stephen and Vandenberghe, Lieven: *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2009.

[13] Boyd, Stephen; Parikh, Neal; Chu, Eric; Peleato, Borja, and Eckstein, Jonathan: "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers". In: *Foundations and Trends in Machine Learning* 3.1 (Jan. 2011), pp. 1–122.

[14] Bronshtein, Ilja N.; Semendjajew, Konstantin A.; Musiol, Gerhard, and Mühlig, Heiner: *Handbook of Mathematics*. Springer-Verlag Berlin Heidelberg, 2007.

[15] Candès, Emmanuel J.; Wakin, Michael B., and Boyd, Stephen P.: "Enhancing Sparsity by Reweighted l1 Minimization". In: *Journal of Fourier Analysis and Applications* 14.5 (2008), pp. 877–905.

[16] Christofides, Panagiotis D.; Scattolini, Riccardo; Pena, David Munoz de la, and Liu, Jinfeng: "Distributed Model Predictive Control: A Tutorial Review and Future Research Directions". In: *Computers and Chemical Engineering* 51 (2013), pp. 21 –41.

[17] Cichocki, Andrzej: "Era of Big Data Processing: A New Approach via Tensor Networks and Tensor Decompositions". In: *CoRR* abs/1403.2048 (2014). arXiv: 1403.2048.

[18] Cichocki, Andrzej; Zdunek, Rafal; Phan, Anh Huy, and Amari, Shun-Ichi: *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-Way Data Analysis and Blind Source Separation*. Wiley, Chichester, 2009.

[19] Cichocki, Andrzej; Mandic, Danilo P.; Phan, Anh Huy; Caiafa, Cesar F.; Zhou, G.; Zhao, Qibin, and Lathauwer, Lieven De: "Tensor Decompositions for Signal Processing Applications From Two-way to Multiway Component Analysis". In: *CoRR* abs/1403.4462 (2014).

[20] Cichocki, Andrzej; Lee, Namgil; Oseledets, Ivan; Phan, Anh-Huy; Zhao, Qibin, and Mandic, Danilo P.: "Tensor Networks for Dimensionality Reduction and Large-scale Optimization: Part 1 Low-Rank Tensor Decompositions". In: *Foundations and Trends in Machine Learning* 9.4-5 (2016), pp. 249–429.

[21] Cichocki, Andrzej; Phan, Anh Huy; Zhao, Qibin; Lee, Namgil; Oseledets, Ivan V.; Sugiyama, Masashi, and Mandic, Danilo P.: "Tensor Networks for Dimensionality Reduction and Large-scale Optimization: Part 2 Applications and Future Perspectives". In: *Foundations and Trends in Machine Learning* 9 (2017), pp. 431–673.

[22] Domingues, Pedro; Carreira, Paulo; Vieira, Renato, and Kastner, Wolfgang: "Building Automation Systems: Concepts and Technology Review". In: *Computer Standards & Interfaces* 45 (2016), pp. 1 –12.

[23] Dorf, Richard C. and Bishop, Robert H.: *Modern Control Systems*. 12th ed. Pearson Prentice Hall, 2011.

[24] Dück, Natalia: "Multilineare Zustandsraummodelle und Ansätze zur Parameteridentifikation durch Tensordekomposition". Diploma Thesis. Hamburg University of Technology, 2011.

[25] Eichler, Annika; Darivianakis, Georgios, and Lygeros, John: "Humans in the Loop: A Stochastic Predictive Approach to Building Energy Management in the Presence of Unpredictable Users". In: *IFAC-PapersOnLine* 50.1 (2017). 20th IFAC World Congress, pp. 14471 –14476.

[26] Ekman, Mats: "Modeling and Control of Bilinear Systems: Application to the Activated Sludge Process". PhD thesis. Uppsala University, 2005.

[27] Ellis, Matthew; Durand, Helen, and Christofides, Panagiotis D.: "A Tutorial Review of Economic Model Predictive Control Methods". In: *Journal of Process Control* 24.8 (2014), pp. 1156 –1178.

[28] Fardad, Makan; Lin, Fu, and Jovanovic, Mihailo R.: "Sparsity-Promoting Optimal Control for a Class of Distributed Systems". In: *American Control Conference*. 2011, pp. 2050–2055.

[29] Fernandez, N.; Katipamula, S.; Wang, W.; Xie, Y.; Zhao, M., and Corbin, C.: *Impacts of Commercial Building Controls on Energy Savings and Peak Load Reduction*. Pacific Northwest National Laboratory, 2017.

[30] Forbes, Michael G.; Patwardhan, Rohit S.; Hamadah, Hamza, and Gopaluni, R. Bhushan: "Model Predictive Control in Industry: Challenges and Opportunities". In: *IFAC-PapersOnLine* 48.8 (2015). 9th IFAC Symposium on Advanced Control of Chemical Processes ADCHEM 2015, pp. 531 –538.

[31] Grasedyck, Lars; Kressner, Daniel, and Tobler, Christine: "A Literature Survey of Low-Rank Tensor Approximation Techniques". In: *GAMM-Mitteilungen* 36 (2013), pp. 53–78.

[32] Hackbusch, Wolfgang: *Tensor Spaces and Numerical Tensor Calculus*. Vol. 42. Springer Series in Computational Mathematics. Springer-Verlag Berlin Heidelberg, 2012.

[33] Harish, V.S.K.V. and Kumar, Arun: "A Review on Modeling and Simulation of Building Energy Systems". In: *Renewable and Sustainable Energy Reviews* 56 (2016), pp. 1272 –1292.

[34] Henson, Michael A. and Seborg, Dale E.: *Nonlinear Process Control*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1997.

[35] Henze, Gregor P.: "Model Predictive Control for Buildings: A Quantum Leap?" In: *Journal of Building Performance Simulation* 6.3 (2013), pp. 157–158.

[36] Hitchcock, Frank L.: "The Expression of a Tensor or a Polyadic as a Sum of Products". In: *Journal of Mathematics and Physics* 6.1-4 (1927), pp. 164–189.

[37] Huyck, Bart; Logist, Filip; Brabanter, Jos De; Impe, Jan Van, and Moor, Bart De: "Constrained Model Predictive Control on a Programmable Automation System Exploiting Code Generation: Practical Considerations". In: *IFAC Proceedings Volumes* 44.1 (2011), pp. 12207 –12212.

[38] Ioli, Daniele; Deori, Luca; Falsone, Alessandro, and Prandini, Maria: "A Two-Layer Decentralized Approach to the Optimal Energy Management of a Building District with a Shared Thermal Storage". In: *IFAC-PapersOnLine* 50.1 (2017). 20th IFAC World Congress, pp. 8844 –8849.

[39] Isidori, Alberto: *Nonlinear Control Systems*. 3rd. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1995.

[40] Kang, Chang-Soon; Park, Jong-Il; Park, Mignon, and Baek, Jaeho: "Novel Modeling and Control Strategies for a HVAC System Including Carbon Dioxide Control". In: *Energies* 7 (June 2014), pp. 3599–3617.

[41] Kastner, Wolfgang; Neugschwandtner, Georg; Soucek, Stefan, and Newman, H. Michael: "Communication Systems for Building Automation and Control". In: *Proceedings of the IEEE* 93.6 (2005), pp. 1178–1203.

[42] Kelman, Anthony and Borrelli, Francesco: "Bilinear Model Predictive Control of a HVAC System Using Sequential Quadratic Programming". In: *18th IFAC World Congress*. 2011, pp. 9869–9874.

[43] Khalil, K. H.: *Nonlinear Systems*. 2nd ed. Prentice-Hall, Inc., 1996.

[44] Kolda, Tamara G. and Bader, Brett W.: "Tensor Decompositions and Applications". In: *SIAM Review* 51.3 (2009), pp. 455–500.

[45] Kressner, Daniel and Tobler, Christine: "Algorithm 941: Htucker-A Matlab Toolbox for Tensors in Hierarchical Tucker Format". In: *ACM Trans. Math. Softw.* 40.3 (2014), 22:1–22:22.

[46] Kruppa, Kai: "Comparison of Tensor Decomposition Methods for Simulation of Multilinear Time-Invariant Systems with the MTI Toolbox". In: *IFAC-PapersOnLine* 50.1 (2017). 20th IFAC World Congress, pp. 5610 –5615.

[47] Kruppa, Kai and Lichtenberg, Gerwald: "Comparison of CP Tensors, Tucker Tensors and Tensor Trains for Representation of Right Hand Sides of Ordinary Differential Equations". In: *Workshop on Tensor Decomposition and Applications, Leuven*. 2016.

[48] Kruppa, Kai and Lichtenberg, Gerwald: "Decentralized State Feedback Design for Multilinear Time-Invariant Systems". In: *IFAC-PapersOnLine* 50.1 (2017). 20th IFAC World Congress, pp. 5616 –5621.

[49] Kruppa, Kai and Lichtenberg, Gerwald: "Feedback Linearization of Multilinear Time-Invariant Systems Using Tensor Decomposition Methods". In: *8th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH)* (2018). Accepted.

[50] Kruppa, Kai; Pangalos, Georg, and Lichtenberg, Gerwald: "Multilinear Approximation of Nonlinear State Space Models". In: *19th IFAC World Congress, Cape Town*. IFAC. Cape Town, South Africa, 2014, pp. 9474–9479.

[51] Kruppa, Kai; Pfeiffer, Sven; Lichtenberg, Gerwald; Brinker, Frank; Decking, Winfried; Flöttmann, Klaus; Krebs, Olaf; Schlarb, Holger, and Schreiber, Siegfried: "High Precision Temperature Control for Injector Components of a Free-Electron Laser". In: *Simulation and Modeling Methodologies, Technologies and Applications*. Ed. by al., M. Obaidat et. Vol. 442. Advances in Intelligent Systems and Computing. Springer, 2016, pp. 115–135.

[52] Kruppa, Kai; Müller, Thorsten; Lautenschlager, Björn; Lichtenberg, Gerwald, and Rehault, Nicolas: "State Space Models as a Common Tool for Control Design, Optimization and Fault Detection in Building Systems". In: *Central European Symposium on Building Physics (Bausim), Dresden* (2017).

[53] Kvasnica, Michal: "Implicit vs. explicit MPC - Similarities, differences, and a path owards a unified method". In: *2016 European Control Conference (ECC)*. 2016, pp. 603–603.

[54] Lamoudi, Mohamed Yacine; Alamir, Mazen, and Béguery, Patrick: "Model Predictive Control for Energy Management in Buildings Part 2: Distributed Model Predictive Control". In: *IFAC Proceedings Volumes* 45.17 (2012). 4th IFAC Conference on Nonlinear Model Predictive Control, pp. 226 –231.

[55] Lathauwer, Lieven De; Moor, Bart De, and Vandewalle, Joos: "A Multilinear Singular Value Decomposition". In: *SIAM Journal on Matrix Analysis and Applications* 21.4 (2000), pp. 1253–1278.

[56] Lautenschlager, Björn; Kruppa, Kai, and Lichtenberg, Gerwald: "Convexity Properties of the Model Predictive Control Problem for Subclasses of Multilinear Time-Invariant Systems". In: *IFAC-PapersOnLine* 48.23 (2015). 5th IFAC Conference on Nonlinear Model Predictive Control NMPC 2015, pp. 148 –153.

[57] Lawrynczuk, Maciej: "Model Predictive Control with On-Line Optimal Linearisation". In: *IEEE International Symposium on Intelligent Control, ISIC* (Nov. 2014), pp. 2177–2182.

[58] Lawrynczuk, Maciej: "Nonlinear Predictive Control of a Boiler-Turbine Unit: A State-Space Approach with Successive On-Line Model Linearisation and Quadratic Optimisation". In: *ISA Transactions* 67 (2017), pp. 476 –495.

[59] Lee, Namgil and Cichocki, Andrzej: "Fundamental Tensor Operations for Large-Scale Data Analysis in Tensor Train Formats". In: *CoRR* abs/1405.7786 (2016).

[60] Lee, Namgil and Cichocki, Andrzej: "Fundamental Tensor Operations for Large-Scale Data Analysis using Tensor Network Formats". In: *Multidimensional Systems and Signal Processing* 29.3 (2018), pp. 921–960.

[61] Lewis, Frank L.; Vrabie, Draguna L., and Syrmos, Vassilis L.: *Optimal Control: Third Edition*. John Wiley and Sons, 2012.

[62] Li, Xiwang and Wen, Jin: "Review of building energy modeling for control and operation". In: *Renewable and Sustainable Energy Reviews* 37 (2014), pp. 517 – 537.

[63] Lichtenberg, Gerwald: "Hybrid Tensor Systems". Habilitation. Hamburg University of Technology, 2011.

[64] Lichtenberg, Gerwald: "Tensor Representation of Boolean Functions and Zhegalkin Polynomials". In: *International Workshop on Tensor Decompositions, Bari*. Bari, Italy, 2010.

[65] Lichtenberg, Gerwald: *Theorie und Anwendung der Qualitativen Modellierung Zeitdiskreter Dynamischer Systeme durch Nichtdeterministische Automaten*. Vol. 8. Fortschrittberichte VDI. VDI Verlag, Düsseldorf, 1998.

[66] Lin, Fu; Fardad, Makan, and Jovanovic, Mihailo R.: "Design of Optimal Sparse Feedback Gains via the Alternating Direction Method of Multipliers". In: *IEEE Transactions on Automatic Control* 58.9 (2013), pp. 2426–2431.

[67] Lin, Fu; Fardad, Makan, and Jovanovic, Mihailo R.: "Sparse Feedback Synthesis via the Alternating Direction Method of Multipliers". In: *American Control Conference (ACC)* (2012), pp. 4765–4770.

[68] Ljung, Lennart: *System Identification: Theory for the User*. Prentice-Hall Inc., 1987.

[69] Löhr, Yannik and Mönnigmann, Martin: "Domestic Heat Generation and Distribution with Tme-Variant Receding Horizon Control". In: *IFAC-PapersOnLine* 50.1 (2017). 20th IFAC World Congress, pp. 4191 –4196.

[70] Lunze, Jan: *Control Theory of Digitally Networked Dynamic Systems*. Springer International Publishing, 2014.

[71] Ma, Yudong; Anderson, Garrett, and Borrelli, Francesco: "A Distributed Predictive Control Approach to Building Temperature Regulation". In: *Proceedings of the 2011 American Control Conference*. 2011, pp. 2089–2094.

[72] Maasoumy, Mehdi; Pinto, Alessandro, and Sangiovanni-Vincentelli, Alberto: "Model-Based Hierarchical Optimal Control Design for HVAC Systems". In: *ASME 2011 Dynamic Systems and Control Conference*. 2011.

[73] Maciejowski, J.: *Predictive Control with Constraints*. Pearson Education Limited, 2002.

[74] Mattingley, Jason B.; Wang, Yang, and Boyd, Steven: "Receding Horizon Control". In: *IEEE Control Systems* 31 (2011), pp. 52–65.

[75] Morari, Manfred and Lee, Jay H.: "Model Predictive Control: Past, Present and Future". In: *Computers & Chemical Engineering* 23.4-5 (1999), pp. 667 –682.

[76] Morsi, Abdelrahman; Abbas, Hossam S., and Mohamed, Abdelfatah M.: "Model Predictive Control of a Wind Turbine Based on Linear Parameter-Varying Models". In: *IEEE Conference on Control Applications (CCA)*. 2015, pp. 318–323.

[77] Müller-Eping, Thorsten; Lichtenberg, Gerwald, and Vogelmann, Vivien: "Fault Detection Algorithms based on Decomposed Tensor Representations for Qualitative Models". In: *IFAC-PapersOnLine* 50.1 (2017). 20th IFAC World Congress, pp. 5622 –5629.

[78] Müller, Bernhard and Deutscher, Joachim: "Approximate Input-Output Linearization Using L2-Optimal Bilinearization". In: *Proceedings of the 44th IEEE Conference on Decision and Control and the European Control Conference*. 2005.

[79] Müller, Thorsten; Kruppa, Kai; Lichtenberg, Gerwald, and Rehault, Nicolas: "Fault Detection with Qualitative Models reduced by Tensor Decomposition methods". In: *IFAC-PapersOnLine* 48.21 (2015). 9th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes SAFEPROCESS, pp. 416 –421.

[80] Negenborn, Rudy R. and Maestre, Jose Maria: "Distributed Model Predictive Control: An Overview and Roadmap of Future Research Opportunities". In: *IEEE Control Systems* 34.4 (2014), pp. 87–97.

[81] OBSERVE: *Optimierung und Betriebsführung komplexer Gebäudeenergieversorgungsanlagen, Abschlussbericht*. Tech. rep. Fraunhofer ISE, HAW Hamburg, Kieback& Peter GmbH, Plenum Ingenieursgesellschaft, and IngSoft GmbH, 2018.

[82] Oseledets, I. V.: "Constructive Representation of Functions in Low-Rank Tensor Formats". In: *Constructive Approximation* 37.1 (2013), pp. 1–18.

[83] Oseledets, Ivan: "Tensor-Train Decomposition". In: *SIAM Journal of Scientific Computing* 33.5 (2011), pp. 2295–2317.

[84] Oseledets, Ivan; Dolgov, Sergey; Kazeev, Vladimir; Lebedeva, Olga, and Mach, Thomas: *TT-Toolbox 2.2*. Available online. 2012. URL: `https://github.com/oseledets/TT-Toolbox`.

[85] Pangalos, Georg: "Model-Based Controller Design Methods for Heating Systems". Dissertation. Hamburg: TU Hamburg, 2016.

[86] Pangalos, Georg; Eichler, Annika, and Lichtenberg, Gerwald: *Hybrid Multilinear Modeling and Applications*. Ed. by Obaidat, S. Mohammad; Koziel, Slawomir; Kacprzyk, Janusz; Leifsson, Leifur, and Ören, Tuncer. Springer International Publishing, 2015, pp. 71–85.

[87] Pangalos, Georg; Eichler, Annika, and Lichtenberg, Gerwald: "Tensor Systems: Multilinear Modeling and Applications". In: *3rd Int. Conference on Simulation and Modeling Methodologies, Technologies and Applications*. Reykjavik, 2013.

[88] Patel, Nishith R.; Risbeck, Michael J.; Rawlings, James B.; Wenzel, Michael J., and Turney, Robert D.: "Distributed Economic Model Predictive Control for Large-Scale Building Temperature Regulation". In: *American Control Conference (ACC)*. 2016, pp. 895–900.

[89] Polyak, Boris; Khlebnikov, Mikhail, and Shcherbakov, Pavel S.: "Sparse Feedback in Linear Control Systems". In: *Automation and Remote Control* 75.12 (Dec. 2014), pp. 2099–2111.

[90] Pop, Cristina Ioana and Dulf, Eva Henrietta: *Robust Feedback Linearization Control for Reference Tracking and Disturbance Rejection in Nonlinear Systems, Recent Advances in Robust Control - Novel Approaches and Design Methods*. Ed. by Müller, Andreas. InTech, 2011.

[91] Rehrl, Jakob and Horn, Martin: "Temperature Control for HVAC Systems Based on Exact Linearization and Model Predictive Control". In: *IEEE International Conference on Control Applications (CCA)*. 2011, pp. 1119–1124.

[92] Röbenack, Klaus: "Automatic Differentiation and Nonlinear Controller Design by Exact Linearization". In: *Future Generation Computer Systems* 21 (2005), pp. 1372–1379.

[93] Rostampour, Vahab and Keviczky, Tamas: "Energy Management for Building Climate Comfort in Uncertain Smart Thermal Grids with Aquifer Thermal Energy Storage". In: *IFAC-PapersOnLine* 50.1 (2017). 20th IFAC World Congress, pp. 13156 –13163.

[94] Sastry, Shankar: *Nonlinear Systems: Analysis, Stability, and Control*. Interdisciplinary Applied Mathematics. Springer New York, 1999.

[95] Scattolini, Riccardo: "Architectures for Distributed and Hierarchical Model Predictive Control - A Review". In: *Journal of Process Control* 19.5 (2009), pp. 723 –731.

[96] Schmid, Claudia and Biegler, Lorenz T.: "Quadratic Programming Methods for Tailored Reduced Hessian SQP". In: *Computers and Chemical Engineering* (1993).

[97] Schuler, Simone; Zhou, Wenliang; Münz, Ulrich, and Allgöwer, Frank: "Controller Structure Design for Decentralized Control of Coupled Higher Order Subsystems". In: *IFAC Proceedings Volumes* 43.19 (2010), pp. 269 –274.

[98] Semsar-Kazerooni, Elham; Yazdanpanah, Mohammad Javad, and Lucas, Caro: "Nonlinear Control and Disturbance Decoupling of HVAC Systems Using Feedback Linearization and Backstepping With Load Estimation". In: *IEEE Transactions on Control Systems Technology* 16.5 (2008), pp. 918–929.

[99] Serway, Raymond A. and Jewett, John W.: *Physics for Scientists and Engineers*. Thomson Brooks/Cole, 2004.

[100] Shaikh, Pervez Hameed; Nor, Nursyarizal Bin Mohd; Nallagownden, Perumal; Elamvazuthi, Irraivan, and Ibrahim, Taib: "A Review on Optimized Control Systems for Building Energy and Comfort Management of Smart Sustainable Buildings". In: *Renewable and Sustainable Energy Reviews* 34 (2014), pp. 409 –429.

[101] Shaikh, Pervez Hameed; Nor, Nursyarizal Bin Mohd; Nallagownden, Perumal; Elamvazuthi, Irraivan, and Ibrahim, Taib: "A Review on Optimized Control Systems for Building Energy and Comfort Management of Smart Sustainable Buildings". In: *Renewable and Sustainable Energy Reviews* 34 (2014), pp. 409 – 429.

[102] Siljak, Dragoslav D.: *Decentralized Control of Complex Systems*. Boston : Academic Press, 1991.

[103] Sofroniou, Mark: "Symbolic Derivation of Runge-Kutta Methods". In: *Journal of Symbolic Computation* 18.3 (1994), pp. 265 –296.

[104] Sorber, Laurent; Barel, Marc Van, and Lathauwer, Lieven De: "Structured Data Fusion". In: *IEEE Journal of Selected Topics in Signal Processing* 9.4 (2015), pp. 586–600.

[105] Tanaskovic, Marko; Sturzenegger, David; Smith, Roy, and Morari, Manfred: "Robust Adaptive Model Predictive Building Climate Control". In: *IFAC-PapersOnLine* 50.1 (2017). 20th IFAC World Congress, pp. 1871 –1876.

[106] Tashtoush, Bourhan; Molhim, Mohammad, and Al-Rousan, Mohammad: "Dynamic Model of an HVAC System for Control Analysis". In: *Energy* 30.10 (2005), pp. 1729 –1745.

[107] Thosar, Archana; Patra, Amit, and Bhattacharyya, Souvik: "Feedback Linearization Based Control of a Variable Air Volume Air Conditioning System for Cooling Applications". In: *ISA Transactions* 47.3 (2008), pp. 339 –349.

[108] Van Overschee, Peter and De Moor, Bart: *Subspace Identification for Linear Systems*. Boston, MA: Springer US, 1996.

[109] Vervliet, Nico; Debals, Otto; Sorber, Laurent; Van Barel, Marc, and De Lathauwer, Lieven: *Tensorlab 3.0*. Available online. 2016. URL: `http://www.tensorlab.net`.

[110] Walker, Shalika S.W.; Lombardi, Warody; Lesecq, Suzanne, and Roshany-Yamchi, Samira: "Application of Distributed Model Predictive Approaches to Temperature and CO2 Concentration Control in Buildings". In: *IFAC-PapersOnLine* 50.1 (2017). 20th IFAC World Congress, pp. 2589 –2594.

[111] Wang, Yang; Kuckelkorn, Jens, and Liu, Yu: "A State of Art Review on Methodologies for Control Strategies in Low Energy Buildings in the Period from 2006 to 2016". In: *Energy and Buildings* 147 (2017), pp. 27 –40.

[112] Zheng, Yi and Li, Shaoyuan Y.: "Distributed Predictive Control for Building Temperature Regulation with Impact-Region Optimization". In: *IFAC Proceedings Volumes* 47.3 (2014). 19th IFAC World Congress, pp. 12074 –12079.

A Proofs

This chapter summarizes the proofs of the methods for tensors and MTI systems introduced in Chapters 2 to 4.

A.1 Outer product in CP form

The proof of the computation of the outer product of tensors in CP decomposed format as stated in Proposition 2.2 is given in the following.

Proof A.1 *Inserting the elementwise descriptions (2.11) of* X *and* Y

$$x(i_1,\dots,i_n) = \sum_{k=1}^{r_{cp}(\mathsf{X})} \lambda_X(k) u_1(i_1,k)\cdots u_n(i_n,k), \tag{A.1}$$

$$y(j_1,\dots,j_m) = \sum_{l=1}^{r_{cp}(\mathsf{Y})} \lambda_Y(l) v_1(j_1,l)\cdots v_m(j_m,l). \tag{A.2}$$

into the outer product (2.3), leads to

$$\begin{aligned}
z(i_1,\dots,i_n,j_1,\dots,j_m) &= x(i_1,\dots,i_n)y(j_1,\dots,j_m)\\
&= \sum_{k=1}^{r_{cp}(\mathsf{X})} \lambda_X(k) u_1(i_1,k)\cdots u_n(i_n,k) \cdot \sum_{l=1}^{r_{cp}(\mathsf{Y})} \lambda_Y(l) v_1(j_1,l)\cdots v_m(j_m,l)\\
&= \sum_{k=1}^{r_{cp}(\mathsf{X})}\sum_{l=1}^{r_{cp}(\mathsf{Y})} \lambda_X(k)\lambda_Y(l) u_1(i_1,k)\cdots u_n(i_n,k) v_1(j_1,l)\cdots v_m(j_m,l)\\
&= \lambda_X(1)\lambda_Y(1) u_1(i_1,1)\cdots u_n(i_n,1) v_1(j_1,1)\cdots v_m(j_m,1)\\
&+ \lambda_X(1)\lambda_Y(2) u_1(i_1,1)\cdots u_n(i_n,1) v_1(j_1,2)\cdots v_m(j_m,2)\\
&+ \cdots + \lambda_X(r_{cp}(\mathsf{X}))\lambda_Y(r_{cp}(\mathsf{Y})) u_1(i_1,r_{cp}(\mathsf{X}))\cdots v_m(j_m,r_{cp}(\mathsf{Y})).
\end{aligned}$$

Comparing this representation of the outer product with the elementwise description

$$z(i_1,\dots,j_m) = \sum_{q=1}^{r_{cp}(\mathsf{X})r_{cp}(\mathsf{Y})} \lambda_Z(q) w_1(i_1,q)\cdots w_{n+m}(j_m,q),$$

of the resulting tensor in CP decomposition, gives

$$\begin{aligned}
\lambda_Z(\overline{lk}) &= \lambda_X(k)\lambda_Y(l),\\
w_r(:,\overline{lk}) &= u_r(:,k),\ r=1,\dots,n,\\
w_{n+r}(:,\overline{lk}) &= v_r(:,l),\ r=1,\dots,m,
\end{aligned}$$

with $k = 1, \ldots, r_{cp}(\mathsf{X})$ and $l = 1, \ldots, r_{cp}(\mathsf{Y})$ and leads to the factors in (2.28) to (2.30). The multi-indices notation of Definition 2.4 is used here to merge the indices k and l. □

A.2 Contracted product in CP form

In this section the computation of the contracted product of tensors in CP representation of Proposition 2.3 is proven.

Proof A.2 *The operands X and Y are described elementwise by*

$$x(i_1, \ldots, i_P, j_1, \ldots, j_N) = \sum_{l=1}^{r_{cp}(\mathsf{X})} \lambda_X(l) u_1(i_1, l) \cdots u_P(i_P, l) u_{P+1}(j_1, l) \cdots u_{P+N}(j_N, l), \tag{A.3}$$

$$y(i_1, \ldots, i_P, k_1, \ldots, k_M) = \sum_{q=1}^{r_{cp}(\mathsf{Y})} \lambda_Y(q) v_1(i_1, q) \cdots v_P(i_P, q) v_{P+1}(k_1, q) \cdots v_{P+M}(k_M, q). \tag{A.4}$$

Inserting the elements of X and Y into the definition of the contracted product (2.4) and rearranging gives

$$\begin{aligned}
&z(j_1, \ldots, j_N, k_1, \ldots, k_M) \\
&= \sum_{i_1=1}^{I_1} \cdots \sum_{i_P=1}^{I_P} \sum_{l=1}^{r_{cp}(\mathsf{X})} \lambda_X(l) u_1(i_1, l) \cdots u_P(i_P, l) u_{P+1}(j_1, l) \cdots u_{P+N}(j_N, l) \\
&\cdot \sum_{q=1}^{r_{cp}(\mathsf{Y})} \lambda_Y(q) v_1(i_1, q) \cdots v_P(i_P, q) v_{P+1}(k_1, q) \cdots v_{P+M}(k_M, q) \\
&= \sum_{l=1}^{r_{cp}(\mathsf{X})} \sum_{q=1}^{r_{cp}(\mathsf{Y})} \underbrace{\lambda_X(l) \lambda_Y(q) \sum_{i_1=1}^{I_1} \cdots \sum_{i_P=1}^{I_P} u_1(i_1, l) v_1(i_1, q) \cdots u_P(i_P, l) v_P(i_P, q)}_{\text{Part that belongs to } \lambda_Z.} \\
&\cdot \underbrace{u_{P+1}(j_1, l) \cdots u_{P+N}(j_N, l) v_{P+1}(k_1, l) \cdots v_{P+M}(k_M, l)}_{\text{Part that belongs to the factor matrices } \mathbf{W}_i}.
\end{aligned} \tag{A.5}$$

The terms are sorted, such that the first part contains the terms that do not depend on the indices $j_i,\ i = 1, \ldots, N$ and $k_i,\ i = 1, \ldots, P$ of the resulting tensor Z. Because of that, this part of the sum belongs to the weighting vector as indicated in (A.5). Comparing (A.5) with the elementwise notation of Z

$$\begin{aligned}
&z(j_1, \ldots, j_N, k_1, \ldots, k_M) = \\
&\sum_{n=1}^{r_{cp}(\mathsf{X}) r_{cp}(\mathsf{Y})} \lambda_Z(n) w_1(j_1, n) \cdots w_N(j_N, n) w_{N+1}(k_1, n) \cdots w_{N+M}(k_M, n),
\end{aligned} \tag{A.6}$$

and merging the indices l and q of the summations into one index n by a multi-indices notation $\overline{lq}$, the weighting vector is given elementwise

$$\lambda_Z(\overline{lq}) = \lambda_X(l) \lambda_Y(q) \sum_{i_1=1}^{I_1} \cdots \sum_{i_P=1}^{I_P} u_1(i_1, l) v_1(i_1, q) \cdots u_P(i_P, l) v_P(i_P, q).$$

where $l = 1, \dots, r_{cp}(\mathsf{X})$ *and* $q = 1, \dots, r_{cp}(\mathsf{Y})$*. This describes the elements of (2.33). All terms depending on the indices of* Z *can be found in the second part of the sum in (A.5). Thus, from these elements the factor matrices* $\mathbf{W}_i$*,* $i = 1, \dots, N+M$ *of* Z *are constructed by comparison to (A.6)*

$$w_i(j_i, \overline{lq}) = u_{P+i}(j_i, l),\ i = 1, \dots, N,$$
$$w_{N+i}(k_i, \overline{lq}) = v_{P+i}(k_i, q),\ i = 1, \dots, M,$$

with $l = 1, \dots, r_{cp}(\mathsf{X})$ *and* $q = 1, \dots, r_{cp}(\mathsf{Y})$*. This gives exactly the elements of (2.34) and (2.35), such that Proposition 2.3 is proven.* □

A.3 Concatenation of tensors in CP form

The concatenation of tensors in CP representation was introduced in Proposition 2.2.3 and is proven here.

Proof A.3 *The elements of the resulting tensor* Z *are computed by its CP factors*

$$z(i_1, \dots, i_k, \dots, i_N) = \sum_{j=1}^{r_{cp}(\mathsf{X})r_{cp}(\mathsf{Y})} \lambda_Z(j) w_1(i_1, j) \cdots w_k(i_k, j) \cdots w_N(i_N, j). \tag{A.7}$$

With (2.6) and (2.8) the elements of the factor matrices (2.36) result in

$$\mathbf{w}_i(:, j) = \begin{cases} \mathbf{u}_i(:, j) & \text{, for } 1 \le j \le r_{cp}(\mathsf{X}), \\ \mathbf{v}_i(:, j - r_{cp}(\mathsf{X})) & \text{, for } r_{cp}(\mathsf{X}) < j \le r_{cp}(\mathsf{X}) + r_{cp}(\mathsf{Y}), \end{cases}$$

and the k^{th} *factor matrix (2.37) is given by*

$$w_k(i_k, j) = \begin{cases} u_k(i_k, j) & \text{, for } 1 \le i_k \le I_k,\ 1 \le j \le r_{cp}(\mathsf{X}) \\ v_k(i_k - I_k, j - r_X) & \text{, for } I_k < i_k \le I_k + J_k, r_{cp}(\mathsf{X}) < j \le r_{cp}(\mathsf{X}) + r_{cp}(\mathsf{Y}), \\ 0 & \text{, else,} \end{cases} \tag{A.8}$$

The weighting vector $\boldsymbol{\lambda}_Z$ *is constructed from the weighting vector of* X *and* Y*, such that*

$$\lambda_Z(j) = \begin{cases} \lambda_X(j) & \text{, for } 1 \le j \le r_{cp}(\mathsf{X}), \\ \lambda_Y(j - r_{cp}(\mathsf{X})) & \text{, for } r_{cp}(\mathsf{X}) < j \le r_{cp}(\mathsf{X}) + r_{cp}(\mathsf{Y}). \end{cases} \tag{A.9}$$

Inserting these elementwise descriptions (A.8) and (A.9) in (A.7) and eliminating the zeros terms leads to

$$z(i_1, \dots, i_k, \dots, i_N) = \begin{cases} x(i_1, \dots, i_k, \dots, i_N) & \text{, for } 1 \le i_k \le I_k, \\ y(i_1, \dots, i_k - I_k, \dots, i_N) & \text{, for } I_k \le i_k \le I_k + J_k. \end{cases}$$

This shows, that Z *contains the tensors* X *and* Y *as stated by the concatenation* $\mathsf{Z} = \mathsf{X} \boxplus_k \mathsf{Y}$*.* □

A.4 Multiplication in tensor form

This section presents the proof of the multiplication in tensor form of Theorem 2.1.

Proof A.4 *Considering (2.46) the monomial tensor* $\mathsf{M}_p^{N_1+N_2}(\mathbf{x})$ *of the result is decomposed by*

$$\left\langle \mathsf{H}_1 \circ \mathsf{H}_2 \,\middle|\, \mathsf{M}_p^{N_1+N_2}(\mathbf{x}) \right\rangle = \left\langle \mathsf{H}_1 \circ \mathsf{H}_2 \,\middle|\, \mathsf{M}_p^{N_1}(\mathbf{x}) \circ \mathsf{M}_p^{N_2}(\mathbf{x}) \right\rangle.$$

Using the elementwise representations (2.3) and (2.4) of outer and contracted product and rearranging the terms belonging to the first $h_1(\mathbf{x})$ *and the second* $h_2(\mathbf{x})$ *polynomial respectively gives*

$$\begin{aligned}
h(\mathbf{x}) &= \left\langle \mathsf{H}_1 \circ \mathsf{H}_2 \,\middle|\, \mathsf{M}_p^{N_1}(\mathbf{x}) \circ \mathsf{M}_p^{N_2}(\mathbf{x}) \right\rangle \\
&= \sum_{i_1=1}^{2} \cdots \sum_{i_{nN_1}=1}^{2} \sum_{j_1=1}^{2} \cdots \sum_{j_{nN_2}=1}^{2} h_1(i_1,\ldots,i_{nN_1}) h_2(j_1,\ldots,j_{nN_2}) \\
&\cdot m_p^{N_1}(\mathbf{x})(i_1,\ldots,i_{nN_1}) m_p^{N_2}(\mathbf{x})(j_1,\ldots,j_{nN_2}) \\
&= \sum_{i_1=1}^{2} \cdots \sum_{i_{nN_1}=1}^{2} h_1(i_1,\ldots,i_{nN_1}) m_p^{N_1}(\mathbf{x})(i_1,\ldots,i_{nN_1}) \\
&\cdot \sum_{j_1=1}^{2} \cdots \sum_{j_{nN_2}=1}^{2} h_2(j_1,\ldots,j_{nN_2}) m_p^{N_2}(\mathbf{x})(j_1,\ldots,j_{nN_2}) \\
&= \left\langle \mathsf{H}_1 \,\middle|\, \mathsf{M}_p^{N_1}(\mathbf{x}) \right\rangle \cdot \left\langle \mathsf{H}_2 \,\middle|\, \mathsf{M}_p^{N_2}(\mathbf{x}) \right\rangle \\
&= h_1(\mathbf{x}) \cdot h_2(\mathbf{x}). \quad \square
\end{aligned}$$

This shows that the resulting function $h(\mathbf{x})$ *with parameter tensor* $\mathsf{H}_1 \circ \mathsf{H}_2$ *is equal to the multiplication of two polynomials* $h_1(\mathbf{x})$ *and* $h_2(\mathbf{x})$ *with parameter tensors* H_1 *and* H_2*, respectively.*

A.5 Differentiation in tensor form

The proof of the Theorem 2.2 of the differentiation of a polynomial in tensor form is given here.

Proof A.5 *All terms depending on the variables* x_i*, with* $i = 1,\ldots,n$*, of a polynomial in tensor form are stored inside the monomial tensor. The parameter tensor* H *contains constant elements only. Therefore, the partial derivative of the monomial tensor is investigated first. In the multilinear case the partial derivative with respect to one variable* x_j *can be found by using the product rule of differentiation for the outer product, [14].*
The partial derivative with respect to one variable x_j*, with* $j = 1,\ldots,n$ *of an outer product of two tensors* $\mathsf{A}(\mathbf{x}) \in \mathbb{R}^{I_1 \times \cdots \times I_l}$ *and* $\mathsf{B}(\mathbf{x}) \in \mathbb{R}^{K_1 \times \cdots \times K_m}$ *depending on* n *variables* $\mathbf{x} \in \mathbb{R}^n$ *is given by*

$$\frac{\partial}{\partial x_j}\left(\mathsf{A}(\mathbf{x}) \circ \mathsf{B}(\mathbf{x})\right) = \frac{\partial}{\partial x_j}\left(\mathsf{A}(\mathbf{x})\right) \circ \mathsf{B}(\mathbf{x}) + \mathsf{A}(\mathbf{x}) \circ \frac{\partial}{\partial x_j}\left(\mathsf{B}(\mathbf{x})\right). \tag{A.10}$$

This rule follows from the standard scalar definition of the product rule, by applying the standard product rule to the elementwise description of the outer product (2.3)

$$(\mathsf{A}(\mathbf{x}) \circ \mathsf{B}(\mathbf{x}))\,(i_1,\ldots,i_l,k_1,\ldots,k_m) = a(\mathbf{x})(i_1,\ldots,i_l)b(\mathbf{x})(k_1,\ldots,k_m).$$

The product rule of differentiation for scalars is applied to each element of the resulting tensor

$$\begin{aligned}\frac{\partial}{\partial x_j}\left((\mathsf{A}(\mathbf{x}) \circ \mathsf{B}(\mathbf{x}))\,(i_1,\ldots,k_m)\right) &= \frac{\partial}{\partial x_j}\left(a(\mathbf{x})(i_1,\ldots,i_l)b(\mathbf{x})(k_1,\ldots,k_m)\right)\\ &= \frac{\partial}{\partial x_j}\left(a(\mathbf{x})(i_1,\ldots,i_k)\right)b(\mathbf{x})(k_1,\ldots,k_m) + a(\mathbf{x})(i_1,\ldots,i_l)\frac{\partial}{\partial x}\left(b(\mathbf{x})(k_1,\ldots,k_m)\right).\end{aligned}$$

This leads to the full representation (A.10). This definition of the product rule of differentiation for tensors is applied now to the monomial tensor for multilinear functions (2.42) resulting in

$$\begin{aligned}\frac{\partial}{\partial x_j}\mathsf{M}(\mathbf{x}) = [\mathbf{w}_1,\ldots,\mathbf{w}_n] &= \frac{\partial}{\partial x_j}\left(\begin{pmatrix}1\\x_n\end{pmatrix}\circ\cdots\circ\begin{pmatrix}1\\x_1\end{pmatrix}\right)\\ &= \begin{pmatrix}1\\x_n\end{pmatrix}\circ\cdots\circ\begin{pmatrix}1\\x_{j+1}\end{pmatrix}\circ\begin{pmatrix}0\\1\end{pmatrix}\circ\begin{pmatrix}1\\x_{j-1}\end{pmatrix}\circ\cdots\circ\begin{pmatrix}1\\x_1\end{pmatrix}.\end{aligned} \tag{A.11}$$

The factor matrices $\mathbf{w}_i,\ i = 1,\ldots,n$ *of the derivative of the monomial tensor read*

$$\mathbf{w}_i = \begin{cases}\begin{pmatrix}0 & 1\end{pmatrix}^T & \text{, for } i = n-j+1,\\ \begin{pmatrix}1 & x_{n-i+1}\end{pmatrix}^T & \text{, else.}\end{cases}$$

The differentiation influences the dimension belonging to the differentiation variable x_j *of the monomial tensor only. Since the monomial tensor is a rank-*1 *tensor and has a CP structure, the mode-k tensor product of Proposition 2.1 can be applied here. As given in (2.15) one factor matrix is changed only by a multiplication with the considered matrix which is exactly the desired operation here. The factor matrices of the differentiation of the monomial tensor are computed by multiplying the* $(n-j+1)^{th}$ *factor matrix* $\begin{pmatrix}1 & x_j\end{pmatrix}^T$ *by a* 2×2 *matrix*

$$\mathbf{w}_{n-j+1} = \begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}0 & 0\\1 & 0\end{pmatrix}\begin{pmatrix}1\\x_j\end{pmatrix} = \mathbf{\Theta}^T\begin{pmatrix}1\\x_j\end{pmatrix},$$

as in the definition of the mode-k product for CP tensors. Thus, the differentiation can be expressed in terms of a mode-k product by

$$\frac{\partial}{\partial x_j}\mathsf{M}(\mathbf{x}) = \mathsf{M}(\mathbf{x}) \times_{n-j+1} \mathbf{\Theta}^T,$$

setting the factor matrix of $\mathsf{M}(\mathbf{x})$ *belonging to* x_j *to* $\begin{pmatrix}0 & 1\end{pmatrix}^T$.
Since the partial derivative can be found for the multilinear monomial tensor, the concept is extended to polynomials with higher monomial orders $N \geq 1$. *In contrast to the multilinear*

case, the variable x_j does not occur in one factor matrix only, but in N factor matrices. According to the product rule (A.10) the differentiation of $\mathsf{M}_p^N(\mathbf{x})$ leads to a sum of N rank-1 tensors

$$\frac{\partial}{\partial x_j}\mathsf{M}_p^N(\mathbf{x}) = \sum_{k=1}^{N}\left[\mathbf{w}_1^k, \ldots, \mathbf{w}_{nN}^k\right],$$

with factor matrices

$$\mathbf{w}_i^k = \begin{cases} \begin{pmatrix} 0 & 1 \end{pmatrix}^T & , \text{for } i = nk - j + 1 \\ \begin{pmatrix} 1 & x_{nk-i+1} \end{pmatrix}^T & , \text{else,} \end{cases},$$

for all $k = 1, \ldots, N$. As before, this change in the factor matrices can be expressed by mode-k product leading to the partial derivative of the monomial tensor

$$\frac{\partial}{\partial x_j}\mathsf{M}_p^N(\mathbf{x}) = \sum_{k=1}^{N}\mathsf{M}_p^N(\mathbf{x}) \times_{kn-j+1} \boldsymbol{\Theta}^T. \tag{A.12}$$

With the derivative of the monomial tensor (A.12) the derivative of the function is given by

$$\frac{\partial}{\partial x_j}\left\langle \mathsf{H} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle = \left\langle \mathsf{H} \,\middle|\, \sum_{k=1}^{N}\mathsf{M}_p^N(\mathbf{x}) \times_{kn-j+1} \boldsymbol{\Theta}^T \right\rangle \overset{!}{=} \left\langle \mathsf{H}_{x_j} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle. \tag{A.13}$$

To find the parameter tensor H_{x_j}, (A.13) can be written as

$$\left\langle \mathsf{H} \,\middle|\, \sum_{k=1}^{N}\mathsf{M}_p^N(\mathbf{x}) \times_{kn-j+1} \boldsymbol{\Theta}^T \right\rangle = \sum_{k=1}^{N}\left\langle \mathsf{H} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \times_{kn-j+1} \boldsymbol{\Theta}^T \right\rangle, \tag{A.14}$$

because of the linearity property of the inner product. The elements of the mode-k product read

$$\left(\mathsf{M}_p^N(\mathbf{x}) \times_l \boldsymbol{\Theta}^T\right)(i_1, \ldots, i_l, \ldots, i_{nN}) = m_p^N(i_1, \ldots, 1, \ldots, i_{nN})\Theta^T(i_l, 1),$$

because $\Theta^T(i_l, 2) = 0,\ i_l = 1, 2$. With that, the terms of the sum in (A.14) are written as

$$\left\langle \mathsf{H} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \times_l \boldsymbol{\Theta}^T \right\rangle = \sum_{i_1=1}^{2}\cdots\sum_{i_{nN}=1}^{2} h(i_1, \ldots, i_{nN}) m_p^N(i_1, \ldots, 1, \ldots, i_{nN})\Theta^T(i_l, 1). \tag{A.15}$$

The goal is to isolate the monomial tensor on the right side of the contracted product, such that

$$\left\langle \mathsf{H} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \times_l \boldsymbol{\Theta}^T \right\rangle \overset{!}{=} \left\langle \tilde{\mathsf{H}} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle = \sum_{i_1=1}^{2}\cdots\sum_{i_{nN}=1}^{2} \tilde{h}(i_1, \ldots, i_{nN}) m_p^N(i_1, \ldots, i_{nN}).$$

Therefore, comparing this description with (A.15), the elements of $\tilde{\mathsf{H}}$ are given by

$$\tilde{h}(i_1, \ldots, i_l, \ldots, i_{nN}) = \begin{cases} h(i_1, \ldots, 2, \ldots, i_{nN}) & , \text{for } i_l = 1, \\ 0 & , \text{for } i_l = 2, \end{cases}$$

since

$$\sum_{i_l=1}^{2} h(i_1,\ldots,i_l,\ldots,i_{nN})\Theta^T(l,1) = h(i_1,\ldots,2,\ldots,i_{nN}).$$

Using again the Definition 2.9 of the mode-k product, the parameter tensor $\tilde{\mathsf{H}}$ *can be directly computed by*

$$\tilde{\mathsf{H}} = \mathsf{H} \times_l \mathbf{\Theta},$$

leading to

$$\left\langle \mathsf{H} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \times_l \mathbf{\Theta}^T \right\rangle = \left\langle \mathsf{H} \times_l \mathbf{\Theta} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle.$$

Inserting this to (A.14) shows, that the partial derivative of a polynomial $h(\mathbf{x})$ *is computed by*

$$\begin{aligned}\frac{\partial}{\partial x_j}\left\langle \mathsf{H} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle &= \sum_{k=1}^{N}\left\langle \mathsf{H} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \times_{kn-j+1} \mathbf{\Theta}^T \right\rangle = \sum_{k=1}^{N}\left\langle \mathsf{H} \times_{kn-j+1} \mathbf{\Theta} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle \\ &= \left\langle \sum_{k=1}^{N} \mathsf{H} \times_{kn-j+1} \mathbf{\Theta} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle,\end{aligned}$$

such that the parameter tensor of the derivative reads

$$\mathsf{H}_{x_j} = \sum_{k=1}^{N} \mathsf{H} \times_{kn-j+1} \mathbf{\Theta}. \qquad \square$$

A.6 Lie derivative and bracket in tensor form

Theorem 2.3 on the Lie derivative and the Lie bracket of Lemma 2.1 in tensor form are proven in this section.

Proof A.6 *For* $l = 0$ *the Lie derivative is equal to the scalar function* $g(\mathbf{x})$ *by definition, which leads to the tensor description*

$$\begin{aligned}L_{\mathbf{h}}^0 g(\mathbf{x}) &= \left\langle \mathsf{L}_{\mathsf{H},\mathsf{G},0} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle \\ &= \left\langle \mathsf{G} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle = g(\mathbf{x}),\end{aligned}$$

with $\mathsf{L}_{\mathsf{H},\mathsf{G},0} = \mathsf{G}$. *The first Lie derivative, i.e.* $l = 1$, *of* $g(\mathbf{x})$ *along* $\mathbf{h}(\mathbf{x})$ *is defined by*

$$L_{\mathbf{h}} g(\mathbf{x}) = \sum_{i=1}^{n} h_i(\mathbf{x}) \frac{\partial}{\partial x_i} g(\mathbf{x}). \tag{A.16}$$

Since the scalar function $g(\mathbf{x})$ *is given in tensor form, (2.55) is applied to get the partial derivative*

$$\frac{\partial}{\partial x_i} g(\mathbf{x}) = \frac{\partial}{\partial x_i}\left\langle \mathsf{G} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle = \left\langle \sum_{k=1}^{N} \mathsf{G} \times_{kn-i+1} \mathbf{\Theta} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle, \; i = 1,\ldots,n.$$

The multiplication with function $h_i(\mathbf{x}) = \left\langle \mathsf{H}_i \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle$, *where* $i = 1, \ldots, n$, *using (2.53) yields*

$$\begin{aligned} h_i(\mathbf{x}) \frac{\partial}{\partial x_i} g(\mathbf{x}) &= \left\langle \mathsf{H}_i \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle \cdot \left\langle \sum_{k=1}^{N} \mathsf{G} \times_{kn-i+1} \boldsymbol{\Theta} \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle \\ &= \left\langle \mathsf{H}_i \circ \left(\sum_{k=1}^{N} \mathsf{G} \times_{kn-i+1} \boldsymbol{\Theta} \right) \,\middle|\, \mathsf{M}_p^{2N}(\mathbf{x}) \right\rangle. \end{aligned}$$

Summing up these elements to get the Lie derivative (A.16) leads to

$$\begin{aligned} L_{\mathbf{h}} g(\mathbf{x}) &= \sum_{i=1}^{n} h_i(\mathbf{x}) \frac{\partial}{\partial x_i} g(\mathbf{x}) = \sum_{i=1}^{n} \left\langle \mathsf{H}_i \circ \left(\sum_{k=1}^{N} \mathsf{G} \times_{kn-i+1} \boldsymbol{\Theta} \right) \,\middle|\, \mathsf{M}_p^{2N}(\mathbf{x}) \right\rangle \\ &= \left\langle \sum_{i=1}^{n} \mathsf{H}_i \circ \left(\sum_{k=1}^{N} \mathsf{G} \times_{kn-i+1} \boldsymbol{\Theta} \right) \,\middle|\, \mathsf{M}_p^{2N}(\mathbf{x}) \right\rangle. \end{aligned}$$

This approach is extended to multiple Lie derivatives along $\mathbf{h}(\mathbf{x})$ *as given by (2.60) for arbitrary* $l \in \mathbb{N}$ *again by applying the concepts of multiplication (2.53) and differentiation (2.55) by*

$$\begin{aligned} L_{\mathbf{h}}^l g(\mathbf{x}) &= \sum_{i=1}^{n} h_i(\mathbf{x}) \frac{\partial}{\partial x_i} L_{\mathbf{h}}^{l-1} g(\mathbf{x}) = \sum_{i=1}^{n} \left\langle \mathsf{H}_i \,\middle|\, \mathsf{M}_p^N(\mathbf{x}) \right\rangle \cdot \left\langle \sum_{k=1}^{lN} \mathsf{L}_{\mathsf{H},\mathsf{G},l-1} \times_{kn-i+1} \boldsymbol{\Theta} \,\middle|\, \mathsf{M}_p^{lN}(\mathbf{x}) \right\rangle \\ &= \left\langle \sum_{i=1}^{n} \mathsf{H}_i \circ \left(\sum_{k=1}^{lN} \mathsf{L}_{\mathsf{H},\mathsf{G},l-1} \times_{kn-i+1} \boldsymbol{\Theta} \right) \,\middle|\, \mathsf{M}_p^{(l+1)N}(\mathbf{x}) \right\rangle. \quad \square \end{aligned}$$

Proof A.7 *The Lie bracket of two vector functions is defined by (2.63). With Theorem 2.3 the Lie derivatives of a vector function* $\mathbf{g}(\mathbf{x})$ *along a vector field* $\mathbf{h}(\mathbf{x})$ *and vice versa are written as*

$$\begin{aligned} L_{\mathbf{h}} \mathbf{g}(\mathbf{x}) &= \sum_{i=1}^{n} h_i(\mathbf{x}) \frac{\partial}{\partial x_i} \mathbf{g}(\mathbf{x}) = \left\langle \mathsf{L}_{\mathsf{H},\mathsf{G},1} \,\middle|\, \mathsf{M}_p^{2N}(\mathbf{x}) \right\rangle, \\ L_{\mathbf{g}} \mathbf{h}(\mathbf{x}) &= \sum_{i=1}^{n} g_i(\mathbf{x}) \frac{\partial}{\partial x_i} \mathbf{h}(\mathbf{x}) = \left\langle \mathsf{L}_{\mathsf{G},\mathsf{H},1} \,\middle|\, \mathsf{M}_p^{2N}(\mathbf{x}) \right\rangle. \end{aligned}$$

Inserting this to (2.63) and rearranging results in

$$[\mathbf{h}, \mathbf{g}] = \left\langle \mathsf{L}_{\mathsf{H},\mathsf{G},1} \,\middle|\, \mathsf{M}_p^{2N}(\mathbf{x}) \right\rangle - \left\langle \mathsf{L}_{\mathsf{G},\mathsf{H},1} \,\middle|\, \mathsf{M}_p^{2N}(\mathbf{x}) \right\rangle = \left\langle \mathsf{L}_{\mathsf{H},\mathsf{G},1} - \mathsf{L}_{\mathsf{G},\mathsf{H},1} \,\middle|\, \mathsf{M}_p^{2N}(\mathbf{x}) \right\rangle. \qquad \square$$

A.7 Scaling of MTI models

The proof of the scaling of MTI state space model introduced in Section 3.5 is given here.

Proof A.8 *Inserting the linear transformations of the states and inputs to the factor matrices of the monomial tensor (3.12) gives*

$$\begin{aligned}\mathsf{M}\left(\tilde{\mathbf{x}},\tilde{\mathbf{u}}\right) &= \left[\begin{pmatrix}1\\ \tilde{a}_{n+m}\tilde{u}_{n+m}+\tilde{b}_{n+m}\end{pmatrix},\ldots,\begin{pmatrix}1\\ \tilde{a}_1\tilde{x}_1+\tilde{b}_1\end{pmatrix}\right]\\ &= \left[\begin{pmatrix}1 & 0\\ \tilde{b}_{n+m} & \tilde{a}_{n+m}\end{pmatrix}\begin{pmatrix}1\\ \tilde{u}_m\end{pmatrix},\ldots,\begin{pmatrix}1 & 0\\ \tilde{b}_1 & \tilde{a}_1\end{pmatrix}\begin{pmatrix}1\\ \tilde{x}_1\end{pmatrix}\right].\end{aligned}$$

This description of the monomial tensor in the scaled variables is used together with scaling of the time derivative of the states $\dot{x}_i = \tilde{a}_i\dot{\tilde{x}}_i$ in (3.37) to describe the dynamics of the scaled system with (3.37) by

$$\begin{aligned}\dot{\mathbf{x}} &= \mathsf{F}\bar{\times}_1\begin{pmatrix}1\\ u_m\end{pmatrix}\bar{\times}_2\begin{pmatrix}1\\ u_{m-1}\end{pmatrix}\bar{\times}_3\cdots\bar{\times}_{n+m}\begin{pmatrix}1\\ x_1\end{pmatrix},\\ \operatorname*{diag}_{i=1,\ldots,n}\left(\tilde{a}_i\right)\dot{\tilde{\mathbf{x}}} &= \mathsf{F}\bar{\times}_1\begin{pmatrix}1 & 0\\ \tilde{b}_{n+m} & \tilde{a}_{n+m}\end{pmatrix}\begin{pmatrix}1\\ \tilde{u}_m\end{pmatrix}\bar{\times}_2\cdots\bar{\times}_{n+m}\begin{pmatrix}1 & 0\\ \tilde{b}_1 & \tilde{a}_1\end{pmatrix}\begin{pmatrix}1\\ \tilde{x}_1\end{pmatrix}.\end{aligned}$$

With the Definition 2.9 of the mode-k tensor matrix product, the parameters and the scaled variables can be separated yielding

$$\begin{aligned}\dot{\tilde{\mathbf{x}}} &= \operatorname*{diag}_{i=1,\ldots,n}\left(\tilde{a}_i^{-1}\right)\mathsf{F}\times_1\begin{pmatrix}1 & 0\\ \tilde{b}_{n+m} & \tilde{a}_{n+m}\end{pmatrix}\times_2\cdots\times_{n+m}\begin{pmatrix}1 & 0\\ \tilde{b}_1 & \tilde{a}_1\end{pmatrix}\bar{\times}_1\begin{pmatrix}1\\ \tilde{u}_m\end{pmatrix}\bar{\times}_2\cdots\bar{\times}_{n+m}\begin{pmatrix}1\\ \tilde{x}_1\end{pmatrix}\\ &= \mathsf{F}\times_1\begin{pmatrix}1 & 0\\ \tilde{b}_{n+m} & \tilde{a}_{n+m}\end{pmatrix}\times_2\cdots\times_{n+m}\begin{pmatrix}1 & 0\\ \tilde{b}_1 & \tilde{a}_1\end{pmatrix}\times_{n+m+1}\operatorname*{diag}_{i=1,\ldots,n}\left(a_i^{-1}\right)\bar{\times}_1\begin{pmatrix}1\\ \tilde{u}_m\end{pmatrix}\bar{\times}_2\cdots\bar{\times}_{n+m}\begin{pmatrix}1\\ \tilde{x}_1\end{pmatrix}\\ &= \left\langle\,\tilde{\mathsf{F}}\mid\mathsf{M}\left(\tilde{\mathbf{x}},\tilde{\mathbf{u}}\right)\right\rangle.\end{aligned}$$

The proof of the transformation of the output equation

$$\mathbf{y} = \mathsf{G}\bar{\times}_1\begin{pmatrix}1\\ u_m\end{pmatrix}\bar{\times}_2\begin{pmatrix}1\\ u_{m-1}\end{pmatrix}\bar{\times}_3\cdots\bar{\times}_{n+m}\begin{pmatrix}1\\ x_1\end{pmatrix}$$

follows from the monomial tensor in the scaled variables, too. Inserting the scaled monomial to the output equation gives

$$\begin{aligned}&\operatorname*{diag}_{i=1,\ldots,n}\left(\tilde{c}_i\right)\tilde{\mathbf{y}}+\begin{pmatrix}\tilde{e}_1 & \cdots & \tilde{e}_p\end{pmatrix}^T =\\ &\mathsf{G}\times_1\begin{pmatrix}1 & 0\\ \tilde{b}_{n+m} & \tilde{a}_{n+m}\end{pmatrix}\times_2\cdots\times_{n+m}\begin{pmatrix}1 & 0\\ \tilde{b}_1 & \tilde{a}_1\end{pmatrix}\bar{\times}_1\begin{pmatrix}1\\ \tilde{u}_m\end{pmatrix}\bar{\times}_2\cdots\bar{\times}_{n+m}\begin{pmatrix}1\\ \tilde{x}_1\end{pmatrix},\\ \tilde{\mathbf{y}} &= \mathsf{G}\times_1\begin{pmatrix}1 & 0\\ \tilde{b}_{n+m} & \tilde{a}_{n+m}\end{pmatrix}\times_2\cdots\times_{n+m}\begin{pmatrix}1 & 0\\ \tilde{b}_1 & \tilde{a}_1\end{pmatrix}\\ &\quad\times_{n+m+1}\operatorname*{diag}_{i=1,\ldots,n}\left(\tilde{c}_i^{-1}\right)\bar{\times}_1\begin{pmatrix}1\\ \tilde{u}_m\end{pmatrix}\bar{\times}_2\cdots\bar{\times}_{n+m}\begin{pmatrix}1\\ \tilde{x}_1\end{pmatrix}-\operatorname{diag}\left(\tilde{c}_i^{-1}\right)\begin{pmatrix}\tilde{e}_1 & \cdots & \tilde{e}_p\end{pmatrix}^T\\ &= \left\langle\,\hat{\mathsf{G}}\mid\mathsf{M}\left(\tilde{\mathbf{x}},\tilde{\mathbf{u}}\right)\right\rangle-\operatorname*{diag}_{i=1,\ldots,n}\left(\tilde{c}_i^{-1}\right)\begin{pmatrix}\tilde{e}_1 & \cdots & \tilde{e}_p\end{pmatrix}^T.\end{aligned}\tag{A.17}$$

The parameter tensor E *is defined by (3.69) to (3.71) in CP description, leading to*

$$\underset{i=1,\ldots,n}{\operatorname{diag}}\left(\tilde{c}_i^{-1}\right)\begin{pmatrix}\tilde{e}_1 & \cdots & \tilde{e}_p\end{pmatrix}^T = \langle\, \mathsf{E} \mid \mathsf{M}\left(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}\right) \rangle\,.$$

Putting this into (A.17)

$$\tilde{\mathbf{y}} = \left\langle\, \hat{\mathsf{G}} \mid \mathsf{M}\left(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}\right) \right\rangle - \langle\, \mathsf{E} \mid \mathsf{M}\left(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}\right) \rangle = \left\langle\, \hat{\mathsf{G}} - \mathsf{E} \mid \mathsf{M}\left(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}\right) \right\rangle,$$

gives the output equation in the transformed variables. □

A.8 Feedback linearization for MTI systems

The section presents the proofs of the Lemmas (4.1) to (4.3) belonging the the feedback linearization of decomposed MTI systems. The proofs are given in the same order than the lemmas in Section (4.2).

Proof A.9 *The representation of the Lie derivatives of the system (4.21) and (4.22) follows obviously from the tensor description of the Lie derivatives of Theorem 2.3.*

Proof A.10 *Inserting the tensor representation of the Lie derivatives defined in Lemma 4.1 into the controller function (4.16) and rearranging leads to*

$$u = \frac{-\sum\limits_{i=0}^{\rho_{sys}} \mu_i \left\langle\, \mathsf{L}_{\mathsf{A},\mathsf{C},i} \,\middle|\, \mathsf{M}_p^{i+1}\left(\mathbf{x}\right) \right\rangle + \mu_0\, r}{\left\langle\, \mathsf{L}_{\mathsf{B},\mathsf{A},\mathsf{C},\rho_{sys}-1} \,\middle|\, \mathsf{M}_p^{\rho_{sys}+1}\left(\mathbf{x}\right) \right\rangle} = \frac{-\left\langle\, \sum\limits_{i=0}^{\rho_{sys}} \mu_i \mathsf{L}_{\mathsf{A},\mathsf{C},i} \,\middle|\, \mathsf{M}_p^{\rho_{sys}+1}\left(\mathbf{x}\right) \right\rangle + \mu_0\, r}{\left\langle\, \mathsf{L}_{\mathsf{B},\mathsf{A},\mathsf{C},\rho_{sys}-1} \,\middle|\, \mathsf{M}_p^{\rho_{sys}+1}\left(\mathbf{x}\right) \right\rangle}.$$ □

Proof A.11 *The conditions for an affine SISO system to be feedback linearizable with relative degree of n are given for nonlinear systems in [39]. The matrix*

$$\begin{pmatrix}\mathbf{b} & ad_{\mathbf{a}}\mathbf{b}(\mathbf{x_0}) & \cdots & ad_{\mathbf{a}}^{n-1}\mathbf{b}(\mathbf{x_0})\end{pmatrix}$$

must have rank n and can be expressed in the case of an MTI model given in tensor form by $\left\langle\, \mathsf{T} \,\middle|\, \mathsf{M}_p^n\left(\mathbf{x}_0\right) \right\rangle$. *Each repeated Lie bracket* $\left\langle\, \mathsf{L}_{ad_{\mathbf{a}}\mathbf{b}}^{k-1} \,\middle|\, \mathsf{M}_p^n\left(\mathbf{x}\right) \right\rangle$ *is described with respect to the monomial tensor of order n and concatenated resulting in*

$$\mathsf{T} = \mathsf{L}_{ad_{\mathbf{a}}\mathbf{b}}^0 \boxplus_{n+2} \cdots \boxplus_{n+2} \mathsf{L}_{ad_{\mathbf{a}}\mathbf{b}}^{n-1}.$$

The evaluation of T *with the monomial tensor of order n gives the matrix of the Lie brackets*

$$\left\langle\, \mathsf{T} \,\middle|\, \mathsf{M}_p^n\left(\mathbf{x}_0\right) \right\rangle = \begin{pmatrix}\mathbf{b} & ad_{\mathbf{a}}\mathbf{b}(\mathbf{x_0}) & \cdots & ad_{\mathbf{a}}^{n-1}\mathbf{b}(\mathbf{x_0})\end{pmatrix}.$$

The matrix should have rank n, which can be easily checked, by evaluation of the contracted product at $\mathbf{x}_0$. *The second condition is the tensor formulation of the condition that the distribution* $\operatorname{span}\{\mathbf{b}, ad_{\mathbf{a}}\mathbf{b}, \ldots, ad_{\mathbf{a}}^{n-2}\mathbf{b})\}$ *is involutive at* $\mathbf{x}_0$. *The distribution is involutive if and only if*

$$\operatorname{rank}\left(\begin{pmatrix}\mathbf{b}(\mathbf{x}_0) & \cdots & ad_{\mathbf{a}}^{n-2}\mathbf{b}(\mathbf{x_0}) & [ad_{\mathbf{a}}^i\mathbf{b}(\mathbf{x_0}), ad_{\mathbf{a}}^j\mathbf{b}(\mathbf{x_0})]\end{pmatrix}\right) = \operatorname{rank}\left(\begin{pmatrix}\mathbf{b}(\mathbf{x}_0) & \cdots & ad_{\mathbf{a}}^{n-2}\mathbf{b}(\mathbf{x_0})\end{pmatrix}\right)$$

is fulfilled for all i and $j = 0, \ldots, n-2$. *Describing the two matrices in terms of parameter tensors, like it was done for the rank condition before, leads to the second condition.* □

B Application models

In Chapter 5 applications of the MTI model representation and design methods are described. In the following the state equations of the heating system and HVAC example are given. Furthermore the Lie derivatives of the feedback linearization application are presented.

B.1 Heating system

The thermal behavior of the heating system is described by the system of differential equations (5.1), (5.9) and (5.10). Putting the definitions of the states, inputs and disturbances (5.25) to (5.27) into the state equations leads to the state space model with the state equations belonging to the boilers

$$\begin{aligned}\dot{x}_i =& \frac{k_{b,i}}{c\rho V_{b,i}} T_{amb} - \frac{k_{b,i}}{c\rho V_{b,i}} x_i + \frac{P_{max,i}}{c\rho V_{b,i}} u_i + \sum_{j=1}^{N_{HC}} -\frac{\dot{V}_{max,j}}{N_B V_{b,i}} x_i u_{N_B+j} \\ &+ \frac{\dot{V}_{max,j}}{N_B V_{b,i}} x_i u_{N_B+j} u_{N_B+N_{HC}+j} + \frac{\dot{V}_{max,j}}{N_B V_{b,i}} x_{N_B+j} u_{N_B+j} - \frac{\dot{V}_{max,j}}{N_B V_{b,i}} x_{N_B+j} u_{N_B+j} u_{N_B+N_{HC}+j},\end{aligned}$$

with $i = 1, \ldots, N_B$ and the state equations belonging to the consumers

$$\begin{aligned}\dot{x}_{N_B+i} =& -\frac{k_{rb,i}}{c\rho V_{c,i}} x_{N_B+i} + \frac{k_{rb,i}}{c\rho V_{c,i}} x_{N_B+N_{HC}+i} - \frac{\dot{V}_{max,i}}{V_{c,i}} x_{N_B+i} u_{N_B+i} \\ &+ \frac{\dot{V}_{max,i}}{V_{c,i}} x_{N_B+i} u_{N_B+i} u_{N_B+N_{HC}+i} + \sum_{j=1}^{N_B} \frac{\dot{V}_{max,i}}{N_B V_{c,i}} x_j u_{N_B+i} - \frac{\dot{V}_{max,i}}{N_B V_{c,i}} x_j u_{N_B+i} u_{N_B+N_{HC}+i}, \\ \dot{x}_{N_B+N_{HC}+i} =& \frac{k_{rb,i}}{C_{b,i}} x_{N_B+i} + \frac{-k_{rb,i} - k_{bo,i}}{C_{b,i}} x_{N_B+N_{HC}+i} + \frac{k_{bo,i}}{C_{b,i}} d_1 + \frac{k_{s,i}}{C_{b,i}} d_2,\end{aligned}$$

with $i = 1, \ldots, N_{HC}$.

B.2 HVAC system

Inserting the definitions of states, inputs and disturbances (5.28) to (5.30) to the state equations (5.11) to (5.24) leads to the state space model of the HVAC system of Section 5.1.2.

The state equations are given by

$$
\begin{aligned}
\dot{x}_1 =& -\left(\frac{2A_{w,1}U_{w,1}+2A_{w,2}U_{w,2}+A_RU_R}{C_b}\right)x_1+\frac{2A_{w,1}U_{w,1}}{C_b}x_2+\frac{2A_{w,2}U_{w,2}}{C_b}x_3+\frac{A_RU_R}{C_b}x_4\\
&-\frac{c_a\rho_a}{C_b}x_1x_{12}+\frac{1}{C_b}d_1,\\
\dot{x}_2 =& \frac{A_{w,1}U_{w,1}}{C_{w,1}}x_1-\frac{2A_{w,1}U_{w,1}}{C_{w,1}}x_2+\frac{A_{w,1}U_{w,1}}{C_{w,1}}d_3,\\
\dot{x}_3 =& \frac{A_{w,2}U_{w,2}}{C_{w,2}}x_1-\frac{2A_{w,2}U_{w,2}}{C_{w,2}}x_3+\frac{A_{w,2}U_{w,2}}{C_{w,2}}d_3,\\
\dot{x}_4 =& \frac{A_RU_R}{C_R}x_1-\frac{2A_RU_R}{C_R}x_4+\frac{A_RU_R}{C_R}d_3,\\
\dot{x}_5 =& -\frac{1}{V_b}x_5x_{12}+\frac{1}{V_b}x_{11}x_{12}+\frac{1}{V_b}d_2,\\
\dot{x}_6 =& \frac{(h_i+h_o)\rho_ac_a}{h_iM_dC_d}x_1x_{12}-\frac{(h_i+h_o)\rho_ac_a}{h_iM_dC_d}x_6x_{12},\\
\dot{x}_7 =& -\frac{U_cA_c}{C_h}x_7-\frac{c_a\rho_a}{C_h}x_7x_{12}+\frac{c_a\rho_a}{C_h}x_6x_{12}x_{13}+\frac{c\rho}{C_h}x_{14}x_{15}-\frac{c\rho}{C_h}x_{14}x_{16}+\frac{U_cA_c}{C_h}d_3\\
&+\frac{c_a\rho_a}{C_h}x_{12}d_3-\frac{c_a\rho_a}{C_h}x_{12}x_{13}d_3,\\
\dot{x}_8 =& -x_8x_{12}+x_5x_{12}x_{13}-x_{12}d_4-x_{12}x_{13}d_4,\\
\dot{x}_9 =& \frac{(h_i+h_o)\rho_ac_a}{h_iM_dC_d}x_7x_{12}-\frac{(h_i+h_o)\rho_ac_a}{h_iM_dC_d}x_9x_{12},\\
\dot{x}_{10} =& -\frac{U_hA_h}{C_h}x_{10}+\frac{c_a}{C_h}x_9x_{12}-\frac{c_a}{C_h}x_{10}x_{12}+\frac{U_hA_h}{C_h}d_3,\\
\dot{x}_{11} =& \frac{1}{V_h}x_8x_{12}-\frac{1}{V_h}x_{11}x_{12}+\frac{1}{\rho_aV_h}x_{17},\\
\dot{x}_{12} =& -\frac{1}{\tau_{\dot{V},a}}x_{12}+\frac{1}{\tau_{\dot{V},a}}u_1,\\
\dot{x}_{13} =& -\frac{1}{\tau_\mu}x_{13}+\frac{1}{\tau_\mu}u_2,\\
\dot{x}_{14} =& -\frac{1}{\tau_{\dot{V}w}}x_{14}+\frac{1}{\tau_{\dot{V}w}}u_3,\\
\dot{x}_{15} =& -\frac{1}{\tau_{Twi}}x_{15}+\frac{1}{\tau_{Twi}}u_4,\\
\dot{x}_{16} =& -\frac{1}{\tau_{Two}}x_{16}+\frac{1}{\tau_{Two}}u_5,\\
\dot{x}_{17} =& -\frac{1}{\tau_{hvapor}}x_{17}+\frac{1}{\tau_{hvapor}}u_6.
\end{aligned}
$$

B.3 Lie derivatives for the feedback linearzation controller

In Section 5.3.1 a feedback linearization controller was derived for a heating systems example. The heating system model is represented as decomposed MTI system. To design the controller, the Lie derivatives of the model have to be computed. This is done here by the tensor based approach derived in Section 4.2. Evaluating the contracted product of the parameter and monomial tensors leads to the symbolic description of the Lie derivatives

$$
\begin{aligned}
L_{\mathbf{a}}^{0}c(\mathbf{x}) &= \langle\, \mathsf{C} \mid \mathsf{M}\left(x_1, x_2, x_3\right) \rangle = x_3, \\
L_{\mathbf{a}}^{1}c(\mathbf{x}) &= \left\langle\, \mathsf{L}_{\mathsf{A},\mathsf{C},1} \,\middle|\, \mathsf{M}_p^2\left(x_1, x_2, x_3\right) \right\rangle = \frac{1}{C_b}\left(T_o k_{bo} + \left(-k_{rb} - k_{bo}\right) x_3 + k_{rb} x_1\right), \\
L_{\mathbf{a}}^{2}c(\mathbf{x}) &= \left\langle\, \mathsf{L}_{\mathsf{A},\mathsf{C},2} \,\middle|\, \mathsf{M}_p^3\left(x_1, x_2, x_3\right) \right\rangle = -\frac{k_{rb}}{c\rho V_c C_b}\left(c\rho x_1 x_2 - c\rho T_s x_2 + k_{rb}\left(x_1 - x_3\right)\right) \\
&\quad - \frac{k_{bo} + k_{rb}}{C_b^2}\left(k_{bo} T_o - \left(k_{bo} + k_{rb}\right) x_3 + k_{rb} x_1\right), \\
L_{\mathbf{a}}^{3}c(\mathbf{x}) &= \left\langle\, \mathsf{L}_{\mathsf{A},\mathsf{C},3} \,\middle|\, \mathsf{M}_p^4\left(x_1, x_2, x_3\right) \right\rangle \\
&= \left(\frac{c\rho V_c \left(k_{bo} + k_{rb}\right)^2 + C_b k_{rb}^2}{c\rho V_c C_b^3}\right)\left(k_{bo} T_o - \left(k_{bo} + k_{rb}\right) x_3 + k_{rb} x_1\right) - \frac{k_{rb} x_2 \left(T_s - x_1\right)}{V_c C_b \tau_{\dot{V}}} \\
&\quad + \left(\frac{k_{rb}\left(c\rho x_2 + k_{rb}\right)}{c^2\rho^2 V_c^2 C_b} + \frac{k_{rb}\left(k_{bo} + k_{rb}\right)}{c\rho V_c C_b^2}\right)\left(c\rho x_1 x_2 - c\rho T_s x_2 + k_{rb}\left(x_1 - x_3\right)\right).
\end{aligned}
$$

The control law (5.38) of the feedback linearization controller follows from these Lie derivatives.