OBSERVE

Arbeitspaket AP A.4

Dezentrale Netzwerkregelungen, Methoden und Design

Kai Kruppa, Björn Lautenschlager, Gerwald Lichtenberg
kai.kruppa @ haw-hamburg.de,
bjoern.lautenschlager@ haw-hamburg.de,
gerwald.lichtenberg @ haw-hamburg.de

Hochschule für Angewandte Wissenschaften Hamburg
Fakultät Life Sciences
Ulmenliet 20, 21033 Hamburg

Datum: 2. Juni 2018

Gefördert durch:

Bundesministerium
für Wirtschaft
und Energie

aufgrund eines Beschlusses
des Deutschen Bundestages

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Inhalt

6 Dezentrale Netzwerkregelungen, Methoden und Design

Die Verantwortung für den Inhalt dieser Veröffentlichung liegt bei den Autoren.

Das folgende Kapitel erläutert die Ergebnisse des Arbeitspaketes A4 ”Dezentrale Netzwerkregelungen, Methoden und Design”. Dazu werden in dem Kapitel 6.1 zunächst die Modellierung von dynamischen System mithilfe von multilinearen Modellen, die sich durch Tensoren darstellen lassen, beschrieben. Für diese spezielle Klasse der Systeme, die gut für die Modellbildung im Bereich der Gebäudetechnik geeignet ist, wurden verschiedene Regelungsverfahren untersucht. Die Methode der Feedback Linearisierung wurde für den Spezialfall der mutlilinearen Systeme in Kapitel 6.2 angepasst. Um den Kommunikationsaufwand in sehr großen Anlagen zu reduzieren, bieten sich dezentrale oder verteilte Reglerkonzepte an. Ein Verfahren für eine dezentrale Zustandsrückführung wird in Kapitel 6.3 vorgestellt. Die prädiktive Regelung ist eine interessante Regelungsmethode für Heizungssysteme. Im Kapitel 6.4 wird die Anpassung des Standardverfahrens bei der prädiktiven Regelung auf multilineare Modelle untersucht, die dann in verteilten Reglerstrukturen in Kapitel 6.5 angewendet wird, um den Berechnungsaufwand zu reduzieren. Die Methoden, die für die multilinearen Systeme entwickelt wurden, sind in einer Toolbox in MATLAB zusammengefasst, die in Kapitel 6.6 beschrieben wird. Abschließend wurde ein prädiktiver Regler an einem realen Gebäude implementiert. Die Ergebnisse der Tests sind im Kapitel 6.7 erläutert.

6.1 Modellierung von Heizungsanlagen mit multilinearen Modellen

In der Regelungstechnik sind viele Methoden und Werkzeuge für die Modellierung und den Reglerentwurf mit linearen Modellen verfügbar, [1]. Diese Methoden liefern jedoch nicht das gewünschte Ergebnis, wenn nichtlineare Effekte einen deutlichen Einfluss auf das Systemverhalten haben. Multilineare Systeme erweitern die Klasse der linearen, indem sie auch Multiplikationen von Zuständen und Eingängen mit Zuständen und Eingängen in der Form erlauben, dass sich ein lineares System ergibt, wenn alle bis auf eine Variable konstant gehalten werden. Somit lassen sich komplexere Dynamiken abbilden. Dies soll hier an einem einfachen Beispiel aus dem Bereich der Heizungstechnik verdeutlicht werden. Ein statischer Heizkörper, wie in Abbildung 6.1-1 dargestellt, wird über einen Volumenstrom $\dot{V}$ mit der Vorlauftemperatur T_{Vl} versorgt. Der Heizkörper gibt eine gewisse Wärmeleistung $\dot{Q}_{waerme}$, die hier als konstant angenommen wird, an einen Raum ab, sodass kühleres Wasser mit dem gleichen Volumenstrom und der Rücklauftemperatur T_{Rl} den Heizkörper verlässt.

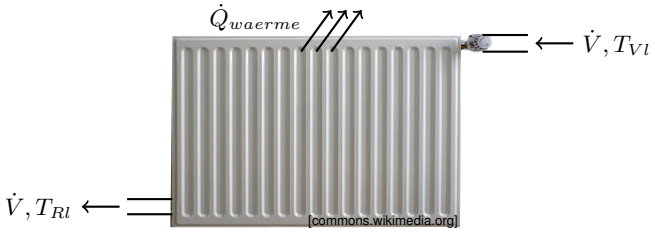


Abbildung 6.1-1: Heizkörper Beispiel

Mithilfe einer Wärmeleistungsbilanz lässt sich das thermische Verhalten des Heizkörper durch eine Differentialgleichung 1. Ordnung

$$\dot{T}_{Rl} = \frac{1}{V_{rad}} \left(\dot{V} T_{Vl} - \dot{V} T_{Rl} \right) - \frac{1}{c \rho V_{rad}} \dot{Q}_{waerme}$$

mit dem Heizkörpervolumen V_{rad} beschrieben und so die Rücklauftemperatur bestimmen, [40]. Die Rücklauftemperatur ist der Zustand $x = T_{Rl}$ dieses Modells. Die Eingänge sind durch den Volumenstrom $u_1 = \dot{V}$ und die Vorlauftemperatur $u_2 = T_{Vl}$ gegeben. Setzt man dies in die Differentialgleichung

ein, ergibt sich

$$\dot{x} = \frac{1}{V_{rad}} (u_1 u_2 - u_1 x) - \frac{1}{c\rho V_{rad}} \dot{Q}_{waerme}.$$

Es wir deutlich, dass hier aufgrund der Berechnung der Wärmeleistung und der damit einhergehenden Multiplikation von Temperaturen und Volumenströmen, Multiplikationen von Eingängen und Eingängen sowie Eingängen und Zuständen auftreten. Das Modell ist somit nicht mehr linear, jedoch in der Klasse der multilinearen Modelle enthalten. Um die Auswirkungen zu bestimmen, soll hier die Linearisierung des Modells mit der entsprechenden multilinearen Implementierung anhand von einfachen Simulationen verglichen werden. Dazu wird das Modell um einen Arbeitspunkt (OP, operating point) linearisiert. In der Abbildung 6.1-2 sind die Eingangssignale $\dot{V}$ und T_{Vl} sowie die mit dem linearen und dem multilinearen Modell berechnete Rücklauftemperatur zu sehen.

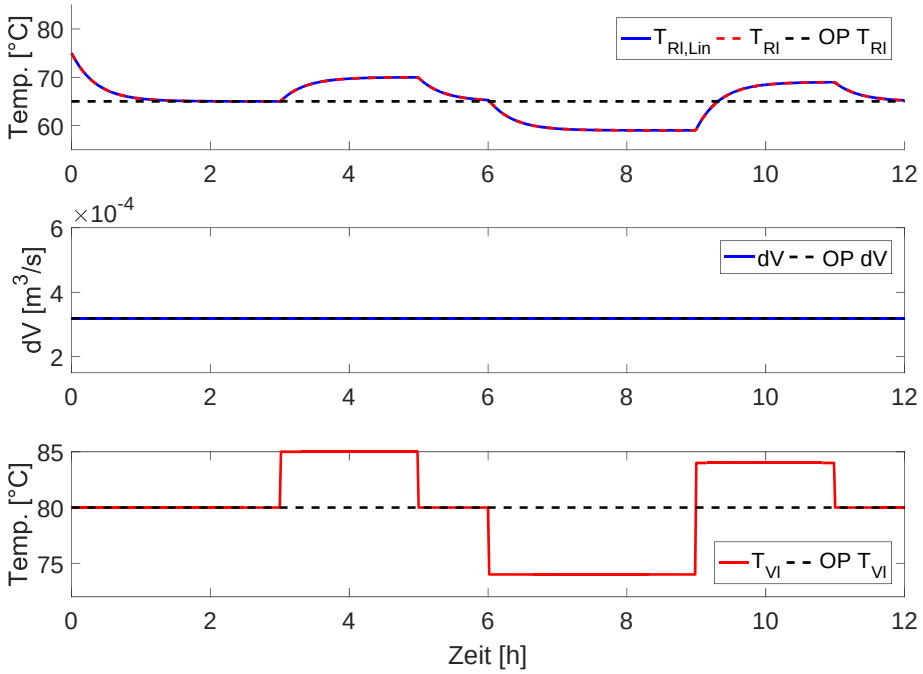


Abbildung 6.1-2: Simulationsergebnis des Heizkörperbeispiels mit konstantem Volumenstrom

Der Volumenstrom war hierbei konstant auf dem Wert des Arbeitspunktes und es ist zu sehen, dass das lineare Modell das Systemverhalten gut abbildet. Ein zweite Simulation wurde mit einem sich ändernden Volumenstrom durchgeführt, welches in Abbildung 6.1-3 gezeigt ist.

Ändert sich der Volumenstrom, d.h. weicht er vom Arbeitspunkt ab, kann das lineare Modell nicht mehr alle Dynamiken abbilden. Mit dem multilinearen ist dies möglich. Aufgrund dieser strukturellen Eigenschaft, die hier Anhand eines einfachen Beispiels gezeigt wurde, ist diese Modellklasse gut für die Modellierung in der Anwendung in der Heizungstechnik geeignet.

Multilineare zeit-invariante (MTI, multilinear time invariant) Systeme können mithilfe von Tensoren dargestellt werden. Viele Modelle aus dem Bereich der Heizungstechnik fallen in diese Modellklasse und ihr dynamisches Verhalten kann somit oftmals sehr gut mit solchen MTI Systemen abgebildet werden, [41]. Für komplexe Systeme ergibt sich jedoch das Problem, dass die Anzahl der Parameter exponentiell mit der Systemgröße ansteigt und somit nicht mehr effizient gespeichert werden kann. Daher bietet es sich an, Tensordekompositionsverfahren anzuwenden, um die Modellkomplexität hinsichtlich der Anzahl der Parameter zu reduzieren. Verschiedene aktuelle Zerlegungsverfahren aus der Mathematik, die Canonical Polyadic (CP), Tucker, Tensor Train (TT) und Hierarchical Tucker (HT) Dekomposition wurden als geeignete Methoden ermittelt,[15]. Es wird im Folgenden untersucht, wie eine Darstellung der MTI Systeme in jeder dieser vier Zerlegungen möglich ist. Zudem wird betrachtet, inwiefern diese Darstellung auch für die Simulation verwendet werden kann.

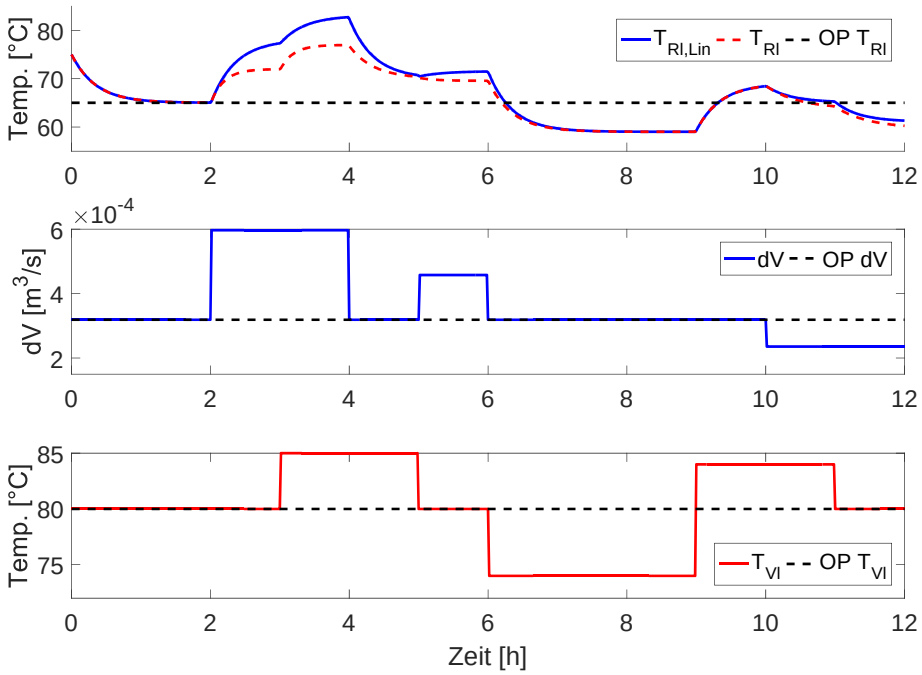


Abbildung 6.1-3: Simulationsergebnis des Heizkörperbeispiels mit variablem Volumenstrom

6.1.1 Multilineare Modelle

Modelle von realen dynamischen Systemen helfen, ihr zukünftiges Verhalten vorherzusagen. Diese Modelle können daraufhin z.B. unter anderem für den Test neuer Systemkomponenten, die Fehlerdetektion oder den Reglerentwurf verwendet werden. Das System wird von außen durch Eingänge $\mathbf{u} \in \mathbb{R}^m$ beeinflusst. Die Ausgangssignale $\mathbf{y} \in \mathbb{R}^p$ werden durch Sensoren gemessen. Die gesamte relevante Information über das System ist durch die Zustände $\mathbf{x} \in \mathbb{R}^n$ gegeben. Es wird angenommen, dass alle Signale reell sind. In diesem Kapitel werden keine schaltenden oder hybriden Systemen betrachtet, bei denen auch boolesche Signale auftreten. Ein System mit Eingängen, Ausgängen und Zuständen ist in Abbildung 6.1-4 dargestellt.

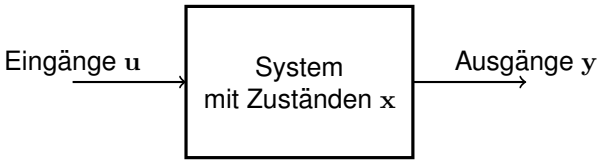


Abbildung 6.1-4: System mit Eingängen, Ausgängen und Zuständen

Es wird angenommen, dass sich das dynamische Verhalten des Systems nicht mit der Zeit ändert, d.h. es werden zeitinvariante Systeme betrachtet. Die zweite Annahme ist, dass das dynamische Verhalten der Systeme im zeit-kontinuierlichen Fall durch gewöhnliche Differentialgleichungen beschrieben werden kann. Die Modelle werden in Form von Zustandsraummodellen durch ein System von gewöhnlichen Differentialgleichungen 1. Ordnung dargestellt.

Den allgemeinsten Fall bildet die Systemklasse der nichtlinearen Modelle, [20]. Ein zeitkontinuierliches, nichtlineares Zustandsraummodell

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \tag{6.1-1}$$
$$\mathbf{y}(t) = \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t)), \tag{6.1-2}$$

mit der Zeit $t \in \mathbb{R}$ hat die Zustandsübergangsfunktion $\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ und die Ausgangsfunktion $\mathbf{g}$:

$\mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^p$. Im zeitdiskreten Fall wird das Modell durch Differenzengleichungen beschrieben

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k)), \tag{6.1-3}$$
$$\mathbf{y}(k) = \mathbf{g}(\mathbf{x}(k), \mathbf{u}(k)). \tag{6.1-4}$$

Die Abtastzeit des Systems ist T_{sample} und $k = 0, 1, 2, \ldots \in \mathbb{N}$ beschreibt den Zeitindex für die Zeitschritte kT_{sample}. Die Struktur der rechten Seiten der zeitkontinuierlichen Systeme und der zeitdiskreten Systeme ist vergleichbar, sodass viele Methoden für beide Modelle angewendet werden können. Daher kann deren Beschreibung durch den Operator $\Phi(\mathbf{x})$ zusammengefasst werden. Im zeitkontinuierlichen beschreibt $\Phi(\mathbf{x})$ die zeitliche Ableitung $\dot{\mathbf{x}}$ der Zustände $\mathbf{x}$, im zeitdiskreten den Folgezustand $\mathbf{x}(k+1)$. Um die Notation zu vereinfachen, wird im Folgenden auf die explizite Angabe des Zeitindex t bzw. k verzichtet. Mit einem gegebenen Anfangswert $\mathbf{x}_0 = \mathbf{x}(0)$ und einer Eingangstrajektorie kann mit den Modellen (6.1-1) und (6.1-2) bzw. (6.1-3) und (6.1-4) das zukünftige Systemverhalten mithilfe von Integrationsroutinen im kontinuierlichen Fall oder durch rekursive Anwendung der Modellgleichungen im diskreten berechnet werden. Dies wird Simulation genannt, [31].

Häufig werden in der Regelungstechnik lineare Zustandsraummodelle aufgrund ihrer einfachen Handhabung z.B. für die Systemanalyse oder den Reglerentwurf verwendet, [32]. Lineare Modelle haben allerdings den Nachteil, dass die Klasse sehr restriktiv ist, da nur lineare Kombinationen der Zustände und Eingänge in den rechten Seiten erlaubt sind. Ein lineares Zustandsraummodell ist gegeben durch

$$\Phi(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}, \tag{6.1-5}$$
$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}, \tag{6.1-6}$$

mit der Systemmatrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, Eingangsmatrix $\mathbf{B} \in \mathbb{R}^{n \times m}$, Ausgangsmatrix $\mathbf{C} \in \mathbb{R}^{p \times n}$ und Durchgriffsmatrix $\mathbf{D} \in \mathbb{R}^{p \times m}$.

Eine Systemklasse, mit der es möglich ist komplexere Dynamiken abzubilden als mit linearen Modellen, die aber trotzdem nicht beliebige Funktionen in den rechten Seiten erlaubt, ist die Klasse der multilinearen zeitinvarianten (MTI) Modelle. Wie in [28], [41] oder [40] eingeführt, werden die rechten Seiten von multilinearen Modellen durch multilineare Funktionen der Zustände und Eingänge beschrieben.

Definition 6.1.1 *Multilineare Funktionen*

$$h(\mathbf{x}) = \boldsymbol{\alpha}^T \mathbf{m}(\mathbf{x}) \tag{6.1-7}$$

mit Koeffizientenvektor $\boldsymbol{\alpha} = \begin{pmatrix} \alpha_1 & \cdots & \alpha_{2^n} \end{pmatrix}^T \in \mathbb{R}^{2^n}$ *und Monomvektor*

$$\mathbf{m}(\mathbf{x}) = \begin{pmatrix} 1 \\ x_n \end{pmatrix} \otimes \cdots \otimes \begin{pmatrix} 1 \\ x_1 \end{pmatrix} \in \mathbb{R}^{2^n} \tag{6.1-8}$$

sind Polynome in n *Variablen die linear sind, wenn alle Variablen bis auf eine konstant gehalten werden.*

Beispiel 6.1.1 *Eine mutlilineare Funktion mit 2 Variablen* x_1 *and* x_2 *ist gegeben durch*

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

Festhalten einer Variablen, z.B. $x_2 = 2$ *führt zu einer linearen Funktion in* x_1

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

Durch diese Darstellung mit einem Parametervektor und dem Monomvektor wird eine Aufteilung in Parameter und Variablen erreicht. Der Monomvektor enthält alle multiplikativen Kombinationen der Variablen, die in der multilinearen Klasse erlaubt sind. Verwendet man solche multilinearen Funktionen als rechte Seite eines Zustandsraummodells, erhält man die Zustandsraumdarstellung von multilinearen

Systemen in Matrixdarstellung

$$\Phi(\mathbf{x}) = \mathbf{F}\mathbf{m}(\mathbf{x}, \mathbf{u}), \tag{6.1-9}$$

$$\mathbf{y} = \mathbf{G}\mathbf{m}(\mathbf{x}, \mathbf{u}), \tag{6.1-10}$$

mit dem Monomvektor

$$\mathbf{m}(\mathbf{x}, \mathbf{u}) = \begin{pmatrix} 1 \\ u_m \end{pmatrix} \otimes \ldots \otimes \begin{pmatrix} 1 \\ u_1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ x_n \end{pmatrix} \otimes \cdots \otimes \begin{pmatrix} 1 \\ x_1 \end{pmatrix} \in \mathbb{R}^{2^{n+m}}, \tag{6.1-11}$$

der alle multiplikativen Kombinationen der Eingänge und Zustände enthält. Die Zustandsmatrix $\mathbf{F}$ und die Ausgangsmatrix $\mathbf{G}$ haben jeweils die Dimension $\mathbb{R}^{n \times 2^{n+m}}$ und $\mathbb{R}^{m \times 2^{n+m}}$.

Beispiel 6.1.2 *Die Zustandsgleichung eines multilinearen Zustandsraummodells mit einem Eingang und zwei Zuständen in Matrixdarstellung lautet*

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

Ein Einordnung der Klasse der multilinearen Systeme in häufig verwendete Systemklassen ist in der Abbildung 6.1-5 gegeben.

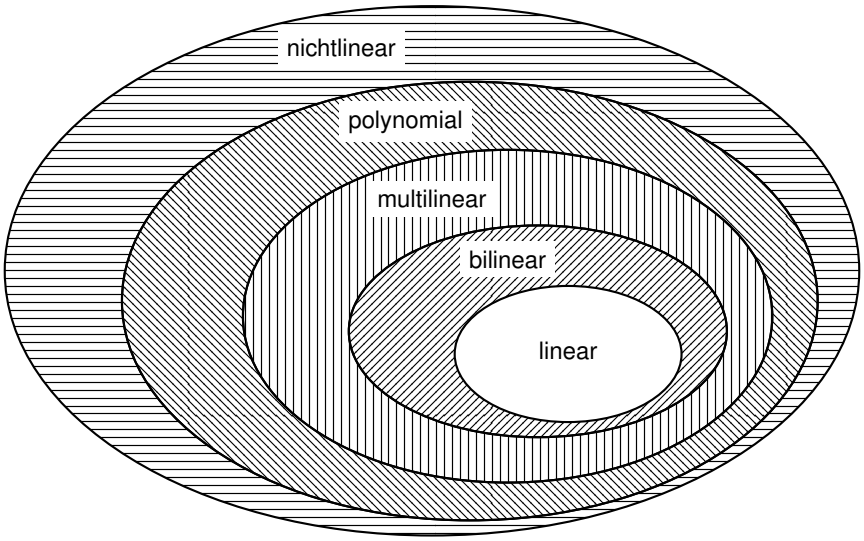


Abbildung 6.1-5: Modellklassen

Die allgemeinste Klasse sind nichtlineare Systeme, die beliebige polynomiale Terme und nichtlineare Funktionen wie Exponentialfunktionen in den rechten Seiten erlauben. Alle andere Klassen sind Subklassen der nichtlinearen Systemklasse. Die sehr allgemeine Formulierung der nichtlinearen Systeme ermöglicht eine große Flexibilität in der Modellierung, kann jedoch auch zu einer großen Komplexität in der Systemanalyse und dem Reglerentwurf führen, da keine Modellstruktur vorgegeben ist. Daher

existieren in vielen Fällen mathematisch komplexe Entwurfsmethoden, die teilweise nur für bestimmte Kategorien nichtlinearer Systeme gültig sind, [20].

Die Klasse mit den größten Einschränkungen ist die lineare Klasse, die häufig in der Regelungstechnik verwendet wird und häufig zu einfachen Algorithmen für den Entwurf führt, [31]. Allerdings scheitern diese Modelle, wenn Systeme betrachtet werden, bei denen nichtlineare Effekte einen großen Einfluss haben, da die linearen Modelle in diesen Fällen die Systemdynamiken nicht ausreichend abbilden können. Dies ist der Nachteil der Einfachheit. Für die Modellierung von Heizungssystemen wurde gezeigt, dass bilineare Systeme nicht ausreichend sind, um das Systemverhalten abzubilden, [41]. Durch die Modellierung mithilfe von Wärmeleistungsbilanzen treten Multiplikationen von Temperaturen und Volumenströmen bei der Modellierung von Heizungssystemen auf, welche zu multilinearen Termen in den Modellgleichungen führen. Daher eignen sich multilineare Modelle gut für die Modellierung dieser Systeme. Sie könnten auch durch polynomiale Modelle abgebildet werden. Jedoch zeigen sie nicht häufig Verhalten höherer Ordnung. Um die Modelle so einfach wie möglich zu halten, wird daher angenommen, dass die Effekte höherer Ordnung nur einen geringen Effekt auf das Systemverhalten haben und durch multilinear Modelle angenähert werden können, [23].

Daher sollen hier multilineare Modelle betrachtet werden. Die Modelle dieser Klasse sind nicht so allgemein formuliert, wie die nichtlinearen Modelle, da gewisse Strukturvorgaben gemacht werden. Dies ist ein großer Vorteil in der Entwicklung von Algorithmen in der Modellierung oder dem Reglerentwurf. Im Allgemeinen verringert die Spezialisierung auf eine gewisse Systemstruktur die Komplexität des Designprozesses und erhöht dessen Robustheit. Multilineare Systeme lassen sich mithilfe von Tensoren effizient darstellen.

6.1.2 Tensoren und Tensordekompositionen

In der Mathematik ist die Tensorrechnung ein aktives Forschungsfeld. Tensoren finden im Bereich der Signalverarbeitung, der Neurowissenschaften, der Datenanalyse oder im maschinellen Lernen Anwendung, [10], [15]. Daten werden häufig in Form von Skalaren, Vektoren oder Matrizen dargestellt. Oftmals haben die Daten jedoch eine höherdimensionale Struktur, die durch Skalare, Vektoren oder Matrizen nicht wiedergegeben werden können. Daher ist es sinnvoll diese Konzepte zu generalisieren und solche Daten in höherdimensionalen Arrays, d.h. Tensoren zu speichern. Im Folgenden werden einige Definitionen von Tensoren und Tensoroperationen eingeführt, wie sie in z.B. in [21] oder [9] zu finden sind.

Definition 6.1.2 (Tensor) *Ein Tensor*

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

der Ordnung n ist ein n-dimensionales Array. Die Elemente $x(i_1, i_2, \ldots, i_n)$ werden durch $i_j \in \{1, 2, \ldots, I_j\}$ in jeder Dimension $j = 1, \ldots, n$ indiziert.

Beispiel 6.1.3 *Ein 3-dimensionaler Tensor $\mathsf{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ mit $I_1 = 6$, $I_2 = 8$ und $I_3 = 5$ ist in Abbildung 6.1-6 dargestellt, [9].*

Für die Beschreibung von Tensoren wird hier eine an MATLAB angelehnte Notation verwendet.

Tensoroperationen

Aus der linearen Algebra sind für Vektoren und Matrizen Operationen, wie die Matrix Multiplikation oder das innere Produkt definiert. Um auch mit Daten im Tensorformat Berechnungen durchführen zu können, müssen solche oder vergleichbare Operationen auch für Tensoren definiert werden. Die Definitionen der hier im folgenden verwendeten Operationen werden kurz zusammengefasst und können detaillierte in der Fachliteratur wie [21] oder [9] gefunden werden. Auf den ersten Blick wirken diese Operationen aufgrund der Multiindex-Struktur deutlich komplexer als im Matrix Fall. Aber sie basieren genauso auf Standardoperationen wie der Multiplikation und Addition.

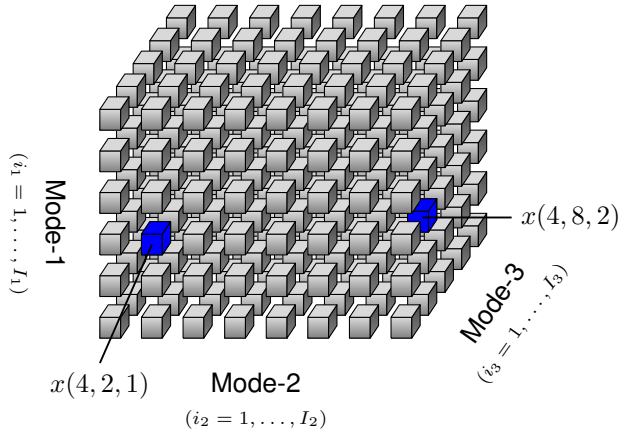


Abbildung 6.1-6: Elemente eines 3-dimensionalen Tensors $\mathsf{X} \in \mathbb{R}^{6\times 8\times 5}$, [9]

Definition 6.1.3 (Äußeres Produkt) *Das äußere Produkt*

$$\mathsf{Z} = \mathsf{X} \circ \mathsf{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N \times J_1 \times J_2 \times \cdots \times J_M}, \tag{6.1-12}$$

zweier Tensoren $\mathsf{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ *und* $\mathsf{Y} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_M}$ *ist ein Tensor der Ordnung* $N + M$ *mit Elementen*

$$z(i_1, \ldots, i_N, j_1, \ldots, j_M) = x(i_1, \ldots, i_N) y(j_1, \ldots, j_M). \tag{6.1-13}$$

Da das äußere Produkt assoziativ ist, kann es nacheinander angewendet werden durch

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

Für die Multiplikation von Tensoren und Matrizen oder Vektoren wird ein spezielles Produkt definiert, das die übliche Matrix-Matrix oder Matrix-Vektor Multiplikation in einer Mode des Tensors durchführt.

Definition 6.1.4 (Mode-k Produkt) *Das Mode-k Produkt eines Tensors* $\mathsf{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ *und einer Matrix* $\mathbf{W} \in \mathbb{R}^{J \times I_k}$ *ist ein Tensor*

$$\mathsf{Y} = \mathsf{X} \times_k \mathbf{W} \in \mathbb{R}^{I_1 \times \cdots \times I_{k-1} \times J \times I_{k+1} \times \cdots \times I_N}. \tag{6.1-14}$$

Der resultierende Tensor ist elementweise gegeben durch

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

durch die Multiplikation von den Fibern $\mathbf{x}(i_1, \ldots, i_{k-1}, :, i_{k+1}, \ldots, i_N)$ *von* X *mit der Matrix* $\mathbf{W}$

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

Unter anderem für die Darstellung der rechten Seiten von MTI Systemen im Tensorformat ist das kontrahierte Produkt ein wichtiges Produkt zweier Tensoren.

Definition 6.1.5 (Kontrahiertes Produkt) *Das kontrahierte Produkt entlang der ersten* N *Dimensio-*

nen zweier Tensoren $\mathsf{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N \times I_{N+1} \times \cdots \times I_{N+M}}$ *und* $\mathsf{Y} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$

$$\begin{aligned} z(k_1, \ldots, k_m) &= \langle \mathsf{X} \mid \mathsf{Y} \rangle_{1,\ldots,N;1,\ldots,N} (k_1, \ldots, k_m) \\ &= \sum_{i_1=1}^{I_1} \cdots \sum_{i_n=1}^{I_n} x(i_1, \ldots, i_n, k_1, \ldots, k_m) y(i_1 \ldots i_n), \end{aligned} \tag{6.1-15}$$

mit $k_i \in \{1, 2, \ldots, I_{N+i}\}$*,* $i = 1, \ldots, M$*, ist ein Tensor* Z *der Dimension* $\mathbb{R}^{I_{N+1} \times \cdots \times I_{N+M}}$ *.*

Das kontrahierte Produkt kann entlang beliebiger Dimensionen beider Tensoren berechnet werden. Es ist jedoch notwendig, dass die Dimensionen jeweils die gleich Größe haben.

Beispiel 6.1.4 *Das kontrahierte Produkt zweier Tensoren* $\mathsf{X} \in \mathbb{R}^{5 \times 2 \times 3}$ *und* $\mathsf{Y} \in \mathbb{R}^{2 \times 4 \times 3 \times 5}$ *jeweils entlang der Moden* $1, 2$ *und* $4, 1$ *ergibt einen Tensor* $\langle \mathsf{X} \mid \mathsf{Y} \rangle_{1,2;4,1} \in \mathbb{R}^{3 \times 4 \times 3}$*.*

Zur Vereinfachung der Notation wird im Folgenden auf die Angabe der Dimensionen entlang welcher das kontrahierte Produkt berechnet wird, verzichtet, wenn es sich dabei um die ersten N Moden der jeweiligen Tensoren handelt

$$\langle \mathsf{X} \mid \mathsf{Y} \rangle_{1,\ldots,N;1,\ldots,N} = \langle \mathsf{X} \mid \mathsf{Y} \rangle \tag{6.1-16}$$

wie in (6.1-15). Zur weiteren Vereinfachung der Notation erhält das kontrahierte Produkt von zwei Tensoren in der letzten Dimension des ersten Tensors und der ersten Dimension des zweiten ein spezielles Symbol, [26].

Definition 6.1.6 (Mode-$(M,1)$ kontrahiertes Produkt) *Das Mode-(M,1) kontrahierte Produkt zweier Tensoren* $\mathsf{X} \in \mathbb{R}^{I_1 \times \cdots \times I_M}$ *und* $\mathsf{Y} \in \mathbb{R}^{I_M \times J_2 \times \cdots \times J_N}$ *ist ein Tensor*

$$\mathsf{Z} = \langle \mathsf{X} \mid \mathsf{Y} \rangle_{M;1} = \mathsf{X} \bullet \mathsf{Y} \in \mathbb{R}^{I_1 \times \cdots \times I_{M-1} \times J_2 \times \cdots \times J_N}, \tag{6.1-17}$$

mit Elementen

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

Tensordekompositionen

Für Tensoren mit einer großen Anzahl an Dimensionen ergibt sich ein sehr hoher Speicherbedarf, da die Anzahl der zu speichernden Elemente eines Tensor exponentiell mit Anzahl an Dimensionen ansteigt. Für einen Tensor K der Dimension $\mathbb{R}^{\times^{(N)} I}$ ergibt sich ein Speicheraufwand von

$$\xi_{Full}(\mathsf{K}) = I^N, \tag{6.1-18}$$

wobei mit der Notation $\mathbb{R}^{\times^{(N)} 2}$ der Raum $\mathbb{R}^{\overbrace{2 \times \ldots \times 2}^{N \text{ mal}}}$ bezeichnet wird. In den letzten Jahrzehnten wurden in der Mathematik einige Dekompositionsmethoden entwickelt, um den Speicheraufwand von Tensoren zu reduzieren. Vier der am häufigsten verwendeten Methoden sind die Canonic Polyadic (CP), die Tucker, die Tensor Train (TT) und die Hierarchical Tucker (HT) Zerlegungen, [15]. Im Folgenden werden diese Methoden kurz anhand des N-dimensionalen Tensors $\mathsf{K} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ vorgestellt.

CP Dekomposition
Bei der CP Zerlegung wird ein Tensor in eine Summe von Rang-1 Elementen faktorisiert.

Definition 6.1.7 (Rang-1 Tensor) *Ein Tensor* $\mathsf{K} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ *der Ordnung* N *ist ein Rang-*1 *Tensor,*

wenn er durch das äußere Produkt

$$\begin{aligned} \mathsf{X} &= \mathbf{x}_1 \circ \mathbf{x}_2 \circ \cdots \circ \mathbf{x}_N \\ &= \bigcirc_{i=1}^{N} \mathbf{x}_i. \end{aligned} \tag{6.1-19}$$

von N Vektoren $\mathbf{x}_i \in \mathbb{R}^{I_i}$ berechnet werden kann.

Beispiel 6.1.5 *Ein Rang-1 Tensor $\mathsf{K} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ dritter Ordnung ist in Abbildung 6.1-7 als äußeres Produkt von drei Vektoren $\mathbf{x}_1 \in \mathbb{R}^{I_1}$, $\mathbf{x}_2 \in \mathbb{R}^{I_2}$ und $\mathbf{x}_3 \in \mathbb{R}^{I_3}$ dargestellt.*

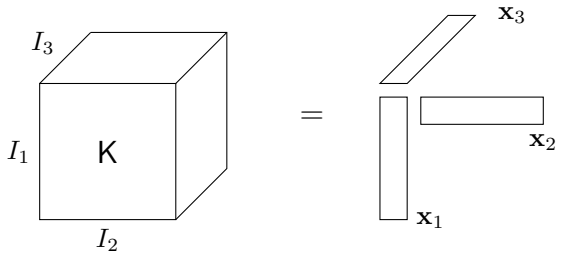


Abbildung 6.1-7: Dreidimensionaler Rang-1 Tensor

Definition 6.1.8 (CP Tensor) *Ein CP Tensor der Dimension $I_1 \times \cdots \times I_N$ ist gegeben durch*

$$\begin{aligned} \mathsf{K} &= [\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N] \cdot \boldsymbol{\lambda} \\ &= \sum_{k=1}^{r_{CP,K}} \lambda(k) \mathbf{x}_1(:,k) \circ \cdots \circ \mathbf{x}_N(:,k), \end{aligned} \tag{6.1-20}$$

wobei die Elemente durch die Summe der äußeren Produkte der Spaltenvektoren der Faktormatrizen $\mathbf{X}_i \in \mathbb{R}^{I_i \times r_{CP,K}}$ gewichtet mit den Elementen des Wichtungs- oder Paramtervektors $\boldsymbol{\lambda} \in \mathbb{R}^{r_{CP,K}}$ berechnet werden,[17]. Die k-te Spalte der Matrix $\mathbf{X}_i$ wird mit $\mathbf{x}_i(:,k)$ bezeichnet. Ein Element des Tensors K ergibt sich aus

$$k(i_1, \ldots, i_n) = \sum_{k=1}^{r_{CP,K}} \lambda(k) x_1(i_1,k) \cdots x_n(i_n,k), \tag{6.1-21}$$

mit den Einträgen des Wichtungsvektors $\lambda(k)$ und der Faktormatrizen $x_j(i_j,k)$ mit $j = 1, \ldots, n$ und $k = 1, \ldots, r_{CP,K}$.

Beispiel 6.1.6 *Ein CP Tensor der Ordnung 3 ist gegeben durch*

$$\begin{aligned} \mathsf{K} &= [\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3] \cdot \boldsymbol{\lambda} \\ &= \sum_{k=1}^{r_{CP,K}} \lambda(k) \mathbf{x}_1(:,k) \circ \mathbf{x}_2(:,k) \circ \mathbf{x}_3(:,k). \end{aligned}$$

Abbildung 6.1-8 zeigt den Tensor als Summe von äußeren Produkten der Spaltenvektoren $\mathbf{x}_i(:,k)$ der Faktormatrizen $\mathbf{X}_i$. Der Tensor setzt sich aus einer Summe von $r_{CP,K}$ Rang-1 Komponenten zusammen.

Tucker Dekomposition

Die zweite Dekompositionsmethode ist die Tucker Dekomposition, bei der der Tensor in N Faktormatrizen und einen Coretensor zerlegt wird.

Definition 6.1.9 (Tucker Tensor) *Ein Tucker Tensor ist gegeben durch*

$$\mathsf{K} = [\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n] \cdot \Lambda, \tag{6.1-22}$$

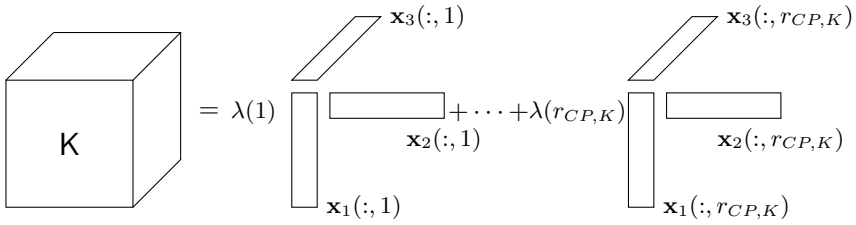


Abbildung 6.1-8: CP Tensor dritter Ordnung

mit einem Coretensor $\Lambda \in \mathbb{R}^{r_{T,K,1} \times \cdots \times r_{T,K,N}}$ *mit Rängen* $r_{T,K,i},\ i = 1, \ldots, N$ *und Faktormatrizen* $\mathbf{X}_i \in \mathbb{R}^{I_i \times r_{T,K,i}}$. *Der volle Tensor wird aus der Tucker Darstellung berechnet, indem der Coretensor entlang jeder Mode mit einer Faktormatrix multipliziert wird*

$$\begin{aligned} \mathsf{K} &= \Lambda \times_1 \mathbf{X}_1 \times_2 \mathbf{X}_2 \times_3 \cdots \times_N \mathbf{X}_N \\ &= \sum_{j_1=1}^{r_{T,K,1}} \cdots \sum_{j_N=1}^{r_{T,K,N}} \lambda(j_1, \ldots, j_N) \mathbf{x}(:, j_1) \circ \cdots \circ \mathbf{x}(:, j_N). \end{aligned} \tag{6.1-23}$$

Ein Element von K *ergibt sich aus dem Coretensor und den Faktormatrizen durch*

$$k(i_1, \ldots, i_N) = \sum_{j_1=1}^{r_{T,K,1}} \cdots \sum_{j_N=1}^{r_{T,K,N}} \lambda(j_1, \ldots, j_N) \mathbf{x}(i_1, j_1) \cdots x(i_N, j_N), \tag{6.1-24}$$

mit $i_l = 1, \ldots, I_l\ \forall l = 1, \ldots, N$, *[49].*

Beispiel 6.1.7 *Ein dreidimensionaler Tucker Tensor wird beschrieben durch*

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

Der Tucker Tensor wird durch Tensor Matrix Produkte des Coretensors und einer Faktormatrix in jeder Dimension berechnet, wie in Abbildung 6.1-9 gezeigt.

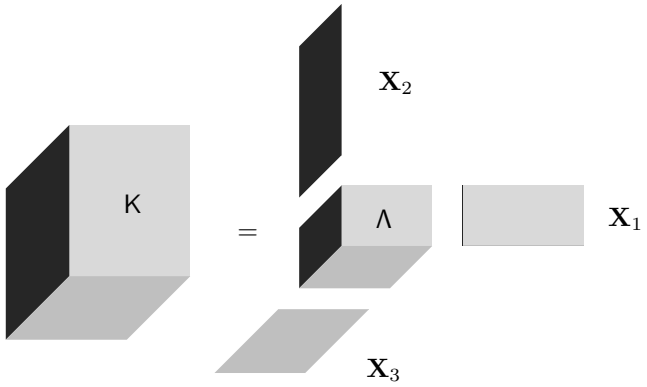


Abbildung 6.1-9: Tucker Tensor dritter Ordnung, [28]

TT Dekomposition
Anders als bei den vorangegangenen Zerlegungen wird der Ausgangstensor bei der TT Dekomposition durch ein Produkt von N Tensoren dritter Ordnung approximiert, [39].

Definition 6.1.10 (Tensor Train) *Ein Tensor* K *im TT Format*

$$\mathsf{K} = [\mathsf{G}_1, \mathsf{G}_2, \ldots, \mathsf{G}_{N-1}, \mathsf{G}_N], \tag{6.1-25}$$

mit dreidimensionalen TT Tensorkernen $\mathsf{G}_j \in \mathbb{R}^{r_{TT,K,j-1} \times I_j \times r_{TT,K,j}},\ j = 1, \ldots, N$ *und TT-Rängen* $r_{TT,K,j}$ *ist gegeben durch*

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

Durch diese Konstruktion sind der erste und der letzte Rang gleich Eins $r_{TT,K,0} = r_{TT,K,N} = 1$*, sodass der erste und letzte Tensorkern durch Matrizen gegeben sind. Ein Element des Tensors wird berechnet durch*

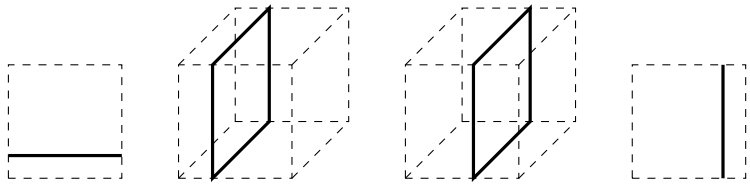
$$\begin{aligned} k(i_1, \ldots, i_N) &= \mathbf{g}_1(1, i_1, :)\mathbf{G}_2(:, i_2, :) \cdots \mathbf{G}_{N-1}(:, i_{N-1}, :)\mathbf{g}_N(:, i_N, 1) \\ &= \sum_{j_1=1}^{r_{TT,K,1}} \cdots \sum_{j_N=1}^{r_{TT,K,N}} g_1(1, i_1, j_1) g_2(j_1, i_2, j_2) \cdots g_{N-1}(j_{N-2}, i_{N-1}, j_{N-1}) g_N(j_{N-1}, i_N, 1), \end{aligned} \tag{6.1-26}$$

wobei $\mathbf{G}_j(:, i_j, :) \in \mathbb{R}^{R_{j-1} \times R_j}$ *Scheiben des Tensorkerns* G_j *sind, [39].*

Beispiel 6.1.8 *Die elementweise Beschreibung eines Tensors vierter Ordnung im TT Format führt zu*

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

Die Tensorkerne sind durch zwei Tensoren G_1 *und* G_4 *mit jeweils einer Dimension gleich Eins, die als Matrix interpretiert werden können, und zwei dreidimensionalen Tensoren* G_2 *und* G_3 *gegeben. Die Elemente von* K *ergeben sich aus dem Produkt eines Spaltenvektors, zweier Matrizen und einem Zeilenvektor, wie in Abbildung 6.1-10 dargestellt.*



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

Abbildung 6.1-10: Tensor Train vierter Ordnung

HT Dekomposition

Die HT Dekomposition ist eine Generalisierung der TT Zerlegung, [15]. Die Methode basiert darauf, die Moden des Tensors K in einen binären Baum $\mathcal{T}$ aufzuteilen. Jeder Knoten enthält eine Untermenge an Moden $t \subset \{1, 2, \ldots, N\}$, wie beispielhaft in Abbildung 6.1-11 für einen Tensor der Ordnung 6 gezeigt.

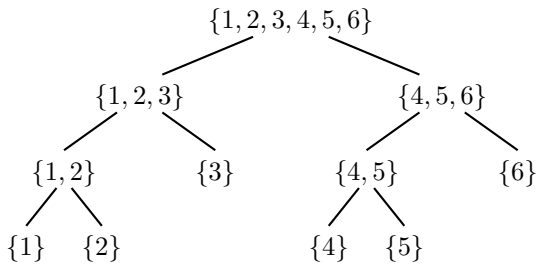


Abbildung 6.1-11: Baum eines Tensors sechster Ordnung

Hier werden balancierte Bäume verwendet mit der Annahme, das der linke Kinderknoten immer kleinere Elemente als der rechte Kinderknoten enthält. Anhand dieser Aufteilung der Moden werden Matrizisierungen $\mathbf{K}^{(t)}$ des Tensors K bestimmt, [14]. Die Basismatrizen $\mathbf{U}_t$ werden bestimmt, sodass ihre Spalten das

Bild der Matrizisierung $\mathbf{K}^{(t)}$ für jedes $t \in \mathcal{T}$ aufspannen. Daher sind

$$r_{HT,K,t} = \text{rank}(\mathbf{K}^{(t)}) \tag{6.1-27}$$

Spalten bei den Basismatrizen für eine exakte Darstellung nötig. Für jeden Elternknoten existiert eine sogenannte Transfermatrix $\mathbf{B}_t \in \mathbb{R}^{r_{HT,K,t_l} r_{HT,K,t_r} \times r_{HT,K,t}}$, sodass die Basismatrix $\mathbf{U}_t$ aus den linken $\mathbf{U}_{t_l}$ und rechten $\mathbf{U}_{t_r}$ Kinderknoten

$$\mathbf{U}_t = (\mathbf{U}_{t_r} \otimes \mathbf{U}_{t_l})\, \mathbf{B}_t, \tag{6.1-28}$$

mit den Rängen $r_{HT,K,t}$, r_{HT,K,t_l} und r_{HT,K,t_r} der jeweiligen Matrizisierungen und $t = t_l \cup t_r,\ t_l \cap t_r = \emptyset$ berechnet wird. Abbildung 6.1-12 zeigt den entsprechenden Subbaum mit dem Elternknoten und den linken und rechten Kinderknoten.

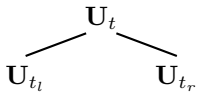


Abbildung 6.1-12: Elternknoten mit linken und rechten Kinderknoten

Für einige Anwendungen ist es vorteilhaft die Transfermatrizen in dreidimensionalen Transfertensoren anzuordnen, d.h.

$$\mathbf{B}_t \in \mathbb{R}^{r_{HT,K,t_l} r_{HT,K,t_r} \times r_{HT,K,t}} \Rightarrow \mathrm{B}_t \in \mathbb{R}^{r_{HT,K,t_l} \times r_{HT,K,t_r} \times r_{HT,K,t}}.$$

In der Tensordarstellung wird der Elternknoten aus Abbildung 6.1-12 analog zu Matrixfall (6.1-28) berechnet durch

$$\mathbf{u}_t(:,q) = \sum_{i=1}^{r_{HT,K,t_l}} \sum_{j=1}^{r_{HT,K,t_r}} (\mathbf{u}_{t_r}(:,j) \otimes \mathbf{u}_{t_l}(:,i))\, b_t(i,j,q),\ q = 1,\ldots,r_{HT,K,t}. \tag{6.1-29}$$

Dies vereinfacht die Notation in einigen Fällen, [22]. In der Definition der HT Zerlegung wird hier die Matrixnotation verwendet. Später für die Darstellung von MTI System führt die Tensornotation zu einer einfacheren Formulierung der Faktoren. Die Aufteilung der Moden wie in Abbildung 6.1-11 beschreibt eine Hierarchie an Matrizen $\mathbf{U}_t$. Aufgrund von (6.1-28) müssen jedoch nicht alle Basismatrizen $(\mathbf{U}_t)_{t\in\mathcal{T}}$ explizit gespeichert werden, da die Matrix $\mathbf{U}_t$ aus ihren linken und rechten Kinderknoten berechnet werden kann. Die rekursive Anwendung von (6.1-28) führt zur HT Dekomposition. Daher müssen nur die Basismatrizen $\mathbf{U}_t \in \mathbb{R}^{I_t \times r_t}$ der Blattknoten $t = \{1\}, \ldots, \{N\}$ und die Transfermatrizen $\mathbf{B}_t$ aller Knoten gespeichert werden, um die Elemente des vollen Tensors bestimmen zu können. Die zu speichernden Elemente sind in Abbildung 6.1-13 gezeigt.

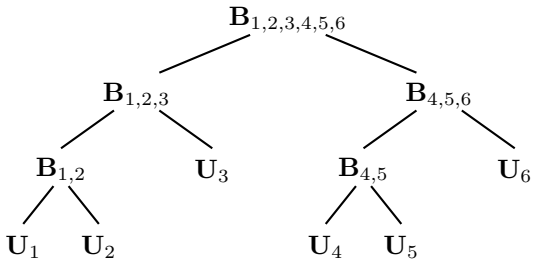


Abbildung 6.1-13: Baum eines Tensors der Ordnung 6 mit Basis- und Transfermatrizen

Beispiel 6.1.9 *Ein Baum einer HT Dekomposition eines Tensors dritter Ordnung ist in Abbildung 6.1-14 dargestellt.*

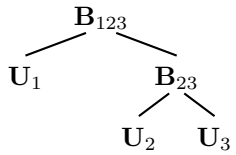


Abbildung 6.1-14: Baum eines Tensors dritter Ordnung

Die rekursive Anwendung von (6.1-28) führt zu der vektorisierten Rekonstruktion von K

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

wobei $\mathbf{I}$ *eine Einheitsmatrix von geeigneter Dimension ist.*

6.1.3 Dekomponierte multilineare Modelle

Im vorangegangenen Kapitel wurde die Zustandsraumdarstellung für MTI Systeme (6.1-9) und (6.1-10) in Matrixnotation vorgestellt. Die rechten Seiten des Zustandsraummodells berechnen sich aus dem Produkt einer Parametermatrix und einem Monomvektor. Neben dem Matrixformat können multilineare Funktionen auch mithilfe von Tensoren dargestellt werden, wie in [28] oder [41] eingeführt. Dazu werden die multilnearen Monome statt in einem Vektor in einem Tensor angeordnet, der durch eine Folge von äußeren Produkten berechnet wird. Für eine Funktion in n Variablen $\mathbf{x} \in \mathbb{R}^n$ ergibt sich ein Tensor

$$\begin{aligned} \mathsf{M}(\mathbf{x}) &= \begin{pmatrix} 1 \\ x_n \end{pmatrix} \circ \begin{pmatrix} 1 \\ x_{n-1} \end{pmatrix} \circ \cdots \circ \begin{pmatrix} 1 \\ x_1 \end{pmatrix} \\ &= \bigcirc_{i=1}^{n} \begin{pmatrix} 1 \\ x_{n-i+1} \end{pmatrix}, \end{aligned} \tag{6.1-30}$$

der Dimension $\mathbb{R}^{\times^{(n)}2}$. Da er aus einer Folge von äußeren Produkten von Vektoren berechnet wird, ist der Monometensor ein Rang-1 Tensor

$$\mathsf{M}(\mathbf{x}) = \left[\begin{pmatrix} 1 \\ x_n \end{pmatrix}, \ldots, \begin{pmatrix} 1 \\ x_1 \end{pmatrix} \right]. \tag{6.1-31}$$

Beispiel 6.1.10 *Abbildung 6.1-15 zeigt, wie die multilinearen Monome einer Funktion in 3 Variablen* x_1*,* x_2 *und* x_3 *in dem Monomtensor* $\mathsf{M}(x_1, x_2, x_3) \in \mathbb{R}^{2\times2\times2}$ *angeordnet sind.*

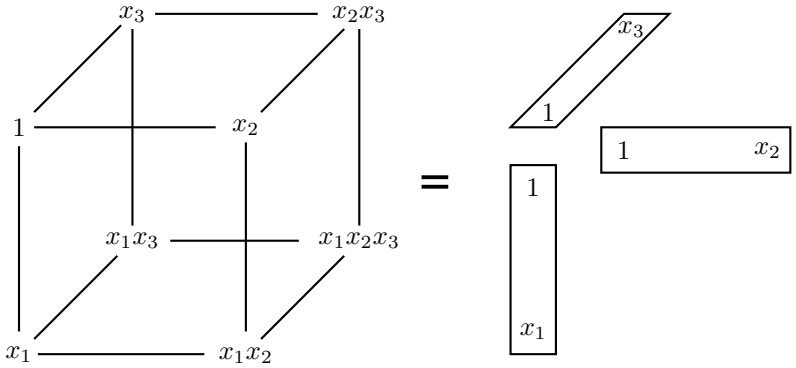


Abbildung 6.1-15: Monomtensor $\mathsf{M}(x_1, x_2, x_3)$ einer Funktion in 3 Variablen

Ordnet man die Parameter der multilinearen Funktion in der gleichen Weise in einem Tensor an, erhält man die Tensordarstellung einer multilinearen Funktion in n Variablen

$$h(\mathbf{x}) = \langle\, \mathsf{H} \mid \mathsf{M}\,(\mathbf{x}) \,\rangle \ , \tag{6.1-32}$$

aus dem kontrahierten Produkt des Parametertensors $\mathsf{H} \in \mathbb{R}^{\times^{(n)}2}$ und des Monomtensors (6.1-31).

Beispiel 6.1.11 *Die Tensordarstellung einer multilinearen Funktion*

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

in 3 *Variablen* x_1*,* x_2*und* x_3 *ist in Abbildung 6.1-16 dargestellt.*

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

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

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

Abbildung 6.1-16: Multilineares Polynom in Tensordarstellung

Das Beispiel verdeutlicht die Aufteilung von Parametern und Variablen in zwei verschiedene Tensoren. Die Berechnung des kontrahierten Produkts ergibt das gegebene Polynom, was die Korrektheit der Tensordarstellung zeigt.

Die Tensordarstellung soll auch auf die MTI Systeme angewendet werden. Dadurch ergibt sich die Möglichkeit Methoden zur Reduzierung des Speicheraufwands in Form von Tensordekompositionsverfahren anzuwenden, um z.B. die Komplexität der Darstellung von sehr großen Modellen zu reduzieren. Der Monomtensor (6.1-31) der rechten Seiten hängt von den Eingängen und Zuständen ab

$$\begin{aligned} \mathsf{M}\,(\mathbf{x}, \mathbf{u}) &= \begin{pmatrix} 1 \\ u_m \end{pmatrix} \circ \begin{pmatrix} 1 \\ u_{m-1} \end{pmatrix} \circ \cdots \circ \begin{pmatrix} 1 \\ u_1 \end{pmatrix} \circ \begin{pmatrix} 1 \\ x_n \end{pmatrix} \circ \begin{pmatrix} 1 \\ x_{n-1} \end{pmatrix} \circ \cdots \circ \begin{pmatrix} 1 \\ x_1 \end{pmatrix} \\ &= \left[\begin{pmatrix} 1 \\ u_m \end{pmatrix}, \ldots, \begin{pmatrix} 1 \\ u_1 \end{pmatrix}, \begin{pmatrix} 1 \\ x_n \end{pmatrix}, \ldots, \begin{pmatrix} 1 \\ x_1 \end{pmatrix} \right] \in \mathbb{R}^{\times^{(n+m)}2}. \end{aligned} \tag{6.1-33}$$

Somit ergibt sich die Zustandsraumdarstellung eines MTI Systems in Tensordarstellung zu

$$\Phi\,(\mathbf{x}) = \langle\, \mathsf{F} \mid \mathsf{M}\,(\mathbf{x}, \mathbf{u}) \,\rangle\,, \tag{6.1-34}$$

$$\mathbf{y} = \langle\, \mathsf{G} \mid \mathsf{M}\,(\mathbf{x}, \mathbf{u}) \,\rangle\,, \tag{6.1-35}$$

mit den Parametertensoren $\mathsf{F} \in \mathbb{R}^{\times^{(n+m)}2\times n}$ und $\mathsf{G} \in \mathbb{R}^{\times^{(n+m)}2\times p}$, die die gleichen Parameter enthalten wie die Parametermatrizen $\mathbf{F}$ und $\mathbf{G}$, hier allerdings in einer multidimensionalen Struktur angeordnet sind.

Beispiel 6.1.12 *Die Zustandsgleichung eines MTI Systems mit zwei Zuständen ist in dem Tensorformat gegeben durch*

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

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

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

In den heutigen Anwendungen werden die betrachteten Systeme immer komplexer, wie im Bereich der Smart Grids oder der Heizungssysteme. Solche Anlagen können mithilfe von MTI Systemen modelliert werden. Mit dicht besetzten Parametertensoren lassen sich sehr komplexe Dynamiken abgebildet werden. Jedoch führt eine hohe Ordnung n oder viele Eingangsgrößen m zu einer großen Anzahl an Parametern in voller Darstellung. Die Anzahl an Elementen in den Tensoren F und G steigt exponentiell mit der Anzahl an Zuständen und Eingängen. So sind z.B. $n \cdot 2^{n+m}$ Elemente für den Parametertensor F nötig. Daher ist eine Systembeschreibung für große Systeme in der vorgestellten Weise nicht möglich, da sie zu Speicherproblemen führt. Die Abbildung 6.1-17 verdeutlicht dies, indem sie die zu speichernden Elemente logarithmisch über der Anzahl der Zustände und Eingänge darstellt.

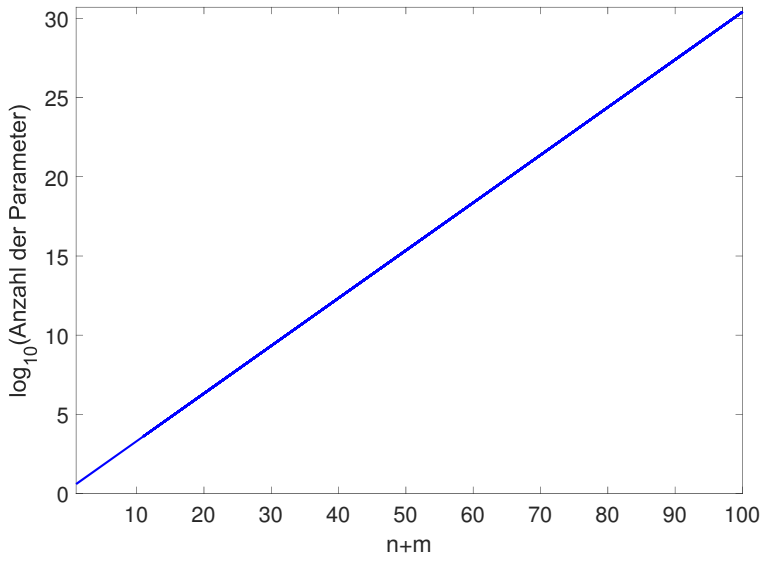


Abbildung 6.1-17: Anzahl der Parameter für ein MTI System in voller Darstellung

Hier zeigt sich deutlich, dass z.B. für ein System mit 100 Eingängen und Zuständen ca. 10^{32} Elemente gespeichert werden müssten, was das zur Verfügung stehenden Speicherangebot deutlich überschreitet. Die Darstellung und auch der modellbasierte Reglerentwurf sind somit in der Form nicht möglich. Um diesen Fluch der Dimensionalität zu durchbrechen, wurden in [28] oder [40] Tensordekompositionsverfahren angewendet um den Aufwand zu reduzieren. Approximationen der Parametertensoren mit niedrigem Rang können den Speicheraufwand um mehrere Größenordnungen verringern. Aus der Mathematik sind

verschiedene Tensordekompositionsverfahren verfügbar. Es soll in Folgenden untersucht werden, wie sich die einzelnen Verfahren für die Darstellung von MTI Systemen eigenen, um diese dann auch im Weiteren für den Reglerentwurf für große Systeme verwenden zu können.

Der Monomtensor liegt bereits in zerlegter Form als Rang-1 Tensor vor. Um auch die Parametertensoren F und G für große Systeme handhabbar zu machen, werden hier die 4 vorgestellten Dekompositionsverfahren, die CP, Tucker, TT und HT Zerlegung auf die Anwendbarkeit auf die Parametertensoren von MTI Systemen untersucht. Im Folgenden wird die Zerlegung des Paramtertensors F der Zustandsübergangsfunktion betrachtet. Die Zerlegung des Tensors G der Ausgangsfunktion kann analog erfolgen und wird daher nicht gesondert dargestellt. Dabei wird zum einen die effiziente Systemdarstellung, als auch die Auswertung der rechten Seite auf Basis der dekomponierten Darstellung betrachtet, wie sie bei der Simulation durchgeführt werden muss. Für die Integration zur Lösung der Differentialgleichungen während der Simulation werden Standardverfahren verwendet, [4].

CP Darstellung

Aus [28] and [40] ist bekannt, dass die Parameter eines MTI Systems direkt Term für Term in eine CP Darstelltung von F übersetzt werden können, wie das folgende Beispiel zeigt.

Beispiel 6.1.13 *Ein MTI System zweiter Ordnung mit einem Eingang*

$$\Phi(x_1) = a_1 + a_2 x_1 + a_3 u_1 x_1, \tag{6.1-36}$$
$$\Phi(x_2) = b_1 x_2 + b_2 x_2 x_1 + b_3 u_1 x_2 x_1, \tag{6.1-37}$$

hat einen CP Parametertensor $\mathsf{F} = [\mathbf{F}_{u_1}, \mathbf{F}_{x_2}, \mathbf{F}_{x_1}.\mathbf{F}_\Phi] \cdot \boldsymbol{\lambda}_F$*. Die Konstruktion der Faktormatrizen aus den Zustandsgleichungen erfolgt spaltenweise nach*

$$\begin{array}{rcccccccccccc}
\Phi(x_1) = & a_1 & + & a_2 x_1 & + & a_3 u_1 x_1, & & & & & & \\
\Phi(x_2) = & & & & & & b_1 x_2 & + & b_2 x_2 x_1 & + & b_3 u_1 x_2 x_1, &
\end{array}$$

$$\mathbf{F}_{x_1} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix},$$

$$\mathbf{F}_{x_2} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix},$$

$$\mathbf{F}_{u_1} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix},$$

$$\mathbf{F}_\Phi = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix},$$

$$\boldsymbol{\lambda}_F = \begin{pmatrix} a_1 & a_2 & a_3 & b_1 & b_2 & b_3 \end{pmatrix}.$$

Wie aus dem Beispiel ersichtlich wird, hat der CP Parametertensor eine Faktormatrix für jeden Zustand $\mathbf{F}_{x_i},\ i = 1,\ldots,n$ und Eingang $\mathbf{F}_{u_j},\ j = 1,\ldots,m$. Die Faktormatrizen werden spaltenweise aufgebaut, wie in Beispiel 6.1.13 dargestellt. Jeder Koeffizient a_i oder b_i wird durch eine Spalte der Faktormatrix abgebildet. Die Anzahl an Termen in der Zustandsgleichung ist gleich der Anzahl der Nichtnull-Elemente in F

$$\Psi_F = \mathrm{card}(\mathsf{F}), \tag{6.1-38}$$

wobei $\mathrm{card}(\cdot)$ die Kardinalitätsfunktion ist, die die Anzahl an Nichtnull-Elementen eines Tensors wiedergibt. Da jeder Term eine Spalte in der Faktormatrix erzeugt, hat jede Faktormatrix Ψ_F Spalten. Wenn ein Zustand x_i oder Eingang u_j in dem Term verwendet wird, wird die dazugehörige Spalte von $\mathbf{F}_{x_i}$ oder $\mathbf{F}_{u_j}$ gleich $\begin{pmatrix} 0 & 1 \end{pmatrix}^T$ gesetzt. Ansonsten ist sie gleich $\begin{pmatrix} 1 & 0 \end{pmatrix}^T$. Der dazugehörige Koeffizient wird in dem Wichtungsvektor $\boldsymbol{\lambda}_F$ eingesetzt. Dies führt zu einem Wichtungsvektor der Länge $\mathbb{R}^{\Psi_F}$ und Faktormatrizen $\mathbf{F}_{x_i}$ and $\mathbf{F}_{u_j}$ der Dimension $\mathbb{R}^{2\times\Psi_F}$. Die letzte Faktormatrix $\mathbf{F}_\Phi$ gibt an, zu welcher Zustandsgleichung $\Phi(x_i),\ i = 1,\ldots,n$ der Term gehört, d.h. $\Phi(x_1)$ oder $\Phi(x_2)$ in diesem Beispiel. Da die rechte Seite der Zustandsgleichung aus n skalaren Funktionen besteht hat die Faktormatrix $\mathbf{F}_\Phi$ die

Dimension $\mathbb{R}^{n\times\Psi_F}$. Aufgrund dieser direkten Übersetzung von den Zustandsgleichungen kann jedes MTI System in CP Darstellung durch den Parametertensor

$$\mathsf{F} = [\mathbf{F}_{u_m}, \ldots, \mathbf{F}_{u_1}, \mathbf{F}_{x_n}, \ldots, \mathbf{F}_{x_1}, \mathbf{F}_\Phi] \cdot \boldsymbol{\lambda}_F \tag{6.1-39}$$

beschrieben werden, [40]. Der so erstellte Tensor ist in dekomponierter Darstellung gegeben und bildet die Parameter und somit auch den vollen Tensor exakt ab. Im Gegensatz zu den Dekompositionen, die häufig in der Datenverarbeitung verwendet werden, handelt es sich hierbei durch die direkte Übersetzung nicht um eine Approximation der Daten bzw. Parameter, sondern eine exakte Darstellung. Es ist jedoch nicht sicher, dass dies auch die minimale Darstellung ist. Durch Dekompositionsverfahren kann ausgehend von der direkten Übersetzung eine Zerlegung mit noch weniger Rang-1 Komponenten gesucht werden, um noch effizientere Darstellungen zu finden. Jedoch kann mit (6.1-39) eine obere Grenze für den Speicheraufwand eines MTI Systems in CP Darstellung abgeschätzt werden

$$\begin{aligned}\xi_{CP}(\mathsf{F}) &\leq \Psi_F + (n+m)2\Psi_F + n\Psi_F \\ &= \Psi_F\left(1+3n+2m\right),\end{aligned} \tag{6.1-40}$$

die sich aus der Summe der Größe der Faktormatrizen und des Wichtungsvektors ergibt.

Dadurch, dass der Parametertensor in CP Darstellung angegeben werden kann, ist es möglich, die rechte Seite der Zustandsgleichung effektiv auf Basis der Faktoren der Dekomposition zu berechnen, [28]

$$\dot{\mathbf{x}} = \mathbf{F}_\Phi\left(\boldsymbol{\lambda}_F \circledast \left(\mathbf{F}_{u_m}^T\begin{pmatrix}1\\u_m\end{pmatrix}\right) \circledast \cdots \circledast \left(\mathbf{F}_{u_1}^T\begin{pmatrix}1\\u_1\end{pmatrix}\right) \circledast \left(\mathbf{F}_{x_n}^T\begin{pmatrix}1\\x_n\end{pmatrix}\right) \circledast \cdots \circledast \left(\mathbf{F}_{x_1}^T\begin{pmatrix}1\\x_1\end{pmatrix}\right)\right). \tag{6.1-41}$$

Beispiel 6.1.14 *Mit den Zerlegungsfaktoren aus dem Beispiel 6.1.13 wird die rechte Seite der Zustandsgleichung ausgewertet durch*

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

Das Beispiel zeigt, dass das Ergebnis durch einfache Standard Matrixprodukte berechnet wird. Keine aufwendigen Operationen sind notwendig. Es werden nur die niedrigdimensionalen Faktoren der Zerlegung verwendet und kein voller Tensor. Dadurch ergibt sich ein deutlich geringer Berechnungsaufwand im Vergleich zur Verwendung von vollen Tensoren.

Dieses CP Format hilft bei der Erstellung der Systemdarstellungen mit Tucker, TT oder HT Tensoren.

Tucker Darstellung

Die CP Darstellung kann verwendet werden, um eine Tucker Darstellung des Parametertensors zu erstellen. Ein Tensor in Tucker Darstellung hat eine ähnliche Struktur, wie der CP Tensor. Mit der Verwendung eines superdiagonalen Coretensors mit den Elementen des Wichtungsvektors auf der Diagonalen lässt sich die Tucker Darstellung eines N dimensionalen Tensors K zu einer CP Darstellung

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

vereinfachen. Die Faktormatrizen sind die gleichen. Ist der Parametertensor F des MTI System in CP Darstellung gegeben, lässt sich der Coretensor einer Tucker Darstellung bestimmen, indem die Elemente des Wichtungsvektors $\boldsymbol{\lambda}_F$ in die Diagonale des Coretensors geschrieben werden durch

$$\Lambda_F(i_1,\ldots,i_{n+m+1}) = \begin{cases} \lambda_F(j) & , \forall j = i_1 = \cdots = i_{n+m+1}, \\ 0 & , \text{ sonst}, \end{cases} \tag{6.1-42}$$

mit $j = 1,\ldots,\Psi_F$. Dies führt zu einer Größe von $\mathbb{R}^{\times^{(n+m+1)}\Psi_F}$, die von der Anzahl an Termen Ψ_F in der Zustandsgleichungen abhängt. Die Faktormatrizen können damit wie bei der CP Darstellung gewählt werden, wodurch sich die Tucker Darstellung zu

$$\mathsf{F} = [\mathbf{F}_{u_m},\ldots,\mathbf{F}_{u_1},\mathbf{F}_{x_n},\ldots,\mathbf{F}_{x_1},\mathbf{F}_\Phi]\cdot\Lambda_F. \tag{6.1-43}$$

ergibt. Der Speicheraufwand setzt sich aus der Anzahl an Elementen im Coretensor und der Faktormatrizen zusammen und lässt sich abschätzen durch

$$\begin{aligned} \xi_T(\mathsf{F}) &\leq \Psi_F^{n+m+1} + 2(n+m)\Psi_F + n\Psi_F \\ &= \Psi_F^{n+m+1} + \Psi_F(3n+2m). \end{aligned} \tag{6.1-44}$$

Aufgrund der Größe des Coretensors in dieser direkten Übersetzung ist der Speicheraufwand immer noch sehr groß mit einer exponentiellen Abhängigkeit von der Anzahl der Zustände und Eingänge. Dies lässt sich reduzieren, z.B. indem man den Coretensor als CP Tensor darstellt mit dem Wichtungsvektor $\boldsymbol{\lambda}_F$ und $\Psi_F \times \Psi_F$ Einheitsmatrizen als Faktormatrizen. So ergibt sich eine Grenze für den Speicherbedarf von

$$\xi_T(\mathsf{F}) \leq \Psi_F + (n+m+1)\Psi_F^2 + 2(n+m)\Psi_F + n\Psi_F. \tag{6.1-45}$$

Damit hängt der Speicheraufwand nur noch linear von der Anzahl an Zuständen und Eingängen ab, wodurch die Tucker Darstellung auch für große Systeme möglich wird.

Während der Simulation wird die rechte Seite der Zustandsgleichung auf Basis des Coretensors und der Faktormatrizen ausgewertet durch eine Folge von Mode-k Tensor Vektor Produkten

$$\Phi(\mathbf{x}) = \mathbf{F}_\Phi\left(\Lambda_F \bar{\times}_1\left(\mathbf{F}_{u_m}^T\begin{pmatrix}1\\u_m\end{pmatrix}\right)\bar{\times}_2\cdots\bar{\times}_m\left(\mathbf{F}_{u_1}^T\begin{pmatrix}1\\u_1\end{pmatrix}\right)\bar{\times}_{m+1}\left(\mathbf{F}_{x_n}^T\begin{pmatrix}1\\x_n\end{pmatrix}\right)\bar{\times}_{m+2}\cdots\bar{\times}_{m+n}\left(\mathbf{F}_{x_1}^T\begin{pmatrix}1\\x_1\end{pmatrix}\right)\right) \tag{6.1-46}$$

TT Darstellung

Die dritte Dekompositionsmethode, die hier auf den Parametertensor F eines MTI Systems angewendet werden soll, ist die TT Zerlegung. Ist der Tensor im CP Format bekannt, können die TT Kerne einfach aus den CP Faktoren bestimmt werden, [39]. Dafür müssen die Zeilen der CP Faktormatrizen $\mathbf{F}_{x_i}$ und $\mathbf{F}_{u_i}$ als Diagonalen der dazugehörigen Scheiben der Tensorkerne $\mathsf{F}_{x_i}(:,j_{n+m-i+1},:)$ und $\mathsf{F}_{u_i}(:,j_{m-i+1},:)$

$$\mathbf{F}_{u_m}(1,:,:) = \mathbf{F}_{u_m}, \tag{6.1-47}$$

$$\mathbf{F}_{u_k}(:,j_{m-k+1},:) = \operatorname{diag}\left(\mathbf{f}_{u_k}(j_{m-k+1},:)\right), k = 1,\ldots,m-1, \tag{6.1-48}$$

$$\mathbf{F}_{x_k}(:,j_{n+m-k+1},:) = \operatorname{diag}\left(\mathbf{f}_{x_k}(j_{n+m-k+1},:)\right), k = 1,\ldots,n, \tag{6.1-49}$$

$$\mathsf{F}_\Phi = \mathbf{F}_\Phi^T \circledast \left(\boldsymbol{\lambda}_F \mathbf{1}_n^T\right), \tag{6.1-50}$$

gewählt werden, wobei $\mathbf{1}_n \in \mathbb{R}^{n\times 1}$ einen Vektor voller Einsen bezeichnet. Die Größen der TT Tensorkerne ist abhängig von der Anzahl der Rang-1 Elemente des CP Tensors

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

Damit ergibt sich der Parametertensor im TT Format zu

$$\mathsf{F} = [\mathsf{F}_{u_m}, \ldots, \mathsf{F}_{u_1}, \mathsf{F}_{x_n}, \ldots, \mathsf{F}_{x_1}, \mathsf{F}_\Phi]\,. \tag{6.1-51}$$

Durch diese direkte Übersetzung von der CP in die TT Darstellung, werden die Systemparameter auch von dem TT Format exakt wiedergegeben. Der Speicheraufwand dieser TT Darstellung gibt eine obere Grenze für den Speicheraufwand für das MTI System in TT an. Durch die Anzahl der Elemente der Tensorkerne ergibt er sich zu

$$\begin{aligned}\xi_{TT}(\mathsf{F}) &\leq 2\Psi_F + 2(n+m-1)\Psi_F^2 + n\Psi_F \\ &= (n+2)\Psi_F + 2(n+m-1)\Psi_F^2.\end{aligned} \tag{6.1-52}$$

Die direkte Transformation von der CP in die TT Repräsentation ist nicht optimal hinsichtlich des Speicheraufwands. Daher sollte das TT Dekompositionsverfahren, dass auf der Grundlage der Singulärwertzerlegung arbeitet, angewendet werden, um geeignete Approximationen mit niedrigeren Rängen zu finden, [39].

Um die rechte Seite der Zustandsgleichung während der Simulation auszuwerten, muss das kontrahierte Produkt des Parametertensors F mit dem Rang-1 Monomtensor bestimmt werden. Dies kann ohne das Erzeugen des vollen Tensors auf Basis der Tensorkerne der TT Darstellung erfolgen, indem eine Folge von Mode-2 Tensor-Vektor Produkten

$$\Phi(\mathbf{x}) = \mathsf{F}_\Phi^T \left(\mathsf{F}_{x_1} \bar{\times}_2 \begin{pmatrix}1\\ x_1\end{pmatrix}\right)^T \cdots \left(\mathsf{F}_{x_n} \bar{\times}_2 \begin{pmatrix}1\\ x_n\end{pmatrix}\right)^T \left(\mathsf{F}_{u_1} \bar{\times}_2 \begin{pmatrix}1\\ u_1\end{pmatrix}\right)^T \cdots \left(\mathsf{F}_{u_m} \bar{\times}_2 \begin{pmatrix}1\\ u_m\end{pmatrix}\right)^T, \tag{6.1-53}$$

von den Tensorkernen und den Faktoren des Monomtensors (6.1-33) berechnet wird.

HT Darstellung

Abschließend wird die Darstellung von MTI Modellen im HT Format betrachtet. Wie bei den Tucker und TT Formaten ist es auch für die HT Form möglich, die CP Faktoren in diese Darstellung zu übersetzen, [22]. Dazu muss zunächst ein Baum für den Tensor F erstellt werden. In Abbildung 6.1-18 ist ein Baum für ein System mit zwei Zuständen und einem Eingang gezeigt.

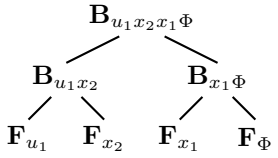


Abbildung 6.1-18: Baum eines Parametertensors F eines Systems mit zwei Zuständen und einem Eingang

Die Matrizen der Blätter $\mathbf{F}_\Phi$, $\mathbf{F}_{x_i}$ mit $i = 1, \ldots, n$ und $\mathbf{F}_{u_i}$ mit $i = 1, \ldots, m$ entsprechen der Faktormatrizen in der CP Darstellung. Die Transfermatrix des oberen Knotens

$$\mathbf{B}_{u_m \cdots u_1 x_n \cdots x_1 \Phi} = \operatorname{diag}(\boldsymbol{\lambda}_F) \tag{6.1-54}$$

ist eine Diagonalmatrix mit den Einträgen des Wichtungsvektors $\boldsymbol{\lambda}_F$ der CP Darstellung auf der Diagonalen. Die Transfertensoren der inneren Knoten sind gegeben durch

$$\mathsf{B}_t = \mathsf{I}_{\Psi_F,3} \in \mathbb{R}^{\Psi_F \times \Psi_F \times \Psi_F}, \tag{6.1-55}$$

wobei $\mathsf{I}_{\Psi_F,3}$ ein dreidimensionaler Diagonaltensor

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

mit Einsen auf der Superdiagonalen ist, [22]. Damit lassen sich alle Komponenten der HT Darstellung aus den CP Faktoren bestimmen. Die Elemente der Blätter und der Transfermatrizen müssen gespeichert werden. Jeder Zustand und Eingang hat ein Blatt. Zudem erhält der Faktor $\mathbf{F}_\Phi$ ein weiteres Blatt, sodass insgesamt $n+m+1$ Blätter nötig sind. Mit der verwendeten Baumstruktur hat ein Tensor der Ordnung N, $N-1$ Transfermatrizen, [22]. Für die Darstellung des Parametertensors eines MTI Systems ergeben sich daher $n+m$ Transfermatrizen bzw. -tensoren. Eine obere Grenze für den Speicherbedarf der HT Darstellung kann daher abgeschätzt werden durch

$$\begin{aligned}\xi_{HT}(\mathsf{F}) &\leq \Psi_F^2 + (n+m-1)\Psi_F^3 + 2(n+m)\Psi_F + n\Psi_F \\ &= \Psi_F^2 + (n+m-1)\Psi_F^3 + (3n+2m)\Psi_F. \end{aligned} \tag{6.1-56}$$

Das direkte Übersetzen der CP Faktoren in die HT Darstellung ist ineffizient hinsichtlich des Speicherbedarfs, da alle Ränge gleich der Anzahl der Rang-1 Komponenten des CP Tensors ist. Zur Reduzierung der Ränge können Dekompositionsmethoden für HT Tensoren verwendet werden, um Approximationen der Parametertensoren zu bestimmen.

Auch für den Fall, dass der Parametertensor in HT Darstellung vorliegt, kann die Auswertung der rechten Seite der Zustandsgleichung effizient auf Basis der Dekompositionsfaktoren erfolgen. Das kontrahierte Produkt von Parameter- und Monomtensor wird durch

$$\Phi(\mathbf{x}) = \mathsf{F} \bar{\times}_1 \begin{pmatrix} 1 \\ u_m \end{pmatrix} \bar{\times}_2 \begin{pmatrix} 1 \\ u_{m-1} \end{pmatrix} \bar{\times}_3 \cdots \bar{\times}_{n+m} \begin{pmatrix} 1 \\ x_1 \end{pmatrix}. \tag{6.1-57}$$

als Sequenz von Mode-k Tensor Vektor Produkten berechnet. Dies lässt sich einfach in der Baumdarstellung abbilden, indem man die Blätter $\mathbf{F}_{u_i}$ und $\mathbf{F}_{x_i}$ von F jeweils durch $\begin{pmatrix} 1 & u_i \end{pmatrix} \mathbf{F}_{u_i}$ und $\begin{pmatrix} 1 & x_i \end{pmatrix} \mathbf{F}_{x_i}$ ersetzt. Für das oben in Abbildung 6.1-18 genannte Beispiel ist diese Darstellung der rechten Seite in Abbildung 6.1-19 gezeigt. Die Zustandsänderung oder der nächste Zustand $\Phi(\mathbf{x})$ wird daraus bestimmt, indem der Baum unter Verwendung von (6.1-28) rekursiv aufgelöst wird.

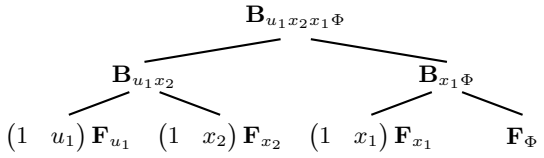


Abbildung 6.1-19: Auswertung der rechten Seite der Zustandsgleichung mit einem HT Tensor

6.1.4 Vergleich der Dekompositionsmethoden

Es wurde gezeigt, dass alle vier vorgestellten Dekompositionsmethoden die Parametertensoren von MTI Systemen exakt darstellen können. Jedoch ist die direkte Ableitung der Zustandsgleichungen in das CP Format oder vom CP Format in die anderen Darstellungen nicht optimal hinsichtlich des Speicheraufwands. Abbildung 6.1-20 fast zusammen, wie die einzelnen Darstellungen erzeugt werden können.

Die Übersetzungen in CP und die anderen dekomponierten Formate machen die Parametertensoren handhabbar in normalen Speichern jedoch sind sie nicht speicheroptimal. Mit den hier entwickelten Darstellungen können jedoch für alle Darstellungsformen obere Grenzen für den Speicherbedarf ermittelt werden. Niedrigrang-Approximationen sollten jedoch Anwendung finden, um den Speicherbedarf weiter zu reduzieren. Dafür sind in MATLAB verschiedene Toolboxen verfügbar, die verschiedene Algorithmen zur Zerlegung zur Verfügung stellen, [2], [50], [38], [22]. Die Dekompositionsalgorithmen für die verschiedenen Darstellungen können in zwei Kategorien eingeordnet werden.

Zum einen ist das Vorgehen bei der CP und der Tucker Dekomposition vergleichbar. Bei diesen Zerlegungen gibt der Anwender eine Anzahl an Rang-1 Komponenten bzw. die Größe des Coretensors vor, mit welcher der ursprüngliche Tensor so gut wie möglich approximiert werden soll. Dafür ist ein nichtlineares Optimierungsproblem zu lösen mit Methoden wie dem Alternating Least-Squares (ALS)

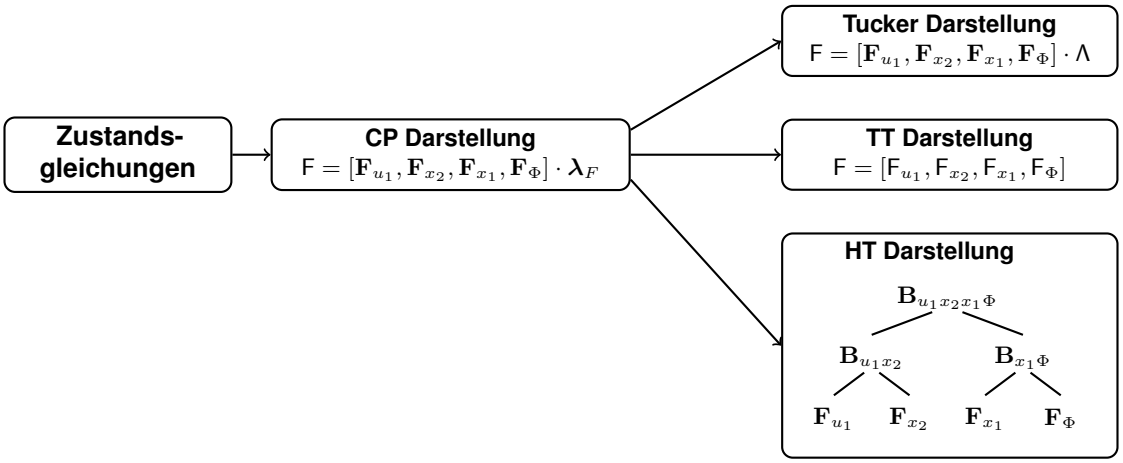


Abbildung 6.1-20: Erstellung der Parametertensoren in verschiedenen Darstellungen

Verfahren, [21]. Der Vorteil hierbei ist, dass durch die Vorgabe eines gewünschten Ranges, der dekomponierte Tensor eine vorhersagbare Größe hat. Allerdings ist es nicht möglich, die Genauigkeit des Ergebnisses vorherzusagen. Zudem ist es nicht möglich den optimialen CP Rang zu bestimmen, da dies ein NP hartes Problem ist, [21]. Die Genauigkeit muss nach der Zerlegung durch Vergleich mit dem Ausgangstensor bestimmt werden. Wenn die Genauigkeit ausreichend ist, kann die Zerlegung weiter verwendet werden. Falls nicht, muss der Rang angepasst und weitere Zerlegungen berechnet werden, bis die Dekompostition den Ansprüchen genügt. Ein weiterer Nachteil bei den CP und Tucker Dekompositionen ist, dass die Optimierungsprobleme, die berechnet werden müssen, nicht konvex sind. Somit ist es nicht garantiert, dass ein globales Minimum gefunden wird. Abhängig vom jeweilgen Startwert kann es sein, dass nur ein lokales Minimum bestimmt wird. Verschiedene Methoden zur Bestimmung guter Startwerte sind vorhanden, können aber auch kein globales Minimum garantieren, [50], [21]. Aufgrund der Dimensionalität der Parametertensoren von sehr großen Systemen ist es nicht möglich diese in voller Darstellung zu speichern, sodass sie direkt in dekomponierter Darstellung beschrieben werden müssen, wie in den vorangegangenen Kapitel erläutert. Wie bereits erwähnt sollte der Speicheraufwand durch Approximationen weiter reduziert werden. Daher ist es notwendig, dass die Dekompositionsalgorithmen auch mit bereits zerlegten Darstellungen arbeiten können, um den Rang zu reduzieren. Die Tucker Zerlegung hat den großen Nachteil, dass kein Algorithmus in den Standard Toolboxen verfügbar ist, mit dem eine weitere Reduktion auf Basis eines bereits zerlegten Tensors verfügbar ist. Es wird immer ein voller Tensor benötigt. Da dies für sehr große Systeme nicht machbar ist, kann der Rang der direkten Übersetzung in die Tucker Darstellung in diesem Fall nicht weiter reduziert werden.

Die zweite Kategorie von Dekompositionsalgorithmen sind durch die TT und HT Zerlegung gegeben. Der Vorteil dieser Zerlegungen ist, dass sie auf der Singulärwertzerlegung (Singular value decomposition, SVD) basieren. Es werden SVDs verschiedener Matrizisierungen des Tensors bestimmt und wenig signifikante Singulärwerte abgeschnitten. Dadurch ergibt sich eine quasi-optimale Lösung bei der Approximation und es ist möglich ein Ergebnis mit einer vorgegebenen Mindestgenauigkeit zu berechnen. Es müssen keine Startwerte oder Ränge vorgegeben werden, wie bei den CP und Tucker Zerlegungen. Der Anwender kann einfach eine gewünschte Genauigkeit der Approximation vorgeben, die sich aus der jeweiligen Anwendung ergibt. Die entsprechenden Ränge der Dekompositionen werden von den Algorithmen berechnet, sodass die bestimmte Zerlegung die Genauigkeitsgrenze erfüllt. Der Nachteil ist, dass nicht a priori vorhersagbar ist, wie viel Reduktion damit erreicht werden kann, d.h. wie viel Speicherplatz für den zerlegten Tensor nötig ist. Dies ist abhängig von den Daten. Im Gegensatz zu der Tucker Zerlegung ist es mit TT und HT Tensoren möglich Zerlegungen ohne Erstellung des vollen Tensors zu bestimmen, sodass diese Zerlegungen auch für sehr große Systeme geeignet sind.

Für die direkte Übersetzung in die vier dekomponierten Darstellungen kann eine obere Grenze für die Anzahl der zu speichernden Elemente durch (6.1-40), (6.1-44), (6.1-45), (6.1-52) und (6.1-56) abgeschätzt werden. Die Zerlegungen sind jeweils ähnlich aufgebaut mit je einem Faktor für jeden Eingang und Ausgang. Zudem ist ein weiterer Faktor $\mathbf{F}_\Phi$ notwendig, der jedoch bei allen Zerlegungen die gleiche Größe hat. In Abbildung 6.1-21 sind daher die oberen Grenzen für den Speicheraufwand der

Darstellungen in den einzelnen Zerlegungen ohne den Faktor $\mathbf{F}_\Phi$ in Abhängigkeit von der Anzahl der Terme der Zustandsgleichungen und der Anzahl der Zustände und Eingänge logarithmisch dargestellt.

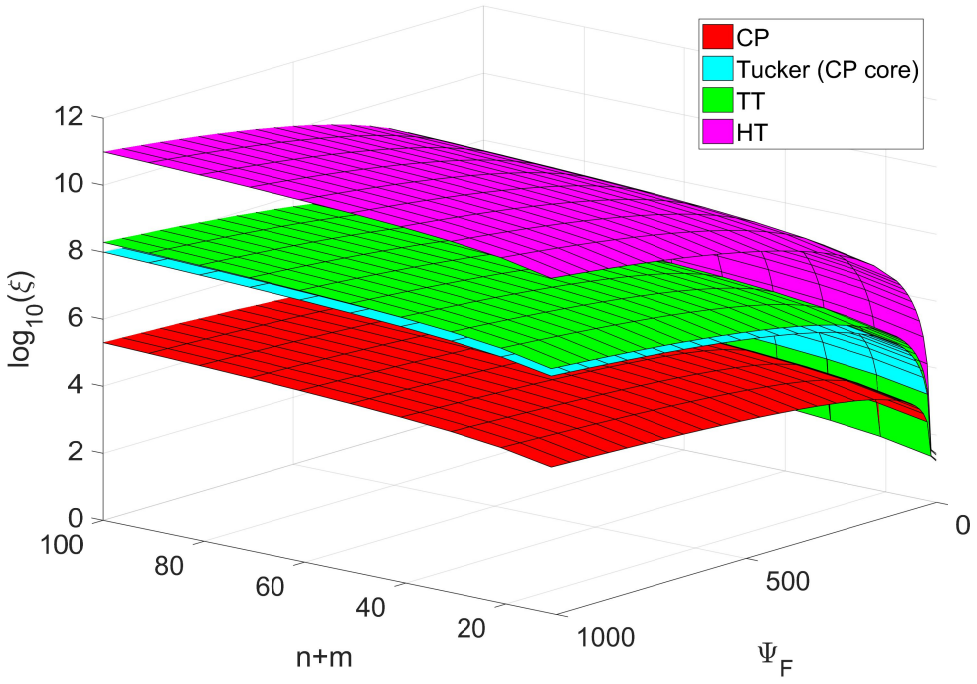


Abbildung 6.1-21: Obere Grenze des Speicherbedarfs der verschiedenen Dekompositionsmethoden

Es wird deutlich, dass die CP Zerlegung den geringsten Speicherbedarf hat, gefolgt von der Tucker Zerlegung mit einem CP Coretensor. Allerdings kann der Speicherbedarf für die Tuckerdarstellung für große Tensoren nicht weiter reduziert werden. Dies ist möglich für die TT und HT Darstellungen, die zunächst jedoch einen größeren Bedarf haben. Aus der Abbildung wird zudem deutlich, dass der Speicheraufwand für die Tensoren aus der direkten Übersetzung in die dekomponierten Formate hauptsächlich von der Anzahl an Termen in den Zustandsgleichungen abhängt.

Es wurde gezeigt, dass bei der Simulation der Systeme in den verschiedenen dekomponierten Darstellungen, die Auswertung der rechten Seite durch das kontrahierte Produkt von Parameter- und Monomtensor für alle vier Dekompositionsmethoden auf Basis der Zerlegungsfaktoren berechnet werden kann. Es muss kein voller Tensor erstellt werden. Für die CP und die HT Darstellung müssen nur Standard Matrix Operationen verwendet werden, die in vielen Programmierumgebungen verfügbar sind. Für die Tucker und TT Darstellungen sind Tensor Vektor oder Tensor Matrix Produkte notwendig. Diese sind in den Tensor Toolboxen wie [2] oder [38] verfügbar oder einfach zu implementieren. Die Komplexität dieser Operationen hängt von der Größe der Faktormatrizen oder Tensorkerne ab. Somit ist es auch für eine effiziente Simulation wichtig, Darstellungen mit niedrigem Rang zu haben, da der Rang die Größe der einzelnen Faktoren beeinflusst.

Um diesen Teil zusammenzufassen, lässt sich sagen, dass MTI Systeme in allen vier betrachteten Dekompositionsmethoden dargestellt werden können, es jedoch deutliche Unterschiede im Speicherbedarf und der Berechnung von Niedrigrang-Approximationen gibt. Die Ergebnisse dieses allgemeinen Vergleichs sind in der Tabelle 6.1-1 zusammengefasst dargestellt.

Tabelle 6.1-1: Vergleich der allgemeinen Eigenschaften der dekomponierten Darstellungen von MTI Systemen (Bewertung beginnt dem Besten:**+** ∘ **−**; ✓: möglich, ×: nicht möglich)

	CP	Tucker	TT	HT
Darstellung von MTI Systemen	✓	✓	✓	✓
Vorhersagbare Größe der Zerlegung	✓	✓	×	×
Approximation mit gewünschter Genauigkeit	×	×	✓	✓
Approximation unabhängig vom Anfangswert	×	×	✓	✓
Approximation auf Basis bereits zerlegter Tensoren	✓	×	✓	✓
Speicherbedarf der direkten Übersetzung	**+**	∘	**−**	**−**
Simulation basierend auf den Faktoren der Zerlegung	✓	✓	✓	✓

6.1.5 Anwendungsbeispiel

Im Abschnitt 6.1.3 wurde gezeigt, dass sich MTI Systeme mit allen vier vorgeschlagenen Dekompositionsverfahren darstellen lassen und diese Darstellungen auch z.B. für die Simulation verwendet werden können. Im Folgenden werden die Darstellungen auf ein Beispiel aus dem Bereich der Heizungstechnik angewendet und hinsichtlich bestimmter Kriterien verglichen. Dafür wird die exakte CP Darstellung als Referenzsystem

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

gewählt, dass wie in (6.1-39) aufgebaut ist. Die anderen Systeme

$$\dot{\tilde{\mathbf{x}}} = \left\langle\, \tilde{\mathsf{F}} \mid \mathsf{M}(\tilde{\mathbf{x}}, \mathbf{u}) \,\right\rangle,$$

d.h. die anderen dekomponierten Parametertensoren und ihre Niedrigrang-Approximationen, werden mit der exakten CP Darstellung verglichen. Die Tensoren werden zum einen dahingehend bewertet, wie gut sie den originalen Parametertensor approximieren

$$\frac{\left\| \mathsf{F} - \tilde{\mathsf{F}} \right\|_F^2}{\|\mathsf{F}\|_F^2}.$$

Neben der reinen Approximationsgenauigkeit der Parameter in den Tensoren F und G wird auch das dynamische Verhalten anhand einer Simulation eines bestimmten typischen Szenarios verglichen

$$\int_0^{T_{end}} \sum_{i=1}^{n} \frac{\left(x_i(t) - \tilde{x}_i(t)\right)^2}{x_i(t)^2} dt.$$

Zudem werden die Eigenwerte einer Linearisierung um einen Arbeitspunkt $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ bestimmt und verglichen. Das Verfahren zur Linearsierung wird in Abschnitt 6.3.2 näher vorgestellt. Der Speicheraufwand ist das letzte Vergleichskriterium.

Das hier betrachtete Beispielsystem ist ein komplexes Modell einer RLT Anlage, das in [48] entworfen und beschrieben wurde. Abbildung 6.1-22 zeigt ein Schema des Systems.

Das Modell wird über Wärme- und Massebilanzen zur Berechnung der Temperaturen und Luftfeuchtigkeiten erstellt und hat eine multilineare Struktur. Die Darstellung als MTI Zustandsraummodell hat 17 Zustände, 10 Eingänge und 54 Parameter. In voller Darstellung würde dies $17 \cdot 2^{27} = 2.3 \cdot 10^9$ Elemente in dem Parametertensor F ergeben. Dies führt zu einem sehr großen Speicherbedarf, sodass Dekompositionsverfahren für die Darstellung dieses Systems angewendet werden sollten. Die Ergebnisse sind in der Tabelle 6.1-2 zusammengefasst.

Zunächst wurden die Zustandsgleichungen als Referenzsystem in das CP Format übersetzt und mithilfe der MTI Toolbox, die im Abschnitt 6.6 vorgestellt wird, dargestellt. Das Referenzsystem wird mit

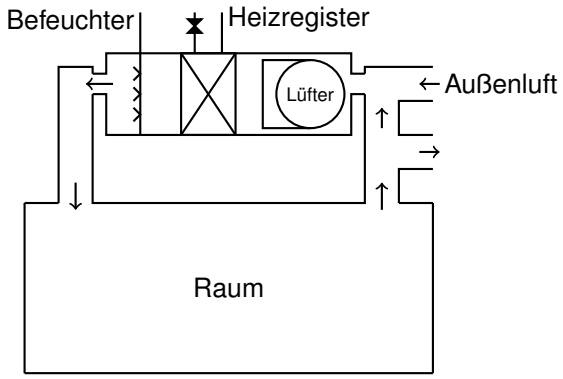


Abbildung 6.1-22: RLT Anlage, [48]

Tabelle 6.1-2: Darstellung des Modells der RLT Anlage in den verschiedenen Formaten (Bewertung beginnt dem Besten:**+** ∘ **−**; ✓: möglich, ×: nicht möglich)

	CP	Tucker	TT	HT
Speicherbedarf der direkten Übersetzung	**+**	∘	**−**	**−**
Speicherbedarf des reduzierten Formats	**+**	**−**	∘	∘
Approximation auf Basis bereits zerlegter Tensoren	×	×	✓	✓
Tensorgenauigkeit	**+**	**+**	**+**	**+**
Simulationsfehler	**+**	**+**	**+**	**+**
Eigenwerte	**+**	**+**	**+**	**+**

den Tucker, TT und HT Übersetzungen verglichen. Zudem werden Approximationsmethoden auf alle Repräsentationen angewendet, um Niedrigrang-Approximationen in den jeweiligen Darstellungen zu finden, um den Speicherbedarf zu reduzieren. Die Ergebnisse bestätigen zum einen den allgemeinen Vergleich aus Tabelle 6.1-1. Eine Darstellung des Systems ist in allen vier Formaten sehr gut möglich. Sowohl bei den Eigenwerten als auch bei den Parametertensoren und den Simulationen sind nur geringe Abweichungen zum Referenzsystem festzustellen. Ein signifikanter Unterschied zwischen den verschiedenen Formaten besteht beim Speicherbedarf, d.h. bei der Anzahl an Elementen, die jeweils gespeichert werden müssen, wie Tabelle 6.1-3 zeigt.

Tabelle 6.1-3: Speicherbedarf von F

	Voll	CP	Tucker	TT	HT
Exakt	$2.3 \cdot 10^9$	3 834	85 482	152 658	4 100 814
Minimal	$2.3 \cdot 10^9$	3 834	85 482	9 779	7 634

Der Speicherbedarf der direkten Übersetzungen zeigt die Notwendigkeit Niedrigrang-Approximationen für die Tucker, TT und HT Darstellungen zu finden, [21], [39], [22]. Für die CP und die Tucker Darstellung war es nicht möglich eine rangreduzierte Darstellung zu finden. Bei der Tucker Darstellung war es nicht möglich, da ein voller Tensor erstellt werden muss, was hier nicht möglich ist. Bei der CP Darstellung konnte keine Darstellung mit weniger Rang-1 Komponenten gefunden werden, die eine noch ausreichende Approximationsgenauigkeit aufweist. Trotz der Tatsache, dass die CP Darstellung nicht weiter reduziert werden konnte, ist der Speicherbedarf bereits sehr niedrig in der exakten Darstellung. Daher ist eine weitere Reduzierung hierbei nicht so wichtig. Niedrigrang-Approximationen im TT und HT Format können hier bestimmt werden, indem eine gewünschte Genauigkeitsgrenze für die Approximation des Parametertensors vorgegeben wird. Somit lässt sich eine einfache Reduzierung des Speicherbedarfs für diese Repräsentationen ohne signifikanten Verlust im dynamischen Systemverhalten erreichen. Dies bestätigt die Resultate aus dem allgemeinen Vergleich aus Abschnitt 6.1.4. Die niedrige Anzahl an zu speichernden Elementen erlaubt eine effiziente Simulation der Systeme und den Entwurf von Reglern,

wenn die Algorithmen mit den dekomponierten Faktoren arbeiten. Der Vergleich an dem Beispiel zeigt, dass CP, TT und HT Tensoren am besten für die Darstellung von MTI Systemen geeignet sind.

6.2 Feedback Linearisierung für MTI Systeme

Modelle von Heizungsanlagen lassen sich sehr gut durch MTI Systeme abbilden, [40]. Die Regelung von nichtlinearen Systemen durch lineare oder nichtlineare Regler sind die Herausforderung von heutigen Anwendungen. Nichtlineare Regelungsansätze führen häufig zu sehr komplexen Reglern oder Entwurfsprozessen, [20]. Einige Ansätze spezialisieren sich auf auf bestimmte Systemklassen, wie bilineare Systeme, [12]. Allgemein führt eine Spezialisierung auf bestimmte Systemklassen zu einer geringeren Komplexität und höheren Robustheit des Designprozesses. Jedoch sind einige Systemklassen zu restriktiv für bestimmte Anwendungen, wie die bilineare für Heizzungssysteme, [40]. Gerade für MTI Systeme, die gut im Bereich der Heizungssysteme anwendbar sind, ist kein spezielles Entwurfsverfahren bekannt. Daher wird im Folgenden das nichtlineare Reglerentwurfsverfahren der Feedback Linearisierung für die Anwendung auf MTI Systeme betrachtet. In Standardanwendungen wird der Zustandsrückführungsregler dabei meist symbolisch berechnet, [19]. Zudem wurden Ansätze untersucht, die eine numerische Berechnung des Reglers ermöglichen durch automatische Differentiation, [43] oder durch die Verwendung von multivariaten Legendre Polynomen, [11]. Die Spezialisierung der Feedback Linearisierung auf Systeme, die gut durch bilineare Modelle approximiert werden können, erfolgte in [36]. Diese Methoden für allgemeine nichtlineare Systeme haben gemeinsam, dass die Struktur der Regelungsfunktion bezüglich der notwendigen mathematischen Operationen beliebig ist und von der jeweiligen Regelstrecke abhängt.

Hier wird eine Methode zur Feedback Linearisierung für MTI Systeme ohne symbolische Berechnungen entwickelt, die zu einem Regler mit fixer Struktur führt, bei der nur die Reglerparameter für die Anwendung auf eine bestimmte Strecke angepasst werden müssen. Die Berechnung des Reglers basiert auf der Tensoralgebra, da MTI Systeme effizient durch dekomponierte Tensoren dargestellt werden können. Der Speicheraufwand für die Reglerparameter, die in dekomponierter Tensorform beschrieben werden können, ist vorhersagbar, was auch eine Anwendung auf sehr große Anlagen ermöglicht.

6.2.1 Feedback Linearisierung

Im folgenden Kapitel wird die Feedback Linearisierung, eine Reglerentwurfsmethode für nichtlineare Systeme, kurz vorgestellt, wie sie in [19] oder [20] eingeführt wurde. Die Idee dieser Methode ist es einen nichtlinearen Regler mit Zustandsrückführung für eine nichtlineare Regelstrecke zu entwerfen, sodass das sich ein lineares Verhalten im geschlossenen Kreis vom Referenzeingang r zum Ausgang y ergibt. Der geschlossene Kreis ist in der Abbildung 6.2-23 dargestellt.

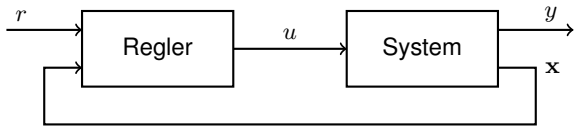


Abbildung 6.2-23: Regelkreis der Feedback Linearisierung

Das Systemverhalten wird durch ein zeitkontinuierliches, nichtlineares, affines, SISO (single input single output) Zustandsraummodell

$$\dot{\mathbf{x}} = \mathbf{a}(\mathbf{x}) + \mathbf{b}(\mathbf{x})u, \tag{6.2-58}$$

$$y = c(\mathbf{x}), \tag{6.2-59}$$

mit nichtlinearen Funktionen $\mathbf{a}, \mathbf{b} : \mathbb{R}^n \to \mathbb{R}^n$ der Zustandsgleichung und Ausgangsfunktion $c : \mathbb{R}^n \to \mathbb{R}$ abgebildet. Es wird angenommen, dass die Funktionen ausreichend glatt sind, sodass alle später auftretenden partiellen Ableitungen definiert und kontinuierlich sind, [20]. Ein nichtlinearer Regler mit

Zustandsrückführung wird entworfen durch, [19]

$$u = k(\mathbf{x}) + v(\mathbf{x})r = \frac{-\sum_{i=0}^{\rho} \mu_i L_{\mathbf{a}}^i c(\mathbf{x}) + \mu_0\, r}{L_{\mathbf{b}} L_{\mathbf{a}}^{\rho-1} c(\mathbf{x})}. \tag{6.2-60}$$

mit dem Referenzeingang r, der Reglerfunktion $k(\mathbf{x})$ und dem Vorfilter $v(\mathbf{x})$, sodass der geschlossene Kreis ein lineares Verhalten von r nach y aufweist, welches durch die lineare Differentialgleichung

$$\mu_\rho \frac{\partial^\rho}{\partial t^\rho} y + \mu_{\rho-1} \frac{\partial^{\rho-1}}{\partial t^{\rho-1}} y + \ldots + \mu_1 \dot{y} + \mu_0 y = \mu_0 r. \tag{6.2-61}$$

gegeben ist. Ohne Beschränkung der Allgemeinheit wird der Faktor μ_ρ auf Eins gesetzt. Die Lie Ableitung einer skalaren Funktion $g(\mathbf{x})$ entlang eines Vektorfelds $\mathbf{h}(\mathbf{x})$ mit $\mathbf{x} \in \mathbb{R}^n$ ist gegeben durch

$$L_{\mathbf{h}} g(\mathbf{x}) = \sum_{i=1}^{n} h_i(\mathbf{x}) \frac{\partial g(\mathbf{x})}{\partial x_i}. \tag{6.2-62}$$

Eine i-fache Anwendung der Lie Ableitung auf eine Funktion wird durch $L_{\mathbf{h}} \cdots L_{\mathbf{h}} g(\mathbf{x}) = L_{\mathbf{h}}^i g(\mathbf{x})$ beschrieben. Der relative Rang oder Index ρ_{sys} des Systems beschreibt die niedrigste Ableitung das Ausgangs y, die direkt von dem Eingang u beeinflusst wird. Daher ist der Index definiert durch

$$\begin{aligned} L_{\mathbf{b}} L_{\mathbf{a}}^i c(\mathbf{x}) &= 0 \;\forall\; \mathbf{x}, \quad 0 \le i \le \rho - 2, \\ L_{\mathbf{b}} L_{\mathbf{a}}^{\rho-1} c(\mathbf{x}) &\neq 0 \;\forall\; \mathbf{x}. \end{aligned}$$

6.2.2 Tensordarstellung von Polynomen

Multilineare Polynome

$$h(\mathbf{x}) = \langle\, \mathsf{H} \mid \mathsf{M}\,(\mathbf{x}) \,\rangle \tag{6.2-63}$$

lassen sich mithilfe von Tensoren darstellen, wie in (6.1-32) vorgestellt. Damit können auch Operationen mit den Polynomen auf Basis dieser Tensorstruktur berechnet werden. Das Ziel ist, aus den Parametertensoren der Operanden den Parametertensor des Ergebnisses von Operationen wie der Multiplikation oder der partiellen Ableitung zu berechnen. Bisher wurde nur die Tensordarstelung von multilinearen Polynomen betrachtet. Da jedoch z.B. bei der Multiplikation von multilinearen Funktionen auch Polynome höherer Ordnung entstehen können, sollen diese auch mithilfe von Tensoren beschrieben werden. Ein Polynom mit maximaler Ordnung N der Monome in n Variablen $\mathbf{x} \in \mathbb{R}^n$ ist gegeben durch

$$h(\mathbf{x}) = \left\langle\, \mathsf{H} \;\middle|\; \mathsf{M}_p^N\,(\mathbf{x}) \,\right\rangle, \tag{6.2-64}$$

mit dem Parametertensor $\mathsf{H} \in \mathbb{R}^{\times^{(nN)}2}$. Der Monomtensor $\mathsf{M}_p^N\,(\mathbf{x})$ enthält alle polynomialen Kombinationen der Variablen bis zur Ordnung N. Analog zu (6.1-30) wird er durch eine Folge von äußeren Produkten

$$\begin{aligned} \mathsf{M}_p^N\,(\mathbf{x}) &= \begin{pmatrix} 1 \\ x_n \end{pmatrix} \circ \cdots \circ \begin{pmatrix} 1 \\ x_1 \end{pmatrix} \circ \cdots\cdots \circ \begin{pmatrix} 1 \\ x_n \end{pmatrix} \circ \cdots \circ \begin{pmatrix} 1 \\ x_1 \end{pmatrix} \\ &= \bigcirc_{j=1}^{N} \bigcirc_{i=1}^{n} \begin{pmatrix} 1 \\ x_{n-i+1} \end{pmatrix} \mathbb{R}^{\times^{(nN)}2}, \end{aligned} \tag{6.2-65}$$

bestimmt. Somit ist auch dieser Monomtensor ein Rang-1 Tensor

$$\mathsf{M}_p^N\,(\mathbf{x}) = \underbrace{\left[\begin{pmatrix} 1 \\ x_n \end{pmatrix}, \ldots, \begin{pmatrix} 1 \\ x_1 \end{pmatrix}, \ldots\ldots, \begin{pmatrix} 1 \\ x_n \end{pmatrix}, \ldots, \begin{pmatrix} 1 \\ x_1 \end{pmatrix} \right]}_{\text{N times}}. \tag{6.2-66}$$

Auf diesem Weg können auch Polynome höherer Ordnung in der Tensorstruktur abgebildet werden. Allerdings ergibt sich hier eine nicht eindeutige Beschreibung, da die Monome zum Teil mehrfach in dem Monomtensor auftauchen. Diese Redundanzen sind der Art der Konstruktion des Monomtensors geschuldet, der teilweise zu einer sehr großen Anzahl an Dimensionen für Polynome höherer Ordnung führt. Er lässt sich jedoch effizient als Rang-1 Tensor darstellen. Daher soll er hier trotzdem für die Beschreibung von Polynomen höherer Ordnung beibehalten werden, da sich die Darstellung gut für die Entwicklung der Algorithmen eignet.

Durch diese Darstellungsform von Polynomen höherer Ordnung können nun Operationen wie die Multiplikation auf Basis der Parametertensoren definiert werden. Das Ergebnis der Multiplikation zweier Polynome in n Variablen mit maximaler Ordnung N_1 und N_2 in Tensordarstellung

$$h_1(\mathbf{x}) = \left\langle \, \mathsf{H}_1 \, \middle| \, \mathsf{M}_p^{N_1}(\mathbf{x}) \, \right\rangle, \tag{6.2-67}$$

$$h_2(\mathbf{x}) = \left\langle \, \mathsf{H}_2 \, \middle| \, \mathsf{M}_p^{N_2}(\mathbf{x}) \, \right\rangle, \tag{6.2-68}$$

kann durch das äußere Produkt $\mathsf{H}_1 \circ \mathsf{H}_2$ der Parametertensoren der Operanden bestimmt werden zu

$$h_1(\mathbf{x}) \cdot h_2(\mathbf{x}) = \left\langle \, \mathsf{H}_1 \circ \mathsf{H}_2 \, \middle| \, \mathsf{M}_p^{N_1+N_2}(\mathbf{x}) \, \right\rangle. \tag{6.2-69}$$

Es ergibt sich ein Polynom der maximalen Ordnung $N_1 + N_2$. Es ist somit möglich den Parametertensor des Ergebnisses der Multiplikation aus den Parametertensoren der Polynome $h_1(\mathbf{x})$ und $h_2(\mathbf{x})$, die miteinander multipliziert werden, zu berechnen.

Ein ähnliches Vorgehen ist auch bei der partiellen Ableitung eines Polynoms der Ordnung N bezüglich einer Variablen $x_j,\ j = 1, \ldots, n$

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

möglich. Da die Ordnung des Polynoms durch das Ableiten nicht erhöht wird und die Struktur des Polynoms nicht verändert wird, kann auch das Ergebnis der Ableitung durch das kontrahierte Produkt eines Parameter- und Monomtensors

$$\frac{\partial}{\partial x_j} h(\mathbf{x}) = \left\langle \, \mathsf{H}_{x_j} \, \middle| \, \mathsf{M}_p^{N}(\mathbf{x}) \, \right\rangle \tag{6.2-70}$$

beschrieben werden. Der Parametertensor der Ableitung

$$\mathsf{H}_{x_j} = \sum_{k=1}^{N} \mathsf{H} \times_{kn-j+1} \boldsymbol{\Theta} \in \mathbb{R}^{\times^{nN}2}. \tag{6.2-71}$$

berechnet sich einfach durch eine Summe von Tensor Matrix Produkten des Parametertensors H der Ausgangsfunktion mit der Matrix $\boldsymbol{\Theta} = \left(\begin{smallmatrix} 0 & 1 \\ 0 & 0 \end{smallmatrix}\right)$. Im Folgenden wird das Vorgehen anhand eines einfachen Beispiels vorgestellt.

Beispiel 6.2.1 *Gegeben ist eine multilineare Funktion mit zwei Variablen*

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

Durch die Anwendung von (6.2-71) berechnet sich der Parametertensor der Ableitung der Funktion nach x_1 zu

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

der das Polynom

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

beschreibt. Durch die Mode-2 Multiplikation mit $\mathbf{\Theta}$ *werden die Elemente von* H*, die zu den Monomen* x_1 *und* x_1x_2 *gehören so umsortiert, dass sie im Ergebnis den Monomen* 1 *und* x_2 *zugeordnet sind. Alle anderen sind gleich Null. Dies ist genau der Fall, wenn die gegebene Funktion nach* x_1 *abgeleitet wird.*

Zum Entwurf eines Reglers für die Feedback Linearisierung ist eine weitere Operation notwendig. Wie in (6.2-60) vorgestellt, müssen Lie Ableitungen (6.2-62) berechnet werden. Zur Berechnung der Lie Ableitung auf Basis von Parametertensoren können die beiden vorangegangen eingeführten Operationen verwendet werden. Die Funktionen $\mathbf{h}(\mathbf{x})$ und $g(\mathbf{x})$ sind im Tensorformat gegeben durch

$$\mathbf{h}(\mathbf{x}) = \left\langle\, \mathsf{H} \,\middle|\, \mathsf{M}_p^N\,(\mathbf{x})\,\right\rangle, \tag{6.2-72}$$

$$g(\mathbf{x}) = \left\langle\, \mathsf{G} \,\middle|\, \mathsf{M}_p^N\,(\mathbf{x})\,\right\rangle, \tag{6.2-73}$$

mit den Parametertensoren $\mathsf{H} \in \mathbb{R}^{\times^{(nN)}2\times n}$ und $\mathsf{G} \in \mathbb{R}^{\times^{(nN)}2}$. In dieser Darstellung kann die Lie Ableitung des skalaren Polynoms (6.2-73) entlang des Vektorfeldes (6.2-72) mit (6.2-69) und (6.2-70) berechnet werden durch

$$\begin{aligned}L_{\mathbf{h}}^l g(\mathbf{x}) &= \sum_{i=1}^{n} h_i(\mathbf{x}) \frac{\partial}{\partial x_i} L_{\mathbf{h}}^{l-1} g(\mathbf{x}) \\ &= \left\langle\, \mathsf{L}_{\mathsf{H},\mathsf{G},l} \,\middle|\, \mathsf{M}_p^{N(l+1)}\,(\mathbf{x})\,\right\rangle. \end{aligned} \tag{6.2-74}$$

Der Parametertensor ergibt sich zu

$$\mathsf{L}_{\mathsf{H},\mathsf{G},l} = \begin{cases} \mathsf{G} & \textit{für } l = 0, \\ \sum\limits_{i=1}^{n} \mathsf{H}_i \circ \left(\sum\limits_{k=1}^{N} \mathsf{G} \times_{kn-i+1} \mathbf{\Theta} \right) & \textit{für } l = 1, \\ \sum\limits_{i=1}^{n} \mathsf{H}_i \circ \left(\sum\limits_{k=1}^{lN} \mathsf{L}_{\mathsf{H},\mathsf{G},l-1} \times_{kn-i+1} \mathbf{\Theta} \right) & \textit{sonst}, \end{cases} \tag{6.2-75}$$

mit dem Subtensor $\mathsf{H}_i = \mathsf{H}(:,\ldots,:,i)$, bei dem der Index der letzten Dimension von H fixiert ist.

Hinweis 6.2.1 *Es wurde gezeigt, wie sich die Polynomoperationen der Multiplikation, der partiellen Ableitung und der Lie Ableitung von Polynomen in der Tensordarstellung auf Basis der Parametertensoren analytisch korrekt berechnen lassen. Dazu ist keine symbolische Rechnung notwendig, sondern die Berechnungen finden auf den Parametern statt. Diese Berechnungen können gerade für Polynome in vielen Variablen effizient durchgeführt werden, wenn die Parametertensoren der Operanden als dekomponierte Tensoren vorliegen. Da die notwendigen Tensoroperationen, wie das äußere Produkt oder das Tensor Matrix Produkt auch für dekomponierte Tensoren definiert sind, kann auch das Ergebnis direkt in dekomponierter Form berechnet werden [21]. Somit müssen keine vollen Tensoren erstellt werden.*

6.2.3 Feedback Linearisierung für CP dekomponierte MTI Systeme

Im Abschnitt 6.2.1 wurde das Konzept der Feedback Linearisierung, wie aus der Literatur bekannt, für allgemeine nichtlineare Systeme vorgestellt. In diesem Abschnitt soll dieses allgemeine Vorgehen auf die spezielle Struktur von MTI Systemen angepasst werden. Der Fokus liegt hierbei auf der Tensordarstellung und besonders auf der CP dekomponierten Repräsentation, die es ermöglicht, auch Modelle von sehr großen Systemen darzustellen. Ein Entwurfsalgorithmus auf Basis der dekomponierten Tensoren würde es ermöglichen, ein Reglerdesign auch für solche Systeme durchführen zu können, wo es für die volle Darstellung und auch symbolisch nicht möglich ist.

Ein zeitkontinuierliches affines SISO MTI System ist gegeben durch

$$\dot{\mathbf{x}} = \mathbf{a}(\mathbf{x}) + \mathbf{b}(\mathbf{x})u = \langle\, \mathsf{A} \mid \mathsf{M}(\mathbf{x}) \,\rangle + \langle\, \mathsf{B} \mid \mathsf{M}(\mathbf{x}) \,\rangle\, u, \tag{6.2-76}$$

$$y = c(\mathbf{x}) = \langle\, \mathsf{C} \mid \mathsf{M}(\mathbf{x}) \,\rangle\,. \tag{6.2-77}$$

Es wird angenommen, dass das System Feedback linearisierbar mit einem wohl definierten relativen Grad ρ_{sys} ist. Die Lie Ableitung (6.2-74) der multilinearen Ausgangsfunktion $c(\mathbf{x})$ gegeben durch den Tensor C entlang des multilinearen Vektorfeldes $\mathbf{a}(\mathbf{x})$ gegeben durch den Tensor A ergibt sich zu

$$L^l_{\mathbf{a}} c(\mathbf{x}) = \sum_{i=1}^{n} a_i(\mathbf{x}) \frac{\partial}{\partial x_i} L^{l-1}_{\mathbf{a}} c(\mathbf{x}) = \left\langle\, \mathsf{L}_{\mathsf{A},\mathsf{C},l} \;\middle|\; \mathsf{M}_p^{l+1}(\mathbf{x}) \,\right\rangle, \tag{6.2-78}$$

mit dem Parametertensor

$$\mathsf{L}_{\mathsf{A},\mathsf{C},l} = \begin{cases} \mathsf{C} & \textit{für } l = 0, \\ \sum\limits_{i=1}^{n} \mathsf{A}_i \circ (\mathsf{C} \times_{n-i+1} \boldsymbol{\Theta}) & \textit{für } l = 1, \\ \sum\limits_{i=1}^{n} \mathsf{A}_i \circ \left(\sum\limits_{k=1}^{l} \mathsf{L}_{\mathsf{A},\mathsf{C},l-1} \times_{kn-i+1} \boldsymbol{\Theta} \right) & \textit{sonst}, \end{cases} \tag{6.2-79}$$

mit $l = 0, \ldots, \rho_{sys}$ sowie dem Subtensor $\mathsf{A}_i = \mathsf{A}(:, \ldots, :, i),\ i = 1, \ldots, n$ von A. Die Lie Ableitungen entlang $\mathbf{b}(\mathbf{x})$

$$L_{\mathbf{b}} L^l_{\mathbf{a}} c(\mathbf{x}) = \sum_{i=1}^{n} b_i(\mathbf{x}) \frac{\partial}{\partial x_i} L^l_{\mathbf{a}} c(\mathbf{x}) = \left\langle\, \mathsf{L}_{\mathsf{B},\mathsf{A},\mathsf{C},l} \;\middle|\; \mathsf{M}_p^{l+2}(\mathbf{x}) \,\right\rangle, \tag{6.2-80}$$

haben einen Parametertensor

$$\mathsf{L}_{\mathsf{B},\mathsf{A},\mathsf{C},l} = \sum_{i=1}^{n} \mathsf{B}_i \circ \left(\sum_{k=1}^{l+1} \mathsf{L}_{\mathsf{A},\mathsf{C},l} \times_{kn-i+1} \boldsymbol{\Theta} \right), \tag{6.2-81}$$

mit den Subtensoren $\mathsf{B}_i = \mathsf{B}(:, \ldots, :, i),\ i = 1, \ldots, n$ von B. Alle notwendigen Tensoroperationen zur Berechnung der Lie Ableitungen, d.h. die Summation, das äußere Produkt, und das Mode-k Tensor Matrix Produkt, sind auch für dekomponierte Tensoren definiert, sodass keine vollen Tensoren erstellt werden müssen, wenn das MTI Modell in dekomponierter Darstellung angegeben wird. Mit den Tensordarstellungen (6.2-78) und (6.2-80) der Lie Ableitungen des MTI Systems lässt sich der Regler (6.2-60) der Feedback Linearisierung beschreiben durch

$$u = \frac{-\left\langle \sum\limits_{i=0}^{\rho} \mu_i \mathsf{L}_{\mathsf{A},\mathsf{C},i} \;\middle|\; \mathsf{M}_p^{\rho+1}(\mathbf{x}) \right\rangle + \mu_0\, r}{\left\langle \mathsf{L}_{\mathsf{B},\mathsf{A},\mathsf{C},\rho-1} \;\middle|\; \mathsf{M}_p^{\rho+1}(\mathbf{x}) \right\rangle}. \tag{6.2-82}$$

Man erhält somit einen Regler für die Feedback Linearisierung mit einer fixen Struktur. Die Faktoren μ_i und die Tensoren $\mathsf{L}_{\mathsf{A},\mathsf{C},i}$ und $\mathsf{L}_{\mathsf{B},\mathsf{A},\mathsf{C},\rho-1}$ müssen vorgegeben werden, um den Regler auf eine bestimmte Regelstrecke anzupassen. Hierbei handelt es sich um eine Parametrierung des Reglers, wie bei der Vorgabe einer Verstärkungsmatrix beim Design einer Standard Zustandsrückführung im linearen Fall. Die Struktur des Reglers für MTI Systeme ist festgelegt durch (6.2-82) und hängt nicht von der Strecke ab. Diese strukturelle Invarianz des Reglers ist ein großer Vorteil für die Anwendung. Verglichen mit dem allgemeinen, nichtlinearen Fall ist es somit nicht nötig, einen beliebigen Satz an mathematischen Operationen für die Reglerimplementierung vorzusehen, der abhängig von dem betrachteten System ist. In dem hier entwickelten Regler der Feedback Linearisierung für MTI Systeme ist der Regler auf die gegebene Struktur beschränkt und kann durch einen begrenzten Satz an Operationen auf den Parameterräumen bestimmt werden, was ein deutlicher Vorteil ist.

Beispiel 6.2.2 *Die eingeführte Methode der Feedback Linearisierung wird hier angewendet auf ein*

SISO MTI System mit 2 Zuständen

$$\begin{aligned}\begin{pmatrix}\dot{x}_1\\ \dot{x}_2\end{pmatrix} &= \begin{pmatrix}2x_2\\ -x_1+0.2x_1x_2\end{pmatrix} + \begin{pmatrix}0\\1\end{pmatrix} u,\\ y &= 1+2x_1,\end{aligned}$$

um die einzelnen Schritte des Entwurfsprozesses zu zeigen. Das System gehört zu der Klasse der affinen MTI Systeme (6.2-76) und (6.2-77), sodass die Parameter als CP Tensoren

$$\begin{aligned}\mathsf{A} &= \left[\begin{pmatrix}0&1&0\\1&0&1\end{pmatrix}, \begin{pmatrix}1&0&0\\0&1&1\end{pmatrix}, \begin{pmatrix}1&0&0\\0&1&1\end{pmatrix}\right] \cdot \begin{pmatrix}2\\-1\\0.2\end{pmatrix},\\ \mathsf{B} &= \left[\begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}\right] \cdot 1,\\ \mathsf{C} &= \left[\begin{pmatrix}1&1\\0&0\end{pmatrix}, \begin{pmatrix}0&1\\1&0\end{pmatrix}\right] \cdot \begin{pmatrix}2\\1\end{pmatrix},\end{aligned}$$

angegeben werden können. Zur Bestimmung des Reglers müssen die Lie Ableitungen des Systems berechnet werden. Die Parametertensoren der Lie Ableitungen entlang $\mathbf{a}(\mathbf{x})$ ergeben sich mit (6.2-79) in CP Darstellung zu

$$\begin{aligned}\mathsf{L}_{\mathsf{A},\mathsf{C},1} &= \left[\begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}\right] \cdot 4,\\ \mathsf{L}_{\mathsf{A},\mathsf{C},2} &= \left[\begin{pmatrix}1&0\\0&1\end{pmatrix}, \begin{pmatrix}0&0\\1&1\end{pmatrix}, \begin{pmatrix}1&1\\0&0\end{pmatrix}, \begin{pmatrix}1&1\\0&0\end{pmatrix}, \begin{pmatrix}1&1\\0&0\end{pmatrix}, \begin{pmatrix}1&1\\0&0\end{pmatrix}\right] \cdot \begin{pmatrix}-4\\0.8\end{pmatrix},\end{aligned}$$

woraus man die Ableitungen

$$\begin{aligned}L_{\mathbf{a}}c(\mathbf{x}) &= \left\langle\, \mathsf{L}_{\mathsf{A},\mathsf{C},1} \,\middle|\, \mathsf{M}_p^2\left(x_1,x_2\right) \,\right\rangle = 4x_2,\\ L_{\mathbf{a}}^2c(\mathbf{x}) &= \left\langle\, \mathsf{L}_{\mathsf{A},\mathsf{C},2} \,\middle|\, \mathsf{M}_p^3\left(x_1,x_2\right) \,\right\rangle = -4x_1+0.8x_1x_2.\end{aligned}$$

erhält. Die Lie Ableitungen entlang $\mathbf{b}(\mathbf{x})$ führen zu $L_{\mathbf{b}}c(\mathbf{x}) = 0$ und $L_{\mathbf{b}}L_{\mathbf{a}}c(\mathbf{x}) = 4$ durch die Parametertensoren (6.2-81)

$$\begin{aligned}\mathsf{L}_{\mathsf{B},\mathsf{A},\mathsf{C},0} &= \left[\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}0\\0\end{pmatrix}\right] \cdot 0,\\ \mathsf{L}_{\mathsf{B},\mathsf{A},\mathsf{C},1} &= \left[\begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}\right] \cdot 4.\end{aligned}$$

Aufgrund von $L_{\mathbf{b}}L_{\mathbf{a}}c(\mathbf{x}) = 4 \neq 0\ \forall \mathbf{x}$ ist der relative Grad ρ_{sys} des Systems gleich zwei und entspricht somit der Anzahl an Zuständen. Daher hat das System vollen Grad und die Nulldynamiken müssen nicht überprüft werden. Im geschlossenen Kreis soll das lineare Verhalten

$$\mu_2\ddot{y} + \mu_1\dot{y} + \mu_0 y = \mu_0 r$$

erreicht werden, wobei beispielhaft die Faktoren $\mu_1 = \mu_0 = 10$ und $\mu_2 = 1$ gewählt werden. Der Regler für die Feedback Linearisierung (6.2-82) ist unter Verwendung der berechneten Parametertensoren der Lie Ableitungen gegeben durch

$$\begin{aligned}u &= \frac{-\left\langle\, \mu_0\mathsf{C} + \mu_1\mathsf{L}_{\mathsf{A},\mathsf{C},1} + \mu_2\mathsf{L}_{\mathsf{A},\mathsf{C},2} \,\middle|\, \mathsf{M}_p^3(\mathbf{x}) \,\right\rangle + \mu_0\, r}{\left\langle \mathsf{L}_{\mathsf{B},\mathsf{A},\mathsf{C},1} \,\middle|\, \mathsf{M}_p^3(\mathbf{x}) \,\right\rangle}\\ &= \frac{-10-16x_1-40x_2-0.8x_1x_2+10r}{4}.\end{aligned}$$

Abbildung 6.2-24 zeigt das Ergebnis der Simulation im geschlossenen Kreis für einen Sprung im Referenzsignal r. Der geschlossene Kreis verhält sich wie vorgegeben wie ein lineares System zweiter Ordnung.

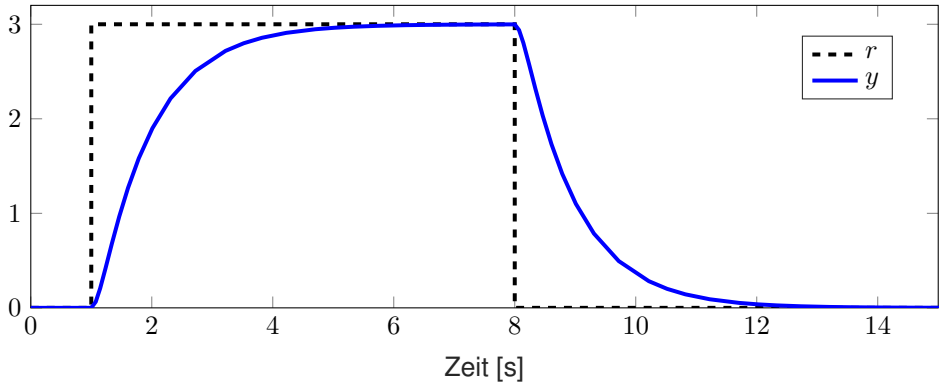


Abbildung 6.2-24: Simulation im geschlossenen Kreis mit Feedback linearisierendem Regler

6.2.4 Komplexitätsanalyse

Die Verwendung von Niedrigrang-Approximationen für Parametertensoren macht es möglich, auch sehr große MTI Systeme darzustellen, wie im Kapitel 6.1.3 vorgestellt. In diesem Kapitel konnte gezeigt werden, dass dies auch die Berechnung von Reglern für diese Systeme ermöglicht. Der hergeleitete Algorithmus zur Feedback Linearisierung arbeitet mit den dekomponierten Darstellungen der Systeme. Alle darin verwendeten Operationen sind auch für dekomponierte Tensoren definiert, sodass keine volle Tensordarstellung während des gesamten Designprozesses erstellt werden muss und ein Regler in dekomponierter Darstellung bestimmt wird. Abbildung 6.2-25 zeigt, dass dadurch der Speicheraufwand auch für große Systeme handhabbar bleibt. In der Abbildung ist eine obere Grenze für die Anzahl an Elementen angegeben, die für die Parametertensoren des Reglers gespeichert werden müssen, wenn diese in CP Darstellung vorliegen. Verschiedene Ordnungen und Anzahlen an Rang-1 Komponenten der Parametertensoren A, B und C des Systems für einen relativen Grad von Eins werden berücksichtigt. Häufig können die Tensoren des Reglers durch weniger Rang-1 Komponenten approximiert werden, was den Speicheraufwand weiter reduziert.

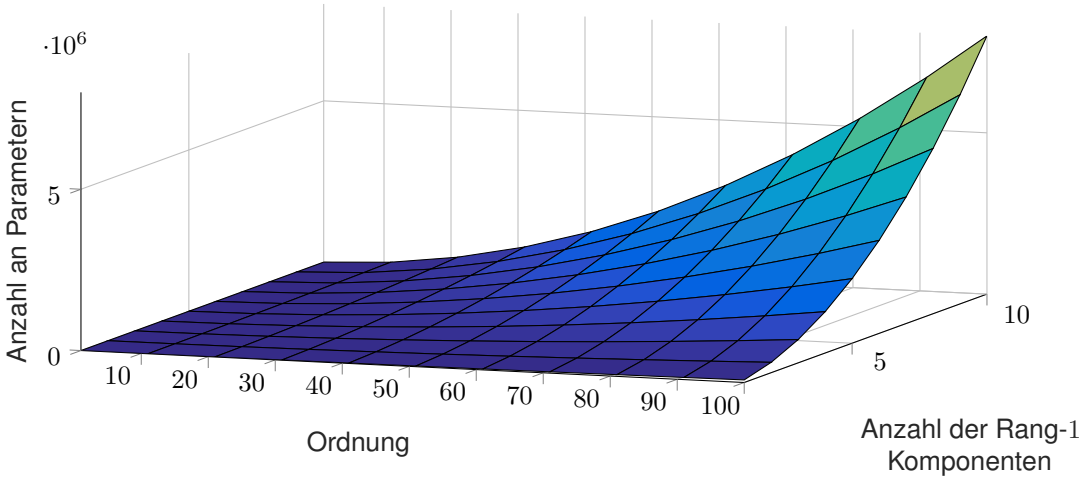


Abbildung 6.2-25: Obere Grenze für die Anzahl der zu speichernden Elemente eines Reglers der Feedback Linearisierung für dekomponierte MTI Systeme

Auch während der Anwendung des Reglers ist die dekomponierte Struktur von Vorteil, da die Regelungsfunktion auf Basis der dekomponierten Faktromatrizen ausgewertet werden kann. Auch hier sind keine vollen Tensoren notwendig, sodass das hier entworfene Verfahren auch für große Systeme angewendet werden kann, die durch Standardverfahren nicht mehr handhabbar sind, wenn Niedrigrang-Approximationen existieren.

6.3 Dezentrale Regelung mit Zustandsrückführung

Der Entwurf von Zustandsrückführungen als LQR (linear quadratic regulator) Regler ist eine etablierte Entwurfsmethode, [27]. Diese führt im Standarddesign zu einem zentralen Regler, der die volle Systeminformation benötigt, d.h. Informationen über alle Zuständen müssen bekannt sein, um die Steuereingänge berechnen zu können. In heutigen Anwendungen werden die Systeme immer komplexer, z.B. im Bereich von Smart Grids, großen RLT Anlagen oder Multiagentensystemen. Ein zentrales Design führt zu einer aufwendigen Kommunikationsinfrastruktur, [33]. Daher bietet sich die Implementierung eines Reglernetzwerkes an, bei dem die Regelungsaufgabe auf mehrere Knoten verteilt ist [47], [3]. Im Bereich der Zustandsrückführung für lineare Systeme haben [30], [42] und [46] Methoden entworfen den Regler zu verteilen. Dies erfolgt über spärlich besetzte Verstärkungsmatrizen, sodass nicht alle Verbindungen zwischen Zuständen und Eingängen hergestellt werden müssen. Damit ist es möglich die Kommunikationsstruktur deutlich zu vereinfachen. Die Reglerstruktur sowie die Verstärkungsmatrix in dieser Struktur werden über die Lösung eines Optimierungsproblems bestimmt. Es wird ein Trade-off zwischen minimalem Kommunikationsaufwand und geringem Verlust der Regelungsgüte im Vergleich zum zentralen Design gesucht.

Da sich MTI Systeme sehr gut für die Modellierung von dynamischen Prozessen im Bereich der Gebäude eignen, soll die Methode für lineare Systeme, die von [30] eingeführt wurde, hier auf MTI Systeme angewendet werden, um eine geeignete Kommunikationsstruktur und somit einen dezentralen Zustandsregler zu finden. Diese Methode wird dann auf ein Beispiel aus dem Bereich der Heizungsanlagen angewendet.

6.3.1 Dezentraler Entwurf einer Zustandsrückführung für lineare Systeme

Die verwendete dezentrale Entwurfsmethode einer Zustandsrückführung wurde für lineare Systeme in [30] eingeführt. Ein Regelkreis mit einer Rückführung der Zustände über eine Verstärkungsmatrix $\mathbf{K}$ ist in Abbildung 6.3-26 dargestellt.

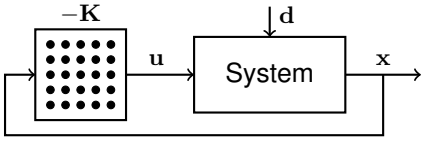


Abbildung 6.3-26: Regelung mit Zustandsrückführung mit zentralem Design

Im Standard LQR (linear quadratic regulator) Entwurf wird ein Zustandsrückführungsregler

$$\mathbf{u} = -\mathbf{K}\mathbf{x}, \tag{6.3-83}$$

entworfen, wobei im allgemeinen Fall alle Zustände $\mathbf{x} \in \mathbb{R}^n$ verwendet werden, um die Eingänge $\mathbf{u} \in \mathbb{R}^m$ des Systems zu berechnen. Im Gegensatz zu diesem zentralen Entwurf haben [30] eine Methode vorgestellt eine gewisse Struktur der Rückführverstärkung vorzugeben. Dadurch erhält man Nullen in der Rückführverstärkungsmatrix $\mathbf{K} \in \mathbb{R}^{m\times n}$, womit der Kommunikationsaufwand reduziert werden kann, da nicht alle Signalverbindungen zwischen den Zuständen und Eingängen hergestellt werden müssen. Es sind nur die Verbindungen zwischen x_j und u_i nötig, für die $k(i,j) \neq 0$ gilt. Der Algorithmus der in [30] entwickelt wurde, bestimmt eine Besetzungsstruktur mit Nichtnull-Elementen von $\mathbf{K}$, sodass ein Trade-off zwischen Kommunikationsaufwand und Regelungsgüte erreicht wird. Ein kurzer Überblick wird hier gegeben, [13].

Dazu wird für die Beschreibung des Systemverhaltens ein lineares Modell

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}_u\mathbf{u} + \mathbf{B}_d\mathbf{d}, \tag{6.3-84}$$

$$z = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}, \tag{6.3-85}$$

verwendet. Mit dem Ausgang $z \in \mathbb{R}$ wird die Regelungsgüte mit den Matrizen $\mathbf{C} = \begin{pmatrix}\mathbf{Q}^{1/2} & \mathbf{0}\end{pmatrix}^T$ und $\mathbf{D} = \begin{pmatrix}\mathbf{0} & \mathbf{R}^{1/2}\end{pmatrix}^T$ beschrieben. Die Eingänge des Modells sind aufgeteilt in Stellgrößen $\mathbf{u} \in \mathbb{R}^m$

und Störgrößen $\mathbf{d} \in \mathbb{R}^{m_d}$. Die Matrizen $\mathbf{Q} = \mathbf{Q}^T \geq 0$ und $\mathbf{R} = \mathbf{R}^T \geq 0$ sind Gewichtungsfaktoren für die Regelungsgüte und bewerten jeweils die Referenzfolge und den Stellaufwand. Es wird angenommen, dass $(\mathbf{A}, \mathbf{B}_u)$ stabilisierbar und $(\mathbf{A}, \mathbf{Q}^{1/2})$ detektierbar ist. Bei dem zentralen Entwurf wird eine Rückführungsverstärkung berechnet, sodass die $\mathcal{H}_2$ Norm der Übertragungsfunktion von $\mathbf{d}$ nach z minimiert wird

$$\min_{\mathbf{K}} J(\mathbf{K}), \tag{6.3-86}$$

wobei die Kostenfunktion durch

$$J(\mathbf{K}) = \begin{cases} \text{trace}\left(\mathbf{B}_d^T \mathbf{P}(\mathbf{K}) \mathbf{B}_d\right) & \text{, für stabilisierendes } \mathbf{K}, \\ \infty & \text{, sonst,} \end{cases} \tag{6.3-87}$$

mit der gramschen Beobachtbarkeitsmatrix $\mathbf{P}(\mathbf{K})$ gegeben ist. Im dezentralen Entwurf soll die Rückführungsmatrix $\mathbf{K}$ eine bestimmte Besetzungsstruktur $\mathcal{S}$ haben, sodass sich das Optimierungsproblem

$$\min_{\mathbf{K}} J(\mathbf{K}), \text{ subject to } \mathbf{K} \in \mathcal{S}, \tag{6.3-88}$$

ergibt, [30]. Um eine geeignete Besetzungsstruktur $\mathcal{S}$ zu bestimmen wird die Kostenfunktion in (6.3-86) erweitert zu

$$\min_{\mathbf{K}} J(\mathbf{K}) + \gamma g(\mathbf{K}), \tag{6.3-89}$$

wobei die Funktion $g(\mathbf{K})$ die Besetzungsdichte von $\mathbf{K}$ bestimmt, [29]. Der Faktor γ ist ein Tuning Parameter, um einen Trade-off zwischen Regelungsgüte und Kommunikationsaufwand, d.h. Besetzungsdichte von $\mathbf{K}$ zu finden. Setzt man $\gamma = 0$ führt dies zu dem zentralen Problem (6.3-86). Ein größerer Wert $\gamma > 0$ bedeutet, dass eine geringere Besetzungsdichte unterstützt wird. Wenn $g(\mathbf{K})$ gleich der Anzahl der Nichtnull Elemente von $\mathbf{K}$ gewählt wird, wird das Optimierungsproblem (6.3-89) ein kombinatorisches, dass nur sehr schwer zu lösen ist. Daher wird in [29] eine gewichtete l_1-Norm

$$g(\mathbf{K}) = \sum_{i=1}^{m} \sum_{j=1}^{n} w(i,j) \, |K(i,j)| \tag{6.3-90}$$

verwendet, um die Besetzungsdichte von $\mathbf{K}$ zu bewerten, wobei die Gewichte $w(i,j)$ in einem iterativen Prozess ermittelt werden, wie in [7] beschrieben. Das Optimierungsproblem 6.3-89 zur Bestimmung der Besetzungsstruktur wird mit Hilfe des ADMM (alternating direction of multipliers) Verfahrens gelöst, [29]. Nachdem die Struktur bestimmt wurde, kann das Ergebnis noch etwas verfeinert werden, indem die optimale Rückführverstärkung für die gegebene Struktur durch Lösen von (6.3-88) berechnet wird. Wie in [30] beschrieben, erfolgt dies durch ein Newton Verfahren mit konjugierten Gradienten. Somit erhält man einen dezentralen Zustandsrückführungsregler mit weniger Kommunikationsaufwand als im zentralen Fall und nur geringen Verlusten in der Regelungsgüte.

6.3.2 Linearisierung von MTI Systemen

In der Regelungstechnik sind viele Werkzeuge und Methoden für lineare Zustandsraummodelle vorhanden, [31]. Da die Systeme in der Realität nichtlinear sind, wird ihr Verhalten häufig durch ein lineares Modell um einen Arbeitspunkt linearisiert, um die Methoden der linearen Regelung anwenden zu können. Im Folgenden wird ein Algorithmus zur Linearisierung von MTI Systemen hergeleitet, der die Ableitungsmethode (6.2-70) basierend auf den Parametertensoren benutzt. Das Ziel ist ein MTI System

$$\Phi(\mathbf{x}) = \mathbf{f}(\mathbf{x}, \mathbf{u}) = \langle \, \mathsf{F} \mid \mathsf{M}(\mathbf{x}, \mathbf{u}) \, \rangle, \tag{6.3-91}$$

$$\mathbf{y} = \mathbf{g}(\mathbf{x}, \mathbf{u}) = \langle \, \mathsf{G} \mid \mathsf{M}(\mathbf{x}, \mathbf{u}) \, \rangle, \tag{6.3-92}$$

in der Umgebung eines Arbeitspunktes $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ durch ein lineares System

$$\Phi(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}, \tag{6.3-93}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}, \tag{6.3-94}$$

zu approximieren. Die Systemmatrizen der linearen Approximation folgen aus den partiellen Ableitungen der Zustands- und Ausgangsgleichungen des MTI Systems bezüglich der Zustände und Eingänge, [31]. Die Berechnung der partiellen Ableitungen auf der Basis der Parametertensoren durch (6.2-70) wird angepasst, um die Trennung der Variablen in Zustände und Eingänge zu ermöglichen. Die partiellen Ableitungen der Zustandsfunktion bezüglich der Zustände und Eingänge sind gegeben durch

$$\begin{aligned}\mathsf{F}_{x_j} &= \mathsf{F} \times_{n+m-j+1} \boldsymbol{\Theta},\ j = 1, \ldots, n,\\ \mathsf{F}_{u_j} &= \mathsf{F} \times_{m-j+1} \boldsymbol{\Theta},\ j = 1, \ldots, m.\end{aligned}$$

Der Ansatz kann in gleicher Weise für die Ausgangsgleichung durchgeführt werden. Die Auswertung im Arbeitspunkt ergibt

$$\mathbf{A}(:,j) = \left.\frac{\partial \mathbf{f}(\mathbf{x},\mathbf{u})}{\partial x_j}\right|_{\substack{\mathbf{x}=\bar{\mathbf{x}},\\ \mathbf{u}=\bar{\mathbf{u}}}} = \left\langle \mathsf{F}_{x_j} \mid \mathsf{M}(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \right\rangle,\ j = 1, \ldots, n, \tag{6.3-95}$$

$$\mathbf{B}(:,j) = \left.\frac{\partial \mathbf{f}(\mathbf{x},\mathbf{u})}{\partial u_j}\right|_{\substack{\mathbf{x}=\bar{\mathbf{x}},\\ \mathbf{u}=\bar{\mathbf{u}}}} = \left\langle \mathsf{F}_{u_j} \mid \mathsf{M}(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \right\rangle,\ j = 1, \ldots, m, \tag{6.3-96}$$

$$\mathbf{C}(:,j) = \left.\frac{\partial \mathbf{g}(\mathbf{x},\mathbf{u})}{\partial x_j}\right|_{\substack{\mathbf{x}=\bar{\mathbf{x}},\\ \mathbf{u}=\bar{\mathbf{u}}}} = \left\langle \mathsf{G}_{x_j} \mid \mathsf{M}(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \right\rangle,\ j = 1, \ldots, n, \tag{6.3-97}$$

$$\mathbf{D}(:,j) = \left.\frac{\partial \mathbf{g}(\mathbf{x},\mathbf{u})}{\partial u_j}\right|_{\substack{\mathbf{x}=\bar{\mathbf{x}},\\ \mathbf{u}=\bar{\mathbf{u}}}} = \left\langle \mathsf{G}_{u_j} \mid \mathsf{M}(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \right\rangle,\ j = 1, \ldots, m. \tag{6.3-98}$$

Daher sind die Spalten der Systemmatrizen des linearen Zustandsraummodells gleich den Spalten der Jacobimatrix

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

der Zustandsfunktion $\mathbf{f}(\mathbf{x},\mathbf{u})$ und der Jacobimatrix $\mathbf{J}_{\mathbf{g}}(\bar{\mathbf{x}}, \bar{\mathbf{u}}) = \left\langle \mathsf{J}_{\mathbf{g}} \mid \mathsf{M}(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \right\rangle$ der Ausgangsfunktion $\mathbf{g}(\mathbf{x},\mathbf{u})$. Die Parametertensoren der Jacobimatrizen haben die Dimensionen $\mathbb{R}^{\times^{n+m}2\times n\times(n+m)}$ sowie $\mathbb{R}^{\times^{n+m}2\times p\times(n+m)}$. Die ersten $(n+m)$ Dimensionen gehören zu den Zuständen $\mathbf{x}$ mit dem Indexvektor $\mathbf{i}_x \in \mathbb{R}^n$ und Eingängen $\mathbf{u}$ mit dem Indexvektor $\mathbf{i}_u \in \mathbb{R}^m$. Die letzten beiden Dimensionen i_{n+m+1} und i_{n+m+2} sind die Dimensionen der Jacobimatrix. Die Parametertensoren werden aus den partiellen Ableitungen (6.3-95) bis (6.3-98) berechnet. Die ersten n Spalten von $\mathbf{J}_{\mathbf{f}}(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ ergeben die Systemmatrix $\mathbf{A}$ und die folgenden m Spalten die Eingangsmatrix $\mathbf{B}$. Aus den Parametertensoren der Jacobimatrizen lassen sich die Parametertensoren der Matrizen des linearen Zustandsraummodell ableiten, mit denen die linearen Systemmatrizen durch Auswertung der kontrahierten Produkte

$$\mathbf{A}(\bar{\mathbf{x}}, \bar{\mathbf{u}}) = \left\langle \mathsf{A}_{lin} \mid \mathsf{M}(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \right\rangle \in \mathbb{R}^{n\times n}, \tag{6.3-99}$$

$$\mathbf{B}(\bar{\mathbf{x}}, \bar{\mathbf{u}}) = \left\langle \mathsf{B}_{lin} \mid \mathsf{M}(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \right\rangle \in \mathbb{R}^{n\times m}, \tag{6.3-100}$$

$$\mathbf{C}(\bar{\mathbf{x}}, \bar{\mathbf{u}}) = \left\langle \mathsf{C}_{lin} \mid \mathsf{M}(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \right\rangle \in \mathbb{R}^{p\times n}, \tag{6.3-101}$$

$$\mathbf{D}(\bar{\mathbf{x}}, \bar{\mathbf{u}}) = \left\langle \mathsf{D}_{lin} \mid \mathsf{M}(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \right\rangle \in \mathbb{R}^{p\times m}, \tag{6.3-102}$$

im Arbeitspunkt berechnet werden. Die Parametertensoren $\mathsf{A}_{lin} \in \mathbb{R}^{\times^{(n+m)}2\times n\times n}$ und $\mathsf{B}_{lin} \in \mathbb{R}^{\times^{(n+m)}2\times n\times m}$ der Zustandsgleichung haben die Fibern

$$\mathbf{a}_{lin}(\mathbf{i}_u, \mathbf{i}_x, :, j) = \mathbf{j}_{\mathbf{f}}(\mathbf{i}_u, \mathbf{i}_x, :, j) = \mathbf{f}_{x_j}(\mathbf{i}_u, \mathbf{i}_x, :),\ j = 1, \ldots, n, \tag{6.3-103}$$

$$\mathbf{b}_{lin}(\mathbf{i}_u, \mathbf{i}_x, :, j) = \mathbf{j}_{\mathbf{f}}(\mathbf{i}_u, \mathbf{i}_x, :, n+j) = \mathbf{f}_{u_j}(\mathbf{i}_u, \mathbf{i}_x, :),\ j = 1, \ldots, m. \tag{6.3-104}$$

Die Fibern der Parametertensoren $\mathsf{C}_{lin} \in \mathbb{R}^{\times^{(n+m)}2\times p\times n}$ und $\mathsf{D}_{lin} \in \mathbb{R}^{\times^{(n+m)}2\times p\times m}$ der Ausgangsgleichung werden aus der Jacobimatrix $\mathbf{J}_{\mathbf{g}}(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ der Ausgangsgleichung abgeleitet

$$\mathbf{c}_{lin}(\mathbf{i}_u, \mathbf{i}_x, :, j) = \mathbf{j}_{\mathbf{g}}(\mathbf{i}_u, \mathbf{i}_x, :, j) = \mathbf{g}_{x_j}(\mathbf{i}_u, \mathbf{i}_x, :),\ j = 1, \ldots, n, \tag{6.3-105}$$

$$\mathbf{d}_{lin}(\mathbf{i}_u, \mathbf{i}_x, :, j) = \mathbf{j}_{\mathbf{g}}(\mathbf{i}_u, \mathbf{i}_x, :, n+j) = \mathbf{g}_{u_j}(\mathbf{i}_u, \mathbf{i}_x, :),\ j = 1, \ldots, m. \tag{6.3-106}$$

Die Größen der ersten $n+m$ Dimensionen der Tensoren A_{lin} bis D_{lin} sind gleich derer von F und G, bezeichnet mit den Indexvektoren $\mathbf{i}_x$ und $\mathbf{i}_u$. Die letzten beiden Dimensionen, d.h. i_{n+m+1} und i_{n+m+2}, sind die Dimensionen der Matrizen des linearen Systems $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ und $\mathbf{D}$. Sind die Tensoren A_{lin} bis D_{lin} einmal bestimmt, kann die Linearisierung um einen bestimmten Arbeitspunkt sehr einfach durch die Auswertung der kontrahierten Produkte (6.3-99)-(6.3-102) im Arbeitspunkt berechnet werden. Es ist somit eine arbeitspunktabhängige Darstellung der Linearisierung gefunden. Auch hier ist wieder eine effiziente Darstellung über dekomponierte Tensoren möglich, da alle Operationen auch für dekomponierte Tensoren definiert sind.

6.3.3 Dezentrale Zustandsrückführung für MTI Systeme

Die im Abschnitt 6.3.1 nach [30] beschriebene dezentrale Zustandsrückführungsregelung für lineare Systeme soll nun auf MTI Systeme angewendet werden.

Regelungsproblem

Ein dezentraler Zustandsrückführungsregler soll für ein MTI System

$$\dot{\mathbf{x}} = \langle\, \mathsf{F} \mid \mathsf{M}\,(\mathbf{x}, \mathbf{u}, \mathbf{d})\, \rangle\,, \tag{6.3-107}$$

$$z = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}, \tag{6.3-108}$$

mit einem Störgrößeneingang $\mathbf{d} \in \mathbb{R}^{m_d}$ entworfen werden. Die Zustandsgleichung ist dabei in einer MTI Tensorform gegeben. Der Ausgang z, der die Regelungsgüte angibt, ist gleich aufgebaut, wie in (6.3-85) mit den Wichtungsmatrizen $\mathbf{Q}$ und $\mathbf{R}$. Es wird angenommen, dass alle Zustände gemessen werden und der Rückführung zur Verfügung stehen. Die Ausgänge

$$\mathbf{y} = \mathbf{\Xi}\mathbf{x} \in \mathbb{R}^p, \tag{6.3-109}$$

sollen einer gegeben, gültigen Referenz $\mathbf{r} \in \mathbb{R}^p$ folgen. Die Matrix $\mathbf{\Xi} \in \mathbb{R}^{p\times n}$ wählt die Zustände, die auf eine Referenz geregelt werden sollen. Der Regler wird für eine Linearisierung von (6.3-107) um einen Arbeitspunkt $(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}})$ entworfen. Wenn sich die Referenz oder die Störgrößen ändern, erfolgt eine Änderung des Arbeitspunktes. In diesem Fall wird eine neue Linearisierung bestimmt und der Regler aktualisiert.

Um die in Abschnitt 6.3.1 vorgestellte Methode für MTI Systeme anwenden zu können, muss das MTI System um einen Arbeitspunkt linearisiert werden. Der Arbeitspunkt wird durch die aktuellen Werte der Referenz $\mathbf{r}$ und der Störgrößen $\mathbf{d}$ bestimmt. Der Arbeitspunkt $(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}})$ wird berechnet, indem die rechte Seite der Zustandsgleichung (6.3-107) gleich Null gesetzt wird

$$\langle\, \mathsf{F} \mid \mathsf{M}\,(\mathbf{x}, \mathbf{u}, \mathbf{d})\, \rangle = 0. \tag{6.3-110}$$

Um die Referenz, die Störgrößen und die Eingangsbeschränkungen bei der Berechnung des Arbeitspunktes zu berücksichtigen, sollen bei der Berechnung von (6.3-110) folgende Nebenbedingungen erfüllt

sein

$$\mathbf{\Xi}\bar{\mathbf{x}} = \mathbf{r}, \bar{\mathbf{d}} = \mathbf{d}(t), \tag{6.3-111}$$

$$\mathbf{u}_{min} \leq \bar{\mathbf{u}} \leq \mathbf{u}_{max}. \tag{6.3-112}$$

Dies bedeutet, dass der Arbeitspunkt auf der aktuellen Referenz und Störung und innerhalb der Eingangsbeschränkungen liegen soll. Mit dem so bestimmten Arbeitspunkt kann die Linearisierung von (6.3-107) durch (6.3-99) und (6.3-100) erfolgen, welches die linearen Systemmatrizen $\mathbf{A}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}})$ und $\mathbf{B}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}})$ ergibt. Die Eingangsmatrix kann aufgeteilt werden, um die verschiedenen Einflüsse von Stell- und Störgrößen zu berücksichtigen

$$\mathbf{B}_u(:, i) = \mathbf{B}(:, i),\ i = 1, \ldots, m, \tag{6.3-113}$$

$$\mathbf{B}_d(:, i) = \mathbf{B}(:, m + i),\ i = 1, \ldots, m_d. \tag{6.3-114}$$

Um den Regler zu entwerfen, müssen die Matrizen $\mathbf{Q}$ und $\mathbf{R}$ eingestellt werden. Die Wichtungsmatrix $\mathbf{Q}$ wird zu Beginn gleich

$$\mathbf{Q} = \mathbf{\Xi}^T\mathbf{\Xi}, \tag{6.3-115}$$

gewählt, sodass die Ausgänge, die einer Referenz folgen sollen, gewichtet werden. Der Stellaufwand muss auch bestraft werden mit der Wichtung $\mathbf{R}$, damit die Stellgrößen nicht die von der Anlage gegebenen Stellgrößenbeschränkungen überschreiten. Mit den Systemmatrizen $\mathbf{A}$, $\mathbf{B}_u$, $\mathbf{B}_d$ und den Wichtungsmatrizen $\mathbf{Q}$ und $\mathbf{R}$ kann der dezentrale Regler durch das Lösen von (6.3-89) für einen gegeben Wert γ berechnet werden.

Vorverarbeitung

Die Besetzungsstruktur der Rückführverstärkung soll während des Betriebes konstant sein, d.h. dass die Kommunikationsstruktur sich nicht ständig ändern soll. Somit ist vor dem Betrieb des Reglers eine Analyse der Systemstruktur notwendig. Dies wird im Vorverarbeitungsschritt durchgeführt mit dem Ziel eine Besetzungsstruktur für $\mathbf{K}$ zu finden, sodass der Verlust der Regelungsgüte gering ist im Vergleich zu dem zentralen Regler (6.3-86). Der Verlust der Regelungsgüte wird durch

$$\Delta J = \frac{|J_d - J_c|}{J_c} \cdot 100\%, \tag{6.3-116}$$

bewertet, wobei der Wert der Kostenfunktion (6.3-87) von dem dezentralen Regler J_d mit dem des zentralen J_c verglichen wird. Allgemein sinkt die Regelungsgüte mit geringerer Besetzungsdichte der Verstärkungsmatrix. Eine Struktur mit minimalem Kommunikationsaufwand ist hier gesucht, bei der der Verlust an Regelungsgüte noch akzeptabel ist $\Delta J < \delta$, d.h. kleiner als ein bestimmter Grenzwert δ. Da hier kein lineares, sondern ein multilineares System betrachtet wird, soll diese Begrenzung des Güteverlusts für die Struktur mit minimalem Kommunikationsaufwand für jeden Arbeitspunkt in dem Arbeitsbereich der entsprechenden Anlage gelten.

Zur Bestimmung der Struktur ist das Optimierungsproblem (6.3-89) zu lösen. Die Struktur kann nicht direkt vorgegeben werden, wird jedoch durch die Wahl des Wichtungsfaktors γ für die Besetzung beeinflusst. Die Wahl eines bestimmten Wertes für γ führt zu einer gewissen Struktur der Verstärkungsmatrix, wobei die Kommunikationskomplexität mit steigendem γ geringer wird. Um eine Reglerstruktur zu finden, die die Gütebeschränkung für alle Arbeitspunkte erfüllt, wird hier ein heuristischer Ansatz gewählt, bei dem über den Faktor γ für eine ausreichende Anzahl an gültigen Arbeitspunkten iteriert wird. Die Besetzungsstruktur $\mathcal{S}$ von $\mathbf{K}$ mit minimalem Kommunikationsaufwand und einem Verlust in der Regelungsgüte, der geringer als δ für alle untersuchten Arbeitspunkte ist, wird gewählt. Durch diesen heuristischen Ansatz wird im Allgemeinen nicht das globale Optimum dieses multikriteriellen Optimierungsproblems gefunden, es wird jedoch angenähert. Das Vorgehen im Vorverarbeitungsschritt ist im Flussdiagramm in der Abbildung 6.3-27 dargestellt.

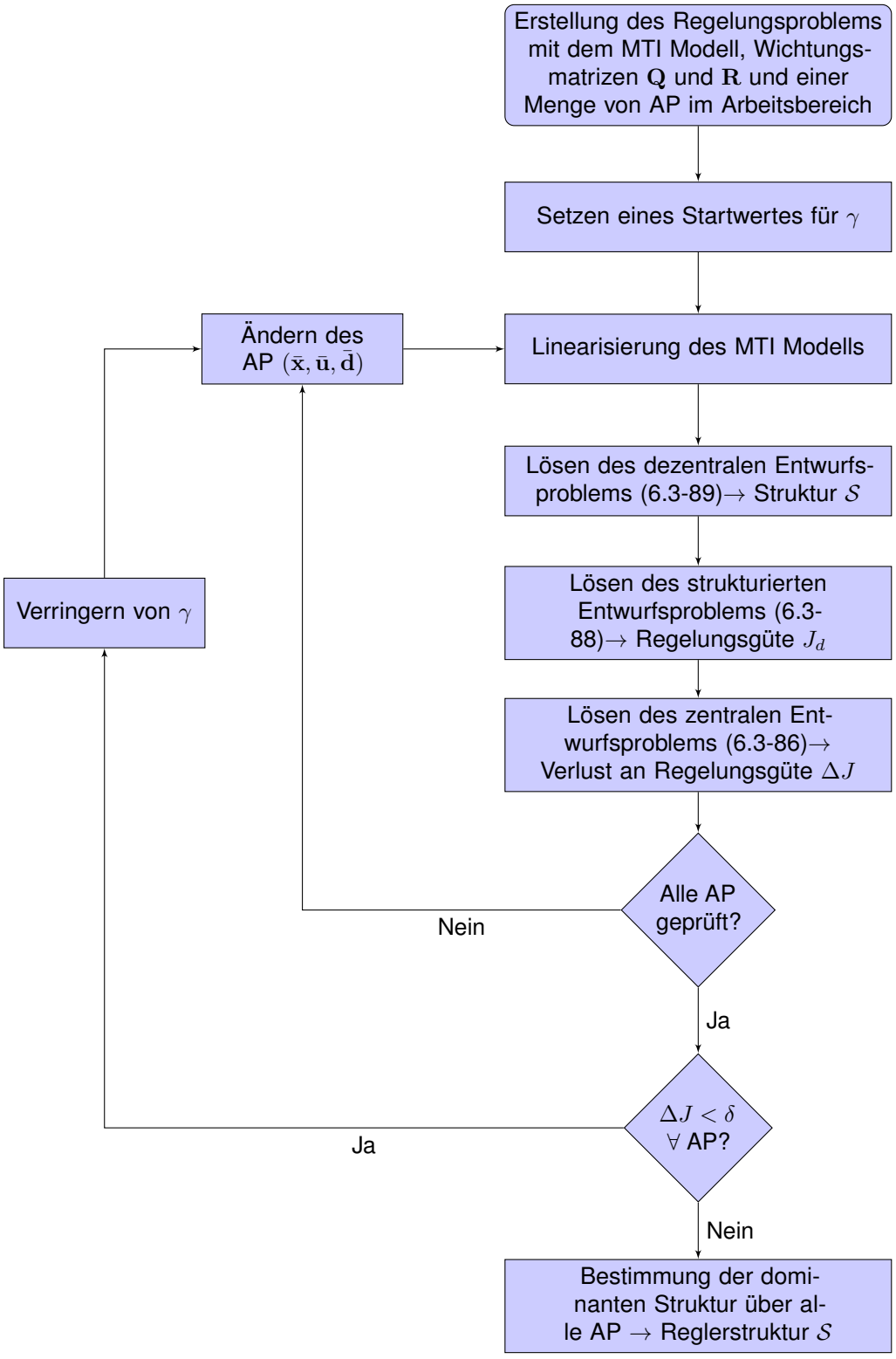


Abbildung 6.3-27: Flussdiagramm des Vorverarbeitungsschrits

Betriebsphase

Während des Betriebes des Reglers soll sich die zuvor bestimmte Struktur $\mathcal{S}$ nicht mehr ändern. Die Nichtnull Werte der Rückführverstärkung $\mathbf{K}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}})$ sind abhängig von den Arbeitspunkten und sind nur in einer Umgebung um den jeweiligen Arbeitspunkt gültig. Ändert sich die Referenz $\mathbf{r}$, führt dies zu

einer Änderung des Arbeitspunktes. Ein neuer Arbeitspunkt wird mit (6.3-110) für die aktuelle Referenz berechnet. Außerdem wird der Arbeitspunkt aktualisiert, wenn sich das Störsignal ändert. Ein neuer Arbeitspunkt mit einem neuen Wert für $\bar{\mathbf{d}}$ wird berechnet, wenn sich die Störgröße mehr als $\Delta\mathbf{d} \in \mathbb{R}^{m_d}$ ändert

$$\bar{d}_i = \begin{cases} d_i(t) & \text{, für } \left|d_i(t) - \bar{d}_i\right| > \Delta d_i \\ \bar{d}_i & \text{, sonst.} \end{cases}, i = 1, \ldots, m_d. \tag{6.3-117}$$

Nach der Bestimmung eines neuen Arbeitspunktes, wird die Linearisierung durch (6.3-99) und (6.3-100) neu bestimmt. Die Linearisierung kann sehr effizient berechnet werden, da in der Tensordarstellung nur ein kontrahiertes Produkt ausgewertet werden muss. Dies vereinfacht sich weiter, wenn die Tensoren als dekomponierte Tensoren vorliegen, was hier möglich ist, wie in Kapitel 6.1.3 gezeigt. Durch das das Lösen von (6.3-88) mit der aktuellen Linearsierung des MTI Systems und der im Vorverarbeitungsschritt bestimmten Struktur $\mathcal{S}$ wird die Rückführverstärkung aktualisiert. Nach der Berechnung der arbeitspunktabhängigen Verstärkungsmatrix $\mathbf{K}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}})$ wird die Stellgröße

$$\mathbf{u} = -\mathbf{K}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}})(\mathbf{x} - \bar{\mathbf{x}}) + \bar{\mathbf{u}}, \tag{6.3-118}$$

an das System gegeben. Der geschlossene Kreis mit dem spärlich besetzten Regler und dem MTI System ist in Abbildung 6.3-28 gezeigt.

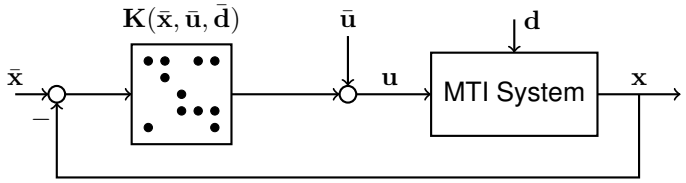


Abbildung 6.3-28: Dezentrale Zustandsrückführung mit spärlich besetzter Verstärkung $\mathbf{K}$

Der Regelungsalgorithmus der dezentralen Zustandsrückführung für MTI Systeme während des Betriebes ist in der Abbildung 6.3-29 in einem Flussdiagramm zusammengefasst.

6.3.4 Anwendungsbeispiel

Das entworfene Verfahren zur dezentralen Zustandsrückführung für MTI Systeme soll hier auf das Beispiel einer großen Heizungsanlage angewendet werden, die durch ein multilineares Modell abgebildet werden kann. Es wird eine Heizkreisstruktur betrachtet, die typisch für Nichtwohngebäude ist. Für die Wärmeversorgung der 7 Heizkreise sind 2 Kessel vorgesehen. Jeder Heizkreis wird als einzelner Verbraucher modelliert. Das Verhalten der einzelnen Komponenten wird über Wärmeleistungsbilanzen abgebildet. Räume in den jeweiligen Verbrauchern werden alle zu einem zusammengefasst und es wird angenommen, dass die Luft vollständig verteilt mit der Gebäudetemperatur $T_{g,i}$ ist. Man erhält somit ein Ein-Zonen Modell für jeden Verbraucher. Der Wärmebedarf der Verbraucher wird über Radiatoren abgedeckt. Der Wärmeübergang von den Heizkörpern zu der Luft in der Zone wird proportional zu der Differenz zwischen Raumtemperatur und Rücklauftemperatur $T_{Rl,i}$ des Heizkörpers mit dem Wärmeübergangskoeffizienten $k_{rg,i}$ angenommen. Die thermischen Verluste an die Umgebung mit der Außentemperatur T_a werden proportional zu der Differenz zwischen Gebäude- und Umgebungstemperatur mit dem Wärmeübergangskoeffizient $k_{ga,i}$ angegeben. Über die Wärmeleistungsbilanz ergibt sich für die Raum- bzw. Gebäudetemperaturen der 7 Verbraucher

$$\dot{T}_{g,i} = \frac{k_{rg,i}}{C_{g,i}}(T_{Rl,i} - T_{g,i}) - \frac{k_{ga,i}}{C_{g,i}}(T_{g,i} - T_a),\ i = 1, \ldots, 7, \tag{6.3-119}$$

wobei $C_{g,i}$ die Wärmekapazitäten der Heizkreise beschreibt. Der Wärmeübergang zwischen den Zonen wird hier vernachlässigt.
Die Verbraucher werden über einen Volumenstrom, der von den Kesseln kommt, mit warmem Wasser versorgt. Es wird angenommen, dass die Kessel baugleich und hydraulisch abgeglichen sind, sodass

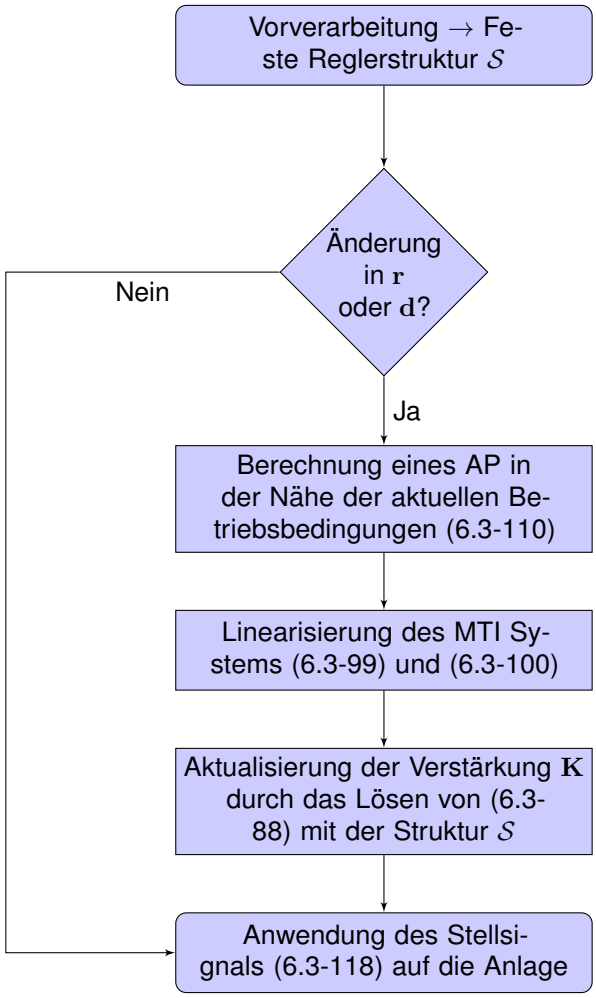


Abbildung 6.3-29: Flussdiagramm des dezentralen Zustandsrückführungsreglers in jedem Abtastschritt während des Betriebs

sie beide mit dem gleichen Volumenstrom durchflossen werden. Die gesamte Vorlauftemperatur, die die Verbraucher erreicht, ist die Mischtemperatur der Vorläufe beider Kessel $T_{Vl,i},\ i = 1, 2$

$$T_{Vl,ges} = 0.5 \cdot (T_{Vl,1} + T_{Vl,2})\,. \tag{6.3-120}$$

Jeder Verbraucher besteht aus einer Pumpe und einem 3-Wege Ventil zur Rücklaufbeimischung, um die Vorlauftemperatur an den jeweiligen Verbraucher anzupassen. Die Struktur ist in Abbildung 6.3-30 dargestellt.

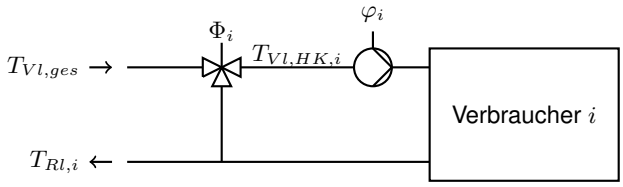


Abbildung 6.3-30: Aufbau der Heizkreise $i = 1, \ldots, 7$

Die 3-Wege Ventile werden durch Signale $\Phi_i \in [0, 1]$ angesteuert, sodass die Heizkörper mit der

Mischtemperatur

$$T_{Vl,HK,i} = \Phi_i T_{Rl,i} + (1 - \Phi_i) T_{Vl,ges} \tag{6.3-121}$$

versorgt werden. Die Flüsse in der Verbraucherkreisen

$$\dot{V}_{HK,i} = \varphi_i \dot{V}_{max,i} \tag{6.3-122}$$

werden über Pumpen bestimmt, abhängig vom Eingangssignal $\varphi_i \in [0, 1]$ und dem maximalen Fluss $\dot{V}_{max,i}$. Wärme wird an das Gebäude abgegeben, sodass kühleres Wasser die Heizkörper mit der Rücklauftemperatur

$$\dot{T}_{Rl,i} = \varphi_i \frac{\dot{V}_{max,i}}{V_{HK,i}} \left(T_{Vl,HK,i} - T_{Rl,i}\right) - \frac{k_{rg,i}}{c\rho V_{HK,i}} \left(T_{Rl,i} - T_{g,i}\right), \tag{6.3-123}$$

verlässt, wobei $V_{HK,i}$ das Volumen des Wassers im Heizkreis mit der Dichte ρ und der spezifischen Wärmekapazität c ist.
Die Kessel versorgen die Verbraucher mit Wärme. Beide Kessel haben ein Modulationssignal $\alpha_i \in [0, 1]$ mit $i = 1, 2$ als Eingang, über das ihre thermische Leistung $P_{in,i} = \alpha_i P_{max,i}$ eingestellt wird, wobei $P_{max,i}$ die maximale Kesselleistung ist. Es wird angenommen, dass die Volumenströme durch die Kessel genau den Flussanforderungen der Heizkreise entsprechen. Beide Kessel werden mit dem gleichen Volumenstrom durchflossen. Jeden Kessel erreicht somit die Hälfte des gesamten Volumenstroms $\dot{V}_{ges}$, der von den Verbrauchern kommt. Die Flüsse auf der Erzeugerseite der 3-Wege Ventile der Verbraucherkreise sind an einen Rücklaufsammler angeschlossen, sodass der gesamte Fluss

$$\dot{V}_{ges} = \sum_{j=1}^{7} (1 - \Phi_j) \dot{V}_{HK,j}, \tag{6.3-124}$$

mit der Temperatur

$$T_{Rl,ges} = \frac{1}{\dot{V}_{ges}} \sum_{j=1}^{7} T_{Rl,j} (1 - \Phi_j) \dot{V}_{HK,j} \tag{6.3-125}$$

als Rücklauf die Kessel erreicht. Das Aufstellen der Wärmeleistungsbilanzen der Kessel mit Volumen $V_{K,i}$ ergibt die Differentialgleichung für die Kesselvorlauftemperatur $T_{Vl,i}$ zu

$$\dot{T}_{Vl,i} = \frac{1}{2V_{K,i}} \dot{V}_{ges} \left(T_{Rl,ges} - T_{Vl,i}\right) + \alpha_i \frac{P_{max,i}}{c\rho V_{K,i}}, \; i = 1, 2. \tag{6.3-126}$$

Einen Überblick über die Anlage zeigt die Abbildung 6.3-31.

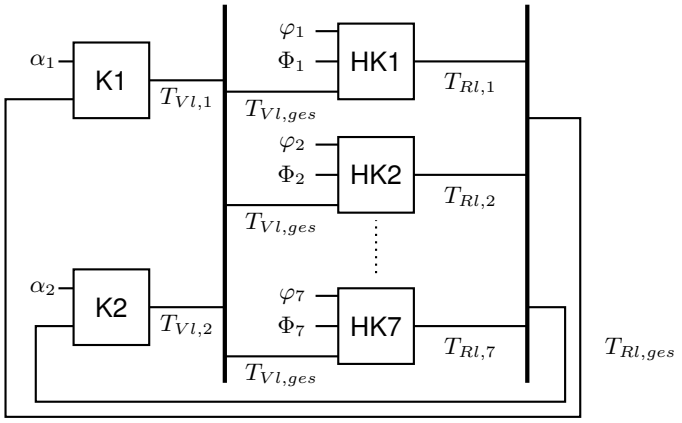


Abbildung 6.3-31: Schema des Heizungssystems mit 2 Kesseln (K) und 7 Heizkreisen (HK)

Die Gleichungen (6.3-119) bis (6.3-126) ergeben ein System von Differentialgleichungen, die die Dynamiken des gesamten Heizungssystems mit Erzeugern und Verbrauchern beschreiben. Die rechte Seite des Gleichungssystems ist multilinear, sodass das Zustandsraummodell zu der Klasse der MTI Systeme gehört und in einer Tensorstruktur

$$\dot{\mathbf{x}} = \langle \, \mathsf{F} \mid \mathsf{M}\,(\mathbf{x}, \mathbf{u}, d) \, \rangle \, . \tag{6.3-127}$$

dargestellt werden kann. Die Zustände des Systems sind die Vorlauftemperaturen der Kessel und die Rücklauf- sowie die Raumtemperaturen der Verbraucher, sodass sich ein Zustandsvektor

$$\mathbf{x} = \begin{pmatrix} T_{Vl,1} & T_{Vl,2} & T_{g,1} & T_{Rl,1} & \cdots & T_{g,7} & T_{Rl,7} \end{pmatrix}^T \in \mathbb{R}^{16} \tag{6.3-128}$$

ergibt. Die Stellgrößen sind die Modulationssignale der Kessel sowie die Stellsignale der Pumpe und Ventile der Verbraucher, die in dem Eingangsvektor

$$\mathbf{u} = \begin{pmatrix} \alpha_1 & \alpha_2 & \varphi_1 & \Phi_1 & \cdots & \varphi_7 & \Phi_7 \end{pmatrix}^T \in \mathbb{R}^{16} \tag{6.3-129}$$

zusammengefasst werden. Die Außentemperatur ist die Störgröße $d = T_a$ des Systems. Der Parametertensor F wird als CP Tensor angegeben.

Das Ziel des Reglers ist zum einen die Vorlauftemperaturen der Kessel auf eine bestimmte Solltemperatur einzustellen. In Standardanwendungen wird die Sollvorlauftemperatur $T_{Vl,i,ref}$ in linearer Abhängigkeit von der Außentemperatur über eine Heizkurve bestimmt

$$T_{Vl,i,ref} = \beta_i T_a + \eta_i, \; i = 1, 2, \tag{6.3-130}$$

sodass die Vorlauftemperaturen bei niedriger Außentemperatur hoch und bei hoher Außentemperatur niedrig sind. Die Ventile und Pumpen dienen dazu den jeweiligen Heizkreis an den Erzeugerkreis anzupassen und werden so geregelt, dass die Raumtemperaturen ihrer Referenz $T_{g,i,ref}$ folgen. Daher erhält man den Referenzvektor

$$\mathbf{r} = \begin{pmatrix} T_{Vl,1,ref} & T_{Vl,2,ref} & T_{g,1,ref} & \cdots & T_{g,7,ref} \end{pmatrix}^T . \tag{6.3-131}$$

Hier wurden die Referenz für Räume konstant auf $21\,^\circ\mathrm{C}$ gesetzt. Beide Kessel haben die gleichen Heizkurven (6.3-130). Die Eingänge der Anlage, d.h. die Stellsignale der Kessel auf der Erzeugerseite

$$\alpha_i \in [0, 1], \; i = 1, 2,$$

und die Stellsignale der Pumpen und Ventile auf der Verbraucherseite

$$\varphi_i \in [0, 1], \; \Phi_i \in [0, 1], \; i = 1, \ldots, 7,$$

sind begrenzt. Für den Entwurf des Reglers muss im ersten Schritt die Struktur der Rückführverstärkung und somit auch die notwendige Kommunikationsstruktur der Anlage bestimmt werden. Dafür wird in der Vorverarbeitungsphase das Optimierungsproblem (6.3-89) für verschiedene Arbeitspunkte gelöst. Die hier betrachteten Arbeitspunkte hängen nur von der Außentemperatur ab. Für eine bestimmte Außentemperatur folgt die Referenz für die Vorlauftemperatur aus der Heizkurve (6.3-130). Die Außentemperatur ist zudem die Störung. Die Referenztemperatur der Verbraucher ist konstant vorgegeben. Alle anderen Zustände und Eingänge der jeweiligen Arbeitspunkte werden durch das Lösen von (6.3-110) bestimmt. Um mögliche Arbeitspunkte zu bestimmen, wurde die Außentemperatur über das Intervall $T_a \in [-10\,^\circ\mathrm{C}, 10\,^\circ\mathrm{C}]$ iteriert. Der dezentrale Regler wird durch Lösen von (6.3-89) für Linearisierungen (6.3-99) und (6.3-100) an verschiedenen Arbeitspunkten und für verschiedene Werte von γ bestimmt. Der Verlust an Regelungsgüte für verschiedene Werte von γ und T_a ist in Abbildung 6.3-32 dargestellt.

Die Abbildung zeigt, dass der Verlust an Regelungsgüte mit größer werdendem γ ansteigt, da der Fokus immer mehr auf eine spärliche Besetzung der Rückführverstärkung gesetzt wird. Bis zu einem Wert für γ von $1 \cdot 10^{-5}$ ist der Verlust an Regelungsgüte sehr klein. Für größere Werte steigt der Verlust stark an, sodass dieser Wert hier verwendet werden soll. Es ist in der Grafik deutlich zu erkennen, dass die Regelungsgüte nicht signifikant vom gewählten Arbeitspunkt abhängt, sodass der selbe Wert für γ für jeden Arbeitspunkt verwendet werden kann. Als nächstes ist die Frage zu klären, welche Struktur sich

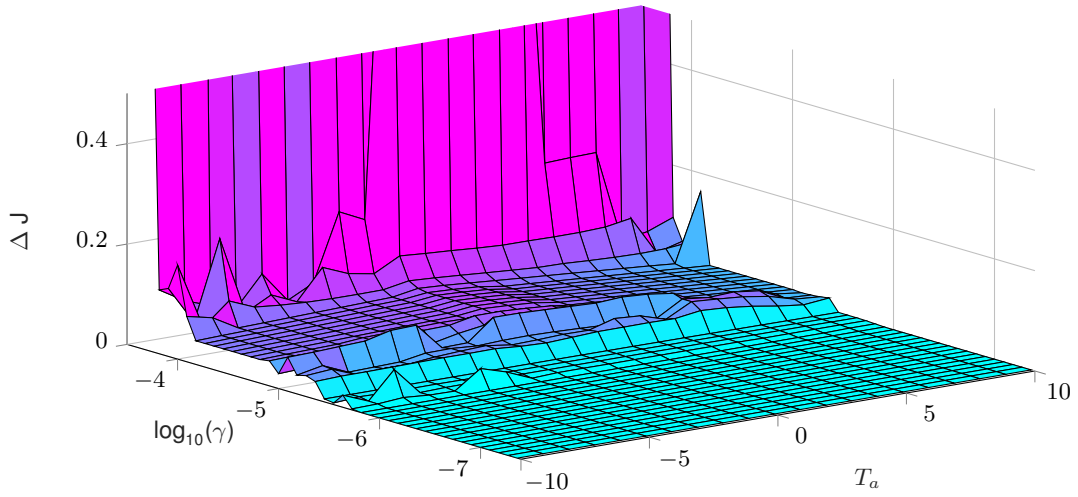


Abbildung 6.3-32: Verlust an Regelungsgüte des dezentralen Reglers

aus dieser Wahl von γ ergibt. In diesem Beispiel ergibt sich die in der Abbildung 6.3-33 dargestellte Struktur für alle Arbeitspunkte. Sie wird für den Betrieb der Reglers verwendet.

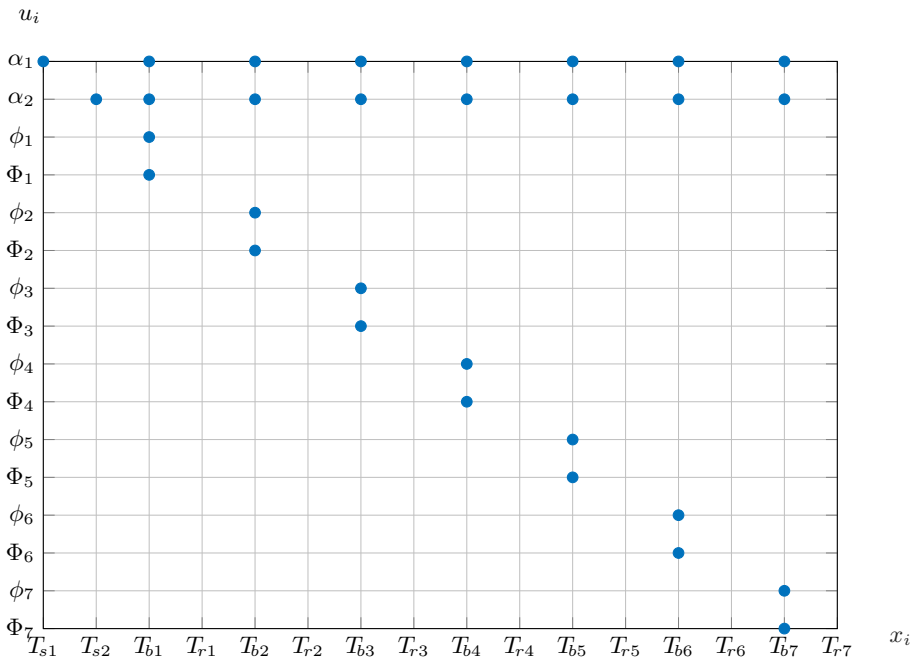


Abbildung 6.3-33: Besetzungsstruktur der Verstärkungsmatrix $\mathbf{K}$

Im Gegensatz zum zentralen Entwurf mit $16 \cdot 16 = 256$ Einträgen sind im dezentralen Design nur 30 Nicht-Null Einträge in der Rückführverstärkung nötig. Dies führt zu einem deutlich geringeren Kommunikationsaufwand. Trotzdem ist die Regelungsgüte nur geringfügig niedriger. Die automatisch über die Lösung eines Optimierungsproblems bestimmte Struktur zeigt, dass nur die Regler der Kessel Informationen über das Gesamtsystem benötigen, d.h. die Vorlauftemperaturen und die Raumtemperaturen der Verbraucher. Die Regler der Pumpen und der 3-Wege Ventile der Verbraucher benötigen nur lokale Informationen der Verbraucher, d.h. ihre eigene Raumtemperatur $T_{g,i}$. Die lokalen Regler der Verbraucher benötigen somit nicht die Informationen des Gesamtsystems, wie die Raum- oder Rücklauftemperaturen der anderen Verbraucher, was zu einer deutlich einfacheren Kommunikationsinfrastruktur im Vergleich zu dem zentralen Design führt. Im Fall des zentralen Entwurfs müsste der Regler Zugriff auf alle Mess- und

Stellsignale haben. Dies ist im dezentralen nicht nötig, wie die resultierende Struktur mit der Aufteilung auf mehrere Reglerknoten in der Abbildung 6.3-34 zeigt.

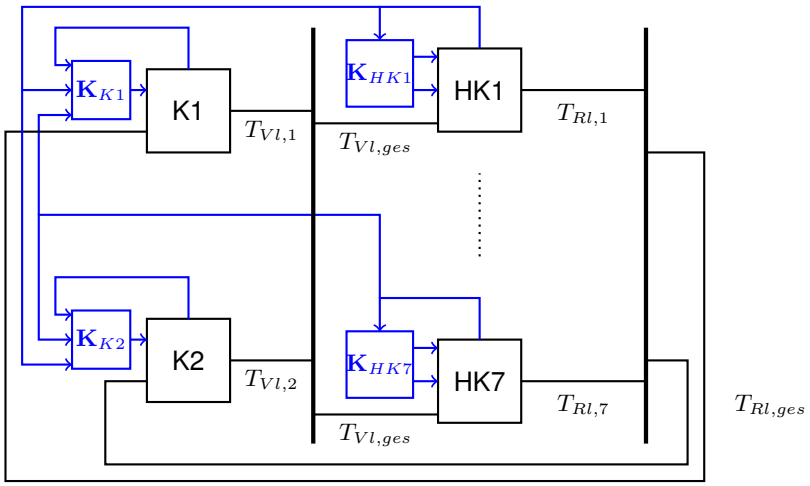


Abbildung 6.3-34: Dezentrale Reglerstruktur an der Anlage

Die Ergebnisse der Simulation der Anlage im geschlossenen Kreis für einen Tag ist für die Vorlauftemperatur in Abbildung 6.3-35 dargestellt.

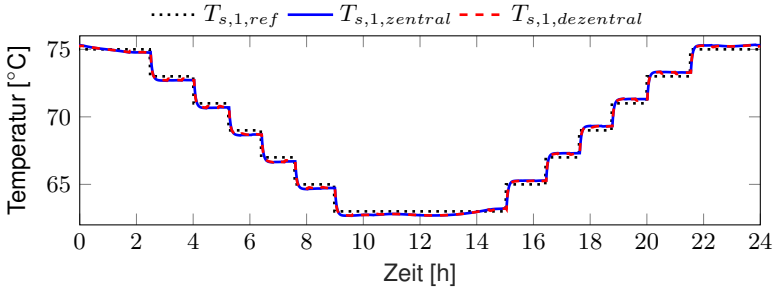


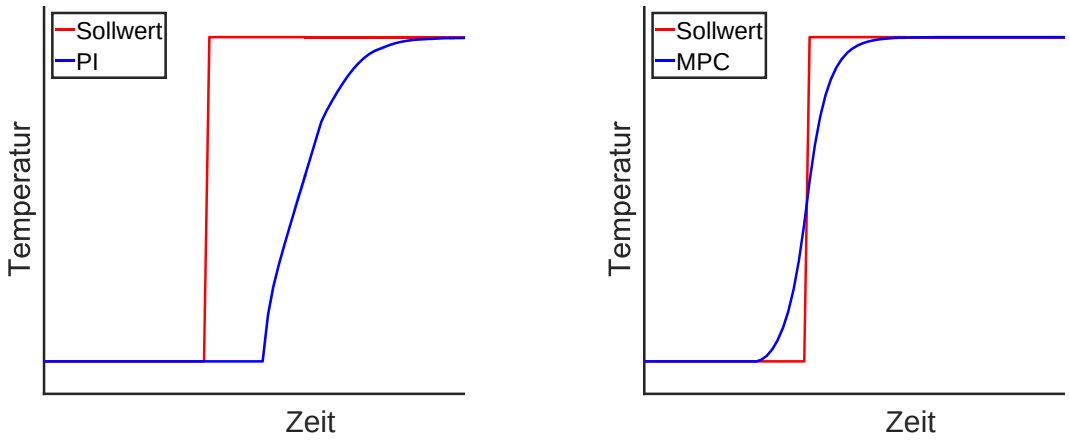
Abbildung 6.3-35: Vergleich der Simulation im geschlossenen Kreis für das zentrale und dezentrale Design

Obwohl viele Einträge in der Rückführmatrix auf Null gesetzt wurden zeigt die Simulation das die Temperaturen ihrer Referenz folgen, wie hier für den Vorlauf des ersten Kessels dargestellt. Ein Vergleich mit einem zentralen Regler zeigt keine großen Unterschiede. Die Vorlauftemperatur des zweiten Kessels zeigt gleiche Ergebnisse. Die Raumtemperaturen erreichen den Referenzwert mit einer maximalen Abweichung von $\pm 0.2\,\mathrm{K}$, was deutlich in den akzeptablen Grenzen liegt. Somit ergibt sich ein gutes Regelungsergebnis. Jede Änderung in der Referenz und der der Störgröße um mehr als $0.5\,\mathrm{K}$ führt zu einer Anpassung der Verstärkungsmatrix, sodass der Regler gute Ergebnisse liefert, obwohl die Strecke nicht linear sondern multilinear ist.

6.4 Adaptiver prädiktiver Regelungsentwurf mit sukzessiver Linearisierung

Aufgrund der Möglichkeiten Mehrgrößensysteme unter Einbeziehung von Nebenbedingungen und Störungen optimal zu regeln, ist die modellprädiktive Regelung (MPC, model predictive control) eine beliebte Regelungsmethode gerade für Systeme mit Verzögerungen oder großen Zeitkonstanten. Einen beispielhaften Vergleich von einer Standardregelung, wie einem PI Regler und einem MPC zeigt die Abbildung 6.4-36.

Bei der in der Abbildung dargestellten Regelungsaufgabe, soll eine Temperatur einem Sollwert folgen. Vereinfacht wird angenommen, dass sich die Regelstrecke durch ein System 1. Ordnung abbilden lässt,



(a) PI Regler

(b) MPC Regler

Abbildung 6.4-36: Beispielhafter Vergleich zwischen PI und MPC Regler

das zusätzlich eine Totzeit hat. Diese Zeitverzögerung ist bei der PI-Regelung deutlich zu erkennen, da das Ausgangssignal erst verspätet auf die Änderung im Sollwert reagiert. Dies ist typisch für die PI Regelung in einem solchen Fall. Bei der prädiktiven Regelung ist die Information über die Zeitkonstanten und mögliche Totzeiten in Form von einem Modell der Strecke im Regler hinterlegt. Dies wird darin ersichtlich, dass der prädiktive Regler bereits vor dem eigentlichen Anstieg des Sollwertes eine Reaktion zeigt und somit eine deutlich bessere Sollwertfolge aufweist.

Innerhalb des MPC Reglers wird ein Modell benutzt, um das zukünftige Verhalten der Anlage vorherzusagen. Häufig werden lineare Modelle dafür verwendet. Daraus ergibt sich der Vorteil, dass das resultierende MPC Optimierungsproblem konvex ist und somit effizient durch Standardmethoden, wie dem Inneren-Punkte Verfahren gelöst werden kann, [6]. Dadurch ist dieser Ansatz auch für eine Echtzeitimplementierung gut geeignet, [18]. Im Allgemein ist das Verhalten der Anlagen jedoch nichtlinear. Ihre Dynamiken werden durch ein lineares Modell im Bereich eines Arbeitspunktes nur angenähert. Somit bildet das lineare Modell das Systemverhalten nur in der Nähe des Arbeitspunktes mit einer ausreichenden Genauigkeit ab. Entfernt man sich vom Arbeitspunkt nimmt die Modellgenauigkeit ab. Sind die nichtlinearen Effekte entscheidend reicht die Beschreibung mit einem linearen Modell nicht aus. In diesem Fall können auch allgemeine nichtlineare Modelle verwendet, was jedoch zu einem deutlich höheren Berechnungsaufwand führen kann. In [40] wurde gezeigt, dass Heizungssysteme sehr gut durch MTI Modelle beschrieben werden können. Daher sollen hier im folgenden MTI Modelle im MPC verwendet werden. Es kann jedoch im Arbeitspaket B.4 gezeigt werden, dass die positiven Eigenschaften des Optimierungsproblems, d.h. die Konvexität, bei der Verwendung von linearen Modellen, nicht mehr allgemein gegeben ist, wenn MTI Modelle verwendet werden. Dies kann zu langen Berechnungszeiten bei der Lösung führen. Zudem ist es nicht garantiert, dass ein globales Minimum gefunden wird. Die große Komplexität kann für die Implementierung im Echtzeitbetrieb problematisch werden, da in diesem Fall sichergestellt sein muss, dass das Ergebnisse der Optimierung am Ende jedes Abtastzeitschrittes vorliegt. Ist dies nicht der Fall kann der vorgegebene Takt nicht eingehalten werden. Daher soll in diesem Kapitel eine Methode vorgestellt werden, die die guten Modellierungseigenschaften des MTI Systems und die Konvexität des Optimierungsproblems verbindet.

6.4.1 Lineare modellprädiktive Regelung

Die modellprädiktive Regelung ist eine bekannte modellbasierte Reglerentwurfsmethode. Innerhalb des Reglers wird ein Modell der Regelstrecke verwendet, um das Systemverhalten auf zukünftige Eingänge vorherzusagen. Mithilfe des Modells wird die optimale zukünftige Trajektorie der Stellsignale in jedem Abtastzeitschritt durch die Minimierung einer Kostenfunktion berechnet. Die Kostenfunktion gibt das gewünschte Systemverhalten, z.B. das Folgen einer Referenz, an. Die Grundidee der prädiktiven Regelung wird hier nach [34] eingeführt. Der Regelkreis ist in der Abbildung 6.4-37 gezeigt.

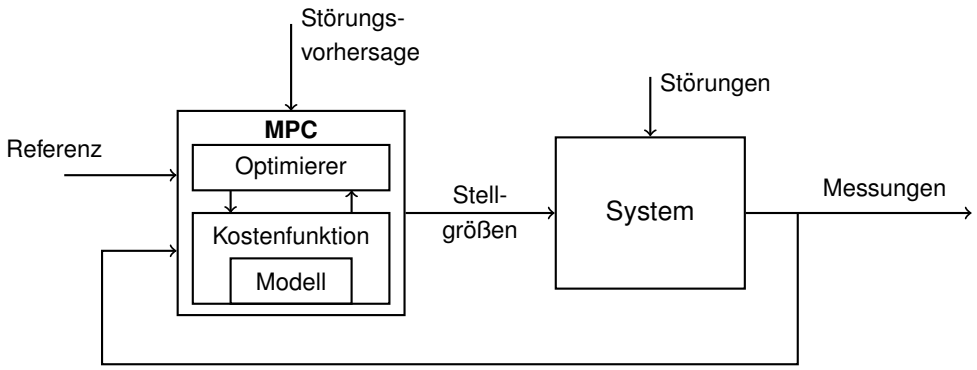


Abbildung 6.4-37: Regelkreis für MPC

Wie in der Abbildung deutlich wird, beeinflussen die Stellgrößen $\mathbf{u}$ und die Störgrößen $\mathbf{d}$ das System. Messsignale $\mathbf{y}$ aus der Anlage sind dem Regler verfügbar. Der MPC berechnet eine optimale Eingangssequenz mit einer Länge von H_u Zeitschritten, dem sog. Stellhorizont, durch die Minimierung einer Kostenfunktion $J(k)$ durch einen Optimierer

$$\min_{\substack{\mathbf{u}(k+i),\\ i=0,\dots,H_u-1}} J(k).$$

Das zukünftige Systemverhalten auf verschiedene Stellgrößen wird für H_p Zeitschritte, dem Vorhersagehorizont, durch Vorwärtssimulation mit dem Modell bestimmt und mithilfe der Kostenfunktion bewertet. Aus der berechneten Stellgrößenfolge

$$\hat{\mathbf{U}} = \begin{pmatrix} \hat{\mathbf{u}}(k) & \hat{\mathbf{u}}(k+1) & \cdots & \hat{\mathbf{u}}(k+H_u-1) \end{pmatrix}$$

für den Stellhorizont wird nur das Eingangssignal $\hat{\mathbf{u}}(k)$ des ersten Zeitschritts an die Anlage gegeben. Im nächsten Zeitschritt erfolgt die Optimierung erneut.

In der MPC Formulierung sind verschiedene Kostenfunktionen und Modelle möglich. Im Folgenden soll hier die Standardformulierung der prädiktiven Regelung eingeführt werden, wie sie häufig Anwendung findet. Dabei wird ein lineares Modell und eine quadratische Kostenfunktion für die Referenzfolge verwendet, [34]. Das Systemverhalten wird durch ein lineares zeitdiskretes Modell

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) + \mathbf{B}_d\mathbf{d}(k), \tag{6.4-132}$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k). \tag{6.4-133}$$

mit der Störgröße $\mathbf{d} \in \mathbb{R}^{m_d}$ angenähert. Der Regler optimiert die Eingangsfolge, sodass die Kostenfunktion

$$J(k) = \sum_{i=1}^{H_p} \|\hat{\mathbf{y}}(k+i) - \mathbf{r}(k+i)\|_{\mathbf{Q}}^2 + \sum_{i=0}^{H_u-1} \|\Delta\mathbf{u}(k+i)\|_{\mathbf{R}}^2 \tag{6.4-134}$$

mit den Änderungen des Stellsignals $\Delta\mathbf{u}(k+i) = \Delta\mathbf{u}(k+i) - \Delta\mathbf{u}(k+i-1)$ von einem Zeitschritt zum nächsten und einer Referenz $\mathbf{r} \in \mathbb{R}^{p\times 1}$ minimiert wird. Der Ausgang $\hat{\mathbf{y}}(k+i),\ i = 1,\dots,H_p$ wird mithilfe des Modells vorhergesagt. Dies bedeutet, dass das Eingangssignal so bestimmt werden soll, dass die Ausgänge einer Referenz mit einem gewissen Regelungsaufwand folgen. Die verwendete Norm ist beispielhaft für den ersten Term gegeben durch

$$\|\mathbf{y}(k+i) - \mathbf{r}(k+i)\|_{\mathbf{Q}} = \left(\hat{\mathbf{y}}(k+i) - \mathbf{r}(k+i)\right)^T \mathbf{Q} \left(\hat{\mathbf{y}}(k+i) - \mathbf{r}(k+i)\right). \tag{6.4-135}$$

Die Wichtungsmatrizen $\mathbf{Q} \in \mathbb{R}^{p\times p}$ und $\mathbf{R} \in \mathbb{R}^{m\times m}$ mit $\mathbf{Q} \geq 0$ und $\mathbf{R} \geq 0$ werden jeweils verwendet, um die Abweichung der Ausgänge von der Referenz sowie die Änderungen der Stellsignale zu gewichten. Durch eine Erhöhung von $\mathbf{Q}$ ist die Gewichtung mehr auf der Referenzfolge, was im Allgemeinen zu einem größeren Stellaufwand führt. Erhöht man $\mathbf{R}$ ergibt sich eine geringere Änderung in den Stellsignalen,

was jedoch oft zu einem langsameren System führt. Somit ist das Einstellen der Gewichte ein Trade-off zwischen Stellaufwand und Güte der Referenzfolge. Das Prinzip des MPC ist in Abbildung 6.4-38 dargestellt.

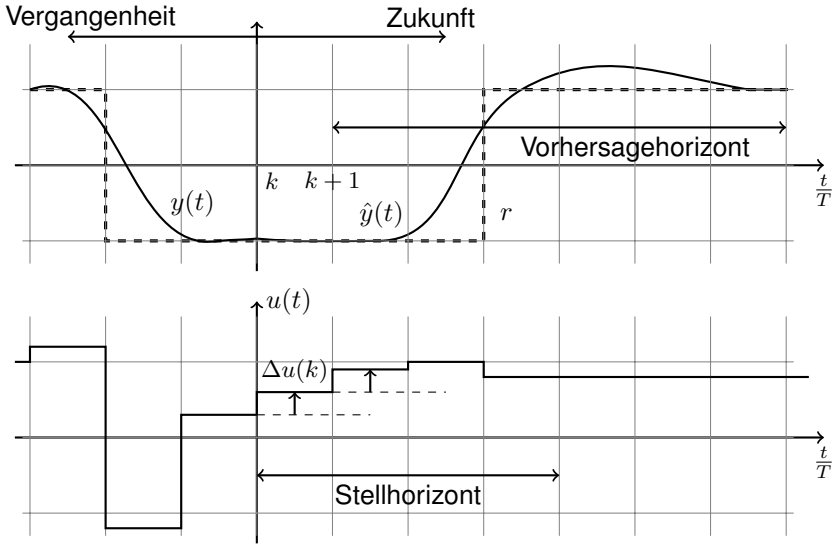


Abbildung 6.4-38: Prinzip des MPC

In jedem Abtastschritt wird die Kostenfunktion (6.4-134) minimiert. Das Optimierungsproblem

$$\min_{\substack{\mathbf{u}(k+i),\\ i=0,\dots,H_u-1}} \quad J(k) \tag{6.4-136}$$

$$\begin{aligned}
\text{subject to} \quad \hat{\mathbf{x}}(k+i+1) &= \mathbf{A}\hat{\mathbf{x}}(k+i) + \mathbf{B}\mathbf{u}(k+i) + \mathbf{B}_d\hat{\mathbf{d}}(k+i),\ i=1,\dots,H_p-1,\\
\hat{\mathbf{y}}(k+i) &= \mathbf{C}\hat{\mathbf{x}}(k+i),\ i=1,\dots,H_p-1,\\
\mathbf{u}_{min} &\le \mathbf{u}(k+i) \le \mathbf{u}_{max},\ i=0,\dots,H_u-1,\\
\mathbf{x}_{min} &\le \hat{\mathbf{x}}(k+i) \le \mathbf{x}_{max},\ i=1,\dots,H_p.
\end{aligned}$$

wird mit verschiedenen Gleichheits- und Ungleichheitsnebenbedingungen gelöst. Diese enthalten zum einen das lineare Modell, das die Dynamiken der Zustände und Ausgänge vorgibt. Die Störgröße $\mathbf{d}$ wird gleich der Störgrößenvorhersage $\hat{\mathbf{d}}$ gesetzt. Zudem sind Intervalle für jeden Zustand und Eingang angegeben, um den Arbeitsbereich der Anlage festzulegen, der nicht verlassen werden soll. Aufgrund der Verwendung eines linearen Modells und einer quadratischen Kostenfunktion ergibt sich ein quadratisches Optimierungsproblem, das somit konvex ist und sehr effizient von Standard Optimiereren gelöst werden kann, [34].

6.4.2 Prädiktive Regelung mit sukzessiver Linearisierung des MTI Systems

Um die Vorteile bei der Verwendung von linearen und nichtlinearen Modellen zu kombinieren, d.h. die guten Modellierungseigenschaften und die Konvexität des Optimierungsproblems, wurden in der Literatur veschiedene Konzepte betrachtet, das lineare System an die aktuellen Betriebsbedingungen der Anlage anzupassen, wie durch die Linearisierung während des Betriebes oder ein LPV (linear parameter variying) Modell, [24], [35]. Damit sollen die guten Modellierungseigenschaften des nichtlinearen Modells zum Teil erhalten werden, da die lineare Approximation an die aktuellen Bedingungen in der Anlage angepasst wird. Das nichtlineare Verhalten kann so zwar trotzdem nicht genau wiedergegeben werden, es wird jedoch besser angenähert als mit nur einem linearen Modell. Zudem ermöglicht es einen größeren Arbeitsbereich des Reglers, da dieser nicht auf die Umgebung eines festgelegten Arbeitspunktes eingeschränkt ist. Während eines Zeitschritts wird ein lineares Modell verwendet. Somit ist die Konvexität des Optimierungsproblems garantiert. In dieser Arbeit wird der Ansatz verfolgt das MTI System im Regler in jedem Abtastschritt um den aktuellen Arbeitspunkt sukzessiv zu linearisieren. Für allgemeine nichtlineare Systeme wurde dies bereits umgesetzt, [25]. Hier erfolgt die Spezialisierung auf MTI Modelle.

Es wird angenommen, dass das Systemverhalten durch ein MTI Modell

$$\dot{\mathbf{x}} = \langle\, \mathsf{F} \mid \mathsf{M}\,(\mathbf{x}, \mathbf{u}, \mathbf{d})\, \rangle\,, \tag{6.4-137}$$

$$\mathbf{y} = \langle\, \mathsf{G} \mid \mathsf{M}\,(\mathbf{x}, \mathbf{u})\, \rangle \tag{6.4-138}$$

beschrieben wird, bei dem die Eingänge in Stell- und Störsignale aufgeteilt sind. Das MTI Modell ist im Regler bekannt und soll in jedem Zeitschritt um den aktuellen Arbeitspunkt linearisiert werden. Daher ist im ersten Schritt ein Arbeitspunkt in der Nähe der aktuellen Betriebsbedingungen der Anlage zu bestimmen. Dies erfolgt wie beim Ansatz der dezentralen Zustandsrückführung durch Lösen von (6.3-110) mit den Nebenbedingungen (6.3-111) und (6.3-112), sodass der Arbeitspunkt auf der aktuellen Referenz und den Störgrößen liegt. Die Begrenzungen für die Eingangssignale werden durch die Formulierung des MPC Optimierungsproblems (6.4-136) gegeben. Daraus ergibt sich der Arbeitspunkt $\begin{pmatrix}\bar{\mathbf{x}} & \bar{\mathbf{u}} & \bar{\mathbf{d}} & \bar{\mathbf{y}}\end{pmatrix}$. Da das Modell des Systems durch ein MTI Modell in Tensorstruktur gegeben ist, kann die Linearisierung während des Betriebes sehr effizient ohne symbolische Berechnungen auf Basis des Differentiationsansatzes aus Kapitel 6.3.2 bestimmt werden. Eine analytisch korrekte Linearisierung kann ohne großen Berechnungsaufwand auf Basis von mathematischen Standardoperationen berechnet werden, sodass sich dieser Ansatz auch für Echtzeitimplementierungen eignet. Die linearen Systemmatrizen der Zustandsgleichung $\mathbf{A}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}})$, $\mathbf{B}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}})$ und $\mathbf{B}_d(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}})$ werden in Abhängigkeit vom Arbeitspunkt durch die Auswertung der kontrahierten Produkte (6.3-99) und (6.3-100) berechnet. Es wird angenommen, dass der Ausgang nicht gestört wird, sodass mit (6.3-101) die Matrix $\mathbf{C}(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ der linearen Ausgangsgleichung bestimmt wird. Da der prädiktive Regler ein zeitdiskretes Modell benötigt, erfolgt die Diskretisierung der soeben erstellten linearen Approximation mit Standardmethoden, [32]. Somit ergibt sich ein zeitdiskretes lineares Modell, welches das MTI Modell in der Nähe des Arbeitspunktes annähert zu

$$\begin{aligned}
\mathbf{x}(k+1) - \bar{\mathbf{x}} &= \mathbf{A}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}})\,(\mathbf{x}(k) - \bar{\mathbf{x}}) + \mathbf{B}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}})\,(\mathbf{u}(k) - \bar{\mathbf{u}}) + \mathbf{B}_d(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}})\left(\mathbf{d}(k) - \bar{\mathbf{d}}\right), \\
\mathbf{y}(k) - \bar{\mathbf{y}} &= \mathbf{C}(\bar{\mathbf{x}}, \bar{\mathbf{u}})\,(\mathbf{x}(k) - \bar{\mathbf{x}})\,,
\end{aligned}$$

Dieses lineare Modell wird für die Lösung des MPC Optimierungsproblems verwendet

$$\min_{\substack{\mathbf{u}(k+i),\\ i=0,\ldots,H_u-1}} \quad J(k) \tag{6.4-139}$$

$$\begin{aligned}
\text{subject to} \quad \hat{\mathbf{x}}(k+i+1) - \bar{\mathbf{x}} &= \mathbf{A}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}})\,(\hat{\mathbf{x}}(k+i) - \bar{\mathbf{x}}) + \mathbf{B}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}})\,(\mathbf{u}(k+i) - \bar{\mathbf{u}}) \cdots \\
&\quad + \mathbf{B}_d(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}})\left(\hat{\mathbf{d}}(k+i) - \bar{\mathbf{d}}\right),\ i = 1, \ldots, H_p - 1, \\
\hat{\mathbf{y}}(k+i) + \bar{\mathbf{y}} &= \mathbf{C}\,(\hat{\mathbf{x}}(k+i) - \bar{\mathbf{x}})\,,\ i = 1, \ldots, H_p - 1, \\
\mathbf{u}_{min} &\le \mathbf{u}(k+i) \le \mathbf{u}_{max},\ i = 0, \ldots, H_u - 1, \\
\mathbf{y}_{min} &\le \hat{\mathbf{y}}(k+i) \le \mathbf{y}_{max},\ i = 1, \ldots, H_p.
\end{aligned}$$

Mit einer quadratischen Kostenfunktion, wie (6.4-134), ist das Optimierungsproblem konvex. Die Lösung des Problems ergibt die optimale Stellgrößenfolge, wobei nur die Stellgrößen des ersten Zeitschritts $\hat{\mathbf{u}}(k)$ an die Anlage gegeben werden. Im nächsten Zeitschritt wird wieder ein Arbeitspunkt bestimmt, um den das MTI Modell linearisiert wird, um das Optimierungsproblem (6.4-139) mit dieser Linearisierung zu lösen.

Bis zu diesem Punkt wurde die Grundidee des prädiktiven Reglers mit sukzessiver Linearisierung für MTI Systeme beschrieben. Von einer numerischen Sichtweise aus können jedoch Probleme bei der Berechnung des Arbeitspunktes und des Optimierungsproblems entstehen, wenn die Signale sehr unterschiedliche Wertebereiche haben. Dies ist z.B. bei der Anwendung im Bereich von Heizungssystemen der Fall, wo Flüsse in $^{m^3}/_s$ mit Werten $\sim 10^{-3}$ und Temperaturen in K im Bereich von $300K$ auftreten können. Daher ist es sinnvoll die Signale zu skalieren und den MPC Algorithmus mit den skalierten Werten zu berechnen.

Für die Skalierung eines MTI Systems muss zunächst ein Arbeitsbereich für die Signale vorgegeben

werden

$$x_i \in [x_{i,l}, x_{i,u}]\,,\ i = 1, \ldots, n, \tag{6.4-140}$$
$$u_i \in [u_{i,l}, u_{i,u}]\,,\ i = 1, \ldots, m, \tag{6.4-141}$$
$$d_i \in [d_{i,l}, d_{i,u}]\,,\ i = 1, \ldots, m_d, \tag{6.4-142}$$
$$y_i \in [y_{i,l}, y_{i,u}]\,,\ i = 1, \ldots, p. \tag{6.4-143}$$

Die Signale sollen alle in den Bereich $[0, 1]$ skaliert werden, um numerische Probleme zu vermeiden. Für die skalierten Signale gilt somit

$$\tilde{x}_i \in [0, 1]\,,\ i = 1, \ldots, n, \tag{6.4-144}$$
$$\tilde{u}_i \in [0, 1]\,,\ i = 1, \ldots, m, \tag{6.4-145}$$
$$\tilde{d}_i \in [0, 1]\,,\ i = 1, \ldots, m_d, \tag{6.4-146}$$
$$\tilde{y}_i \in [0, 1]\,,\ i = 1, \ldots, p. \tag{6.4-147}$$

Diese Skalierung wird durch eine lineare Transformation der Zustands-, Eingangs-, und Ausgangsgrößen erreicht

$$x_i = a_i \tilde{x}_i + b_i,\ i = 1, \ldots, n, \tag{6.4-148}$$
$$u_i = a_{n+i} \tilde{u}_i + b_{n+i},\ i = 1, \ldots, m, , \tag{6.4-149}$$
$$d_i = a_{n+m+i} \tilde{d}_i + b_{n+m+i},\ i = 1, \ldots, m_d, , \tag{6.4-150}$$
$$y_i = c_i \tilde{y}_i + e_i,\ i = 1, \ldots, p. \tag{6.4-151}$$

Die Skalierungsfaktoren folgen aus den Betriebsbereichen (6.4-140) bis (6.4-143) und den skalierten Intervallen (6.4-144) bis (6.4-147) der Zustände, Stell-, Stör- und Ausgangsgrößen. Für die Zustände ergibt sich

$$a_i = \frac{x_{i,l} - x_{i,u}}{\tilde{x}_{i,l} - \tilde{x}_{i,u}}, \tag{6.4-152}$$
$$b_i = x_{i,l} - a_i \tilde{x}_{i,l}. \tag{6.4-153}$$

Die Skalierungsfaktoren der anderen Signale lassen sich analog berechnen. Die Skalierung von MTI Systemen in Matrixdarstellung wurde bereits in [23] und [40] eingeführt. Hier wird nun die lineare Transformation in der Tensordarstelllung betrachtet.

Der Arbeitsbereich (6.4-140) bis (6.4-143) des MTI Systems (6.4-137) und (6.4-138) wird durch eine lineare Transformation in den Bereich (6.4-144) bis (6.4-147) skaliert. Die Dynamiken in dem skalierten Arbeitsbereich wird durch die transformierte Zustandsgleichung

$$\dot{\tilde{\mathbf{x}}} = \left\langle \tilde{\mathsf{F}} \,\middle|\, \mathsf{M}\left(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}, \tilde{\mathbf{d}}\right) \right\rangle,$$

beschrieben. Der Parametertensor des skalierten Systems ergibt sich mit den Skalierungsfaktoren zu

$$\tilde{\mathsf{F}} = \mathsf{F} \times_1 \begin{pmatrix} b_{n+m+m_d} & 0 \\ a_{n+m+m_d} & 1 \end{pmatrix} \times_2 \begin{pmatrix} b_{n+m+m_d-1} & 0 \\ a_{n+m+m_d-1} & 1 \end{pmatrix} \times_3 \cdots \times_{n+m+m_d} \begin{pmatrix} b_1 & 0 \\ a_1 & 1 \end{pmatrix} \times_{n+m+m_d+1} \underset{i=1,\ldots,n}{\operatorname{diag}} (a_i)^{-1}.$$

Die Ausgangsgleichung des skalierten Systems

$$\begin{aligned} \tilde{\mathbf{y}} &= \left\langle \hat{\mathsf{G}} - \mathsf{E} \mid \mathsf{M}\left(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}\right) \right\rangle \\ &= \left\langle \tilde{\mathsf{G}} \mid \mathsf{M}\left(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}\right) \right\rangle, \end{aligned}$$

hat einen Parametertensor, der sich aus zwei Tensoren zusammensetzt

$$\hat{\mathsf{G}} = \mathsf{G} \times_1 \begin{pmatrix} b_{n+m+m_d} & 0 \\ a_{n+m+m_d} & 1 \end{pmatrix} \times_2 \begin{pmatrix} b_{n+m+m_d-1} & 0 \\ a_{n+m+m_d-1} & 1 \end{pmatrix} \times_3 \cdots \times_{n+m+m_d} \begin{pmatrix} b_1 & 0 \\ a_1 & 1 \end{pmatrix} \times_{n+m+m_d+1} \underset{i=1,\ldots,n}{\operatorname{diag}} (c_i)^{-1},$$

$$\mathsf{E} = \left[\mathbf{E}_{d_{m_d}}, \ldots, \mathbf{E}_{d_1}, \mathbf{E}_{u_m}, \ldots, \mathbf{E}_{u_1}, \mathbf{E}_{x_n}, \ldots, \mathbf{E}_{x_1}, \mathbf{E}_{\Phi} \right] \cdot \boldsymbol{\lambda}_E.$$

Die CP Faktoren von E sind

$$\mathbf{E}_{x_i} = \mathbf{E}_{u_j} = \mathbf{E}_{d_k} = \begin{pmatrix} 1 & \cdots & 1 \\ 0 & \cdots & 0 \end{pmatrix} \in \mathbb{R}^{2\times p},\ i = 1, \ldots, n,\ j = 1, \ldots, m,\ k = 1, \ldots, m_d, \tag{6.4-154}$$

$$\mathbf{E}_{\Phi} = \mathbf{I}_p, \tag{6.4-155}$$

$$\boldsymbol{\lambda}_E = \underset{i=1,\ldots,n}{\operatorname{diag}} (c_i)^{-1}\mathbf{e}, \tag{6.4-156}$$

wobei $\mathbf{I}_p$ eine $p \times p$ Einheitsmatrix ist.

Um die numerischen Probleme durch unterschiedliche Größenordnungen der Signale zu vermeiden, arbeitet der MPC mit der skalierten Version des MTI Modells. Das Blockdiagramm des geschlossenen Kreises des adaptiven MPC mit sukzessiver Linearisierung (AMPC-SL) und der Skalierung ist in der Abbildung 6.4-39 gezeigt.

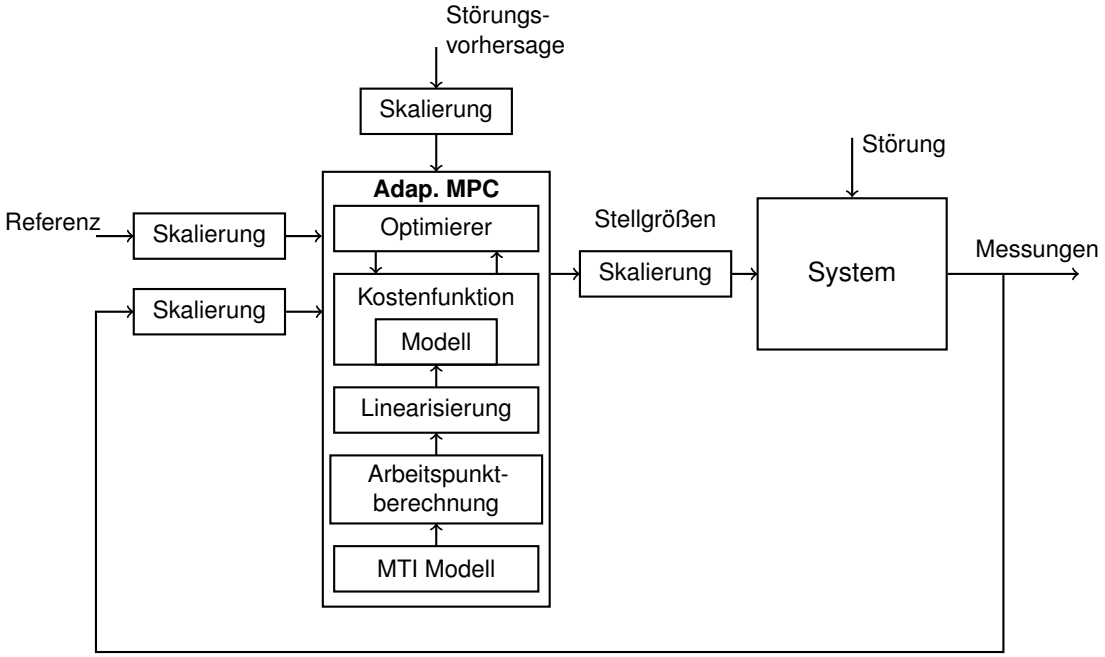


Abbildung 6.4-39: Regelkreis des AMPC mit sukzessiver Linearisierung mit skalierten Signalen

Somit wird auch der Arbeitspunkt, das lineare System sowie das Optimierungsproblem in den skalierten Signalen berechnet. Daher müssen alle Eingangssignale des MPC zunächst einmal skaliert werden durch

$$\tilde{x}_i = \frac{1}{a_i}\left(x_i - b_i\right),\ i = 1, \ldots, n,$$

$$\tilde{d}_i = \frac{1}{a_{n+m+i}}\left(d_i - b_{n+m+i}\right),\ i = 1, \ldots, m_d,$$

$$\tilde{y}_i = \frac{1}{c_i}\left(y_i - e_i\right),\ i = 1, \ldots, n.$$

Das berechnete optimale Stellsignal der Anlage muss zum Abschluss wieder in den Arbeitsbereich der

Strecke zurück skaliert werden

$$u_i = a_{n+i}\tilde{u}_i + b_{n+i},\ i = 1, \ldots, m,$$

bevor es an die Anlage gegeben wird. Durch die Verwendung von linearen Modellen für das MPC Optimierungsproblem wird vermieden, dass ein nichtlineares Optimierungsproblem während des Betriebes gelöst werden muss, was gerade bei der Echtzeitimplementierung ein wichtiger Vorteil ist. Jedoch muss für die Berechnung des Arbeitspunktes (6.3-110) trotzdem ein nichtlineares Optimierungsproblem gelöst werden. Die Lösung dieses Optimierungsproblems während des Betriebes kann zu Zeitproblemen führen. Daher wird hier ein Ansatz vorgeschlagen, die Arbeitspunkte bereits vor dem Betrieb zu berechnen. Aus der Definition des MPC ist der Arbeitsbereich des Systems (6.4-140) bis (6.4-142) bekannt. Es sollen nun Arbeitspunkte des MTI Modells für verschiedene Punkte im Arbeitsbereich berechnet werden. Dazu wird ein Gitter im Arbeitsbereich erzeugt, indem die Intervalle für jedes Signal in $N_{x,i}$, $N_{u,i}$, $N_{d,i}$ und $N_{y,i}$ Teile aufgeteilt werden. Damit ergeben sich die Gittervektoren für die einzelnen Signale, hier gezeigt am Beispiel von x_i, zu

$$\mathbf{X}_j = \begin{pmatrix} x_{j,min} & x_{j,min} + \gamma_{x,j} & \cdots & x_{j,min} + (N_{x,j} - 1)\,\gamma_{x,j} & x_{j,max} \end{pmatrix} \in \mathbb{R}^{N_{x,j}+1},\ j = 1, \ldots, n,$$

mit der Schrittweite $\gamma_{x,j} = \frac{x_{j,max} - x_{j,min}}{N_{x,j}}$. Die Gittervektoren der anderen Signale $\mathbf{U}_j$ und $\mathbf{D}_j$ werden analog erzeugt. Für jede Kombination der Elemente $\mathbf{X}_j(i_{x,j})$, $\mathbf{U}_j(i_{u,j})$ und $\mathbf{D}_j(i_{d,j})$ der Gittervektoren wird (6.3-110) mit den Nebenbedingungen (6.3-111) und (6.3-112) gelöst, um einen Arbeitspunkt für die Betriebsbedingungen, die durch die Gitterelemente gegeben werden, zu bestimmen. Die jeweiligen Werte für die Zustände werden als Referenz verwendet. Der resultierende Arbeitspunkt $\begin{pmatrix} \bar{\mathbf{x}} & \bar{\mathbf{u}} & \bar{\mathbf{d}} & \bar{\mathbf{y}} \end{pmatrix}$ wird in einem mehrdimensionalen Array, d.h. einem Tensor Θ_{OP}, mit den Elementen

$$\Theta_{OP}(i_{x,1}, \ldots, i_{x,n}, i_{u,1}, \ldots, i_{u,m}, i_{d,1}, \ldots, i_{d,m_d}, :) = \begin{pmatrix} \bar{\mathbf{x}} & \bar{\mathbf{u}} & \bar{\mathbf{d}} & \bar{\mathbf{y}} \end{pmatrix}, \tag{6.4-157}$$

der Dimension $\mathbb{R}^{N_{x,1}+1 \times \cdots \times N_{x,n}+1 \times N_{u,1}+1 \times \cdots \times N_{u,m}+1 \times N_{d,1}+1 \times \cdots \times N_{d,m_d}+1 \times n+m+m_d+p}$ gespeichert. Jeder Zustand sowie jede Stell- und Störgröße gehört zu einer Dimension des Tensors. In der letzten Dimension wird das Ergebnis der dazugehörigen Berechnungen des Arbeitspunktes gespeichert. Um den Speicheraufwand zu reduzieren, kann der Tensor Θ_{OP} durch Standard Dekompositionsverfahren, wie in Abschnitt 6.1.2 beschrieben, zerlegt werden. Die TT oder die HT Zerlegung sind hier sehr gut geeignet, da eine gewünschte Genauigkeit vorgegeben werden kann. Somit erhält man eine speichereffiziente Darstellung des Feldes (6.4-157) von Arbeitspunkten.

Dieser Tensor wird im Regler hinterlegt. Während des Betriebes muss dann keine Optimierung zur Festlegung des Arbeitspunktes berechnet werden. Der Arbeitspunkt wird durch Auswahl aus dem Feld an Arbeitspunkten bestimmt. Dazu wird der Gitterpunkt bestimmt, der am nächsten an den jeweiligen aktuellen Werten aus der Anlage liegt. Das bedeutet z.B. für den aktuellen Wert der Störgröße d_j, dass der Punkt des Gittervektors $\mathbf{D}_j(i_{d,j})$ mit dem Index $i_{d,j}$ ausgewählt wird, so dass die absolute Differenz $|d_j - \mathbf{D}_j(i_{d,j})|$ zwischen Messwert und dem Punkt minimal ist, verglichen mit allen anderen Elementen $\mathbf{D}_j(k),\ k = 1, \ldots, N_{d_j},\ k \neq i_{d,j}$ des Gittervektors. Die Indizes der anderen Signale werden in der gleichen Art berechnet. Mit dem so bestimmten Indizes $i_{x,j}$, $i_{u,j}$ und $i_{d,j}$ werden die Werte des Arbeitspunktes aus dem Tensor Θ_{OP} ausgewählt durch

$$\begin{aligned}
\bar{\mathbf{x}} &= \Theta_{OP}(i_{x,1}, \ldots, i_{x,n}, i_{u,1}, \ldots, i_{u,m}, i_{d,1}, \ldots, i_{x,m_d}, 1 : n),\\
\bar{\mathbf{u}} &= \Theta_{OP}(i_{x,1}, \ldots, i_{x,n}, i_{u,1}, \ldots, i_{u,m}, i_{d,1}, \ldots, i_{x,m_d}, n+1 : n+m),\\
\bar{\mathbf{d}} &= \Theta_{OP}(i_{x,1}, \ldots, i_{x,n}, i_{u,1}, \ldots, i_{u,m}, i_{d,1}, \ldots, i_{x,m_d}, n+m+1 : n+m+m_d),\\
\bar{\mathbf{y}} &= \Theta_{OP}(i_{x,1}, \ldots, i_{x,n}, i_{u,1}, \ldots, i_{u,m}, i_{d,1}, \ldots, i_{x,m_d}, n+m+m_d+1 : n+m+m_d+p).
\end{aligned}$$

Somit wird der Arbeitspunkt anhand der aktuellen Betriebsbedingungen von der Anlage aus dem Tensor Θ_{OP} ausgewählt. Daher ist es während des Betriebes nicht notwendig ein nichtlineares Optimierungsproblem zur Bestimmung des Arbeitspunktes zu lösen. Aufgrund der Diskretisierung führt dies zu einem geringen Verlust an der Genauigkeit, der jedoch akzeptierbar ist, wenn die Anzahl an Gitterpunkten ausreichend groß für die entsprechende Anwendung ist. Diese Arbeitspunktbestimmung von geringer Komplexität ist gerade hinsichtlich der Echtzeitimplementierung sinnvoll. Der Algorithmus für den AMPC-SL mit sukzessiver Linearisierung lässt sich somit in dem Flussdiagramm in Abbildung 6.4-40 zusammenfassen.

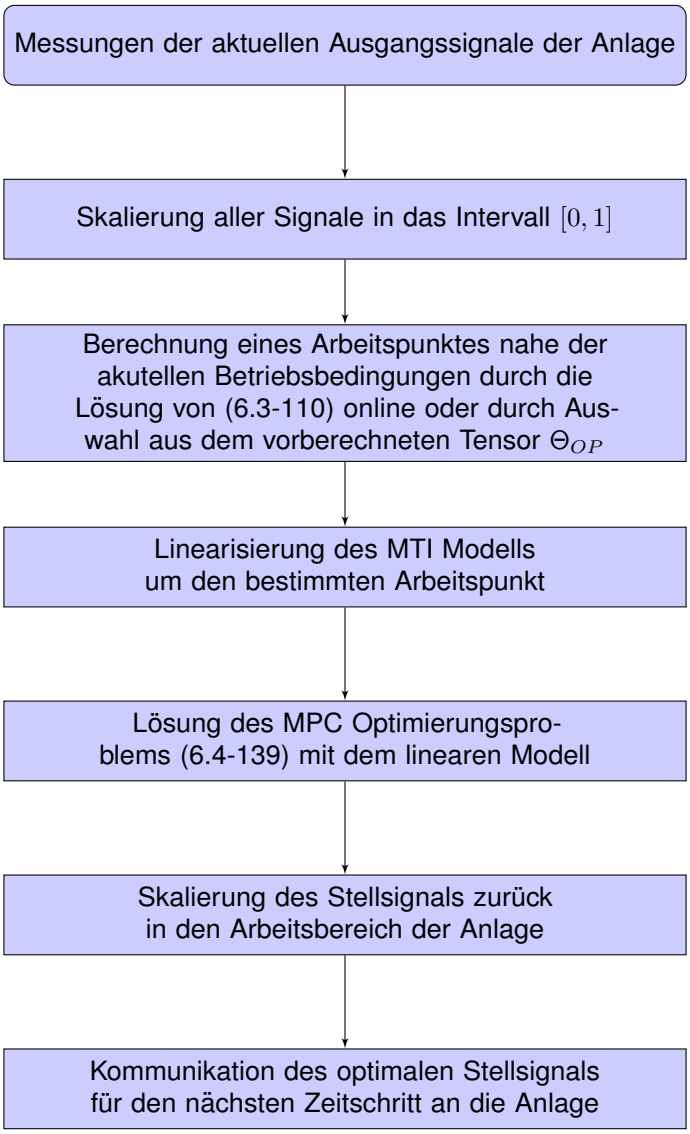


Abbildung 6.4-40: Flussdiagramm des AMPC-SL Algorithmus mit sukzessiver Linearisierung für jeden Abtastzeitschritt

6.4.3 Anwendungsbeispiel

Der im Kapitel 6.4.2 beschriebene adaptive MPC Ansatz mit sukzessiver Linearisierung soll auf ein Beispiel einer realen Heizungsanlage angewendet werden. Es soll ein Heizkreis eines Bürogebäudes geregelt werden. Das Schema des Heizkreises ist wie in Abbildung 6.3-30 dargestellt. Der Heizkreis wird mit einer gesamt Vorlauftemperatur $T_{Vl,ges}$ gespeist. Für die Anpassung der Vorlauftemperatur des Heizkreises $T_{Vl,HK}$ ist ein Mischventil vorhanden, mit dem die Temperatur des Vorlaufs abgesenkt werden kann. Eine Pumpe sorgt für den Volumenstrom im Heizkreis. Diese lässt sich bei dem betrachteten Gebäude nur manuell steuern, sodass der Regler keinen Zugriff darauf erhält. Die einzige Stellgröße des Reglers ergibt sich somit über das 3-Wege Ventil. In der bisherigen Regelung wird die Sollvorlauftemperatur des Heizkreises über eine Heizkurve bestimmt. Diese gibt die Sollvorlauftemperatur in Abhängigkeit von der Außentemperatur an, sodass die Vorlauftemperatur hoch für niedrige Außentemperaturen und niedrig für hohe Außentemperaturen ist. Für den Nachtbetrieb der Anlage wird die Heizkurve um $10\,\mathrm{K}$ abgesenkt. Ein PI-Regler erhält die Sollvorlauftemperatur als Referenz und stellt das 3-Wegeventil entsprechend ein. Der konventionelle Regelkreis ist in Abbildung 6.4-41 dargestellt.

Die Heizkurve ist meist so ausgelegt, dass der dahinter liegende Verbraucher auf jeden Fall mit ausrei-

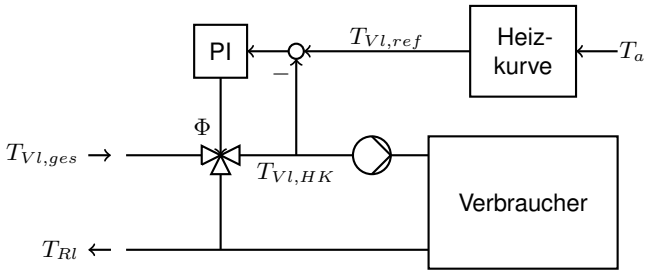


Abbildung 6.4-41: Aufbau des Verbraucherkreises mit Standardregelung

chend Leistung versorgt wird. Somit kann es sein, dass der Heizkreis oftmals mit zu hohen Vorlauftemperaturen betrieben wird, die eigentlich gar nicht notwendig wären. Gerade die Nachtabsenkung bietet hier ein Einsparpotenzial. Die Idee ist nun die Heizkurve durch einen AMPC-SL Regler zu ersetzen, um die Sollvorlauftemperatur zu bestimmen und diese besser an die wirklichen aktuellen Betriebsbedingungen anzupassen. Dadurch soll der Betrieb mit zu hohen Vorlauftemperaturen vermieden werden. Der Regler soll die Raumtemperatur in einem Komfortbereich halten, indem einer Referenz für die Raumtemperatur gefolgt wird. Zudem benötigt der Regler Messdaten aus der Anlage über die aktuellen Raum-, Rücklauf und Außentemperaturen sowie den Volumenstrom. Um prädiktiv handeln zu können, werden auch Wettervorhersagen über die Außentemperatur und die solare Einstrahlung mit einbezogen. Mithilfe eines Modells wird die Sollvorlauftemperatur bestimmt, die daraufhin über einen PI Regler eingestellt wird. Der Regelkreis mit dem prädiktiven Regler ist in Abbildung 6.4-42 dargestellt.

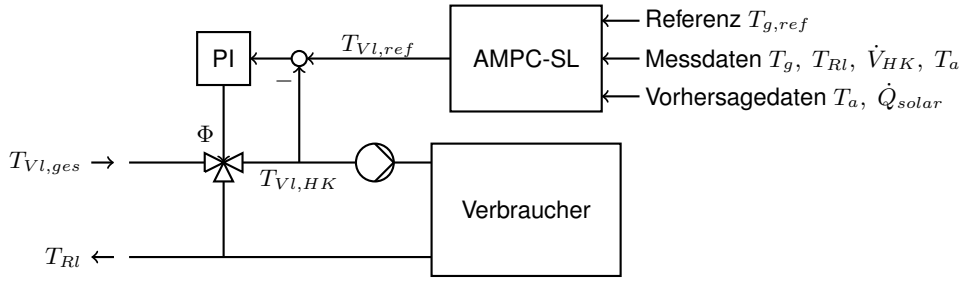


Abbildung 6.4-42: Aufbau des Verbraucherkreises mit AMPC-SL

Zur Berechnung der optimalen Sollvorlauftemperaturen ist ein Modell des Verbrauchers notwendig, um zu bestimmen, welche Gebäudetemperaturen sich für gewisse Vorlauftemperaturen, Volumenströme, Außentemperaturen sowie solare Einstrahlungen zukünftig einstellen werden. Die Ein- und Ausgänge des Modells sind in Abbildung 6.4-43 zusammengefasst.

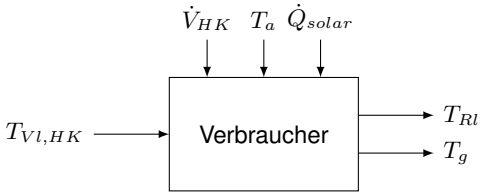


Abbildung 6.4-43: Ein- und Ausgänge des Verbrauchermodells für den AMPC-SL

Wie im Kapitel 6.3.4 wird das dynamische Verhalten des Verbrauchers über Wärmeleistungsbilanzen abgebildet. Die Modellbildung erfolgt analog zu den Verbrauchern in Kapitel 6.3.4 mit dem Unterschied, dass die Pumpe hier nicht angesteuert werden kann und die solare Einstrahlung mit einbezogen wird. Die Zustände sind weiterhin die Rücklauf- und die Raumtemperatur

$$\mathbf{x} = \begin{pmatrix} T_{Rl} & T_g \end{pmatrix}^T .$$

Es wird angenommen, dass der Verbraucher als Eingangsgröße mit einer bestimmten Vorlauftemperatur, der Stellgröße, versorgt wird. Der Volumenstrom, die Außentemperatur und die Solarstrahlung sind Störgrößen, die sich nicht beeinflussen lassen, sodass sich die Eingänge des Modells aufteilen zu

$$u = T_{Vl,HK},$$
$$\mathbf{d} = \begin{pmatrix} \dot{V}_{HK} & T_a & \dot{Q}_{solar} \end{pmatrix}^T.$$

Durch das Aufstellen der Wärmeleistungsbilanzen ergeben sich die Modellgleichungen zu

$$\dot{T}_{Rl} = \frac{1}{V_{HK}} \dot{V}_{HK}(T_{Vl,HK} - T_{Rl}) - \frac{k_{rg}}{c\rho V_{HK}}(T_{Rl} - T_g),$$
$$\dot{T}_g = \frac{k_{rg}}{C_g}(T_{Rl} - T_g) - \frac{k_{ga}}{C_g}(T_g - T_a) + \frac{k_s}{C_g}\dot{Q}_{solar}.$$

Aufgrund der Multiplikationen der Störgröße $\dot{V}_{HK,i}$ mit der Stellgröße $T_{Vl,HK}$ sowie der Zustandsgröße T_{Rl} ist das Modell in die Klasse der MTI Modelle einzuordnen, sodass es sich im Tensorformat

$$\dot{\mathbf{x}} = \langle\, \mathsf{F} \mid \mathsf{M}(\mathbf{x}, u, \mathbf{d}) \,\rangle,$$
$$\mathbf{y} = \langle\, \mathsf{G} \mid \mathsf{M}(\mathbf{x}, u, \mathbf{d}) \,\rangle,$$

mithilfe von CP Tensoren darstellen lässt. Die Parameter des Modells wurden mithilfe von Messdaten identifiziert. Ein Vergleich von Messdaten und Simulationsdaten mit einem Validierungsdatensatz sind in der Abbildung 6.4-44 gezeigt.

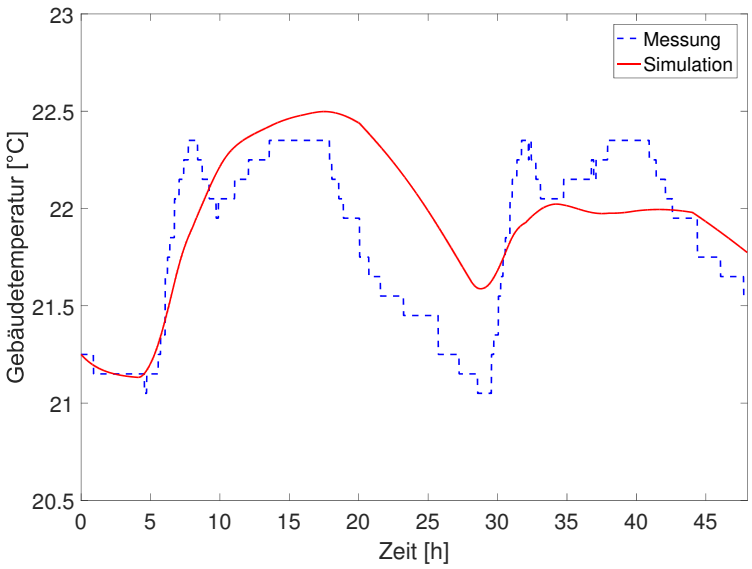


Abbildung 6.4-44: Validierung des Verbrauchermodells

Um die Signale skalieren zu können, wurde ein Arbeitsbereich für das Modell festgelegt

$$15\,^\circ\mathrm{C} \leq T_{Rl} \leq 80\,^\circ\mathrm{C}, \qquad 0\,\tfrac{\mathrm{m}^3}{\mathrm{s}} \leq \dot{V}_{HK} \leq 1.4 \cdot 10^{-3}\,\tfrac{\mathrm{m}^3}{\mathrm{s}},$$
$$18\,^\circ\mathrm{C} \leq T_g \leq 23.5\,^\circ\mathrm{C}, \qquad -5\,^\circ\mathrm{C} \leq T_a \leq 10\,^\circ\mathrm{C},$$
$$30\,^\circ\mathrm{C} \leq T_{Vl,HK} \leq 80\,^\circ\mathrm{C}, \qquad 0\,\tfrac{\mathrm{W}}{\mathrm{m}^2} \leq \dot{V}_{HK} \leq 500\,\tfrac{\mathrm{W}}{\mathrm{m}^2}.$$

Das MTI Modell wird so skaliert, dass der Arbeitsbereich aller Signale in den Intervallen $[0, 1]$ liegt. Dieses

skalierte Modell

$$\dot{\tilde{\mathbf{x}}} = \left\langle \tilde{\mathsf{F}} \,\middle|\, \mathsf{M}(\tilde{\mathbf{x}}, \tilde{u}, \tilde{\mathbf{d}}) \right\rangle,$$
$$\tilde{\mathbf{y}} = \left\langle \tilde{\mathsf{G}} \,\middle|\, \mathsf{M}(\tilde{\mathbf{x}}, \tilde{u}, \tilde{\mathbf{d}}) \right\rangle,$$

wird im Regler verwendet und in jedem Abtastschritt um einen Arbeitspunkt linearisiert. Vor dem Betrieb des Reglers wird ein Feld Θ_{OP} von Arbeitspunkten für das skalierte Modell berechnet. Zur speichereffizienten Darstellung wird die TT Zerlegung des Tensors Θ_{OP} verwendet.

Das MPC Optimierungsproblem verwendet somit eine die lineare Approximation des skalierten Modells zur Vorhersage des zukünftigen Systemverhaltens. Die Kostenfunktion ist gegeben durch

$$J = \sum_{i=1}^{H_p} \left\| \hat{\tilde{\mathbf{x}}}(k+i) - \tilde{\mathbf{x}}_{ref}(k+i) \right\|_{\mathbf{Q}}^2 + \sum_{i=0}^{H_u-1} \|\tilde{\mathbf{u}}(k+i) - \tilde{\mathbf{u}}(k+i-1)\|_{\mathbf{R}}^2 + \cdots$$
$$\sum_{i=0}^{H_u-1} \|\tilde{\mathbf{u}}(k+i) - \tilde{\mathbf{u}}_{target}(k+i)\|_{\mathbf{S}}^2 + \rho_\epsilon \epsilon_k,$$

wobei $\tilde{\mathbf{u}}_{target}(k+i)$ ein Zielwert für die Eingangsgröße und ϵ_k eine Schlupfvariable zur Aufweichung der harten Begrenzung der Signale mit der Gewichtung ρ_ϵ ist. Da immer eine möglichst niedrige Sollvorlauftemperatur verwendet werden soll, wird die minimale Vorlauftemperatur hier als Zielwert $\tilde{\mathbf{u}}_{target}(k+i)$ für die Eingangsgröße $T_{Vl,ref}$ gewählt. In der hier betrachteten Anwendung ist es nur wichtig, dass der zweite Zustand, d.h. die Gebäudetemperatur einer Referenz folgt und nicht der erste, sodass die Wichtungsmatrix für die Referenzfolge die folgende Struktur hat

$$\mathbf{Q} = \begin{pmatrix} 0 & 0 \\ 0 & q \end{pmatrix}.$$

Somit ergibt sich die Kostenfunktion hier zu

$$J = \sum_{i=1}^{H_p} q \left(\tilde{T}_g(k+i) - \tilde{T}_{g,ref}(k+i) \right)^2 + \sum_{i=0}^{H_u-1} r \left(\tilde{T}_{Vl,ref}(k+i) - \tilde{T}_{Vl,ref}(k+i-1) \right)^2 + \cdots$$
$$\sum_{i=0}^{H_u-1} s \left(\tilde{T}_{Vl,ref}(k+i) - \tilde{T}_{Vl,min}(k+i) \right)^2 + \rho_\epsilon \epsilon_k. \tag{6.4-158}$$

Der erste Term bewertet die Referenzfolge der Gebäudetemperatur mit dem Wichtungsfaktor q. Der zweite Term bestraft die Änderungsrate der Vorlaufsolltemperatur mit dem Faktor r und sorgt damit dafür, dass sich das Stellsignal nicht beliebig schnell ändert. Der dritte Term belegt die Abweichung der Vorlaufsolltemperatur von dem Minimalwert mit Kosten. In dem betrachteten Bürogebäude konnte eine Nutzungszeit von 6 bis 20 Uhr identifiziert werden. Um die Komfortanforderungen einzuhalten, soll die Gebäudetemperatur während des Tages möglichst nah an ihrer Referenz liegen. Auch die obere und untere Grenze für die Gebäudetemperatur sind hier eng um die Referenz gelegt. Während der Nacht ist eine Referenzfolge nicht wichtig. Die Grenzen werden weiter gesetzt. Der Regler muss dafür sorgen, dass das Gebäude nicht zu stark auskühlt und rechtzeitig wieder aufgeheizt wird. Abbildung 6.4-45 zeigt, wie die Referenz und die Begrenzungen der Raumtemperatur über den Tag variieren.

Da es in der betrachteten Implementierung nur möglich ist, dem Regler eine Prädiktion für die Referenz und nicht für die Begrenzungen zu übergeben, ist die untere Begrenzung gegenüber der Referenz um eine Stunde nach vorn verschoben. Da die Anforderungen im Tag- und Nachtbetrieb unterschiedlich sind, werden auch verschiedene Wichtungsfaktoren verwendet. Tagsüber soll die Raumtemperatur nahe der Referenz sein, sodass hier die Wichtung auf die Referenzfolge gelegt wird. Nachts ist die Referenzfolge nicht wichtig, sodass q hier reduziert wird. Der Fokus ist mehr auf dem absoluten Wert der Vorlauftemperatur, sodass diese möglichst minimal wird. Die Raumtemperatur darf nur nicht unter einen minimalen Wert fallen.

Der Vorhersagehorizont wurde auf 3 Stunden und der Stellhorizont auf 2 Stunden gesetzt. Die Abtastzeit

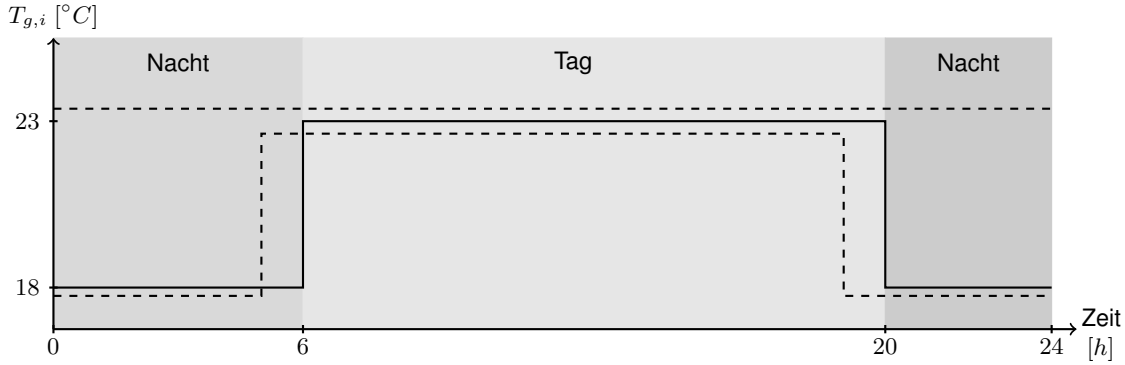


Abbildung 6.4-45: Tagesverlauf der Referenz (—) sowie der oberen und unteren Grenze (- - -) für die Gebäudetemperatur

ist auf eine Minute festgelegt. Diese Wahl ergibt sich aus den Dynamiken, d.h. den Zeitkonstanten, der Anlage. Das Simulationsergebnis für einen Tag ist in der Abbildung 6.4-46 dargestellt.

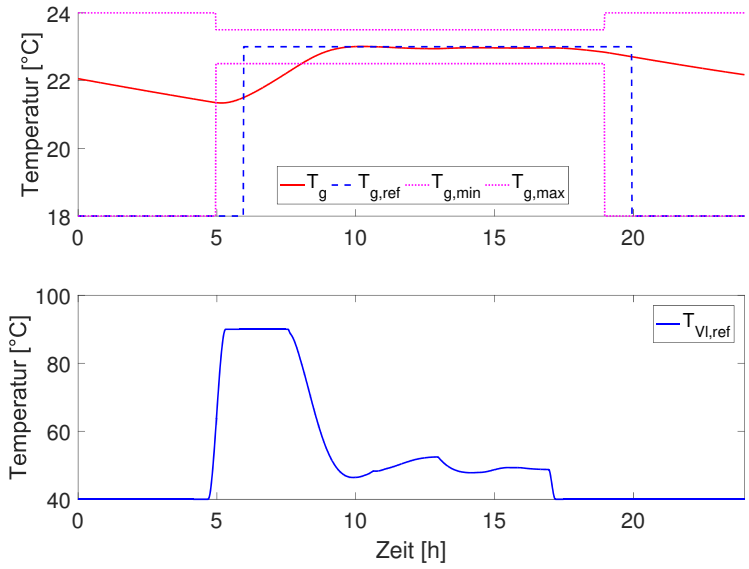


Abbildung 6.4-46: Simulationsergebnis im geschlossenen Kreis

Es ist deutlich zu erkennen, wie der Regler die Vorlauftemperatur während der Nacht auf den minimalen Wert absenkt. Zu Beginn des Tages erfolgt dann eine Aufheizung des Gebäudes auf die Raumtemperatur. Danach wird die gewünschte Referenztemperatur gehalten. In Standardregelungen ist meistens ein festgelegter Aufheizzeitpunkt hinterlegt. Dieser ist so gewählt, dass es zu Beginn der Nutzzeit warm im Gebäude ist auch wenn es draußen sehr kalt ist. Ist es draußen wärmer, wird der gewünschte Komfortbereich früher erreicht, was dazu führt, dass das Gebäude bereits vor dem Nutzzeitbeginn warm ist. Der Vorteil hier ist, dass der Aufheizzeitpunkt nicht mehr fest angegeben wird, sondern über das Modell abhängig von den aktuellen Umgebungsbedingungen, wie der Außentemperatur bestimmt wird. Somit wird das zu frühe Aufheizen des Gebäudes verhindert.

6.5 Dezentraler Entwurf einer prädiktiven Regelung

Die steigende Komplexität der heutigen Anlagen im Gebäudebereich, erschwert auch das Regelungsproblem für solche Anlagen. Die Verwendung eines zentralen Reglers für die gesamte Anlage kann z.B. bei

prädiktiven Reglern zu einem sehr großen Berechnungsaufwand führen. Prädiktive Regler eigenen sich aufgrund der großen Zeitkonstanten und Totzeiten im Gebäudebereich sehr gut für die Regelung von Heizungsanlagen, [16]. Zudem können Störgrößenvorhersagen, wie die Wettervorhersage mit einbezogen werden. Allerdings ist in jedem Zeitschritt die Lösung eines Optimierungsproblems notwendig. Wird ein zentraler Regler für eine komplexe Anlage mit vielen Stellsignalen verwendet, führt dies zu einem komplexen Optimierungsproblem, da der Suchraum für die Optimierungsvariablen sehr groß wird. Dies ist ein begrenzender Faktor für die Anwendung, da die Lösung des Optimierungsproblem innerhalb eines Abtastzeitschritts sicher vorliegen muss. Ist das Problem zu aufwendig, d.h. dauert die Lösung zu lange ist der Ansatz in der Form nicht anwendbar.
Daher wurde ein Ansatz untersucht, die prädiktive Regelungsaufgabe auf einzelne Reglerknoten zu verteilen. Die einzelnen Reglerknoten müssen deutlich kleinere Optimierungsprobleme lösen, sodass sich eine geringere Komplexität der einzelnen Optimierungsprobleme und somit auch für die gesamte Regelungsaufgabe ergibt.

6.5.1 Dezentrale Reglerstruktur

Das Ziel bei der dezentralen prädiktiven Regelung ist es, die Regelungsaufgabe auf mehrere Reglerknoten zu verteilen, [37]. Bei dem zentralen MPCs, die bisher betrachtet wurden, ist das Problem, dass die Modelle für große Anlagen sehr komplex werden. Die Komplexität des Optimierungsproblems steigt dadurch auch an. Zudem vergrößert sich der Suchraum des Optimierungsproblems mit der Anzahl an Eingangssignalen der Regelstrecke. Da das Optimierungsproblem in der Echtzeitanwendung in einer begrenzten Zeit, der Abtastzeit, berechnet werden muss, ist für die Lösung komplexer Optimierungsprobleme viel Rechenleistung nötig. Die Bereitstellung von Hardware mit großer Rechenleistung ist in der Anwendung zum Teil nicht möglich oder mit großen Kosten verbunden, sodass ein zentraler Entwurf hier zu Problemen führen kann. Zudem ist für einen zentralen Regler ein großer Kommunikationsaufwand notwendig. Dies kann zu einem sehr hohen Datenfluss führen. Kommt es bei einem zentralen Design z.B. zu Änderungen an der Anlage, muss der gesamte Regler geändert werden. Das Modell der Gesamtanlage ist neu zu erstellen und der gesamte Regler muss daraufhin neu getunt werden. Bei einem dezentralen Aufbau muss nur der Regler betrachtet werden, der dem geänderten Subsystem zugeordnet ist. Hier ergibt sich somit ein deutlich geringerer Änderungsaufwand, [37].

Um eine solche Verteilung der prädiktiven Regelungsaufgabe durchzuführen, sind verschiedene Reglerstrukturen möglich, [44]. Bisher wurde ein zentrales Design betrachtet, bei dem der prädiktive Regler Zugriff auf alle Mess- und Stellsignale hat, wie in der Abbildung 6.5-47 für eine Strecke dargestellt, die sich aus zwei Subsystemen zusammensetzt.

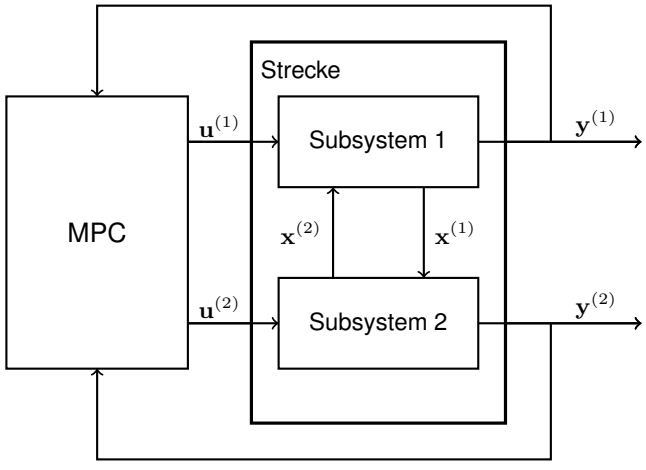


Abbildung 6.5-47: Zentraler MPC, [8]

Der höchste Grad der Verteilung wird bei der dezentralen Struktur erreicht. Hierbei hat jedes Subsystem seinen eigenen lokalen Regler. Die lokalen Regler haben nur Zugriff auf die Sensoren und Aktoren des

jeweiligen Subsystems und tauschen keine Informationen z.B. über zukünftige Stellsignale untereinander aus. Eine dezentrale Struktur für zwei Subsysteme ist in der Abbildung 6.5-48 gezeigt.

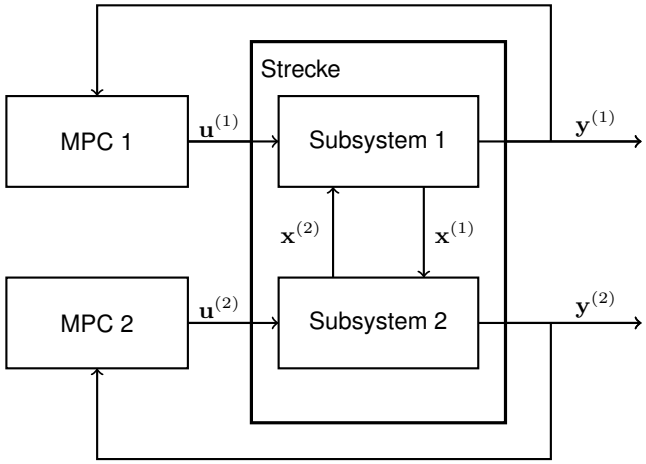


Abbildung 6.5-48: Dezentraler MPC, [8]

Da es zu gar keinem direkten Austausch von Informationen zwischen den Reglern kommt, sondern diese nur den Einfluss des anderen Reglers über die Anlage sehen können, kann eine solche Struktur gerade bei Systeme mit Querkopplungen zwischen den Subsystemen zu keinem gewünschten Regelungsergebnis führen, [44]. Daher soll hier ein Ansatz gewählt werden, der eine gewisse Kommunikation zwischen den Reglern erlaubt. Dies wird in der Literatur verteilter MPC genannt und mit DMPC (distributed MPC) abgekürzt. Die Regler tauschen Informationen über zukünftige Stellsignale oder Verläufe der Zustandsvariablen des eigenen Subsystems aus. Diese können dann in den Reglern des anderen Subsystems beachtet werden. Dies führt zwar zu einem erhöhten Kommunikationsaufwand, gibt jedoch auch einen deutlichen Vorteil bei der Regelungsgüte gerade bei verkoppelten Systemen. Die Wahl der kommunizierten Signale ist sehr flexibel und von der jeweiligen Anwendung abhängig. Abbildung 6.5-49 stellt eine verteilte Reglerstruktur dar.

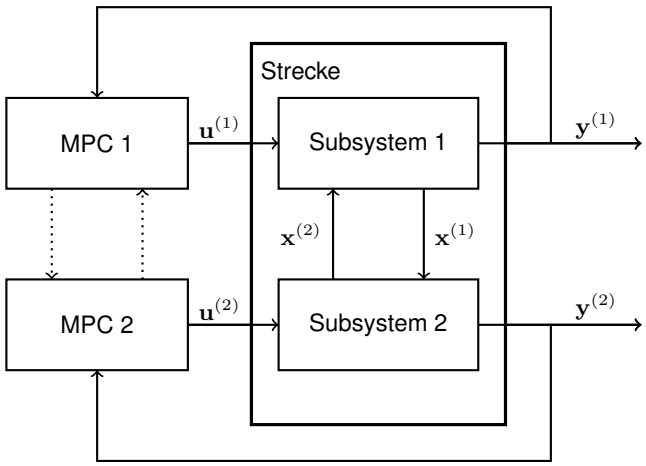


Abbildung 6.5-49: Verteilter MPC, [8]

Für den Entwurf einer verteilten MPC Struktur für MTI Subsysteme wird hier angenommen, dass sich das Gesamtsystem aus N_{sub} Subsystemen zusammensetzt. Jedes der Subsysteme lässt sich durch ein multilineares Modell

$$\dot{\mathbf{x}}^{(i)} = \left\langle \mathsf{F}^{(i)} \;\middle|\; \mathsf{M}\left(\mathbf{x}^{(i)}, \mathbf{u}^{(i)}, \mathbf{d}^{(i)}\right) \right\rangle,$$
$$\mathbf{y}^{(i)} = \left\langle \mathsf{G}^{(i)} \;\middle|\; \mathsf{M}\left(\mathbf{x}^{(i)}, \mathbf{u}^{(i)}, \mathbf{d}^{(i)}\right) \right\rangle$$

mit $i = 1, \ldots, N_{sub}$ beschreiben. Jedes Subsystem erhält seinen eigenen prädiktiven Reglerknoten, in dem der AMPC-SL Ansatz implementiert ist. Die Ein- und Ausgänge des jeweiligen Reglerknotens sind in der Abbildung 6.5-50 dargestellt.

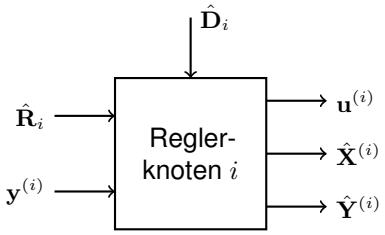


Abbildung 6.5-50: Reglerknoten des i-ten Subsystems

Jeder AMPC-SL arbeitet mit dem MTI Modell des eigenen Subsystems. Die Eingangsgrößen des Reglerknotens sind ähnlich wie bei den vorherigen zentralen MPC Entwürfen. Es werden jedoch nur Messgrößen $\mathbf{y}^{(i)}$ aus dem eigenen Subsystem verwendet, sowie eine Vorhersage der Sollwerte

$$\hat{\mathbf{R}}_i = \begin{pmatrix} \mathbf{r}^{(i)}(k+1) & \mathbf{r}^{(i)}(k+2) & \cdots & \mathbf{r}^{(i)}(k+H_p) \end{pmatrix}.$$

Das Störsignal setzt sich hier nun jedoch nicht nur aus Größen außerhalb der Anlage, wie das Wetter zusammen. Auch Signale aus den anderen Subsystemen, die einen Einfluss auf des Verhalten des i-ten Subsystems haben gehen hier als Störung mit ein. Eine Schätzung der Störgrößentrajektorien

$$\hat{\mathbf{D}}_i = \begin{pmatrix} \hat{\mathbf{d}}^{(i)}(k+1) & \hat{\mathbf{d}}^{(i)}(k+1) & \cdots & \hat{\mathbf{d}}^{(i)}(k+H_p) \end{pmatrix},$$

ist möglich, da die prädiktiven Regler der übrigen Reglerknoten die Stellsignale sowie Zustandssignale über ihren Stell- bzw. Vorhersagehorizont bestimmen können. Daher gibt der allgemeine Reglerknoten neben dem üblichen Stellsignal für den nächsten Zeitschritt $\hat{\mathbf{u}}^{(i)}(k)$ auch die optimale Stellsignalfolge

$$\hat{\mathbf{U}}^{(i)} = \begin{pmatrix} \hat{\mathbf{u}}^{(i)}(k) & \hat{\mathbf{u}}^{(i)}(k+1) & \cdots & \hat{\mathbf{u}}^{(i)}(k+H_u-1) \end{pmatrix}, \tag{6.5-159}$$

für den Stellhorizont heraus. Mit der vorhergesagten Stellsignaltrajektorie und dem Modell des Subsystems lassen sich Vorhersagen der Zustands- und Ausgangssignale über eine Vorwärtssimulation über den Vorhersagehorizont bestimmen, sodass man die Vorhersagen

$$\hat{\mathbf{Y}}^{(i)} = \begin{pmatrix} \hat{\mathbf{y}}^{(i)}(k+1) & \hat{\mathbf{y}}^{(i)}(k+2) & \cdots & \hat{\mathbf{y}}^{(i)}(k+H_p) \end{pmatrix}$$

erhält, die am Ausgang des Reglers anliegen. Jeder Regler hat eine quadratische Kostenfunktion wie

$$J_i(k) = \sum_{j=1}^{H_p} \|\hat{\mathbf{y}}^{(i)}(k+j) - \mathbf{r}^{(i)}(k+j)\|^2_{\mathbf{Q}} + \sum_{j=0}^{H_u-1} \|\Delta\mathbf{u}^{(i)}(k+j)\|^2_{\mathbf{R}} \tag{6.5-160}$$

für die Referenzfolge. Die Kostenfunktion des Gesamtsystems setzt sich über eine Summe aus den Kostenfunktionen der einzelnen Subsysteme zusammen

$$J = \sum_{i=1}^{N_{sub}} J_i(k),$$

sodass in dem Regler jedes Subsystems jeweils ein Teil der Kostenfunktion minimiert wird. Das bedeutet, dass in dem MPC des Subsystems i das Optimierungsproblem

$$\min_{\substack{\mathbf{u}^{(i)}k+i, \\ i=0,\ldots,H_u-1}} J_i(k)$$

mit den Nebenbedingungen des Modells und den üblichen Begrenzungen der Stell- und Zustandsgrößen,

wie in (6.4-136) gelöst wird. Der MPC des Subsystems i berechnet somit nur die Stellsignale des i-ten Subsystems, was den Suchraum und somit auch die Komplexität des jeweiligen Optimierungsproblems deutlich reduziert. Zudem ist es je nach Anwendung möglich, dass einige Optimierungsprobleme parallel berechnet werden können. Die Kommunikation der Vorhersagen erfolgt nur zwischen Subsystemen, bei denen es nötigt ist. Es wird z.B. über ein Netzwerk kommuniziert. Zudem ist ein Koordinator vorhanden, der den einzelnen Reglerknoten die entsprechenden Signale aufbereitet und zur Verfügung stellt. Somit ergibt sich eine Gesamtstruktur des Regelkreises wie in Abbildung 6.5-51 dargestellt.

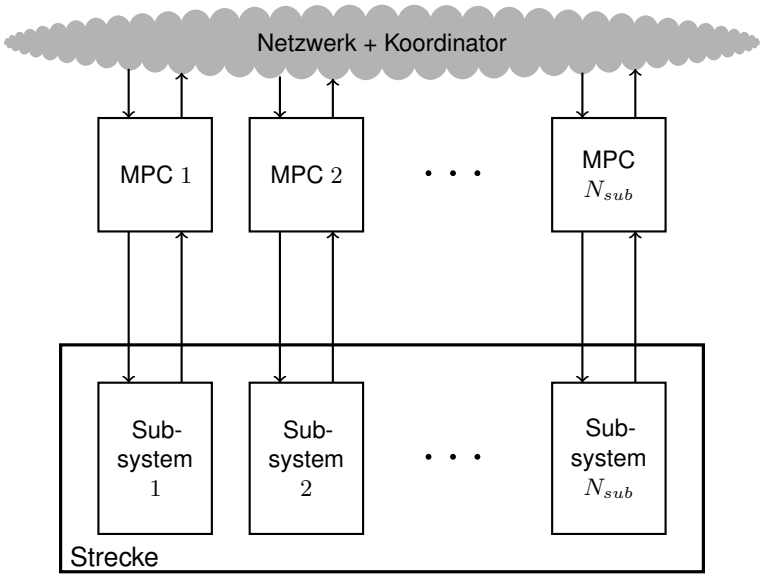


Abbildung 6.5-51: Gesamtstruktur des verteilten AMPC-SL Reglers

6.5.2 Anwendungsbeispiel

Betrachtet wird ein Beispiel eines typischen Aufbaus eines Heizungssystems, bei dem mehrere Heizkreise von einem Kessel versorgt werden. Ein Schema der Anlage ist in der Abbildung 6.5-52 gezeigt.

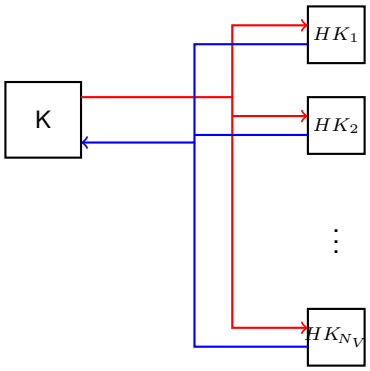


Abbildung 6.5-52: Schema der Beispielanlage

Die Modellierung des Gesamtsystems erfolgt analog zum Kapitel 6.3.4 über Wärmeleistungsbilanzen. Die Struktur der Komponenten ist hier jedoch leicht verändert. Die Modellierung des Kessels erfolgt wie in (6.3-126) durch

$$\dot{T}_{Vl} = \frac{1}{V_K}\dot{V}_{ges}\left(T_{Rl,ges} - T_{Vl}\right) + \alpha\frac{P_{max}}{c\rho V_K}. \tag{6.5-161}$$

Die Stellsignal des Kessels ist das Modulationssignal $u = \alpha$. Die Rücklauftemperatur $T_{Rl,ges}$ und der Volumenstrom $\dot{V}_{ges}$ sind Störungen

$$\mathbf{d}^{(K)} = \begin{pmatrix} T_{Rl,ges} & \dot{V}_{ges} \end{pmatrix}^T,$$

die von den Verbraucherkreisen kommen. Der Rücklauf ergibt sich aus der Mischung der Volumenströme und Temperaturen aus den Verbrauchern durch

$$T_{Rl,ges} = \frac{\sum_{i=1}^{N_V} \varphi_i \dot{V}_{max,i} T_{Rl,V_i}}{\dot{V}_{ges}}, \tag{6.5-162}$$

$$\dot{V}_{ges} = \sum_{i=1}^{N_V} (1 - \Phi_i) \varphi_i \dot{V}_{max,i}, \tag{6.5-163}$$

wobei N_V die Anzahl der angeschlossenen Heizkreise ist. Die Verbraucherkreise sind wie in der Abbildung 6.3-30 dargestellt aufgebaut. Die Modelle der Verbraucher haben wie in (6.3-123) und (6.3-119) 2 Zustände

$$\mathbf{x}^{(HK_i)} = \begin{pmatrix} T_{Rl,i} & T_{g,i} \end{pmatrix}^T,$$

mit der Rücklauf- und der Gebäudetemperaturen $T_{Rl,i}$ und $T_{g,i}$. Die Eingangssignale

$$\mathbf{u}^{(HK_i)} = \begin{pmatrix} \varphi_i & \Phi_i \end{pmatrix}^T,$$
$$\mathbf{d}^{(HK_i)} = \begin{pmatrix} T_{Vl,ges} & T_a \end{pmatrix}^T,$$

setzen sich aus den Stellsignale mit den Pumpen- und Ventilsignalen sowie den Störgrößen mit der gesamten Vorlauftemperatur und der Außentemperatur zusammen. Damit ergeben sich die Modellgleichungen des Verbrauchers zu

$$\begin{aligned} \dot{T}_{Rl,i} &= \varphi_i \frac{\dot{V}_{max,i}}{V_{HK,i}} \left(Tscii - T_{Rl,i} \right) - \frac{k_{rg,i}}{c\rho V_{HK,i}} \left(T_{Rl,i} - T_{g,i} \right) \\ &= \varphi_i \frac{\dot{V}_{max,i}}{V_{HK,i}} \left(\Phi_i T_{Rl,i} + (1 - \Phi_i) T_{Vl} i - T_{Rl,i} \right) - \frac{k_{rg,i}}{c\rho V_{HK,i}} \left(T_{Rl,i} - T_{g,i} \right), \end{aligned} \tag{6.5-164}$$

$$\dot{T}_{g,i} = \frac{k_{rg,i}}{C_{g,i}} (T_{Rl,i} - T_{g,i}) - \frac{k_{ga,i}}{C_{g,i}} (T_{g,i} - T_a) \tag{6.5-165}$$

Die Submodelle gehören alle in die Klasse der MTI Modelle. Das Gesamtmodell ergibt sich aus den vorangegangenen Submodellen des Kessels und der Heizkreise zu einem MTI Modell

$$\dot{\mathbf{x}} = \langle \, \mathsf{F} \mid \mathsf{M} \left(\mathbf{x}, \mathbf{u}, d \right) \rangle,$$
$$\mathbf{y} = \langle \, \mathsf{G} \mid \mathsf{M} \left(\mathbf{x}, \mathbf{u}, d \right) \rangle.$$

Da jeder Verbraucher 2 Zustände und Stellsignale und der Kessel jeweils nur eins hat, ergeben sich $2N_V + 1$ Zustände, $2N_V + 1$ Stellsignale und eine Störung für das Gesamtmodell.

Im Falle eines zentralen MPC zur Regelung wird dieses Gesamtmodell im Regler verwendet. Den zentralen Regelkreis zeigt die Abbildung 6.5-53.

Der zentrale MPC soll das Modulationssignal des Kessels sowie die Stellsignale der Pumpen und Ventile der Verbraucher so einstellen, dass die Raumtemperaturen der einzelnen Heizkreise die gewünschten

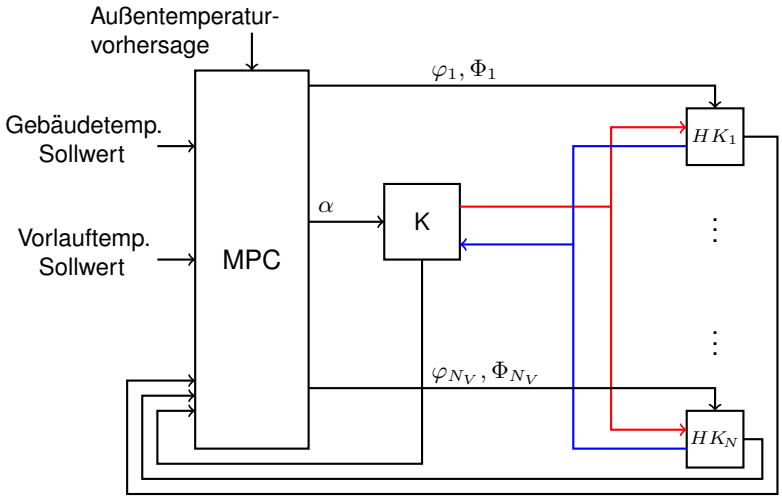


Abbildung 6.5-53: Zentraler MPC für das Heizungsbeispiel

Sollwerte erreichen. Dazu wird die Kostenfunktion

$$J = \sum_{i=1}^{N_V} \sum_{j=1}^{H_p} \|T_{g,HK_i}(k+j) - T_{g,HK_i,ref}(k+j)\|_{q_{HK,i}} + \sum_{j=0}^{H_u-1} \left\|\Delta \mathbf{u}^{(HK_i)}(k+j)\right\|_{\mathbf{R}_{HK,i}} \cdots$$
$$+ \sum_{j=1}^{H_p} \|T_{Vl}(k+j) - T_{Vl,ref}(k+j)\|_{q_K} + \sum_{j=0}^{H_u-1} \left\|\Delta \mathbf{u}^{(K)}(k+j)\right\|_{\mathbf{R}_K}$$

definiert. Dabei wird die Wettervorhersage in Form von der Außentemperaturvorhersage mit einbezogen. In jedem Abtastschritt wird die optimale Eingangsfolge $\hat{\mathbf{U}}^{(i)}$ für jedes Stellsignal für den Stellhorizont über eine Optimierung berechnet. Die Anzahl der Optimierungsvariablen ergibt sich somit aus dem Produkt $(2N_V + 1)H_u$ der Anzahl der Stellsignale und dem Stellhorizont. Mit der Größe des Suchraums steigt auch die Komplexität des Optimierungsproblems. Der Stellhorizont ergibt sich aus der Dynamik der Anlage und kann nicht beliebig reduziert werden. Daher soll hier betrachtet werden, wie sich die Größe des Suchraums der Optimierungsprobleme reduzieren lässt. Die Idee ist, das Optimierungsproblem in Subprobleme aufzuteilen, die einen kleineren Suchraum haben. Das globale Ziel der Optimierung soll trotzdem noch beachtet werden. Es wird somit für jede Komponente ein Optimierungsproblem bestimmt. Der Kessel und jeder Heizkreis erhält einen eigenen Reglerknoten. Somit ist die Anzahl der Stellsignale, die ein solcher Reglerknoten berechnen muss deutlich geringer als im zentralen Fall, sodass auch der Rechenaufwand für einen Knoten deutlich reduziert wird. Ein Reglerknoten eines Verbrauchers muss nur 2 Stellsignale berechnen, für den Kessel ist es nur 1 Stellsignal. Um trotzdem ein gutes Regelungsergebnis für die Gesamtanlage zu erzielen, ist eine Kommunikation zwischen den einzelnen Reglerknoten notwendig. Die Reglerknoten der Heizkreise kommunizieren die Vorhersage des zukünftigen Verbrauchs an den Kesselregler, der daraus die Referenz der Vorlauftemperatur für den Kessel bestimmt damit der Kessel Wasser der angeforderten Temperatur bereitstellen kann. Die einzelnen Reglerknoten werden im Folgenden vorgestellt.

Verbraucher
Jeder Verbraucher erhält ein AMPC-SL Regler, der berechnet, welche Vorlauftemperatur und welcher Volumenstrom benötigt wird, um die Gebäudetemperatur einer Referenz folgen zu lassen, sodass der Raum im Komfortbereich bleibt. Die Referenz wird z.B. über ein Nutzungsprogramm des jeweiligen Heizkreises vorgegeben. Die Referenzfolge wird durch die Kostenfunktion

$$J_{V_i} = \sum_{j=1}^{H_p} \|T_{g,HK_i}(k+j) - T_{g,HK_i,ref}(k+j)\|_{q_{HK,i}} + \sum_{j=0}^{H_u-1} \left\|\Delta \mathbf{u}^{(HK_i)}(k+j)\right\|_{\mathbf{R}_{HK,i}}$$

abgebildet. Der prädiktive Regler berechnet eine gewünschte Vorlauftemperatur und das Stellsignal

der Pumpe. Daher muss das 3-Wege Ventil zunächst nicht im Modell des Verbrauchers im Regler berücksichtigt werden. Es wird angenommen, dass die Ventilposition im Anschluss eingestellt werden kann, sodass die gewünschte Vorlauftemperatur als Mischtemperatur zur Verfügung steht. Somit hat das Modell im Regler die Struktur

$$\dot{T}_{Rl,i} = \varphi_i \frac{\dot{V}_{max,i}}{V_{HK,i}} \left(T_{Vl,HK,i} - T_{Rl,i}\right) - \frac{k_{rg,i}}{c\rho V_{HK,i}} \left(T_{Rl,i} - T_{g,i}\right), \tag{6.5-166}$$

$$\dot{T}_{g,i} = \frac{k_{rg,i}}{C_{g,i}}(T_{Rl,i} - T_{g,i}) - \frac{k_{ga,i}}{C_{g,i}}(T_{g,i} - T_a), \tag{6.5-167}$$

mit den Zuständen $\begin{pmatrix} T_{Rl,i} & T_{g,i} \end{pmatrix}^T$, den Eingängen $\begin{pmatrix} \varphi_i & T_{Vl,HK,i} \end{pmatrix}^T$ und der Störung T_a. Das 3-Wege Ventil stellt daraufhin die gewünscht Sollvorlauftemperatur des Heizkreises über einen PI Regler ein. Es wird angenommen, dass dieser Regler perfekt funktioniert und die Einstellung schneller als die Abtastzeit von $60\,\mathrm{s}$ abläuft, sodass sich das Stellsignal des Ventils statisch berechnet zu

$$\Phi_i = \frac{T_{Vl,HK,i} - T_{Vl,ges}}{T_{Rl,i} - T_{Vl,ges}} \in [0,1]\,.$$

Der Regelkreis ist in der Abbildung 6.5-54 dargestellt.

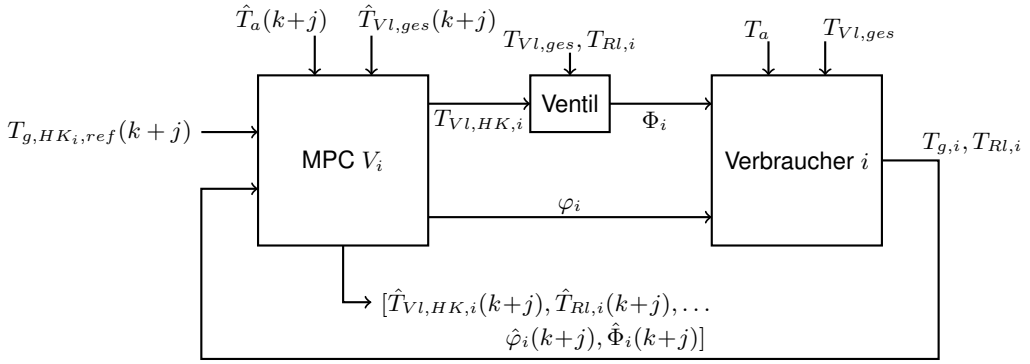


Abbildung 6.5-54: Lokaler Regelkreis eines Verbrauchers mit Vorhersagen $j = 1, \ldots, H_p$ über den Prädiktionshorizont

Mithilfe der Information über die zukünftige Gesamtvorlauftemperatur, die die Verbraucher vom MPC des Kessels erhalten, und den berechneten Trajektorien der Stellsignale des Verbrauchers können Vorhersagen über die zukünftige Rücklauftemperatur sowie den zukünftigen Volumenstrom und die zukünftig angeforderte Vorlauftemperatur gemacht werden, indem eine Vorwärtssimulation des Modells (6.5-166) und (6.5-167) über den Vorhersagehorizont berechnet wird.

Kessel
Im Reglerknoten für den Kessel wird das Modell (6.5-161) für den AMPC-SL Algorithmus verwendet. Auf Grundlage der Vorhersage des Verbrauchs der Heizkreise wird dem Regler eine Referenz für die Sollvorlauftemperatur vorgegeben. Dieser soll der Kessel folgen, sodass sich die Kostenfunktion

$$J_K = \sum_{j=1}^{H_p} \left\| T_{Vl,ges}(k+j) - T_{Vl,ges,ref}(k+j) \right\|_{q_K} + \sum_{j=0}^{H_u-1} \left\| \Delta\alpha(k+j) \right\|_{\mathbf{R}_K}$$

ergibt. Die Störgrößen des Reglers sind die Gesamtrücklauftemperatur und der Gesamtvolumenstrom, die als Vorhersage aus den Reglern der Heizkreise vorliegen. Der Regelkreis für den Kessel folgt daraus wie in Abbildung 6.5-55 dargestellt.

Die zukünftige Vorlauftemperatur kann über eine Vorwärtssimulation des Modells (6.5-161) mit dem berechneten Stellsignal und den Störgrößenvorhersagen bestimmt werden.

Koordinator
Der Reglerknoten des Kessels benötigt Informationen über die zukünftigen Sollvorlauftemperaturen, der

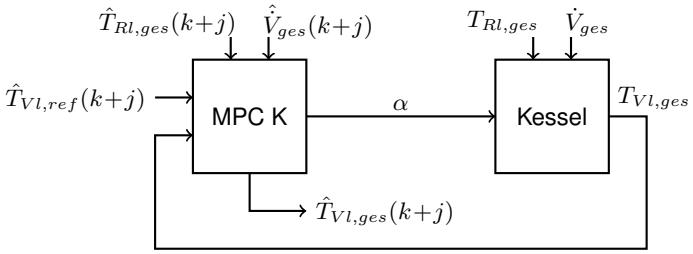


Abbildung 6.5-55: Lokaler Regelkreis des Kessels mit Vorhersagen $j = 1, \ldots, H_p$ über den Prädiktionshorizont

zukünftigen Gesamtrücklauftemperatur und den Gesamtvolumenstrom. Die Verbraucherkreise stellen diese Informationen jeweils für ihren Regelkreis zur Verfügung. Im Koordinator werden diese Informationen für den Kessel zusammengeführt. Abbildung 6.5-56 zeigt die Ein- und Ausgänge des Koordinators.

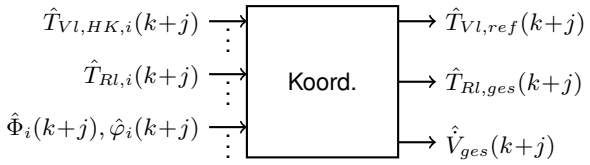


Abbildung 6.5-56: Koordinator des Kessels mit Vorhersagen $j = 1, \ldots, H_p$ über den Prädiktionshorizont

Somit erhält er die Informationen über zukünftige Rücklauftemperaturen und Volumenströme von den jeweiligen Verbrauchern. In der Anlage werden die Volumenströme der einzelnen Verbraucher über einen Vorlaufsammler zusammengeführt. Das Mischprodukt ergibt sich daher aus (6.5-162) und (6.5-163). Zudem wird aus den angeforderten zukünftigen Vorlauftemperaturen eine Referenz für die Gesamtvorlauftemperatur bestimmt. Dies erfolgt über eine Auswahl der maximalen Vorlaufanforderung von den Verbrauchern für jeden Zeitschritt des Vorhersagehorizonts

$$T_{Vl,ges,ref}(k+j) = \max_{i=1,\ldots,N_V} (T_{Vl,HK,i}(k+j)) \, \forall j = 1, \ldots, H_p.$$

Außerdem werden die zukünftigen Vorlauftemperaturen des Kessels an die Verbraucher gesendet. Der Aufbau des verteilten Reglers mit den zuvor beschrieben lokalen Reglerknoten ist in der Abbildung 6.5-57 gezeigt.

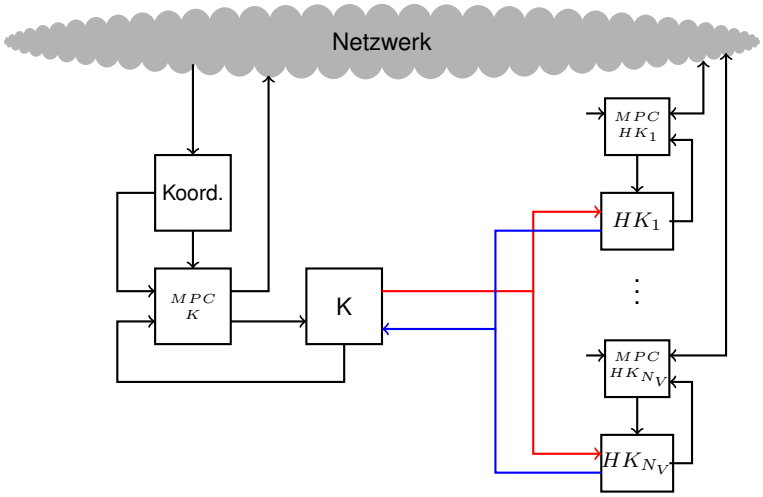


Abbildung 6.5-57: Verteilter prädiktiver Regelkreis für das Gesamtsystem

Für ein Simulationsbeispiel wurde hier eine Anlage mit $N_V = 3$ Heizkreisen betrachtet, für die ein zentraler MPC sowie ein verteilter MPC entworfen wurde. Es wurde eine Abtastzeit von $60\,\mathrm{s}$ verwendet und verschiedene Vorhersagehorizonte untersucht. Die Referenz für die Gebäudetemperaturen wird so vorgegeben, dass eine Nachtabsenkung erfolgt. Abbildung 6.5-58 zeigt die Simulationsergebnisse bei der Verwendung des dezentralen MPCs für die 3 Gebäudetemperaturen.

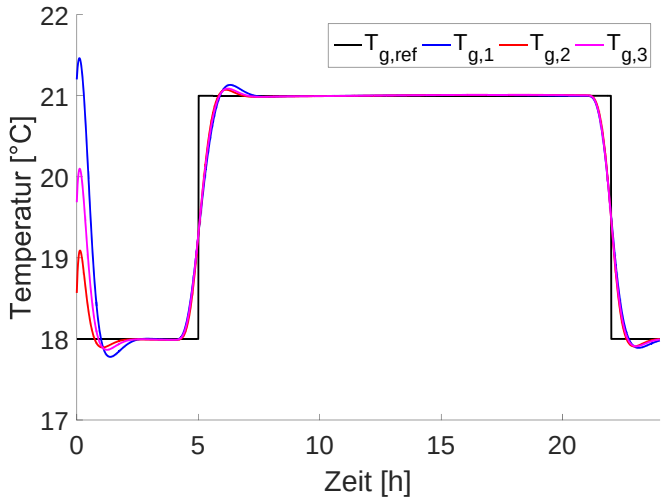


Abbildung 6.5-58: Simulationsergebnis der Gebäudetemperaturen für einen Tag

Es ist zu sehen, dass die Raumtemperaturen sehr gut der Referenz folgen, obwohl die Regelungsaufgabe auf mehrere Knoten verteilt wurde. Zudem wurde betrachtet, ob sich hier ein iteratives Verfahren notwendig ist. Dies würde bedeuten, dass innerhalb eines Abtastschrittes die jeweiligen Optimierung mehrmals iterativ berechnet werden. Dies brachte hier jedoch keine deutlichen Verbesserungen. Die Systemdynamik ist im Vergleich zu der Abtastzeit so langsam, dass es ausreichend ist die Optimierungsprobleme nur einmal während eines Abtastzeitschrittes zu berechnen.

Dies zeigt, dass der verteilt geschlossene Regelkreis das gewünschte Ergebnis liefert. Der große Vorteil gegenüber dem zentralen MPC ergibt sich bei der Betrachtung der Komplexität. Hierbei bezeichnet t_{MPC} die Zeit, die für die Lösung des Optimierungsproblems des zentralen MPCs benötigt wird. Das Optimierungsproblem des dezentralen MPCs wird in der Zeit t_{DMPC} gelöst. Sie setzt sich zusammen aus den Zeiten, die zur Lösung der Optimierungsprobleme der einzelnen Reglerknoten der Kessel t_K und Verbraucher $t_{HK,i}$ notwendig sind. Zwischen den Optimierungsproblemen der Verbraucher gibt es keine Querkopplungen, sodass diese parallel berechnet werden können. Mit den so bestimmten zukünftigen Verbrauchsdaten kann dann der MPC des Kessels berechnet werden. Somit ergibt sich die Berechnungsdauer des dezentralen MPC aus der Summer der Zeit die für die Lösung des Optimierungsproblems des Kessels benötigt wird und der maximalen Zeit für die Lösung der Verbraucherprobleme, da diese parallel berechnet werden

$$t_{DMPC} = t_K + \max_{i=1,\ldots,N_V} t_{HK,i}.$$

Die Simulation wurde auf einem Computer mit einem Intel Core i7-3540M Prozessor mit $3.00\,\mathrm{GHz}$ und einem Arbeitsspeicher von $16\,\mathrm{GB}$ RAM durchgeführt. Die durchschnittlich benötigten Zeiten für die Berechnung der Optimierung sind in der Tabelle 6.5-4 zusammengefasst.

Tabelle 6.5-4: Vergleich der Simulationszeiten

$\mathbf{H_p}$	$\mathbf{t_{MPC}}$	$\mathbf{t_K}$	$\mathbf{t_{HK,1}}$	$\mathbf{t_{HK,2}}$	$\mathbf{t_{HK,3}}$	$\mathbf{t_{DMPC}}$
$60\,\mathrm{min}$	$3.39\,\mathrm{s}$	$0.045\,\mathrm{s}$	$0.115\,\mathrm{s}$	$0.121\,\mathrm{s}$	$0.118\,\mathrm{s}$	$0.166\,\mathrm{s}$

Der Prädiktionshorizont ist auf eine Stunde festgelegt und ist gleich dem Stellhorizont, was bei einer Abtastzeit von $T_{sample} = 60\,\mathrm{s}$ auch 60 Zeitschritten entspricht. Der zentrale MPC berechnet somit die Stellgrößenfolge für 7 Stellsignale, da die Verbraucher jeweils 2 Stellsignale und der Kessel 1 Stellsignal besitzt. Somit ergeben sich für das Gesamtproblem ein Suchraum von $7 \cdot 60 = 420$ Optimierungsvariablen. Die Reglerknoten des verteilten MPC haben jeweils nur Einfluss auf die Aktoren des jeweiligen Subsystems. Somit ergeben sich für den Kessel mit einem Stellsignal $1 \cdot 60 = 60$ Optimierungsvariablen und für die Verbraucher mit zwei Stellsignalen jeweils $2 \cdot 60 = 120$ Optimierungsvariablen. Diese unterschiedliche Komplexität zeigt sich auch deutlich in den Zeiten in Tabelle 6.5-4. Durch die Aufteilung der Regelungsaufgabe auf mehrere Knoten kann eine deutlich geringe Berechnungszeit $t_{MPC} > t_{DMPC}$ erreicht werden, da die einzelnen Probleme der Reglerknoten deutlich schneller berechnet werden, als das Optimierungsproblem des zentralen MPC. Bereits bei den einzelnen Subproblemen ist zu erkennen, dass das Problem des Kessels mit nur einem Stellsignal schneller zu lösen ist, als das Problem der Verbraucher. Es ergibt sich verglichen mit dem zentralen Problem ein deutlicher Vorteil. Erhöht man den Vorhersage- und Stellhorizont, vergrößert sich auch die Komplexität des Optimierungsproblems, da sich auch der Suchraum vergrößert. In der Abbildung 6.5-59 ist dargestellt, wie sich die Zeit pro Optimierung für den zentralen und den verteilten MPC für verschiedene Vorhersagehorizonte entwickelt.

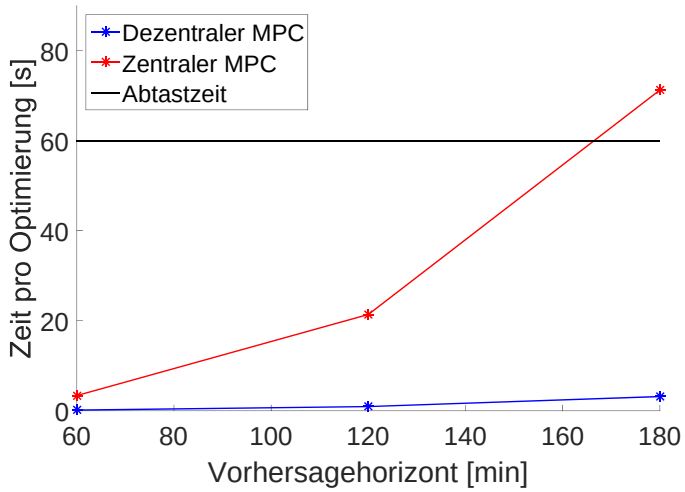


Abbildung 6.5-59: Zeit für die Optimierung für verschiedene Vorhersagehorizonte

Es ist deutlich zu erkennen, dass die Zeit für die Optimierung im zentralen Fall deutlich stärker ansteigt. Bereits für einen Vorhersagehorizont von 3 Stunden benötigt die die Optimierung im zentralen Fall länger als die Abtastzeit von $60\,\mathrm{s}$, was für die Echtzeitimplementierung nicht akzeptabel wäre. Der Zeitbedarf für den verteilten MPC zeigt einen deutlich geringeren Anstieg, sodass hier auch noch deutlich längere Vorhersagehorizonte möglich wären. Somit lässt sich abschließend sagen, dass die Komplexität gemessen an der Lösungsdauer der verteilten Probleme deutlich geringer ist als im zentralen Fall. Zudem können die verteilten Subprobleme teilweise parallel gelöst werden. Der verteilte Ansatz ist auch für große Vorhersagehorizonte zu verwenden, da die Komplexität deutlich langsamer ansteigt als für das zentrale Problem. Zudem ist zu erwarten, dass auch mit weiteren Heizkreisen der Aufwand des verteilten Problems geringer ansteigt als im zentralen Fall, zumal die einzelnen Subprobleme der Verbraucher parallel gelöst werden können. Somit ergibt sich gerade für große Anlagen ein Vorteil bei der Verwendung eines verteilten Ansatzes.

6.6 MTI Toolbox

Vorangegangene Untersuchungen haben gezeigt, dass multilineare zeitinvariante (MTI) Systeme sich für die Modellierung von Systemen im Gebäudebereich sehr gut eignen. Tensoren und Tensordekompositionsverfahren helfen dabei diese Modelle effektiv darzustellen und anzuwenden, gerade um das

Verhalten von sehr großen Anlagen abbilden zu können. Um diese MTI Systeme für die Modellierung und Reglerentwicklung nutzen zu können, wurde die MTI Toolbox für MATLAB /Simulink entwickelt. Diese bietet die Möglichkeit bekannte Tensorverfahren aus der Mathematik, wie die CP Tensorzerlegung oder Tensor Trains, für MTI Systeme anzuwenden. Die Toolbox enthält Methoden zur Modelldarstellung, Simulation und dem Reglerentwurf.

6.6.1 Überblick

Viele der vorgestellten Methoden für MTI Systeme wurden in MATLAB /Simulink implementiert und in einer Toolbox zusammengefasst. Die entwickelte MTI Toolbox bietet die Möglichkeit einfach auf die entwickelten Algorithmen zuzugreifen und diese zu verwenden. Die implementierten Funktionen nutzen Tensormethoden von bekannten mathematischen Tensortoolboxen, [2], [50], [38], [22] und wenden diese in der Modellierung und dem Reglerentwurf für MTI Systeme an. Die Hauptbestandteile sind in der Abbildung 6.6-60 dargestellt.

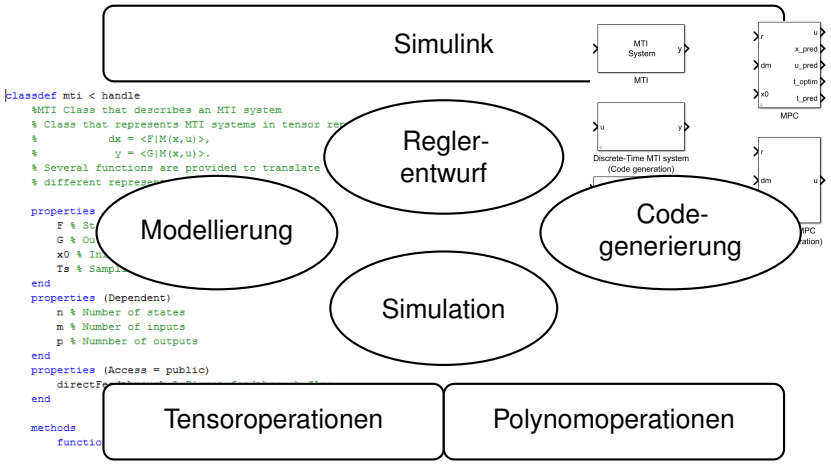


Abbildung 6.6-60: Überblick über die MTI Toolbox

Die einzelnen Bereiche der MTI Toolbox enthalten:

- *Tensoroperationen*: Tensoroperationen wie die Transformation zwischen verschiedenen dekomponierten Darstellungen, z.B. von der CP in die Tucker, TT oder HT Darstellung , oder spezielle Tensoroperationen für CP Tensoren, wie das kontrahierte Produkt oder das äußere Produkt.
- *Polynomoperationen*: Methoden für die Darstellung von Polynomen der Ordnung N und Operationen wie die Multiplikation und die partielle Differentiation auf Basis der Parametertensoren
- *Modellierung*: Darstellung von dekomponierten MTI Systemen in kontinuierlicher und diskreter Zeit
- *Simulation*: Simulationsmethoden für MTI Systeme unter Verwendung der dekomponierten Struktur der Parametertensoren
- *Reglerentwurf*: Reglerentwurfsmethoden für MTI Systeme wie die Feedback Linearisierung
- *Simulink*: Simulink Bibliothek für die Simulation von MTI Systemen und deren Reglern in der Simulink Umgebung.
- *Codegenerierung*: Ausgewählte Funktionen mit angepasster Implementierung, so dass sie für die Codegenerierung geeignet sind.

Somit bietet die MTI Toolbox viele Werkzeuge für die Arbeit mit MTI Systemen im Bereich des modellbasierten Reglerentwurfs von der Systemrepräsentation über die Simulation bis hin zum Reglerdesign. Diese einzelnen Methoden werden im Folgenden näher dargestellt.

6.6.2 Klassendokumentation

Die MTI Toolbox ist in einer objektorientierten Weise implementiert. Unterschiedliche Bereiche bei der Arbeit mit MTI Systemen mit ihren Attributen und Methoden sind in verschiedenen Klassen strukturiert. Die allgemeinste Klasse ist die Klasse *CPtensor_operations*, in der Basis-Tensoroperationen für CP dekomponierte Tensoren hinterlegt sind, welche über die Funktionalität der Standard Toolboxen, d.h. Tensor Toolbox und Tensorlab, die für die Rechnung mit CP Tensoren verfügbar sind, hinaus geht, [2], [50]. Für die Darstellung und Rechnung mit Polynomen in Tensordarstellung ist die Klasse *polynomial_operations* vorgesehen. Die Darstellung und Analyse von allgemeinen MTI Systemen oder affinen MTI Systemen mit verschiedenen Dekompositionsverfahren sowohl zeitdiskret als auch zeitkontinuierlich kann mithilfe der Klassen *mti* oder *input_affine_mti* erfolgen. Die Methode der Feedback Linearisierung für MTI Systeme aus Kapitel 6.2 ist in der Klasse *feedback_linearization* implementiert. Einige der Klassen treten doppelt auf, einmal mit der Endung *_codegen* einmal ohne. Die Klassen mit der Endung enthalten einige Funktionen aus der jeweiligen Klasse mit einer angepassten Implementierung, sodass nur Funktionen und Variablen verwendet werden, die in MATLAB für die Codegenerierung geeignet sind. Zusammenfassend sind die Klassen in der Tabelle 6.6-5 dargestellt.

Tabelle 6.6-5: Klassen der MTI Toolbox

Klasse	**Beschreibung**
`CPtensor_operations`	Operationen für CP Tensoren
`CPtensor_operations_codegen`	Operationen für CP Tensoren für die Codegenerierung geeignet
`tensor_operations`	Operationen für die Arbeit mit Tensoren
`polynomial_operations`	Rechnung mit Polynomen in Tensordarstellung
`polynomial_operations_codegen`	Rechnung mit Polynomen in Tensordarstellung für die Codegenerierung geeignet
`mti`	Darstellung von MTI Systemen
`mti_functions_codegen`	Darstellung von MTI Systemen für die Codegenerierung geeignet
`input_affine_mti`	Darstellung von affinen MTI Systemen
`simulateMTI`	Simulation von MTI Systemen
`feedback_linearization`	Feedback Linearisierung für MTI Systeme

Im Folgenden werden die einzelnen Klassen mit ihren Attributen und Methoden kurz näher vorgestellt.

CPtensor_operations
Verschiedene Operationen, die für volle Tensoren jedoch nicht für CP Tensoren in den Standard Toolboxen, der Tensor Toolbox, [2] und Tensorlab, [50], definiert sind, werden hier auf CP Tensoren angepasst, sodass das berechnete Ergebnis bei Operanden in CP Darstellung auch direkt als CP Tensor vorliegt. Während der Berechnung des Algorithmus wird nur mit den Dekompositionsfaktoren der Operanden gearbeitet. Eine volle Darstellung wird nicht erstellt. Somit ermöglichen diese Funktionen Berechnungen mit Tensoren in CP Darstellung auch für sehr große Tensoren mit vielen Dimensionen.

Attribute

- Statische Klasse, die nur Methoden und keine Attribute enthält.

Methoden

- `squeezeCP`:
  Dimensionen der Größe 1 eines CP Tensors werden detektiert und entfernt, indem die dazugehörige Faktormatrix, in diesem Fall ein Vektor, über eine elementweise Multiplikation in den Wichtungsvektor integriert wird.
- `reduceRank`:
  Es wird versucht die Anzahl an Rang-1 Komponenten eines CP Tensors zu reduzieren. Dies geschieht zum einen durch Entfernen von Nullelementen im Wichtungsvektor. Die Spalten aller Faktormatrizen, die zu diesem Eintrag gehören, haben keinen Einfluss auf den vollen Tensor. Zudem werden Nullspalten aus den Faktormatrizen entfernt. Hat eine Faktormatrix eine Spalte voller Nullen, so kann diese Spalte aus allen Faktormatrizen und dem Wichtungsvektor entfernt werden, da diese Einträge beim Erstellen des vollen Tensors immer mit Null multipliziert werden.

Diese beiden Möglichkeiten werden geprüft, um die Anzahl an Rang-1 Komponenten eines Tensors zu reduzieren. Der resultierende Tensor hat unter Umständen weniger Rang-1 Komponenten, bildet den ursprünglichen Tensor jedoch immer noch exakt ab. Es erfolgt keine Approximation, sondern es werden nur strukturell verzichtbare Rang-1 Komponenten entfernt.

- `concatenateCP`:
  Die Funktion implementiert die Konkatenation von M Tensoren $\mathsf{X}_i,\ i = 1, \ldots, M$ der Ordnung k in CP Darstellung, die alle die gleichen Dimensionen $I_1 \times I_2 \times \cdots \times I_k$ haben. Ein Tensor Y der Dimension $M \times I_1 \times I_2 \times \cdots \times I_k$ wird erstellt, sodass die Tensoren in der ersten Dimension von Y verbunden werden, d.h. $\mathsf{Y}(m, :, \ldots, :) = \mathsf{X}_m$.
- `outerCP`:
  Das äußere Produkt zweier Tensoren in CP Darstellung wird berechnet, sodass auch das Ergebnis als CP Tensor zurückgegeben wird. Während der Berechnung wird kein voller Tensor erstellt, sondern nur mit den Faktormatrizen der Operanden gearbeitet.
- `contractCP`:
  Das kontrahierte Produkt zweier Tensoren in CP Darstellung entlang beliebiger Moden wird berechnet. Das Ergebnis ist ein CP Tensor, dessen Faktoren direkt berechnet werden.

CPtensor_operations_codegen
Die CP Tensorformate aus der Tensor Toolbox und Tensorlab arbeiten mit den Matlab Datentypen *Struct* und *Cell*. Da diese Datentypen nicht für die Codegenerierung unter MATLAB unterstützt werden, muss eine alternative Speicherform für die Parametertensoren von MTI Systemen gefunden werden. In Kapitel 6.7.2 wird daher ein Speicherformat beschrieben, das auch für die Codegenerierung verwendet werden kann. Es müssen ein dreidimensionales Array F_{xu}, die Faktormatrix $\mathbf{F}_\Phi$ und der Wichtungsdvektor $\boldsymbol{\lambda}_F$ übergeben werden. Diese drei Arrays sind alle vom Datentyp *double* und somit für die Codegenerierung geeignet. Mit diesen Arrays ist der Tensor komplett beschrieben. Die hier beschriebene Klasse enthält Funktionen, um die Parametertensoren von dem Standard CP Format in das Speicherformat, das für die Codegenerierung geeignet ist, zu überführen und stellt Berechnungsfunktionen für dieses Format zur Verfügung.

Attribute

- Statische Klasse, die nur Methoden und keine Attribute enthält.

Methoden

- `cp2numeric`:
  Aus dem CP Tensorformat der Tensor Toolbox wird die Darstellung für CP Tensoren erzeugt, die für die Codegenerierung geeignet ist.
- `ttm`:
  Tensor-Matrix Produkt für CP Tensoren im Speicherformat für die Codegenerierung. Auch das Ergebnis ist in diesem Format.
- `reduceRank`:
  Durch Entfernen von Nulleinträgen wird versucht die Anzahl der Rang-1 Komponenten zu reduzieren, wie in der vorangegangen Klasse beschrieben. Hier ist nur die Anpassung erfolgt, dass nur Darstellungen und Operationen verwendet werden, die für die Codegenerierung geeignet sind.

tensor_operations
Die Klasse enthält Funktionen für die Arbeit mit Tensoren, die die einzelnen Tensortoolboxen erweitern oder kombinieren, wie Transformationen zwischen verschiedenen dekomponierten Darstellungen.

Attribute

- Statische Klasse, die nur Methoden und keine Attribute enthält.

Methoden

- `cp2ttTens`:
  Transformation eines Tensors von der CP Darstellung in die TT Darstellung.
- `cp2tuckerTens`:
  Transformation eines Tensors von der CP Darstellung in die Tucker Darstellung.

- `cp2htTens`:
  Transformation eines Tensors von der CP Darstellung in die HT Darstellung.
- `memReq`:
  Berechnung des Speicheraufwands für die volle oder dekomponierte Tensordarstellung. Um den Speicheraufwand zu messen, wird die Anzahl der Elemente, die für die einzelnen Darstellungen gespeichert werden müssen, bestimmt.

polynomial_operations
Polynome können mithilfe von Tensoren dargestellt werden. Operationen wie die Multiplikation oder die Differentiation können auf Basis der Parametertensoren berechnet werden. Die Methoden arbeiten mit Tensoren in CP Darstellung. Das Ergebnis wird auch als CP Tensor zurückgegeben.

Attribute

- Statische Klasse, die nur Methoden und keine Attribute enthält.

Methoden

- `multiply`:
  Die Multiplikation zweier Polynome auf Basis der Parametertensoren wird berechnet. Die Parametertensoren beider Polynome werden als CP Tensoren angegeben und der Parametertensor der Multiplikation in CP Darstellung nach (6.2-69) zurückgegeben.
- `differentiate`:
  Die partielle Ableitung eines Polynoms nach einer Variablen wird berechnet. Zurückgegeben wird der Parametertensor des Resultats.
- `lie_derivative`:
  Berechnung der Lie Ableitung eines skalaren Polynoms entlang eines Vektorfeldes auf Basis der Parametertensoren.
- `increase_order`:
  Der Parametertensor eines Polynoms der Ordnung N_{old} wird so erweitert, dass das Polynom bezüglich eines Monomtensors höherer Ordnung $N_{new} > N_{old}$ ausgedrückt werden kann.
- `buildMonomial`:
  Konstruktion eines Monomtensors für ein multilineares Polynom, d.h. der Ordnung $N = 1$.
- `buildMonomialN`:
  Konstruktion eines Monomtensors für ein Polynom beliebiger Ordnung N.
- `eval_poly`:
  Auswertung eines Polynoms an einem gegebenen Punkt.
- `jacobian`:
  Berechnung und Auswertung der Jacobi-Matrix eines Polynoms an einem gegebenen Punkt.
- `hessian`:
  Berechnung und Auswertung der Hesse-Matrix eines Polynoms an einem gegebenen Punkt.
- `eval_poly_jacob`:
  Auswertung des Polynoms und der Jacobi-Matrix an einem gegebenen Punkt.
- `findRoot`:
  Numerische Berechnung der Nullstellen eines Polynoms.

polynomial_operations_codegen
Polynome können mithilfe von Tensoren dargestellt werden. Einige Operationen dazu werden hier so angepasst, dass sie für die Codegenerierung geeignet sind. Die Methoden arbeiten mit dem Speicherformat für CP Parametertensoren für MTI Systeme, das die Codegenerierung unterstützt.

Attribute

- Statische Klasse, die nur Methoden und keine Attribute enthält.

Methoden

- `differentiate`:

Die partielle Ableitung eines Polynoms nach einer Variablen wird berechnet. Zurückgegeben wird der Parametertensor des Resultats.

- `eval_poly`:
  Auswertung eines Polynoms an einem gegebenen Punkt.
- `jacobian`:
  Berechnung und Auswertung der Jacobi-Matrix eines Polynoms an einem gegebenen Punkt.

mti
Die Darstellung von MTI Systemen ist mit dieser Klasse möglich. Die Definition von MTI Systemen ist in der Matrix- oder der Tensordarstellung möglich. Die Parametertensoren können in voller oder dekomponierter Form angegeben werden. Die Toolbox unterstützt die CP, Tucker, TT und HT Dekomposition. Neben der Definition von MTI Systemen sind zudem weitere Funktionen z.B. für die Linearisieung oder die Diskretisierung implementiert.

Attribute

- `F`:
  Parametertensor `F` der Zustandsgleichung
- `G`:
  Parametertensor `G` der Ausgangsgleichung
- `x0`:
  Anfangswert der Zustände
- `Ts`:
  Abtastzeit (gleich 0 für zeitkontinuierliche Modelle)
- `n`:
  Anzahl der Zustände
- `m`:
  Anzahl der Eingänge
- `p`:
  Anzahl der Ausgänge
- `directFeedthrough`:
  Angabe, ob es einen direkten Durchgriff von den Eingängen zu den Ausgängen gibt

Methoden

- `mti`:
  Konstruktor der Klasse, um eine Instanz zu erstellen.
- `set.G`:
  Der Ausgangstensor des MTI Systems wird gesetzt. Wenn kein bestimmter Ausgangstensor angegeben wird, wird angenommen, dass alle Zustände gemessen werden. Zudem erfolgt eine Analyse, ob es einen direkten Durchgriff von den Eingängen zu den Ausgängen gibt.
- `get.n`:
  Bestimmung der Anzahl der Zustände des Systems
- `get.m`:
  Bestimmung der Anzahl der Eingänge des Systems
- `get.p`:
  Bestimmung der Anzahl der Ausgänge des Systems
- `tens2matMTI`:
  Transformation eines MTI Systems von der Tensor- in die Matrixdarstellung
- `mat2tensF`:
  Transformation der Parameter der Zustandsgleichung eines MTI Systems von der Matrix- in die Tensordarstellung
- `mat2tensG`:

Transformation der Parameter der Ausgangsgleichung eines MTI Systems von der Matrix- in die Tensordarstellung

- `cp2ttMTI`:
  Transformation eines MTI Systems von der CP in die TT Darstellung
- `cp2tuckerMTI`:
  Transformation eines MTI Systems von der CP in die Tucker Darstellung
- `cp2htMTI`:
  Transformation eines MTI Systems von der CP in die HT Darstellung
- `buildFtens`:
  Konstruktion eines Parametertensors der Zustandsgleichung eines MTI Systems. Die rechte Seite wird durch ein *Cell* Array angegeben, aus dem der Parametertensor in CP Darstellung erstellt wird. Jede Zeile des *Cell* Arrays steht für einen Term der rechten Seite. Die erste Spalte des *Cell* Arrays gibt an, zu welcher Komponente der Vektorfunktion der Term gehört. Die zweite und dritte Spalte zeigen, welche Zustände und Eingänge verwendet werden. Die vierte Spalte gibt den dazugehörigen Parameterwert an. Die Zustandsgleichung des MTI Systems

  $$\begin{aligned}\dot{x}_1 &= 2x_1 + 3x_2 + 4x_1x_2,\\ \dot{x}_2 &= 5x_1x_2 + 1x_2u_1,\end{aligned}$$

  wird z.B. dargestellt durch

  $$\texttt{rhs} = \begin{matrix}\{1, & [1], & [\,], & 2;\\ 1, & [2], & [\,], & 3;\\ 1, & [1\ 2], & [\,], & 4;\\ 2, & [1\ 2], & [\,], & 5;\\ 2, & [2], & [1], & 1\};\end{matrix}$$

- `buildGtens`:
  Konstruktion eines Parametertensors der Ausgangsgleichung eines MTI Systems. Die rechte Seite wird durch ein *Cell* Array angegeben, aus dem der Parametertensor in CP Darstellung erstellt wird. Die Konstruktion des *Cell* Arrays erfolgt analog zu der Zustandsgleichung.
- `scaleMTI`:
  Skalierung eines MTI Systems, sodass die Eingangs-, Ausgangs- und Zustandssignale in einem gewünschten Bereich liegen.
- `memReq`:
  Berechnung des Speicheraufwands für die volle oder dekomponierte Tensordarstellung eines MTI Systems. Um den Speicheraufwand zu messen, wird die Anzahl der Elemente, die für die einzelnen Parametertensoren gespeichert werden müssen, bestimmt.
- `buildDefaultOut`:
  Erstellung eines Ausgangstensors, sodass der Ausgangsvektor gleich dem Zustandsvektor ist, d.h. alle Zustände werden gemessen.
- `trimMTI`:
  Berechnung eines stationären Punktes $\dot{\mathbf{x}} = 0$ eines MTI Systems mit Standardsolvern für nichtlineare Funktionen unter Verwendung der Jacobimatrix der Zustandsfunktion.
- `linearizeMTI`:
  Linearisierung eines MTI Systems um einen gegebenen Arbeitspunkt auf Basis der Parametertensoren.
- `checkLocalStability`:
  Die lokale Stabilität eines MTI Systems in der Umgebung eines gegebenen Punktes wird bestimmt, indem das MTI System um diesen Punkt linearisiert und die Stabilität des linearen System untersucht wird.
- `c2dMTI`:
  Diskretisierung eines zeitkontinuierlichen MTI Systems durch eine Euler Approximation erster

Ordnung

$$\dot{\mathbf{x}} \approx \frac{\mathbf{x}(k+1) - \mathbf{x}(k)}{T_s},$$

mit der Abtastzeit T_s.

mti_functions_codegen
Funktionen für MTI Systeme, die für die Codegenerierung verwendet werden können, sind in dieser Klasse enthalten. Die Methoden nutzen das entsprechende Speicherformat für CP Tensoren und verwenden nur Operationen, die von der Codegenerierung unterstützt werden.

Attribute

- Statische Klasse, die nur Methoden und keine Attribute enthält.

Methoden

- `linearizeMTI`:
  Linearisierung eines MTI Systems um einen gegebenen Arbeitspunkt auf Basis der Parametertensoren, die als CP Tensoren in dem Speicherformat für die Codegenerierung angegeben werden.

input_affine_mti
Affine MTI Systeme können mit dieser Klasse dargestellt werden

$$\begin{aligned}\dot{\mathbf{x}} &= \langle\, \mathsf{F} \mid \mathsf{M}(\mathbf{x}, \mathbf{u}) \,\rangle_{=} \mathbf{a}(\mathbf{x}) + \mathbf{b}(\mathbf{x})\mathbf{u} = \langle\, \mathsf{A} \mid \mathsf{M}(\mathbf{x}) \,\rangle \; + \langle\, \mathsf{B} \mid \mathsf{M}(\mathbf{x}) \,\rangle \; \mathbf{u}, \\ \mathbf{y} &= \langle\, \mathsf{G} \mid \mathsf{M}(\mathbf{x}, \mathbf{u}) \,\rangle_{=} c(\mathbf{x}) = \langle\, \mathsf{C} \mid \mathsf{M}(\mathbf{x}) \,\rangle .\end{aligned}$$

Die Klasse erbt die Attribute und Methoden von der Klasse `mti` und erweitert diese Klasse.

Attribute

- `A`:
  Parametertensor A
- `B`:
  Parametertensor B
- `C`:
  Parametertensor C

Methoden

- `input_affine_mti`:
  Konstruktor der Klasse, um eine Instanz zu erstellen.
- `get.A`:
  Bestimmung des Parametertensors A aus dem Tensor F der allgemeinen MTI Darstellung.
- `get.B`:
  Bestimmung des Parametertensors B aus dem Tensor F der allgemeinen MTI Darstellung.
- `get.C`:
  Bestimmung des Parametertensors C aus dem Tensor G der allgemeinen MTI Darstellung.
- `set.A`:
  Bestimmung des Parametertensors F der allgemeinen MTI Darstellung aus den Parametertensoren A und B, wenn sich A ändert.
- `set.B`:
  Bestimmung des Parametertensors F der allgemeinen MTI Darstellung aus den Parametertensoren A und B, wenn sich B ändert.
- `set.C`:
  Bestimmung des Parametertensors G der allgemeinen MTI Darstellung aus dem Parametertensor C, wenn sich C ändert.

simulateMTI
Funktionen für die Simulation von MTI Systemen sind hier implementiert. Die Auswertung der rech-

ten Seiten der MTI Systeme erfolgt auf Basis der Dekompositionsfaktoren. Die Simulation wird mit Standardsolvern durchgeführt.

Attribute

- `sys`:
  MTI System
- `usim`:
  Eingangssignal
- `tsim`:
  Simulationszeit
- `xsim`:
  Simulierte Zustandstrajektorie
- `ysim`:
  Simulierte Ausgangstrajektorie

Methoden

- `simulateMTI`:
  Konstruktor der Klasse, um eine Instanz zu erstellen.
- `oneStep`:
  Berechnung der rechten Seiten von Zustands- und Ausgangsgleichung eines MTI Systems in Tensordarstellung durch Auswertung des kontrahierten Produkts von Parametertensor und Monomtensor an einem gegebenen Punkt.
- `oneStepDx`:
  Berechnung der rechten Seite von der Zustandsgleichung eines MTI Systems in Tensordarstellung durch Auswertung des kontrahierten Produkts von Parametertensor und Monomtensor an einem gegebenen Punkt.
- `oneStepY`:
  Berechnung der rechten Seite der Ausgangsgleichung eines MTI Systems in Tensordarstellung durch Auswertung des kontrahierten Produkts von Parametertensor und Monomtensor an einem gegebenen Punkt.
- `simContEulerMTI`:
  Simulation eines zeitkontinuierlichen MTI Systems mit einem Vorwärts-Euler-Verfahren.
- `simContOde45MTI`:
  Simulation eines zeitkontinuierlichen MTI Systems mit einem ODE45-Verfahren.
- `simDiscMTI`:
  Simulation eines zeitdiskreten MTI Systems.
- `plotSim`:
  Darstellung der Simulationsergebnisse in einem Plot. Die Eingangs-, Zustands- und Ausgangssignale werden über der Zeit aufgetragen.
- `createLabel`:
  Erzeugung einer Legende für den Plot der Simulationsergebnisse.

feedback_linearization
Ein Regler für die Feedback Linearisierung für MTI Systeme wird auf Basis der Parametertensoren bestimmt, wie in Kapitel 6.2.3 beschrieben.

Attribute

- `sys`:
  Regelstrecke, durch ein affines MTI System angegeben
- `lin_coeff`:
  Koeffizienten der Differentialgleichung des gewünschten linearen Verhaltens im geschlossenen Kreis

- `rho`:
  Relativer Grad oder Index der Strecke
- `controller_num`:
  Zählerpolynom des Reglers
- `controller_den`:
  Nennerpolynom des Reglers
- `prefilter_num`:
  Zählerpolynom des Vorfilters

Methoden

- `feedback_linearization`:
  Konstruktor der Klasse, um eine Instanz zu erstellen.
- `compute_controller`:
  Berechnung der Parametertensoren für die Tensordarstellung des Reglers der Feedback Linearisierung

Simulink Bibliothek
Die MTI Toolbox stellt eine Simulink Bibliothek zur Verfügung, die es ermöglicht MTI Systeme und deren Regler auch in Simulink zu verwenden und zu simulieren.

Blöcke

- MTI System:
  Simulation eines zeitkontinuierlichen MTI Systems als Instanz der Klasse `mti` in Simulink.
- Discrete-time MTI System:
  Simulation eines zeitdiskreten MTI Systems in Simulink. Es werden nur Funktionen und die Speicherdarstellung für CP Tensoren verwendet, die auch für die Codegenerierung geeignet sind, sodass mit diesem Block MTI Systeme auf Hardwarekomponenten wie einem Raspberry Pi implementiert werden können.
- Feedback Linearization:
  Simulink Implementierung eines Reglers der Feedback Linearisierung für MTI Systeme. Der Regler wird durch eine Instanz der Klasse `feedback_linearization` angegeben.
- Model predictive controller:
  Implementierung des prädiktiven Reglers mit linearem Modell und quadratischer Kostenfunktion nach, [34]. Der Ansatz wurde hier erweitert und auf MTI Systeme angepasst mit der Implementierung der sukzessiven Linearisierung wie in Kapitel 6.4.2 vorgestellt. Zudem ist es möglich, durch eine Vorwärtssimulation mit der bestimmten optimalen Eingangstrajektorie eine Vorhersage des zukünftigen Systemverhaltens zu berechnen.
- Adaptive model predictive controller:
  Durch diesen Block ist der MPC Algorithmus mit sukzessiver Linearisierung aus Kapitel 6.4.2 für die Echtzeitanwendung implementiert. Die Arbeitspunkte werden als vorberechneter Tensor übergeben. Das Modell wird in CP Darstellung in dem Speicherformat für die Codegenerierung abgelegt. In jedem Abtastschritt wird das MTI Modell linearisiert und das MPC Optimierungsproblem mit dem linearen Modell berechnet. Aus dem Block lässt sich Code generieren, der sich dann für die Anwendung an realen Anlagen auf externen Hardwarekomponenten verwenden lässt.

6.7 Anwendung am Demonstrationsgebäude

Abschließend erfolgt die Demonstration der entwickelten Regelungsansätze am Demonstrationsgebäude. Die Regelungsverfahren des AMPC-SL aus dem Kapitel 6.4 erwies sich als besonders gut geeignet für die Demonstrationsanwendung, da es u.A. eine Mehrgrößenregelung erlaubt und den Einfluss von Störgrößen, wie sie bei der Gebäudeanwendung z.B. durch das Wetter auftreten, direkt berücksichtigt. Für große Gebäude kann dieser Ansatz in einer dezentralen Regelungsstruktur eingesetzt werden, wie in Kapitel 6.5 beschrieben. Für einen solchen dezentralen Einsatz zeigt sich das Demonstrationgebäude der

LMT Group Schwarzenbek als sehr gut geeignet. Die Erzeuger sowie mehrere Verbraucher sind räumlich voneinander getrennt auf einem Werksgelände verteilt. Aufgrund dieser Anlagenstruktur bietet es sich an, diese verteilte Anordnung auch in die Regelungsstruktur zu übernehmen. Jede Komponente würde damit einen einzelnen Reglerknoten erhalten, wie im Abschnitt 6.5 beschrieben. Ein entsprechendes Reglerkonzept wurde entworfen. Es ergaben sich jedoch Probleme bei der Umsetzung, wie im Arbeitspaket AP B.5 beschrieben. Daher erfolgte hier keine Umsetzung dieses Konzepts. Jedoch zeigte sich einer Analyse, dass auch das Demonstrationsgebäude der Kieback&Peter GmbH eine geeignete Verbraucherstruktur hat.

Somit soll das zuvor entwickelte Verfahren des AMPC-SL für MTI Systeme an einem Heizkreis des Bürogebäudes von Kieback&Peter angewendet werden. Dazu muss er auf einer echtzeitfähigen Hardwarekomponente implementiert werden. Das folgende Kapitel stellt die verwendeten Hard- und Softwarekomponenten vor. Für den Test der Implementierung wurde eine Hardware in the Loop (HiL) Umgebung entworfen. Nach erfolgreichen Tests erfolgte der Einsatz am realen Gebäude. Die Demonstrationsergebnisse werden abschließend vorgestellt.

6.7.1 Hardware und Software

Um den entwickelten Ansatz des MPC mit sukzessiver Linearisierung an einem realen Gebäude anwenden zu können, muss dieser auf einer Hardwareplattform mit bestimmter Software implementiert werden. Diese werden hier kurz vorgestellt.

Hardware

Die Wahl bei der Hardwareplattform fiel auf den Einplatinencomputer Raspberry Pi. Hierbei handeltet es sich einen Computer mit geringen Abmessungen von $93\,\mathrm{mm} \times 63.5\,\mathrm{mm} \times 20\,\mathrm{mm}$, bei dem alle Komponenten wie der Prozessor, Arbeitsspeicher oder die Anschlüsse auf einer Platine zu finden sind. Es ist kein interner Festplattenspeicher vorgesehen. Als Speichermedium dient eine austauschbare SD Karte. Diese Speicherkarte ist bootfähig und enthält das Betriebssystem, sodass der Start des Raspberry Pi von der SD Karte erfolgt. Als Betriebssystem wird ”Raspbian” verwendet. Dabei handelt es sich um eine auf Debian basierende Linux Distribution, die speziell für den Betrieb mit dem Raspberry optimiert wurde. Es sind mehrere Versionen des Raspberry Pi verfügbar. Hier wurde die Version Raspberry Pi 3 Version B verwendet. Dieser enthält einen Quad Core Prozessor mit einer Taktfrequenz von $1.2\,\mathrm{GHz}$. Als Arbeitsspeicher sind $1\,\mathrm{GB}$ RAM vorhanden. Der Raspberry Pi verfügt über viele verschiedene Anschlüsse. So ist z.B. ein Ethernet Anschluss, mehrere USB Anschlüsse sowie frei konfigurierbare GPIO (general purpose input output) Anschlüsse neben weiteren vorhanden. Der Raspberry wird in vielen Anwendungen aufgrund seiner Flexibilität und auch seinen geringen Kosten von ca. $40€$ als Testumgebung eingesetzt.

Software

Die Modellierung und der Reglerentwurf wurde in der Entwicklungsumgebung MATLAB und Simulink durchgeführt. Neben den Standard-Toolboxen sind dazu für die Kommunikation mit dem Raspberry Pi und die Verwendung des prädiktiven Reglers zusätzliche Toolboxen notwendig, die im Folgenden kurz beschrieben werden.

MATLAB Support Package for Raspberry Pi
MATLAB stellt zwei Pakete zur Verfügung, die die Arbeit von MATLAB mit dem Raspberry Pi unterstützten. Dies ist zum einen das ”MATLAB Support Package for Raspberry Pi”. Dieses ermöglicht eine Remote Kommunikation vom Host-PC mit dem Raspberry Pi. Die Verbindung zwischen dem Raspberry und dem Host-PC wird über Ethernet oder WLAN hergestellt. Daraufhin ist es möglich Zugriff auf die Schnittstelle des Raspberry aus MATLAB heraus zu erhalten, welches auf dem Host-PC läuft. Es können somit z.B. bestimmte Pins der GPIO gesetzt oder ausgelesen werden, um Aktoren anzusteuern oder Sensoren auszulesen. Allerdings ist hierbei zu beachten, dass kein eigenständiger Betrieb des Raspberry möglich ist. Es erfolgt keine lokale Ausführung des MATLAB Codes auf dem Raspberry Pi. Der Code läuft auf dem Host-PC, der mit dem Raspberry kommuniziert. Es ist somit immer eine Verbindung zu dem Host-PC

notwendig. Dies ist jedoch für die Anwendung des Reglerdemonstrators nicht erwünscht, da der Regler eigenständig arbeiten soll.

Simulink Support Package for Raspberry Pi
Zur Entwicklung von Algorithmen, die eigenständig auf dem Raspberry Pi ausgeführt werden, kann Simulink in Verbindung mit dem ”Simulink Support Package for Raspberry Pi” verwendet werden. Durch dieses Paket wird die Simulink Standard-Bibliothek um Komponenten erweitert, mit denen auf die Eingabe-Ausgabeschnittstellen des Raspberry Pi zugegriffen werden kann. Damit ist es z.B. möglich aus einem Simulink Modell heraus die GPIO anzusteuern oder über das Netzwerk über UDP zu kommunizieren, wie beispielhaft in Abbildung 6.7-61 dargestellt.

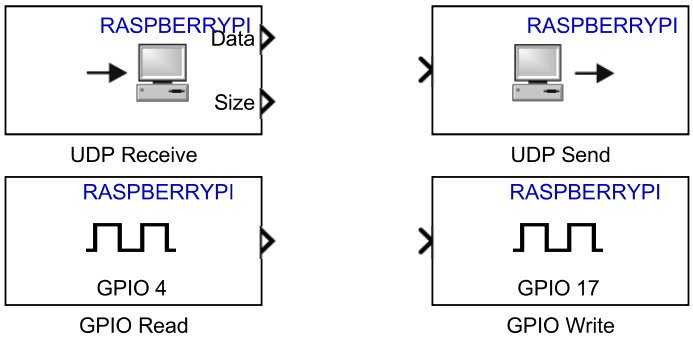


Abbildung 6.7-61: Einige Ein- und Ausgabeblöcke in Simulink für den Raspberry Pi

Die Blöcke werden einfach als Datenquellen oder -senken dem Simulink Modell hinzugefügt. Der Algorithmus, der auf dem Raspberry implementiert werden soll, wird zunächst in Simulink auf dem Host-PC programmiert. MATLAB Code kann über ”MATLAB Functions” in das Simulink Modell eingebunden werden. Dabei ist zu beachten, dass das Modell so aufgebaut wird, dass es die Codegenerierung unterstützt. Dies bedeutet z.B., dass nur Funktionen verwendet werden, die für die Codegenerierung geeignet sind oder dass sich die Datentypen und die Größen der Variablen eindeutig ergeben und sich während der Laufzeit nicht ändern. Ein dynamisches Speichermanagement ist in MATLAB möglich, später im Betrieb auf der Echtzeithardware jedoch nicht mehr, sodass das Modell so vorbereitet werden muss, dass die Anforderungen an den späteren Betrieb erfüllt sind. Die Verbindung zwischen Host-PC und Raspberry erfolgt über Ethernet. Aus Simulink erfolgt eine Codegenerierung und das Modell wird auf den Raspberry Pi geladen und dort ausgeführt. Eine Verbindung mit dem Host-PC ist danach nicht mehr notwendig, da das Modell eigenständig auf dem Raspberry Pi läuft. Somit kann der Raspberry Pi ohne weiteren Computer arbeiten. In der hier betrachteten Anwendung bedeutet das, dass der Regler, der auf dem Pi implementiert ist, direkt an der Anlage eingesetzt werden kann.

MPC Toolbox
Für die Analyse, den Entwurf und die Simulation von prädiktiven Reglern bietet MATLAB die ”MPC Toolbox” an. Mithilfe dieser Toolbox kann ein prädiktiver Regler in MATLAB erstellt und parametriert werden. Ein erstelltes lineares Modell lässt sich direkt einbinden. Der Regler wird durch Angabe von z.B. Nebenbedingungen, Wichtungen und Zeithorizonten parametriert. Daraufhin wird ein MPC Objekt erstellt, das die Definition des gesamten Reglers enthält. Dieses MPC Objekt kann auch Simulink für die Simulation des MPCs genutzt werden. Liegt auch das Modell der Anlage in Simulink vor, lässt sich der Regler einfach im geschlossenen Kreis in der Simulation testen und tunen. Die ”MPC Toolbox” unterstützt den Standard MPC Ansatz, d.h. die Verwendung von linearen Modellen und einer quadratischen Kostenfunktion. Hierfür werden effiziente QP Löser zur Lösung des Optimierungsproblems bereitgestellt. Zudem werden auch weitere Ansätze, wie ein expliziter MPC Entwurf oder lineare Modelle, die sich während der Laufzeit ändern, unterstützt. Aufgrund des zweiten Punktes eignet sich die Toolbox für die Implementierung des MPC Ansatzes mit sukzessiver Linearisierung aus Kapitel 6.4.2. Ein weiterer Vorteil der Toolbox ist, dass sie die Codegenerierung unterstützt. Andere Optimierer in MATLAB sind nicht codegenerierbar, sodass dabei ein größerer Programmieraufwand nötig wäre. Da die MPC Toolbox für die hier geplante Anwendung passend ist, kann der dort implementiere Optimierer direkt für die Lösung des MPC Optimierungsproblems und somit auch für die Echtzeitimplementierung verwendet werden.

6.7.2 Echtzeit Simulator für MTI Systeme

Ein Simulator für MTI Systeme für die Echtzeitanwendung soll entwickelt werden. Das Modell soll auf einem Raspberry Pi verwendet werden, der dann z.B. die Anlage in einem Hardware in the Loop Test darstellen kann. Da die Hardwarekomponenten auf einem festen Takt arbeiten, wird hier ein zeitdiskretes MTI System verwendet

$$\begin{aligned}\mathbf{x}(k+1) &= \langle\, \mathsf{F} \mid \mathsf{M}\left(\mathbf{x}(k), \mathbf{u}(k)\right) \rangle\,,\\ \mathbf{y}(k) &= \langle\, \mathsf{G} \mid \mathsf{M}\left(\mathbf{x}(k), \mathbf{u}(k)\right) \rangle\,.\end{aligned}$$

Um die Anlage auf dem Raspberry Pi zu parametrieren, müssen die Parametertensoren angegeben werden. Dies soll hier im CP Format passieren. Die CP Tensorformate aus der Tensor Toolbox und Tensorlab arbeiten allerdings mit den Matlab Datentypen *Struct* und *Cell*. Da diese Datentypen nicht für die Codegenerierung unter MATLAB unterstützt werden, muss eine alternative Speicherform für die Parametertensoren gefunden werden. Der Parametertensor der Zustandsgleichung bei n Zuständen und m Eingängen ist durch

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

mit $r_{CP,F}$ Rang-1 Komponenten, Faktormatrizen $\mathbf{F}_{u_i}, \mathbf{F}_{x_i} \in \mathbb{R}^{2\times r_{CP,F}}$, $\mathbf{F}_\Phi \in \mathbb{R}^{n\times r_{CP,F}}$ und einem Wichtungsvektor $\boldsymbol{\lambda}_F \in \mathbb{R}^{r_{CP,F}}$ gegeben. Der Parametertensor der Ausgangsgleichung bei p Ausgängen lautet

$$\mathsf{G} = [\mathbf{G}_{u_m}, \ldots, \mathbf{G}_{u_1}, \mathbf{G}_{x_n}, \ldots, \mathbf{G}_{x_1}, \mathbf{G}_\Phi] \cdot \boldsymbol{\lambda}_{\mathbf{G}}$$

mit $r_{CP,G}$ Rang-1 Komponenten, Faktormatrizen $\mathbf{G}_{u_i}, \mathbf{G}_{x_i} \in \mathbb{R}^{2\times r_{CP,G}}$, $\mathbf{G}_\Phi \in \mathbb{R}^{n\times r_{CP,G}}$ und einem Wichtungsvektor $\boldsymbol{\lambda}_G \in \mathbb{R}^{r_{CP,G}}$. Für die Codegenerierung geeignet sind Arrays vom Datentyp *double*. Für das Speicherformat wird ausgenutzt, dass die Faktormatrizen $\mathbf{F}_{u_i}$ und $\mathbf{F}_{x_i}$ bzw. $\mathbf{G}_{u_i}$ und $\mathbf{G}_{x_i}$ alle die gleiche Dimension $2\times r_{CP,F}$ bzw. $2\times r_{CP,G}$ haben. Somit können diese Faktormatrizen jeweils verbunden und in dreidimensionalen Arrays $\mathsf{F}_{xu} \in \mathbb{R}^{2\times r_{CP,F}\times n+m}$ und $\mathsf{G}_{xu} \in \mathbb{R}^{2\times r_{CP,G}\times n+m}$ gespeichert werden. Die einzelnen Faktormatrizen werden über die dritte Dimension ausgewählt. Für den Parametertensor der Zustandsgleichung bedeutet das

$$\begin{aligned}\mathsf{F}_{xu}(:,:,1) &= \mathbf{F}_{u_m},\\ &\vdots\\ \mathsf{F}_{xu}(:,:,m) &= \mathbf{F}_{u_1},\\ \mathsf{F}_{xu}(:,:,m+1) &= \mathbf{F}_{x_n},\\ &\vdots\\ \mathsf{F}_{xu}(:,:,n+m) &= \mathbf{F}_{x_1}.\end{aligned}$$

Neben diesem Array müssen zusätzlich die Faktormatrix $\mathbf{F}_\Phi$ und der Wichtungsvektor $\boldsymbol{\lambda}_F$ gespeichert werden. Diese drei Arrays sind alle vom Datentyp *double* und somit für die Codegenerierung geeignet. Mit diesen Arrays ist der Tensor komplett beschrieben. Für die Beschreibung des Parametertensors G der Ausgangsgleichung müssen somit die Faktormatrix $\mathbf{G}_\Phi$, der Wichtungsdvektor $\boldsymbol{\lambda}_G$ und das dreidimensionale Array mit den Scheiben

$$\begin{aligned}\mathsf{G}_{xu}(:,:,1) &= \mathbf{G}_{u_m},\\ &\vdots\\ \mathsf{G}_{xu}(:,:,m) &= \mathbf{G}_{u_1},\\ \mathsf{G}_{xu}(:,:,m+1) &= \mathbf{G}_{x_n},\\ &\vdots\\ \mathsf{G}_{xu}(:,:,n+m) &= \mathbf{G}_{x_1}.\end{aligned}$$

vorliegen. Für die Simulation des zeitdiskreten MTI Modell müssen die Modellgleichungen iterativ angewendet werden. Dies bedeutet, dass das kontrahierte Produkt von Parameter- und Monomtensor jeweils gelöst werden muss, um $\mathbf{x}(k+1)$ und $\mathbf{y}(k)$ zu bestimmen. In (6.1-41) wurde gezeigt, wie dies auf Basis der Parametertensoren möglich ist. Um das eingeführte Speicherformat zu berücksichtigen, ist das Verfahren anzupassen, sodass in jedem Abtastschritt folgende Berechnungen nötig sind

$$\begin{aligned}
\mathbf{x}(k+1) =& \mathbf{F}_\Phi \left(\boldsymbol{\lambda}_F \circledast \left(\mathsf{F}_{xu}(:,:,1)^T \begin{pmatrix} 1 \\ u_m \end{pmatrix} \right) \circledast \cdots \circledast \left(\mathsf{F}_{xu}(:,:,m)^T \begin{pmatrix} 1 \\ u_1 \end{pmatrix} \right) \cdots \right. \\
& \left. \circledast \left(\mathsf{F}_{xu}(:,:,m+1)^T \begin{pmatrix} 1 \\ x_n \end{pmatrix} \right) \circledast \cdots \circledast \left(\mathsf{F}_{xu}(:,:,n+m)^T \begin{pmatrix} 1 \\ x_1 \end{pmatrix} \right) \right), \\
\mathbf{y}(k) =& \mathbf{G}_\Phi \left(\boldsymbol{\lambda}_G \circledast \left(\mathsf{G}_{xu}(:,:,1)^T \begin{pmatrix} 1 \\ u_m \end{pmatrix} \right) \circledast \cdots \circledast \left(\mathsf{G}_{xu}(:,:,m)^T \begin{pmatrix} 1 \\ u_1 \end{pmatrix} \right) \cdots \right. \\
& \left. \circledast \left(\mathsf{G}_{xu}(:,:,m+1)^T \begin{pmatrix} 1 \\ x_n \end{pmatrix} \right) \circledast \cdots \circledast \left(\mathsf{G}_{xu}(:,:,n+m)^T \begin{pmatrix} 1 \\ x_1 \end{pmatrix} \right) \right).
\end{aligned}$$

Da hier nur Standard Operationen wie die Addition oder die Multiplikation verwendet wurden und die Parametertensoren in Form von Arrays vom Datentyp *double* vorliegen, ist es nun möglich, Code aus diesem Modell zu generieren und es in Echtzeit auf dem Raspberry Pi auszuführen. Die Eingänge $\mathbf{u}$ können z.B. über die GPIO Pins oder die UDP Schnittstelle eingelesen werden. Die Ausgabe von $\mathbf{y}(k)$ kann ebenso über diese Schnittstellen erfolgen. Somit ist z.B. die Kommunikation mit einem Reglerprototypen möglich.

6.7.3 Echtzeitumsetzung adaptiver MPC

Der Ansatz des AMPC-SL aus dem Kapitel 6.4.2 für das Beispiel der Regelung eines Heizkreises aus dem Unterabschnitt 6.4.3 soll auf einer Echtzeithardware, dem Raspberry Pi, implementiert werden, um den Regler auch am realen Gebäude anwenden zu können. Die Implementierung des Regleralgorithmus soll aus Simulink heraus über eine Codegenerierung über die in Kapitel 6.7.1 beschriebenen Pakete erfolgen. Somit müssen alle notwendigen Operationen, die für den AMPC-SL nötig sind, so implementiert werden, dass sie codegenerierbar sind. Dies bedeutet, dass das MTI Modell der Anlage im CP Format in dem Speicherformat, das für die Codegenerierung geeignet ist, vorliegen muss. Während der Linearisierung des MTI Modells ist ein Tensor Matrix Produkt zu berechnen. Die Tensoroperation kann leicht angepasst werden, sodass sie alle Anforderungen erfüllt. Die weiteren Operationen, wie die Skalierung, brauchen keine Überarbeitung.

Ein Kernelement des prädiktiven Reglers ist der Optimierer. Hier muss ein geeigneter Optimierer gefunden werden, der die Codegenerierung unterstützt. Mit den Standard MATLAB Routinen wie `quadprog` ist dies nicht möglich. Die MPC Toolbox von MATLAB , die in Kapitel 6.7.1 kurz vorgestellt wurde, erlaubt die Codegenerierung. Mit der MPC Toolbox ist es möglich Standard MPC Optimierungsprobleme mit linearem Modell und quadratischer Kostenfunktion und linearen Nebenbedingungen zu formulieren und diese als quadratisches Problem zu lösen, [5]. Zudem ist es möglich Referenzen und Störgrößenvorhersagen mit einzubinden. Zusätzlich können zeitveränderliche Grenzen für die Signale und Wichtungen verwendet werden, welches hier auch notwendig ist, um zwischen Tag- und Nachtbetrieb zu unterscheiden. Die Besonderheit des vorgestellten AMPC-SL Algorithmus für MTI Systeme ist die Anpassung des linearen Modells im prädiktiven Regler. Hierfür bietet die MPC Toolbox den ”Adaptiven MPC”, der in Abbildung 6.7-62 dargestellt ist.

Dieser hat einen zusätzlichen Eingang, über den es möglich ist ein lineares Modell und einen Arbeitspunkt für jeden Zeitschritt neu vorzugeben. Somit kann die Linearisierung des MTI Modells vorgelagert in einer ”Matlab Function” in Simulink erfolgen. Das Ergebnis wird einfach an den ”Adaptiven MPC”-Block aus der Toolbox übergeben, der dann die Lösung des Optimierungsproblems übernimmt und das optimale Stellsignal ausgibt. Zur Lösung des quadratischen Optimierungsproblems wird der KWIK Algorithmus verwendet, [45]. Damit lässt sich der Regler in Simulink auf dem Host-PC entwerfen und dann über eine Codegenerierung mit dem Simulink Support Package auf dem Raspberry Pi implementieren.

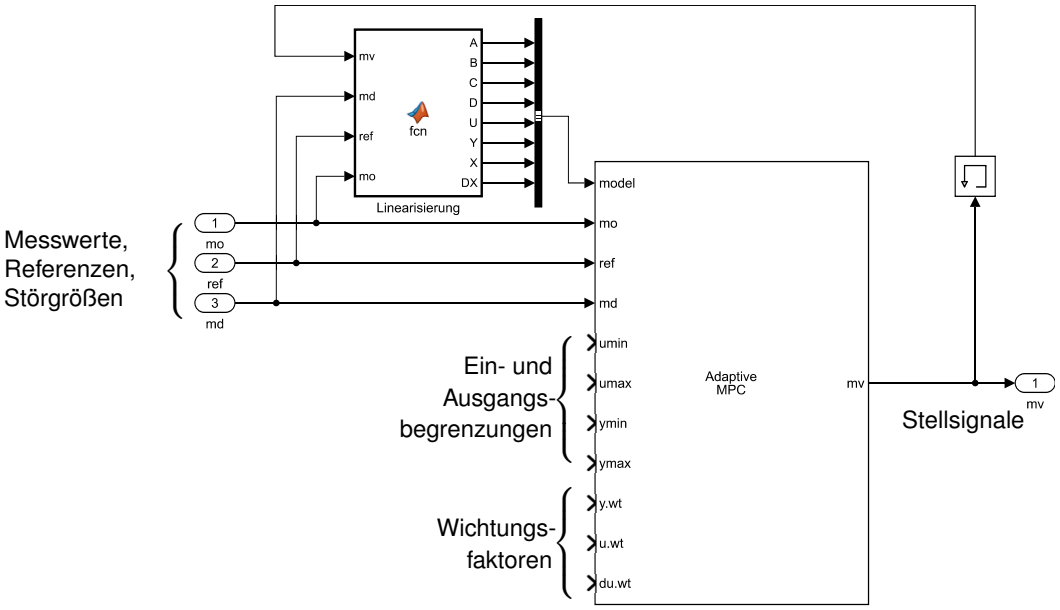


Abbildung 6.7-62: Simulink Umsetzung des AMPC-SL für MTI Systeme

6.7.4 Aufbau einer Hardware in the Loop Umgebung

Im vorangegangenen Kapitel wurde beschrieben, wie sich ein Regler auf dem Raspberry Pi implementieren lässt. Ein üblicher Entwicklungsablauf ist, dass der Regler zunächst auf dem Host-PC entworfen und in Simulationen im geschlossenen Kreis mit einem Modell der Anlage getestet wird. Daraufhin erfolgt dann die Hardware Implementierung. Bevor diese Implementierung an der realen Anlage eingesetzt wird, ist zu testen, ob sich auch die Implementierung, wie zuvor auf dem Host-PC verhält. Dies ist zum einen über die die Vorgabe gewisser Testsignale möglich. Um jedoch das Verhalten im geschlossenen Kreis zu überprüfen, muss der Regler zusammen mit der Anlage oder einem Modell der Anlage betrieben werden. In dem Kapitel 6.7.2 wurde vorgestellt, wie sich ein MTI Modell auf einem Raspberry Pi simulieren lässt. Dies soll hier dazu dienen, um den Regler in eine Hardware in the Loop Umgebung einzubinden. Hierbei wird auf einem Raspberry Pi der zu testende Regler implementiert. Ein zweiter Raspberry Pi stellt die Anlage dar, indem ein Modell der Anlage auf ihm implementiert ist. Die Kommunikation zwischen den Komponenten erfolgt in dem hier betrachteten Beispiel über des Netzwerkprotokoll UDP (User Datagram Protocol). Die Raspberry Pis haben unterschiedliche IP Adressen und können dann auf verschiedenen Ports über UDP miteinander kommunizieren. Die Kommunikation aus dem codegenerierten Simulink Modell heraus ist über das Simulink Support Package für den Raspberry Pi möglich, indem die in Abbildung 6.7-61 gezeigten Kommunikationsblöcke verwendet werden. Der Aufbau ist schematisch in der Abbildung 6.7-63 dargestellt.

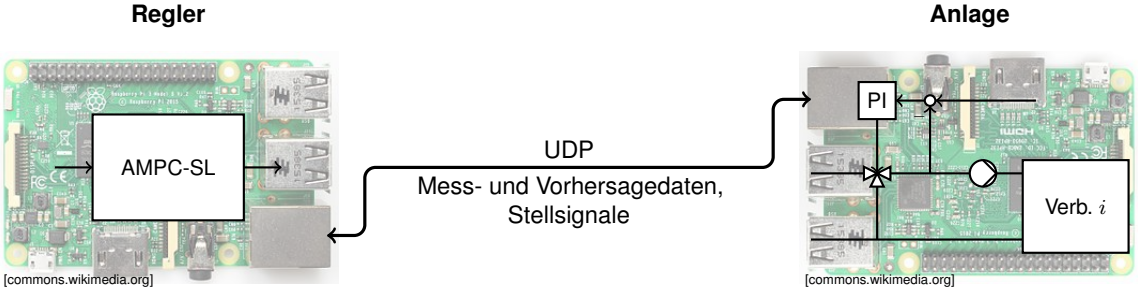


Abbildung 6.7-63: Hardware in the loop Umgebung

Somit ist eine Testumgebung für den Regler im geschlossenen Kreis geschaffen. Ein wichtiger Test hierbei ist die Echtzeitfähigkeit auf der jeweiligen Hardwarekomponente. Bei prädiktiven Reglern ist es wichtig, dass das Ergebnis der Optimierung immer innerhalb eines Abtastschrittes vorliegt, um das Stellsignal für den nächsten Zeitschritt bereitzustellen. Erst in solchen HiL Tests mit typischen Szenarien kann ermittelt werden, ob dies während des Betriebes auch für die jeweilige Implementierung erfüllt ist. Die HiL Tests mit dem in Kapitel 6.4.3 vorgestellten AMPC-SL Algorithmus für einen Heizkreis eines Bürogebäudes, wurden erfolgreich durchgeführt. Der Regler arbeitet auf einem Takt von einer Minute. Bei einem Prädiktionshorizont von 3 Stunden und einem Stellhorizont von 2 Stunden konnte der Takt von dem Raspberry in allen Tests eingehalten werden, d.h. der AMPC-SL Algorithmus war immer vor Ablauf der Abtastzeit fertig mit der Berechnung des nächsten Stellsignals. Auch die Ergebnisse waren wie zuvor erwartet, d.h. der implementierte Regler arbeitet, wie zuvor auf dem Host-PC entworfen. Nach diesen Tests wird der Regler nun an dem realen Gebäude betrieben.

6.7.5 Implementierung am Gebäude

Der AMPC-SL Regler wurde erfolgreich in der HiL Umgebung getestet, wie in dem Kapitel 6.7.4 beschrieben. Daher wird er nun am realen Gebäude angewendet. Der Einsatz erfolgt an einem Heizkreis eines Bürogebäudes, wie in dem Kapitel 6.4.3 vorgestellt. Die Implementierung ist auf einem Raspberry Pi erfolgt. Nun sollen die Messdaten und Stellsignale nicht mehr von einer Implementierung eines Modells kommen, sondern die Kommunikation mit der realen Anlage erfolgen. Dazu werden die Messsignale aus der realen Anlage verwendet. In jedem Heizkreis sind Wärmemengenzähler installiert. Die Messinformation über die Raumtemperatur in dem Heizkreis ist über einen Referenzraum gegeben. Die Vorhersagen über die zukünftige Außentemperatur und die Solarstrahlung erfolgt über eine Wetterstation. Die Kommunikationsschnittstelle zwischen dem Regler auf dem Raspberry Pi und der Anlage, d.h. der DDC wurde von Kieback&Peter in Absprache mit der HAW Hamburg entwickelt. Der Regler wurde in Simulink implementiert. Die Kommunikation mit der Peripherie mit dem Einlesen von Messwerten und dem Schreiben der Stellsignale erfolgt aus dem Simulink Modell heraus über das Netzwerkprotokoll UDP. Es ist jedoch keine direkte Kommunikation über UDP mit der DDC möglich. Daher sendet das Modell die UDP Pakete zunächst einmal an die eigene Hardwarekomponente, d.h. den eigenen Raspberry Pi, dem sog. ”localhost”. Auf dem Raspberry Pi läuft zudem ein Python Script, welches die UDP Pakete nach BACnet umsetzt. BACnet ist ein Standard Protokoll in der Gebäudetechnik. Somit können die Daten über das Netzwerk mit dem BACnet Protokoll an die DDC kommuniziert werden, die als Schnittstelle zur Anlage dient. Die Messdaten werden genau auf dem umgekehrten Weg verarbeitet. Die DDC sendet die Daten über BACnet an den Raspberry Pi. Die BACnet Pakete werden ausgelesen und nach UDP übersetzt, sodass der Regler sie einlesen kann. Damit ist die Kommunikation zwischen Regler und Anlage erfolgreich auf einer Komponente hergestellt. Ein Schema ist in der Abbildung 6.7-64 gezeigt.

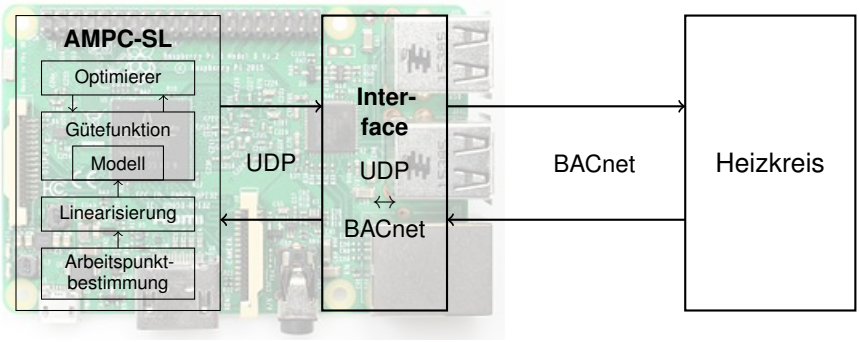


Abbildung 6.7-64: Kommunikationsschema

Durch die Implementierung der Kommunikationsschnittelle sowie des Regleralgorithmus auf einer Komponente, können Synchronisationsprobleme zwischen verschiedenen Hardwarekomponenten vermieden werden. Der Regler gibt hier den Takt vor.

Der Testzeitraum an der Anlage war mit kleinen Unterbrechungen vom 2. November 2017 bis 15. De-

zember 2017. Die Durchführung der Tests erfolgte durch Kieback&Peter. Die Reglerupdates wurden in enger Abstimmung mit der HAW Hamburg eingespielt und entwickelt. Hierbei wurde der Regler an der Anlage betrieben und der Betrieb durch mehrere Updates optimiert. Der Verlauf der Test soll hier anhand einiger Zwischenergebnisse erläutert werden. Die Hardware lief über den gesamten Testzeitraum sehr zuverlässig. Der Regleralgorithmus lief sehr stabil auf der Raspberry Pi Hardware. Die Abtastschrittweite $T_{sample} = 60\,\mathrm{s}$ von einer Minute wurde konstant eingehalten. Auch gegenüber kurzzeitigen Netzwerkausfällen oder Fehlern in der Messdatenerfassung zeigte sich der gewählte Ansatz robust. Um solche einfachen Ausfälle oder Fehler zu erkennen, wurde im Regler für jedes Messsignal ein Intervall hinterlegt, indem sich das entsprechende Signal befinden soll, um einen nominalen Arbeitsbereich zu charakterisieren. Wird ein nicht plausibler Messwert empfangen, d.h. befindet er sich außerhalb des Intervalls, wird dies erkannt. Statt des Ergebnis des MPC wird ein Defaultwert für die Sollvorlauftemperatur ausgegeben. Der Defaultwert ist hier auf $55\,^{\circ}\mathrm{C}$ festgelegt, was einem Erfahrungswert für das Gebäude entspricht, sodass das Gebäude auch bei niedrigen Außentemperaturen nicht komplett auskühlt. Der MPC würde bei unplausiblen Messwerten kein brauchbares Ergebnis liefern, da er so nicht den richtigen aktuellen Anlagenzustand für die Initialisierung des Modells bestimmen kann. Nachdem jedoch wieder plausible Daten vorliegen, erfolgt ein automatisches Zurückschalten auf den AMPC-SL Algorithmus, dessen berechnete Stellsignale dann wieder an die Anlage gegeben werden. Weitere Sicherheitsabfragen sind in diesem Reglerdemostrator nicht enthalten. Alle üblichen in der Anlage vorhandenen Sicherheitsmechanismen sind nachgelagert und somit auch beim Betrieb des Reglerdemonstrators aktiv. In der Simulation wurde der Prädiktionshorizont des AMPC-SL auf drei Stunden und der Stellhorizont auf zwei Stunden festgelegt. In der Simulation wurden die übrigen Gewichtungsfaktoren des prädiktiven Reglers wie in Tabelle 6.7-6 gezeigt eingestellt.

Tabelle 6.7-6: Reglerparameter am Beginn des Betriebs

	Tag	Nacht
Referenzfolge q	100	0
Änderung der Vorlauftemperatur r	50	20
Abweichung von min. Vorlauftemperatur s	2	2

Exemplarisch für die Ergebnisse des ersten Testlaufs in der Einstellungsphase sind die Messdaten für die Gebäudetemperatur sowie die Vor- und Rücklauftemperatur in der Abbildung 6.7-65 dargestellt.

Bereits in diesem ersten Test zeichnet sich ein typisches Verhalten des Reglers ab. Nachts wird der Heizkreis mit der minimalen Vorlauftemperatur betrieben und der Heizkreis kühlt aus. Die Raumtemperatur ist allerdings noch weit von der unteren erlaubten Grenze von $18\,^{\circ}\mathrm{C}$ entfernt, da der Heizkreis nur langsam auskühlt. Somit ist in der Nacht kein weiteres Heizen nötig und die Vorlaufsolltemperatur wird auf ihre untere Grenze gesetzt, d.h. $40\,^{\circ}\mathrm{C}$ in dieser Implementierung. Morgens wird der Heizkreis dann mit der maximal möglichen Vorlauftemperatur betrieben, um die Gebäudetemperatur wieder auf die Referenz am Tag von $22\,^{\circ}\mathrm{C}$ zu bringen. Hierbei zeigt sich ein prädiktives Verhalten, da die Referenz dem Regler für den gesamten Prädiktionshorizont bekannt ist und er somit schon drei Stunden vor dem eigentlichen Anstieg der Referenz Kenntnis von dem Sprung hat und dies berücksichtigt. Für die Grenzen der Raumtemperatur ist eine solche Vorhersage für den gesamten Vorhersagehorizont in der verwendeten MPC Implementierung von Simulink nicht vorgesehen. Daher wurde die Begrenzung eine Stunde gegenüber der Referenz verschoben. Das prädiktive Verhalten der Reglers beim Aufheizen bietet somit noch Optimierungspotenzial. Es ist in der Abbildung 6.7-65 bei dem Vergleich der geforderten Sollvorlauftemperatur und der tatsächlich eingestellten Vorlauftemperatur des Heizkreises zudem zu erkennen, dass die geforderte Vorlauftemperatur nicht erreicht wird. Der Grund dafür ist eine manuelle Begrenzung der maximalen Kesselvorlauftemperatur, sodass die Anforderung des Reglers teilweise nicht bereitgestellt werden kann. Nach dem Aufheizen weist die Raumtemperatur noch eine gewisse Abweichung zur Referenz auf. Der Regler hält die Gebäudetemperatur aber in dem gewünschten Band. Hier zeigt sich somit das typische Tag-Nacht Verhalten. Auch das Wochenende wird vom Regler erkannt. Es erfolgt ein Betrieb wie in der Nacht, d.h. es wird nur darauf geachtet, dass das Gebäude nicht zu stark auskühlt und der Heizkreis ansonsten mit einer möglichst geringen Vorlauftemperatur versorgt wird. Mit den so gewonnen Daten wurde ein Update für den Regler erstellt.

Um die Vorlauftemperatur für die Phasen, in denen nicht geheizt werden muss, d.h. gerade Nachts

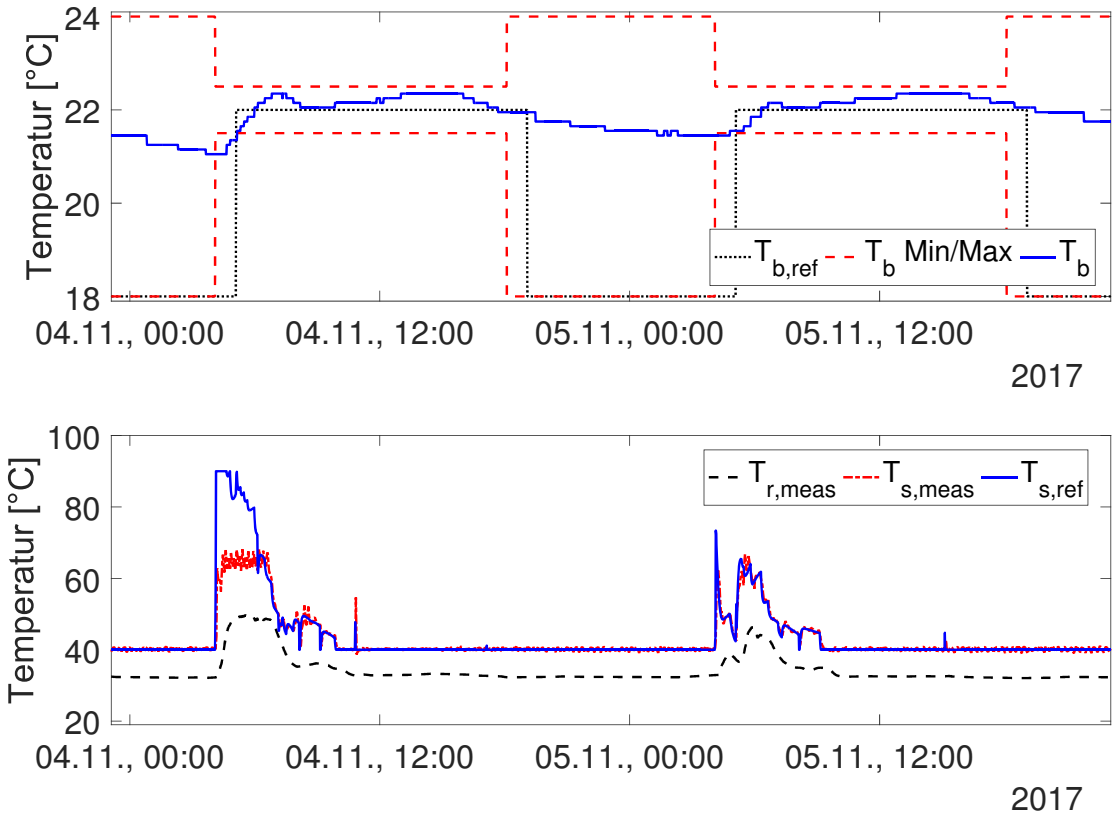


Abbildung 6.7-65: Messdaten des AMPC-SL Tests vom 04.11., 00:00 Uhr bis zum 06.11., 00:00 Uhr

weiter absenken zu können wurde die minimale Sollvorlauftemperatur auf $30\,^{\circ}\mathrm{C}$ gesetzt. Um die minimale Grenze der Rücklauftemperatur nicht zu unterschreiten, wurde sie auf $15\,^{\circ}\mathrm{C}$ festgelegt. Die maximal erlaubte Sollvorlauftemperatur wurde mit $80\,^{\circ}\mathrm{C}$ näher an die manuell eingestellte Grenze der maximalen Kesseltemperatur heran gelegt. Die Erfahrung aus dem ersten Testlauf hat gezeigt, dass die Nutzer bei einer Referenztemperatur von $22\,^{\circ}\mathrm{C}$, obwohl sie eingehalten wurde, ein Kälteempfinden hatten. Daher wurde die Referenz auf $23\,^{\circ}\mathrm{C}$ erhöht, um den Komfort der Nutzer zu erfüllen. Die Parameter des Modells wurden unter Verwendung der neu gewonnen Daten leicht angepasst. Die Modellstruktur musste nicht verändert werden. Um die zum Teil sehr starken Reaktionen des Reglers auf die Abweichung der Raumtemperatur von der Referenz etwas zu reduzieren, wurden auch die Reglerparameter upgedatet, indem der Faktor r, mit dem die Änderungen des Stellsignals gewichtet werden, erhöht wurde, wie in der Tabelle 6.7-7 gezeigt.

Tabelle 6.7-7: Reglerparameter nach dem ersten Update

	Tag	Nacht
Referenzfolge q	100	0
Änderung der Vorlauftemperatur r	200	100
Abweichung von min. Vorlauftemperatur s	2	2

Mit diesen Einstellungen wurde ein Update für den Regler erstellt. Die Messergebnisse mit diesen Reglereinstellungen sind für einen Tag in Abbildung 6.7-66 gezeigt.

Auch bei dieser stärkeren Absenkung und einer höheren Referenz wird der Raum tagsüber in dem gewünschten Band für die Raumtemperatur betrieben, sobald die Raumtemperatur einmal das Band erreicht hat. Dies geschieht jedoch später, als es eigentlich gewünscht ist. Zudem ist ein Schwingen in der Sollvorlauftemperatur zu erkennen, das sich auch in abgeschwächter Form in der Raumtemperatur widerspiegelt. Dies ist darauf zurückzuführen, dass das Modell das Verhalten der realen Anlage nicht

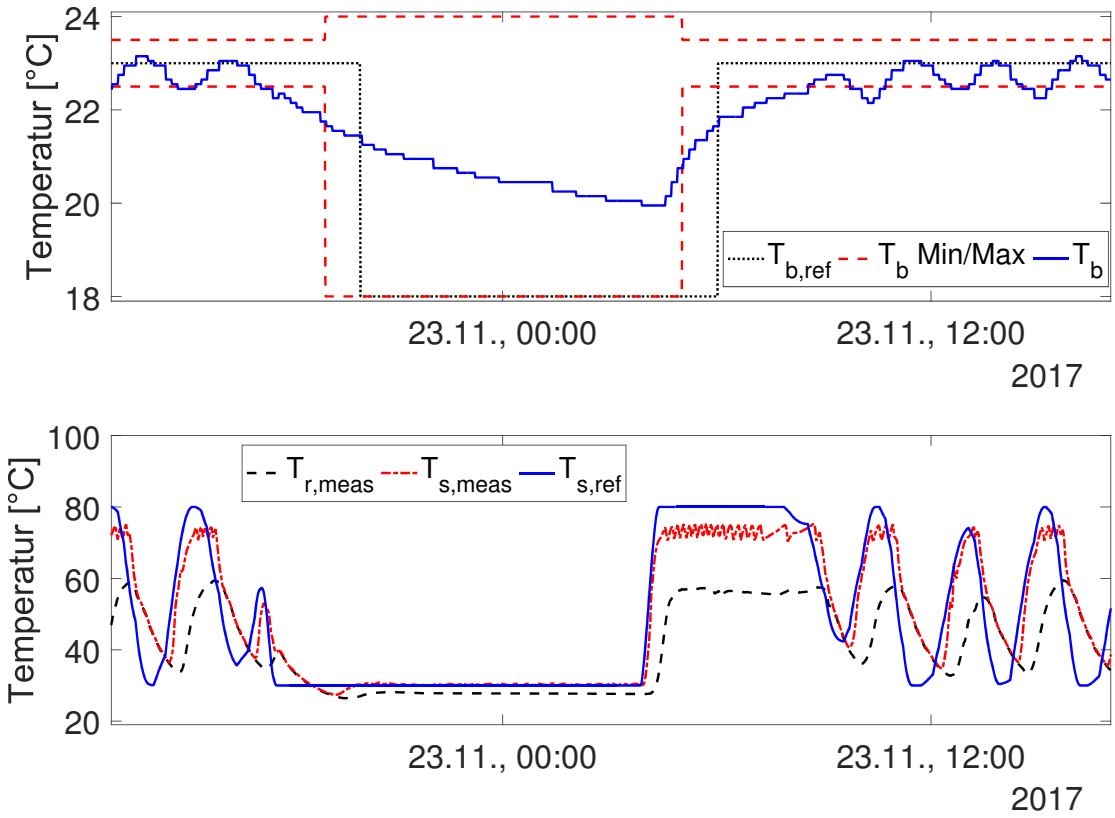


Abbildung 6.7-66: Messdaten des AMPC-SL Tests vom 22.11., 13:00 Uhr bis zum 23.11., 17:00 Uhr

exakt abbildet und der Regler noch sehr stark auf Abweichungen reagiert. Gewünscht ist ein weniger stark schwingendes Stellsignal. Somit wird der Faktor r für die Gewichtung der Stellsignaländerungen weiter erhöht, um das Stellsignal zu beruhigen, siehe Tabelle 6.7-8. Messdaten mit eingearbeiteten Änderungen im AMPC-SL sind in der Abbildung 6.7-67 zu sehen.

Tabelle 6.7-8: Reglerparameter nach dem zweiten Update

	Tag	Nacht
Referenzfolge q	100	0
Änderung der Vorlauftemperatur r	600	100
Abweichung von min. Vorlauftemperatur s	2	2

Die Abbildung zeigt, dass die Schwingungen im Stellsignal zwar noch vorhanden sind, sie jedoch wie erwartet reduziert werden konnten. Das prädiktive Verhalten des Reglers zu Beginn und am Ende des Tages ist noch nicht zufriedenstellend, da die Begrenzungen der Gebäudetemperatur zu diesen Zeiten verletzt werden. Es kommt zu einem zu späten Aufheizen am Morgen und einem zu frühen Abkühlen am Abend. Der Grund dafür ist das typische Verhalten bei der Folge einer Referenz, die vorhersagbar ist, so wie es hier der Fall ist. Dies ist beispielhaft in Abbildung 6.4-36 gezeigt. Aufgrund der langsamen Gebäudedynamik kann die Temperatur nicht sprunghaft dem Anstieg der Referenz folgen. Nach der Kostenfunktion 6.4-158 ist es daher optimal die Referenz bereits vor dem Nutzzeitende zu verlassen. Dies zeigt sich auch in den bisherigen Messergebnissen. Das gewünschte Verhalten hier ist jedoch, dass die Raumtemperatur bereits zum Nutzzeitbeginn auf dem Referenzwert ist. Um dies zu erreichen, soll im Folgenden eine rampenförmige Referenz verwendet werden. Die Steigung der Rampe wird an die Dynamik des Raumes angepasst, sodass die Raumtemperatur dieser Referenz auch folgen kann. Zum Nutzzeitbeginn hat die Rampe dann den Tageswert der Referenz erreicht. Somit ist für die Zeit des Anstiegs der Rampe auch während der Nacht der Wichtungsfaktor für die

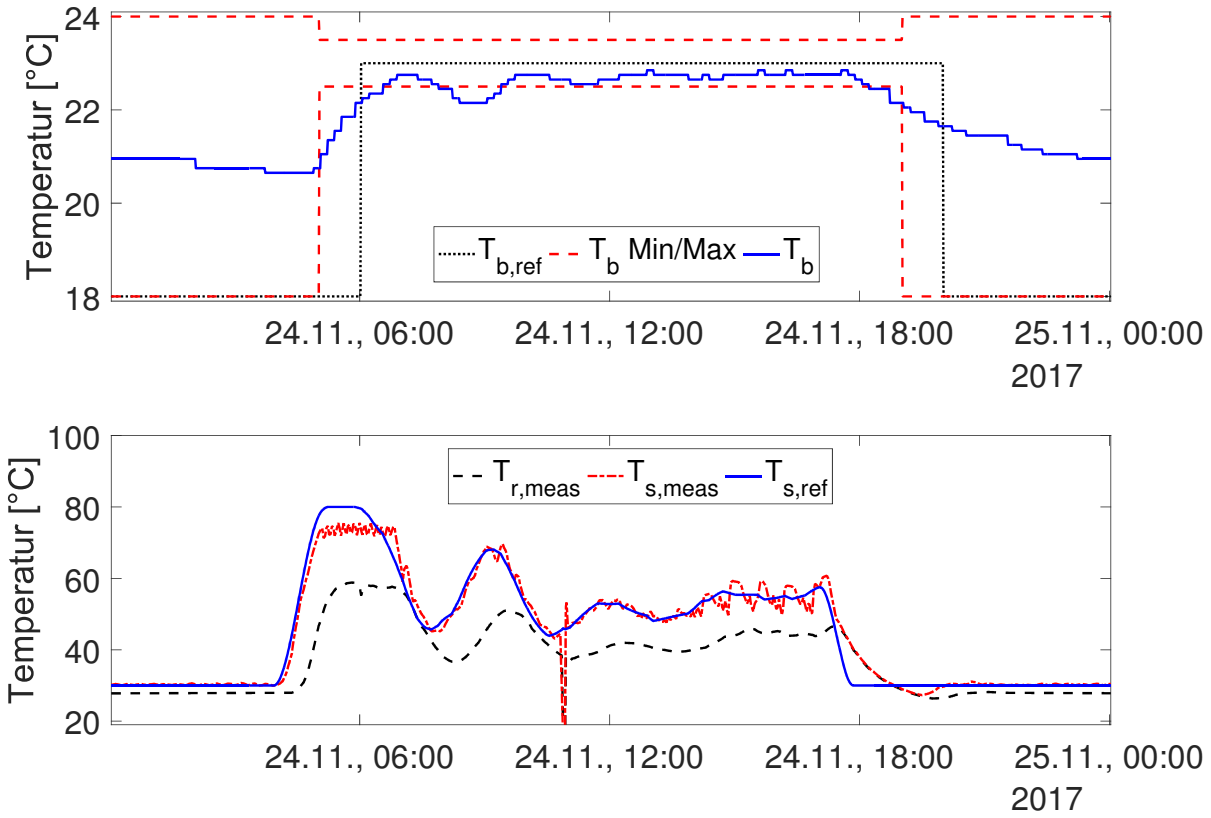


Abbildung 6.7-67: Messdaten des AMPC-SL Tests vom 24.11., 00:00 Uhr bis zum 25.11., 00:00 Uhr

Referenzfolge ungleich Null $q \neq 0$ zu wählen. Die Steigung der Referenz ist abhängig von der Dynamik des Heizkreises. Daher wurden die bisherigen Aufheizverhalten am Morgen während der Tests betrachtet. Daraus ergab sich eine relativ langsames Aufheizverhalten, das hier nun linear approximiert wird, sodass die Raumtemperatur der Referenz folgen kann. Daraus ergab sich eine Steigung von $0.5\,{}^{K}/_{\mathrm{h}}$. Die verschiedenen gemessenen Aufheizkurven durch durchgezogene Linien sowie die verwendete Steigung der Rampe durch die gestrichelte Gerade sind in der Abbildung 6.7-68 dargestellt.

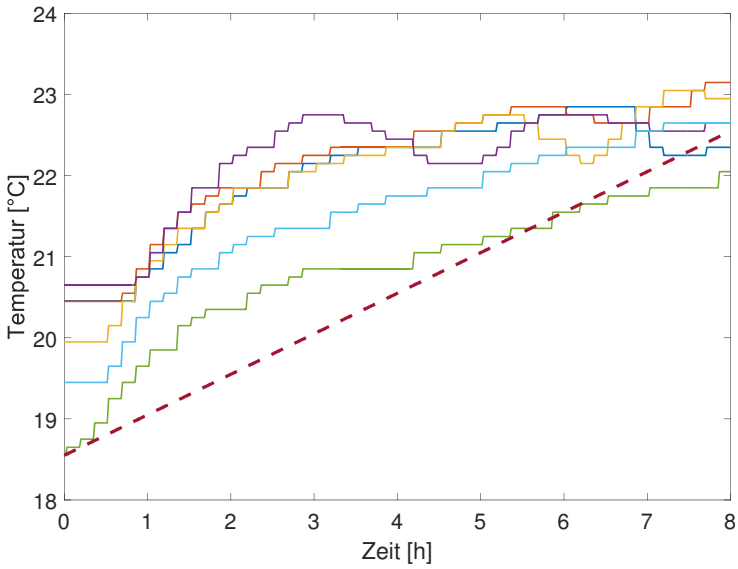


Abbildung 6.7-68: Bestimmung der Steigung der Referenzrampe, Messwerte: durchgezogen, Rampensteigung: gestrichelt

Dies bedeutet, dass eine Aufheizung des Raumes um $5\,\mathrm{K}$, was der Differenz zwischen minimaler und maximaler Referenz der Raumtemperatur entspricht, $10\,\mathrm{h}$ benötigen würde. Die Rampe der Referenz erstreckt sich somit über die ganze Nacht vom der Ende der Nutzzeit um $20:00$ Uhr bis zum Beginn der Nutzzeit am Folgetag um $6:00$ Uhr. Am Wochenende wird die Raumtemperaturreferenz weiterhin auf den minimalen Wert gesetzt. Die untere Grenze für die Raumtemperatur wird parallel um $0.5\,\mathrm{K}$ unterhalb der Referenz geführt. Die oberere Grenze wird konstant gehalten. Der Verlauf der Referenz und der oberen und unteren Grenzen für die Raumtemperatur sind für eine Woche in der Abbildung 6.7-69 gezeigt.

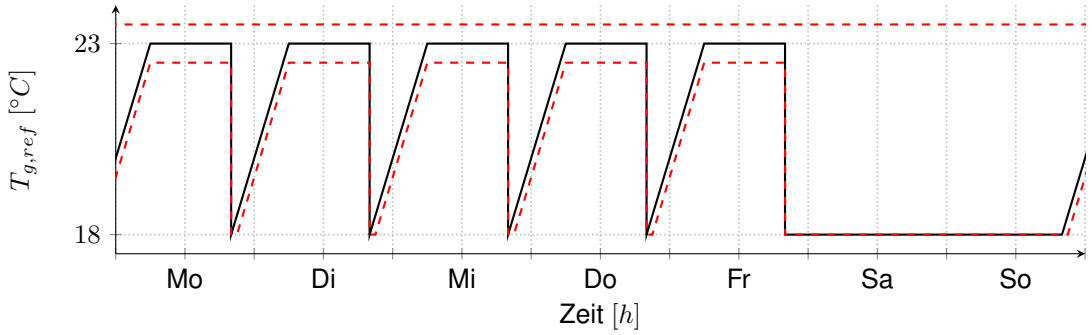


Abbildung 6.7-69: Verlauf der Referenz und der unteren und oberen Grenze der Raumtemperatur für eine Woche

Am Ende der Nutzzeit kühlt der Raum zu früh aus. Das Abkühlen des Raumes beginnt bereits drei Stunden bevor die Referenz abfällt. Diese drei Stunden entsprechen genau dem Vorhersagehorizont. Aufgrund der Vorhersage der Referenz für drei Stunden, bekommt der Regler die Information, dass die Referenztemperatur in drei Stunden bei $18\,°\mathrm{C}$ liegt und senkt daher die Vorlauftemperatur ab, was zu einem Auskühlen führt. Dies ist somit ein Verhalten, dass über den MPC erklärbar, jedoch von praktischer Seite nicht erwünscht ist, da die Raumtemperatur bis zur Ende der Nutzzeit auf dem Referenzwert gehalten werden soll. Dies ist jedoch in der verwendeten MPC Formulierung nicht möglich, wenn die Referenzvorhersage wie bisher angegeben wird. Die Vorhersage der Referenz wird so geändert, dass sie während der Nutzzeit konstant auf $23\,°\mathrm{C}$ gehalten wird. Somit erhält der Regler die Information, dass er diese Referenztemperatur auch über die Nutzzeit heraus halten soll. Nach Ende der Nutzzeit wird die korrekte Referenz anhand der Rampe an den AMPC-SL übergeben. Abbildung 6.7-70 zeigt die jeweilige Vorhersage der Referenz an zwei verschiedenen Zeiten des Tages, zum einen am Ende der Nutzzeit und zum anderen in der Nacht, um das zuvor beschriebene Verhalten der Vorhersage zu erläutern.

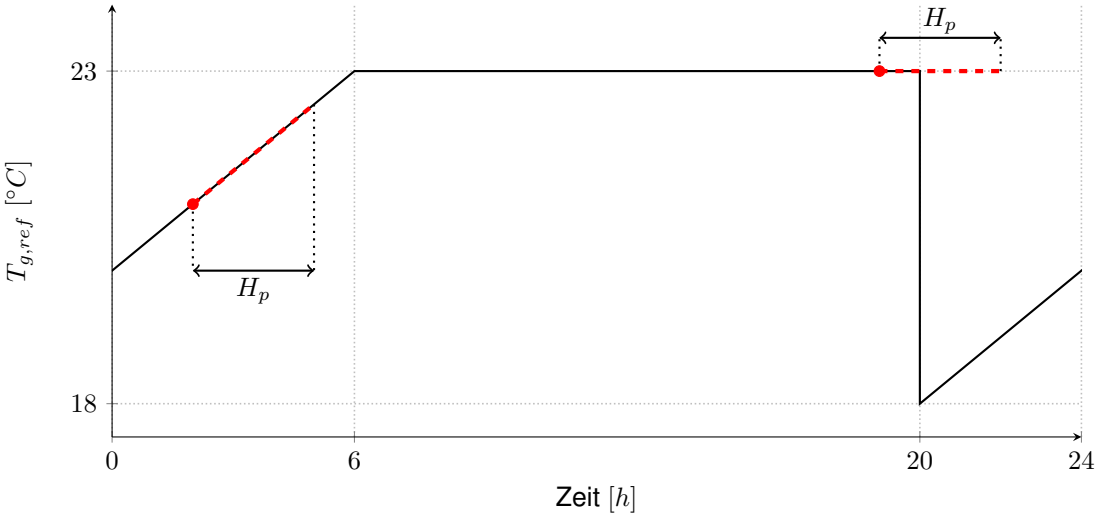


Abbildung 6.7-70: Referenz der Raumtemperatur und dessen Vorhersage an zwei ausgewählten Punkten

Der Fokus bei der MPC Optimierung wurde wieder etwas mehr auf die Referenzfolge gelegt, sodass sich die Reglerparameter in Tabelle 6.7-9 ergeben.

Tabelle 6.7-9: Reglerparameter nach dem dritten Update

	Tag	Nacht
Referenzfolge q	100	100
Änderung der Vorlauftemperatur r	400	400
Abweichung von min. Vorlauftemperatur s	2	2

Messergebnisse eines Tages mit dieser geänderten Referenz und Parametern sind in Abbildung 6.7-71 gezeigt. Es ist deutlich zu erkennen, dass nun die Raumtemperatur schon zu Beginn der Nutzzeit den gewünschten Referenzwert erreicht hat. Die Aufheizphase mit maximaler Vorlauftemperatur wurde somit nun weiter in die Nacht verschoben und die Raumtemperatur folgt während des Aufheizens der Referenz. Allerdings wird die Temperatur vor der eigentlich nötigen Zeit erreicht, sodass etwas zu früh aufgeheizt wurde. Hier ist noch Optimierungspotenzial z.B. hinsichtlich der Modellierung oder der Reglerparametrierung vorhanden. Auch am Ende der Nutzzeit wird die Temperatur auf der Referenz gehalten. Die Vorlauftemperatur wird erst nach Ende der Nutzzeit stark reduziert, sodass der Heizkreis auskühlt.

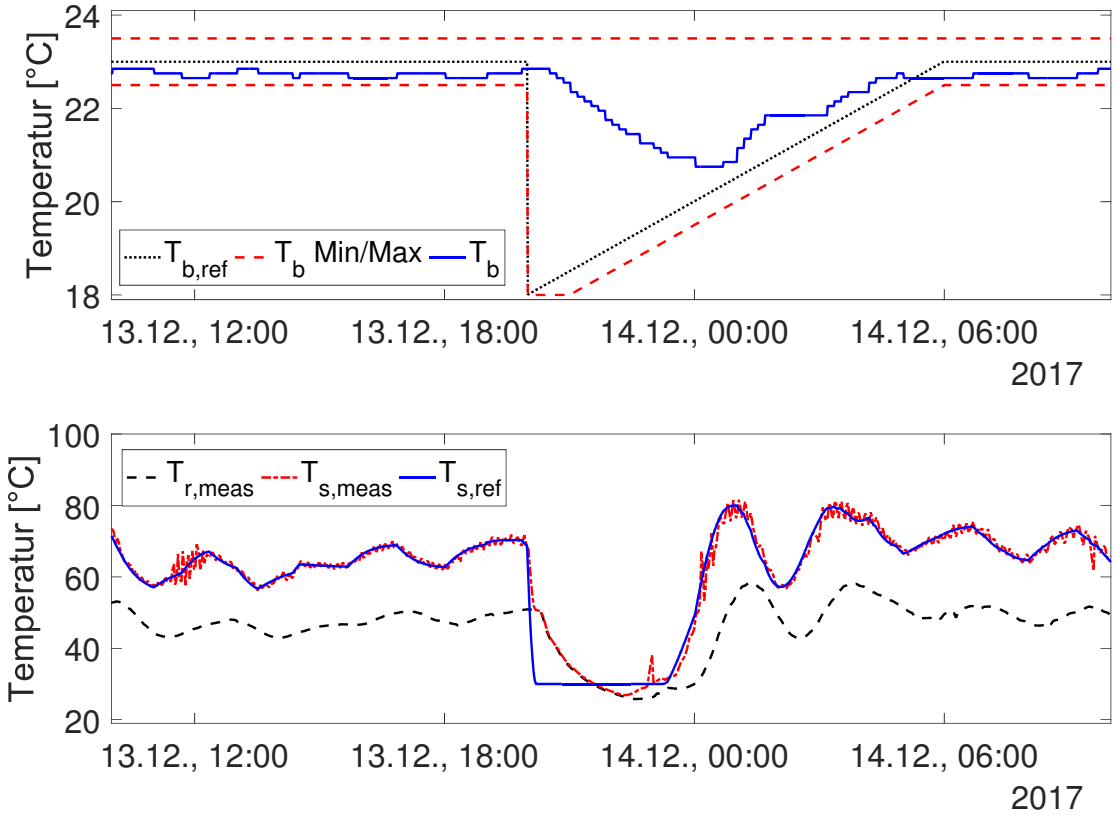


Abbildung 6.7-71: Messdaten des AMPC-SL Tests vom 13.12., 10:00 Uhr bis zum 14.12., 10:00 Uhr

Die Test des AMPC-SL an dem Heizkreis haben gezeigt, dass das prädiktive Reglerkonzept mit einem einfachen Modell auch an einer realen Anlage funktioniert. Eine Testumgebung wurde aufgebaut und Messergebnisse generiert, die mit der Theorie in großen Teilen vereinbar waren. Zudem sind noch einige zusätzliche Effekte aus dem praktischen Betrieb hinzugekommen. Der Heizkreis zeigt ein deutlich anderes Verhalten als mit der vorherigen Reglung, bei der die Sollvorlauftemperatur über eine Heizkurve vorgegeben wurde. Außerhalb der Nutzzeit ($4:30-20:00$ Uhr) wird diese um $10\,\mathrm{K}$ abgesenkt. Bei den eingestellten Werten für die Heizkurve und deren Absenkung handelt es sich um Erfahrungswerte und

eine Worst-Case Abschätzung, sodass das Gebäude zur Nutzzeit die Komfortanforderungen der Nutzer auf jeden Fall erfüllt. Dadurch kann es passieren, dass eine zu hohe Vorlauftemperatur angefordert wird, die gar nicht nötig ist. Der AMPC-SL Ansatz ist hier flexibler. Er regelt direkt auf die Raumtemperatur und handelt prädiktiv. Damit ermöglicht er ein deutlich größeres Abkühlen des Heizkreises in der Nacht, da er deutlich niedrigere Vorlauftemperaturen verwendet. Würde die Raumtemperatur dabei unter einen Minimalwert sinken, würde er diesem entgegenwirken. Zudem ist der Beginn des Aufheizpunktes, d.h. des Anstiegs der Vorlauftemperatur nicht zu einem festen Zeitpunkt, sondern er wird mithilfe des Modells unter Einbeziehung der Umgebungsbedingungen wie der Außentemperatur vom Regler bestimmt. Aufgrund des stärkeren Abkühlens sind zum Aufheizen somit höhere Vorlauftemperaturen als beim Heizkurvenansatz nötig. Dafür wird der Heizkreis am Wochenende und in der Nacht mit deutlich niedrigen Vorlauftemperaturen betrieben. Wie stark der Heizkreis auskühlen darf und wie das Aufheizen erfolgen soll, lässt sich über Parameter des MPC einstellen.

Neben den beschriebenen Entwicklungen hat die Implementierung am realen Gebäude folgende Erfahrungen gebracht, die zum Teil selbstverständlich sind, hier jedoch noch einmal aufgeführt werden sollen.

- Die Anlage ”lebt”, d.h. es kommt zu kurzzeitigen Datenausfällen oder anderen unvorhergesehen Beeinflussungen der Anlage z.B. durch Wartungen, die so in der Simulation nicht vorkommen.
- Die Anlage wandelt sich. Das bedeutet, dass die Daten mit denen die Parameter identifiziert wurden, unter Umständen nicht mehr dem aktuellen Anlagenzustand entsprechen.
- Da der Betrieb der Anlage sich durch die geänderte Regelung verändert, können zusätzliche Begrenzungen einen Einfluss bekommen, die in der Auslegung des Reglers noch nicht auffielen, da die Anlage zuvor nicht in dem Bereich betrieben wurde.
- Untergeordnete Regelkreise müssen zum Teil neu parametriert werden, da sich das Betriebsverhalten ändert und z.B. höhere Sollwertsprünge auftreten können.
- Durch manuelle Eingriffe können anlagenseitig zusätzliche Einschränkungen vorliegen, z.B. durch Begrenzung der Vorlauftemperatur.
- Das Reglertuning sollte nicht sehr aggressiv ausgeführt werden, da ansonsten Modellungenauigkeiten zu Oszillationen führen. Dies führt zu einem robusteren Verhalten.

Die Tests haben gezeigt, dass sich der Regler größtenteils plausibel verhält. Ausblickend lässt sich sagen, dass zusätzliche Forschung gerade dahingehend nötig ist, wie das optimale Verhalten des Heizkreise bezüglich des Tag-Nacht Betriebes ist. Hier ist zu klären, wo aus ökonomischer und energieeffizienter Sicht das Optimum liegt, d.h. wie stark der Heizkreis auskühlen darf. Ein starkes Auskühlen hat zum einen den positiven Effekt, dass der Heizkreis lange mit einer niedrigen Vorlauftemperatur betrieben wird, wiederum führt es auch zu einer langen Aufheizphase mit hoher Vorlauftemperatur am Morgen. Dies lässt sich über die Parameter des AMPC-SL , z.B. über die Wichtungen oder die Wahl der Referenzen einstellen. Kriterien für eine aus ökonomisch und energieeffizient optimale Wahl der Parameter sind zu untersuchen.
Zudem wurde bisher mit einer Implementierung auf einer Demonstrator Hardware, dem Raspberry Pi gearbeitet. Für die weitere Anwendung sollte eine Implementierung auf Ebene der DDC oder der GLT erfolgen. Hierbei stellt sich auch die Frage inwiefern es möglich ist, das Reglertuning sowie die Modellbildung zu automatisieren und während des Betriebes nachzuführen, um möglichst wenige Einstellparameter bei der letztendlichen Anwendung des Reglers zu haben. Um eine Leistungsregelung zu ermöglichen wäre auch ein Zugriff auf die Pumpe des jeweiligen Heizkreises von Vorteil. Somit lässt sich abschließend sagen, dass die Implementierung des AMPC-SL Ansatzes an einem Heizkreis eines Bürogebäudes die Ergebnisse aus der Simulation in vielen Punkten bestätigen konnte und sich darauf aufbauend weitere Punkte für die Optimierung ergeben haben.

6.8 Zusammenfassung und Ausblick

Im Rahmen des Arbeitspaketes A4 ”Dezentrale Netzwerkregelungen, Methoden und Design” wurde gezeigt, wie sich Heizungsanlagen effizient über multilineare Systeme modellieren lassen und wie sich diese Modelle im Reglerentwurf einsetzen lassen. Aus Vorgängerprojekt war bekannt, dass Modelle von Heizungssystemen sehr gut mithilfe von MTI Systemen modelliert werden können. Um auch sehr große

Anlagen abbilden zu können, wurde hier zunächst untersucht, welche Dekompositionsverfahren geeignet sind, um die Modelle möglichst effizient darzustellen. Hierbei wurden 4 Verfahren untersucht, wobei sich besonders die CP, die TT und die HT Zerlegung als geeignet herausstellten.

Im weiteren Verlauf wurde untersucht, wie sich die MTI Systeme für den modellbasierten Reglerentwurf einsetzen lassen. Dazu wurde zunächst die nichtlineare Entwurfsmethode der Feedback Linearsierung betrachtet. Mit dem Ziel einen robusten und einheitlichen Reglerentwurf für MTI Systeme zu ermöglichen, wurde das allgemeine Verfahren auf die spezielle Struktur der MTI Systeme angepasst. Das Ergebnis ist ein Entwurfsverfahren, speziell für MTI Systeme, dass mit der dekomponierten Tensorstruktur arbeitet und rein numerisch mit einem festen Satz an Operationen auskommt. Aufgrund ihrer dekomponierten Tensordarstellung sind MTI Systeme gut geeignet, um Modelle von sehr großen Anlagen speichereffizient zu beschreiben. Für sehr große Anlagen können zentrale Regler sehr komplex werden und einen großen Kommunikationsaufwand erfordern. Daher erfolgt die Untersuchung, wie sich eine geeignete dezentrale Reglerstruktur für MTI Systeme bestimmen lässt. Dies wurde für einen dezentralen Zustandsrückführungsregler realisiert und an einem Beispielgebäude in der Simulation getestet. Als weiteres Regelungsverfahren wurde die prädiktive Regelung betrachtet und untersucht, wie sich MTI Modelle bei der prädiktiven Regelung anwenden lassen. Dies wurde in ein Verfahren umgesetzt, bei dem eine sukzessive Linearisierung des MTI Modells durchgeführt wird. Somit konnten die positiven Modellierungseigenschaften der MTI Modelle und die gute Eigenschaften des Optimierungsproblems bei der Verwendung linearer Modelle kombiniert werden. Dieser Ansatz wurde auf einem Raspberry Pi für die Echtzeitanwendung implementiert und eine HiL Testumgebung aufgebaut. Nach erfolgreichen Tests wurde der Regler am dem Demonstrationsgebäude von Kieback&Peter am realen Gebäude eingesetzt. Auch beim prädiktiven Ansatz steigt die Komplexität, wenn sehr große Anlagen betrachtet werden, sodass untersucht wurde, wie sich die prädiktive Regelungsaufgabe für eine bestimmte typische Heizungsanlagenstruktur auf mehrere Regelungsknoten verteilen lässt. So konnte der Berechnungsaufwand deutlich reduziert werden. Die Methoden, die für MTI Systeme entwickelt wurden, wurden in einer MATLAB Toolbox zusammengefasst, sodass sie einfach wiederverwendbar sind.

Die Untersuchungen haben das große Potential für die Anwendung der MTI Systeme und der Tensorrechnung für die Modellierung und den modellbasierten Reglerentwurf gezeigt. Im Bereich der Systemanalyse der MTI Systeme wurden hier Eigenschaften wie die Stabilität nicht näher untersucht. Zudem ist es interessant, wie sich weitere Regelungsverfahren, wie die prädiktive Regelung komplett durch MTI Systeme beschreiben lassen, ohne das Hilfsmittel der linearen Modelle zu verwenden.

Literaturverzeichnis

[1] ASTROM, Karl J. ; MURRAY, Richard M.: *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton, NJ, USA : Princeton University Press, 2008. – ISBN 0691135762, 9780691135762

[2] BADER, Brett W. ; KOLDA, Tamara G. u. a.: *MATLAB Tensor Toolbox Version 2.6*. `http://www.sandia.gov/~tgkolda/TensorToolbox/`. Version: February 2015. – Available online

[3] BAKULE, Lubomir: Decentralized control: An overview. In: *Annual Reviews in Control* 32 (2008), Nr. 1, S. 87 – 98

[4] BÄRWOLFF, G.: *Höhere Mathematik für Naturwissenschaftler und Ingenieure*. 2. Auflage. Spektrum Akademischer Verlag, 2006. – ISBN 3–8274–1688–4

[5] BEMPORAD, A. ; MORARI, M. ; RICKER, N. L.: *Model Predictive Control Toolbox User's Guide*. The Mathworks Inc., 2016

[6] BOYD, Stephen ; VANDENBERGHE, Lieven: *Convex Optimization*. New York, NY, USA : Cambridge University Press, 2009. – ISBN 0521833787

[7] CANDÈS, Emmanuel J. ; WAKIN, Michael B. ; BOYD, Stephen P.: Enhancing Sparsity by Reweighted l1 Minimization. In: *Journal of Fourier Analysis and Applications* 14 (2008), Nr. 5, S. 877–905. – ISSN 1531–5851

[8] CHRISTOFIDES, Panagiotis D. ; SCATTOLINI, Riccardo ; PENA, David M. l. ; LIU, Jinfeng: Distributed model predictive control: A tutorial review and future research directions. In: *Computers and Chemical Engineering* 51 (2013), Nr. Supplement C, S. 21 – 41

[9] CICHOCKI, A. ; ZDUNEK, R. ; PHAN, A. ; AMARI, S.: *Nonnegative matrix and tensor factorizations*. Wiley, Chichester, 2009

[10] CICHOCKI, Andrzej ; MANDIC, Danilo P. ; PHAN, Anh H. ; CAIAFA, Cesar F. ; ZHOU, G. ; ZHAO, Qibin ; LATHAUWER, Lieven D.: Tensor Decompositions for Signal Processing Applications From Two-way to Multiway Component Analysis. In: *CoRR* abs/1403.4462 (2014). `http://arxiv.org/abs/1403.4462`

[11] DEUTSCHER, J.: Input-Output Linearization of Nonlinear Systems Using Multivariable Legendre Polynomials. In: *Automatica* 41 (2005), S. 299–304

[12] EKMAN, M.: *Modeling and Control of Bilinear Systems: Application to the Activated Sludge Process*, Uppsala University, Diss., 2005

[13] FARDAD, Makan ; LIN, Fu ; JOVANOVIC, Mihailo R.: Sparsity-promoting optimal control for a class of distributed systems. In: *American Control Conference*, 2011. – ISBN 9781457700804, S. 2050–2055

[14] GRASEDYCK, Lars: Hierarchical Singular Value Decomposition of Tensors. 31 (2010), 01, S. 2029–2054

[15] GRASEDYK, L. ; KRESSNER, L. ; TOBLER, C.: A literature survey of low-rank tensor approximation techniques. In: *GAMM-Mitteilungen* 36 (2013), S. 53–78

[16] HENZE, Gregor P.: Model predictive control for buildings: a quantum leap? In: *Journal of Building Performance Simulation* 6 (2013), Nr. 3, S. 157–158

[17] HITCHCOCK, Frank L.: The Expression of a Tensor or a Polyadic as a Sum of Products. In: *Journal of Mathematics and Physics* 6 (1927), Nr. 1-4, S. 164–189

[18] HUYCK, B. ; LOGIST, F. ; BRABANTER, J. D. ; IMPE, J. V. ; MOOR, B. D.: Constrained Model Predictive Control on a Programmable Automation System Exploiting Code Generation: Practical Considerations. In: *IFAC Proceedings Volumes* 44 (2011), Nr. 1, S. 12207 – 12212. – 18th IFAC

World Congress

[19] ISIDORI, Alberto: *Nonlinear Control Systems*. 3rd. Secaucus, NJ, USA : Springer-Verlag New York, Inc., 1995. – ISBN 3540199160

[20] KHALIL, K. H.: *Nonlinear Systems*. Prentice-Hall, Inc., 1996

[21] KOLDA, T. ; BADER, B.: Tensor Decompositions and Applications. In: *SIAM Review* 51 (2009), Nr. 3, S. 455–500

[22] KRESSNER, Daniel ; TOBLER, Christine: Algorithm 941: Htucker-A Matlab Toolbox for Tensors in Hierarchical Tucker Format. In: *ACM Trans. Math. Softw.* 40 (2014), April, Nr. 3, S. 22:1–22:22. – ISSN 0098–3500

[23] KRUPPA, K. ; PANGALOS, G. ; LICHTENBERG, G.: Multilinear Approximation of Nonlinear State Space Models. In: *19th IFAC World Congress, Cape Town* IFAC, 2014, S. 9474–9479

[24] LAWRYNCZUK, Maciej: Model predictive control with on-line optimal linearisation. (2014), 11, S. 2177–2182

[25] 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), Nr. Supplement C, S. 476 – 495

[26] LEE, Namgil ; CICHOCKI, Andrzej: Fundamental Tensor Operations for Large-Scale Data Analysis in Tensor Train Formats. In: *CoRR* (2016)

[27] LEWIS, Frank L. ; VRABIE, Draguna L. ; SYRMOS, Vassilis L.: *Optimal Control: Third Edition*. John Wiley and Sons, 2012. – ISBN 9780470633496

[28] LICHTENBERG, G.: *Hybrid Tensor Systems*, Hamburg University of Technology, Habilitation, 2011

[29] LIN, Fu ; FARDAD, Makan ; JOVANOVIC, Mihailo R.: Sparse feedback synthesis via the alternating direction method of multipliers. In: *American Control Conference*, 2012. – ISBN 9781457710957, S. 4765–4770

[30] LIN, Fu ; FARDAD, Makan ; JOVANOVIC, Mihailo R.: Design of optimal sparse feedback gains via the alternating direction method of multipliers. In: *IEEE Transactions on Automatic Control* 58 (2013), Nr. 9, S. 2426–2431. – ISSN 0018–9286

[31] LUNZE, J.: *Regelungstechnik 1*. 5. Springer, 2006. – ISBN 3–540–28326–9

[32] LUNZE, J.: *Regelungstechnik 2*. 4. Springer, 2006. – ISBN 3–540–32335–X

[33] LUNZE, Jan: *Control Theory of Digitally Networked Dynamic Systems*. Springer International Publishing, 2014

[34] MACIEJOWSKI, J.: *Predictive Control with Constraints*. Pearson Education Limited 2002, 2002. – ISBN 0 201 39823 0

[35] MORSI, A. ; ABBAS, H. S. ; MOHAMED, A. M.: Model predictive control of a wind turbine based on linear parameter-varying models. In: *2015 IEEE Conference on Control Applications (CCA)*, 2015, S. 318–323

[36] MÜLLER, B. ; DEUTSCHER, J.: 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

[37] NEGENBORN, R. R. ; MAESTRE, J. M.: Distributed Model Predictive Control: An Overview and Roadmap of Future Research Opportunities. In: *IEEE Control Systems* 34 (2014), Nr. 4, S. 87–97

[38] OSELEDETS, I. ; DOLGOV, S. ; KAZEEV, V. ; LEBEDEVA, O. ; MACH, T.: *TT-Toolbox 2.2*. `http://spring.inm.ras.ru/osel`. Version: 2012. – Available online.

[39] OSELEDETS, I.V.: Tensor-Train Decomposition. In: *SIAM Journal of Scientific Computing* 33 (2011), Nr. 5, S. 2295–2317

[40] PANGALOS, Georg: *Model-based controller design methods for heating systems*. Hamburg, TU Hamburg, Dissertation, 2016. `http://dx.doi.org/10.15480/882.1324`. – DOI 10.15480/882.1324

[41] PANGALOS, Georg ; EICHLER, Annika ; LICHTENBERG, Gerwald ; OBAIDAT, S. M. (Hrsg.) ; KOZIEL, Slawomir (Hrsg.) ; KACPRZYK, Janusz (Hrsg.) ; LEIFSSON, Leifur (Hrsg.) ; ÖREN, Tuncer (Hrsg.): *Hybrid Multilinear Modeling and Applications*. Springer International Publishing, 2015. – 71–85 S.

[42] POLYAK, B. T. ; KHLEBNIKOV, M. V. ; SHCHERBAKOV, P. S.: Sparse Feedback in Linear Control Systems. In: *Autom. Remote Control* 75 (2014), Dezember, Nr. 12, S. 2099–2111. – ISSN 0005–1179

[43] RÖBENACK, K.: Automatic Differentiation and Nonlinear Controller Design by Exact Linearization. In: *Future Generation Computer Systems* 21 (2005), S. 1372–1379

[44] SCATTOLINI, Riccardo: Architectures for distributed and hierarchical Model Predictive Control - A review. In: *Journal of Process Control* 19 (2009), Nr. 5, S. 723 – 731

[45] SCHMID, C. ; BIEGLER, L. T.: Quadratic Programming Methods for Tailored Reduced Hessian SQP. In: *Computers and Chemical Engineering* (1993)

[46] SCHULER, Simone ; ZHOU, Wenliang ; MÜNZ, Ulrich ; ALLGÖWER, Frank: Controller Structure Design for Decentralized Control of Coupled Higher Order Subsystems. In: *IFAC Proceedings Volumes* 43 (2010), Nr. 19, S. 269 – 274. – ISSN 1474–6670

[47] SILJAK, Dragoslav D.: *Decentralized control of complex systems*. Boston : Academic Press, 1991

[48] TASHTOUSH, Bourhan ; MOLHIM, M. ; AL-ROUSAN, M.: Dynamic model of an HVAC system for control analysis. In: *Energy* 30 (2005), Nr. 10, S. 1729 – 1745. – ISSN 0360–5442

[49] TUCKER, Ledyard R.: Some mathematical notes on three-mode factor analysis. In: *Psychometrika* 31 (1966), Nr. 3, S. 279–311

[50] VERVLIET, N. ; DEBALS, O. ; SORBER, L. ; VAN BAREL, M. ; DE LATHAUWER, L.: *Tensorlab 3.0*. `http://www.tensorlab.net`. Version: March 2016. – Available online.