Fakultät für
Maschinenbau,
Fahrzeugtechnik,
Flugzeugtechnik

HOCHSCHULE
FÜR ANGEWANDTE
WISSENSCHAFTEN
MÜNCHEN

# Entwicklung einer Temperatur-Folgeregelung für eine Heisswasser-Strecke mit Heizung und eine Mischwasser-Strecke mit Dosierventil mithilfe von Python und MATLAB®

Fakultät 03
der Hochschule München

**Abschlussarbeit**

zur Erlangung des akademischen Grades
Bachelor of Science

vorgelegt von

**Felix Müller**
geboren am 19.03.1996 in München

im Juni 2020

**Prüfer:** Dipl.-Ing. Armin Rohnen

## Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Abschlussarbeit selbstständig und nur unter Verwendung der von mir angegebenen Quellen und Hilfsmittel verfasst zu haben. Sowohl inhaltlich als auch wörtlich entnommene Inhalte wurden als solche kenntlich gemacht. Die Arbeit hat in dieser oder vergleichbarer Form noch keinem anderem Prüfungsgremium vorgelegen.

Datum: _____ Unterschrift: _____

# Danksagungen

Weit hinten, hinter den Wortbergen, fern der Länder Vokalien und Konsonantien leben die Blindtexte. Abgeschieden wohnen Sie in Buchstabhausen an der Küste des Semantik, eines großen Sprachozeans. Ein kleines Bächlein namens Duden fließt durch ihren Ort und versorgt sie mit den nötigen Regelialien. Es ist ein paradiesmatisches Land, in dem einem gebratene Satzteile in den Mund fliegen. Nicht einmal von der allmächtigen Interpunktion werden die Blindtexte beherrscht – ein geradezu unorthographisches Leben. Eines Tages aber beschloß eine kleine Zeile Blindtext, ihr Name war Lorem Ipsum, hinaus zu gehen in die weite Grammatik. Der große Oxmox riet ihr davon ab, da.

# Zusammenfassung / Abstract

## Abstract

# Inhaltsverzeichnis

# Listings

# Abbildungsverzeichnis

# Tabellenverzeichnis

# 1 Einleitung

Ziel dieser Arbeit ist es, eine Mischtemperaturfolgeregelung für den ersten Prototypen einer neuartigen Espressomaschine zu entwickeln. Zum Start der Arbeit besteht der Aufbau aus einem funktionsfähigen Wasserkreislauf, bestehend aus einer Pumpe zur Förderung des Wassers, einem Heizkessel, zwei Dosierventilen, jeweils eins in der Heiß- und Kaltwasserleitung, einem Brühkopf und einem Ablassblech. Zur Steuerung und Programmierung sind alle elektrischen Komponenten mit einem Einplatinen-Computer (Rasperry Pi 3b+) verbunden.

Für die Entwicklung der Mischregelung wird sowohl eine Kalt- als auch eine Heißwasserquelle benötigt. Die Quelle des Heißwassers besteht aus dem Heizkessel. Dieser besteht aus einem metallischen Zylinder, in dessen Inneren das Wasser durch eine elektrische Heizspirale erhitzt wird. Die Heizung liefert 1400W Leistung, und kann lediglich ein- oder ausgeschaltet werden, eine Regulierung der Leistung ist nicht möglich.
Um während des Kaffeebezugs eine konstante Heißwassertemperatur zu ermöglichen, wird der Entwicklung der Mischregelung die Entwicklung einer Boilertemperaturregelung vorangestellt.
Ziel der Boilertemperaturregelung ist es, sowohl während eines Pumpenbetriebs, als auch im statischen Zustand einen möglichst konstanten Sollwert der Temperatur zu erhalten.

Ein Regelkreis bestehlt im Allgemeinen aus dem Regelglied, der Regelstrecke und der Rückführung des Zustands der zu regelnden Größe. Zur Einstellung der Regelung muss das Übertragungsverhalten der Regelstrecke untersucht werden. Das Übertragungsverhalten der Strecke beschreibt, wie sich die Regelgröße verhält, wenn eine Änderung der Stellgröße erfolgt. Es wird beschrieben durch eine Übertragungsfunktion (ÜF), welche in vereinfachter Weise aussagt, wie sich der Ausgang im Vergleich zum Eingang der Strecke verändert.

Um das Übertragungsverhalten der Boiler-Regelstrecke zu definieren, werden zwei Ansätze beschrieben. Die ÜF muss aussagen, um wie viel °C sich die Temperatur des Wassers erhöht, wenn dem Wasser Energie in Form von elektrische Leistung zugeführt wird.
Im ersten Anlauf wird versucht, das Ohmsche Gesetz als Vergleichsmodell zu verwenden, um eine Differentialgleichung der erreichten Temperaturdifferenz (Spannung) in Abhängigkeit des thermischen Widerstands und des Wärmestroms zu erhalten.

Da sich der erste Anlauf als sehr umfangreich und zeitintensiv erweist, und das Hauptaugenmerk der Arbeit die erfolgreiche Entwicklung einer Mischregelung ist, wird als zweiter Ansatz die klassische Untersuchung des Übertragungsverhaltens durch das Aufnehmen von Sprung- und Impulsantworten durch Messungen angewendet.

# 2 Grundlagen

## 2.1 Raspberry Pi

Der Raspberry Pi (im folgenden RPi genannt) besteht aus einem Rechner und der Speicherkarte (SD Karte), auf der das Betriebssystem installiert wird. Die Speicherkarte fungiert also als Festplatte. Es handelt sich nicht um einen Mikrocontroller sondern um einen sogenannten Ein-Platinen Computer. Der RPi besitzt einen ARM-basierten Prozessor und einen Arbeitsspeicher wie ein normaler PC. Er verfügt über Schnittstellen wie HDMI, USB, Ethernet (Netzwerkanschluss) sowie zwei Flachbandkabelanschlüsse (z.B. für Kamera/Bildschirm). Für Ein- und Ausgänge existieren die sog. GPIOs, die General Purpose In- und Outputs. An diese können zum Beispiel Sensoren angeschlossen werden. Sie werden mit einer Spannung von 3,3 V versorgt und verkraften einen maximalen Strom von 16 mA.

### 2.1.1 Ordner-Architektur mdynlabc01

Da zu Beginn der Arbeiten die labortechnische Espressomaschine bereits vorhanden war, befindet sich auf dem Raspberry bereits Dateien, die mit dieser Arbeit nichts zu tun haben. Die aktuelle Ordnerstruktur wird grafisch im Anhang dargestellt. Wichtige, zum Betrieb erforderliche Verzeichnisse und Programme sollen hier kurz erläutert werden.

Nach der Anmeldung auf dem RPi (siehe 2.1.2) muss in das Verzeichnis *espresso* gewechselt werden. Hier befinden sich alle Dateien und Verzeichnisse für die Espressomaschine.
Die Datei *cOS_ExpanderInit.py* muss nach Inbetriebnahme einmal ausgeführt werden, um die Pins des RPi als Ein-/Ausgänge zu initialisieren.

Im Verzeichnis *BG2* befinden sich einige für den Testbetrieb nützliche Programme um die kalte Brühgruppe zu steuern. Diese werden in Tabelle 5.1 beschrieben.

Weiterhin befinden sich in dem Verzeichnis einige Dateien, die im Dateinamen mit *cOS_MCP3301* beginnen. Alle diese Programme bauen auf der Datei *cOS_MCP3301_CSV.py* auf, welche die Spannungen der an der Temperaturmessplatine angeschlossenen Sensoren ausließt, mithilfe der Temperatur-Spannungs-Kennlinien (*chanX.csv*) in Temperaturwerte wandelt, und in einer Datei mit dem Namen *buffer.csv* abspeichert.
Dieses Programm ist der Ort, an dem die Regelung (sowohl Boiler- als auch Mischregelung) implementiert wird.

### 2.1.2 Bedienung des RPi direkt über Konsole/Terminal

Um über die Kommandozeile auf den Rasperry zugreifen zu können, sind einige grundlegende Befehle nötig.

#### Anmeldung auf Raspberry Pi

Die Anmeldung erfolgt mit der folgenden Eingabe:

*ssh pi@mdynlabc01*

Anschließend wird das Passwort abgefragt:

pi@mdynlabc01 password: *faemae61*

#### Navigation auf dem RPi

Das Navigieren auf dem Rasperry erfolgt mit Unix Befehlen.

Um ein Verzeichnis zu wechseln, wird *cd* (change directory), gefolgt von dem Namen des nächsten gewünschten Unterordners, eingegeben. Mit Eingabe von *cd ..* gelangt man zurück in den jeweiligen Überordner.
Durch Eingabe von *ls -l* (long listing), werden alle Dateien und Ordner des momentanen Verzeichnis dargestellt.

Um ein neues Verzeichnis zu erstellen, wird *mkdir* (make directory), gefolgt von dem Namen des zu Erstellenden Ordners eingegeben. Um ein Verzeichnis zu löschen, wird *rmdir* (remove directory), gefolgt vom Namen der Verzeichnisses eingegeben.

Um ein Python Programm aus der Kommandozeile auszuführen, muss vor dem Dateinamen *./* eingegeben werden, getrennt mit einem Leerzeichen. Mit *STRG + C* kann ein Programm sofort unterbrochen werden.
Da jedoch die Steuerung der Ventile über Solid State Relais erfolgt, welche mit einem Befehl im Programmcode angesprochen werden, bewirkt die Eingabe von *STRG + C* beispielsweise kein Abschalten der Pumpe oder eine Ventil Verstellung. Daher sollte sich in jedem Verzeichnis mit Programmcode für die Steuerung ein Python-Programm befinden, welches alle Ventile auf ihre Ausgangsposition stellt, und die Pumpe (und Heizung) auf ihre Ausgangslage, bzw. ausschaltet (im Falle der Pumpe/Heizung).

#### Editieren von Programmen direkt auf RPi

Der Befehl *nano*, gefolgt von einem Dateinamen, lässt das Editieren von bestehenden Dateinen direkt aus der Kommandozeile zu, ebenso wie die Neuerstellung eines Programmcodes. Mit *STRG + O* wird die jeweilige Datei Äusgeschrieben". *STRG + X* beendet den Editier-Modus.
Möchte man nur den Code eines Programms einsehen, ohne Änderungen vorzunehmen, gibt man den Befehl *cat*, gefolgt vom Dateinamen ein.

Um ein Programm im Hintergrund laufen zu lassen, wird der *nohup*-Befehl verwendet.
Um beispielsweise das Programm *cOS_MCP*3301*_CSV.py* zu starten, wird die folgenden Eingabe getätigt: *nohub ./ cOS_MCP*3301*_CSV.py &*.
Aufgrund fehlender Zugriffsrechte kann der nohup-Befehl verweigert werden. Um dies zu beheben, können mit dem Befehl textitchmod 755, gefolgt vom Dateinamen, die Rechte geändert werden.
Um ein mit dem *nohup*-Befehl gestartetes Programm wieder zu beenden, muss die *process ID* (PID) bekannt sein. Diese wird nach Eingabe des Befehls zum Ausführen in der Kommandozeile angezeigt. Falls die ID-Nr. nicht bekannt ist, kann durch Eingabe von *ps -ef* eine Ausgabe aller laufenden Prozesse angezeigt werden.
Der Befehl *kill*, gefolgt von der PID, beendet das Programm.
(*sudo kill-9*)

### 2.1.3 Steuerung des RPi über MATLAB®

Anstelle der Kommandozeile kann der RPi auch über MATLAB® bedient werden. Zur Anmeldung wird hierfür der Befehl *raspi('IP-Adresse', 'Benutzername', 'Passwort')* in einer Variable gespeichert. Für die Labortechnische Espressomaschine sieht dies folgendermaßen aus: *rpi = raspi('192.168.1.3', 'pi', 'faemae61')*.

Das Ausführen von Programmen funktioniert mit dem *system*-Befehl. Beispiel:
*a = system(rpi, './espresso/cOS_*0.1*/cOS_ExpanderInit.py')*

### 2.1.4 Steuerung der SSRs

Die SSRs werden über eine Binärzahl (Bitmuster) gesteuert. Der Befehl zum Schalten eines SSRs lautet wie folgt:

**i2c = smbus.SMBus(1)**
**i2c.write_byte_data(0x21, 0x13, 0b00000000)**

Die erste Position *0x21* beschreibt den Portexpander.
Die zweite Position bestimmt den Port, wobei

'0x12' -> Port A
'0x13' -> Port B

entspricht.
Die Binärzahl am Ende des Befehls definiert, was ein- oder ausgeschaltet wird. Dabei entspricht jedes Bit einem SSR, welches ein- (1) bzw. ausgeschaltet (0) sein kann.
Die Bitmuster der für die Steuerung der kalten Brühgruppe relevanten SSRs sind in den untenstehenden Tabellen zusammengefasst.

Tabelle 2.1: Bitmuster **Port A** / 1

| **Last** | **PIN-Nr.** | **Bitmuster** | **Dezimalzahl** |
|---|---|---|---|
| Ventil Boiler Ablauf BG2 | 2 | 0b00000010 | 2 |
| Ventil Boiler Einlass BG2 | 5 | 0b00010000 | 16 |

Tabelle 2.2: Bitmuster **Port B** / 2

| **Last** | **PIN-Nr.** | **Bitmuster** | **Dezimalzahl** |
|---|---|---|---|
| Heizung | 9 | 0b00000001 | 1 |
| Ventil Ablauf BG2 (Y7) | 11 | 0b00000100 | 4 |
| Ventil Kaffeebezug BG2 (Y6) | 12 | 0b00001000 | 8 |
| Pumpe | 16 | 0b10000000 | 128 |

### Heizung Einschalten

Um das Ein-/Ausschalten der Heizung unabhängig vom Schaltzustand der restlichen SSRs zu ermöglichen, soll das Bit für die Heizung zum vorliegenden Bitmuster hinzuaddiert werden.
Der Tabelle 2.2 ist das Bitmuster der Heizung zu entnehmen. Um den Zustand der Heizung zu bestimmen, wird in jedem Schleifendurchlauf der Messroutine (2.1.5) der PORTB ausgelesen. Dies geschieht analog zum Schalt-Befehl der SSRs (write) mit *i2c.read_byte_data()*.
Die Addition des Bits darf nur erfolgen, wenn die Heizung zum gegebenen Zeitpunkt ausgeschaltet ist. Ist bei der Addition das Bit der Heizung bereits '1', kommt es zu einer fehlerhaften Situation:

Angenommen, an Port B liegt folgendes Bitmuster vor:

'0b00001001 = 5'

Wird nun eine 1 zu diesem Muster addiert, ändert sich der Dezimalwert des Bitmusters auf

'5 + 1 = 6 = 00000110',

wobei ein ungewollter Schaltzustand hervorgerufen wird.
Um dieses Problem zu umgehen, soll die Addition des Heizungsbits nur nach Überprüfung des Bitmusters an Port B erfolgen.
Betrachtet man alle möglichen Bitmuster-Kombinationen an Port B mit eingeschalteter Heizung,

0b00000001 = 1
0b00000101 = 5
0b00001001 = 9
0b00001101 = 13

fällt auf, dass der Dezimalwert bei eingeschalteter Heizung (letztes Bit = 1) stets ungerade ist.
Der Befehl zum Einschalten der Heizung durch die Addition

'**PORTB = i2c.read_byte_data(0x21, 0x13)**'
'**i2c.write_byte_data(0x21, 0x13, PORTB + 1)**'

darf also nur nach bestandener Prüfung

'if PORTB % 2'

erfolgen.

### 2.1.5 Ort der Reglercode-Implementierung

Der entwickelte Reglercode muss in einem Programm stehen, welches die Temperaturen der Messplatine in Echtzeit abfrägt und speichert. Auf dem RPi heißt dieses Programm: cOS_MCP3301_CSV.py. Wie die Dateiendung vermuten lässt, wird die Programmiersprache *Python* verwendet. Alle Programmcodes der Messungen zum Bestimmen des Systemverhaltens werden hier implementiert. Am Ende werden die Programmcodes der Boilerregelung sowie der Mischregelung hier gespeichert.

#### Struktureller Aufbau

Die für diese Arbeit geschriebenen Programme werden alle nach dem gleichen Schema aufgebaut.
Im ersten Teil werden alle benötigten Pakete und Programme importiert, die GPIOs initialisiert. Außerdem werden hier die Arrays für die Speicherung der Mess- und Programmdaten vorbelegt, und die Temperaturkennlinien der Sensoren eingelesen.

Im zweiten Teil werden die Variablen und Flags definiert. Die als *Flags* bezeichneten Variablen werden als Schalter verwendet, um Rückmeldung über den Zustand geben. Bevor beispielsweise der Befehl zum Verstellen der Dosierventile ausgeführt wird, muss festgestellt werden, ob sich der Sollwert der Ventilposition verändert hat. Wenn im Programmcode eine neue Ventilposition berechnet wird, und diese von der vorherigen abweicht, so wird das *verstell_flag* auf 1 gesetzt. Nachdem der Befehl zum Verstellen ausgeführt wurde, wird dieses flag wieder auf 0 zurückgesetzt.

Im dritten Teil beginnt die while-Schleife für den eigentlichen Mess-/Regelprozess. Zuerst werden die Messwerte (Spannungswerte) der Temperatursensoren erfasst. Mithilfe der oben

eingelesenen Temperaturkennlinien werden dann die Temperaturwerte in einem Array (werte_f_csv[ ]) gespeichert. Anschließend werden diese Daten in einer csv-Datei (comma separated values) gespeichert. Bei jedem Schleifendurchlauf werden die neuen Messwerte an diese Datei angehängt.
Der Code zum Verstellen der Dosierventile wird in Abschnitt 7.3.5 erläutert.
Weitere Programmteile werden in den folgenden Kapiteln beschrieben.

## 2.2 Regelungstechnik

Die Regelung eines Systems setzt sich von einer Steuerung eines Systems in der Art ab, dass durch eine Signalrückführung der Ist-Zustand der zu regelnden Größe an das Regelsystem gemeldet wird.
Tritt bei einer Steuerung eine Störung auf, wird diese nicht automatisch ausgeglichen. Eine Steuerung wirkt nur in eine Richtung. Bei der Regelung handelt es sich um einen geschlossenen Wirkungsablauf. Ist die Regelung aktiv, wird ein permanenter Vergleich zwischen Sollwert und Istwert der Regelgröße durchgeführt. Weicht der Sollwert vom Istwert ab, soll eine Veränderung der Stellgröße erfolgen.
In dieser Arbeit wird ein Regler für die Boilertemperatur und die Mischtemperatur erarbeitet. Die Stellgröße für die Boilerregelung stellt die Heizung des Boilers dar, für die Mischtemperaturregelung ist eines der Dosierventile als Stellgröße zuständig.
Um einen Wert für die Sollgröße zu erhalten, berechnet der Regler fortlaufend, welchen Wert der Sollwert in Abhängigkeit der aktuellen Bedingungen haben sollte. Als Reglerform wird für beide Regelungsfälle ein PID-Regler gewählt. PID bezeichnet die Regleranteile und steht für Proportional, Integral, und Derivativ. Die drei Regleranteile werden jeweils so ausgelegt, dass durch Multiplikation jeweils mit dem Regelfehler, dem integrierten Regelfehler und dem Derivativ des Regelfehlers über der Zeit eine Größe für den Stellwert des Regelfehlers aus der Reglerformel hervorgeht.
Der Regelfehler ist als Differenz des Sollwerts und Istwerts zu verstehen:

$$Regelfehler = Sollwert - Istwert \tag{2.1}$$

Die Formel eines PID-Reglers kann in vereinfachter Weise folgendermaßen dargestellt werden.

$$Stellgrösse = P \cdot Regelfehler + I \cdot Integral(Regelfehler) + D \cdot Derivativ(Regelfehler) \tag{2.2}$$

### 2.2.1 Klassifikation der Regelung

Regelungen können nach verschiedenen Kriterien eingeteilt werden. Eingeteilt nach der Art der Führungsgröße ist als erstes die Festwertregelung zu nennen. Diese Regelungsart kennzeichnet sich durch eine konstante Führungsgröße. Sie wird als passend angesehen, wenn sich der Sollwert der Regelgröße nicht ändert. Daher wird für die Boilerregelung eine Festwertregelung ausgewählt.

Eine weitere Variante stellt die Folgeregelung dar. Im Gegensatz zur Festwertregelung muss der Regler hier dafür Sorge tragen, dass auch bei Störeinflüssen der Führungsgröße die Regelgröße möglichst gut und möglichst schnell folgt.

In dem Buch (Einstieg in die Regelungstechnik mit Python) werden drei Regelstrategien aufgezählt. Die erste besagt, je größer die Regeldifferenz ist, desto größer soll die Gegenreaktion sein. Dies wird durch den Proportionalanteil realisiert.
Weiterhin gilt, je länger die Regeldifferenz vorliegt, desto größer die Gegenreaktion. Dies wird durch den Integralteil sichergestellt. Das Integral des Regelfehlers stellt dar ...
Als dritte Strategie gilt, je schneller die Änderung der Regeldifferenz erfolgt, desto größer die Gegenreaktion. Sie wird durch den Derivativanteil abgedeckt.
Alle diese Strategien zielen auf eine möglichst schnelle Anpassung der Stellgröße ab. Werden alle drei kombiniert, erhält man einen optimalen Regler.

### 2.2.2 Kenngrößen und Begriffe eines Regelkreises

In Abbildung 2.1 ist der Signalfluss eines Grundregelkreises dargestellt. Die darin vorkommenden Prozessgrößen werden im Folgenden kurz erläutert.

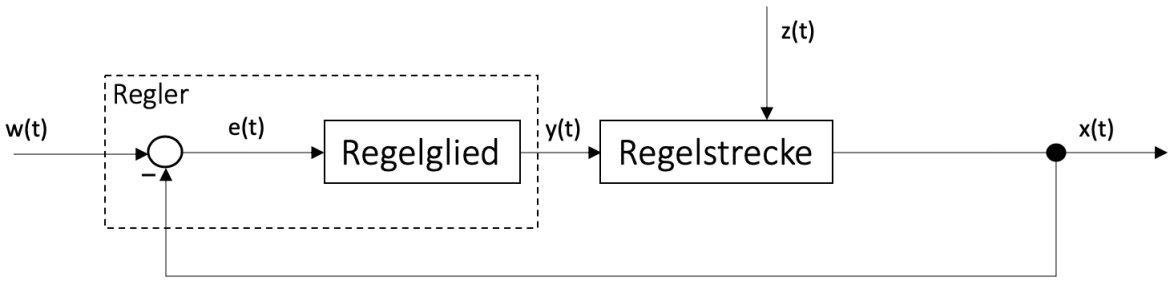


Abbildung 2.1: Regelkreis

**Regler**
Der Regler setzt sich zusammen aus dem Regelglied und dem Vergleichsglied. Der Regler erhält als Eingangsgrößen die Regelgröße und Führungsgröße, aus denen er die Stellgröße berechnet.

**Regelglied**
Das Regelglied bezeichnet den PID-Regler an sich. Schematisch kann das Regelglied wie in Abbildung 2.2 dargestellt werden.
Über die Konstanten der Regelungsanteile $K_P, K_I$ und $K_D$ lässt sich das Verhalten des Reglers beeinflussen.
**Regelstrecke**

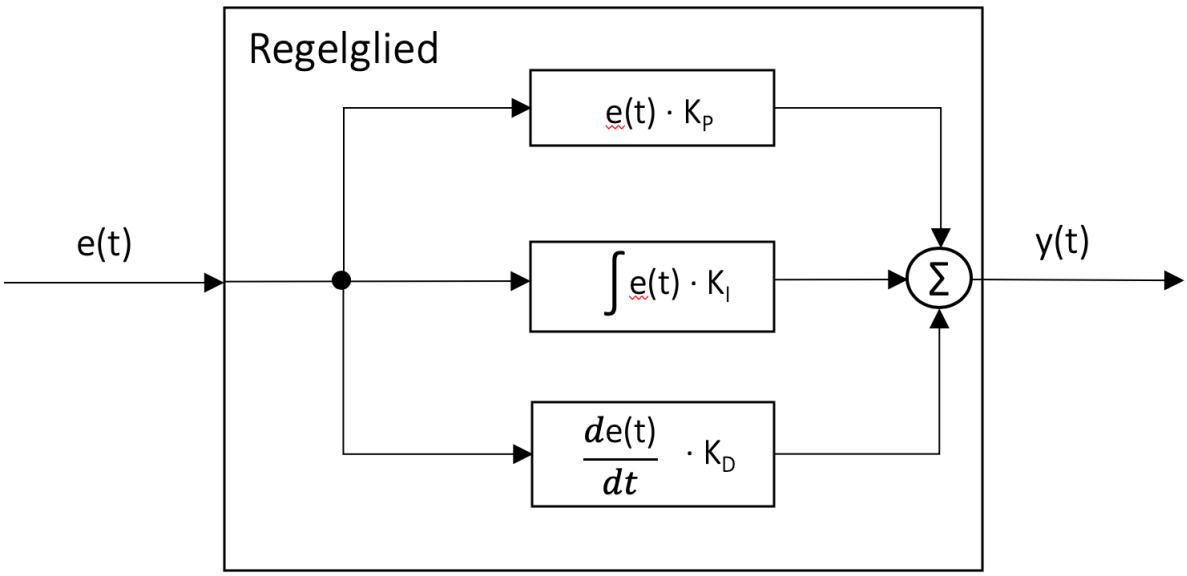


Abbildung 2.2: Regelglied

Die Regelstrecke bezeichnet das zu beeinflussende System. Sowohl im Falle der Boiler- als auch Mischtemperaturregelung besteht die Regelstrecke aus dem Medium Wasser, dessen Temperatur jeweils Gegenstand der Regelung ist.

**Führungsgröße** w(t)
Die Führungsgröße stellt die Zielgröße dar. Sie hat einen vorgegebenen Sollwert.
Für die Boilertemperaturregelung ist die Führungsgröße die Soll-Heißwassertemperatur, welche 100 - 110°C beträgt. Die Führungsgröße der Mischtemperaturregelung ist die Soll-Mischtemperatur.

$$w_{Boiler}(t) = T_{Boiler,Soll} = [100...110]°C \tag{2.3}$$
$$w_{Misch}(t) = T_{Mischwasser,Soll}(t) \tag{2.4}$$

**Regelgröße** x(t)
Die Regelgröße stellt die Systemgröße dar, die geregelt werden soll. Für die Boilertemperaturregelung ist die Regelgröße die Wassertemperatur im Inneren des Boilers. Für die Mischtemperaturregelung ist sie die Temperatur des Mischwassers.

$$x_{Boiler}(t) = T_{Boiler}(t) \tag{2.5}$$
$$x_{Misch}(t) = T_{Mischwasser}(t) \tag{2.6}$$

**Regelfehler** e(t)

Der Regelfehler stellt die Abweichung der Regelgröße von der Führungsgröße dar.

$$e_{Boiler}(t) = x_{Boiler}(t) - w_{Boiler}(t) \tag{2.7}$$
$$e_{Misch}(t) = x_{Misch}(t) - w_{Misch}(t) \tag{2.8}$$

**Störgröße** z(t)
Die Störgröße setzt sich aus allen äußeren Einflüssen zusammen, welche eine Veränderung auf die Regelstrecke bewirken können.
Als Störgröße für die Boilertemperaturregelung ist vor Allem der abführende Wärmestrom durch das Metall des Boilerkörpers zu nennen. Gleiches gilt für die Mischtemperaturregelung, wobei hier der abführende Wärmestrom durch das schlechter leitende Material der Wasserleitungen (Kunststoff) deutlich geringer ausfällt, und daher im Weiteren vernachlässigt wird. Als weitere Störgrößen können hier Strömungsverluste im Inneren der Dosierventile in Betracht gezogen werden.
Da sich die Störgrößen für die Mischtemperaturregelung nur schwer beschreiben lassen, wird auf sie nicht näher eingegangen.
**Stellglied und Stellgröße** y(t)
Wichtiger für die Regelung ist die Stellgröße. Sie geht als Ergebnis aus der Reglerformel hervor. Das Stellglied überträgt die Stellgröße als steuernde Wirkung des Reglers auf die Regelstrecke.
Das Stellglied der Boilertemperaturregelung stellt die Heizung dar. Als Stellgröße kommen entweder die Leistung (in Watt), oder die Betriebsdauer (in Sek) in Frage. Da die Heizung zum derzeitigen Stand nur ein- oder ausgeschaltet werden kann, wird hier zunächst die Betriebsdauer als Stellgröße festgelegt.
Für die Mischtemperaturregelung ist entweder das Kalt- oder Heißwasserdosierventil als Stellglied zuständig. Gewählt wird das Kaltwasserdosierventil. Die Stellgröße des Dosierventils ist die Eingangsspannung in mV (im Intervall von 0 - 9900 mV). Im Weiteren wird die Stellgröße der Mischtemperaturregelung in Prozent ausgedrückt.

$$y_{Boiler}(t) = t_{ein}[sek] \tag{2.9}$$
$$y_{Misch}(t) = O_{Kalt}[\%] \tag{2.10}$$

### 2.2.3 Einstellung und Anforderung eines Reglers

Als allgemeine Anforderungen an die Einstellung eines Reglers gilt, dass die Regelgröße nach einem Sprung der Führungsgröße möglichst schnell und ohne großes Überschwingen folgen.

Der Regelkreis muss stabil sein, dass heißt, es darf nicht zu einem Aufschwing-Verhalten kommen.

### 2.2.4 Übertragungsverhalten

Das Übertragungsverhalten eines zu regelnden Systems ist in das statische und das dynamische Übertragungsverhalten zu unterteilen. Dabei wird beim statischen Übertragungsver-

halten nur der sich einstellende Ausgangswert des Systems auf eine konstante Eingangsgröße untersucht. Die statische Kennlinie des Systems zeigt, ob eine Regelstrecke linear oder nicht linear ist. Zur Aufnahme der Kennlinie werden nacheinander unterschiedliche konstante Eingangsgrößen (Stellgröße) eingestellt, wobei die sich einstellende Ausgangsgröße aufgezeichnet wird.

Beim Untersuchen des dynamischen Übertragungsverhalten wird das Zeitverhalten des Systems auf eine Sprungeingabe des Stellglieds betrachtet. Die Sprungantwort bezeichnet die Reaktion des Systems auf einen rampenförmigen Anstieg der Stellgröße.

### 2.2.5 Totzeit durch Ansprechverzögerung

### 2.2.6 Einschwingverhalten

## 2.3 Reglerentwurfsverfahren

Der Reglerentwurf wird gemäß dem Vorgehensmodell aus [1] durchgeführt (Siehe Abbildung 2.3).

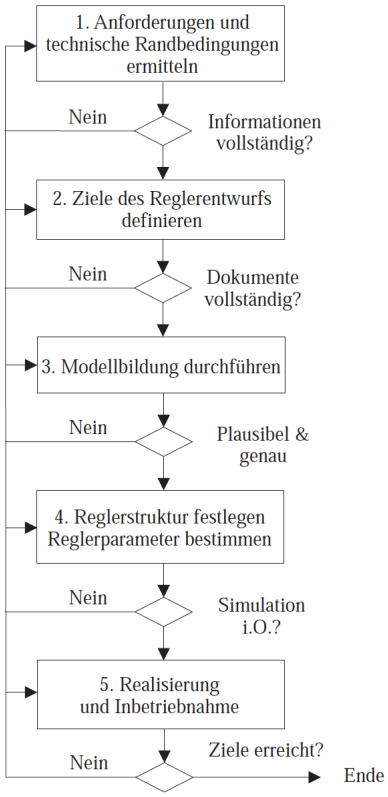


Abbildung 2.3: Rekursives Vorgehensmodell zum Reglerentwurf

# 3 Ansätze zur Definition des Übertragungsverhalten der Boiler-Regelstrecke

## 3.1 DGL aus Vergleichsmodell des Ohmschen Gesetzes

Der Kaffeeboiler mit der Oberfläche A ist mit Wasser gefüllt. Zur Erhitzung des Wassers wird Energie in Form von elektrischer Leistung zugeführt. Während die Temperatur ansteigt, findet ein Wärmeaustausch mit der Umgebung statt.
Vereinfachend wird angenommen, dass der Wärmeaustausch mit der Umgebung nur als Wärmeleitung über die Oberfläche $A_i$ (innen) stattfindet. Der Wärmeaustausch zwischen der Boilerwand und der Umgebung wird zunächst vernachlässigt.
Zum Zeitpunkt 0 entspricht die Wassertemperatur $\vartheta(0)$ der Boilerwandtemperatur $\vartheta_B$. Zum Startzeitpunkt wird die Heizung eingeschaltet, welche eine konstante elektrische Leistung $p_{el}(t)$ umsetzt. Mit steigender Temperatur tauscht das Wasser zunehmend Wärme mit der Boilerwand aus.
Die Wassertemperatur wird sich so lange erhöhen, bis sich Gleichgewischt zwischen zugeführter Leistung und des über die Oberfläche $A_i$ abgeführten Wärmestroms $p_{A,i}$ einstellt.

Um eine Differentialgleichung der erreichten Temperaturdifferenz zu erhalten, wird das Ohmsche Gesetz als Vergleichsmodell herangezogen.

$$U = R \cdot I \tag{3.1}$$

Dabei kann der elektrischen Widerstand mit dem thermischen Widerstand des Wassers bzw. des Metalls gleichgesetzt werden.
Bei der an einer Fläche A mit einer Wärmeübergangszahl $\alpha$ anliegenden Temperaturdifferenz $\Delta\vartheta(t)$ strömt über die Fläche A der Wärmestrom $p_A(t)$. Dieser ist zu vergleichen mit dem elektrischen Strom i(t), der aus einer anliegenden Spannung u(t) resultiert.
Der thermische Widerstand $R_{th}$ resultiert analog zum Ohmschen Gesetz zu:

$$R_{th} = \frac{\Delta\vartheta(t)}{p_A(t)} = \frac{1}{\alpha \cdot A} \tag{3.2}$$

Nach Umstellen der obigen Gleichung ergibt sich die Temperaturdifferenz zu

$$\Delta\vartheta(t) = \frac{1}{\alpha \cdot A} \cdot p_A(t) \tag{3.3}$$

Umgestellt nach dem Wärmestrom:

$$p_A(t) = \alpha \cdot A \cdot \Delta\vartheta(t) \tag{3.4}$$

Die Wärmekapazität ist in Anlehnung an die elektrische Kapazität (C = Q/U) definiert als der Quotient aus zugeführter Leistung d $p_c$ und der damit verbundenen Temperaturänderung d$\vartheta$.

$$C_{th} = \frac{dp_c}{d\vartheta} \tag{3.5}$$

Daraus resultiert die zugeführte Leistung zu

$$p_c(t) = C_{th} \cdot \frac{d\Delta\vartheta}{dt} \tag{3.6}$$

Wenn die Gleichung nach der Temperaturänderung aufgelöst und über der Zeit integriert wird, ergibt sich die Temperaturänderung des Wassers zu

$$\Delta\vartheta(t) = \frac{1}{C_{th}} \int_{-\infty}^{t} p_c(t)\ dt \tag{3.7}$$

Zur Verknüpfung der elektrischen und thermischen Größen wird eine Leistungsbilanz erstellt. Die elektrische Leistung $p_{el}(t)$ wird dem System von außen zugeführt. Über die Oberfläche gibt das System die thermische Leistung $p_A(t)$ ab, sobald die Wassertemperatur über die Metalltemperatur steigt. Die Differenz beider Leistungen $p_c(t)$ wird dazu verwendet, die Wassertemperatur zu erhöhen.

$$p_c(t) = p_{el}(t) - p_A(t) \tag{3.8}$$

### 3.1.1 Wärmeübergang von Wasser an Metall

Die Temperaturdifferenz zwischen dem Wasser und dem Boilermetall entspricht dem Produkt aus thermischen Widerstand und Wärmestrom:

$$\Delta\vartheta_i(t) = \frac{1}{\alpha_i \cdot A_i} \cdot p_{A,i}(t) \tag{3.9}$$

Daraus leitet sich der abgehende Wärmestrom ab zu

$$p_{A,i}(t) = \alpha_i \cdot A_i \cdot \Delta\vartheta_i(t) \tag{3.10}$$

wobei die Temperaturdifferenz im Boiler-Innenraum die Differenz aus der Wasser- und der umgebenden Metalltemperatur darstellt. Beide Temperaturen sind abhängig von der Zeit. Die Wärmeübergangszahl des Wassers ist eine Funktion der Wärmestromdichte und des Drucks.

Einsetzen der Gleichungen 3.4 und 3.6 in Gl. 3.8 liefert die Differentialgleichung der Temperaturänderung des Wassers:

$$C_{th,Wasser} \cdot \frac{d\Delta\vartheta_i}{dt} = p_{el}(t) - \alpha_i \cdot A_i \cdot (\vartheta_{Wasser}(t) - \vartheta_{Metall}(t)) \tag{3.11}$$

$$C_{th,Wasser} \cdot \Delta\vartheta_i = \int_{t=0}^{t} p_{el}(t) - \alpha_i \cdot A_i \cdot (\vartheta_{Wasser}(t) - \vartheta_{Metall}(t)) \; dt \tag{3.12}$$

### 3.1.2 Wärmeübergang von Metall an Umgebung

Analog ergibt sich die Differenzialgleichung der Metalltemperatur wie folgt:

$$C_{th,Metall} \cdot \frac{d\Delta\vartheta_a}{dt} = \alpha_i \cdot A_i \cdot (\vartheta_{Wasser}(t) - \vartheta_{Metall}(t)) - \alpha_a \cdot A_a \cdot (\vartheta_{Boiler}(t) - \vartheta_{Umgebung}) \tag{3.13}$$

$$C_{th,Metall} \cdot \Delta\vartheta_a = \int_{t=0}^{t} \alpha_i \cdot A_i \cdot (\vartheta_{Wasser}(t) - \vartheta_{Metall}(t)) - \alpha_a \cdot A_a \cdot (\vartheta_{Boiler}(t) - \vartheta_{Umgebung}) \; dt \tag{3.14}$$

### 3.1.3 Gesamt - Differentialgleichung

# 4 Hardware-Komponenten

## 4.1 Temperatursensoren

Bei den verwendeten Temperatursensoren handelt es sich um sogenannte NTCs, wobei im Laufe der Arbeiten verschiedene Typen verbaut wurden. Zum Zeitpunkt dieser Arbeit gibt es fünf Temperaturmessstellen:

Tabelle 4.1: Temperaturmessstellen

| **Messstelle** | **Position Array (Code)** | **Position CSV** |
|---|---|---|
| Kaltwasser-Temperatur | werte_f_csv[1] | buffer[2] |
| Heißwasser-Temperatur | werte_f_csv[2] | buffer[3] |
| Mischwasser-Temperatur | werte_f_csv[3] | buffer[4] |
| Boilerwasser-Temperatur | werte_f_csv[4] | buffer[5] |
| Brühgruppe-Temperatur | werte_f_csv[5] | buffer[6] |

### 4.1.1 Verschiedene Sensortypen

Dieser Arbeit voran ging eine Projektarbeit, im Zuge derer eigene Sensoren hergestellt wurden. Aufgrund von Dichtheitsproblemen wurden diese ersetzt.

Tabelle 4.2: Verbaute Temperatursensoren

| **Messstelle** | **Position Array (Code)** |
|---|---|
| Kaltwasser-Temperatur | Eigenbau |
| Heißwasser-Temperatur | Römer |
| Mischwasser-Temperatur | NTC10-S85_AVSRoemer |
| Boilerwasser-Temperatur | 2590 |
| Brühgruppe-Temperatur | Eigenbau |

### 4.1.2 Kalibrierung der Temperatursensoren

Bei den verwendeten Temperatursensoren handelt es sich um NTCs - thermische Widerstände. Diese sind ab Herstellung bereits kalibriert, jedoch ist bei jedem Sensor eine Standardabweichung zu erwarten.
Um diese Abweichung auszugleichen, werden die Sensoren vor dem Einbau ~~noch einmal~~ kalibriert.

### Aktuell verwendete Methode zur Temperatursensorkalibrierung

Die bisher verwendete Methode besteht aus einem offenen Gefäß gefüllt mit Wasser, welches von Raumtemperatur startend auf einer Herdplatte erhitzt wird. Dabei befindet sich möglichst nah am zu kalibrierenden Sensor ein Thermoelement, welches über eine NI-Messkarte die Temperatur misst und in MATLAB® live eingelesen wird, während der Widerstandswert des Sensors - über den Vorwiderstand umgerechnet in eine Spannung - ebenfalls in MATLAB® eingelesen wird.
Diese Methode hat sich als fehleranfällig erwiesen, weshalb die Spannungskurven nun mithilfe von MATLAB® "per Hand"berechnet werden. Die Berechnung wird unten beschrieben.

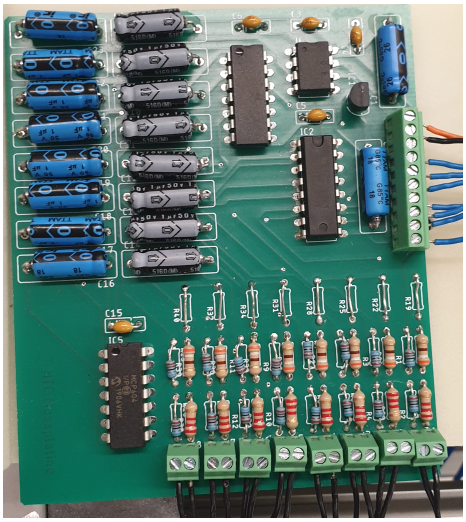

Abbildung 4.1: Temperaturmessplatine

### Vorschlag zur Verbesserung der Temperatursensorkalibrierung

Für zukünftige Kalibrierungen wird eine neue Methode vorgeschlagen.

### Berechnung der Spannungskurven

Vom Hersteller des Sensors ist eine Tabelle mit Widerstandswerten für Temperaturen gegeben. Zu Berechnen ist die am Spannungsteiler zu messende Spannung über die Temperatur. Der obere Widerstand des Spannungsteilers ist fest und wurde vorausgelegt, der untere Widerstand wird durch den NTC gebildet.
Die Berechnung der Spannung $U_A$ erfolgt über die Gleichung

$$U_A = U_E \cdot \frac{R_{NTC}}{R_V + R_{NTC}} \tag{4.1}$$

$U_E$ wird durch einen Referenzspannungsbaustein auf 4096 mV festgelegt.
Aus den Werten für Widerstand und Temperatur des Datenblatts wird mit MATLAB® der Vektor für die Spannung $U_A$ berechnet. Die Temperatur wird nun über der Spannung geplottet.
Über das Basic-Fitting Tool der Diagrammausgabe wird eine möglichst optimale Approximation durchgeführt. Aus den sich ergebenden Parametern wird die Temperatur in Abhängigkeit von der zuvor berechneten Spannung $U_A$ berechnet.
Dieser Temperaturvektor wird in einem csv-Format gespeichert und auf den RPi übertragen.

### 4.1.3 NI-Messgerät

Für die Streckenanalyse ist es erforderlich, neben der Wassertemperatur auch das Aufheizverhalten des Boilermetalls während den Messungen zu beobachten. Hierfür wurden zwei Thermoelemente von NI außen am Boilermetall angeklebt, und über eine NI-Messkarte in MATLAB® ausgelesen. Aufgrund der bereits beschriebenen anzunehmenden Temperaturschichtung im Boiler-Inneren wurde je ein Sensor oben und unten angebracht. Um für die Auswertung bestimmen zu können, zu welchen Zeitpunkten die Heizung in Betrieb ist, wird neben den beiden Kanälen der Thermoelemente ein dritter Kanal der Messkarte verwendet, um den Spannungswert am SSR der Heizung zu messen, und später plotten zu können.

## 4.2 Stellglied der Boilerregelung - Heizung

Das Stellglied des Regelkreises besteht aus einer Heizspirale, welche 1400W Leistung erbringt. Die Spirale befindet sich ca. im unteren Drittel der Boilerhöhe. Durch den Umstand, dass das Wasser über die Höhe nicht konstant erhitzt werden kann, ergibt sich unausweichlich eine Temperaturschichtung des Wassers im inneren.
Im Zuge der Streckenanalyse wird mithilfe von aufgeklebten Thermoelementen auf der Außenseite des Zylinders untersucht, wie sich die Temperaturschichtung während des Aufheizvorgangs verhält.
Es ist wichtig zu wissen, wie lange es nach der erstmaligen Inbetriebnahme beim Aufheizen dauert, bis sich ein annäherndes Temperaturgleichgewicht zwischen Wasser und Metall einstellt. Dies ist der Fall, wenn die Metalltemperatur gleich der Wassertemperatur im Inneren ist.
Des weiteren gilt es zu beachten, dass mit einer Temperaturerhöhung von Wasser auch eine Volumenänderung einhergeht. Diese Volumenänderung beläuft sich auf ca. 4%. Um sicherzustellen, dass sich im Zuge des Aufheizens nicht zu viel Druck im Boilerinneren aufbaut, muss sichergestellt werden, dass während die Heizung in Betrieb ist, Luft bzw. Wasser durch den Ausgang entweichen kann. Ebenso muss während dem Abkühlen, und während dem Entwässern des Boilers, Luft in das Innere eintreten können. Um dies zu gewährleisten, befindet sich nach dem Heißwasserauslass am Boiler ein Rückschlagventil.

## 4.3 Stellglied der Mischregelung - Dosierventile

## 4.4 Durchflussmessung

# 5 MATLAB® App - Bedienoberfläche

Zur leichteren Bedienbarkeit, sowie zur Überwachung der Parameter der labortechnischen Espressomaschine wird während dieser Arbeit eine grafische Benutzeroberfläche entwickelt. Nachdem die Verbindung zum Raspberry Pi hergestellt ist, können Bedienoberflächen (buttons) genutzt werden, um die auf dem RPi gespeicherten Python-Programme auszuführen. Mithilfe eines MATLAB®-Skripts werden die mit Python aufgenommenen Messdaten in MATLAB® importiert, und können anschließend in Echtzeit durch Achsensysmte dargestellt werden.

## 5.1 Einführung in App Designer/Unterschiede zu GUIDE

Die Gestaltung von grafischen Benutzeroberflächen kann in MATLAB® auf zwei Wege erfolgen. Die bis vor Kurzem gängige Variante ist das MATLAB® GUIDE (GUI Design Environment). MathWorks® empfielt jedoch inzwischen, bei der Neugestaltung einer App mit MATLAB® ihr neues Produkt zu verwenden, da das GUIDE in zukünftigen Versionen entfernt wird.
Seit dem Release R2016a bietet MATLAB® mit dem App Designer eine neue Entwicklungsumgebung zur Gestaltung von grafischen Benutzeroberflächen (GUI - Graphical User Interface) an. Die größten Unterschiede und Vorzüge des App Designers sollen hier kurz dargestellt werden.

Der App Designer bietet ein sehr benutzerfreundliches grafisches Design, dass es auch Nutzern ohne viel Programmiererfahrung ermöglicht, eine Anwendungsorientierte Nutzeroberfläche mit vielen verschiedenen Funktionen zu gestalten.
Im Gegensatz zur GUIDE gibt es im App Designer die Möglichkeit, direkt zwischen der Design- und der Code Ansicht umzuschalten. Ähnlich zur GUIDE wird im App Designer bei der Erzeugung einer Komponente (z.B. Push Button) automatisch die für die Programmausführung benötigte Code-Struktur erzeugt. Anders als der GUIDE lässt die neue Code-Umgebung jedoch nicht mehr zu, den erzeugten Code zu manipulieren. Dies schließt das versehentliche Entfernen von wichtigem GUI-Code und eine damit verbundene Fehlersuche aus.

## 5.2 Funktionen der Bedienoberfläche

**Push Buttons** Push Buttons sind Bedienelemente, die bei Betätigung eine Funktion ausführen. Die auszuführende Aktion kann in der Code View mithilfe von MATLAB®-Code

genau programmiert werden. Die Push Buttons der Bedienoberfläche werden alle nach dem gleichen Schema programmiert.
Jeder Button erhält ein 'Flag', welches beschreibt, ob das Element betätigt ist oder nicht. Die Flags sind boolesche Operatoren, welche 'wahr' oder 'falsch' sein können (in MATLAB® true oder false).
Als Beispiel wird der Programmcode des 'Messung Starten' - Push Button beschrieben:
Zu Beginn müssen die im Code verwendeten Variablen als global definiert werden, da alle Flags beim Start der App auf 'false' gesetzt werden. Damit keine Fehlermeldung ausgegeben wird, soll der Code nur ausgeführt werden, wenn das 'systemFlag' wahr ist. Dies ist der Fall, wenn der 'Start'-Button (unten beschrieben) zum ersten Mal betätigt wird. Besteht zum Zeitpunkt des Klicks keine Verbindung zum RPi, und ist somit das systemFlag false, wird im oberen Statusfenster die Meldung ausgegeben: 'XXX nicht möglich, System abgeschaltet'.
Wenn bei Klick auf den 'Messung Starten'-Button bereits 'true' war, bedeutet es, dass die Fläche bereits betätigt wurde, und nun wieder deaktiviert wird. In diesem Fall soll das Programm 'OFF_BG2_E61.py' auf dem RPi ausgeführt werden.
Wenn der Button bei Betätigung zum ersten Mal geklickt wird, soll das Messprogramm 'cOS_MCP3301_CSV_Regler_V6.py' ausgeführt werden.
Für beide Fälle muss nach der Ausführung des Codes das Flag auf den gegensätzlichen Wert gesetzt werden, also bei true auf false, bei false auf true.

Listing 5.1: MATLAB® Programmcode eines Push Buttons

```
1  function MessungstartenButtonPushed(app, event)
2          global systemFlag MessungFlag
3          if systemFlag == true
4                  if MessungFlag == true
5                          a = system(rpi, './espresso/cOS_0.1/BG2/
                               OFF_BG2_E61.py');
6                          MessungFlag = false;
7                  else
8                  a = system(rpi, './espresso/cOS_0.1/
                       cOS_MCP3301_CSV_Regler_V6.py');
9                  MessungFlag = true;
10                 app.AnzeigeSzenarienEditField.Value = 'Regler laeuft';
11                 end
12         end
13 end
```

**startupFcn** Bei Klick auf 'Run' in der MATLAB® App wird als erstes die startupFcn aufgerufen. Diese passt das Fenster der Anwendung auf den Desktop zentriert an. Anschließend werden alle Flags auf 'false' gesetzt. Außerdem werden beim ersten Aufruf der App die Achsensysteme unsichtbar gemacht. Diese sollen nur sichtbar werden, wenn die Temperaturmessdaten visualisiert werden sollen.

**StartButtonPushed** Bei Klick auf den StartButton wird die Verbindung zum RPI hergestellt (siehe 2.1.3). Das 'systemFlag' wird auf true gesetzt. Die startTime wird 0 gesetzt. Dies signalisiert der visualisierung_v2.m (siehe 5.1 Temperaturen PushButtons), dass das Array der Messdaten 'daten' neu von Zeile 1 ab befüllt wird.

Anschließend wird das Programm 'cOS_ExpanderInit.py' ausgeführt. Dies setzt Pin 20 und 21 des RPi als Eingang und Schalter. Im oberen Statusfeld wird der String 'System eingeschaltet' ausgegeben. Anschließend wird die alte buffer.csv Datei gelöscht. Bei Klick auf Start in der App muss der Befehl *'nohup./cOS_MCP*3301*_CSV.py&'* bereits ausgeführt sein. Der Timer wird aufgerufen und an die Funktion 'visualisierung_v2.m' übergeben. Sobald dies geschehen ist, nach einem StartDelay von 5 Sekunden, wird die aktuelle Zeit im oberen Statusfenster ausgegeben. Sie aktualisiert sich mit der Periode 1, also einmal pro Sekunde.

**StopButtonPushed** Bei Klick auf StopButton wird der systemFlag zurück auf false gesetzt und der scheduler (timer) gestoppt und gelöscht. Die Verbindung zum RPI wird beendet, indem die globale Variable rpi gelöscht (clear) wird. Im oberen Statusfenster wird der String 'System abgeschaltet' ausgegeben.

**Szenarien - PushButtons** Die Szenarien Push Buttons dienen zum Bedienen der Maschinenfunktionen.

Tabelle 5.1: Szenarien

| **Szenario** | **Flag** | **py.-Programm** | **Pumpe** | **Heizung** | **Y6** | **Y7** | **Y8** | **Y9** |
|---|---|---|---|---|---|---|---|---|
| Aufheizen | BG2Flag | BG2_Aufheizen.py | OFF | ON | x | | | x |
| Einregeln | BG2Flag | BG2_Einregeln.py | ON | OFF | | | | x |
| Kaffeebezug | BG2Flag | BG2_Kaffeebezug.py | ON | OFF | x | | | x |
| Spülen | BG2Flag | BG2_Spülen.py | ON | OFF | x | x | | x |
| Entwässern | BG2Flag | BG2_Entwässern.py | OFF | OFF | x | | x | x |

**Kaffeebezug**: Pumpe muss laufen, Wasser muss in Boiler einlaufen können (-> Y9), Durchfluss zwischen Mischwasser und Brühkopf geöffnet (-> Y6), Dosierventil muss leicht geöffnet werden, um Druckanstieg im Boiler zu vermeiden
**Aufheizen**: Heizung muss an sein, Wasser muss in Boiler einlaufen können (-> Y9), Durchfluss zwischen Mischwasser und Brühkopf geöffnet (-> Y6)
**Einregeln**: Pumpe muss laufen, Wasser muss in Boiler einlaufen können (-> Y9), Dosierventil muss leicht geöffnet werden, Wasser wird in Ablaufblech geleitet (Y6 ist nicht geschaltet)
**Spülen**: Pumpe muss laufen, Wasser muss in Boiler einlaufen können (-> Y9), Wasser muss in Brühkopf (Blindsieb eingesetzt) geleitet werden (-> Y6), und wird von dort rückgespült in das Ablaufblech (-> Y7)

**Entwässern**: Wasser muss aus dem Boiler unten abgelassen werden (-> Y8) und wird in das Ablaufblech geleitet

**Temperaturen PushButtons** Alle folgenden PushButtons sind nach dem selben Prinzip programmiert, wie die SzenarienButtons. Bei Klick auf die jeweiligen Temperaturbuttons wird das zugehörige Achsensystem wieder sichtbar gemacht, und der jewilige Flag true gesetzt. Je nach gesetztem Flag wird die entsprechende Spalte im Array daten geplottet.

Tabelle 5.2: AppDesigner Array: daten()

| | |
|---|---|
| daten(:, 1) | Zeitstempel |
| daten(:, 2) | BG2 Kaltwasser |
| daten(:, 3) | BG2 Heißwasser |
| daten(:, 4) | BG2 Mischwasser |
| daten(:, 5) | BG2 Boiler |
| daten(:, 6) | BG2 Brühgruppe |
| daten(:, 7) | $\Delta T_{Boiler}$ |

**Sollwertübergabe durch csv-Datei**

# 6 Boilertemperaturregelung

Für die Regelung der Wassertemperatur des Boilers wird ein Regler entwickelt. In Abschnitt 3.1 wurde bereits das System beschrieben. Daraus geht hervor, dass sich die Metallhülle des Boilers langsam und ungleichmäßig aufwärmt. Es wird angenommen, dass sich eine (annähernde) Linearität des Systems erst ergibt, wenn sich das System in einem eingeschwungenen Gleichgewichtszustand befindet. Dieser Zustand wird als erreicht gesehen, wenn die Temperatur der unteren Metallumhüllung des Boilers eine Temperatur von etwa 80°C erreicht. Um das Aufheizverhalten zu untersuchen, werden auf der Außenseite des Boilers zwei Temperaturfühler angeklebt, die über eine NI-Messkarte den Temperaturverlauf oben und unten aufnehmen.
Anschließend werden mehrere Streckenmessungen durchgeführt, um das Übertragungsverhalten der Regelstrecke Boiler zu bestimmen.

### Anforderungen an den Regler

**Genauigkeit**
Die Boilerregelung muss dafür sorgen, dass die Heißwassertemperatur stets über der gewünschten Brühtemperatur (Mischwasser-Sollwert) liegt. Diese beträgt zwischen 90 und 120°C. Für die Solltemperatur des Boilerwassers wird ein Wert von 115°C angesetzt. Die Maximaltemperatur des Boilerwassers, welche nie überschritten werden darf, wird auf 120°C festgelegt (siehe 6.3.3).
Die Siedetemperatur von Wasser liegt unter den gegebenen Bedingungen bei 98,262°C. Es ist erstrebenswert, dass sich im Inneren des Boilers nicht zu viel Wasserdampf bildet, da dieser neben dem Aufbauen eines erhöhten Boilerdrucks auch die Messung des Temperatursensors verfälscht. Daher wird eine höhere Genauigkeit angestrebt.

Da die Heizung lediglich ein- oder ausgeschaltet werden kann, beträgt die geringste einzuführende Heizleistung bei normalem Betrieb 1400J (1 Sekunde * 1400J/sek). Um eine geringere Heizleistung zu realisieren, soll eine Pulsweitenmodulation im Programmcode simuliert werden (siehe Abschnitt **??**).

**Stellgröße**
Das Regelglied soll den Wert '1' ausgeben, wenn die Heizung mit voller Leistung betrieben werden soll. Beträgt der berechnete Wert der Reglerformel weniger als '1', soll die Heizung mittels Pulsweitenmodulation betrieben werden.
Bei einem Tastgrad von 10% würde die Heizung nur für ein Zehntel einer Sekunde betrieben werden. Da dies nicht realisierbar ist (u.A. wegen Signalverzögerung), soll sich die Regelgröße

nur alle 10 Sekunden ändern.

## 6.1 Boiler - Regelkreis

Der Regelkreis besteht aus der Regelstrecke (Boiler aus Metall gefüllt mit Wasser), dem PID Regler, der Rückführung der erreichten Temperaturdifferenz, und der Solldifferenz, welche sich aus Solltemperatur und aktueller Temperatur berechnet. Der Regelkreis ist in Abbildung 6.1 abgebildet.
Die Differenz der Soll-Differenz abzüglich der Ist-Differenz ergibt den Regelfehler, welcher den Eingang in den PID-Regler darstellt. Als Ausgang des Reglers geht die Stellgröße hervor.
Die Übertragungsfunktion der Regelstrecke gibt an, um wie viel °C sich das Wassers durch die gegebene Stellgröße erhitzt.

Tabelle 6.1: Regelgrößen der Boilerregelung

| | | |
|---|---|---|
| $w_{Boiler}(t)$ | Führungsgröße | $T_{Boiler-Soll}$ |
| $x_{Boiler}(t)$ | Regelgröße | $T_{Boiler}$ |
| $e_{Boiler}(t)$ | Regelfehler | Führungsgröße - Regelgröße |

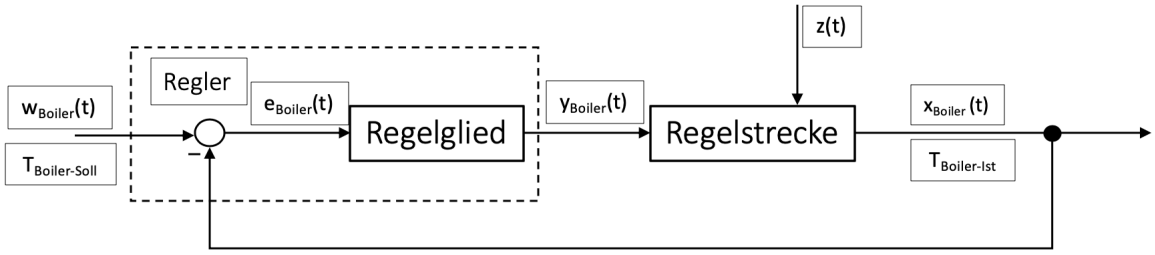


Abbildung 6.1: Regelkreis-Schema des Boilerreglers

## 6.2 Boiler - Reglerentwurf

Aus dem Aufheizverhalten wird zunächst eine Übertragungsfunktion abgeleitet, mit der ein erster Regler entworfen wird. Im Anschluss wird das Verhalten der Regelstrecke genauer untersucht, um den Regler für verschiedene Szenarien zu optimieren.

### 6.2.1 Aufnehmen der Sprungantwort der Boiler-Regelstrecke

Zum Aufnehmen einer Sprungantwort nimmt die Stellgröße zum Startzzeitpunkt sprunghaft den Wert 1 an, und hält diesen bei.
Um das Aufheizverhalten zu untersuchen, sowie eine erste Annahme über das Übertragungsverhalten der Regelstrecke treffen zu können, wird zur Aufnahme der Sprungantwort bei der erstmaligen Inbetriebnahme die Heizung für 200 Sekunden eingeschaltet.

#### Analyse des Aufheizverhalten

Um eine Annahme darüber treffen zu können, wann der Gleichgewichtszustand nach dem erstmaligem Einschalten der Heizung eintritt, muss festgestellt werden, wie schnell die Temperatur steigt, wenn die Heizung ein-, oder ausgeschaltet ist.
Untenstehende Abbildung 6.2 zeigt den Verlauf der Messung. Nach 770 Sekunden wird über einen Zeitraum von 35 Sekunden ein PWM-Signal (siehe Abs. **??**) mit 50% Tastgrad eingeschaltet.
Während der Messung werden die Temperaturverläufe des Boilerwassers, der unteren und oberen Boilermetall-Temperatur, sowie die Spannung des Heizungs-SSRs aufgenommen. Wenn die Heizung in Betrieb ist, befindet sich die Spannung nahe 0V, während des Betriebs nahe 10V.

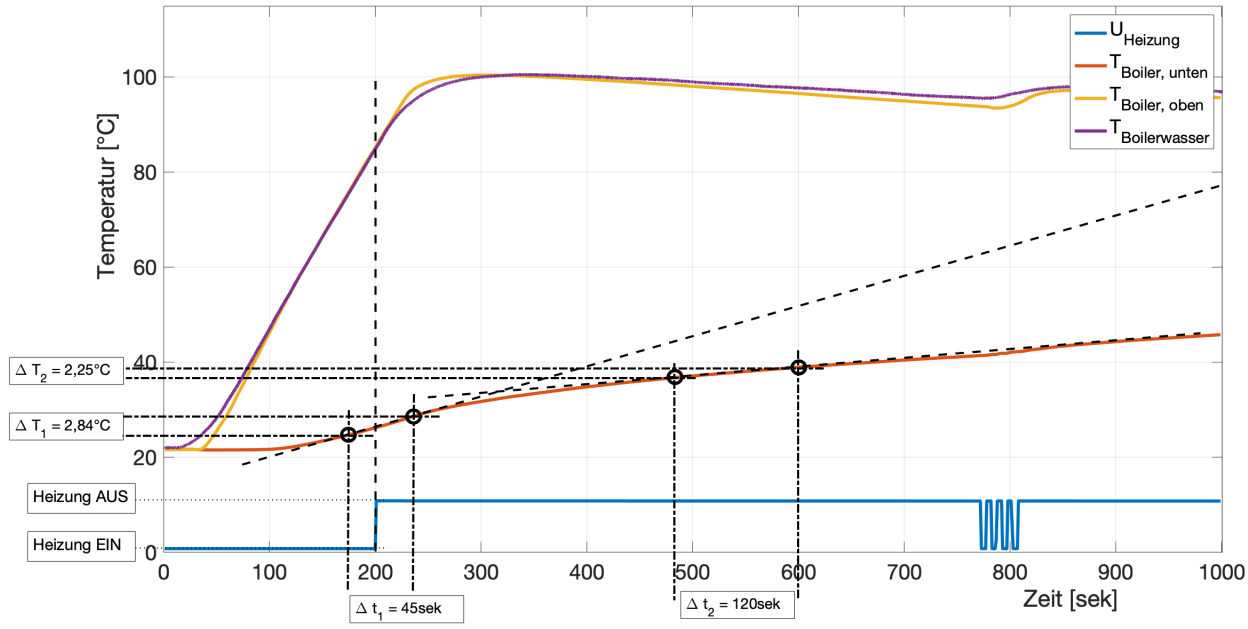


Abbildung 6.2: Aufheizverlauf Boilertemperaturen

Wie zu erkennen ist, fängt die Temperatur des Wassers nach einer kurzen Verzögerung an zu steigen. Die Aufheizgeschwindigkeit beträgt:

$$\frac{dT_{Boilerwasser}}{dt} = \frac{80 - 60[°C]}{187 - 133,2[sec]} = 0,372 \left[\frac{°C}{sec}\right] \tag{6.1}$$

Die Metalltemperatur an der oberen Messstelle folgt der Wassertemperatur nach kurzer Zeit. Die Temperatur der unteren Messstelle erwärmt sich deutlich langsamer.
Die sich einstellende Aufheizgeschwindigkeit während die Heizung in Betrieb ist, ergibt sich zu

$$\frac{dT_{Boiler,unten}}{dt} = \frac{27,8379 - 24,9956[°C]}{225 - 180[sec]} = 0,063162 \left[\frac{°C}{sec}\right] \tag{6.2}$$

Wie der Abbildung zu entnehmen ist, würde es bei eingeschalteter Heizung über 1000 Sek (>16 min) dauern, bis die untere Boilertemperatur einen Wert von 80°C erreicht. Eine derart lange Heizdauer ist aufgrund der thermodynamischen Grenzen des Systems nicht möglich. Die maximale Boilerwassertemperatur soll zu keinem Zeitpunkt 120°C überschreiten, da sich sonst ein zu hoher Druck aufbauen kann.

Bei ausgeschalteter Heizung stellt sich eine geringere Aufheizgeschwindigkeit der unteren Messstelle ein:

$$\frac{dT_{Boiler,unten}}{dt} = \frac{38,9243 - 36,6699[°C]}{600 - 480[sec]} = 0,01879 \left[\frac{°C}{sec}\right] \tag{6.3}$$

Wird angenommen, dass dieser Gradient konstant bleibt (dies ist nicht der Fall), dauert es nach dem Abschalten der Heizung

$$\frac{(80 - 35)[°C]}{0,01879 \left[\frac{°C}{sec}\right]} = 2395[sec](= ca.40[min]) \tag{6.4}$$

bis eine Temperatur von 80°C erreicht wird.

Folglich muss entweder eine lange Aufheizdauer in Kauf genommen werden, oder eine andere Methodik gewählt werden, um den Gleichgewichtszustand schneller zu erreichen.

### Analyse des Übertragungsverhaltens

Abbildung 6.2 zeigt bereits den Sprungverlauf, wenn die Heizung bei erstmaliger Inbetriebnahme eingeschaltet wird. Die konstante Aufheizrate ohne ein Einschwingen auf einen Endwert weißt auf ein integrales Übertragungsverhalten hin.

Für das Übertragungsverhalten der Boilerregelstrecke wird zunächst ein IT1-Verhalten angenommen. Die Sprungantwort eines solchen Verhaltens ist in nachstehender Abbildung 6.4 dargestellt.
Um aus den Messdaten der Abbildung 6.2 die klassische Darstellungsweise einer Sprungantwort zu erhalten, werden die Daten so umsortiert, dass statt der gemessenen Temperatur

die erreichte Temperaturdifferenz über der Zeit dargestellt werden kann. Aus dem Temperaturverlauf (Abb. 6.3) werden anschließend die Parameter der Übertragungsfunktion abgelesen.

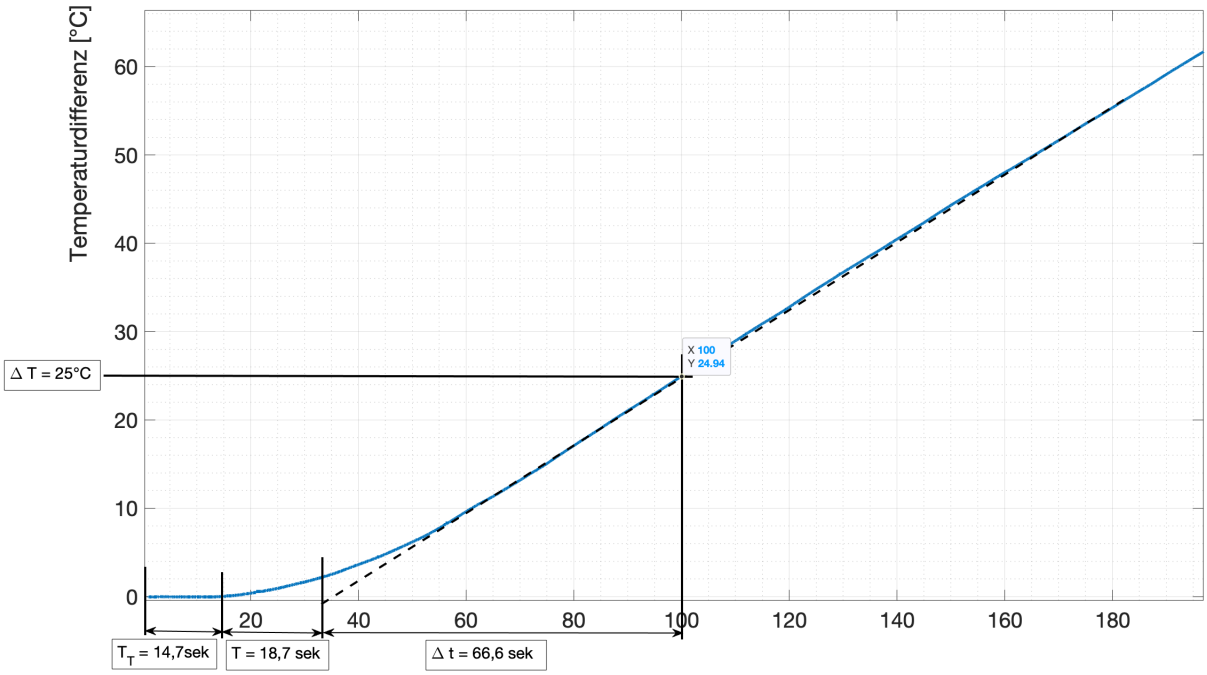


Abbildung 6.3: Übertragungsfunktions-Parameter aus Sprungantwort

Die zeitliche Verzögerung, bevor die Temperatur erstmals ansteigt, wird als Totzeit eingeordnet.

### Verzögerung - Totzeit

Ein Totzeitglied bewirkt den zeitlichen Versatz der Ausgangsgröße um eine konstante Dauer (Totzeit $T_T$).
Die bekannt Problematik eines Totzeitglieds besteht darin, dass es sich mathematisch nicht durch eine gewöhnliche Differentialgleichung beschreiben lässt. Die Übertragungsfunktion des Totzeitglieds im Frequenzbereich (unten dargestellt) enthält eine e-Funktion, welche eine transzendente Funktion ist. Die Rücktransformation einer transzendenten Funktion in den Zeitbereich ist mit gewöhnlichen Mitteln nicht möglich.
Da die Übertragungsfunktion der Regelstrecke lediglich für die Auslegung der Reglerparameter benötigt wird, ist eine Rücktransformation in den Zeitbereich nicht nötig.

Die Übertragungsfunktion eines Totzeitglieds wird beschrieben durch:

$$\frac{G(s)}{y(s)} = K \cdot e^{-T_T s} \tag{6.5}$$

Die Konstante '$T_T$' beschreibt die Totzeit. Aus Abbildung 6.3 wird eine Totzeit von 14,7 Sekunden abgelesen.

Der Parameter 'K' beschreibt die verzögerte Größe. Für den Fall der Regelstrecke wird für 'K' die eigentliche Übertragungsfunktion eingesetzt.

### IT1-Glied

Das IT1-Glied beschreibt ein integrales Übertragungsverhalten mit einer Verzögerung erster Ordnung. Die Übertragungsfunktion einer IT-1 Strecke lautet:

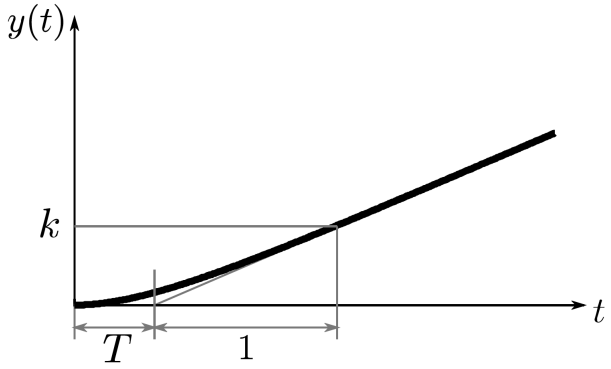


Abbildung 6.4: Sprungantwort eines IT1-Glieds

$$\frac{G(s)}{y(s)} = \frac{k}{s(Ts+1)} \tag{6.6}$$

Der Parameter 'k' drückt die Steigung der Regelgröße aus.

Aus Abbildung 6.3 berechnet sich die Steigung zu:

$$k = \frac{\Delta T}{\Delta t} = \frac{25°C}{66,6sek} = 0,375\frac{°C}{sek} \tag{6.7}$$

### Übertragungsfunktion der Boiler-Regelstrecke

Die gefundenen Parameter werden nun in die Gleichungen 6.5 und 6.6 eingesetzt. Die Übertragungsfunktion der Boiler-Regelstrecke ergibt sich zu:

$$\frac{G(s)}{y(s)} = \frac{0,375}{s(18,7s+1)} \cdot e^{-14,7s} \tag{6.8}$$

### 6.2.2 Einstellung der Regelparameter des Boiler-Reglers

Mithilfe eines Simulink-Modells sollen nun die Regler-Parameter eingestellt werden, bis die Streckenantwort dem gewünschten Verhalten entspricht.

#### Definieren der Übertragungsfunktion in MATLAB®

Für die Simulation des Regelkreises muss die Übertragungsfunktion im MATLAB®-Workspace definiert sein. Dies kann mit folgendem Code durchgeführt werden:

Listing 6.1: Definieren der Boiler-Strecken-Übertragungsfunktion in MATLAB®

```
%% IT1-TF definieren
k = 0.375;                  % Verstaerkung
T = 18.7;                   % Zeitkonstante
T_t = 14.7;                 % Totzeit
tf_IT1 = tf(k, [T, 1, 0]);
set(tf_IT1, 'InputDelay', T_t);
```

#### Simulink-Modell des Boiler-Regelkreises

In Simulink wird ein Regelkreis-Modell aufgebaut, welches dem Schema aus Abbildung 6.1 entspricht.
Da die gefundene Übertragungsfunktion als Ausgang nicht die erreichte Temperatur, sondern eine Temperaturdifferenz in Abhängigkeit der Stellgrößendauer ausgibt, muss die Solldifferenz vor dem Vergleichsglied in der Simulation erst berechnet werden.
Das Simulink-Modell ist in Abb. 6.5 dargestellt.

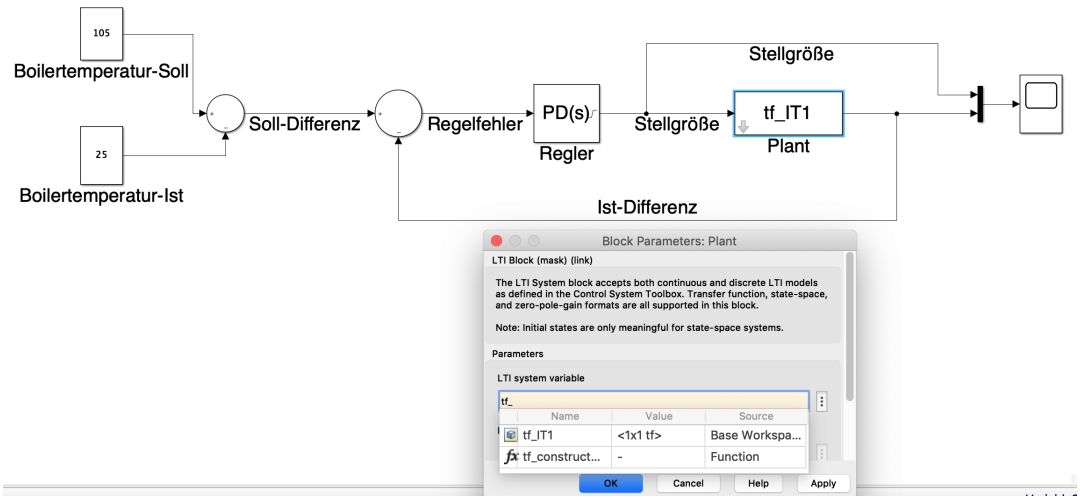


Abbildung 6.5: Simulink-Modell des Boiler-Regelkreises

### Regler-Parameter

Nach [1] kann die Anpassung der Streckenantwort qualitativ mithilfe der folgenden Regelstrategien beeinflusst werden:

**1. Je größer die Regeldifferenz, desto größer die Gegenreaktion - Proportional-Faktor $K_P$**
Dies ist zutreffend auf die Boilerregelung. Je größer die Differenz aus benötigter Temperaturdifferenz und aktueller Temperaturdifferenz ist, um die Boilertemperatur auf den Sollwert anzuheben, desto *länger* soll die Heizung mit voller Leistung betrieben werden.

**2. Je länger die Regeldifferenz vorliegt, desto größer die Gegenreaktion - Integral-Faktor $K_I$**
Diese Strategie trifft nicht auf die Boilerregelung zu. Wie lange eine Regeldifferenz vorliegt, hat keinen Einfluss auf die Betriebsdauer der Heizung, um die Regeldifferenz auszugleichen. Daher ist der I-Teil des Reglers 0.

**3. Je schneller die Änderung der Regeldifferenz erfolgt, desto größer die Gegenreaktion - Derivativ-Faktor $K_D$.**
Die Strategie trifft ebenfalls aus die Boilerregelung zu. Allerdings entspricht die umgekehrte Formulierung eher dem Anwendungsfall: je langsamer die Änderung des Regelfehlers, desto niedriger soll die Gegenreaktion - in diesem Fall der Tastgrad der PWM - sein. Der D-Teil des Reglers hat vor Allem bei kleinen Regeldifferenzen einen Einfluss auf das Streckenverhalten.

Mithilfe der Control System Toolbox ist es in Simulink möglich, die Regelparameter automatisch für das beste Regelverhalten einzustellen. Hierfür wird im PID (PD)-Block des Modells der Button 'tune' betätigt (siehe Abb. 6.6).
Da in der Übertragungsfunktion der Boilerregelstrecke die Temperatur-Trägheit des Systems nicht berücksichtigt wird, kann diese Möglichkeit für Boilerregelung nicht wahrgenommen werden.
Simulink geht anhand des definierten Übertragungsverhaltens davon aus, dass die Stellgröße des Regelsystems auch Werte größer 1 annehmen kann. Mit dieser Annahme wird das Systemverhalten schneller, als es in der Realität möglich ist.
Weiterhin enthält die Übertragungsfunktion keinerlei Information darüber, ob sich das Systemverhalten reversibel ist. In Abb. 6.6 ist zu sehen, dass Simulink davon ausgeht, dass sich das Wasser mit der gleichen Rate abkühlt, mit der es sich aufheizt.

Es zeigt sich, dass die gefundene Übertragungsfunktion das reale Systemverhalten nur bedingt beschreibt. Aus diesem Grund werden die Regelparameter zunächst anhand der oben aufgezählten Regelstrategien und Erkenntnissen aus Streckenmessungen abgeschätzt. Die Werte werden anschließend zur Validierung in Regelkreismodell eingegeben. Das

Systemverhalten ist in 6.6 zu sehen (gestrichelt).

### Abschätzung der Regelparameter

Es werden zunächst Abschätzungen der Parameter vorgenommen, wenn die Pumpe außer Betrieb ist.

Der Proportional-Wert wird in der Reglerformel mit dem Wert des Regelfehlers multipliziert.

$$Stellsignal = P \cdot Regelfehler \tag{6.9}$$

$$P = \frac{Stellsignal}{Regelfehler} \tag{6.10}$$

Der P-Faktor gibt also an, bei welchem maximalen Regelfehler das Stellsignal '1' sein soll, also die Heizung über einen Zeitraum von 10 Sekunden durchgehend betrieben werden kann, ohne dass ein Überschwingen erfolgt. Dieser maximale Regelfehler wird nach verschiedenen Messungen auf 20 geschätzt. Der P-Faktor ergibt sich damit zu 0,05.

Der D-Faktor wird in der Reglerformel mit der Änderungsgeschwindigkeit des Regelfehlers multipliziert.

$$Stellsignal = D \cdot \frac{d_{Regelfehler}}{dt} \tag{6.11}$$

$$D = \frac{Stellsignal \cdot dt}{d_{Regelfehler}} \tag{6.12}$$

Der D-Teil ist dafür verantwortlich, die negative Änderungsrate beim Abkühlen des Wassers abzufangen. Er hat einen großen Einfluss, wenn der Regelfehler klein ist, und die Änderungsgeschwindigkeit groß. Um einen ersten Schätzwert zu erlangen, wird die Heizung mit der PWM betrieben, während die sich einstellende Änderungsrate der Wassertemperatur aufgenommen wird. In jeder Messung wird der Tastgrad der PWM erhöht.
Bei einem Stellsignal von 0,8 wird die Heizung 80% des definierten Zeitraums (10 Sekunden) eingeschaltet. Den Rest der Zeit bleibt sie ausgeschaltet.
Um den D-Faktor zu berechnen, werden die sich einstellenden Änderungsgeschwindigkeiten durch das jeweilige Stellsignal dividiert.
Die Messungen ergeben einen durchschnittlichen Faktor von 0,32. Der Wert wird auf D = 0,4 aufgerundet.

### Regelparameter für Strecke ohne Störgröße

$$K_P = 0.05$$
$$K_I = 0$$
$$K_D = 0.4$$

Die reale Erprobung mit diesen Werten zeigt, dass der Sollwert zwar ohne ein großes Überschwingen erreicht wird, anschließend nach längerer Dauer jedoch ein Regelfehler von ca. 1,5 verbleibt.
Der P-Teil wird daher in folgenden Versuchen so lange erhöht, bis der bleibende Regelfehler verschwindet. Das Überschwingen wird zunächst nicht beachtet.
Es ergibt sich ein neuer P-Faktor von 0,25.

Um das Verhalten bei Pumpenbetrieb zu untersuchen, wird zunächst keine Änderung der Parameter vorgenommen, sondern nur der Einfluss der Störgröße untersucht. Es zeigt sich, dass der Einfluss der Pumpe erst nach sehr langer Betriebsdauer einstellt. Dies ist nachvollziehbar, da bei dem hohen Gegendruck, der sich durch den Widerstand des Siebträgers (simuliert durch Drosselventil am Auslass) einstellt, nur ein geringes kaltes Volumen in den Boiler eintritt. Das von unten kommende kalte Wasser benötigt eine gewisse Zeit, bis es von dem Temperatursensor, der sich oben am Boiler befindet, erkannt wird.
Um das Absinken der Temperatur, das sich erst nach dem Ausschalten der Pumpe einstellt, zu verhindern, wird die Heizung bei Einschalten der Pumpe manuell - also ohne Berechnung - für 10 Sekunden eingeschaltet.
Da sich während des Pumpenbetriebs nur geringe Temperaturgradienten einstellen, und dementsprechend geringe Regelfehler entstehen, kann der P-Faktor während des Betriebs auf null gesetzt werden. Der D-Teil wird entsprechend erhöht.

Nach verschiedenen Versuchen werden für den Pumpenbetrieb folgende Parameter festgelegt.

**Regelparameter für Strecke mit Störgröße**

$$K_{P,Stör} = 0$$
$$K_{I,Stör} = 0$$
$$K_{D,Stör} = 15$$

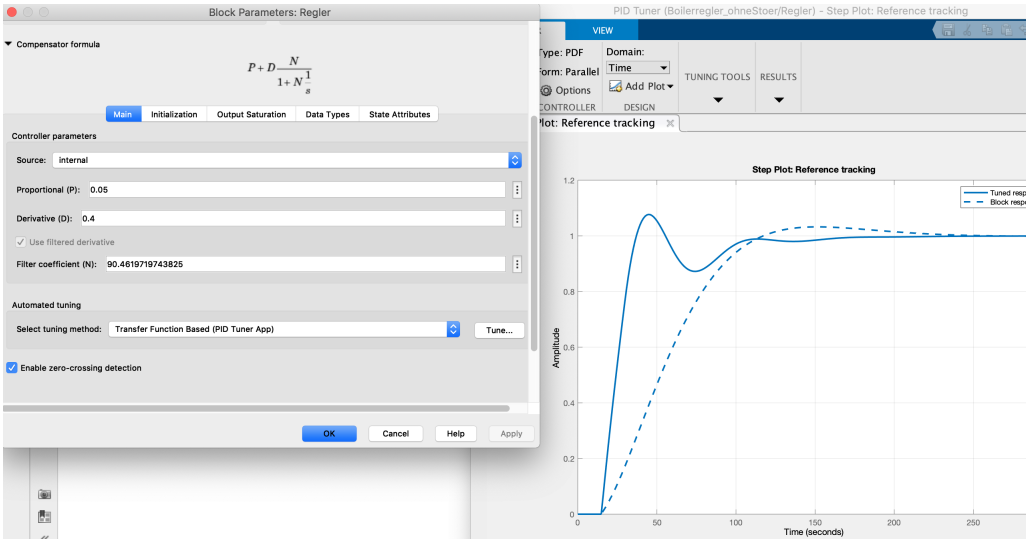


Abbildung 6.6: Simulink PID-Tuner

## 6.3 Implementierung des Boiler-Reglers in die Code-Routine

Zuerst müssen der Regelfehler und dessen Ableitung berechnet werden. Der Vollständigkeit halber wird außerdem das Integral des Regelfehlers berechnet, obwohl es aufgrund des nicht vorhandenen I-Anteils zunächst nicht benötigt wird.
Anschließend wird mithilfe der Regelformel des PID (PD) Reglers das Stellsignal berechnet.

### 6.3.1 Berechnung der Eingangsgrößen des Boilerreglers

#### Berechnung des Regelfehlers

Der Regelfehler berechnet sich aus der Differenz von Soll- und Isttemperatur des Boilerwassers. Der Verlauf wird in werte_f_csv[6] gespeichert.

Listing 6.2: Berechnung des Boiler-Regelfehlers

```
1 werte_f_csv[6] = T_Soll_boiler - werte_f_csv[4] # Boiler- Regelfehler
```

#### Integral-Berechnung des Regelfehlers

Die numerische Berechnung des Regelfehler-Integrals wird mit dem Rechteckverfahren angenähert. Zum Zeitpunkt 0 wird die Startzeit sowie der aktuelle Regelfehler in einer Variable gespeichert. Die Startwerte werden neu belegt, wenn das flag 'flag_int_misch_fehler' = 0 ist.
Wenn die Differenz aus aktueller Zeit abzüglich der Startzeit die Schrittweite (dt_int = 0,1) übersteigt, wird zunächst die reale, vergangene Zeit bestimmt ('dt_i_real'). Anschließend wird der y-Wert aus dem aktuellen Regelfehler, addiert um die Hälfte der Differenz aus aktuellem und vorherigen Wert des Regelfehlers, gebildet. Dieser y-Wert wird dann mit dem vorher berechneten Zeitintervall ('dt_i_real') multipliziert. Die Summe aus der Multiplikation und dem alten Integralwert ergibt den neuen Integralwert des Regelfehlers. Um die Startwerte neu zu belegen, wird das flag auf 0 zurückgesetzt.
Der Wert des Integrals wird in 'werte_f_csv[8]' gespeichert.

Der Integralwert muss in regelmäßigen Abständen zurückgesetzt werden. Die Zeit ist frei wählbar und wird zunächst auf 20 Sekunden festgelegt.

Listing 6.3: Berechnung des Integrals des Boiler-Regelfehlers

```
1 if flag_int_boiler_fehler == 0:
2         x_b_0 = werte_f_csv[0]  # Startzeit
3         y_b_0 = werte_f_csv[6]  # Starttemperatur (Regelfehler)
4         flag_int_boiler_fehler = 1
5
6 if abs(werte_f_csv[0] - x_b_0) >= float(dt_int):
```

```
7           dt_b_real = werte_f_csv[0] - x_b_0
8           y_b_mittel = werte_f_csv[6] + 0.5*(y_b_0 - werte_f_csv[6])
9           int_boiler_fehler = int_boiler_fehler + (y_b_mittel * dt_b_real
                )
10          flag_int_boiler_fehler = 0
11
12  if flag_zeitintboilerfehler == 0:
13          zeit_intboilerfehler = werte_f_csv[0]
14          flag_zeitintboilerfehler = 1
15  if flag_zeitintboilerfehler == 1:
16          if werte_f_csv[0] - zeit_intboilerfehler >= float(20):
17                  int_boiler_fehler = 0
18           zeit_intboilerfehler = 0
19                  flag_zeitintboilerfehler = 0
```

#### Ableitungsberechnung des Regelfehlers

Die Berechnung der Regelfehler-Ableitung wird gemäß Listing 6.4 durchgeführt. Zum Zeitpunkt 0 werden jeweils die Startwerte für die Zeit und den Regelfehler in einer Variable gespeichert. Sobald die Differenz aus aktueller Zeit und Startzeit größer als der Ableitungszeitraum (1 Sekunde gewählt) ist, wird die Ableitung gemäß untenstehender Formel gebildet. Die Ableitung des Regelfehlers wird in 'werte_f_csv[7]' gespeichert.

$$\dot{x}(t) = \frac{x(t-1) - x(t)}{t(t-1) - t(t)} \tag{6.13}$$

Listing 6.4: Berechnung der Ableitung des Boiler-Regelfehlers

```
1   if flag_der_boiler_fehler == 0:
2           x0_boiler_fehler = werte_f_csv[0]     # Startzeit
3           y0_boiler_fehler = werte_f_csv[6]     # Starttemperatur (
                Regelfehler)
4           flag_der_boiler_fehler=1
5
6   # 1 Sekunde spaeter Ableitung berechnen und Flag zuruecksetzen
7   if werte_f_csv[0] - x0_boiler_fehler >= float(1):
8           x1_boiler_fehler = werte_f_csv[0]
9           y1_boiler_fehler = werte_f_csv[6]
10          dy_dx_boiler_fehler = (y1_boiler_fehler - y0_boiler_fehler)/(
                x1_boiler_fehler - x0_boiler_fehler)
11          flag_der_boiler_fehler = 0
```

### 6.3.2 Berechnung der Stellgröße mithilfe der Reglerformel

Der Wert der Stellgröße gibt an, ob die Heizung zum gegebenen Zeitpunkt konstant betrieben wird, oder mithilfe einer programmierten Pulsweitenmodulation reguliert wird,

indem sie ein und ausgeschaltet wird.

In Abschnitt 6.2.2 hat sich gezeigt, dass der Sollwert mit einem kleineren P-Faktor ohne Überschwingen erreicht wird, nach längerer Zeit aber ein konstanter Regelfehler verbleibt. Mit einem erhöhten Faktor wird dieser verhindert, wobei es gleichzeitig zu einem hohen Überschwingen kommt.
Aus diesem Grund wird ein Regelfehler-abhängiger Wechsel der Regelparameter eingeführt. Sobald der Regelfehler kleiner als 1,5 wird, wird der höhere P-Faktor zur Stellwertberechnung verwendet.

Während die Pumpe ausgeschaltet ist, wird die Stellgröße mit den bekannten Faktoren berechnet.
Wenn die Pumpe eingeschaltet ist, muss zunächst dazwischen unterschieden werden, ob sie im aktuellen Schleifendurchlauf gerade erst eingeschaltet wurde - dann ist 'flag_pumpenstart = 0' , oder bereits in Betrieb war - dann ist 'flag_pumpenstart = 1' (siehe Pumpensteuerung 7.3.1).
Wenn die Pumpe bereits lief, wird die Stellgröße anhand der definierten Störgrößen-Parameter berechnet.
Wenn die Pumpe erstmals eingeschaltet wird, soll die Heizung für die konstante definierte Zeit betrieben werden. Hierfür wird die Startzeit der Pumpen-Inbetriebnahme gespeichert. Für den Fall, dass zu diesem Zeitpunkt die PWM aktiv ist, muss das 'pwm_flag' ausgeschaltet werden. Die 'heiz_leistung' wird auf '1' gesetzt.

Anschließend folgt die Ausführung des Stellsignals. Solange das berechnete Stellsignal größer als 0,1 ist, wird die Heizung einschaltet. Wenn der berechnete Wert kleiner ist, soll die Heizung ausgeschalten werden.
Wenn das berechnete Stellsignal kleiner als 1 ist, wird das 'pwm_flag' auf 1 gesetzt, um den Programmteil der PWM zu aktivieren.

Der letzte Programmteil enthält die Heizungssteuerung bei aktiver PWM. Zum Start der PWM wird die Startzeit gespeichert. Die Dauer, während der die Heizung betrieben werden soll, wird aus dem Tastgrad (Stellsignal) multipliziert mit der Periodendauer (10 Sekunden) berechnet. Nachdem die berechnete Zeit vergangen ist, wird die Heizung ausgeschaltet.
Damit die Heizung im nächsten Schleifendurchlauf nicht direkt wieder eingeschaltet wird, bevor die aktuelle Periodendauer abgeschlossen ist, wird das 'flag_warten_regler' auf 1 gesetzt. Nachdem die Periodendauer des aktuellen PWM-Durchlaufs vorüber ist, wird dieses flag auf null zurückgesetzt.

Listing 6.5: Berechnung der Stellgröße

```
1           if flag_Parameterwechsel_boiler == 0:
2                   if werte_f_csv[6] <= float(1.5):
3                           P_boiler = 0.25
4                           I_boiler = 0
```

```
5                       D_boiler = 0.5
6                       flag_Parameterwechsel_boiler = 1
7
8           ### Pumpe ###
9               # Wenn Pumpe aus #
10          if PORTB < float(128):
11                  heiz_leistung = P_boiler * werte_f_csv[6] + I_boiler *
                        werte_f_csv[8] + D_boiler * werte_f_csv[7]
12                  flag_pumpestart = 0
13
14              # Wenn Pumpe ein #
15          else:
16                  if flag_pumpestart == 1:
17                          if werte_f_csv[0] - zeit_pumpestart >= float(
                                heizzeit):
18                                  heiz_leistung = P_boiler_stoer *
                                        werte_f_csv[6] + I_boiler_stoer *
                                        werte_f_csv[8] + D_boiler_stoer *
                                        werte_f_csv[7]
19                                  if heiz_leistung < 0:           #
                                        Beschraenkung, damit keine negativen
                                         Werte auftreten #
20                                          heiz_leistung = 0
21                                  zeit_pumpestart = 0
22
23                  else:
24                          zeit_pumpestart = werte_f_csv[0]
25                          pwm_flag = 0
26                          flag_warten_regler = 0
27                          heiz_leistung = 1 #P_boiler_stoer * werte_f_csv
                                [6] + I_boiler_stoer * werte_f_csv[8] +
                                D_boiler_stoer * werte_f_csv[7]
28
29
30          if flag_warten_regler == 0:
31                  # Heizung einschalten #
32                  if heiz_leistung > float(0.1):
33                          if PORTB % 2 == 0:
34                              i2c.write_byte_data(0x21, 0x13, PORTB+1)
35                              flag_heizung_ON = 1
36
37                          if heiz_leistung < float(1):
38                              pwm_flag = 1
39
40                  else:
41                          if PORTB % 2 == 1:
```

```
42                              i2c.write_byte_data(0x21, 0x13, PORTB-(
                                    PORTB & heizung_check_an))
43                              flag_heizung_ON = 0
44
45
46          if pwm_flag == 1:
47                  if flag_pwmstart == 0:
48                      pwm_start = werte_f_csv[0]
49                      flag_pwmstart = 1
50                      heizdauer = round(heiz_leistung * periode_pwm)
51                  if werte_f_csv[0] - pwm_start >= heizdauer:
52                      if PORTB % 2 == 1:
53                          i2c.write_byte_data(0x21, 0x13, PORTB-(PORTB &
                                heizung_check_an))
54                          flag_heizung_ON = 0
55                          flag_warten_regler = 1
56                  if werte_f_csv[0] - pwm_start >= periode_pwm:
57                          flag_warten_regler = 0
58                          pwm_flag = 0
59                          flag_pwmstart = 0
```

### 6.3.3 Überhitzungsschutz

Auch wenn an letzter Stelle, ist dieser Teil der Temperaturmessroutine nicht zu vergessen, damit im Falle eines Fehlers nicht aus Versehen die Heizung betrieben wird, während schon ein sehr hoher Dampfdruck im Boiler herrscht.

Listing 6.6: Überhitzungsschutz Boilertemperatur

```
1           if werte_f_csv[4] > float(120):
2                   if PORTB % 2:
3                           i2c.write_byte_data(0x21, 0x13, PORTB-(PORTB &
                                heizung_check_an))
4                           flag_heizung_ON = 0
```

### 6.3.4 Betriebsfähigkeit feststellen

Um festzustellen, wann der Boiler in einem Gleichgewichtszustand angekommen ist, soll überprüft werden, ob sich die Temperatur und Temperaturänderung über einen gewissen Zeitraum stabil verhält.

Listing 6.7: Betriebsbereitschaft Boiler

```
1 # Wenn sich T_boiler und dT/dt_boiler ueber Zeitraum von 5min nicht
      aendert, dann boiler_ready = 1
2 if boiler_check = 1:
```

```
3           if Delta_T_boiler < 0.5:
4                   if abs(dy_dx_boiler) < 0.1:
5                           boiler_check_time = werte_f_csv[0]
6                           boiler_check = 0
7                           boiler_check2 = 1
8   if boiler_check2 = 1
9           if werte_f_csv[0] - boiler_check_time > float(300):
10                  if abs(dy_dx_boiler) < 0.1:
11                          if Delta_T_boiler < 0.5:
12                                  boiler_ready = 1
13                                  boiler_check2 = 0
```

# 7 Mischtemperaturfolgeregelung

Um die Mischwassertemperatur auf den gewünschten Sollwert zu regeln, soll ein Mischtemperaturfolgeregler programmiert werden. Folgeregler bedeutet, dass die Regelung der aktuellen Mischtemperatur folgen soll.
Ziel der Mischregelung ist es, den Sollwert nach einer möglichst kurzen Einregelzeit zu erreichen, und diesen anschließend für eine Bezugszeit von ca. 30 Sekunden möglichst konstant zu halten.
Der zu regelnde Mischtemperaturbereich liegt zwischen 85 und ~~105°C~~. Die Temperatur des Heißwassers entspricht zwischen 110 und 120°C, die des Kaltwassers ca. 20°C.

## 7.1 Mischtemperatur - Regelkreis

Die Regelung soll Führungsverhalten aufweisen. Das bedeutet, dass die Regelungsgröße (Mischtemperatur) der Führungsgröße folgen soll. Die Führungsgröße (Sollwert der Mischtemperatur) wird als zeitlich veränderliche Größe angenommen.

Der erweiterte Regelkreis wird in Abbildung 7.1 dargestellt.

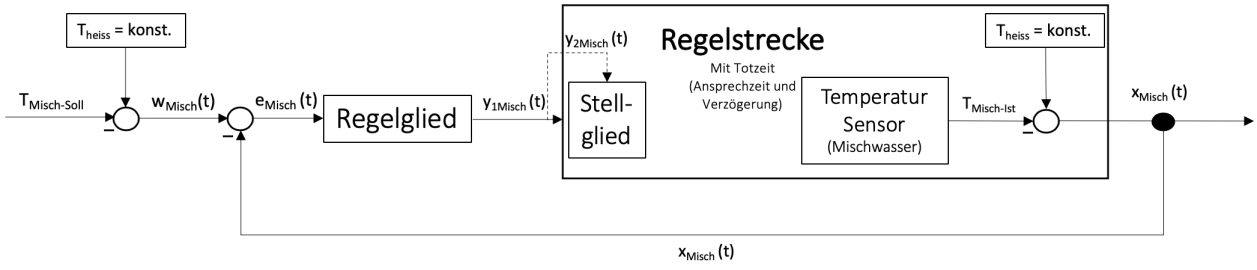


Abbildung 7.1: Erweiterter Regelkreis Mischtemperaturfolgeregler

Der Regelkreis besteht aus dem Regelglied, der Regelstrecke und dem Vergleichsglied, welches Soll und Ist vergleicht, um den Regelfehler zu erhalten. Das Regelglied wird in Form der Regler-Programmierung auf dem RPi in den Regelkreis implementiert. Die Regelstrecke besteht aus zwei Dosierventilen, die den Durchfluss des Heiß- und Kaltwassers regulieren. Da der Durchfluss während des Betriebs konstant bleiben muss, werden die Ventile in Relation zueinander bewegt. Die zwei Dosierventile werden als ein Stellglied zusammengefasst.
Durch die Reglerformel wird die Öffnung des Kaltwasserventils berechnet. Um die Ventile

zu koppeln, wird die Öffnung des Heißwasserventils berechnet, indem die berechnete Kaltwasser-Öffnung (in Prozent) von 100% abgezogen wird.

Die Führungsgröße $w_{Misch}(t)$ des Regelkreises ergibt sich aus der Differenz der Heißwassertemperatur und dem Sollwert der Mischtemperatur. Beides sind zeitlich veränderliche Größen und müssen in der Messroutine in jedem Schleifendurchlauf berechnet werden. Die Einheit der Führungsgröße ist $[°C]$.
Die Regelgröße $x_{Misch}(t)$ ergibt sich aus der Differenz der Heißwassertemperatur und dem Ist-Wert der Mischtemperatur. Auch diese Größen sind zeitlich variabel.
Die Differenz aus Führungsgröße und Regelgröße ergibt den Regelfehler $e_{Misch}(t)$ mit der Einheit $[°C]$. Dieser stellt die Eingangsgröße des Regelglieds dar.

Wie bereits beschrieben, entspricht die Ausgangsgröße des Regelglieds die Öffnung des Dosierventils in Prozent (als Dezimalzahl). Sie kann einen Wert zwischen 0 und 1 annehmen.
Um die Dosierventile zu verstellen, muss der Stellwert in eine Spannung umgerechnet werden.

Beispiel:

$$y_{1Misch}(t) = 0,3 \tag{7.1}$$
$$y_{1Misch}(t)' = 0,3 \cdot 9900mV = 2970mV \tag{7.2}$$

Die Beschaltung des Dosierventils (Stellglied) mit einer Spannung resultiert in einer Änderung der Dosierventilöffnung (symbolisch dargestellt durch $y_{2Misch}(t)$). Da diese Öffnung nicht messbar ist, wird sie nicht weiter betrachtet.

Die Eingangsgröße der Regelstrecke stellt die Ventilstellung (Dezimalzahl) dar. Das Verstellen des Ventils führt zu einer Temperaturänderung des Mischwassers, welche von dem Temperatursensor am Ende der Mischwasser-Strecke gemessen wird. Der Vergleich der Heißwassertemperatur mit der gemessenen Mischtemperatur erfolgt analog zur Führungsgröße bei jedem Schleifendurchlauf des Programmcodes. Die Differenz ergibt die Regelgröße, welche das Ausgangssignal der Strecke darstellt, und als Rückführung den Regelkreis schließt.

Tabelle 7.1: Regelgrößen der Mischregelung

| | | |
|---|---|---|
| $w_{Misch}(t)$ | Führungsgröße | $T_{Heisswasser}$ - $T_{Misch-Soll}$ |
| $x_{Misch}(t)$ | Regelgröße | $T_{Heisswasser}$ - $T_{Mischwasser}$ |
| $e_{Misch}(t)$ | Regelfehler | Führungsgröße - Regelgröße |

## 7.2 Mischtemperatur - Reglerentwurf

Um das Übertragungsverhalten der Dosierventile kennenzulernen, wird zunächst eine Untersuchung der Regelstrecke durchgeführt.
Anschließend wird eine Übertragungsfunktion aufgestellt. Mithilfe einer Simulink Simulation des Regelkreises kann anhand des gefundenen Übertragungsverhalten die Einstellung der Reglerparameter erfolgen.

### 7.2.1 Streckenanalyse der Mischwasser-Regelstrecke

Im Zuge der Streckenanalyse wird das statische und das dynamische Übertragungsverhalten der Dosierventile untersucht. Es werden zwei Messungen durchgeführt, aus denen zum einen die statische Kennlinie, sowie die Übertragungsfunktion aus einer Sprungantwort abgeleitet werden.

Da der Bereich, in dem die Mischtemperatur geregelt werden soll, sehr gering ist, kann das Kaltwasser-Dosierventil als Stellglied der Mischregelung als weit überdimensioniert angesehen werden. Es wird daher zunächst ein Ansatz gewählt, bei dem die Kaltwasser-Zufuhr gedrosselt wird. Hierfür stehen zwei feste Drosseln zur Verfügung, jeweils mit einem Durchmesser von 0,8 und 0,6 [mm](?). Die folgenden Messungen werden jeweils ohne verbaute Drossel, mit verbauter 0,8-, und 0,6-Drossel durchgeführt.

#### Messdaten mithilfe von Python aufnehmen und in MATLAB® analysieren

Die Programmierung des Stellwerts wird in die Messroutine (cOS_MCP3301_CSV_Regler_V3_1.py) eingebunden. (Auswertung siehe Abschnitt 7.2.2)
Zu Beginn der Messungen ist das Dosierventil komplett geschlossen, d.h. der Wert der Stellgröße ist null. Die Aufnahme der Messdaten soll in einem Mischtemperaturbereich von 85 bis 105°C stattfinden. Dazu muss sichergestellt sein, dass die Boilertemperatur dem Sollwert der Boilerregelung entspricht.
Da während der Messung die Pumpe in Betrieb ist, wird dem Boiler Wasser entnommen, wodurch die Heißwassertemperatur nicht konstant bleibt. Daher wird vor der Verstellung zusätzlich geprüft, ob der Gradient der Boilertemperatur kleiner als 0,1 ist.
Das Stellglied soll immer dann um eine Größenordnung erhöht werden, wenn der Gradient der Mischtemperatur kleiner als 0,1 ist, und alle oben genannten Bedingungen über einen Zeitraum von 20 Sekunden erfüllt sind.

Für die Messungen muss eine Größe definiert werden, um die das Stellglied nach jedem Durchlauf erhöht wird. Die Messungen werden jeweils mit einer Verstellung von 1 und 5 % durchgeführt..

### Messung 1

In der ersten Messung wird ein treppenförmiger Anstieg des Stellwerts programmiert. Dabei wird der Wert in jedem Schritt um einen konstanten Faktor erhöht. Eine Verstellung des Ventils findet statt, sobald sich die Mischtemperatur nicht mehr ändert.
Während der Messung werden die Messdaten von Kaltwasser-, Boilerwasser und Mischwasser, sowie die aktuelle Dosierventilstellung in einer csv-Datei (buffer2.csv) abgelegt. Diese Daten werden anschließend in MATLAB® importiert und analysiert.
Um den Einfluss der Schwankung der Heißwassertemperatur auf die Mischwassertemperatur (Boilerwasser) auszugleichen, werden die Abweichungen der Heißwasser-Temperatur von der Temperatur zum Start der Messung abgezogen, sodass sich ein konstanter Verlauf einstellt. Die sich ergebenden Differenzen werden anschließend zu den Messwerten der Mischtemperatur addiert (siehe Abb. 7.2).
Anschließend werden die Mischtemperaturwerte über die Matlabfunktion *filter* geglättet.

Als Beispiel wird in Abbildung 7.3 der Verlauf der ersten Messung mit verbauter

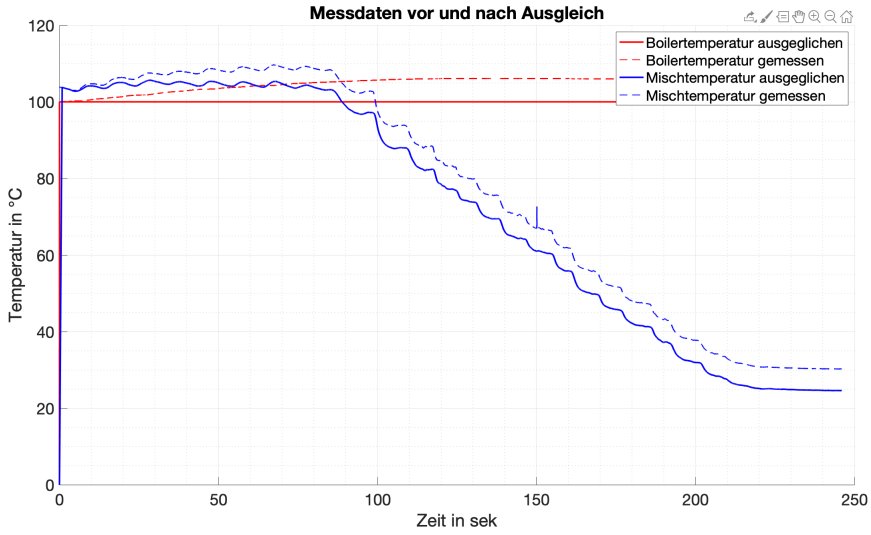


Abbildung 7.2: Ausgeglichene Messdaten der Dosierventil-Messung

0,6-Drossel dargestellt.

### Messung 2

In einer zweiten Messung werden die vorher iterativ angefahrenen Ventilpositionen nun von einer vollständig geschlossenen Position ab angefahren. (cOS_MCP3301_CSV_Regler_V3_2.py)
Die Schrittgröße bleibt gleich. Beispielhaft ist in Abbildung 7.4 der Verlauf der Messung ohne verbaute Drossel abgebildet. Die Diagramme der restlichen Messungen befinden sich im Anhang.

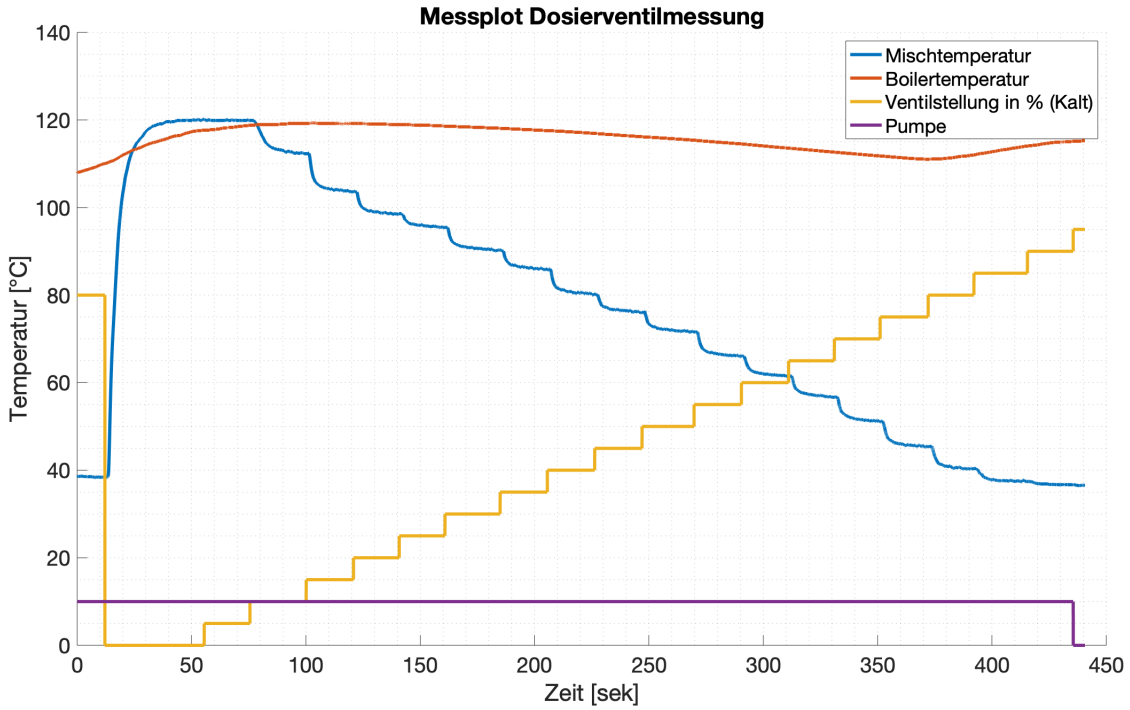


Abbildung 7.3: Streckenantwort auf treppenförmigen Stellsignal-Verlauf (0,6-Drossel)

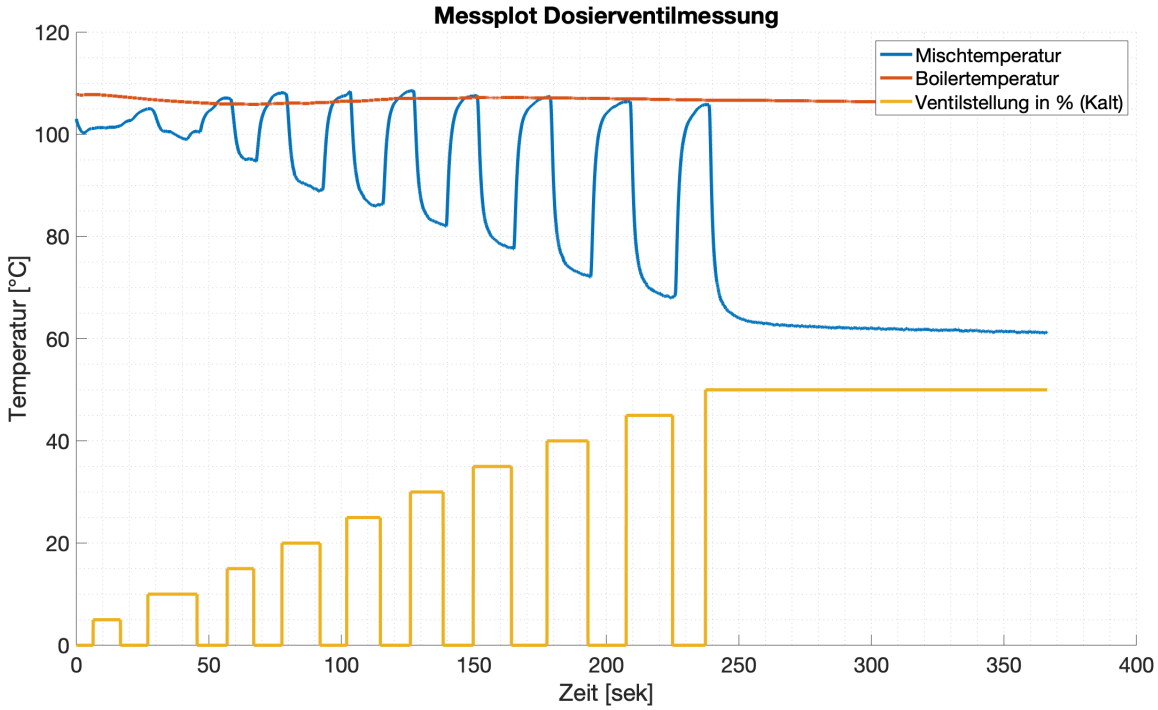


Abbildung 7.4: Streckenantwort auf treppenförmigen Stellsignal-Verlauf ohne Drossel

### 7.2.2 Statisches Übertragungsverhalten der Mischwasser-Regelstrecke

Aus den um die Heißwasserschwankung berichtigten Messdaten der ersten Messung wird nun die statische Kennlinie berechnet.
Aus den letzten zwei Sekunden vor einer Ventilstellungsänderung wird der Mittelwert der Mischtemperatur gebildet, und in einem Array gespeichert. Die Ergebnisse der Temperaturdifferenz werden über der Stellgröße - die Öffnung des Kaltwasserdosierventils in Prozent - in einem Diagramm dargestellt (Abbildung 7.5).
Die statische Kennlinie der absolut angefahrenen Ventilstellungen ändert sich unmerklich.

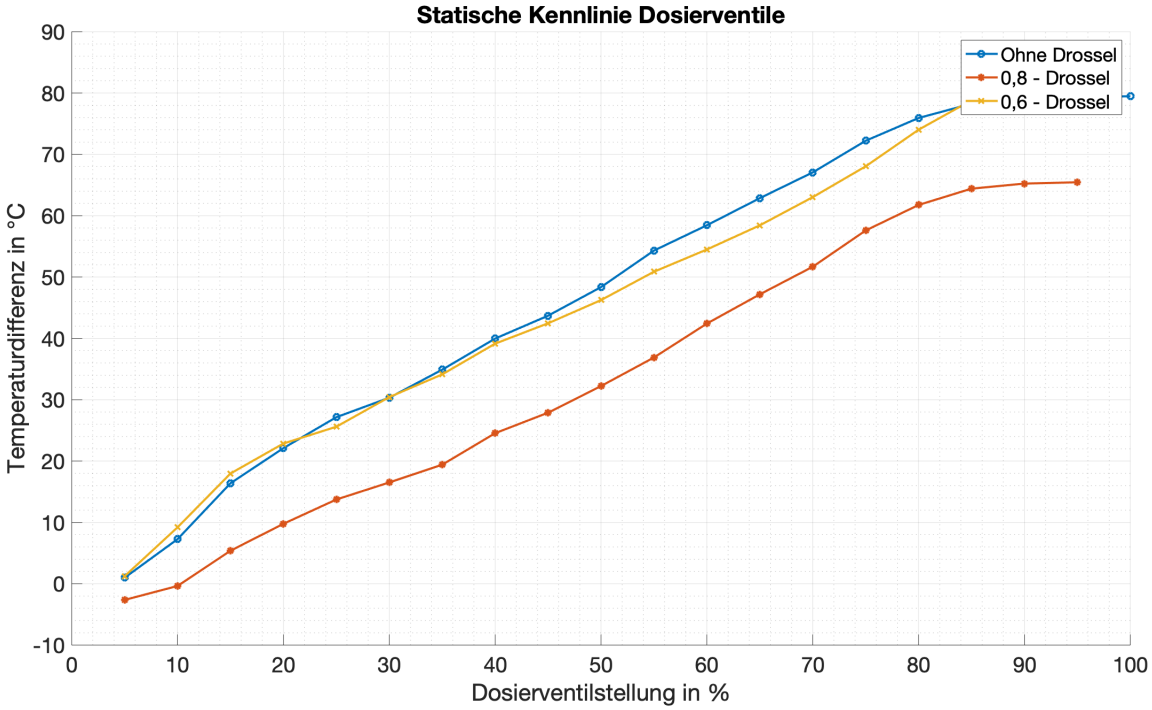


Abbildung 7.5: Statische Kennlinie des Misch-Stellglieds

Die Kennlinie der Messung ohne Drossel (blau) zeigt, dass bereits eine Verstellung des Ventils um 35 % ausreicht, um eine Temperaturdifferenz von 30°C zu erreichen.
Da die Regelung der Mischtemperatur nur in einem Bereich von ca. 0 - 30°C Temperaturdifferenz benötigt wird, wird vor das Kaltwasserdosierventil eine feste Drossel verbaut, um den nutzbaren Regelbereich des Stellglieds zu maximieren. Damit soll eine präzisere Regelung ermöglicht werden. Für den weiteren Verlauf bleibt die 0,6-Drossel verbaut.

### 7.2.3 Dynamisches Übertragungsverhalten der Mischwasser-Regelstrecke

Die Untersuchung des dynamisches Übertragungsverhalten soll das Zeitverhalten der Regelstrecke definieren. Hierfür muss die Sprungantwort einer einzelnen Ventilstellung untersucht werden.
Aus den Messdaten der Messung 2 (mit 0,6-Drossel) werden zunächst für jede Ventilstellung die Sprungantworten dargestellt (siehe Abb. 7.6).

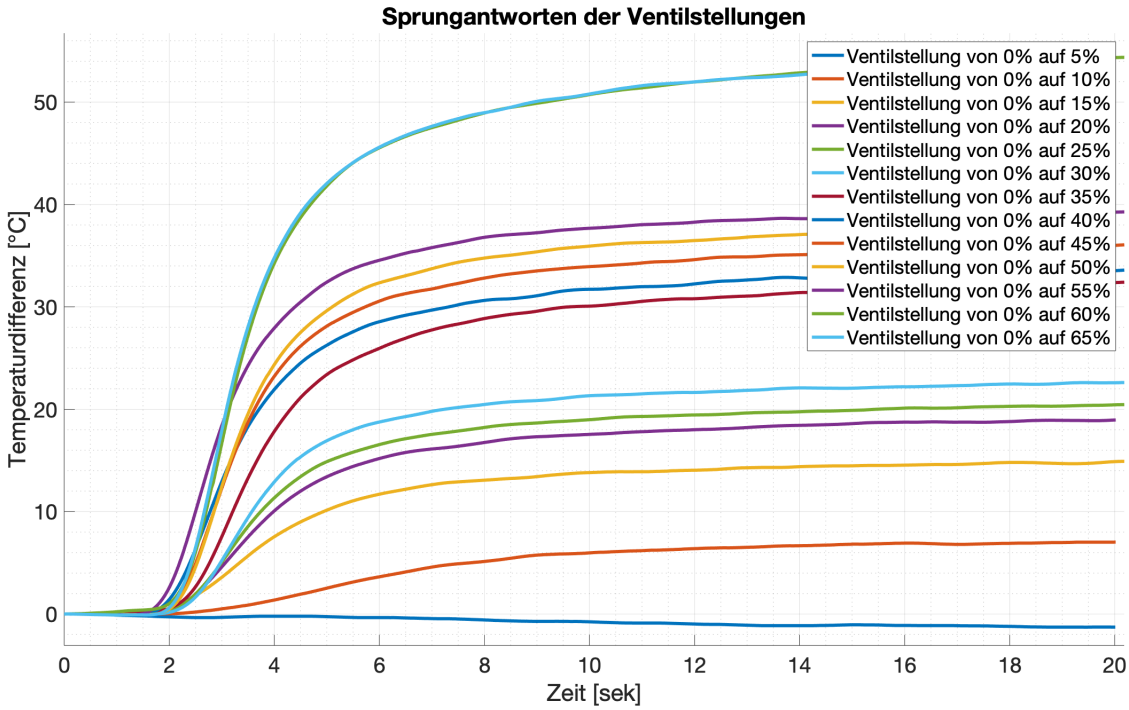


Abbildung 7.6: Sprungantworten verschiedener Ventilstellungen

Es zeigt sich, dass das Kaltwasser-Ventil selbst bei einem gedrosselten Kaltwasserstrom überdimensioniert ist.

Da der zu regelnde Temperaturbereich 40°C nicht übersteigt, wird eine Ventilstellung von 55% als Maximum festgelegt (Stellgröße = 1).

### 7.2.4 Zeitverhalten der Mischwasser-Regelstrecke: PT-1 mit Totzeit

Der Verlauf der Sprungantworten aus Abbildung 7.6 weißt auf ein Verzögerungsglied 1. Ordnung hin ($PT_1$). Die Verzögerung vor dem erstmaligen Temperaturanstieg wird durch ein Totzeitglied beschrieben.
Als Totzeit versteht man in der Regelungstechnik die Zeitspanne zwischen einer Signaländerung am Systemeingang und der Signalantwort am Systemausgang der Regelstrecke. Für den Anwendungsfall dieser Arbeit kann die Totzeit durch die Ansprechzeit des Temperatursensors sowie durch Verzögerung beim Stellen des Dosierventils erklärt werden.

#### PT1 Glied

Die Übertragungsfunktion einer PT1 Strecke im Frequenzbereich wird laut ... wie folgt beschrieben:

$$\frac{G(s)}{y(s)} = \frac{K}{Ts+1} \tag{7.3}$$

Im Zeitbereich lautet die Übertragungsfunktion:

$$\frac{G(t)}{y(t)} = K \cdot (1 - e^{\frac{-t}{T}}) \tag{7.4}$$

wobei
K ... Stationäre Verstärkung
T ... Zeitkonstante
y(s) ... Stellsignal|
t ... Zeit in sek

Die Sprungantwort eines PT1-Glieds sieht laut [...] wie folgt aus: (Abb. 7.7)

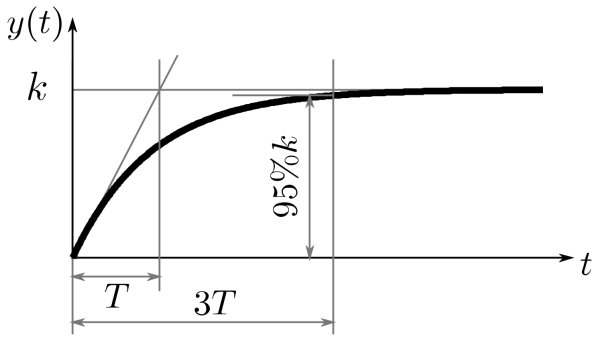


Abbildung 7.7: Sprungantwort eines PT1 Glieds

### Übertragungsfunktion

Das Totzeitglied wird in Abschnitt 6.2.1 beschrieben.
Die Totzeit-Konstante der Misch-Regelstrecke beträgt ca. 2,1 Sekunden.

Die Verstärkung und die Zeitkonstante T können aus der Sprungantwort abgelesen werden (Abbildung 7.6).

Die Übertragungsfunktion der Misch-Regelstrecke ergibt sich zu:

$$G(s) = \frac{38,2}{1,5s+1} e^{-s2,1} \tag{7.5}$$

### Verzögerungsglied n-ter Ordnung: $PT_n$ Glied

Alternativ kann die Regelstrecke durch Reihenschaltung von zwei Verzögerungsgliedern 1. Ordnung beschrieben werden.

Mit einer Verzugszeit (Totzeit) $T_U$ von 2,1 und einer Ausgleichszeit $T_G$ von 1,5 Sekunden lautet die Übertragungsfunktion:

$$\frac{G(s)}{y(s)} = \frac{38,2}{(1+1,73s)(1+1,962s)} \tag{7.6}$$

In Abbildung 7.8 werden neben den Messdaten der 55%-Ventilstellung die Sprungantworten des PT1-Glieds, sowie des PTn-Glieds dargestellt.
Wie zu sehen ist, trifft der Verlauf des PT1-Glieds den Verlauf der Messdaten sehr gut.

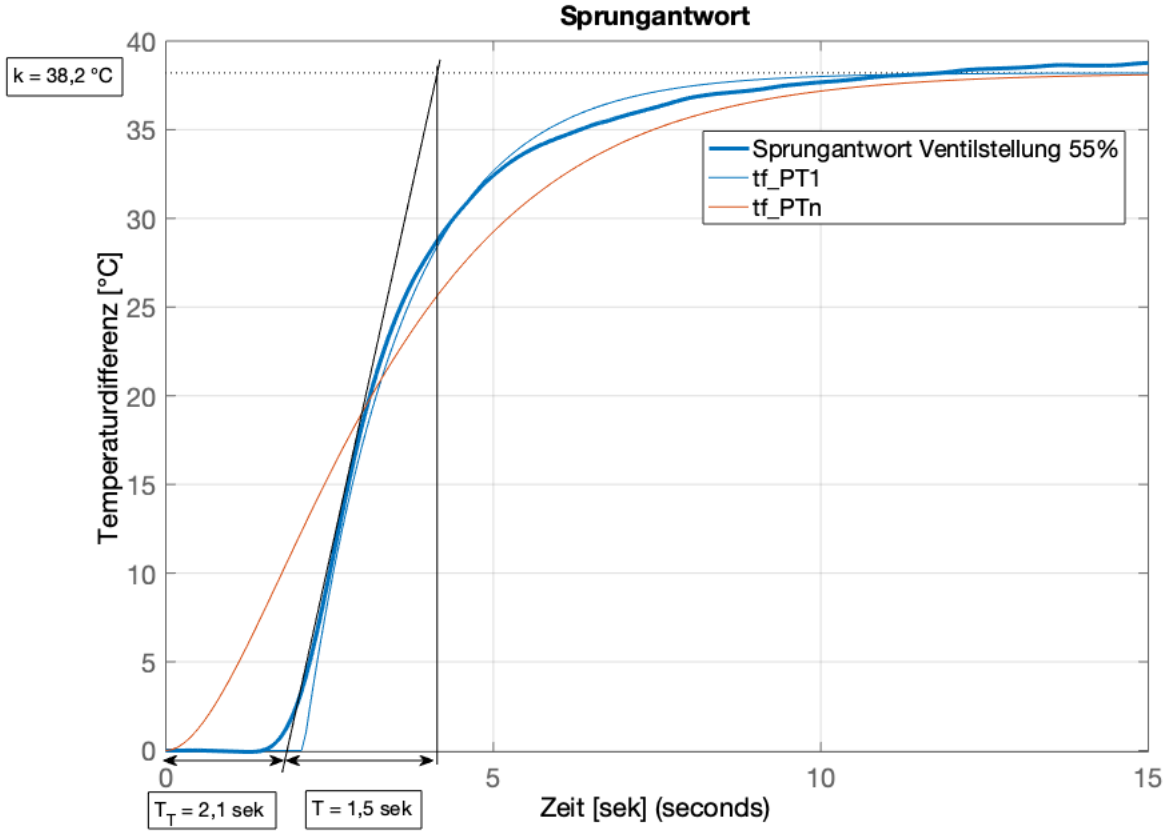


Abbildung 7.8: Sprungantwort der Messdaten, PT1- und PTn-Glied

### 7.2.5 Einstellung der Regelparameter des Mischtemperatur-Reglers

Für den Reglerentwurf wird zunächst ein PID Regler gewählt, mit der Übertragungsfunktion

$$R(s) = K_P \cdot K_I \cdot \frac{1}{s} \cdot K_D \cdot s \tag{7.7}$$

Für die Einstellung der Reglerparameter $K_P$, $K_I$ und $K_D$ wird in Simulink ein Modell des Regelkreises gemäß Abbildung 7.1 aufgebaut (siehe Abb. 7.9). Als Regelglied fungiert der 'PID'-Block.
Die Übertragungsfunktion der Regelstrecke muss im MATLAB® Workspace definiert werden (siehe Listing 7.1). Im Anschluss wird diese im Simulinkblock 'LTI System' hinterlegt wird (siehe Abb. 7.9).

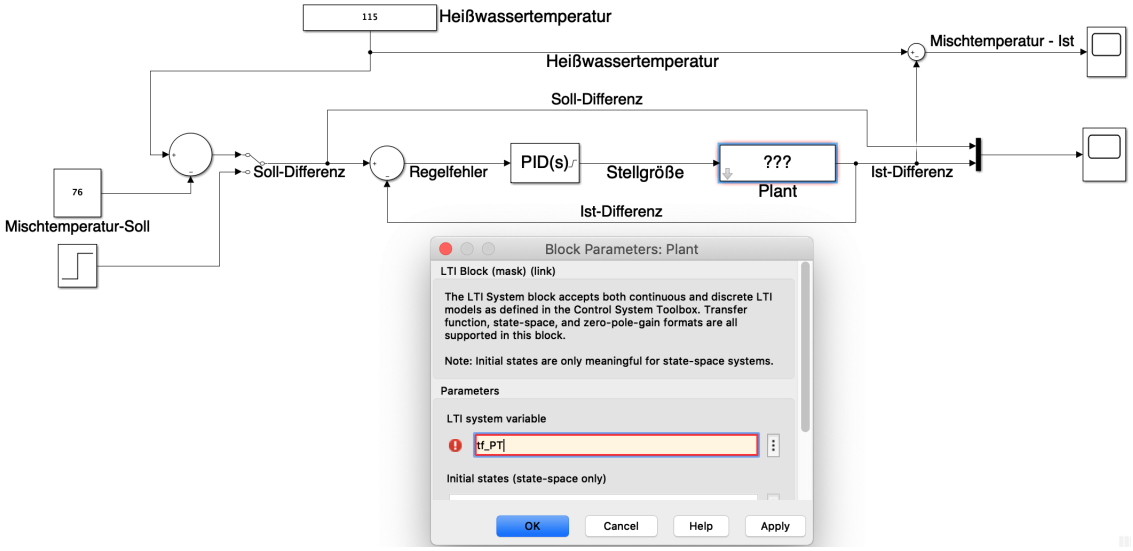


Abbildung 7.9: Simulink Model des Misch-Regelkreis

Listing 7.1: Definieren der Strecken-Übertragungsfunktion in MATLAB®

```
%% PT1-TF definieren
k = 38.2;                   % Verstaerkung
T = 1.5;                    % Zeitkonstante
T_t = 2.1;          % Totzeit
tf_PT1 = tf(k, [T, 1]);
set(tf_PT1, 'InputDelay', T_t);
```

Das Einstellen der Parameter kann mit Simulink automatisiert werden. Hierfür wird in den Einstellungen des PID-Blocks unter 'Automated tuning' der Button 'tune' betätigt. (Control System Toolbox muss installiert sein)
Im PID Tuner Fenster (Abbildung 7.10) kann das Zeitverhalten des Reglers angepasst werden. Entspricht die Sprungantwort dem gewünschten Reglerverhalten, können die

Reglerparameter des PID-Blocks mit Klick auf 'Update Block' automatisch übernommen werden.

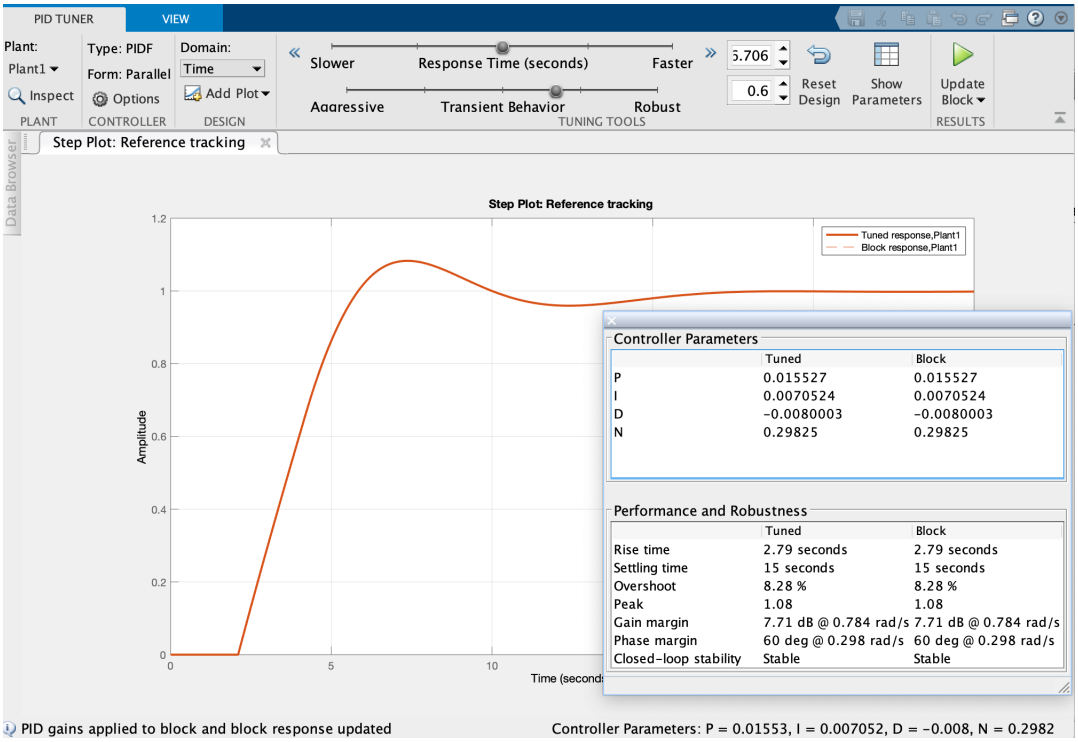


Abbildung 7.10: Simulink PID Tuner

Folgende Werte wurden für die Verstärkungsfaktoren der Regleranteile gefunden:

$$K_P = \mathbf{0{,}01553}$$
$$K_I = \mathbf{0{,}007052}$$
$$K_D = \mathbf{-0{,}008}$$

## 7.3 Implementierung des Mischwasser-Reglers in Code-Routine

Der mit Simulink/MATLAB® definierte Regler muss nun in den Python Code übersetzt werden.
Zunächst müssen die Führungsgröße, die Regelgröße, sowie der Regelfehler berechnet werden. Anschließend werden die Ableitung und das Integral des Regelfehlers berechnet. Alle diese Werte werden in einem Array gespeichert.

Die Reglerformel soll als Ergebnis das Stellsignal ausgeben. Die Einheit des Stellsignals ist zunächst dimensionslos. Der Ergebniswert der Reglerformel liegt zwischen 0 und 1, wobei der Wert '0' der geschlossenen Ventilstellung, und der Wert '1' einer voll geöffneten Ventilstellung entspricht, wobei 'voll geöffnet' seit Abschnitt 7.2.3 einer Öffnung von 55% entspricht.

Die Mischregelung wird aktiviert, wenn die Pumpe in Betrieb ist. Der Betriebsstatus der Pumpe wird analog zum Feststellen des Heizstatus durchgeführt. Aus Tabelle 2.2 ist das Bitmuster bei eingeschalteter Pumpe bekannt. Während die Pumpe betrieben wird, wird zu Auswertungszwecken der Status im Array werte_f_csv vermerkt. Der Regler wird eingeschaltet, indem das 'flag_misch' auf den Wert 2 gesetzt wird.

### 7.3.1 Pumpenabhängige Steuerung

Die Mischregelung soll immer eingeschaltet sein, wenn die Pumpe in Betrieb ist. Wenn dies der Fall ist, sollen sich die beiden Dosierventile in Relation zueinander bewegen, sie werden also synchronisiert.
Wenn die Pumpe außer Betrieb ist, sollen beide Ventile geschlossen sein. Während die Heizung in Betrieb ist, muss jedoch zur Entlüftung eine geringe Öffnung des Heißwasser-Ventils vorhanden sein.
Eine notwendige Startbedingung der Mischregelung ist, dass der Temperatursensor des Mischwassers zum Zeitpunkt des Regelstarts eine Temperatur größer als die Misch-Solltemperatur misst (oder möglichst schnell nach dem Einschalten).
Wenn die Mischtemperatur unter der Misch-Solltemperatur liegt, wird der Regelfehler negativ. Negative Ventilpositionen werden im Code auf 0 begrenzt. Solange der Regelfehler negativ bleibt, bleibt auch das Stellsignal negativ, bzw null. So würde der Regler in eine Schleife geraten.
Um dies zu vermeiden, wird, wenn die Pumpe in Betrieb ist, und gerade zum ersten Mal eingeschaltet wurde, (dann ist 'flag_pumpenstart = 0'), das Kaltwasser-Ventil komplett geschlossen, indem das 'flag_misch' auf den Wert 1 gesetzt wird. Durch die Ventilsynchronisierung öffnet sich dadurch das Heißwasser-Ventil vollständig.
Wie bereits in Abschnitt 7.2.4 festgestellt wurde, liegt zwischen einem Befehl zur Ventilverstellung und einer Reaktion in Form einer Temperaturänderung eine zeitliche Verzögerung.

Daher muss nach dem Öffnen des Heißwasserventils ca. 2-5 Sekunden gewartet werden, bis der Regler, und damit die Berechnung der Regler-Eingangsgrößen, gestartet werden kann.

Der Code der Pumpensteuerung wird in nachstehendem Listing 7.2 dargestellt, ein Flussdiagramm des Codes befindet sich im Anhang.

Listing 7.2: Pumpensteuerung

```
1         if PORTB < float(128):
2                 werte_f_csv[16] = 0
3                 flag_misch = 0
4                 if PORTB % 2:
5                         Oe_heiss = 20
6                         verstell_flag = 1
7                 if flag_pumpestart == 1:
8                         flag_zeit_mischstart = 0
9                         zeit_mischstart = 0
10                        zeit_misch_gesamt = 0
11                        flag_pumpestart = 0
12        else:
13                werte_f_csv[16] = 1
14                if flag_pumpestart == 0:
15                        flag_misch = 1
16                        flag_zeit_mischstart = 0
17                        zeit_mischstart = werte_f_csv[0]
18                        zeit_misch_gesamt = 0
19                        Oe_kalt = 0
20                        verstell_flag = 1
21                        flag_pumpestart = 1
22                if flag_misch == 1:
23                        if werte_f_csv[0] - zeit_mischstart >= float(5)
                              :
24                                flag_misch = 2
```

### 7.3.2 Steuerung der Dosierventile

Die Dosierventile lassen sich variabel in einem Spannungsbereich von 0 bis 9900mV steuern. Da bei aktivierter Mischregelung lediglich ein Wert der Öffnungsposition des Kaltwasserventils berechnet wird, muss die Öffnung des Heißwasser-Dosierventils um einen entsprechenden Wert verringert werden.
Die Spannung des Kaltwasserventils wird berechnet, indem der im Reglercode berechnete Wert mit der Maximalspannung (in Prozent) multipliziert wird. Das Heißwasserventil soll um den Wert geschlossen werden, mit dem das Kaltwasserventil geöffnet wird. Dafür wird zur Heißspannungs-Berechnung der Kaltwert von 100 abgezogen.

Die 'Synchronisierung' der Dosierventile soll nur erfolgen, wenn die Mischregelung aktiv ist. Bei ausgeschalteter Pumpe - und somit inaktiver Mischregelung - bleibt das Kaltwasserventil geschlossen, und es wird nur die Spannung des Heißwasserventils berechnet.

Der Code zum Stellen der Ventile ist in untenstehendem Listing 7.3 dargestellt, ein Flussdiagramm des Codes befindet sich im Anhang.

Listing 7.3: Dosierventil-Code

```
#!/usr/bin/python
import smbus
import time
i2c = smbus.SMBus(1)     # I2C starten
        if flag_misch == 1:
                flag_ventilsynchro = 1
        else:
                flag_ventilsynchro = 0

        if Oe_kalt < float(0):
                Oe_kalt = 0
        if verstell_flag == 1:
                if flag_ventilsynchro == 1:
                        V_kalt = Oe_kalt * 99
                        V_heiss = (100 - Oe_kalt)*99
                else:
                        V_heiss = Oe_heiss * 99
                        V_kalt = 0

                wert_kalt = 1024*int(V_kalt)/10000
                                        # DAC-Wert berechnen
                high_byte_kalt= int(wert_kalt/256)
                                        # High-Byte und Low-Byte
                    berechnen
                low_byte_kalt = int(wert_kalt-high_byte_kalt*256)
                                # High-Byte und Low-Byte berechnen

                wert_heiss = 1024*int(V_heiss)/10000
                                        # DAC-Wert berechnen
                high_byte_heiss= int(wert_heiss/256)
                                        # High-Byte und Low-Byte
                    berechnen
                low_byte_heiss = int(wert_heiss-high_byte_heiss*256)
                                # High-Byte und Low-Byte berechnen

                i2c.write_i2c_block_data(0x58, 0x01, [low_byte_kalt,
                    high_byte_kalt])
```

```
29                  i2c.write_i2c_block_data(0x58, 0x00, [low_byte_heiss,
                        high_byte_heiss])
30                  verstell_flag = 0
```

### 7.3.3 Berechnung der Führungs-, Regelgröße und Regelfehler der Mischwasser-Regelung

Die Führungsgröße berechnet sich aus der Differenz des eingegebenen Sollwerts für die Mischwassertemperatur, und der aktuellen Boilertemperatur. Die Führungsgröße wird in 'werte_f_csv[10]' gespeichert.
Die Regelgröße berechnet sich aus Boilertemperatur abzüglich der Mischwassertemperatur. Sie wird in 'werte_f_csv[11]' abgespeichert.
Anschließend wird der Regelfehler aus der Differenz der Führungs- und Regelgröße gebildet und in 'werte_f_csv[12]' gespeichert.

Listing 7.4: Mischwasserregelung-Führungs- und Regelgröße

```
1 werte_f_csv[10] = werte_f_csv[4] - T_Soll_misch # Misch-
      Fuehrungsgroesse
2 werte_f_csv[11] = werte_f_csv[4] - werte_f_csv[3]        # Misch-
      Regelgroesse
3 werte_f_csv[12] = werte_f_csv[10] - werte_f_csv[11]      # Misch-
      Regelfehler
```

### 7.3.4 Berechnung der Eingangsgrößen des Mischreglers

Die Berechnung der Ableitung und des Integrals des Regelfehlers wird nur durchgeführt, wenn der Regler eingeschaltet ist.

#### Berechnung der Ableitung des Regelfehlers

Die Berechnung der Regelfehler-Ableitung wird analog zu der beschriebenen Weise aus Abschnitt 6.3.1 durchgeführt.
Die Ableitung des Regelfehlers der Mischregelung wird in 'werte_f_csv[13]' gespeichert.

Listing 7.5: Mischwasserregelung- Berechnung der Regelfehler-Ableitung

```
1 # T0 und t0 bestimmen fuer Ableitungsbildung
2 if flag_misch == 2:
3         if flag_der_misch_fehler == 0:
4                 x0_misch_fehler = werte_f_csv[0]     # Startzeit
5                 y0_misch_fehler = werte_f_csv[12]     # Starttemperatur
                      (Regelfehler)
6                 flag_der_misch_fehler=1
7
```

```
# 1 Sekunde spaeter Ableitung berechnen und Flag zuruecksetzen
        if werte_f_csv[0] - x0_misch_fehler >= float(1):
                x1_misch_fehler = werte_f_csv[0]
                y1_misch_fehler = werte_f_csv[12]
                dy_dx_misch_fehler = (y1_misch_fehler - y0_misch_fehler
                    )/(x1_misch_fehler - x0_misch_fehler)
                flag_der_misch_fehler = 0
werte_f_csv[13] = dy_dx_misch_fehler     # Regelfehler-Ableitung
```

### Berechnung des Integrals des Regelfehlers

Die numerische Berechnung des Regelfehler-Integrals wird analog zu der beschriebenen Weise aus Abschnitt 6.3.1 durchgeführt.
Der Wert des Integrals wird in 'werte_f_csv[14]' gespeichert.

Der Integralwert wird zurückgesetzt, wenn der Regler ausgeschaltet wird. Das Löschen des Integralwerts muss auch erfolgen, wenn der Code zum Zeitpunkt des Pumpenabschaltens in der Integralberechnung steht.

Listing 7.6: Mischwasserregelung- Berechnung des Regelfehler-Integral

```
if flag_misch == 2:
        if flag_int_misch_fehler == 0:
                x_m_0 = werte_f_csv[0]          # Startzeit
                y_m_0 = werte_f_csv[12]         # Starttemperatur (
                    Regelfehler)
                flag_int_misch_fehler = 1

        if abs(werte_f_csv[0] - x_m_0) >= float(dt_int_misch):
                dt_m_real = werte_f_csv[0] - x_m_0
                y_m_mittel = werte_f_csv[12] + 0.5*(y_m_0 - werte_f_csv
                    [12])
                int_misch_fehler = int_misch_fehler + (y_m_mittel *
                    dt_m_real)
                flag_int_misch_fehler = 0

        if flag_misch == 0:
                int_misch_fehler = 0

if flag_misch == 0:
        int_misch_fehler = 0
```

### 7.3.5 Berechnung der Stellgröße (Reglerformel) und Ausführung des Stellsignals

Die Mischregelung wird aktiviert, sobald die Pumpe in Betrieb ist. Die Stellgröße der Mischregelung soll berechnet werden, wenn die Boilertemperatur über der Solltemperatur der Mischregelung liegt, und der Mischregler eingeschaltet ist.

Um die Einregelzeit zu bestimmen, wird bei Inbetriebnahme des Reglers die Startzeit in einer Variable gespeichert. Wenn die Differenz aus Soll- und Ist-Temperatur kleiner als die Regel-Toleranz ist, wird die zum Einregeln benötigte Zeit berechnet und ausgegeben. Wenn sich der Regelfehler um mehr als 0,2 ändert, wird die verstrichene Zeit aufaddiert, und erneut ausgegeben.

Da während des Reglerbetriebs dauerhaft ein neues Stellsignal berechnet wird, sollen die Ventile nur verstellt werden, wenn sich das neue Stellsignal im Vergleich zum Vorherigen um mehr als 1% unterscheidet. Daher wird vor der Stellgrößenberechnung der vorherige Wert in einer neuen Variable gespeichert.

Um das Regelverhalten zu beschleunigen, ohne das Überschwingverhältnis zu beeinflussen, wird abhängig von der Größe des Regelfehlers eine Änderung der Regelparameter durchgeführt. Für kleine Regelfehler wird hierfür zunächst der P-Anteil um eine Potenz verringert. Der Grenzwert, ab dem andere Parameter verwendet werden sollen, muss anhand des Regelgrößenverlaufs im eingeschwungenen Zustand abgeschätzt werden. Außerdem wird, ebenfalls in Abhängigkeit des Regelfehlers, die Zeit verändert, die zwischen zwei Ventilverstellungen liegt, sodass im eingeschwungenen Zustand weniger Ventilverstellungen erfolgen.

Anschließend erfolgt die Berechnung der Reglerformel.
Um das Stellsignal zu begrenzen, werden berechnete Werte größer als 1 auf 1 zurückgesetzt. Da, wie oben beschrieben, der alte Positionswert des Ventils vor der Stellsignal-Berechnung gespeichert wird, und später mit dem neuen verglichen wird, muss die Begrenzung von negativen Werten an anderer Stelle erfolgen (Pumpensteuerung).

Im Anschluss wird die für die Ventilverstellung benötigte Variable bestimmt, indem der mit der Reglerformel berechnete Wert mit der in 7.2.3 definierten Maximalstellung des Ventils (in Prozent) multipliziert wird.

Wenn die oben beschriebene Differenz aus altem und neuem Wert größer als 1% ist, wird der Befehl zum Verstellen der Ventile (verstell_flag = 1) ausgegeben. (Alternativ kann der Verstellbefehl nur dann ausgegeben werden, wenn der Regelfehler größer als ein gewisser Grenzwert ist.) Um die Reaktionszeit des Sensors abzuwarten, wird ein Verstellbefehl nur ausgegeben, wenn die vorher definierte Zwischenzeit der Ventilverstellung verstrichen ist.
Um den Warte-Teil der Reaktionszeit überbrücken zu können, wird eine flag-Abfrage

vorangestellt ('flag_verstell_warten'). Wenn die Zeit, die zwischen zwei Ventilverstellungen liegen soll, in Abhängigkeit der Regelfehlergröße geändert wird, muss das genannte flag den Wert '1' annehmen, damit der Warteteil für diesen Schleifendurchlauf ausgesetzt wird.
Der Code des Mischreglers wird in folgendem Listing 7.7 abgebildet. Ein Flussdiagramm des Codes befindet sich im Anhang.

Listing 7.7: Mischwasserregelung- Berechnung der Stellgröße

```
1  if werte_f_csv[4] >= T_Soll_misch:
2          if flag_misch == 1:                # 1 -> Mischregelung AKTIV
3                  # Einregelzeit berechnen #
4           if flag_zeit_mischstart == 0:
5                  zeit_mischstart = werte_f_csv[0]
6                  flag_zeit_mischstart = 1
7                  if flag_zeit_mischstart == 1:
8                          if abs(werte_f_csv[12]) <= float(0.2):
9                          zeit_misch = werte_f_csv[0] - zeit_mischstart
10                         zeit_misch_gesamt = zeit_misch_gesamt +
                               zeit_misch
11                         print 'Einregelzeit Mischregler: ',
                               zeit_misch_gesamt, 'sek'
12                       flag_zeit_mischstart = 2
13                         if flag_zeit_mischstart == 2:
14                                 if abs(werte_f_csv[12]) >= float(0.2):
15                                 flag_zeit_mischstart = 0
16
17                 Oe_kalt_alt = Oe_kalt
18
19         if werte_f_csv[12] < float(0.5):
20                 P_misch = 0.0019183
21              if werte_f_csv[12] < float(0.2):
22                         zwischenzeit_ventil = 2
23                         flag_verstell_warten = 1
24                         else:
25                                 zwischenzeit_ventil = 0.5
26                 else:
27                 P_misch = 0.019183
28                 zwischenzeit_ventil = 0.5
29                 flag_verstell_warten = 1
30
31                 Oe_kalt_regler = P_misch * werte_f_csv[12] + I_misch *
                       werte_f_csv[14] + D_misch * werte_f_csv[13]
32         if Oe_kalt_regler > float(1):
33                 Oe_kalt_regler = 1
34                 Oe_kalt = Oe_kalt_regler * 55;
35
```

```
36          if werte_f_csv[12] > float(10):
37                  P_misch = 0.2
38
39                  if abs(Oe_kalt_alt - Oe_kalt) > float(0.01):    # if abs
                        (werte_f_csv[12]) > float(0.2):
40                          if flag_verstell_warten == 0:
41                          if flag_verstell_start == 0:
42                                                  verstell_start = werte_f_csv[0]
43                                                  flag_verstell_start = 0
44                                  if dosierventil_messung[0] -
                                        verstell_start >= float(
                                        zwischenzeit_ventil):    #
                                        Reaktionszeit abwarten (Totzeit)
45                          verstell_flag = 1
46                          verstell_start = 0
47                                                  flag_verstell_start = 0
48                          else:
49                          verstell_flag = 1
50                          flag_verstell_warten = 0
```

## 7.4 Analyse des Reglerverhaltens

Mit den in Abschnitt 7.2.5 gefundenen Werten ergibt sich folgendes Verhalten:

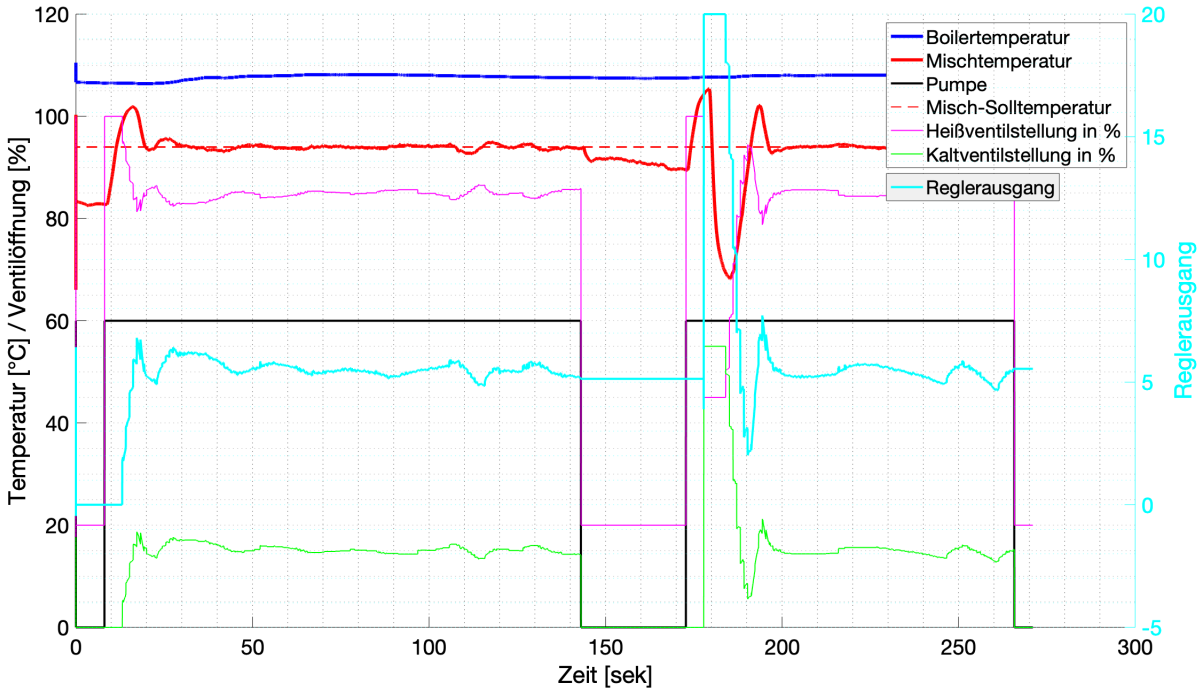


Abbildung 7.11: Reglerverhalten

# 8 Ergebnisse

# 9 Diskussion

## 9.1 Zusammenfassende Bewertung

## 9.2 Ausblick