Fachhochschule Nordwestschweiz
Hochschule für Technik

Reglerdimensionierung mittels Phasengangmethode

Fachbericht

31. Mai 2015

Studiengang	EIT
Modul	Projekt 2
Team	4
Auftraggeber	Peter Niklaus
Fachcoaches	Peter Niklaus, Richard Gut, Pascal Buchschacher, Anita Gertiser
Autoren	Anita Rosenberger, Benjamin Müller, Manuel Suter, Florian Alber, Raphael Frey
Version	`Entwurf`

Abstract

Im Gebiet der Regelungstechnik ist das Dimensionieren von Regler eine zentrale Aufgabe, da mit der korrekten Einstellung der Regler stabil und die Differenz zwischen Ist und Soll-Wert möglichst klein ist.
Die Phasengangmethode ist eine ursprünglich eine graphische Berechnungsart, welche anhand der Schrittantwort die Reglerwerte berechnet. Die Aufgabe der Implementierung dieser Methode in Java war die Hauptaufgabe.

Die Ziel dieses Projektes war, ein benutzerfreundliches Softwaretool, das heisst auch für ein ungeübter Regelungstechniker benutzen kann, zu entwickeln, welches anhand der Phasengangmethode die Dimensionierung eines PI und PID Reglers durchführt. Die Ausgabe des Tools soll anhand der Eingabe der Schrittantwortwerte die numerische wie auch die graphische Lösung ausgeben.

Die Phasengangmethode und die als Vergleich angewendete Faustformeln wurden in Matlab geschrieben und mit Referenzdaten getestet. Die Implementierung in Java war ein zweistufiger Prozess, in welchem zuerst die matlabtypischen Berechungsfunktionen ausprogrammiert und im zweiten Schritt die Regeldimensionierung implementiert wurden.

Das Softwaretool besitzt eine graphische Benutzeroberfläche, über welche auf der linken Seite die Werte der Schrittantwort eingelesen und die numerserischen Lösungen des Reglers ausgegeben und über die rechte Seite die graphischen Lösungen dargestellt werden.

Das Zentrale an der Lösung ist die Berechungungseschwindigkeit mit welcher das Tool arbeitet. Dies ermöglicht eine Ëchtzeit"Dimensionierung des Reglers. Das Neue an dieser Lösung ist das Einbinden der Phasengangmethode in ein Reglerdimensionierungstool.

Projekt P2 - Aufgabenstellung vom Auftraggeber (FS_2015)

Reglerdimensionierung mit Hilfe der Schrittantwort

1. Einleitung

In der Praxis werden die klassischen Regler (PI, PID, PD, …) oft mit sog. Faustformeln dimensioniert. Dazu benötigt man bestimmte Informationen der zu regelnden Strecke. Handelt es sich dabei um „langsame Strecken“ mit Zeitkonstanten im Bereich von Sekunden bis Minuten, so ist das Bestimmen und Ausmessen der Schrittantwort oft die einzige Möglichkeit zur Identifikation der Strecke. Typische Beispiele dafür sind Temperaturheizstrecken, welche meistens mit einem PTn-Verhalten modelliert werden können (Kaffeemaschine, Boiler, Raumheizungen, Lötkolben, Warmluftfön, usw.).
Die Schrittanwort wird mit Hilfe einer Wendetangente vermessen und die Kenngrössen Streckenbeiwert (K_s), Verzugszeit (T_u) und Anstiegszeit (T_g) werden bestimmt. Dies kann sowohl von Hand (grafisch) oder auch automatisiert durchgeführt werden, falls die Messdaten elektronisch vorliegen. Mit diesen drei Kenngrössen können mit Hilfe sog. Faustformeln PI- und PID-Regler dimensioniert werden (Ziegler/Nichols, Chien/Hrones/Reswick, Oppelt, Rosenberg). Die Faustformeln liefern zwar sehr schnell die Reglerdaten, aber die Schrittantworten der entspr. Regelungen sind teilweise weit vom "Optimum" entfernt und der Regelkreis kann sogar instabil werden. In der Praxis muss man diese "Startwerte" häufig nachoptimieren, damit die Schrittantwort der Regelung die Anforderungen erfüllt.

Die sog. "Phasengangmethode zur Reglerdimensionierung" wurde von Jakob Zellweger (FHNW) entwickelt und liefert Reglerdaten, welche näher am "Optimum" sind und für die Praxis direkt verwendet werden können. Dabei kann das Überschwingen der Schrittantwort vorgegeben werden (z.B. 20%, 10%, 2%, oder aperiodisch). Bei dieser Methode kann also das für viele Anwendungen wichtige Verhalten der Schrittantwort beeinflusst werden. Um die Phasengangmethode anwenden zu können, muss der Frequenzgang der Strecke bekannt sein (analytisch oder numerisch gemessen). Mit Hilfe der Hudzovik-Approximation (oder anderer ähnlicher Verfahren) wird dieses Problem gelöst, in dem vorgängig aus den Kenngrössen der Schrittantwort (K_s, T_u,T_g) eine PTn-Approximation der Strecke erzeugt wird. Mit dem Frequenzgang der PTn-Approximation können dann die Regler dimensioniert werden (I, PI, PID). Die Phasengangmethode war ursprünglich eine grafische Methode, basierend auf dem Bodediagramm der Strecke. Aktuell soll die Methode direkt numerisch im Rechner durchgeführt werden.

In dieser Arbeit geht es um die Entwicklung und Realisierung eines Tools zur **Reglerdimensionierung mit der Phasengangmethode**. Ausgehend von der PTn-Schrittantwort der Strecke sollen "optimale Regler" (PI, PID-T1) dimensioniert werden, wobei das Überschwingen der Regelgrösse vorgegeben werden kann. Zum Vergleich sollen die Regler auch mit den üblichen Faustformeln dimensioniert werden. Wünschenswert wäre auch eine Simulation der Schrittantwort des geschlossenen Regelkreises, so dass die Dimensionierung kontrolliert und evtl. noch "verbessert" werden könnte.

2. Aufgaben/Anforderungen an Tool

Entwerfen und realisieren Sie ein benutzerfreundliches Tool/Programm/GUI/usw. mit welchem PI- und PID-Regler mit der Phasengangmethode dimensioniert werden können. Dabei sind folgende Anforderungen und Randbedingungen vorgegeben:

- Die zu regelnden Strecken sind PTn-Strecken, wobei entweder die Schrittantwort grafisch vorliegt oder die Kenngrössen K_s, T_u und T_g schon bekannt sind
- Die Bestimmung einer PTn-Approximation wird vom Auftraggeber zur Verfügung gestellt und muss entsprechend angepasst und eingebunden werden (Matlab zu Java)
- Das Überschwingen der Regelgrösse (Schrittantwort) soll gewählt werden können
- Zum Vergleich sind die Regler auch mit den üblichen Faustformeln zu dimensionieren.
- Das dynamische Verhalten des geschlossenen Regelkreises soll auch berechnet und visualisiert werden (Schrittantwort)

3. Bemerkungen

Die Software und das GUI sind in enger Absprache mit dem Auftraggeber zu entwickeln. Der Auftraggeber steht als Testbenutzer zu Verfügung und soll bei der Evaluation des GUI eingebunden werden. Alle verwendeten Formeln, Algorithmen und Berechnungen sind zu verifizieren, eine vorgängige oder parallele Programmierung in Matlab ist zu empfehlen. Zum Thema der Regelungstechnik und speziell zur Reglerdimensionierung mit der Phasengangmethode werden Fachinputs durchgeführt (Fachcoach).

Literatur

[1] J. Zellweger, *Regelkreise und Regelungen*, Vorlesungsskript.

[2] J. Zellweger, *Phasengang-Methode*, Kapitel aus Vorlesungsskript.

[3] H. Unbehauen, *Regelungstechnik I*, Vieweg Teubner, 2008.

[4] W. Schumacher, W. Leonhard, *Grundlagen der Regelungstechnik*, Vorlesungsskript, TU Braunschweig, 2003.

[5] B. Bate, *PID-Einstellregeln*, Projektbericht, FH Dortmund, 2009.

16.02.2015
Peter Niklaus

Inhaltsverzeichnis

1 Einleitung **7**

2 Grundlagen der Regelungstechnik **8**

2.1 Regelstrecke . . . 8

2.2 Regler . . . 8

2.3 Die Steuerung . . . 9

2.4 Der geschlossene Regelkreis . . . 9

3 Fachlicher Hintergrund zur Regler-Dimensionierung **11**

3.1 Frequenzgang der Regelstrecke . . . 12

3.2 Reglerdimensionierung mittels Faustformeln . . . 13

3.3 Reglerdimensionierung mittels Phasengangmethode: PI-Regler . . . 15

3.4 Reglerdimensionierung mittels Phasengangmethode: PID-Regler . . . 19

3.5 Umrechnung zwischen bodekonformer und reglerkonformer Darstellung . . . 25

4 Software **27**

4.1 View . . . 27

4.2 Controller . . . 27

4.3 Model . . . 27

4.4 Benutzungs-Beispiel (Use-Case) . . . 28

5 Tests **30**

6 Schlussfolgerungen **31**

Appendix **33**

A Beschreibung der Algorithmen **33**

A.1 Sani . . . 33

A.2 Umrechnung von reglerkonformer in bodekonforme Darstellung . . . 34

A.3 Umrechnung von bodekonformer in reglerkonforme Darstellung . . . 35

A.4 utfController . . . 36

A.5 Faustformel Oppelt . . . 37

A.6 Faustformel Rosenberg . . . 38

A.7 Faustformel Ziegler . . . 39

A.8 Faustformel Chien . . . 40

A.9 diskDiff 41
A.10 schrittIfft 42
A.11 overShootOptimisation 43

B Manuelle Berechnung des Hilfsparameteres β **45**

Literaturverzeichnis **47**

Versionsgeschichte

04.05.2015: `Version 0.01`
06.05.2015: `Version 0.02`

1 Einleitung

Im Rahmen des Projektes soll ein Tool entwickelt werden, welches einen PI- respektive einen PID-Regler mittels der von Prof. Jakob Zellweger entwickelten Phasengangmethode dimensioniert. Zum Vergleich soll der entsprechende Regler ebenfalls mittels verschiedenen Faustformeln berechnet werden.

referenz script Zellweger

Die Phasengangmethode ist eine graphische Methode, die bis anhin mit Stift und Papier durchgeführt wurde. Folglich ist die Ausführung zeitaufwändig, speziell wenn Schrittantworten mit unterschiedlichen Parameterwerten durchgespielt werden sollen. Das Tool soll ausgehend von drei Parametern aus der Schrittantwort der Strecke (Verstärkung K_s, Anstiegszeit T_g, Verzögerungszeit T_u) mittels der Phasengangmethode möglichst ideale Regelparameter berechnen sowie die Schrittantwort des darauf basierenden geschlossenen Regelkreises graphisch darstellen. Die Benutzeroberfläche der Software soll intuitiv sein, sodass sich auch mit dem Thema nicht eingehend vertraute Regelungstechniker einfach zurechtfinden.

Die erforderlichen Algorithmen wurden zuerst in Matlab als Prototypen implementiert und anschliessend vollständig in Javakonvertiert. Die graphische Benutzeroberfläche baut ganz auf Java. Um optimale Wartbarkeit, Übersichtlichkeit und Modularität des Codes zu gewährleisten, ist die Software gemäss Model-View-Controllern-Pattern aufgebaut.

mehr/anderer Inhalt?

Nach der Implementierung in Matlab wurde klar, dass die Berechnung durch die hohe Rechenleistung sehr schnell durchgeführt werden kann und somit eine Dimensionierung des geschlossenen Regelkreises anhand dieser Methode von Herrn Zellweger möglich ist.

Der Bericht gliederte sich in zwei Teile: Der ersten Teil erläutert die theoretischen Grundlagen und darauf aufbauend stellt der zweite Teil der Aufbau der Software dar.

2 Grundlagen der Regelungstechnik

2.1 Regelstrecke

In der Regelungstechnik wird die zu regelnde Strecke als Regelstrecke bezeichnet. Die Regelstrecke wird durch ihr Zeitverhalten charakterisiert, welches den Aufwand und die Güte der Regelung bestimmt. Um das Zeitverhalten zu beschreiben, verwendet man die Sprungantwort, welche zeigt, wie die Regelgrösse auf Stellgrössenänderung reagiert. Mit der entstehenden Regelgrösse werden verschiedene Regelstrecken unterschieden:

- P-Regelstrecke
- I-Regelstrecke
- Strecken mit einer Totzeit
- Strecken mit Energiespeicher

Dieses Projekt beschäftigt sich mit den PTn-Strecken, welche eine Kombination einer Strecke mit proportionalen Verhalten und einer mit Totzeit sowie der Angabe der Ordnung n der Strecke sind.

P-Regelstrecke

Bei der Regelstrecke mit proportionalem Verhalten folgt die Regelstrecke proportional der Stellgrösse ohne Verzögerung. Dies kommt in der Praxis nicht vor, da immer eine Verzögerung vorhanden ist. Ist diese jedoch sehr klein spricht man von einer P-Strecke. Der Proportionalitätsfaktor wird mit K_p abgekürzt. Wird $K_p < 1$ wirkt K_p nicht mehr verstärkend sondern abschwächend.

Bild Block-schaltbild P-Strecke

Strecken mit Totzeit

Ändert sich die Stellgrösse, wirkt sich diese Änderung bei einer Strecke mit Totzeit erst nach einer gewissen Zeit auf die Regelgrösse aus. Mit T_t wird das Mass der Totzeit gekennzeichnet.

Totzeiten verursachen schnell Schwingungen, da sich die Stellgrösseänderung zeitverzögert auf die Regelgrösse auswirkt. Die Schwingungen entstehen wenn sich die Stellgrösse und die Regelgrösse periodisch ändern.

Bild Block-schaltbild Totzeit

I-Regelstrecke

Die I-Regelstrecke antwortet auf eine Stellgrössenänderung mit einer fortwährenden Änderung in steigende oder fallende Richtung. Die Begrenzung dieses Vorganges ist mit den systemgegeben Schranken gegeben. Die Integrierzeit T_i ist ein Mass für die Anstiegsgeschwindigkeit der Regelgrösse.

Bild Block-schaltbild I-Strecke

2.2 Regler

Die Aufgabe eines Reglers besteht die zu regelnde Strecke mit einem Stellsignal so zu beeinflussen, dass der Wert der Regelgrösse gleich dem Wert der Führungsgrösse entspricht. Der Regler besteht aus einem Vergleichsglied, welches die Reglerdifferenz aus der Differenz zwischen Führungs- und Reglergrösse bildet und dem Reglerglied. Das Reglerglied erzeugt aus der Reglerdifferenz die Stellgrösse.

Es wird zwischen P-, I- und D-Regler unterschieden. In diesem Projekt werden die PI- und PID-Regler, welche Kombinationen der oben genannten Regler sind, behandelt.

PI-Regler

Der PI-Regler besteht aus einer Parallelschaltung eines P- und eines I-Reglers. Durch diese Kombination werden die Nachteile beider Regler kompensiert und die Vorteile (schnell, stabil) hervorgehoben.

Bild Blockschaltbild PI-Regler

PID-Regler

Wird dem PI-Regler ein D-Anteil parallel geschaltet, entsteht der PID-Regler. Der PID-Regler ist ein sehr oft verwendeter Regler, da durch den D-Anteil die Regelgrösse rascher den Sollwert erreicht und der Einschwingvorgang schneller abgeschlossen ist. Der PID-Regler ist geeignet für Strecken höheren Ordung, welche möglichst schnell und ohne bleibende Regelabweichung geregelt werden müssen.

Bild Blockschaltbild PID-Regler

2.3 Die Steuerung

Unter einer Steuerung versteht man eine offen Wirkungskette wie in Abbildung XX , dass heisst die Wirkglieder sind kettenähnlich aufgereiht und besitzen keine Rückkopplung. Die Steuerkette wird genau für eine Steuerung ausgelegt und kann nur einer Art von Störgrösse entgegenwirken. Ohne die Rückkopplung wird das Ausgangsignal nicht mit dem Eingangssignal verglichen und es können keine Korrekturen vorgenommen werden.

bild offener Regelkreis

Referenz Bild offener Regelkreis

2.4 Der geschlossene Regelkreis

Die Aufgabe eines geschlossenen Regelkreises (Abbildung 1) ist es, einen vorgegeben Sollwert zu erreichen und diesen auch bei Störungen aufrecht zu erhalten. Dabei sollen die unten genannten dynamischen Anforderungen eingehalten werden, damit die Stabilität des Regelsystems garantiert ist. Die wichtigste Bedingung für die Schrittantwort ein geschlossenen Regelkreis heisst, dass der Regelfehler, die Differenz zwischen Ist- und Sollwert, gleich Null oder möglichst klein ist.

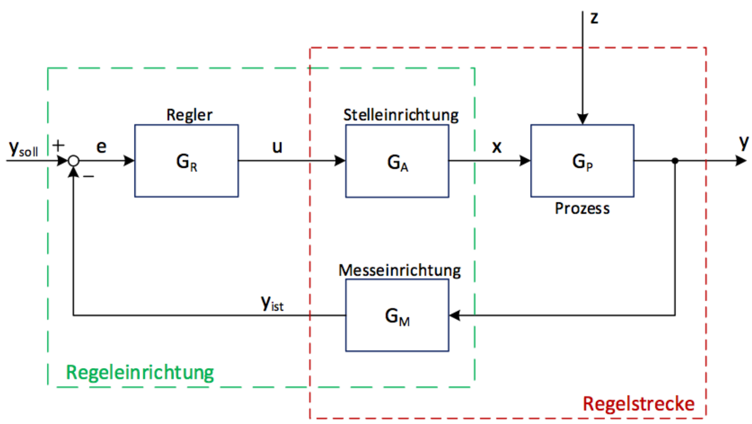

Abbildung 1: Geschlossener Regelkreis

- $y_s oll$ bezeichnet den Sollwert der Regelgrösse.
- e Regelabweichung (Regelfehler)

- u Steuergrösse
- x Stellgrösse
- y Regelgrösse
- z Störgrössen werden in diesem Projekt nicht berücksichtigt
- $y_i st$ ist der Ist-Wert der Regelgrösse und wird auch als die Schrittantwort des Regelkreises bezeichnet.

Grundsätzlich können fünf Anforderungen für einen geschlossenen Regelkreis und deren Schrittantworten zusammengefasst werden:

1. Der Regelkreis muss stabil sein: Für das Regelsystem heisst stabil, dass es in seinen Gleichgewichtszustand zurückgeführt werden kann.
2. Der Regelkreis muss genügend gedämpft sein.
3. Der Regelkreis muss eine bestimmte stationäre Genauigkeit aufweisen: Das bedeutet, der Regelfehler e(t) soll für t-> oo gegenNull gehen.

math mode

4. Der Regelkreis muss hinreichend schnell sein: Ist die Dämpfung zu stark oder zu schwach, braucht der Einschwingvorgang mehr Zeit. Hierbei muss darauf geachtet werden, dass die spezifischen Anforderungen an das Regelsystem eingehalten werden.
5. Der Regelkreis muss robust sein: Der Regelkreis muss so ausgelegt werden, dass das Regelsystem auch im schlimmsten Fall (je nach Regelsystem situationsabhängig) in der Lage ist, das System zurück in den stabilen Zustand (vgl. Punkt 1) zu regeln.

Die Schrittantwort des geschlossenen Regelkreises

Bild Schrittantworten passend zu Aufzählung unten

Als Schrittantwort eines geschlossenen Regelkreises wird das Ausgangssignal $y(t)$ bezeichnet. Im Zusammenhang mit den Anforderungen an den geschlossenen Regelkreis, werden an die Schrittantwort folgende Forderungen gestellt:

1. Die Schrittantwort eines stabilen Regelkreises darf nach dem Erreichen des eingeschwungenen Zustand kein erneutes Überschwingen auftreten.
2. Die Dämpfung der Schrittantwort soll so stark sein, dass der eingeschwungene Zustand möglichst rasch erreicht wird ohne dass das Überschwingen des Systems zu stark wird.
3. Die Schrittantwort muss für ein t->oo gleich $y_s oll$ sein.
4. Die Schnelligkeit des Einschwingvorganges der Schrittantwort ist stark von der Dämpfung abhängig. Wenn diese zu stark oder zu schwach ist, ist der Regelkreis zu langsam.

3 Fachlicher Hintergrund zur Regler-Dimensionierung

Das Kernstück dieser Arbeit und des zugehörigen Softwaretools stellt die so genannte "Phasengang-Methode zur Reglerdimensionierung" von Jakob Zellweger dar [1]. Diese wurde ursprünglich als vereinfachte grafische Methode zur Approximation der -20dB/Dek Methode erarbeitet und im Rahmen dieses Projektes in einem Java-Tool automatisiert. Als Vergleich wertet die Software ebenfalls einige der gängigen Faustformeln aus.

Das Tool führt grob vereinfacht folgende Schritte aus:

- Bestimmung des Frequenzgangs der Regelstrecke aus Verzögerungszeit T_u, Anstiegszeit T_g und Verstärkung K_s (Abschnitt 3.1)
- Dimensionierung des Reglers mittels Faustformeln (Abschnitt 3.2)
- Dimensionierung des Reglers durch Phasengangmethode (Abschnitte 3.3 und 3.4)
- Umrechung der Regler-Darstellung zwischen bodekonformer und reglerkonformer Darstellung (Abschnitt 3.5)
- Berechnung der Schrittantwort des geschlossenen Regelkreises (Abschnitt **??**)

Im folgenden Kapitel wird auf diese Punkte genauer eingegangen und das Vorgehen anhand eines konkreten Beispiels rechnerisch und grafisch erläutert. Die Durchrechnung der Phasengangmethode orientiert sich an den Rezepten, welche im fachlichen Teil des Pflichtenheftes dieses Projektes zu finden sind [2]. Genauere Hintergrundinformationen zur Phasengangmethode selbst sind dem Vorlesungs-Skript von J. Zellweger zu entnehmen [1].

Das Überschwingverhalten kann im Software-Tool vom Benutzer auf einen Zielwert zwischen 0% und 30% eingestellt werden. Das Tool optimiert den resultierenden Regler dann entsprechend, um dieser Vorgabe möglichst nahe zu komen.

3.1 Frequenzgang der Regelstrecke

Als Ausgangspunkt der Reglerdimensionierung dient die Schrittantwort der Strecke. Durch Einzeichnen der Wendetangente [1] ergeben sich Schnittpunkte der Wendetangente mit der Zeitachse $[T_u, 0]$ und mit dem Zielwert $[T_g + T_g, 1]$. Es können nun also die Verzögerungszeit T_u und die Anstiegszeit T_g aus aus Abbildung 2 abgelesen werden.

Wir werden in diesem Bericht folgende Strecke als Beispiel nehmen:

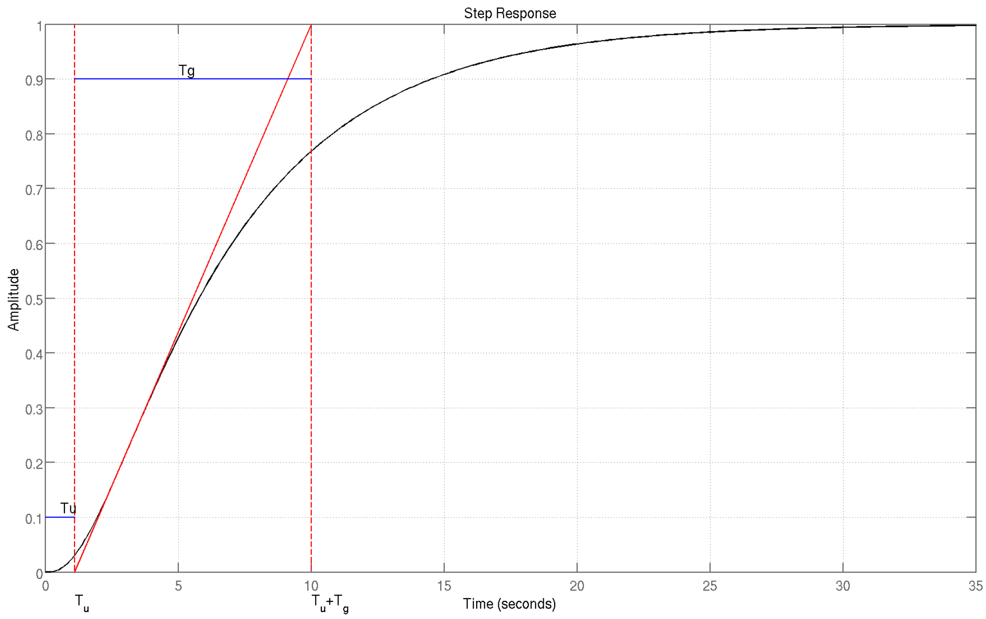

Abbildung 2: Schrittantwort der Beispielstrecke (schwarz), Wendetangende (rot), T_u und T_g (blau)

Ausmessen der Schrittantwort ergibt:

- $K_s = 2$ [2]
- $T_u = 1.1\,\mathrm{s}$
- $T_g = 8.9\,\mathrm{s}$

Der geschlossene Regelkreis soll schlussendlich maximal etwa 16.3% überschwingen.

Da die Reglerdimensionierung mit der Phasengangmethode vom *Frequenzgang* einer Strecke ausgeht und nicht von deren Schrittantwort, besteht der nächste Schritt nun darin, aus den obigen Werten den Frequenzgang der Strecke zu bestimmen. Dies erledigt die Methode `p_sani`[3], welche uns die Werte für die Übertragungsfunktion der Strecke liefert. In unserem Fall ergibt dies folgendes Polynom:

[1] Die Wendetangante ist die Tangente an den Wendepunkt in der Anstiegs-Phase der Schrittantwort.

[2] Abbildung 2 ist auf 1 normiert, die Verstärkung unserer Beispielstrecke beträgt 2. An den Werten für die Verzögerungs- und Anstiegszeit oder am Ausmessen der Schrittantwort ändert sich dadurch nichts

[3] Die Methode `p_sani` wurde zu Beginn des Projektes in einer Matlab-Implementation zur Verfügung gestellt und anschliessend für unser Tool in Java übersetzt.

Sie kann aus der Verzögerungszeit, der Anstiegszeit und der Verstärkung der Strecke ein Polynom für deren Übertragungsfunktion vom Grad 1 bis 8 ausrechnen.

Als Eingabeparameter werden die Werte T_u, T_g und K_s benötigt, als Rückgabewert erhält man ein Array mit den Zeiten T_i für die Nenner der Faktoren des Polynoms (siehe Gleichung 1).

$$
\begin{aligned}
H_s(s) &= K_s \cdot \frac{1}{1+s\cdot T_1} \cdot \frac{1}{1+s\cdot T_2} \cdot \frac{1}{1+s\cdot T_2} \\
&= 2 \cdot \frac{1}{1+s\cdot 0.4134\,\mathrm{s}} \cdot \frac{1}{1+s\cdot 1.4894\,\mathrm{s}} \cdot \frac{1}{1+s\cdot 5.3655\,\mathrm{s}}
\end{aligned} \tag{1}
$$

Mit einem geeigneten Tool kann man sich den dazugehörigen Plot erstellen lassen.

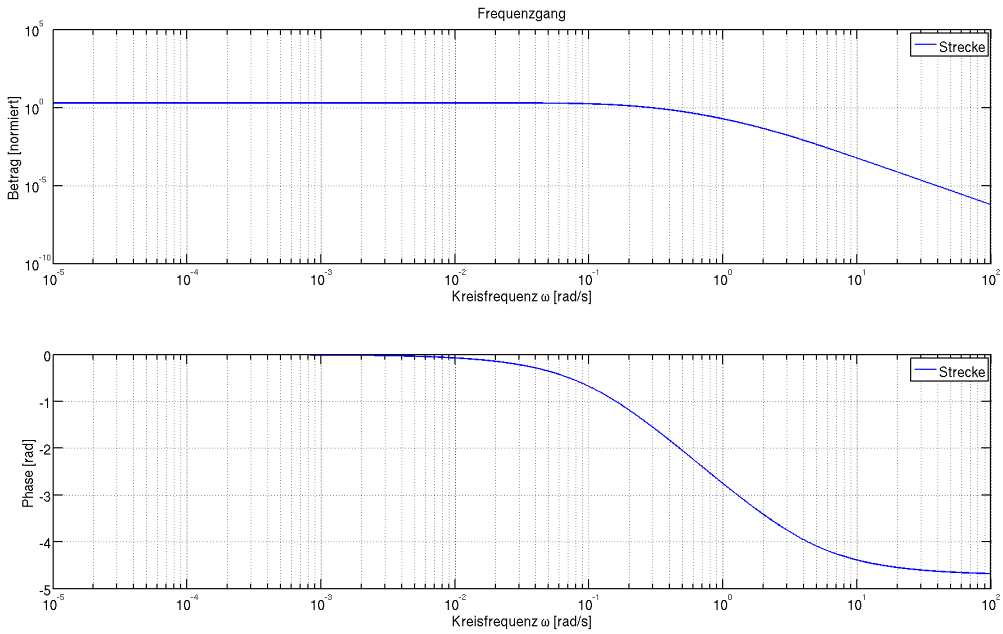

Abbildung 3: Frequenzgang der Strecke

Somit ist der Frequenzgang der Strecke bekannt und man kann mit einer geeigneten Methode den Regler dimensionieren.

3.2 Reglerdimensionierung mittels Faustformeln

Im Praxiseinsatz stehen für die Dimensionierung von Reglern einfache Berechnungsformeln zur Verfügung. Diese Lieferun Einstellwerte anhand von T_u, T_g und K_s. An dieser Stelle wird daher unsere Beispielstrecke zuerst mit einigen der gängigen Faustformeln dimensioniert, um das Ergebnis anschliessend mit dem Resultat der Phasengangmethode vergleichen zu können.

Faustformel	PI-Regler		PID-T1-Regler		
	T_n	K_p	T_n	T_v	K_p
Chiens, Hrones, Reswick **(0% Überschwingen)** [3], [4]	$1.2 \cdot T_g$	$\frac{0.35}{K_s} \cdot \frac{T_g}{T_u}$	T_g	$0.5 \cdot T_u$	$\frac{0.6}{K_s} \cdot \frac{T_g}{T_u}$
Chiens, Hrones, Reswick **(20% Überschwingen)** [3], [4]	T_g	$\frac{0.6}{K_s} \cdot \frac{T_g}{T_u}$	$1.35 \cdot T_g$	$0.47 \cdot T_u$	$\frac{0.95}{K_s} \cdot \frac{T_g}{T_u}$
Oppelt [5]	$3 \cdot T_u$	$\frac{0.8}{K_s} \cdot \frac{T_g}{T_u}$	$2 \cdot T_u$	$0.42 \cdot T_u$	$\frac{1.2}{K_s} \cdot \frac{T_g}{T_u}$

Faustformel	PI-Regler		PID-T1-Regler		
	T_n	K_p	T_n	T_v	K_p
Rosenberg [5]	$3.3 \cdot T_u$	$\frac{0.91}{K_s} \cdot \frac{T_g}{T_u}$	$2 \cdot T_u$	$0.45 \cdot T_u$	$\frac{1.2}{T_s} \cdot \frac{T_g}{T_u}$

Tabelle 1: Faustformeln zur Reglerdimensionierung

Setzt man die Werte für K_s, T_u, T_g in diese Formeln ein, ergibt sich Folgendes:

Faustformel	PI-Regler		PID-T1-Regler		
	T_n	K_p	T_n	T_v	K_p
Chiens, Hrones, Reswick **(0% Überschwingen)** [3], [4]	10.68 s	1.42	8.9 s	0.55 s	2.43
Chiens, Hrones, Reswick **(20% Überschwingen)** [3], [4]	8.9 s	2.43	12.02 s	52 s	3.84
Oppelt [5]	3.3 s	3.24	2.2 s	0.46 s	4.85
Rosenberg [5]	3.63 s	3.68	2.2 s	0.50 s	4.85

Tabelle 2: Reglerparameter bestimmt mit Faustformeln aus Tabelle 1

3.3 Reglerdimensionierung mittels Phasengangmethode: PI-Regler

Es werden nun anhand der Phasengangmethode sowohl ein PI- wie auch ein PID-Regler für die in Abschnitt 3.1 ausgemessene Strecke dimensioniert (siehe nächster Abschnitt für PID-Regler).

Tabelle 3 fasst die häufig verwendeten Begriffe in einer Übersicht zusammen:

$H_s(j\omega)$	Übertragungsfunktion der Regelstrecke
$A_s(j\omega) = \|H_s(j\omega)\|$	Amplitudengang der Regelstrecke
$\varphi_s(j\omega) = arg(H_s(j\omega))$	Phasengang der Regelstrecke
$H_r(j\omega)$	Übertragungsfunktion des Reglers
$A_r(j\omega) = \|H_r(j\omega)\|$	Amplitudengang des Reglers
$\varphi_r(j\omega) = arg(H_r(j\omega))$	Phasengang des Reglers
$H_o(j\omega) = H_s \cdot H_r(j\omega)$	Übertragungsfunktion des offenen Regelkreises
$A_o(j\omega) = \|H_o(j\omega)\|$	Amplitudengang des offenen Regelkreises
$\varphi_o(j\omega) = arg(H_o(j\omega)) = \varphi_s(j\omega) + \varphi_r(j\omega)$	Phasengang des offenen Regelkreises
$H_{rpid} = K_{rk}\left[\frac{(1+sT_{nk})(1+sT_{vk})}{sT_{nk}}\right]$	Übertragungsfunktion des PID-Reglers
$H_{rpi} = K_{rk}\left[1 + \frac{1}{sT_{nk}}\right]$	Übertragungsfunktion des PI-Reglers

Tabelle 3: Die wichtigsten Begriffsdefinitionen

Ziel

Das Ziel ist die Bestimmung der Parameter K_{rk} und T_{nk} in der Übertragungsfunktion des Reglers:

$$H_{rpi} = K_{rk} \cdot \left[1 + \frac{1}{s \cdot T_{nk}}\right] \tag{2}$$

1 Bestimmung der Reglerfrequenz ω_{pi}

Zuerst wird im Phasengang der Strecke die Frequenz ω_{pi} bestimmt, für welche die Phase der Strecke $-90°$ beträgt, ersichtlich in Abbildung 4 [4].

$$\varphi_s(\omega_{pi}) = -90° \tag{3}$$

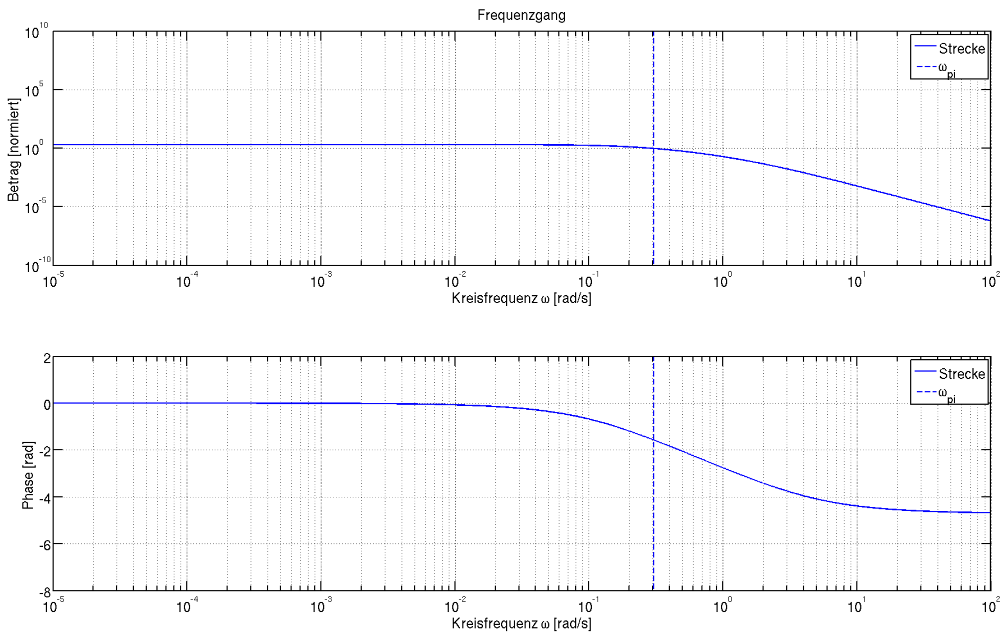

Abbildung 4: ω_{pi} eingetragen (vertikale gestrichelte Linie).

Wie man aus Abbildung 4 ablesen kann, liegt dieser Wert für ω_{pi} in unserem Beispiel bei ungefähr $0.3\,\mathrm{s}^{-1}$. Die Kontrollrechnung mittels Matlab ergibt:

$$\omega_{pi} = 0.3039\,\mathrm{s}^{-1} \tag{4}$$

[4] Der Winkel stellt keinen endgültigen Wert dar. Dieser wurde von Jakob Zellweger fixiert, um eine grafische Evaluation überhaupt zu ermöglichen. Durch Anpassung dieses Wertes kann je nach Regelstrecke das Regelverhalten weiter optimiert werden.

2 Bestimmung von T_{nk}

Damit kann nun T_{nk} direkt bestimmt werden[5]:

$$T_{nk} = \frac{1}{\omega_{pi}} = \frac{1}{0.3039\,\mathrm{s}^{-1}} = 3.2902\,\mathrm{s} \tag{5}$$

3 Bestimmung der Durchtrittsfrequenz ω_d

Die Durchtrittsfrequenz ist die Frequenz, bei der eine betrachtete Übertragungsfunktion eine Verstärkung von 0 dB = 1 aufweist. In der Phasengangmethode soll sie so festgelegt werden, dass der offene Regelkreis Gleichung 6 erfüllt. Dabei ist für φ_s abhängig vom gewünschten Überschwingverhalten ein Wert aus Tabelle 4 auszuwählen [6]. Nach dem Festlegen der Durchtrittsfrequenz wird dann im nächsten Abschnitt die Verstärkung des Reglers noch angepasst.

$$\varphi_o(\omega_d) = \varphi_s. \tag{6}$$

Überschwingen	0%	16.3%	23.3%
φ_s	$-103.7°$	$-128.5°$	$-135°$

Tabelle 4: Werte für φ_s

Um Gleichung 6 auswerten zu können, wird der Phasengang des *offenen Regelkreises* benötigt. Dazu wird der in Gleichung 5 erhaltene Wert für T_{nk} in die Übertragungsfunktion des Reglers (Gleichung 2) eingesetzt. K_{rk} ist noch unbekannt, hat aber auf die Phase keinen Einfluss und wird somit vorerst einfach auf 1 gesetzt.

$$\begin{aligned} H_{rpi} &= K_{rk} \cdot \left[1 + \frac{1}{s \cdot T_{nk}}\right] \\ &= 1 \cdot \left[1 + \frac{1}{s \cdot 3.2902\,\mathrm{s}}\right] \end{aligned} \tag{7}$$

Daraus kann nun der Frequenzgang des offenen Regelkreises (Übertragungsfunktion H_o, Amplitudengang A_o, Phasengang φ_o) bestimmt werden.

$$\begin{aligned} H_o(s) &= H_{rpi}(s) \cdot H_s(s) \\ &= \left(K_{rk} \cdot \left[1 + \frac{1}{s \cdot T_{nk}}\right]\right) \cdot K_s \cdot \left(\frac{1}{1 + s \cdot T_1} \cdot \frac{1}{1 + s \cdot T_2} \cdot \frac{1}{1 + s \cdot T_2}\right) \\ &= \left(1 \cdot \left[1 + \frac{1}{s \cdot 3.2902\,\mathrm{s}}\right]\right) \cdot 2 \cdot \left(\frac{1}{1 + s \cdot 0.4134\,\mathrm{s}} \cdot \frac{1}{1 + s \cdot 1.4894\,\mathrm{s}} \cdot \frac{1}{1 + s \cdot 5.3655\,\mathrm{s}}\right) \end{aligned} \tag{8}$$

[5]Um die Akkumulation von Ungenauigkeiten zu minimieren, werden bei diesen Berechnungen die genauen Werte aus Matlab verwendet und nicht die gerundeten Zwischenresultate, was zu Abweichungen zu den von Hand berechneten Ergebnissen führen kann.

[6]Die Werte für φ_s aus Tabelle 4 stellen keine abschliessende Auflistung dar und sind lediglich als Anhaltspunkte zu betrachten. Weicht das Verhalten des geschlossenen Regelkreises am Schluss zu stark vom gewünschten Ergebnis ab, besteht durch die Wahl anderer Werte für φ_s die Möglichkeit weiterer Optimierung.

Von besonderem Interesse ist der Phasengang $\varphi_o(j\omega)$ dieser Übertragungsfunktion (siehe Tabelle 3). Wie Anfangs spezifiziert, soll ein maximals Überschwingen von ca. 16.3% angestrebt werden. Dazu muss gemäss Tabelle 4 die Durchtrittsfrequenz ω_d gefunden werden, an welcher der offene Regelkreis eine Phase von $-128.5°$ aufweist (Gleichung 6). In Abbildung 5 kann dies grafisch verifiziert werden.

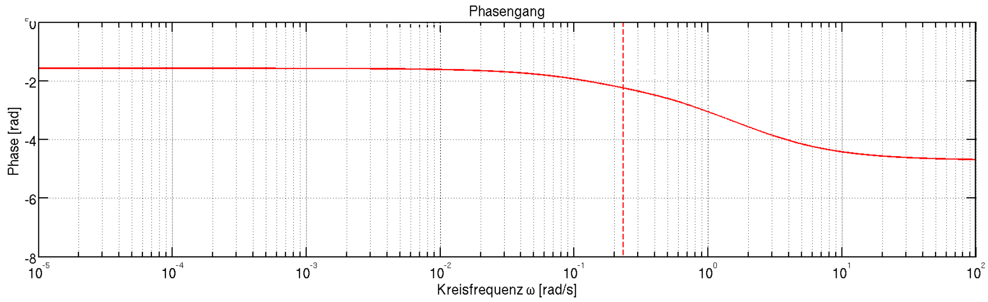

Abbildung 5: Phasengang $\varphi_o(j\omega)$ des offenen Regelkreises mit eingetragener Durchtrittsfrequenz ω_d (vertikale gestrichelte Linie). Wie man sieht, weist der offene Regelkreis unseres Beispiels bei dieser Kreisfrequenz eine Phase von $-128.5°$ auf (etwa $-2.24\,\mathrm{rad}$).

Dies ergibt:

$$\omega_d = 0.2329\,\mathrm{s}^{-1} \tag{9}$$

4 Bestimmung der Reglerverstärkung K_{rk}

Im letzten Schritt muss nun wie im vorherigen Abschnitt erwähnt die Verstärkung K_{rk} des Reglers noch angepasst werden, damit der offene Regelkreis bei der angestrebten Durchtrittsfrequenz ω_d auch effektiv eine Verstärkung von 1 aufweist. Dazu wird $j\omega_d$ in Gleichung 8 für den Parameter s eingesetzt und $|H_o(j\omega_d)| = 1$ gesetzt.

$$\begin{aligned} A_o &= |H_o(j\omega_d)| = |H_{rpi}(j\omega) \cdot H_s(j\omega)| \\ &= \left| \left(K_{rk} \cdot \left[1 + \frac{1}{j \cdot \omega_d \cdot T_{nk}} \right] \right) \cdot K_s \cdot \left(\frac{1}{1 + j \cdot \omega_d \cdot T_1} \cdot \frac{1}{1 + j \cdot \omega_d \cdot T_2} \cdot \frac{1}{1 + j \cdot \omega_d \cdot T_2} \right) \right| \\ &= 1 \end{aligned} \tag{10}$$

Mit den Werten

$$\begin{aligned} K_s &= 2 \\ T_{nk} &= 3.2902\,\mathrm{s} \\ T_1 &= 0.4134\,\mathrm{s} \\ T_2 &= 1.4894\,\mathrm{s} \\ T_3 &= 5.3655\,\mathrm{s} \\ \omega_d &= 0.2329\,\mathrm{rad\,s}^{-1} \end{aligned} \tag{11}$$

löst man Gleichung 10 nun nach K_{rk} auf und erhält:

$$K_{rk} = 0.517577 \tag{12}$$

5 Resultat

Somit ist der PI-Regler vollständig bestimmt und hat folgende Form:

$$H_{rpi} = 0.518 \cdot \left[1 + \frac{1}{s \cdot 3.29\,\mathrm{s}}\right] \tag{13}$$

In Abbildung 6 sind die wichtigsten Werte für diesen Prozess nochmals in einer Übersicht zusammengefasst.

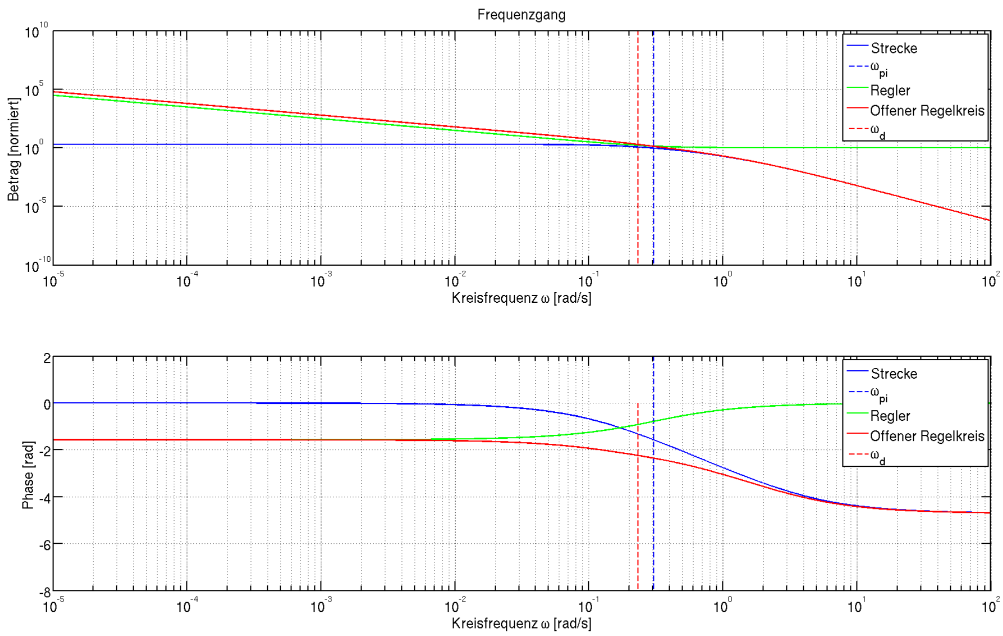

Abbildung 6: Frequenzgang des Reglers (grün), der Strecke (blau) und des offenen Regelkreises (rot).

Fix vertical line ω_d

3.4 Reglerdimensionierung mittels Phasengangmethode: PID-Regler

Ziel

Das Ziel ist die Bestimmung der Parameter K_{rk}, T_{nk} und T_{vk} in der Übertragungsfunktion des Reglers:

$$H_{rpid} = K_{rk} \cdot \left[\frac{(1 + s \cdot T_{nk}) \cdot (1 + s \cdot T_{vk})}{s \cdot T_{nk}}\right] \tag{14}$$

1 Bestimmung der Reglerfrequenz ω_{pid}

Analog zum PI-Regler wird zuerst im Phasengang der Strecke die Frequenz ω_{pid} bestimmt, für welche die Phase einen bestimmten Wert aufweist, nur wird hier $-135°$ benutzt [7]:

$$\varphi_s(\omega_{pid}) = -135° \tag{15}$$

In unserem Beispiel ergibt dies:

$$\omega_{pid} = 0.6714\,\mathrm{s}^{-1} \tag{16}$$

Eine grafische Überprüfung kann anhand von Abbildung 7 durchgeführt werden.

2 Steigung des Phasengangs bei der Reglerfrequenz

Anschliessend wird die Steigung des Phasengangs φ_s der Strecke bei der Frequenz ω_{pid} bestimmt. Ausgangspunkt dafür ist die von `p_sani` bestimmte Übertragungsfunktion der Strecke (siehe Gleichung 1).

$$\left.\frac{d\varphi_s}{d\omega}\right|_{\omega=\omega_{pid}} = \left.\frac{d(arg(H_s(j\omega)))}{d\omega}\right|_{\omega=\omega_{pid}} = -1.5124\,\mathrm{s} \tag{17}$$

Einheit überprüfen

3 Hilfsparameter β

Zwischen den Steigungen der Phasen des offenen Regelkreises (φ_o), der Strecke (φ_s) und des Reglers (φ_r) gilt gemäss Tabelle 3 folgende Beziehung:

$$\varphi_o = \varphi_s + \varphi_r \tag{18}$$

Da die Ableitung eine lineare Funktion ist, gilt somit auch:

$$\frac{d\varphi_o}{d\omega} = \frac{d\varphi_s}{d\omega} + \frac{d\varphi_r}{d\omega} \tag{19}$$

Diese Beziehungen können auch gut in Abbildung 7 von Hand überprüft werden.

Es soll nun gelten:

$$\left.\frac{d\varphi_o}{d\omega}\right|_{\omega=\omega_{pid}} = -\frac{1}{2} \tag{20}$$

Da $\frac{d\varphi_s}{d\omega}$ durch die Strecke gegeben und somit unveränderlich ist, kann lediglich der Wert von $\frac{d\varphi_r}{d\omega}$ angepasst werden, damit $\frac{d\varphi_o}{d\omega}$ Gleichung 20 erfüllt.

Dazu führt man den Hilfsparameter β ein, für den gilt:

$$\begin{aligned} \frac{1}{T_{vk}} &= \frac{\omega_{pid}}{\beta} \\ \frac{1}{T_{nk}} &= \omega_{pid} \cdot \beta \\ 0 &< \beta \leq 1 \end{aligned} \tag{21}$$

[7] Wie auch beim PI-Regler stellt diese Frequenz lediglich einen Ausgangspunkt dar und kann zur weiteren Optimierung des Resultats noch angepasst werden.

Wie in Abbildung 7 gesehen werden kann [8], liegen die beiden Frequenzen $\frac{1}{T_{vk}}$ und $\frac{1}{T_{nk}}$ symmetrisch um den Faktor β respektive $\frac{1}{\beta}$ oberhalb bzw. unterhalb der Frequenz ω_{pid}.

Will man β von Hand berechnen, trifft zuerst eine "vernünftige" Annahme, zum Beispiel:

$$\beta = 0.5 \tag{22}$$

Mit diesem Startwert bestimmt man nun T_{nk} und T_{vk}:

$$\begin{aligned} T_{vk} &= \frac{\beta}{\omega_{pid}} = \frac{0.5}{0.6714\,\mathrm{s}^{-1}} = 0.7447\,\mathrm{s} \\ T_{nk} &= \frac{1}{\omega_{pid}\cdot\beta} = \frac{1}{0.6714\,\mathrm{s}^{-1}\cdot 0.5} = 2.9789\,\mathrm{s} \end{aligned} \tag{23}$$

Die somit erhaltenen Werte setzt man in Gleichung 14 ein, zusammen mit dem Wert für ω_{pid} aus Gleichung 16. Da K_{rk} noch unbekannt ist, aber auf den Phasengang keinen Einfluss hat, setzt man vorerst $K_{rk} = 1$, um weiterrechnen zu können.

$$\begin{aligned} H_{rpid} &= K_{rk}\cdot\left[\frac{(1+j\omega\cdot T_{nk})\cdot(1+j\omega\cdot T_{vk})}{j\omega\cdot T_{nk}}\right] \\ &= 1\cdot\left[\frac{(1+j\omega\cdot 2.9789\,\mathrm{s})\cdot(1+j\omega\cdot 0.7447\,\mathrm{s})}{j\omega\cdot 2.9789\,\mathrm{s}}\right] \end{aligned} \tag{24}$$

Von dieser Gleichung bestimmt man nun den Phasengang und wertet danach dessen Ableitung an der Stelle $\omega = \omega_{pid}$ aus. Die zugehörige Rechnung kann in Anhang **??** gefunden werden.

$$\begin{aligned} \varphi_r(j\omega) &= arg(H_{rpid}(j\omega)) \\ \left.\frac{d\varphi_r}{d\omega}\right|_{\omega=\omega_{pid}} &= 1.1920\,\mathrm{s} \end{aligned} \tag{25}$$

Setzt man dies in Gleichung 18 ein, erhält man:

$$\begin{aligned} \left.\frac{d\varphi_o}{d\omega}\right|_{\omega=\omega_{pid},\beta=0.5} &= \left.\frac{d\varphi_s}{d\omega}\right|_{\omega=\omega_{pid}} + \left.\frac{d\varphi_r}{d\omega}\right|_{\omega=\omega_{pid},\beta=0.5} \\ &= -1.5124\,\mathrm{s} + 1.1920\,\mathrm{s} \\ &= -0.3204\,\mathrm{s} \\ &> -\frac{1}{2} \end{aligned} \tag{26}$$

Mit $\beta = 0.5$ erhält man also eine zu hohe Steigung des offenen Regelkreises an der Stelle ω_{pid}, folglich muss β *verkleinert* werden. Diese Berechnungen werden nun mit jeweils neuen Werten für β solange wiederholt, bis die Steigung des offenen Regelkreises die gewünschte Nähe zu $-\frac{1}{2}$ aufweist.

Da die manuelle Iterierung dieses Prozesses enorm viel Zeit in Anspruch nimmt, bietet sich hier eine Automatisierung an. Die Berechnung mittels eines geeigneten Algorithmus in Matlab liefert schlussendlich folgendes Ergebnis:

Allenfalls Matlab-Algo in Anhang und Verweis

$$\begin{aligned}\beta &= 0.2776\\ T_{vk} &= \frac{\beta}{\omega_{pid}} = 0.4134\,\mathrm{s}\\ T_{nk} &= \frac{1}{\omega_{pid}\cdot\beta} = 5.3656\,\mathrm{s}\end{aligned} \tag{27}$$

Diese Werte sind in ebenfalls in 7 eingetragen.

Wie kann dies egtl. passieren?

Sollte man für β einen komplexen Wert erhalten, wird $\beta = 1$ gesetzt.

4 Durchtrittsfrequenz ω_d

Als letzte Unbekannte verbleibt die Verstärkung K_{rk}. Wie auch beim PI-Regler ist zum Finden der Verstärkung die Durchtrittsfrequenz ω_d zu bestimmen, um anschliessend mit deren Hilfe K_{rk} auszurechnen.

Die Resultate aus Gleichung 27 werden in Gleichung 14 eingesetzt. K_{rk} ist immer noch unbekannt, und wird daher vorerst bei 1 belassen.

$$H_{rpid} = K_{rk}\cdot\left[\frac{(1+s\cdot T_{nk})\cdot(1+s\cdot T_{vk})}{s\cdot T_{nk}}\right] = 1\cdot\left[\frac{(1+s\cdot 5.3656\,\mathrm{s})\cdot(1+s\cdot 0.4134\,\mathrm{s})}{s\cdot 5.3656\,\mathrm{s}}\right] \tag{28}$$

Es interessiert hier der Phasengang des offenen Regelkreises (auch eingetragen in Abbildung 7), wozu die Übertragungsfunktion der Strecke (siehe Gleichung 1) mit der soeben bestimmten provisorischen Übertragungsfunktion des Reglers (Gleichung 28) multipliziert wird.

$$H_o(j\omega) = H_{rpid}(j\omega)\cdot H_s(j\omega) \tag{29}$$

Nun wird die Durchtrittsfrequenz ω_d bestimmt, an welcher der offene Regelkreis eine Verstärkung von $0\,\mathrm{dB} = 1$ aufweisen soll. Wie auch beim PI-Regler werden wir hier ein Überschwingen von 16.3% anstreben, womit gemäss Tabelle 4 gilt:

$$\varphi_s(\omega_d) = \varphi_s = -128.5° \tag{30}$$

Dieser Wert wird analog zum PI-Regler aus dem Phasengang des offenen Regelkreises abgelesen (siehe Abbildung 7). Eine Nachrechnung mittels Matlab ergibt:

$$\omega_d = 0.5341\,\mathrm{s}^{-1} \tag{31}$$

[8] Man beachte dabei, dass der Plot logarithmisch skaliert ist. Eine identische Wegstrecke zwischen zwei Punkte-Paaren auf der Frequenzachse bedeutet also, dass diese um denselben *Faktor* auseinander liegen, und nicht, dass die Differenz zwischen den jeweiligen Punkten identisch ist. Im Falle der Punkte-Paare $[\frac{1}{T_{nk}}, \omega_{pid}]$ und $[\omega_{pid}, \frac{1}{T_{vk}}]$ ist dieser Faktor β, wie in Gleichung 21 ersichtlich.

5 Bestimmung der Reglerverstärkung K_{rk}

Im letzten Schritt wird nun der Amplitudengang des offenen Regelkreises an der Stelle ω_d gleich 1 gesetzt und diese Gleichung nach K_{rk} aufgelöst:

$$\begin{aligned} A_o(j\omega_d) &= |H_o(j\omega_d)| = |H_{rpid}(j\omega_d) \cdot H_s(j\omega_d)| \\ &= \left| K_{rk} \cdot \left[\frac{(1 + j\omega_d \cdot T_{nk}) \cdot (1 + j\omega_d \cdot T_{vk})}{j\omega_d \cdot T_{nk}} \right] \right| \\ &\cdot \left| K_s \cdot \frac{1}{1 + j\omega_d \cdot T_1} \cdot \frac{1}{1 + j\omega_d \cdot T_2} \cdot \frac{1}{1 + j\omega_d \cdot T_2} \right| \\ &= 1 \end{aligned} \tag{32}$$

Die einzusetzenden Werte sind:

$$\begin{aligned} K_s &= 2 \\ T_1 &= 0.4134\,\mathrm{s} \\ T_2 &= 1.4894\,\mathrm{s} \\ T_3 &= 5.3655\,\mathrm{s} \\ T_{nk} &= 5.3656\,\mathrm{s} \\ T_{vk} &= 0.4134\,\mathrm{s} \\ \omega_d &= 0.5341\,\mathrm{s}^{-1} \end{aligned} \tag{33}$$

Womit man für die Verstärkung den Wert

$$K_{rk} = 1.83084 \tag{34}$$

erhält.

6 Resultat

Somit ist der Regler vollständig bestimmt und hat folgende Übertragungsfunktion:

$$H_{rpid}(s) = 1.83084 \cdot \left[\frac{(1 + s \cdot 5.3656\,\mathrm{s}) \cdot (1 + s \cdot 0.4134\,\mathrm{s})}{s \cdot 5.3656\,\mathrm{s}} \right] \tag{35}$$

Zusammenfassend sind in Abbildung 7 die verschiedenen Frequenzgänge und Frequenzen eingetragen.

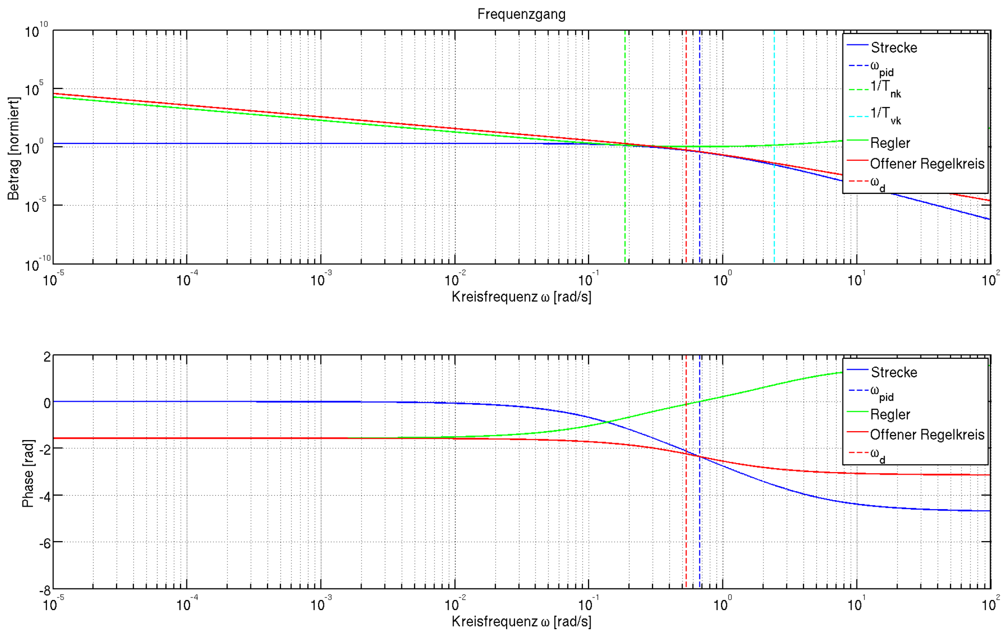

Abbildung 7: Frequenzgang der Strecke (blau), des Reglers (grün) und des offenen Regelkreises (rot). Ebenfalls eingetragen sind die Reglerfrequenz ω_{pid}, die beiden Frequenzen $\frac{1}{T_{vk}}$ und $\frac{1}{T_{nk}}$ sowie die Durchtrittsfrequenz ω_d.

3.5 Umrechnung zwischen bodekonformer und reglerkonformer Darstellung

Ein Regler kann mittels verschiedener mathematischen Gleichungen dargestellt werden. Für die Berechnungen in diesem Projekt sind zwei von besonderer Bedeutung: Die bodekonforme und die reglerkonforme Darstellung.

Dabei gilt es zu beachten, dass sich die einzelnen Darstellungen in ihrer mathematischen Gleichung unterscheiden, jedoch den selben Informationsinhalt haben. Deshalb ist es auch möglich, mittels einfacher Umrechnungen die Darstellungsweise zu wechseln.

Der Grund, weshalb zwei verschiedene Darstellungsarten existieren und verwendet werden, liegt zum einen darin, dass gewisse Berechnungen automatisch die eine oder andere Darstellungsart zurückgeben, jedoch auch, dass je nach Situation der Verständlichkeit halber die eine oder andere Darstellungsart bevorzugt wird.

Bei der bodekonformen Darstellung wie in Gleichungen 36 und 38 ist die Übertragungsfunktion leicht zu interpretieren, was ihren Amplitduden- und Frequenzgang betrifft. Deshalb wird die bodekonforme Darstellung überall dort verwendet, wo mit den Reglerwerten Berechnungen ausgeführt werden.

Die reglerkonforme Darstellung aus den Gleichungen 37 und 39 hingegen ist so aufgebaut, dass die Werte, welche an einem realen Regler einstellbar sind, direkt abgelesen werden können. Diese Darstellung wird verwendet, um die Reglerwerte dem Benutzer zur Verfügung zu stellen.

Zudem liegt das Ergebnis der Faustformeln auch immer in der reglerkonformen Darstellung vor. Für die Umrechnung zwischen den zwei Darstellungsarten ergibt sich durch einen Koeffizientenvergleich Tabelle 5.

$$H_{rpi} = K_{rk} \cdot \left[\frac{1 + s \cdot T_n}{s \cdot T_n}\right] \tag{36}$$

$$H_{rpi} = K_{rk} \cdot \left[1 + \frac{1}{s \cdot T_{nk}}\right] \tag{37}$$

$$H_{rpid} = K_{rk} \cdot \left[\frac{(1 + s \cdot T_{nk}) \cdot (1 + s \cdot T_{vk})}{s \cdot T_{nk}}\right] \tag{38}$$

$$H_{rpid} = K_{rk} \cdot \left[1 + \frac{1}{s \cdot T_n} + \frac{s \cdot T_v}{s \cdot T_p}\right] \tag{39}$$

	bodekonform → reglerkonform	reglerkonform → bodekonform
PI	$T_n = T_{nk}$	$K_{rk} = K_r$
PID	$T_n = T_{nk} + T_{vk} - T_p$	$T_{nk} = 0.5 \cdot (T_n + T_p) \cdot (1 + \epsilon)$
	$T_v = \frac{T_{nk} \cdot T_{vk}}{T_{nk} + T_{vk} - T_p} - T_p$	$T_{vk} = 0.5 \cdot (T_n + t_p) \cdot (1 - \epsilon)$
	$K_r = K_{rk} \cdot (1 + \frac{T_{vk} - T_p}{T_{nk}})$	$K_{rk} = 0.5 \cdot K_r \cdot (1 + \frac{T_p}{T_{nk}}) \cdot (1 + \epsilon)$
	wobei $\epsilon^2 = 1 - (4 \cdot T_n \cdot \frac{T_v - T_p}{(T_n + T_p)^2})$	

Tabelle 5: Formeln zur Umrechung zwischen bode- zu reglerkonformer Darstellung [6], [7]

Für die Berechnungen in diesem Projekt wird, wenn nicht anders angegeben, mit $T_p = \frac{1}{10} \cdot T_v$ gerechnet.

4 Software

Verweis auf Klassendiagramm

Zweck der Applikation ist die Dimensionierung eines Reglers ausgehend von einer Strecke und der zugehörigen Schrittantwort. Abschliessend werden die numerischen Parameter des dimensionierten Reglers ausgegeben sowie die Schrittantwort des geschlossenen Regelkreises grafisch dargestellt.

Die Software ist im bekannten *Model-View-Controller*-Pattern aufgebaut.

4.1 View

Die *View* ist aus zwei übergeordneten Panels aufgebaut. Im linken Panel befinden sich Ein- und Ausgabefelder für numerische Werte, im rechten Panel werden die zugehörigen Plots dargestellt.

Image Gesamt-GUI

Im Bereich 1 werden die Parameter der vermessenen Strecke eingegeben. Darunter befinden sich die Schaltflächen zur Wahl zwischen der Dimensionierung eines PI- respektive eines PID-T1-Reglers.

Image Panel Schrittantwort vermessen

Image Buttons PI-, PID-T1-Regler

Das Panel *Reglerwerte* dient hauptsächlich der Ausgabe der berechneten Reglerwerte mittels der verschiedenen Berechnungsmethoden. Ebenfalls kann für die Phasengangmethode die Zeitkonstante T_p spezifiziert werden.

Image Referenzen

Der obere Bereich des rechten Panels beinhaltet zwei Slider zur Eingabe des gewünschten Überschwingens respektive des Phasenrands.

Image Panel rechts

Im unteren Bereich werden die Plots der mittels Faustformeln und Phasengangmethode errechneten Resultate ausgegeben. Zu jeder Faustformel wird die zugehörige Schrittantwort abgebildet. Die Resultate der Phasengangmethode werden durch drei Kurven dargestellt. Eine Kurve benutzt den Standardwert des Phasenrands gemäss Zellweger , die beiden anderen Kurven basieren auf Benutzereingaben für einen oberen und unteren Offset des Phasenrandes im Bereich von $-45°$ bis $+45°$.

Check: korrekter Begriff

Image Panel Phasengangmethode

Einfügen Wert, Referenz

4.2 Controller

Der *Controller* ist verantwortlich für die Steueraufgaben und erzeugt den Regler. Da *Regler* übersetzt auf Englisch *Controller* ist, heisst die generische Reglerklasse in unserer Software `Controller`. Die Klasse, welche die die Rolle des *Controller* im Kontext von *Model-View-Controller* wahrnimmt, heisst daher `GUIController`, um Namenskonflikte zu vermeiden.

4.3 Model

Leserführung Model. Ausschnitt Klassendiagramm, Verweis auf gesamtes Diagramm.

Im Model wird für jede Berechnungsart, also entwerder für die Faustformeln oder die Phasengangmethode, Objekt des Typs `CloseLoop` erzeugt.

Controller

Der Controller bildet die Oberklasse für alle Faustformeln und die Phasengangmehode. Sie beinhaltet die abstrakte Methode `calculate()` sowie alle nötigen Getter und Setter.

Faustformeln

`Chien20`, `ChienApper`, `Oppelt`, `Rosenberg` und `ZieglerNichols` sind die Klassen zu den zugehörigen Fausformeln. Die abstrakte Methode `calculate()` aus der Klasee `Controller` ist in jeder dieser Klassen implementiert. Sie liest die benötigten Werte aus der Strecke (Klasse `Path`) aus und führt die Berechnungen gemäss der entsprechenden Faustformel aus.

ClosedLoop

In der Klasee `ClosedLoop` wird für jede Berechungsart jeweils ein `Controller` erstellt. Auch diese Klasse enthält eine Methode `calculate()`. In dieser wird einerseits für alle Rechenarten die Berechung der Schrittantwort mittels `calculateStepResponse()` ausgelöst und für die Phasengangmethode wird zusätzlich das Überschwingverhalten durch `overShootOptimazation()` optimiert.

PhaseRespsonseMethod

Dieser Abschnitt sollte vermutlich nomals Aufmerksamkeit von einem Experten zum Code erhalten.

In dieser Klasse wird die Frequenzachse in Abhängigkeit vom Phasengang und des benötigten Winkelbereiches erstellt. Ausserdem finden wir verschiedene calculate Methoden.

Einerseits ist da calculate() , in dieser wird die UTF Strecke aus der Strecke(path) geholt und die Omega-Achse Methode aufgeruffen. In dieser wird Sie Abhängigkeit vom Phasengang und des benötigten Winkelbereiches erstellt.

Anschlissend werden die Werte für Hs und phiS berehnet.

Andererseits sind da die Methoden, calculateTnvTnk() um Tnk und Tnv zu berechnen, calculateKrk um Krk zu berechnen, calculatecontrollerConf() um eine umrechung von Bodekonformen Werten zu Reglerkonformen Werten in der calc Klasse auszulösen, calculatePhaseMargin() bestimmt je nach Reglertyp den Phasenrand und zum schluss calculateOverShoot(), hier wird je nach vorher berechnetem Überschwingen, dem Wert phiu einen der 4 vordefinierten static final Werte zugewiesen.

4.4 Benutzungs-Beispiel (Use-Case)

Leserführung Use-Case. Ausschnitt Klassendiagramm, Verweis auf gesamtes Diagramm.

Beim Programmstart werden durch das Model drei *closedLoops* (geschlossene Regelkreise) für die Phasengang-Methode sowie vier weitere für die Faustformeln erzeugt. Jeder *closedLoop* weiss nun von Beginn an, welcher Berechnungstyp (Phasengang-Methode, Faustformel) er ist. Er ist nun bereit, Daten aufzunehmen und zu verarbeiten.

Über die drei Eingabefelder für K_s, T_u und T_g werden die Werte der vermessenen Regelstrecke durch den Benutzer eingegeben. Durch Drücken des Buttons *Berechnen* werden die Eingaben durch den `GUIController` auf Zulässigkeit überprüft. Erfüllen sie die erforderlichen Kriterien nicht, wird eine Benachrichtigung mit Hinweis auf den Fehler oberhalb des Buttons ausgegeben und die Berechnung nicht ausgelöst.

Haben die Eingaben die Überprüfung durch den GUI-Controller bestanden, fragt dieser zusätzlich die aktuellen Werte/ der Slider für Überschwingen und Optimierung sowie den Reglertyp auf dem GUI ab und leitet alle Daten mittels `setData()` an das Model weiter. Dieses erzeugt einen `Path` (Strecke) aus den Eingabewerten. Das Model errechnet den richtigen Optimierungs-Offset und weist die Daten den entsprechenden `closedLoops` der Phasengang-Methode sowie der Faustformeln mittels der `setData()`-Methode zu. Jeder `closedLoop` leitet die Daten an den zugehörigen Controller weiter, der die Reglerwerte berechnet.

5 Tests

6 Schlussfolgerungen

Ehrlichkeitserklärung

Mit der Unterschrift bestätigt der Unterzeichnende (Projektleiterin), dass das Dokument selbst geschrieben worden ist und alle Quellen sauber und korrekt deklariert worden sind.

Anita Rosenberger: ____

Ort, Datum: _________________________, ____________________

A Beschreibung der Algorithmen

A.1 Sani

Input

T_u	Verzugszeit
T_g	Anstiegszeit

Output

n	Ordnung der Regelstrecke
T	Zeitkonstante

Algorithmus

1. Ungültige Eingaben warden abgefangen und ein Fehler zurückgegeben.
2. Lädt Werte für Tu und Tg.
3. Erstellt 50 Werte zwischen 0 und 1 für ri.
4. Bestimmt die Ordnung der Regelstrecke.
5. Spline für r und w
6. T(n) wird aus w*tg berechnet.
7. Umspeichern & Sortieren

Matlab-Code

```
function [n,T] = p2_sani(tu,tg,p)

if tu<=0 || tg<=0
  disp(' ');
  error('!!!!! unsinnige Zeiten !!!!!');
end;

v=tu/tg;
if v>0.64173
  disp(' ');
  error('!!!!! Tu/Tg zu gross --> N > 8  !!!!!');
end;

if v<0.001
  disp(' ');
  error('!!!!! Tu/Tg zu klein --> N = 1  !!!!!');
end;

load('p2_sani_tu_tg');
pause(0.1);                    % Pause, damit Laden vom File erfolgreich!!!!

% Berechnet mit NN=50 (r-Aufloesung)
ri=linspace(0,1,50);

if v <= 0.103638        % abhaengig von n werden vorberechnete
  n=2;                  % Datenfiles von der Festplatte geladen.
elseif v <= 0.218017    % 2 <= n <= 8
  n=3;                  % n=1 ist trivial und fuehrt zu Abbruch
elseif v <= 0.319357
  n=4;
elseif v <= 0.410303
  n=5;
elseif v <= 0.4933
  n=6;
elseif v <= 0.5700
  n=7;
elseif v<=0.64173
```

```
  n=8;
else
  n=10;
end;

r=spline(Tu_Tg(n,:),ri,v);
w=spline(ri,T_Tg(n,:),r);
T(n)=w*tg;


for i=n-1:-1:1,                  % Umspeicher, damit gleiche Reihenfolge wie bei Hudzovik
  T(i)=T(n)*r^(n-i);
end;

% Plots der Schrittantworten
if p==1
    TT=4*(tg);
    t=linspace(0,TT,2500);
    za=1;
    n1=conv([T(1) 1],[T(2) 1]);
    for k=3:n
        nen1=conv(n1,[T(k) 1]);
        n1=nen1;
    end;
    nens=n1;
    step(za,nens,'k'); grid on;
    hold on;
    %wendepkt(T);
    hold off;
end;
```

A.2 Umrechnung von reglerkonformer in bodekonforme Darstellung

Input

T_v	Vorhaltezeit
T_n	Nachstellzeit
T_p	Periodendauer
K_r	Verstärkungsfaktor des Reglers
Reglertyp	Typ des Reglers (P, PI, PID)

Output

T_{nk}	Nachstellzeit
T_{vk}	Vorhaltezeit
K_{rk}	Verstärkungsfaktor des Reglers

Algorithmus

1. Wählt je nach Reglertyp die Umrechnungsformel.
2. Falls der I-Regler gewählt wird, gibt der Algorithmus einen Fehler zurück, da der I-Regler nicht implementiert ist.
3. PI-Regler: $T_{nk} = T_n$, $K_{rk} = K_r$, $T_{vk} = 0$
4. Für PID-Regler:

$$\varepsilon = \frac{\sqrt{1 - (4 \cdot T_n \cdot (T_v - T_p))}}{(T_n + T_p)^2}$$

$$T_{nk} = \frac{(T_n + T_p) \cdot (1 + \varepsilon)}{2}$$

$$K_{rk} = \frac{K_r \cdot (\frac{1+T_p}{T_{nk}}) \cdot (1+\varepsilon)}{2}$$

$$T_{vk} = \frac{(T_n + T_p) * (1+\varepsilon)}{2}$$

Matlab-Code

```
function [ t_nk, t_vk, k_rk ] = p2_bodekonf(t_n, t_v, t_p, k_r ,reglertyp )

if (reglertyp = 1) %I-Regler
     error('!!!! Nicht implementierter Reglertyp !!!!');
end;

if (reglertyp = 2) %PI-Regler
    t_nk=t_n;
    k_rk=k_r;
    t_vk=0;
end;

if (reglertyp = 3) %PID-Regler
    epsilon=sqrt(1-(4*t_n*(t_v-t_p))/(t_n+t_p)^2;
    t_nk=0.5*(t_n+t_p)*(1+epsilon);
    k_rk=0.5*k_r(1+t_p/t_nk)*(1+epsilon);
    t_vk=0.5*(t_n+t_p)*(1+epsilon);
end;
```

A.3 Umrechnung von bodekonformer in reglerkonforme Darstellung

Input

T_p	Periodendauer
Tnk	Nachstellzeit
Tvk	Vorhaltezeit
Krk	Verstärkungsfaktor des Reglers
Reglertyp	Reglertyp (P, PI, PID)

Output

T_n	Nachstellzeit
T_v	Vorhaltezeit
K_r	Verstärkungsfaktor des Reglers

Algorithmus

1. Wählt je nach Reglertyp die Umrechnungsformel.
2. Falls der I-Regler gewählt wird, gibt der Algorithmus einen Fehler zurück, da der I-Regler nicht implementiert ist.
3. PI-Regler: $T_n = T_{nk}$, $K_k = K_{rk}$, $T_v = 0$
4. Für PID-Regler:

$$K_r = K_{rk} \cdot \frac{1 + T_{vk}}{T_{nk}}$$

$$T_n = T_{nk} + T_{vk} - T_p$$

$$T_v = \frac{T_{nk} \cdot T_{vk}}{T_{nk} + T_{vk} - T_p) - T_p}$$

Matlab-Code

```
function [ t_n, t_v, k_r ] = p2_reglerkonf(t_nk, t_vk, t_p, k_rk ,reglertyp )
if (reglertyp = 1) %I-Regler
     error('!!!! Nicht implementierter Reglertyp !!!!');
end;

if (reglertyp = 2) %PI-Regler
    t_n=t_nk;
    k_r=k_rk;
    t_v=0;
end;


if (reglertyp = 3) %PID-Regler
    k_r=k_rk*(1+t_vk/t_nk);
    t_n=t_nk+t_vk-t_p;
    t_v=(t_nk*t_vk)/(t_nk+t_vk-t_p)-t_p;
end;
```

A.4 utfController

Input

T_p	Verzugszeit
T_{nk}	Nachstellzeit
T_{vk}	Vorhaltezeit
K_{rk}	Verstärkungsfaktor des Reglers
Reglertyp	Reglertyp (P, PI, PID)

Output

korrekt? Oder einfach reelle Koeffizienten?

Zähler	Reeller Zähler
Nenner	Reeller Nenner

Algorithmus

Matrix

1. Wählt je nach Reglertyp die korrekten Formeln.
2.

Matlab-Code

```
function [ Zah_r, Nen_r ] = p2_UTFRegler(t_nk, t_vk, t_p, k_rk ,Reglertyp )

% PI Regler
if (Reglertyp == 2)
    Zah_r = k_rk*[t_nk 1];
    Nen_r = [t_nk 0];
end;

%ID-Regler
if (Reglertyp == 3)
    Zah_r = k_rk * p2_xdiskConv([t_vk 1],[t_nk 1]);
    Nen_r = [t_nk 0];
end;
```

A.5 Faustformel Oppelt

Input

T_p	Verzugszeit
T_u	Anstiegszeit
K_s	Verstärkung der Strecke
Reglertyp	Reglertyp (P, PI, PID)

Output

K_p	Proportionalitätsfaktor
T_n	Nachstellzeit
T_v	Vorhaltezeit

Algorithmus

1. Wählt je nach Reglertyp die korrekten Formeln.
2. Für PI:

$$K_p = \frac{0.8}{K_s} \cdot \frac{T_g}{T_u}$$

$$T_n = 3 \cdot T_u$$

$$t_v = 0$$

3. Für PID:

$$K_p = \frac{1.2}{K_s} \cdot \frac{T_g}{T_u}$$

$$T_n = 2 \cdot T_u$$

$$T_v = 0.42 * T_u$$

Matlab-Code

```
function [k_p, t_n, t_v] = p2_ffoppelt(t_u,t_g,k_s,reglertyp)

if reglertyp <2
     error('!!!! Nicht implementierter Reglertyp !!!!');
elseif reglertyp <= 2 %PI Regler
    k_p=(0.8/k_s)*(t_g/t_u);
    t_n=3*t_u;
    t_v=0;
elseif reglertyp <= 3 %PID Regler
    k_p=(1.2/k_s)*(t_g/t_u);
    t_n=2*t_u;
    t_v=0.42*t_u;
else
    error('!!!! Reglertyp nicht berechenbar !!!!')
end;
```

A.6 Faustformel Rosenberg

Input

T_p	Verzugszeit
T_u	Anstiegszeit
K_s	Verstärkung der Strecke
Reglertyp	Reglertyp (P, PI, PID)

Output

K_p	Proportionalitätsfaktor
T_n	Nachstellzeit
T_v	Vorhaltezeit

Algorithmus

1. Wählt je nach Reglertyp die korrekten Formeln.
2.
3. Für PI:

$$K_p = \frac{0.91}{K_s} \cdot \frac{T_g}{T_u}$$

$$T_n = 3.3 \cdot T_u$$

$$T_v = 0$$

4. Für PID:

$$Kp = \frac{1.2}{K_s} \cdot \frac{T_g}{T_u}$$

$$T_n = 2 \cdot T_u$$

$$T_v = 0.45 \cdot T_u;$$

Matlab-Code

```
function [k_p, t_n, t_v] = p2_ffrosenberg(t_u,t_g,k_s,reglertyp)

if reglertyp <2
    error('!!!! Nicht implementierter Reglertyp !!!!');
elseif reglertyp == 2 %PI Regler
    k_p=(0.91/k_s)*(t_g/t_u);
    t_n=3.3*t_u;
    t_v=0;
elseif reglertyp == 3 %PID Regler
    k_p=(1.2/k_s)*(t_g/t_u);
    t_n=2*t_u;
    t_v=0.45*t_u;
else
    error('!!!! Reglertyp nicht berechenbar !!!!')
end;
```

A.7 Faustformel Ziegler

Input

T_p	Verzugszeit
T_u	Anstiegszeit
K_s	Verstärkung der Strecke
Reglertyp	Reglertyp (P, PI, PID)

Output

K_p	Proportionalitätsfaktor
T_n	Nachstellzeit
T_v	Vorhaltezeit

Algorithmus

1. Wählt je nach Reglertyp die korrekten Formeln.
2. Für PI:

$$K_p = \frac{0.9}{K_s} \cdot \frac{T_g}{T_u}$$

$$T_n = 3.33 \cdot T_u$$

$$T_v = 0$$

3. Für PID:

$$K_p = \frac{1.2}{K_s} \cdot \frac{T_g}{T_u}$$

$$T_n = 2 \cdot T_u$$

$$T_v = 0.5 \cdot T_u$$

Matlab-Code

```matlab
function [k_p, t_n, t_v] = p2_ffziegler(t_u,t_g,k_s,reglertyp)

if reglertyp <2
     error('!!!! Nicht implementierter Reglertyp !!!!');
elseif reglertyp == 2 %PI Regler
    k_p=(0.9/k_s)*(t_g/t_u);
    t_n=3.33*t_u;
    t_v=0;
elseif reglertyp == 3 %PID Regler
    k_p=(1.2/k_s)*(t_g/t_u);
    t_n=2*t_u;
    t_v=0.5*t_u;
else
    error('!!!! Reglertyp nicht berechenbar !!!!')
end;
```

A.8 Faustformel Chien

Input

T_p	Verzugszeit
T_u	Anstiegszeit
K_s	Verstärkung der Strecke
Reglertyp	Reglertyp (P, PI, PID)
Überschwingen	Flag für Überschwingeng

Output

K_p	Proportionalitätsfaktor
T_n	Nachstellzeit
T_v	Vorhaltezeit

Algorithmus

1. Wählt je nach Reglertyp die korrekten Formeln.
2. Für PI, ohne Überschwingen:

$$K_p = \frac{0.35 \cdot T_g}{K_s \cdot T_u}$$

$$T_n = 1.2 \cdot T_u$$

$$T_v = 0$$

3. Für PI, 20% Überschwingen:

$$K_p = \frac{0.7 \cdot T_g}{K_s \cdot T_u}$$

$$T_n = 3 \cdot T_u$$

$$T_v = 0$$

4. Für PID, ohne Überschwingen:

$$K_p = \frac{0.9 \cdot Tg}{Ks \cdot Tu}$$

$$T_n = 2.4 \cdot T_u$$

$$T_v = 0.42 cdot T_u;$$

5. Für PID, 20% Überschwingen:

$$K_p = \frac{1.2 \cdot T_g}{K_s \cdot T_u}$$

$$T_n = 2 \cdot T_u$$

$$T_v = 0.42 \cdot T_u;$$

Matlab-Code

```
function [k_p, t_n, t_v] = p2_ffchien(t_u,t_g,k_s,reglertyp,ueberschwingen)

if reglertyp <2
    error('!!!! Nicht implementierter Reglertyp !!!!');
elseif reglertyp == 2 %PI Regler
    if ueberschwingung == 0 % 0% Ueberschwingen
            k_p=0.35*t_g/(k_s*t_u);
            t_n=1.2*t_u;
            t_v=0;
    else % 20% Ueberschwingen
            k_p=0.7*t_g/(k_s*t_u);
            t_n=2.3*t_u;
            t_v=0;
    end;
elseif reglertyp == 3 %PID Regler
    if ueberschwingung == 0 % 0% Ueberschwingen
            k_p=0.95*t_g/(k_s*t_u);
            t_n=2.4*t_u;
            t_v=0.42*t_u;
    else % 20% Ueberschwingen
            k_p=1.2*t_g/(k_s*t_u);
            t_n=2*t_u;
            t_v=0.42*t_u;
    end;
else
    error('!!!! Reglertyp nicht berechenbar !!!!')
end;
```

A.9 diskDiff

`diskDiff` berechnet die Steigung einer Funktion (repräsentiert durch zwei Arrays) an einem bestimmten Array-Index.

Input

x-Array	Array mit x-Werten
y-Array	Array mit zugehörigen Funktionswerten
Index	Index, an dem die Steigung berechnet werden soll

Output

Steigung	Steigung an gesuchter Stelle

Algorithmus

1. Prüfen, ob Index innerhalb des Arrays liegt.
2. Steigungsdreieck zwischen Element and Index und den unmittelbar daneben liegenden Array-Elementen bilden.
3. Durchschnitt der beiden Steigungsdreiecke ausrechnen.
4. Falls Steigung an erster Array-Stelle verlangt ist: Steigungsdreieck mit dem zweiten Element bilden und Steigung zurückgeben.
5. Falls Steigung an letzter Array-Stelle verlangt ist: Steigungsdreieck mit zweitletztem Array-Element bilden und Steigung zurückgeben.

Java-Code

```java
public static double diskDiff(double[] x, double[] y, int index) {
    if (index > 0 & index < x.length - 1) {
        double diff2 = (y[index + 1] - y[index]) / (x[index + 1] - x[index]);
        double diff1 = (y[index] - y[index - 1]) / (x[index] - x[index - 1]);
        double diff = (diff1 + diff2) / 2;
        return diff;
    } else if (index == 0)
        return (y[index + 1] - y[index]) / (x[index + 1] - x[index]);
    else if (index == x.length)
        return (y[index] - y[index - 1]) / (x[index] - x[index - 1]);
    else
        return 0;
}
```

A.10 schrittIfft

korrekt?

`schrittIfft` berechnet die Schrittantwort im Zeitbereich einer Übertragungsfunktion.

Input

Zähler	Zähler der Übertragungsfunktion
Nenner	Nenner der Übertragungsfunktion
Frequenz	Frequenz, bis zu der die Frequenzachse ausgewertet werden soll.
n	Granulierung der Frequenzachse

Output

Resultat	Zweidimensionales Array mit Zeitachse und zugehörigen Funktionswerten

Algorithmus

1. Array mit für Frequenzachse generieren.
2. Frequenzgang der Übertragungsfunktion berechnen.
3. Impulsantwort im Frequenzbereich berechnen.
4. In den Zeitbereich zurücktransformieren.
5. Aus den Realteilen des Resultat-Arrays die Schrittantwort zusammensetzen (aufsummieren des Realteils des aktuellen Array-Elements mit den Realteilen aller vorhergehenden Array-Elementen).

Java-Code

```java
public static double[][] schrittIfft(double[] zah, double[] nen, double fs, int n) {

    double T = 1 / fs; // Periode
    Complex[] H;

    // Frequenzachse berechnen
    double[] w = linspace(0.0, fs * Math.PI, n / 2); // Kreisfrequenz

    // Frequenzgang berechnen
    H = freqs(zah, nen, w);

    // Symmetrischen Vektor fuer Ifft erstellen:
    Complex[] tmp = new Complex[H.length];
    tmp = colonColon(H, (n / 2) - 1, -1, 1);
```

```
    for (int i = 0; i < tmp.length; i++) {
        tmp[i] = tmp[i].conjugate();
    }

    Complex x = new Complex(0);
    H = concat(colonColon(H, 0, 1, (n / 2) - 1), new Complex[] { x }, tmp);

    // Impulsantwort berechen
    Complex[] h;// = new Complex[H.length];
    FastFourierTransformer f = new FastFourierTransformer(DftNormalization.STANDARD);
    h = f.transform(H, TransformType.INVERSE);

    // Realteil von h extrahieren.
    double[] hReal = new double[h.length];

    for (int i = 0; i < h.length; i++) {
        hReal[i] = h[i].getReal();
    }

    // Schrittantwort berechnen
    // double[] y = Calc.diskConvOnes(hReal, n);
    double[] y = new double[n];
    y[0] = hReal[0];
    for (int i = 1; i < y.length; i++) {
        y[i] = y[i - 1] + hReal[i];
    }

    // Resultate ausschneiden. Halbiert die Laenge von y.
    double[] yShort = colonColon(y, 0, 1, (int) ((y.length / 2) - 1));

    // Zeitachse generieren:
    double[] t;
    t = linspace(0.0, (yShort.length - 1) * T, yShort.length);

    // Fuer Output zusammmensetzen:
    double[][] res = new double[2][yShort.length];

    for (int j = 0; j < res[0].length; j++) {
        res[0][j] = yShort[j];
    }
    for (int i = 0; i < res[1].length; i++) {
        res[1][i] = t[i];
    }
    return res;
}
```

A.11 overShootOptimisation

`overShootOptimisation` optimiert das Überschwingverhalten des generierten Reglers.

Input

Keine Eingabewerte

Output

Keine Rückgabewerte

Algorithmus

1. Das Maximum in der Schrittantwort des geschlossenen Regelkreises finden.
2. Diesen Wert mit dem vom Benutzer gewünschten maximalen Überschwingen vergleichen.
3. Falls zu starkes Überschwingen: Reglerverstärkung K_{rk} schrittweise reduzieren, bis gewünschtes Verhalten eingehalten wird.

4. Falls zu schwaches Überschwingen: Reglerverstärkung K_{rk} schrittweise erhöhen, bis gewünschtes Verhalten eingehalten wird.

Java-Code

```java
private void overShootOptimazation() {
    double max = yt[0][Calc.diskMax(yt[0])];
    PhaseResponseMethod phaseResponseMethod = (PhaseResponseMethod) controller;
    double Krk = phaseResponseMethod.getControllerValues()[PhaseResponseMethod.KrkPOS];
    if (max > controller.overShoot / 100.0 + 1.0) {
        while (max > controller.overShoot / 100.0 + 1.0) {
            Krk = phaseResponseMethod.getControllerValues()[PhaseResponseMethod.KrkPOS];
            phaseResponseMethod.setKrk(Krk / 1.05);
            calculateStepResponse();
            max = yt[0][Calc.diskMax(yt[0])];
        }
    } else {
        while (max < controller.overShoot / 100.0 + 1.0 & Krk < 1000) {
            Krk = phaseResponseMethod.getControllerValues()[PhaseResponseMethod.KrkPOS];
            phaseResponseMethod.setKrk(Krk * 1.05);
            calculateStepResponse();
            max = yt[0][Calc.diskMax(yt[0])];
        }
    }
}
```

B Manuelle Berechnung des Hilfsparameteres β

Der erste Iterationsschtitt der in Abschnitt 3.4 erwähnten manuellen Berechnung des Hilfsparameteres β ist hier im Detail ausgeführt.

Zur Rekapitulation eine kurze Wiederholung der Ausgangslage:

$$\begin{aligned}
\omega_{pid} &= 0.6714\,\mathrm{s}^{-1} \\
T_{vk} &= \frac{\beta}{\omega_{pid}} = \frac{0.5}{0.6714\,\mathrm{s}^{-1}} = 0.7447\,\mathrm{s} \\
T_{nk} &= \frac{1}{\omega_{pid}\cdot\beta} = \frac{1}{0.6714\,\mathrm{s}^{-1}\cdot 0.5} = 2.9789\,\mathrm{s} \\
K_{rk} &= 1
\end{aligned} \tag{40}$$

Diese Werte eingesetzt in Gleichung 14 ergeben:

$$\begin{aligned}
H_{rpid}(j\omega) &= K_{rk}\cdot\left[\frac{(1+s\cdot T_{nk})(1+s\cdot T_{vk})}{s\cdot T_{nk}}\right] \\
&= 1\cdot\left[\frac{(1+j\omega\cdot 0.7447\,\mathrm{s})(1+j\omega\cdot 2.9789\,\mathrm{s})}{j\omega\cdot 2.9789\,\mathrm{s}}\right] \\
&= \frac{1+j\omega\cdot(2.9789\,\mathrm{s}+0.7447\,\mathrm{s})-\omega^2\cdot 0.7447\,\mathrm{s}\cdot 2.9789\,\mathrm{s}}{j\omega\cdot 2.9789\,\mathrm{s}} \\
&= \frac{1-2.2184\,\mathrm{s}^2\cdot\omega^2+j\omega\cdot 3.7236\,\mathrm{s}}{j\omega\cdot 2.9789\,\mathrm{s}} \\
&= \frac{-\omega\cdot 3.7236\,\mathrm{s}+j(1-\omega^2\cdot 2.2184\,\mathrm{s}^2)}{\omega\cdot 2.9789\,\mathrm{s}} \\
&= -1.250+j\cdot(\omega^{-1}\cdot 0.3357\,\mathrm{s}^{-1}-\omega\cdot 0.7450\,\mathrm{s})
\end{aligned} \tag{41}$$

Von dieser Zahl gilt es nun, das Argument zu bestimmen und abzuleiten. $H_{rpid}(j\omega)$ ist eine komplexe Zahl in der linken Halbebene ($Re < 0$), somit kommen folgende Formeln zur Berechnung des Arguments in Frage:

$$\begin{aligned}
\varphi(Re+j\cdot Im) &= atan\left(\frac{Im}{Re}\right)+\pi \qquad Re<0\wedge Im\geq 0 \\
\varphi(Re+j\cdot Im) &= atan\left(\frac{Im}{Re}\right)-\pi \qquad Re<0\wedge Im<0
\end{aligned} \tag{42}$$

Da aber in diesem Fall lediglich die *Ableitung* von φ benötigt wird, fällt der Summand $\pm\pi$ weg und welche Formel für die Berechnung des Arguments verwendet wird, ist ohne Konsequenz.

$$\begin{aligned}
\varphi(H_{rpid}(j\omega)) &= atan\left(\frac{\omega^{-1}\cdot 0.3357\,\mathrm{s}^{-1}-\omega\cdot 0.7450\,\mathrm{s}}{-1.250}\right)\pm\pi \\
&= atan\left(\omega^{-1}\cdot -0.2686\,\mathrm{s}^{-1}-\omega\cdot 0.5960\,\mathrm{s}\right)\pm\pi
\end{aligned} \tag{43}$$

Die Ableitung des Arkustangens ist:

$$\frac{d}{dx} atan(x) = \frac{1}{1+x^2} \tag{44}$$

Mit

$$x(j\omega) = \omega^{-1} \cdot -0.2686\,\mathrm{s}^{-1} - \omega \cdot 0.5960\,\mathrm{s} \tag{45}$$

folgt

$$\begin{aligned} \frac{d}{d\omega}\varphi(H_{rpid}(j\omega)) &= \frac{d}{dx} atan(x(j\omega)) \cdot \frac{d}{d\omega} x(j\omega) \\ &= \frac{0.5960 + \omega^{-2} \cdot 0.2686\,\mathrm{s}^2}{1 + (\omega \cdot 0.5960\,\mathrm{s} - \omega^{-1} \cdot 0.2686\,\mathrm{s}^{-1})^2} \\ &\approx 1.1920 \end{aligned} \tag{46}$$

Wie in Gleichung 26 gezeigt, ist dies noch nicht der gesuchte Wert für β. Für den nächsten Iterationsschritt würde nun ein kleinerer Wert gewählt (z.B. $\beta = 0.25$), der zu neuen Werten für T_{nk} und T_{vk} führen würde, mit denen dann die Berechnungen aus Gleichungen 41 bis 46 erneut ausgeführt würden. Bei zufriedenstellender Nähe der Steigung des offenen Regelkreises zu $-\frac{1}{2}$ ist die Iteration beendet.

Literatur

[1] J. Zellweger, „Phasengang-Methode," Kapitel aus Vorlesungsskript.

[2] A. Rosengerger, B. Müller, M. Suter, F. Alber, und R. Frey, „Projekt 2: Pflichtenheft – Fachlicher Teil," April 2015.

[3] (2011, März) Reglereinstellung nach Chiens, Hrones, Reswick. [Online]. Verfügbar: http://mathematik.tsn.at/content/files1/CHR_mit_ohne_Ausgleich1344.pdf [Stand: 23. März 2015].

[4] (2015, März) Faustformelverfahren (Automatisierungstechnik). [Online]. Verfügbar: http://de.wikipedia.org/wiki/Faustformelverfahren_(Automatisierungstechnik) [Stand: 23. März 2015].

[5] (1999, Jan) Anpassung eines Reglers an eine Regelstrecke – Einstellregeln. [Online]. Verfügbar: http://techni.chemie.uni-leipzig.de/reg/parcalchelp.html [Stand: 23. März 2015].

[6] J. Zellweger, „Regelkreise und Regelungen," Vorlesungsskript.

[7] W. Schumacher und W. Leonhard, „Grundlagen der Regelungstechnik," 2001, Vorlesungsskript.