

Reglerdimensionierung mittels Phasengangmethode

Fachbericht

1. Juni 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

In der Regelungstechnik ist die korrekte Dimensionierung der Regler grundlegend. Die zu regelnde Strecke kann nur durch richtig eingestellte Regelwerte wie gewünscht verändert werden.

Dieses Projekt hat sich zum Ziel gesetzt ein Softwaretool zu entwickeln, welches aus den eingegebenen Streckenwerten K_s , T_u und T_g PI- und PID-Regler dimensioniert. Das Tool berechnet die Reglerwerte mit der sogenannten Phasengangmethode, welche von Jakob Zellweger, ehemaliger Dozent der Fachhochschule Nordwestschweiz, stammt. Diese Methode wurde ursprünglich für die graphische Auswertung des Phasengangs entwickelt und durch Handarbeit mit Geodreieck und Bleistift angewendet. Das Softwaretool ermöglicht die bewährte Methode effizient durch ein automatisiertes Verfahren anzuwenden.

Das entwickelte Softwaretool simuliert neben den berechneten Reglerwerten auch die Schrittantwort des geschlossenen Regelkreises. Zusätzlich wurde die Berechnung dreier gängigen Faustformeln mit implementiert. Zum Vergleich können die Schrittantworten der Faustformeln im gleichen Plot angezeigt werden. Für die Phasengangmethode werden gleichzeitig drei Graphen abgebildet. Einer mit der Standard-Reglerknickfrequenz gemäss Phasengangmethode und zwei weitere welche sich in der Reglerknickfrequenz um $\pm 10^\circ$ unterscheiden.

Zusätzlich kann die Reglerknickfrequenz und das Überschwingen mittels Schieberegler manuell verändert werden. Dies ermöglicht die im Vergleich, zu der ursprünglichen Handarbeit, enorm hohe Geschwindigkeit. Somit wird eine Echtzeit Optimierung der bestehenden Phasengangmethode getätigt.

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 Schrittantwort 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 P- und PID-Regler dimensioniert werden (Ziegler/Nichols, Chien/Hrones/Reidwork, Oppel/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 noch optimieren, 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 die 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. Zoidweger, *Regelkreise und Regelungen*, Vorlesungsskript.
- [2] J. Zoidweger, *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
3	Fachlicher Hintergrund zur Regler-Dimensionierung	9
3.1	Frequenzgang der Regelstrecke	10
3.2	Reglerdimensionierung mittels Faustformeln	11
3.3	Reglerdimensionierung mittels Phasengangmethode: PI-Regler	13
3.4	Reglerdimensionierung mittels Phasengangmethode: PID-Regler	17
3.5	Umrechnung zwischen bodekonformer und reglerkonformer Darstellung	23
4	Software	25
4.1	View	25
4.2	Controller	25
4.3	Model	25
4.4	Benutzungs-Beispiel (Use-Case)	26
5	Tests	28
6	Schlussfolgerungen	29
	Appendix	31
A	Beschreibung der Algorithmen	31
A.1	Sani	31
A.2	Umrechnung von reglerkonformer in bodekonforme Darstellung	33
A.3	Umrechnung von bodekonformer in reglerkonforme Darstellung	34
A.4	utfController	35
A.5	Faustformel Oppelt	35
A.6	Faustformel Rosenberg	37
A.7	Faustformel Ziegler	38
A.8	Faustformel Chien	39
A.9	diskDiff	41
A.10	schrittIfft	42
A.11	overShootOptimisation	44
A.12	Berechnung von T_{vk} und T_{nk}	45

B Manuelle Berechnung des Hilfsparameteres β	47
Literaturverzeichnis	49

Versionsgeschichte

04.05.2015: Version 0.01

06.05.2015: Version 0.02

Entwurf

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 Java konvertiert. 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 den 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 aus einer Strecke mit proportionalen Verhalten und einer mit Totzeit sowie der Angabe der Ordnung n der Strecke.

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. Das Verhalten der Strecke ist in seinem Blockschaltbild (Abb.??) symbolisch dargestellt. Der Proportionalitätsfaktor wird mit K_p abgekürzt. Wird $K_p < 1$ wirkt K_p nicht mehr verstärkend sondern abschwächend.

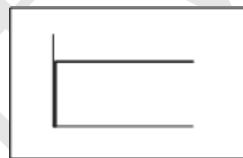


Abbildung 1: Blockschaltbild von P-Strecke

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.

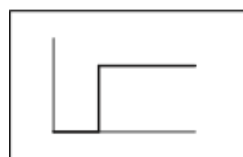


Abbildung 2: Blockschaltbild von Strecke mit Totzeit

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 systemgegebenen Schranken gegeben. Die Integrierzeit T_i ist ein Mass für die Anstiegsgeschwindigkeit der Regelgrösse und das Blockschaltbild (Abb. ??) zeigt das Verhalten sinnbildlich.

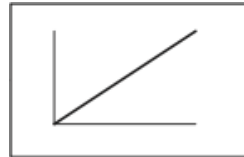


Abbildung 3: Blockschaltbild von I-Strecke

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 Regelgrösse 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.

1 PI-Regler

Der PI-Regler besteht aus einer Parallelschaltung von einem P- und einem I-Regler (Abb.??). Durch diese Kombination werden die Nachteile beider Regler aufgehoben und die Vorteile (schnell, stabil) hervorgehoben. Sein Verhalten wird bildlich in dem Blockschaltbild in Abbildung ??.

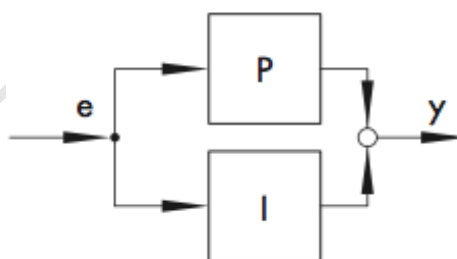


Abbildung 4: Parallelschaltung von P-Regler und I-Regler

2 PID-Regler

Wird dem PI-Regler ein D-Anteil parallel geschaltet (Abb. ??), 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. Das Blockschaltbild zeigt dieses Verhalten (Abb.??) figürlich. Der PID-Regler ist geeignet für Strecken höheren

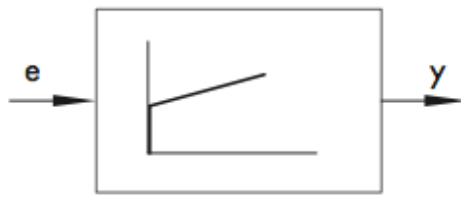


Abbildung 5: Blockschaltbild von PI-Regler

Ordnungen, welche möglichst schnell und ohne bleibende Regelabweichung geregelt werden müssen.

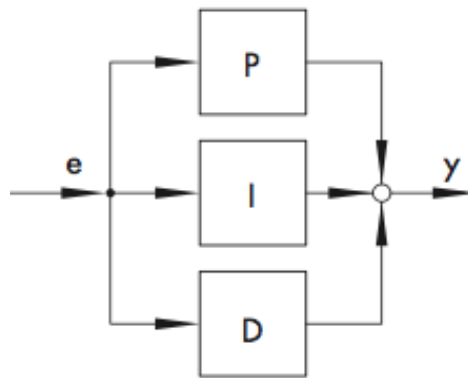


Abbildung 6: Parallelschaltung von P-, I-, und D-Regler

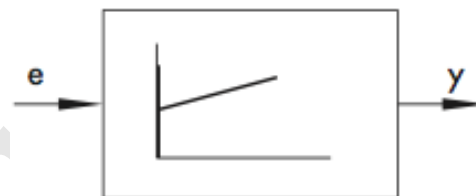


Abbildung 7: Blockschaltbild des PID-Reglers

2.3 Die Steuerung

Unter einer Steuerung versteht man eine offene Wirkungskette wie in Abbildung ??, das heißt die Wirkglieder sind kettenähnlich aufgereiht und besitzen keine Rückkopplung. Die Steuerkette wird genau für eine Steuerung ausgelegt und kann nur einer Störgrösseart entgegenwirken. Ohne die Rückkopplung wird das Ausgangssignal nicht mit dem Eingangssignal verglichen und es können keine Korrekturen vorgenommen werden.

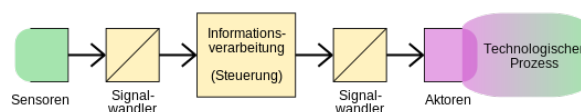


Abbildung 8: Steuerung

2.4 Der geschlossene Regelkreis

Die Aufgabe eines geschlossenen Regelkreises (Abbildung ??) 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.

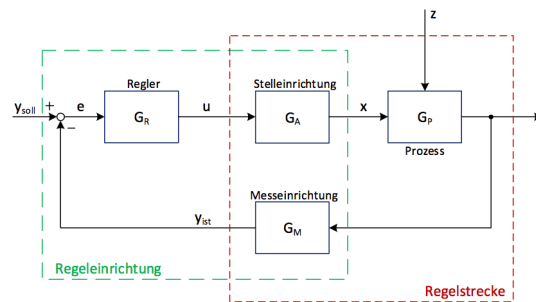


Abbildung 9: Geschlossener Regelkreis

- y_{soll} bezeichnet den Sollwert der Regelgrösse.
- e Regelabweichung (Regelfehler)
- u Steuergrösse
- x Stellgrösse
- y Regelgrösse
- z Störgrössen werden in diesem Projekt nicht berücksichtigt
- y_{ist} 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 $t \rightarrow \infty$ gegen null gehen.
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. 1.) zu regeln.

1 Die Schrittantwort des geschlossenen Regelkreises

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 $t \rightarrow \infty$ gleich y_{soll} 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)
- Umrechnung 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 kommen.

3.1 Frequenzgang der Regelstrecke

Als Ausgangspunkt der Reglerdimensionierung dient die Schrittantwort der Strecke. Durch Einzeichnen der Wendetangente¹ ergeben sich Schnittpunkte der Wendetangente mit der Zeitachse $[T_u, 0]$ und mit dem Zielwert $[T_g + T_u, 1]$. Es können nun also die Verzögerungszeit T_u und die Anstiegszeit T_g aus Abbildung 1 abgelesen werden.

Wir werden in diesem Bericht folgende Strecke als Beispiel nehmen:



Abbildung 10: Schrittantwort der Beispielschleife (schwarz), Wendetangente (rot), T_u und T_g (blau)

Ausmessen der Schrittantwort ergibt:

- $K_s = 2$ ²
- $T_u = 1.1$ s
- $T_g = 8.9$ 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:

¹Die Wendetangente ist die Tangente an den Wendepunkt in der Anstiegs-Phase der Schrittantwort.

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

³Die 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).

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

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

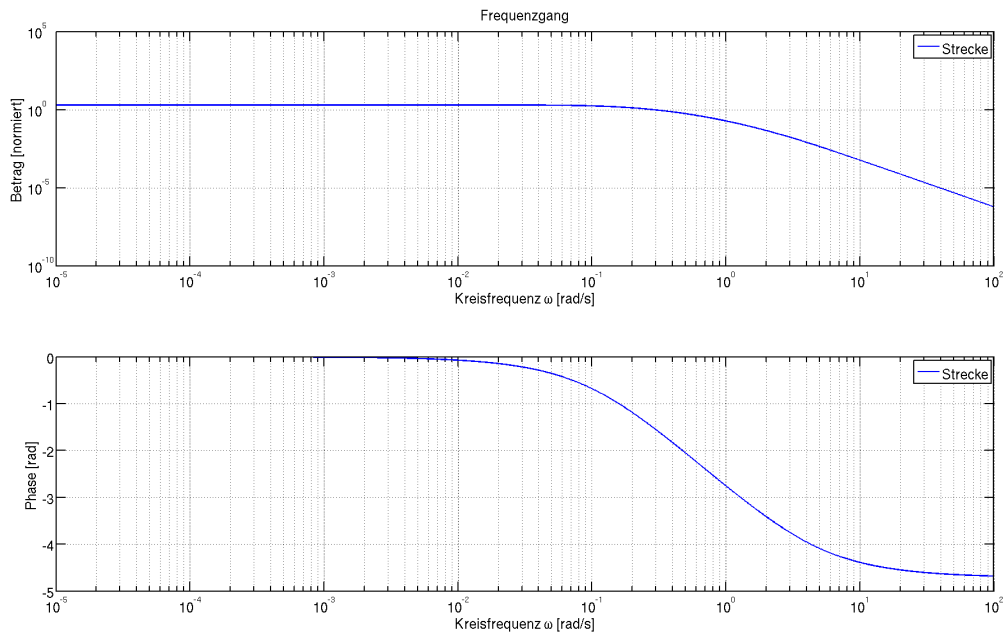


Abbildung 11: Frequenzgang der Strecke

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 liefern 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

Entwurf

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% Überspringen) [3], [4]	10.68 s	1.42	8.9 s	0.55 s	2.43
Chiens, Hrones, Reswick (20% Überspringen) [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:

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

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 3 ⁴.

$$\varphi_s(\omega_{pi}) = -90^\circ \quad (3)$$

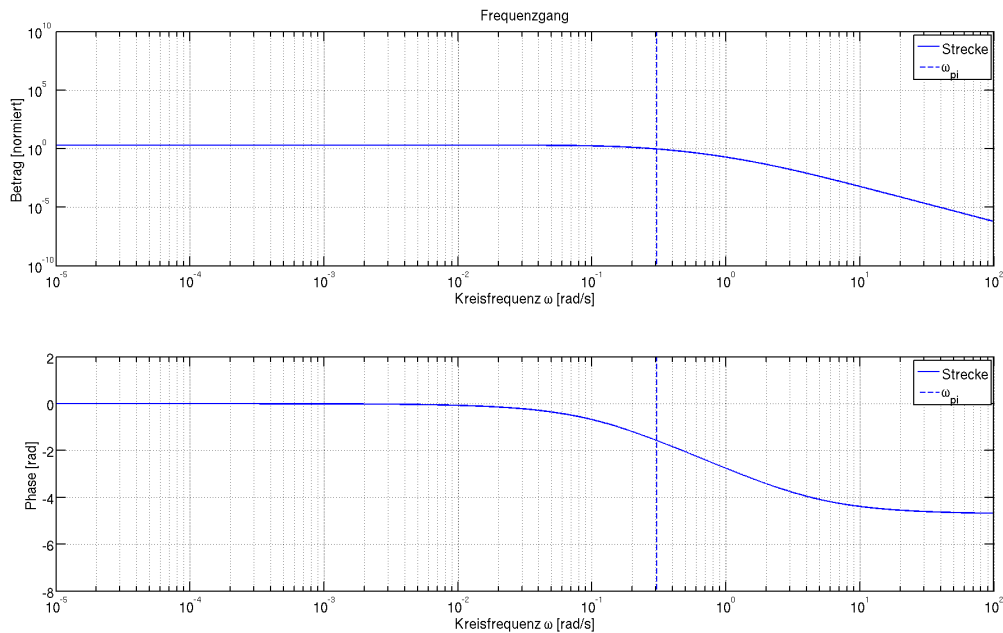


Abbildung 12: ω_{pi} eingetragen (vertikale gestrichelte Linie).

Wie man aus Abbildung 3 ablesen kann, liegt dieser Wert für ω_{pi} in unserem Beispiel bei ungefähr 0.3 s^{-1} . Die Kontrollrechnung mittels Matlab ergibt:

$$\omega_{pi} = 0.3039 \text{ s}^{-1} \quad (4)$$

⁴Der 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 werden⁵:

$$T_{nk} = \frac{1}{\omega_{pi}} = \frac{1}{0.3039 \text{ s}^{-1}} = 3.2902 \text{ s} \quad (5)$$

3 Bestimmung der Durchtrittsfrequenz ω_d

Die Durchtrittsfrequenz ist die Frequenz, bei der eine betrachtete Übertragungsfunktion eine Verstärkung von $0 \text{ dB} = 1$ 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 auszuwählen⁶. Nach dem Festlegen der Durchtrittsfrequenz wird dann im nächsten Abschnitt die Verstärkung des Reglers noch angepasst.

$$\varphi_o(\omega_d) = \varphi_s. \quad (6)$$

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.

$$\begin{aligned} H_{rpi} &= K_{rk} \cdot \left[1 + \frac{1}{s \cdot T_{nk}} \right] \\ &= 1 \cdot \left[1 + \frac{1}{s \cdot 3.2902 \text{ s}} \right] \end{aligned} \quad (7)$$

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

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

⁵Um 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.

⁶Die 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 maximales Überspringen 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 4 kann dies grafisch verifiziert werden.

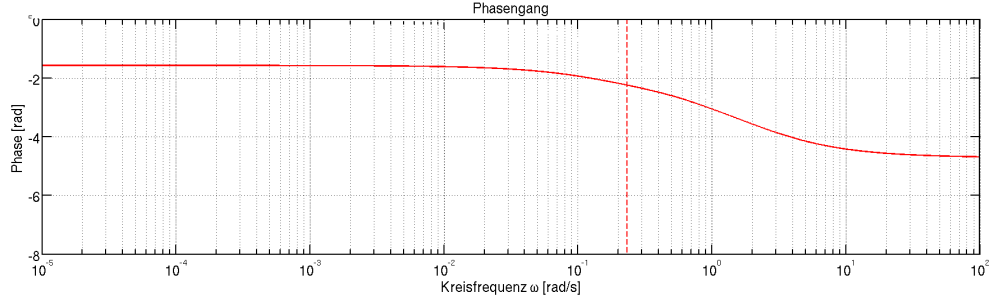


Abbildung 13: Phasengang $\varphi_o(j\omega)$ des offenen Regelkreises mit eingetragener Durchtrittsfrequenz ω_d (vertikale gestrichelte Linie). Wie man sieht, weist der offene Regelkreis unseres Beispiels bei dieser Kreisfrequenz eine Phase von -128.5° auf (etwa -2.24 rad).

Dies ergibt:

$$\omega_d = 0.2329 \text{ s}^{-1} \quad (9)$$

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.

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

Mit den Werten

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

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

$$K_{rk} = 0.517577 \quad (12)$$

5 Resultat

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

$$H_{rpi} = 0.518 \cdot \left[1 + \frac{1}{s \cdot 3.29 \text{ s}} \right] \quad (13)$$

In Abbildung 5 sind die wichtigsten Werte für diesen Prozess nochmals in einer Übersicht zusammengefasst.

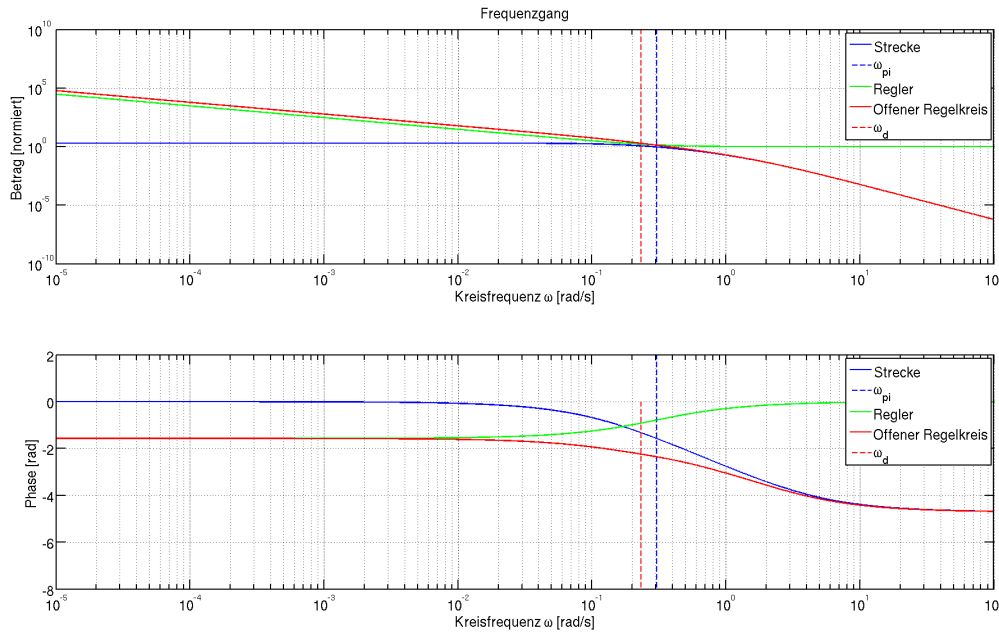


Abbildung 14: Frequenzgang des Reglers (grün), der Strecke (blau) und des offenen Regelkreises (rot).

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:

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

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:

$$\varphi_s(\omega_{pid}) = -135^\circ \quad (15)$$

Fix vertical line
 ω_d

In unserem Beispiel ergibt dies:

$$\omega_{pid} = 0.6714 \text{ s}^{-1} \quad (16)$$

Eine grafische Überprüfung kann anhand von Abbildung 6 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).

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

Einheit
überprü-
fen

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:

$$\varphi_o = \varphi_s + \varphi_r \quad (18)$$

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

$$\frac{d\varphi_o}{d\omega} = \frac{d\varphi_s}{d\omega} + \frac{d\varphi_r}{d\omega} \quad (19)$$

Diese Beziehungen können auch gut in Abbildung 6 von Hand überprüft werden.

Es soll nun gelten:

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

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:

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

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

Wie in Abbildung 6 gesehen werden kann⁸, 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:

$$\beta = 0.5 \quad (22)$$

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

$$\begin{aligned} T_{vk} &= \frac{\beta}{\omega_{pid}} = \frac{0.5}{0.6714 \text{ s}^{-1}} = 0.7447 \text{ s} \\ T_{nk} &= \frac{1}{\omega_{pid} \cdot \beta} = \frac{1}{0.6714 \text{ s}^{-1} \cdot 0.5} = 2.9789 \text{ s} \end{aligned} \quad (23)$$

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.

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

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.

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

Setzt man dies in Gleichung 18 ein, erhält man:

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

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:

$$\begin{aligned}\beta &= 0.2776 \\ T_{vk} &= \frac{\beta}{\omega_{pid}} = 0.4134 \text{ s} \\ T_{nk} &= \frac{1}{\omega_{pid} \cdot \beta} = 5.3656 \text{ s}\end{aligned}\quad (27)$$

Allenfalls
Matlab-
Algo in
Anhang
und Ver-
weis

Diese Werte sind in ebenfalls in 6 eingetragen.

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

Wie kann
dies egtl.
passie-
ren?

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.

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

Es interessiert hier der Phasengang des offenen Regelkreises (auch eingetragen in Abbildung 6), wozu die Übertragungsfunktion der Strecke (siehe Gleichung 1) mit der soeben bestimmten provisorischen Übertragungsfunktion des Reglers (Gleichung 28) multipliziert wird.

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

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

$$\varphi_s(\omega_d) = \varphi_s = -128.5^\circ \quad (30)$$

Dieser Wert wird analog zum PI-Regler aus dem Phasengang des offenen Regelkreises abgelesen (siehe Abbildung 6). Eine Nachrechnung mittels Matlab ergibt:

$$\omega_d = 0.5341 \text{ s}^{-1} \quad (31)$$

⁸Man 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:

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

Die einzusetzenden Werte sind:

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

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:

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

Zusammenfassend sind in Abbildung 6 die verschiedenen Frequenzgänge und Frequenzen eingetragen.

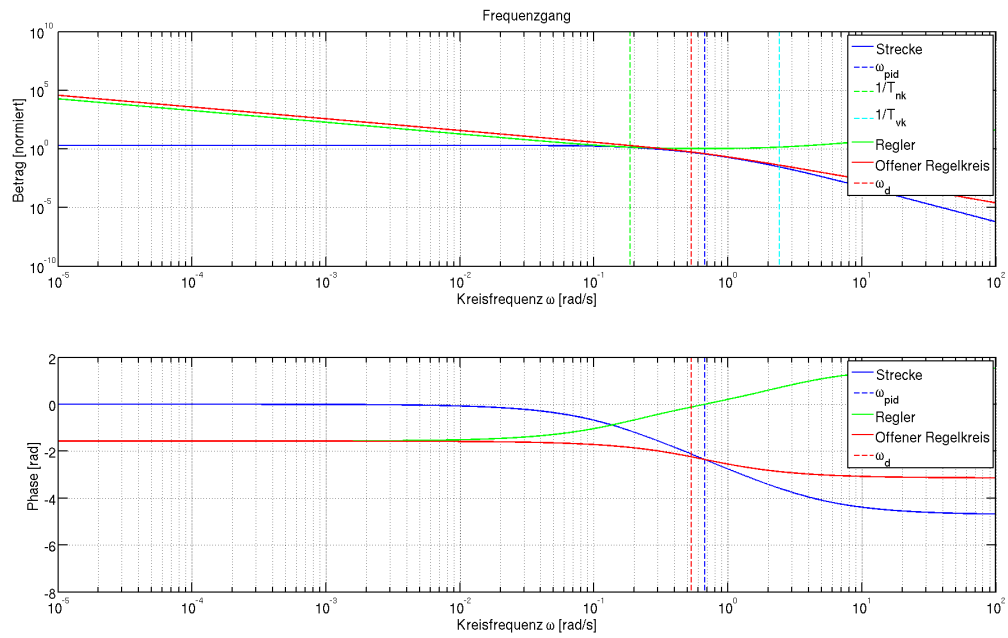


Abbildung 15: Frequenzgang der Strecke (blau), des Reglers (grün) und des offenen Regelkreises (rot). Ebenfalls eingetragen sind die Reglerfrequenz ω_{pid} , die beiden Frequenzen $\frac{1}{T_{vk}}$ und $\frac{1}{T_{nk}}$ sowie die Durchtrittsfrequenz ω_d .

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 Amplitudengang 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.

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

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

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

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

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

Tabelle 5: Formeln zur Umrechnung 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.

Entwurf

4 Software

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.

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.

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.

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

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 $+45^\circ$.

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 entweder für die Faustformeln oder die Phasengangmethode, Objekt des Typs **CloseLoop** erzeugt.

Controller

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

Verweis
auf Klas-
sendiagramm

Image
Gesamt-
GUI

Image
Panel
Schritt-
antwort
vermes-
sen

Image
Referen-
zen

Image
Butt-
tons PI-,
PID-T1-
Regler

Check:
korrekter
Begriff

Image
Panel
Phasen-
gangme-
thode

Image
Panel
rechts

Einfügen
Wert,
Referenz

Faustformeln

Chien20, ChienApper, Oppelt, Rosenberg und ZieglerNichols sind die Klassen zu den zugehörigen Faustformeln. Die abstrakte Methode `calculate()` aus der Klasse `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 Klasse `ClosedLoop` wird für jede Berechnungsart jeweils ein `Controller` erstellt. Auch diese Klasse enthält eine Methode `calculate()`. In dieser wird einerseits für alle Rechenarten die Berechnung der Schrittantwort mittels `calculateStepResponse()` ausgelöst und für die Phasengangmethode wird zusätzlich das Überschwingverhalten durch `overShootOptimizazation()` optimiert.

PhaseResponseMethod

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 aufgerufen. In dieser wird die Abhängigkeit vom Phasengang und des benötigten Winkelbereiches erstellt.

Anschliessend werden die Werte für H_s und ϕ_{iS} berechnet.

Andererseits sind da die Methoden, `calculateTnvTnk()` um T_{nk} und T_{nv} zu berechnen, `calculateKrk` um K_{rk} zu berechnen, `calculatecontrollerConf()` um eine umrechnung 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 ϕ_{iu} 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

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

der Faustformeln mittels der `setData()`-Methode zu. Jeder `closedLoop` leitet die Daten an den zugehörigen Controller weiter, der die Reglerwerte berechnet.

Entwurf

5 Tests

Entwurf

6 Schlussfolgerungen

Das Projekt wurde erfolgreich abgeschlossen und alle Punkte des ursprünglichen Auftrages erfüllt. Aus den Eingabewerten wird das dynamische Verhalten des geschlossenen Regelkreises berechnet und graphisch dargestellt. Die Auftragserweiterungen wurden eingebaut, sodass man den Regler mit der Phasengangmethode dimensionieren und die Knickfrequenz und das Überspringen manuell verändert werden kann.

Die vor der Erweiterung des Auftrags geplanten graphische Ausgabe des Amplitudengangs und der Strecke sowie die Schieberegler für das Verändern der Reglerwerte wurden weggelassen. Der Grund für diesen Entscheid war die Möglichkeit durch die hohe Rechenleistung den Regler echtzeit zu dimensionieren.

Als Erweiterung könnten die Streckenwerte ausgegeben und die Strecke gezeichnet werden. Ausserdem könnte das automatische Einlesen der Streckenwerte aus einer graphisch Form hinzugefügt werden. Das Tool würde an die eingelesene Kurve die Wendetangente legen und die Werte ablesen. Das Ablesen könnte mit der Monte-Carlo-Studie verfeinert werden.

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: _____, _____

Entwurf

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 werden abgefangen und ein Fehler zurückgegeben.
2. Lädt Werte für T_u und T_g .
3. Erstellt 50 Werte zwischen 0 und 1 für r_i .
4. Bestimmt die Ordnung der Regelstrecke.
5. Spline für r und w
6. $T(n)$ wird aus $w \cdot t_g$ berechnet.
7. Umspeichern & Sortieren

Matlab-Code

```

1 function [n,T] = p2_sani(tu,tg,p)
2
3 if tu<=0 || tg<=0
4     disp(' ');
5     error('!!!!!! unsinnige Zeiten !!!!!');
6 end;
7
8 v=tu/tg;
9 if v>0.64173
10     disp(' ');
11     error('!!!!!! Tu/Tg zu gross --> N > 8 !!!!!');
12 end;
13
14 if v<0.001
15     disp(' ');
16     error('!!!!!! Tu/Tg zu klein -->N = 1 !!!!!');
17 end;
18
19 load('p2_sani_tu_tg');
20 pause(0.1); % Pause, damit Laden vom File erfolgreich!!!!
21
22 % Berechnet mit NN=50 (r-Auflösung)
23 ri=linspace(0,1,50);
24
25 if v <= 0.103638 % abhaengig von n werden vorberechnete
26     n=2; % Datenfiles von der Festplatte geladen.
27 elseif v <= 0.218017 % 2 <= n <= 8
28     n=3; % n=1 ist trivial und fuehrt zu Abbruch
29 elseif v <= 0.319357
30     n=4;
31 elseif v <= 0.410303
32     n=5;
33 elseif v <= 0.4933
34     n=6;
35 elseif v <= 0.5700
36     n=7;
37 elseif v<=0.64173

```

```
38     n=8;
39 else
40     n=10;
41 end;
42
43 r=spline(Tu_Tg(n,:),ri,v);
44 w=spline(ri,T_Tg(n,:),r);
45 T(n)=w*tg;
46
47
48 for i=n-1:-1:1,                % Umspeicher, damit gleiche Reihenfolge wie bei Hudzovik
49     T(i)=T(n)*r^(n-i);
50 end;
51
52 % Plots der Schrittantworten
53 if p==1
54     TT=4*(tg);
55     t=linspace(0,TT,2500);
56     za=1;
57     n1=conv([T(1) 1],[T(2) 1]);
58     for k=3:n
59         nen1=conv(n1,[T(k) 1]);
60         n1=nen1;
61     end;
62     nens=n1;
63     step(za,nens,'k'); grid on;
64     hold on;
65     %wendeptk(T);
66     hold off;
67 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:

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

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

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

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

Matlab-Code

```

1 function [ t_nk, t_vk, k_rk ] = p2_bodekonf(t_n, t_v, t_p, k_r ,reglertyp )
2
3 if (reglertyp = 1) %I-Regler
4     error('!!!! Nicht implementierter Reglertyp !!!!');
5 end;
6
7 if (reglertyp = 2) %PI-Regler
8     t_nk=t_n;
9     k_rk=k_r;
10    t_vk=0;
11 end;
12
13 if (reglertyp = 3) %PID-Regler
14     epsilon=sqrt(1-(4*t_n*(t_v-t_p))/(t_n+t_p)^2);
15     t_nk=0.5*(t_n+t_p)*(1+epsilon);
16     k_rk=0.5*k_r*(1+t_p/t_nk)*(1+epsilon);
17     t_vk=0.5*(t_n+t_p)*(1+epsilon);
18 end;

```

A.3 Umrechnung von bodekonformer in reglerkonforme Darstellung

Input

T_p	Periodendauer
T_{nk}	Nachstellzeit
T_{vk}	Vorhaltezeit
K_{rk}	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

```

1 function [ t_n, t_v, k_r ] = p2_reglerkonf(t_nk, t_vk, t_p, k_rk ,reglertyp )
2 if (reglertyp = 1) %I-Regler
3     error('!!!! Nicht implementierter Reglertyp !!!!');
4 end;
5
6 if (reglertyp = 2) %PI-Regler
7     t_n=t_nk;
8     k_r=k_rk;
9     t_v=0;
10 end;
11
12
13 if (reglertyp = 3) %PID-Regler
14     k_r=k_rk*(1+t_vk/t_nk);
15     t_n=t_nk+t_vk-t_p;
16     t_v=(t_nk*t_vk)/(t_nk+t_vk-t_p)-t_p;
17 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

Zähler	Reeller Zähler
Nenner	Reeller Nenner

Algorithmus

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

Matlab-Code

```

1 function [Zah_r, Nen_r] = p2_UTFRegler(t_nk, t_vk, t_p, k_rk, Reglertyp)
2
3 % PI Regler
4 if (Reglertyp == 2)
5     Zah_r = k_rk*[t_nk 1];
6     Nen_r = [t_nk 0];
7 end;
8
9 %ID-Regler
10 if (Reglertyp == 3)
11     Zah_r = k_rk * p2_xdiskConv([t_vk 1],[t_nk 1]);
12     Nen_r = [t_nk 0];
13 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.

korrekt?
Oder ein-
fach reel-
le Koeffi-
zienten?

Matrix

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:

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

$$T_n = 2 \cdot T_u$$

$$T_v = 0.42 \cdot T_u$$

Matlab-Code

```

1 function [k_p, t_n, t_v] = p2_ffoppelt(t_u,t_g,k_s,reglertyp)
2
3 if reglertyp <2
4     error('!!!! Nicht implementierter Reglertyp !!!!');
5 elseif reglertyp <= 2 %PI Regler
6     k_p=(0.8/k_s)*(t_g/t_u);
7     t_n=3*t_u;
8     t_v=0;
9 elseif reglertyp <= 3 %PID Regler
10    k_p=(1.2/k_s)*(t_g/t_u);
11    t_n=2*t_u;
12    t_v=0.42*t_u;
13 else
14    error('!!!! Reglertyp nicht berechenbar !!!!')
15 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:

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

$$T_n = 2 \cdot T_u$$

$$T_v = 0.45 \cdot T_u;$$

Matlab-Code

```

1 function [k_p, t_n, t_v] = p2_ffrosenberg(t_u,t_g,k_s,reglertyp)
2
3 if reglertyp <2
4     error('!!!! Nicht implementierter Reglertyp !!!!');
5 elseif reglertyp == 2 %PI Regler
6     k_p=(0.91/k_s)*(t_g/t_u);
7     t_n=3.3*t_u;
8     t_v=0;
9 elseif reglertyp == 3 %PID Regler
10    k_p=(1.2/k_s)*(t_g/t_u);
11    t_n=2*t_u;
12    t_v=0.45*t_u;
13 else
14    error('!!!! Reglertyp nicht berechenbar !!!!')
15 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

```

1 function [k_p, t_n, t_v] = p2_ffziegler(t_u,t_g,k_s,reglertyp)
2
3 if reglertyp <2
4     error('!!!! Nicht implementierter Reglertyp !!!!');
5 elseif reglertyp == 2 %PI Regler
6     k_p=(0.9/k_s)*(t_g/t_u);
7     t_n=3.33*t_u;
8     t_v=0;
9 elseif reglertyp == 3 %PID Regler
10    k_p=(1.2/k_s)*(t_g/t_u);
11    t_n=2*t_u;
12    t_v=0.5*t_u;
13 else
14    error('!!!! Reglertyp nicht berechenbar !!!!')
15 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:

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

$$T_n = 2.4 \cdot T_u$$

$$T_v = 0.42 \cdot T_u;$$

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

```
1 function [k_p, t_n, t_v] = p2_ffchien(t_u,t_g,k_s,reglertyp,ueberschwingen)
2
3 if reglertyp <2
4     error('!!!! Nicht implementierter Reglertyp !!!!');
5 elseif reglertyp == 2 %PI Regler
6     if ueberschwingung == 0 % 0% Ueberschwingen
7         k_p=0.35*t_g/(k_s*t_u);
8         t_n=1.2*t_u;
9         t_v=0;
10    else % 20% Ueberschwingen
11        k_p=0.7*t_g/(k_s*t_u);
12        t_n=2.3*t_u;
13        t_v=0;
14    end;
15 elseif reglertyp == 3 %PID Regler
16     if ueberschwingung == 0 % 0% Ueberschwingen
17         k_p=0.95*t_g/(k_s*t_u);
18         t_n=2.4*t_u;
19         t_v=0.42*t_u;
20     else % 20% Ueberschwingen
21         k_p=1.2*t_g/(k_s*t_u);
22         t_n=2*t_u;
23         t_v=0.42*t_u;
24     end;
25 else
26     error('!!!! Reglertyp nicht berechenbar !!!!')
27 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

```
1 public static double diskDiff(double[] x, double[] y, int index) {
2     if (index > 0 & index < x.length - 1) {
3         double diff2 = (y[index + 1] - y[index]) / (x[index + 1] - x[index]);
4         double diff1 = (y[index] - y[index - 1]) / (x[index] - x[index - 1]);
5         double diff = (diff1 + diff2) / 2;
6         return diff;
7     } else if (index == 0)
8         return (y[index + 1] - y[index]) / (x[index + 1] - x[index]);
9     else if (index == x.length)
10        return (y[index] - y[index - 1]) / (x[index] - x[index - 1]);
11    else
12        return 0;
13 }
```

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

```

1 public static double[][] schrittIfft(double[] zah, double[] nen, double fs, int n) {
2
3     double T = 1 / fs; // Periode
4     Complex[] H;
5
6     // Frequenzachse berechnen
7     double[] w = linspace(0.0, fs * Math.PI, n / 2); // Kreisfrequenz
8
9     // Frequenzgang berechnen
10    H = freqs(zah, nen, w);
11
12    // Symmetrischen Vektor fuer Ifft erstellen:
13    Complex[] tmp = new Complex[H.length];
14    tmp = colonColon(H, (n / 2) - 1, -1, 1);
15
16    for (int i = 0; i < tmp.length; i++) {
17        tmp[i] = tmp[i].conjugate();
18    }
19
20    Complex x = new Complex(0);
21    H = concat(colonColon(H, 0, 1, (n / 2) - 1), new Complex[] { x }, tmp);
22
23    // Impulsantwort berechnen
24    Complex[] h; // = new Complex[H.length];
25    FastFourierTransformer f = new FastFourierTransformer(DftNormalization.STANDARD);
26    h = f.transform(H, TransformType.INVERSE);
27
28    // Realteil von h extrahieren.
29    double[] hReal = new double[h.length];
30
31    for (int i = 0; i < h.length; i++) {

```

```
32     hReal[i] = h[i].getReal();
33 }
34
35 // Schrittantwort berechnen
36 // double[] y = Calc.diskConvOnes(hReal, n);
37 double[] y = new double[n];
38 y[0] = hReal[0];
39 for (int i = 1; i < y.length; i++) {
40     y[i] = y[i - 1] + hReal[i];
41 }
42
43 // Resultate ausschneiden. Halbiert die Laenge von y.
44 double[] yShort = colonColon(y, 0, 1, (int) ((y.length / 2) - 1));
45
46 // Zeitachse generieren:
47 double[] t;
48 t = linspace(0.0, (yShort.length - 1) * T, yShort.length);
49
50 // Fuer Output zusammensetzen:
51 double[][] res = new double[2][yShort.length];
52
53 for (int j = 0; j < res[0].length; j++) {
54     res[0][j] = yShort[j];
55 }
56 for (int i = 0; i < res[1].length; i++) {
57     res[1][i] = t[i];
58 }
59 return res;
60 }
```

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

```

1 private void overShootOptimization() {
2     double max = yt[0][Calc.diskMax(yt[0])];
3     PhaseResponseMethod phaseResponseMethod = (PhaseResponseMethod) controller;
4     double Krk = phaseResponseMethod.getControllerValues()[PhaseResponseMethod.KrkPOS];
5     if (max > controller.overShoot / 100.0 + 1.0) {
6         while (max > controller.overShoot / 100.0 + 1.0) {
7             Krk = phaseResponseMethod.getControllerValues()[PhaseResponseMethod.KrkPOS];
8             phaseResponseMethod.setKrk(Krk / 1.05);
9             calculateStepResponse();
10            max = yt[0][Calc.diskMax(yt[0])];
11        }
12    } else {
13        while (max < controller.overShoot / 100.0 + 1.0 & Krk < 1000) {
14            Krk = phaseResponseMethod.getControllerValues()[PhaseResponseMethod.KrkPOS];
15            phaseResponseMethod.setKrk(Krk * 1.05);
16            calculateStepResponse();
17            max = yt[0][Calc.diskMax(yt[0])];
18        }
19    }
20 }

```


A.12 Berechnung von T_{vk} und T_{nk}

`calculateTnkTvK` bestimmt T_{nk} und T_{vk} . Kern des Algorithmus ist die Automatisierung der Berechnung von β (siehe Anhang B für die manuelle Rechnung);

Input

Keine Eingabewerte

Output

Keine Rückgabewerte

Algorithmus

1. Bestimmen der Steigung der Strecke bei Frequenz ω_{pid}
2. Unteren Wert für β festlegen: 10^{-12} . Wert muss grösser als null, aber sehr klein sein, damit der Rest des Algorithmus funktioniert.
3. Obere Grenze für β auf 1 legen.
4. Über 20 Iterationen folgende Arbeitsschritte ausführen:
 - (a) Aktuellen Testwert für β definieren: $\beta_{oben} - \beta_{unten} \cdot 0.5 + \beta_{unten}$.
 - (b) Mit diesem Wert für β nun T_{nk} und T_{vk} berechnen gemäss Gleichung 21.
 - (c) T_{nk} und T_{vk} in die Reglergleichung (siehe Gleichung 24) einsetzen. Man beachte, dass der Algorithmus nicht symbolisch rechnet, sondern mit Werte-Arrays für die Frequenzachse und Funktionswerte der Übertragungsfunktion.
 - (d) Steigung des Phasengangs des Reglers an der Stelle ω_{pid} bestimmen, aufsummieren mit der Steigung der Strecke (bereits bekannt) zur Steigung des Phasengangs des offenen Regelkreises (siehe Gleichung 18).
 - (e) Falls die Steigung φ_o an der Stelle ω_{pid} kleiner ist als $-\frac{1}{2}$, muss β vergrössert werden. In diesem Fall die untere Grenze β_{unten} auf den aktuellen Testwert β setzen.
 - (f) Falls die Steigung φ_o an der Stelle ω_{pid} grösser ist als $-\frac{1}{2}$, muss β verkleinert werden. Daher neue Obergrenze β_{oben} auf den aktuellen Testwert β setzen.

Java-Code

```

1  private void calculateTnkTvK() {
2      double dPhiS = Calc.diskDiff(omega, phiS, omegaControllerIndex);
3      double beta_u = Math.pow(10, -12);
4      double beta_o = 1;
5      double beta;
6      double dPhiR,
7      dPhi0;
8
9      for (int iteration = 0; iteration < 20; iteration++) {
10         beta = (beta_o - beta_u) / 2 + beta_u;
11         TvK = 1 / (omegaController / beta);
12         TnK = 1 / (omegaController * beta);
13
14         for (int i = 0; i < pointNumber; i++) {
15             Hr[i] = new Complex(1, omega[i] * TnK).multiply(
16                 new Complex(1, omega[i] * TvK)).divide(
17                     new Complex(0, omega[i] * TnK));
18             Ho[i] = Hs[i].multiply(Hr[i]);
19         }
20         phiR = Calc.ComplexAngleUnwrapped(Hr);
21
22         dPhiR = Calc.diskDiff(omega, phiR, omegaControllerIndex);
23         dPhi0 = dPhiS + dPhiR;

```

```
24
25     if (dPhi0 * omegaController < -0.5) {
26         beta_u = beta;
27     } else {
28         beta_o = beta;
29     }
30 }
31 }
```

B Manuelle Berechnung des Hilfsparameteres β

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

Zur Rekapitulation eine kurze Wiederholung der Ausgangslage:

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

Diese Werte eingesetzt in Gleichung 14 ergeben:

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

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:

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

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.

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

Die Ableitung des Arkustangens ist:

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

Mit

$$x(j\omega) = \omega^{-1} \cdot -0.2686 \text{ s}^{-1} - \omega \cdot 0.5960 \text{ s} \quad (45)$$

folgt

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

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.

Eine Beschreibung des in unserem Tool verwendeten Algorithmus ist in Anhang A.12 zu finden.

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_\(Automatisierungstechnik\)](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.