Fachhochschule Nordwestschweiz
Hochschule für Technik

Reglerdimensionierung mittels Phasengangmethode

Fachbericht

9. 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 essentiell. Die zu regelnde Strecke kann nur durch richtig eingestellte Regelwerte wie gewünscht beeinflusst werden.

Dieses Projekt hat sich zum Ziel gesetzt, eine Software zu entwickeln, welche 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. Die Software 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 gängiger Faustformeln implementiert. Zum Vergleich können die Schrittantworten der Faustformeln im gleichen Plot angezeigt werden.

Für die Phasengangmethode werden gleichzeitig drei Graphen abgebildet. Einer verwendet den Standardwert für φ_r gemäss Zellweger, die anderen beiden basieren auf Benutzereingaben durch einen Schieberegler.

Ein weiterer Schieberegler ermöglicht das Festlegen des gewünschten maximalen Überschwingens. Die Software optimiert den zu dimensionierenden Regler automatisch in Echtzeit, sodass der geschlossene Regelkreis das gewünschte Verhalten aufweist. Dies ermöglicht enorme Zeitgewinne verglichen mit der manuellen Durchführung der Phasengangmethode.

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 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 vermessen). 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. Zeidweger, *Regelkreise und Regelungen*, Vorlesungsskript.

[2] J. Zeidweger, *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

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 Jakob Zellweger entwickelten Phasengangmethode dimensioniert. Zum Vergleich soll der entsprechende Regler ebenfalls mittels verschiedener Faustformeln berechnet werden.

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. Zudem kann das Überschwingen nur grob gewählt werden. Das Tool soll ausgehend von drei Parametern aus der Schrittantwort der Strecke (Verstärkung K_s, Verzögerungszeit T_u, Anstiegszeit T_g) 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 vertraute Regelungstechniker einfach zurechtfinden.

Die erforderlichen Algorithmen wurden zuerst in Matlab als Prototypen implementiert und anschliessend vollständig in Java übersetzt. Um optimale Wartbarkeit, Übersichtlichkeit und Modularität des Codes zu gewährleisten, ist die Software gemäss *Model-View-Controller*-Pattern aufgebaut.

Nach der Implementierung in Matlab wurde klar, dass die Berechnung durch die hohe Rechenleistung sehr schnell durchgeführt werden kann und somit eine Berechnung des Verhaltens des geschlossenen Regelkreises anhand dieser Methode möglich ist. Die Schrittantworten können in Echtzeit angezeigt werden. Damit eröffneten sich neue Möglichkeiten, was dazu benutzt wurde, zwei Schieberegler zur Anpassung des dimensionierten Reglers in Echtzeit zu implementieren. Mit dem einen kann das gewünschte Überschwingen spezifiziert werden, der andere Schieber dient zur weiteren Anpassung der Kurvenform des geschlossenen Regelkreises gemäss den Wünschen des Benutzers.

Der Bericht gliedert sich in drei Hauptteile: In einem ersten Teil werden einige theoretische Grundlagen der Regelungstechnik erläutert (Kapitel 2). Kapitel 3 geht genauer auf die eigentliche Dimensionierung von Reglern ein, indem ein PI- und ein PID-Regler mittels Faustformeln und Phasengangmethode für eine Beispielstrecke dimensioniert werden. Kapitel 4 und 5 beschreiben die Software und die zugehörigen Testverfahren; wichtige Algorithmen sind dabei in Anhang B genauer dokumentiert. Es werden die Benutzeroberfläche und das Zusammenspiel der verschiedenen Module der Software genauer betrachtet.

2 Grundlagen der Regelungstechnik

In diesem Kapitel wird auf die Grundlagen der Regelungstechnik eingegangen und die wichtigsten Grundbegriffe erläutert.

2.1 Die Steuerung

Unter einer Steuerung versteht man eine offen Wirkungskette wie in Abbildung 1, das heisst die Wirkglieder sind kettenähnlich aufgereiht und besitzen keine Rückkopplung. Die Steuerkette wird genau für eine Steuerung ausgelegt und kann nur auf Steuergrössen reagieren. Ohne die Rückkopplung wird das Ausgangsignal nicht mit dem Eingangssignal verglichen und es können keine Korrekturen vorgenommen werden.

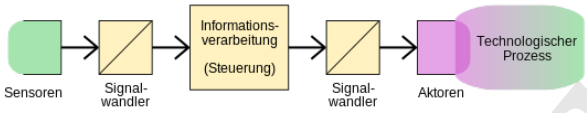

Abbildung 1: Steuerung

2.2 Der geschlossene Regelkreis

Die Aufgabe eines geschlossenen Regelkreises (Abbildung 2) 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. Daraus folgt auch die wichtigste Bedienung für die Schrittantwort ein geschlossenen Regelkreis heisst, dass der Regelfehler, die Differenz zwischen Ist- und Sollwert, möglichst schnell gleich Null oder möglichst klein ist. Die einzelnen Elemente werden in der Aufzählung weiter unten erklärt.

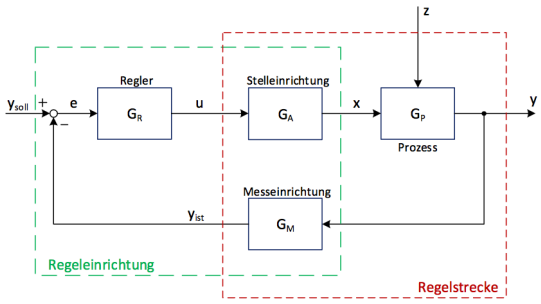

Abbildung 2: Geschlossener Regelkreis

- 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 seinen 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 $t \to \infty$ 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.

Die Schrittantwort des geschlossenen Regelkreises

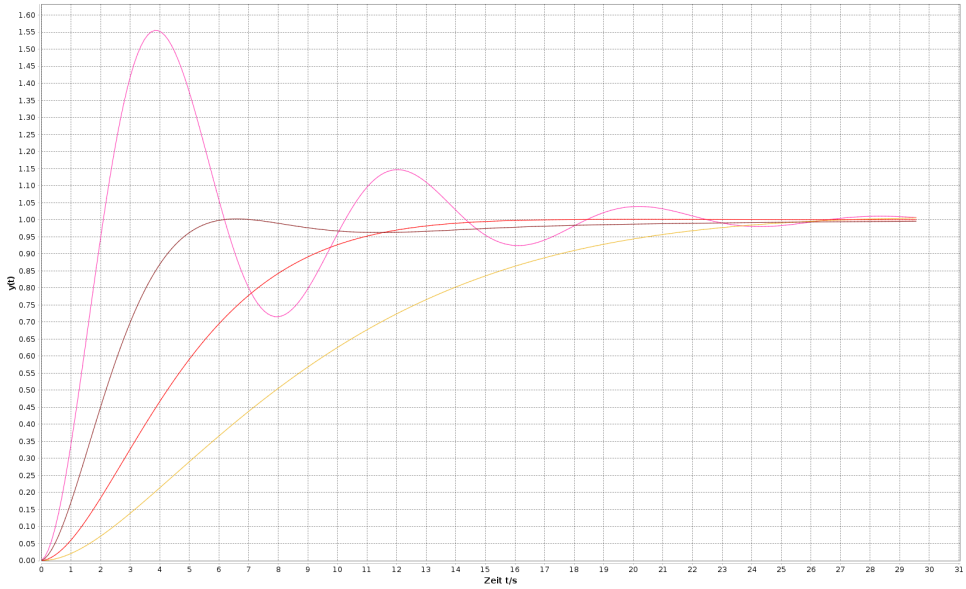

Abbildung 3: Schrittantworten verschiedener Charakteristiken

Als Schrittantwort eines geschlossenen Regelkreises wird der zeitliche Verlauf des Ausgangssignals $y(t)$ bezeichnet, welches entsteht wenn y_{soll} von 0 auf 1 springt. In der Abbildung 3 werden drei verschiedene Schrittantworten gezeigt. 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 Zustands kein erneutes Schwingen aufweisen.
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. Die pinke Kurve in Abbildung 3 zeigt eine Schrittantwort, welche vor dem Erreichen des eingeschwungenen Zustands überschwingt.
3. Die Schrittantwort muss für ein $t \to \infty$ 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. Die gelbe Kurve in Abbildung 3 zeigt eine zu langsame Schrittantwort.

Die Berechnung der Schrittantwort des geschlossenen Regelkreises wird in unserem Tool mittels der inversen schnellen Fourier-Transformation durchgeführt, genauere Informationen dazu sind in Abschnitt 3.6 und Anhang B.10 zu finden.

2.3 Regelstrecken

In der Regelungstechnik wird die zu regelnde Strecke als Regelstrecke bezeichnet. Die zu regelnde Strecke ist zum Beispiel die Temperatur im Raum oder die Luftfeuchtigkeit in der Sauna. 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 ist. Die Ordnung der Strecke wird durch die Variable n angegeben.

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 die Verzögerung jedoch sehr klein spricht man von einer P-Strecke. Das Verhalten der Strecke ist in ihrem Blockschaltbild (Abbildung 4) symbolisch dargestellt. Der Proportionalitätsfaktor wird mit K_p abgekürzt. Wird $K_p < 1$ wirkt K_p nicht mehr verstärkend sondern abschwächend.

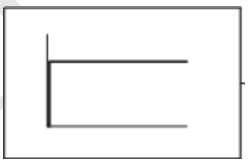

Abbildung 4: Blockschaltbild von P-Strecke

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. Im Blockschaltbild (Abbildung 5) wird die Totzeit durch eine Verzögerung am Anfang gekennzeichnet.

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

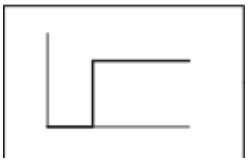

Abbildung 5: Blockschaltbild von Strecke mit Totzeit

I-Regelstrecke

Die I-Regelstrecke antwortet auf eine Stellgrössenänderung mit einer stetigen Änderung in steigende oder fallende Richtung. Die Begrenzung dieses Vorganges ist mit den systembedingten Schranken gegeben. Die Integrierzeit T_i ist ein Mass für die Anstiegsgeschwindigkeit der Regelgrösse und das Blockschaltbild (Abbildung 6) zeigt das Verhalten sinnbildlich.

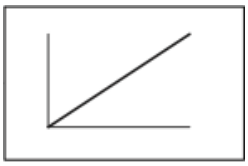

Abbildung 6: Blockschaltbild von I-Strecke

2.4 Regler

Die Aufgabe eines Reglers besteht darin 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.

P-Regler

Der P-Regler regelt der Stellgrösse x proportional zur Regeldifferenz e. Aus diesem Grund reagiert der P-Regler ohne Verzögerung auf eine Ausgangsänderung. Das Proportionale Verhalten ist in seinem Blockdiagramm (Abbildung 7) dargestellt.

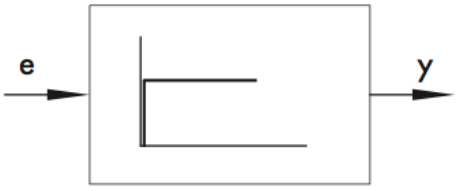

Abbildung 7: Blockschaltbild des P-Reglers

I-Regler

Der I-Regler ist ein integrierender Regler, welcher Regelabweichungunen zu jedem Betriebspunkt ausregelt. Die Steigung des I-Reglers hängt von der Reglerabweichung und der Integrationszeit T_i ab. Hier gilt die Regel: Je grösser die Integrationszeit ist, umso langsamer steigt die Kurve. Das Verhalten des I-Reglers ist in seinem Blockschaltbild (Abbildung 8) dargestellt.

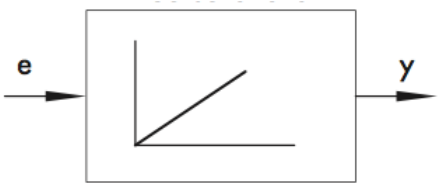

Abbildung 8: Blockschaltbild des I-Reglers

D-Regler

Der Differentialregler bildet seine Stellgrösse x aus der Geschwindigkeit der Änderung der Regeldifferenz. Aus diesem Grund erzeugt der D-Regler auch bei kleiner Reglerdifferenz eine grosse Stellamplitude und reagiert schneller als der P-Regler. Dieses Verhalten wird in seinem Blockschaltbild (Abbildung 9) symbolisch dargestellt.

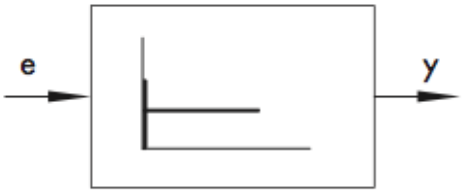

Abbildung 9: Blockschaltbild des D-Reglers

PI-Regler

Der PI-Regler besteht aus einer Parallelschaltung eines P- und eines I-Reglers (Abbildung **??**). Durch diese Kombination werden die Nachteile beider Regler kompensiert und die Vorteile (schnell, stabil) hervorgehoben. Sein Verhalten wird bildlich im Blockschaltbild in Abbildung 11 dargestellt.

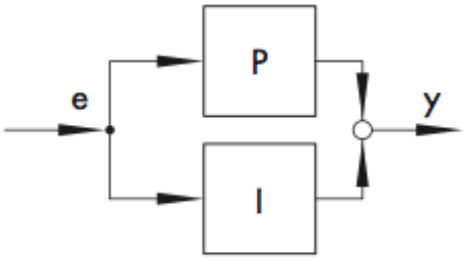

Abbildung 10: Parallelschlatung von P-Regler und I-Regler

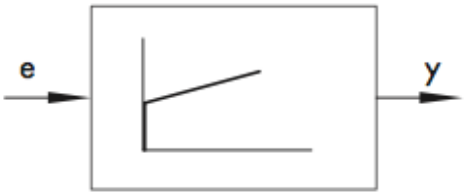

Abbildung 11: Blockschaltbild von PI-Regler

PID-Regler

Wird dem PI-Regler ein D-Anteil parallel geschaltet (Abbildung 12), 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 (Abbildung 13) anschaulich. Der PID-Regler ist geeignet für Regelstrecken höherer Ordnung, welche möglichst schnell und ohne bleibende Regelabweichungen geregelt werden müssen.

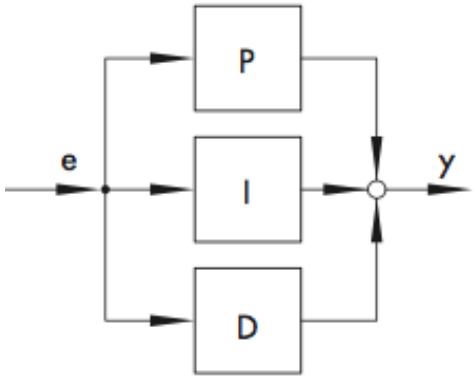

Abbildung 12: Parallelschaltung von P-, I-, und D-Regler

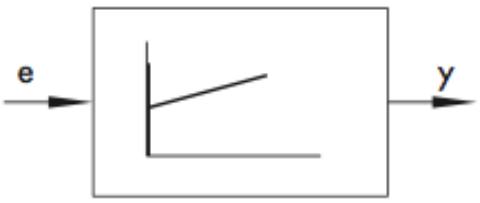

Abbildung 13: Blockschaltbild des PID-Reglers

3 Fachlicher Hintergrund zur Reglerdimensionierung

Das Kernstück dieser Arbeit und des zugehörigen Softwaretools stellt die sogenannte *Phasengangmethode 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 umgesetzt. Als Vergleich wertet die Software ebenfalls einige gängige 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 die 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 3.6)

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 bereits 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].

Bei der Dimensionierung eines Reglers mittels Phasengangmethode erlaubt das Tool dem Benutzer, das Überschwingen auf einen Zielwert zwischen 0.1% und 30% einzustellen. Es optimiert den resultierenden Regler entsprechend, um dieser Vorgabe möglichst nahe zu kommen. Dazu wird die Reglerverstärkung K_{rk} solange angepasst, bis ein passendes Resultat erreicht ist. Dieser direkte Eingriff folgt aus der Erkenntnis, dass in der Berechnung mit der Phasengangmethode das Überschwingen nur von K_{rk} abhängt.

3.1 Frequenzgang der Regelstrecke

Als Ausgangspunkt der Reglerdimensionierung dient die Schrittantwort der Strecke. Durch Einzeichnen der Wendetangente[I] ergeben sich Schnittpunkte der Wendetangente mit der Zeitachse $[T_u, 0]$ und mit dem Zielwert $[T_u + T_g, K_s]$. Es können nun also die Verzögerungszeit T_u und die Anstiegszeit T_g aus Abbildung 14 abgelesen werden.

Wir werden in diesem Bericht folgende Strecke als Beispiel nehmen:

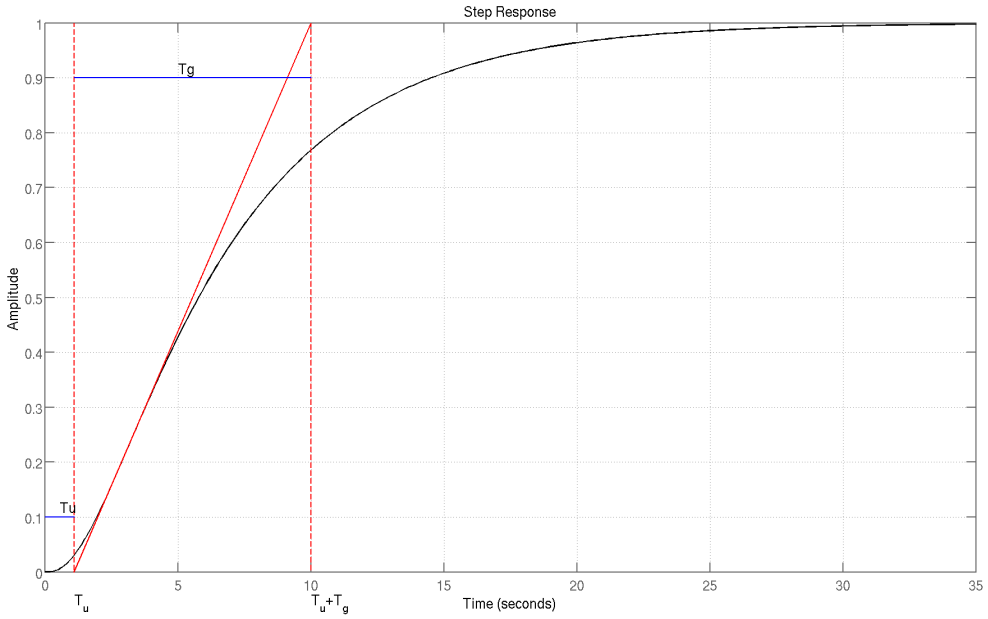

Abbildung 14: Schrittantwort der Beispielstrecke (schwarz), Wendetangende (rot), T_u und T_g (blau)

Ausmessen der Schrittantwort ergibt:

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

Da die Reglerdimensionierung mit der Phasengangmethode vom Frequenzgang einer Strecke ausgeht und nicht von deren Schrittantwort, wird aus den obigen Werten nun der Frequenzgang der Strecke bestimmt. Dies erledigt die Methode `p_sani()`[III], welche uns die Werte für die Übertragungsfunktion der Strecke liefert. In unserem Fall ergibt dies folgendes Polynom:

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

[II] Abbildung 14 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

[III] Die Methode `p_sani()` wurde zu Beginn des Projektes in einer Matlab-Implementation vom Auftraggeber 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).

Genauere Informationen zur Funktionsweise von `p_sani()` sind Anhang B.1 zu entnehmen.

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

Mit einem geeigneten Tool kann man sich den dazugehörigen Plot (Abbildung 15) erstellen lassen.

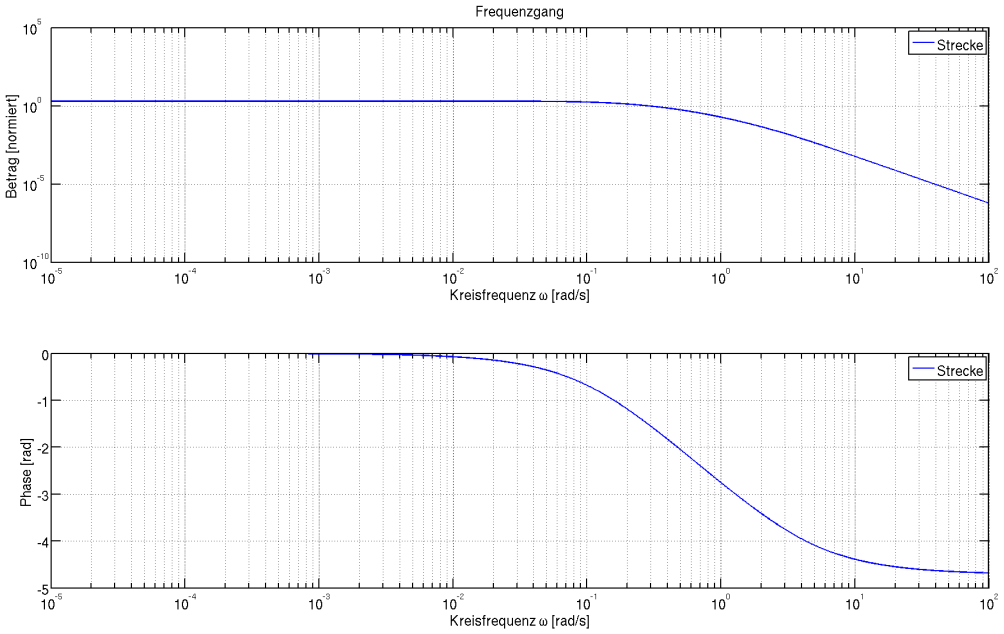

Abbildung 15: Frequenzgang der Strecke, mit Amplitudengang und Phasengang

Somit ist der Frequenzgang der Strecke bekannt und man hat alle erforderlichen Informationen, um den Regler mit der Phasengangmethode zu dimensionieren.

3.2 Reglerdimensionierung mittels Faustformeln

In der Praxis stehen für die Dimensionierung von Reglern einfache Berechnungsformeln zur Verfügung (siehe Tabelle 1). 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]	$3 \cdot T_u$	$\frac{0.8}{K_s} \cdot \frac{T_g}{T_u}$	$2 \cdot T_u$	$0.42 \cdot T_u$	$\frac{1.2}{K_s} \cdot \frac{T_g}{T_u}$
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 und T_g in diese Formeln ein, ergeben sich die Werte aus Tabelle 2.

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	0.52 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

Ein Vergleich mit den Resultaten der Phasengangmethode, sowohl mit Handrechnung wie auch mittels unseres Tools, ist in Tabelle 6 auf 31 zu finden.

Die von der Software verwendeten Algorithmen zur Implementation diesen Faustformeln sind in Anhang B.5, B.6, B.7 und B.8 genauer beschrieben.

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
$A_s(j\omega) = \lvert H_s(j\omega)\rvert$	Amplitudengang der Regelstrecke
$\varphi_s(j\omega) = arg(H_s(j\omega))$	Phasengang der Regelstrecke
$H_r(j\omega)$	Übertragungsfunktion des Reglers
$A_r(j\omega) = \lvert H_r(j\omega)\rvert$	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
$A_o(j\omega) = \lvert H_o(j\omega)\rvert$	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 dieses Abschnittes 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 gemäss Gleichung 3 die Frequenz ω_{pi} bestimmt, für welche die Phase der Strecke $-90°$ beträgt, ersichtlich in Abbildung 16[IV].

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

Wie man aus Abbildung 16 ablesen kann, liegt dieser Wert für ω_{pi} in unserem Beispiel bei ungefähr $0.3\,\mathrm{s}^{-1}$. Die Kontrollrechnung mittels Matlab ergibt:

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

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

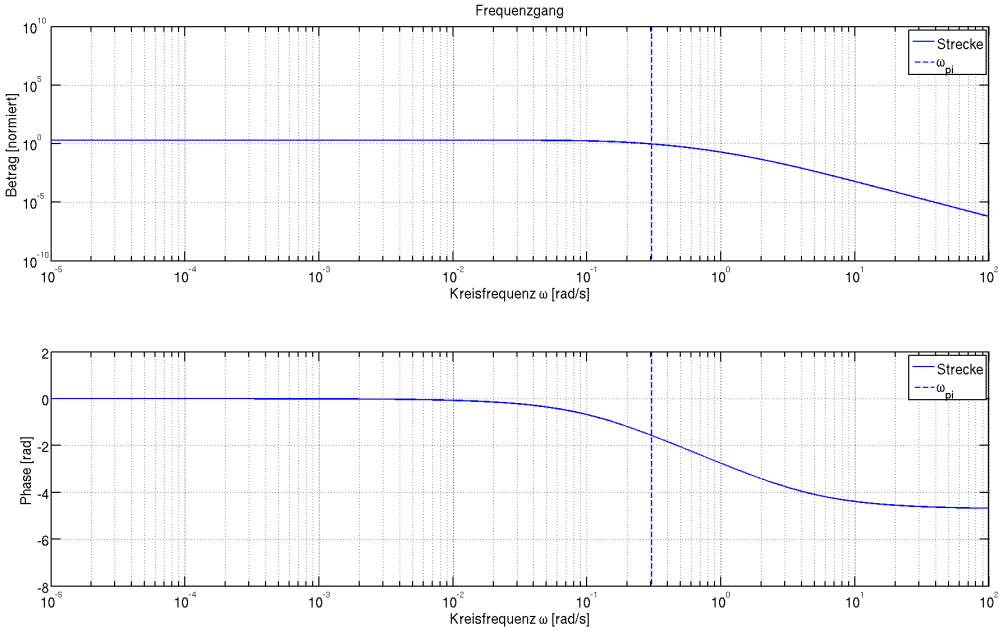

Abbildung 16: Amplituden- und Phasengang der Strecke mit ω_{pi} eingetragen (vertikale gestrichelte Linie).

2 Bestimmung von $\mathbf{T_{nk}}$

Damit kann nun T_{nk} direkt berechnet werden[V]:

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

3 Bestimmung der Durchtrittsfrequenz $\omega_\mathbf{d}$

Die Durchtrittsfrequenz ist die Frequenz, bei der die betrachtete Übertragungsfunktion $H(j\omega)$ eine Verstärkung von $0\,\mathrm{dB} = 1$ 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[VI]. Nach dem Festlegen der Durchtrittsfrequenz (in unserem Beispiel werden wir 16.3% anstreben) 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	$-103.7°$	$-128.5°$	$-135°$

[V] 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.

[VI] 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.

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 H_o des offenen Regelkreises identifiziert werden.

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

Von besonderem Interesse ist der Phasengang $\varphi_o(j\omega)$ dieser Übertragungsfunktion. Wie oben festgelegt, 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 17 kann diese Beziehung graphisch verifiziert werden.

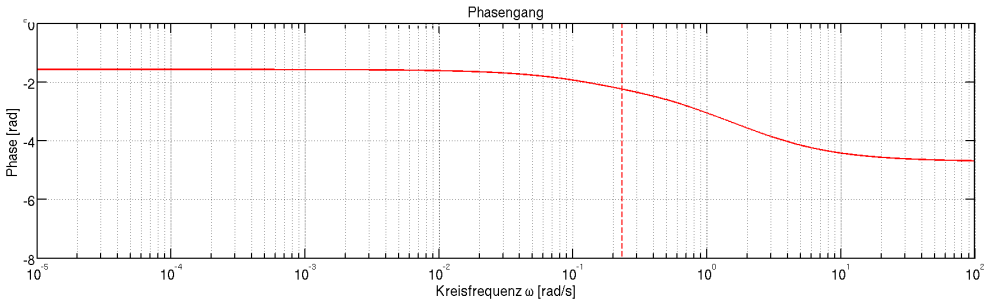

Abbildung 17: Phasengang $\varphi_o(j\omega)$ des offenen Regelkreises mit eingetragener Durchtrittsfrequenz ω_d (vertikale gestrichelte Linie). Wie man sieht, weist der offene Regelkreis unseres Beispiels bei dieser Kreisfrequenz eine Phase von $-128.5°$ auf (etwa $-2.24\,\mathrm{rad}$).

Dies ergibt für die Durchtrittsfrequenz:

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

4 Bestimmung der Reglerverstärkung $\mathbf{K_{rk}}$

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 18 sind die wichtigsten Werte für diesen Prozess nochmals in einer Übersicht zusammengefasst.

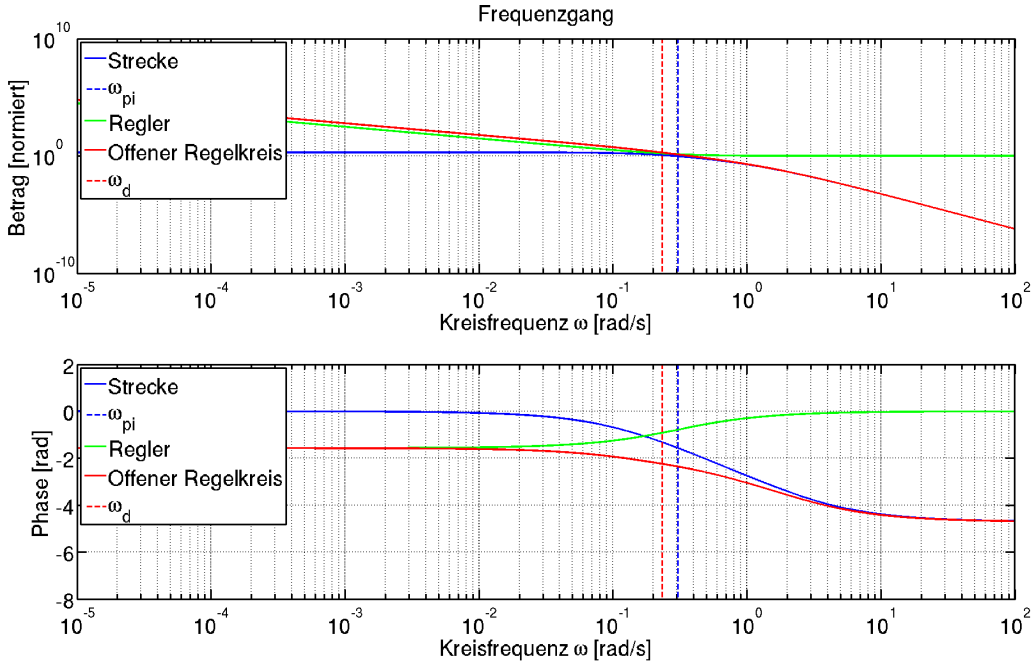

Abbildung 18: Frequenzgang des Reglers (grün), der Strecke (blau) und des offenen Regelkreises (rot).

3.4 Reglerdimensionierung mittels Phasengangmethode: PID-Regler

Ziel

Das Ziel dieses Abschnittes 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 definierten Wert aufweist, aber im Unterschied zum PI-Regler wird hier $-135°$ benutzt[VII]:

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

In unserem Beispiel ergibt dies:

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

Eine graphische Überprüfung kann anhand von Abbildung 19 auf Seite 27 durchgeführt werden.

2 Steigung des Phasengangs bei der Reglerfrequenz

Anschliessend wird die Steigung des Phasengangs φ_s der Strecke bei der Frequenz ω_{pid} berechnet. Ausgangspunkt dafür ist die Übertragungsfunktion der Regelstrecke (siehe Gleichung 1).

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

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 19 (Seite 27) von Hand überprüft werden.

[VII] Wie auch beim PI-Regler stellt diese Frequenz lediglich einen Ausgangspunkt dar und kann zur weiteren Optimierung des Resultats noch angepasst 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 Gleichung 20 erfüllt wird.

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

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

Wie in Abbildung 19 gesehen werden kann[VIII], 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 man zuerst eine "vernünftige" Annahme, zum Beispiel:

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

Mit diesem Startwert berechnet 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 zusammen mit dem Wert für ω_{pid} aus Gleichung 16 in Gleichung 14 ein. 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 A gefunden werden.

[VIII] 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.

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

Setzt man dies in Gleichung 18 ein, erhält man:

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

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 Iteration 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:

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

Diese Werte sind ebenfalls in Abbildung 19 eingetragen.

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

Die zur Berechnung von β, T_{vk} und T_{nk} benutzten Verfahren sind in Anhang B.9 respektive B.12 dokumentiert.

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 19), 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 berechnet, 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 19). Eine Nachrechnung mittels Matlab ergibt:

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

5 Bestimmung der Reglerverstärkung $\mathbf{K_{rk}}$

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 dimensioniert 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 19 die verschiedenen Frequenzgänge und Frequenzen eingetragen.

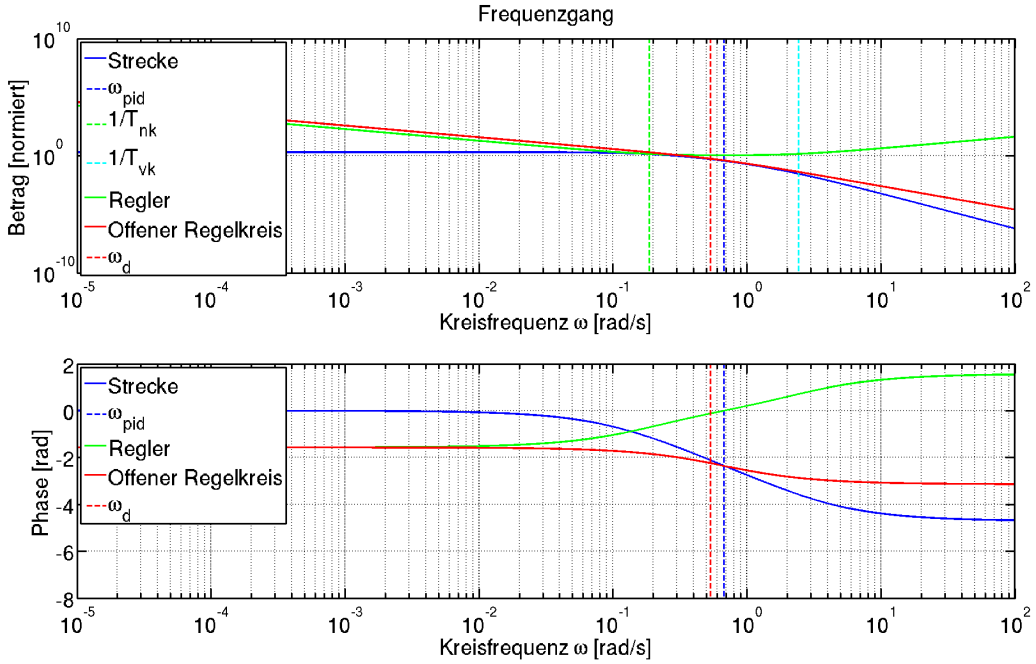

Abbildung 19: Frequenzgang der Strecke (blau), des Reglers (grün) und des offenen Regelkreises (rot). Ebenfalls eingetragen sind die Reglerfrequenz ω_{pid}, die beiden Frequenzen $\frac{1}{T_{vk}}$ und $\frac{1}{T_{nk}}$ sowie die Durchtrittsfrequenz ω_d.

3.5 Umrechnung zwischen bodekonformer und reglerkonformer Darstellung

Ein Regler kann mit unterschiedlichen 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.

Es existieren zwei hauptsächliche Gründe für die Verwendung von zwei verschiedenen Darstellungsarten: Einerseits geben gewisse Berechnungen automatisch die eine oder andere Form zurück, andererseits bieten die beiden Varianten je nach Situation bestimmte Vorteile, was das Verständnis anbelangt.

Bei der bodekonformen Darstellung wie in Gleichungen 36 und 37 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.

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

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

Die reglerkonforme Darstellung aus den Gleichungen 38 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.

$$H_{rpi} = K_r \cdot \left[1 + \frac{1}{s \cdot T_n} \right] \tag{38}$$

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

Zudem liegt das Ergebnis der Faustformeln auch immer in der reglerkonformen Darstellung vor. Für die Umrechnung zwischen den zwei Darstellungsarten ergeben sich durch einen Koeffizientenvergleich die Formeln in Tabelle 5. Für die Berechnungen in diesem Projekt wird, wenn nicht anders angegeben, mit $T_p = \frac{1}{10} \cdot T_v$ gerechnet.

Die algorithmischen Umsetzungen dieser Formeln für unsere Software sind in Anhang B.2 und B.3 zu finden.

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

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

3.6 Berechnung der Schrittantwort des geschlossenen Regelkreises

In diesem Projekt ist im speziellen die Schrittantwort de geschlossenen Regelkreises von Bedeutung. Um auf die Schrittantwort zu gelangen wurde der Weg über die IFFT (Englisch: *Inverse Fast Fourier Transform*, inverse schnelle Fourier-Transformation) gewählt, da diese relativ einfach in Java zu implementieren ist.

Für diese Berechnung sind folgende Schritte notwendig:

- Übertragungsfunktion des geschlossenen Regelkreises bilden
- bestimmen der Abstastfrequenz f_s sowie der Anzahl zu berechnender Punkte.
- diskreten Frequenzgang berechnen
- Frequenzgang für die IFFT vorbereiten
- Frequenzgang mittels IFFT rücktransformieren
- Schrittantwort aus der Impulsantwort bilden

Übertragungsfunktion des geschlossenen Regelkreises bilden

Da die Übertragungsfunktionen des Reglers sowie der Regelstrecke bekannt sind, kann aus ihnen die gesamte Übertragungsfunktion $H_g(s)$ gebildet werden. Dazu werden jeweils die Zähler- sowie die Nennerpolynome mittels diskreter Faltung zusammengeführt. Zusätzlich muss das resultierende Zählerpolynom zum Nennerpolynom addiert werden.

$$\begin{aligned} H_g(s) &= \frac{ZAH(s)}{NEN(s)} \\ &= \frac{ZAH_{Regler} \cdot ZAH_{Strecke}}{NEN_{Regler} \cdot NEN_{Strecke} + ZAH_{Regler} \cdot ZAH_{Strecke}} \end{aligned} \tag{40}$$

wobei die Zähler- und Nennerpolynome folgende Form haben:

$$\begin{aligned} ZAH_{Regler} &= b_{R_1}s^n + b_{R_2}s^{n-1} + \cdots + b_{R_{n-1}} \mid n \in \mathbb{N} \\ ZAH_{Strecke} &= b_{S_1}s^k + b_{S_2}s^{k-1} + \cdots + b_{S_{k-1}} \mid k \in \mathbb{N} \\ NEN_{Regler} &= a_{R_1}s^m + a_{R_2}s^{m-1} + \cdots + a_{R_{m-1}} \mid m \in \mathbb{N} \\ NEN_{Strecke} &= a_{S_1}s^l + a_{S_2}s^{l-1} + \cdots + a_{S_{l-1}} \mid l \in \mathbb{N} \end{aligned}$$

Bestimmen der Abtastrate fs sowie der Anzahl zu berechnender Punkte.

Für die Berechnung von f_s durften wir eine Methode verwenden, welche uns von Prof. Dr. Richard Gut zur Verfügung gestellt wurde. Diese berechnet f_s anhand der Nullstellen des Nennerpolynoms und ermittelt gleichzeitig in Abhängigkeit von f_s die Anzahl N zu berechnender Punkte, deren Anzahl stets eine Potenz von 2 ist. Daher wird an dieser Stellt nicht weiter auf diesen Schritt eingegangen; das Matlab-File kann in Anhang C gefunden werden.

Diskreten Frequenzgang berechnen

Nun werden für s genau $\frac{N}{2}$ Werte eingesetzt, welche sich geichmässig zwischen 0 und $f_s \cdot \pi$ verteilen. Dadurch entsteht ein diskreter Frequenzgang $H_g[k]$ mit $\frac{N}{2}$ Werten.

$$H_g[k] = H_g(s) \mid s = j \cdot \frac{f_s \cdot \pi}{\frac{N}{2} - 1} \cdot k \wedge 0 \leq k < \frac{N}{2} \wedge k \in \mathbb{N}_0$$

Frequenzgang für die inverse Fourier-Transformation vorbereiten

Damit nach der Transformation in den Zeitbereich ein reelles Resultat vorliegt muss der diskrete Frequenzgang $H_g[k]$ folgende Symmetriebedingung erfüllen (L ist die Länge des Arrays H_g):

$$H_g[L-k] = H_g^*[k] \mid 0 \leq k \leq L \wedge k \in \mathbb{N}_0$$

Deshalb wird $H_g[\frac{N}{2}] = 0$ gesetzt und zweite Teil von $H_g[k]$ jeweils durch die entsprechende Konjugiert-Komplexe $H^*[k]$ ergänzt.

Frequenzgang rücktransformieren

Die diskrete inverse Fourier-transformation wird gemäss folgender Formel durchgeführt:

$$h[k] = \frac{1}{N} \cdot \sum_{n=0}^{N-1} H[n] \cdot e^{j2\pi \cdot \frac{n \cdot k}{N}} | 0 \leq k < N \wedge k \in \mathbb{N}_0$$

Die in diesem Projekt verwendetete inverse schnelle Fourier-Transformation ist ein verbessertes Verfahren der oben erklärten IFFT. Dieses ermöglicht eine schnellere Berechnung durch das Zerlegen der Punkte in ihre Primfaktoren. Eine genauere Erläuterung wird an dieser Stelle nicht gemacht, da dieses Verfahren in Matlab sowie in Java als fertige Funktion verfügbar ist.

Schrittantwort aus der Impulsantwort bilden

Aus der Impulsantwort $h[k]$ kann nun in einem einfachen Verfahren die Schrittantwort $y[k]$ gebildet werden. Dabei entspricht jedes Element der Schrittantwort $y[k]$ der Summe der Elemente der Impulsantwort $h[k]$ vom ersten bis zum k-ten Element. Zusätzlich wird die zweite Hälfte der Impulsantwort weggeschnitten, um die zuvor als konjugert-komplexe Werte hinzugefügten Überbleibsel zu eliminieren.

$$y[k] = \sum_{n=0}^{k} h[n] \mid 0 \leq k < \frac{L}{2} \wedge k \in \mathbb{N}_0$$

3.7 Vergleich mit Software-Tool

Zum Vergleich mit den soeben dimensionierten PI- und PID-Reglern sind hier noch die Resultate der Dimensionierung mittels unseres Tools gezeigt. Wie zu erwarten weisen die mit dem Tool berechneten Werte für die Faustformeln keine signifikanten Abweichenungen zu den Werten in Tabelle 2 auf.

Verwendet man in den Berechnungen mittels Phasengangmethode den Standardwert für φ_r, erhält man Zahlen, die ziemlich nahe bei den von Hand berechneten Resultaten liegen. Wird jedoch von den Optimierungsmöglichkeiten des Tools Gebrauch gemacht, ändern sich die Reglerparameter und das Verhalten des zugehörigen geschlossenen Regelkreises teilweise beträchtlich.

Die Zahlenwerte sind in Tabelle 6 zusammengefasst, einige Plots zum Vergleich der Schrittantworten der zugehörigen geschlossenen Regelkreise sind in den Abbildungen 20, 21 und 22 zu sehen.

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	0.52 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
Phasengangmethode, von Hand **(16.3% Überschwingen)**	3.29 s	0.52	5.37 s	0.41 s	1.83
Phasengangmethode, Software **(15% Überschwingen)**	3.29 s	0.47	5.74 s	0.35 s	1.78
Phasengangmethode, Software φ_r Standard **(2% Überschwingen)**	3.29 s	0.17	5.74 s	0.35 s	0.76
Phasengangmethode, Software Optimierungen positiv **(2% Überschwingen)**	7.50 s	1.05	13.02 s	0.636 s	2.70
Phasengangmethode, Software Optimierungen negativ **(2% Überschwingen)**	1.62 s	0.058	5.74 s	0.35 s	0.76

Tabelle 6: Zusammenfassung und Vergleich der mit verschiedenen Methoden berechneten Regelparameter.

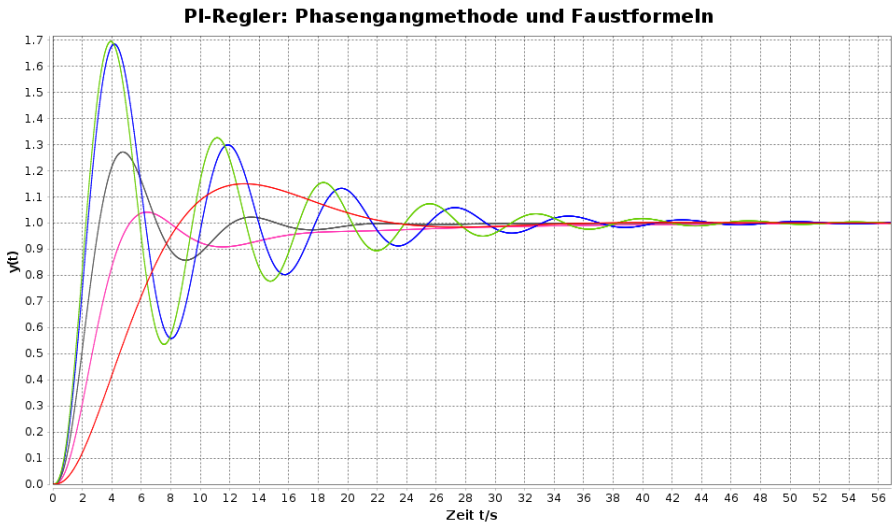

Abbildung 20: Schrittantworten des geschlossenen Regelkreises (PI-Regler): **Pink**: Chiens, Hrones, Reswick (0% Überschwingen), **dunkelgrau**: Chiens, Hrones, Reswick (20% Überschwingen), **blau**: Oppelt, **grün**: Rosenberg, **rot**: Phasengangmethode.

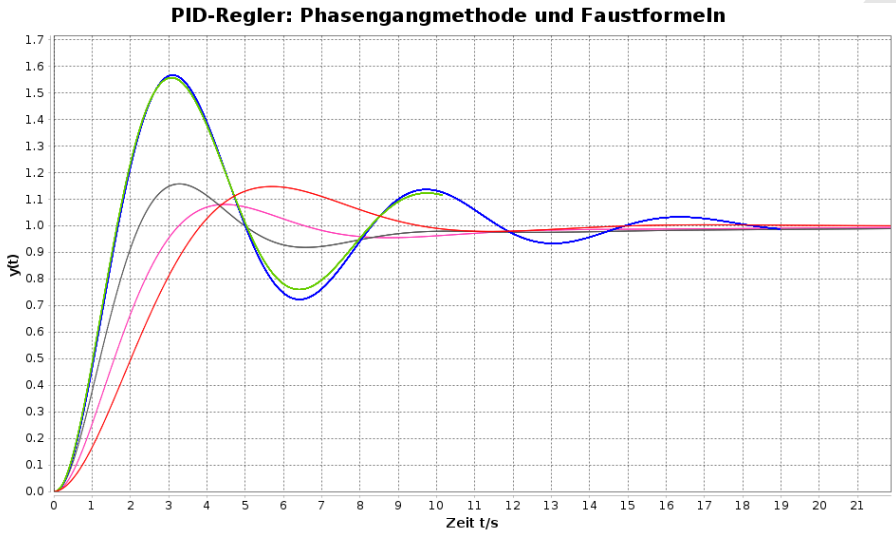

Abbildung 21: Schrittantworten des geschlossenen Regelkreises (PID-Regler): **Pink**: Chiens, Hrones, Reswick (0% Überschwingen), **dunkelgrau**: Chiens, Hrones, Reswick (20% Überschwingen), **blau**: Oppelt, **grün**: Rosenberg, **rot**: Phasengangmethode.

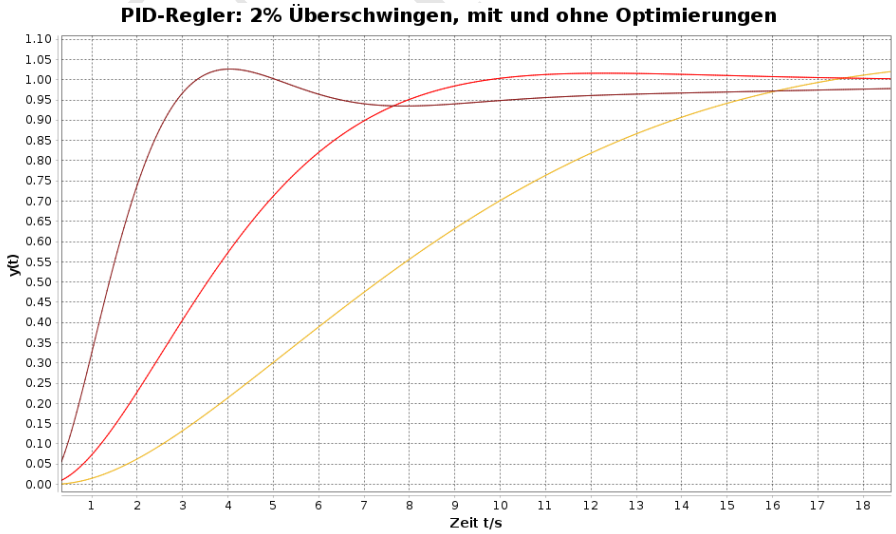

Abbildung 22: Schrittantworten des geschlossenen Regelkreises (PID-Regler), Überschwingen auf 2% begrenzt, mit Optimierungen: **rot**: Phasengangmethode (Standardwerte), **braun**: Phasengangmethode (positiv optimiert), **gelb**: Phasengangmethode (negative optimiert).

4 Software

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

4.1 MVC

Die Software wurde nach dem *MVC-Pattern* aufgebaut. Somit wurde auch das *Obersever/Observable*-Prinzip eingebunden. Das Klassendiagramm ist in Anhang D zu finden.

In der Klasse `Application` werden das `Model`, der `GUIController` sowie die `View` erzeugt und miteinander verknüpft. Dem `GUIcontroller` wird das `Model` und der `View` der `GUIController` übergeben. Somit können Eingaben auf der `View` direkt an den `GUIController` weitergegeben werden. Dieser wiederum kann direkten Einfluss auf das `Model` nehmen.

Die `View` wird als Observer von `Model` registriert. Sobald im `Model` die Auslösung der Methode `notifyObservers()` erfolgt ist, wird in der `View` die Methode `update()` aufgerufen und das ganze `Model` als Objekt übergeben. Die `View` kann nun die gewünschten Daten aus dem `Model` auslesen und für die Darstellung verwenden.

Die ganze Applikation basiert auf einem JFrame, auf welchem die `View` platziert wird.

4.2 View

Die *View* ist aus zwei übergeordneten Panels aufgebaut. Im linken Panel befinden sich Ein- und Ausgabefelder für numerische Werte, das rechte Panel beheimtated die Plots sowie das Optimierungs-Panel (siehe Abbildung 23).

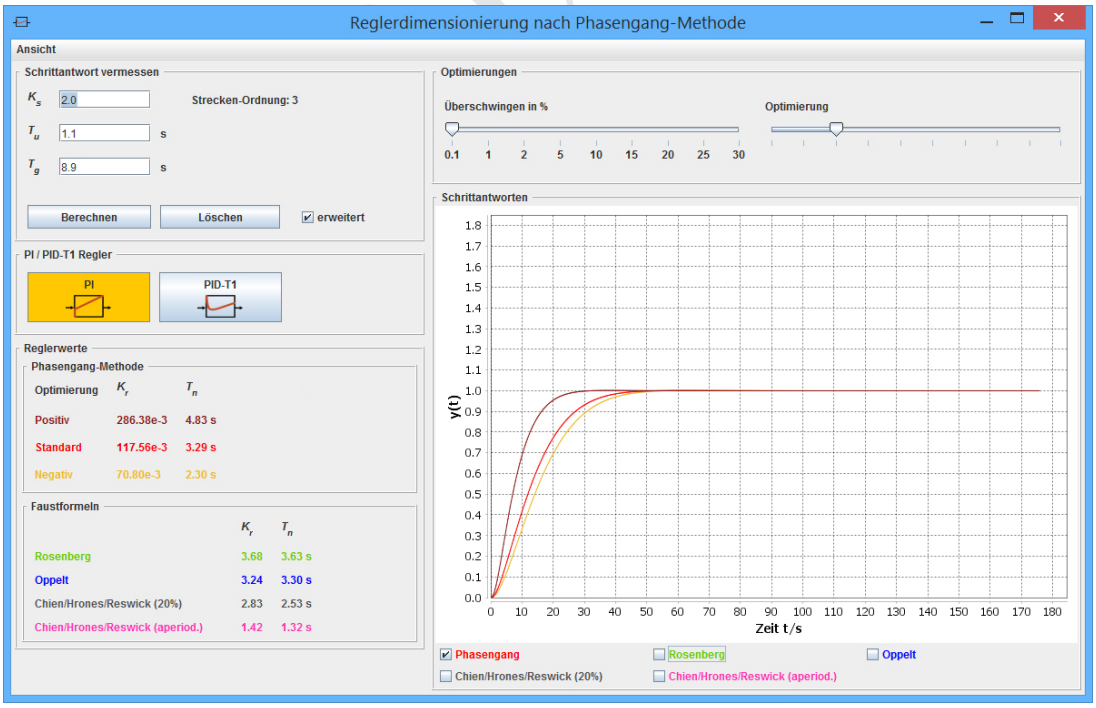

Abbildung 23: Grundlegender Aufbau der Benutzeroberfläche

Das rechte Panel kann mittels der Check-Box *erweitert* (Abbildung 25) ein- und ausgeblendet werden, die reduzierte Benutzeroberfläche ist in Abbildung 24 zu sehen.

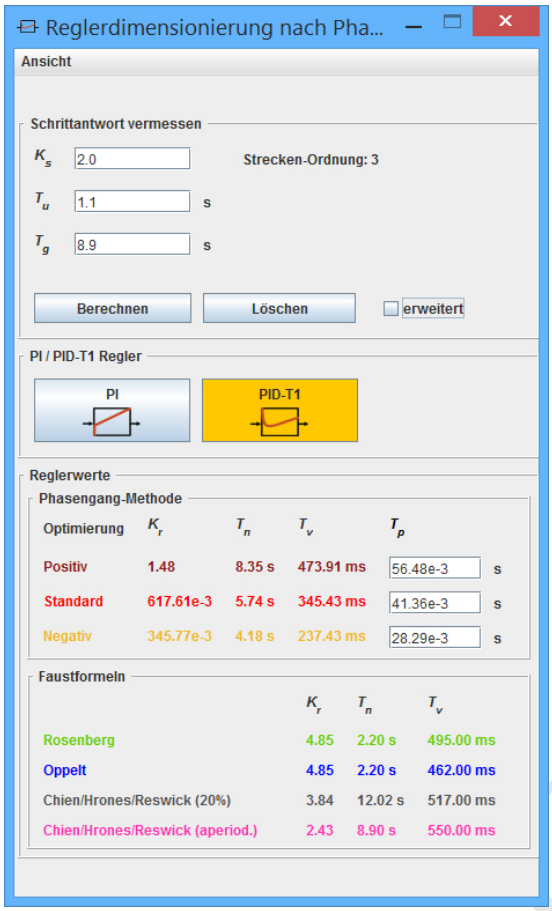

Abbildung 24: Benutzeroberfläche reduziert auf linkes Panel

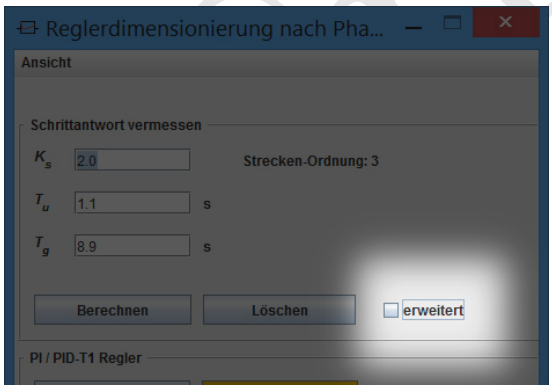

Abbildung 25: Checkbox zum Zu- und Wegschalten des rechten Haupt-Panels

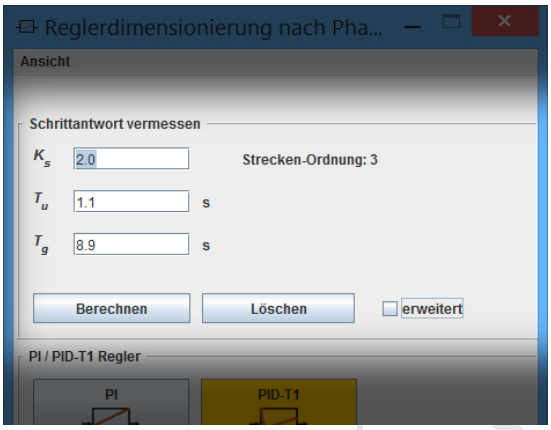

Abbildung 26: Bereich zum Eingeben der Streckenparameter

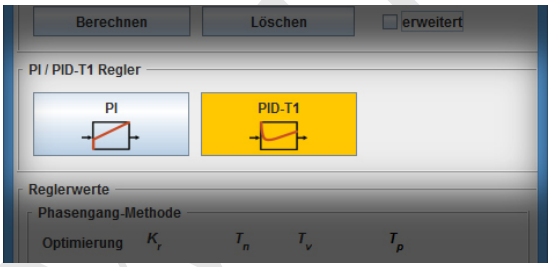

Abbildung 27: Auswahl zwischen PI- und PID-Regler

Abbildung 28: Ausgabewerte des dimensionierten Reglers

Im Bereich *Schrittantwort vermessen* (Abbildung 26) werden die Parameter der vermessenen Strecke eingegeben. Darunter befinden sich die Schalftflächen zur Wahl zwischen der Dimensionierung eines PI- respektive eines PID-T1-Reglers (Abbildung 27).

Das Panel *Reglerwerte* (Abbildung 28) dient der Ausgabe der berechneten Reglerwerte der verschiedenen Berechnungsmethoden. Ebenfalls kann für die Phasengangmethode, sofern ein PID-Regler dimensioniert wird, die Zeitkonstante T_p spezifiziert werden.

Das Optimierungs-Panel (Abbildung 29)beinhaltet zwei Slider zur Eingabe des gewünschten Überschwingens respektive für die Optimierung des Reglers der Phasengangmethode. Über den Slider *Optimierung* kann φ_r beeinflusst werden.

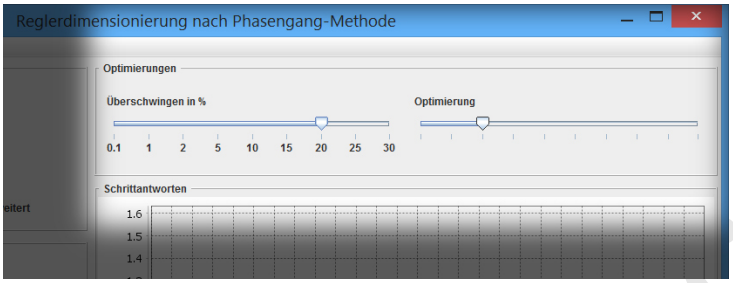

Abbildung 29: Slider im Optimierungs-Panel

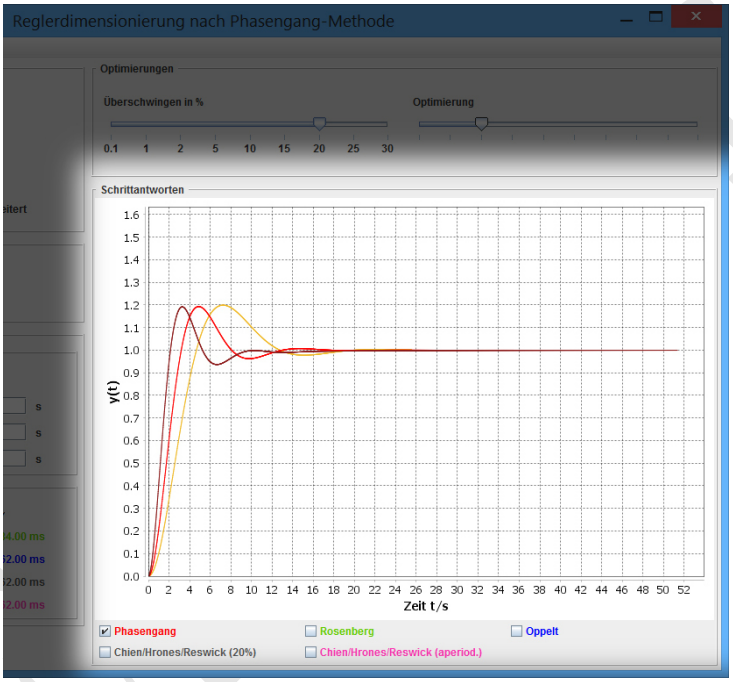

Abbildung 30: Plots der Schrittantworten, PID-Regler berechnet nach Phasengangmethode, 20% Überschwingen

Unterhalb des Optimierungs-Panels werden die Plots der mittels Faustformeln und Phasengangmethode errechneten Schrittantworten graphisch ausgegeben (Abbildung 30). Diese können mittels Check-Boxen zu- und weggeschaltet werden (Bereich entlang des unteren Bildrands in Abbildung 30).

Die Resultate der Phasengangmethode werden durch drei Kurven dargestellt. Eine Kurve benutzt den Standardwert für φ_r gemäss Zellweger ($-90°$ für PI-Regler, $-135°$ für PID-Regler), die beiden anderen Kurven basieren auf Benutzereingaben für einen oberen und unteren Offset von φ_r, der über den Schiebregler *Optimierung* eingestellt werden kann. Das Ergebnis kann unter anderem in Abbildung 22 auf Seite 32 gesehen werden.

Es ist ebenfalls möglich, die dargestellten Plots als Bilder zu speichern; die in Abschnitt 3.7 abgebildeten Beispiele (Abbildungen 20, 21 und 22 auf Seite 32) wurden mit unserem Tool erstellt und exportiert.

4.3 Controller

Regler heisst übersetzt auf Englisch *Controller*. Aus diesem Grund heisst die generische Reglerklasse in unserer Software `Controller`. Die Klasse, welche die Rolle des *Controllers* im Kontext von *Model-View-Controller* wahrnimmt, heisst daher `GUIController`, um Namenskonflikte zu vermeiden.

Der `GUIController` dient als Schnittstelle zwischen der `View` und dem `Model`. Werden Eingaben auf der `View` durch den Benutzer gemacht, wird der `GUIController` darüber informiert. Er prüft die Daten auf Zulässigkeit und leitet berechnungsrelevante Daten, sofern sie die Prüfung bestanden haben, an das `Model` zur Verarbeitung weiter. Zusätzlich übernimmt der `GUIController` die Steuerung der `View`, indem er Objekte ein- und ausblendet sowie Meldungen/Warnungen über die `View` ausgibt.

4.4 Model

Die Model-Klassen beheimaten alle Daten sowie den Grossteil der Algorithmen (zusätzliche Algorithmen, die von den Model-Klassen verwendet werden, sind in der Klasse `Calc` des Package `Utilities` zu finden). Das `Model` ist observable, um das Aktualisieren der `View` zu ermöglichen.

`Model`

Wie im Klassendiagramm (siehe Anhang D) ersichtlich, erzeugt die Klasse `Model` im Konstruktor für jede Berechnungsart eine Instanz der Klasse `ClosedLoop`. Das `Model` hat ein Objekt der Klasse `Path` und übergibt dieses an den `ClosedLoop`. Das `Model` enthält Setter- und Getter-Methoden zur Verarbeitung der Daten mittels weiterer Klassen. Das `Model` nutzt die Möglichkeit `notifyObservers()` aufzurufen, um `update()` auf der `View` zu bewirken.

`ClosedLoop`

Die Instanzen der Klasse `ClosedLoop` repräsentieren die geschlossenen Regelkreise. Sie kennen ihren Berechnungstyp und besitzen einen Regler. Im Konstruktor wird je nach Berechnungsart ein passender `Controller` instanziiert. Zusätzlich beinhaltet die Klasse die Methode `calculate()`. Diese berechnet für alle Berechnungsarten die Schrittantwort mittels `calculateStepResponse()`.

Für die Phasengangmethode wird zusätzlich das Überschwingverhalten mittels der Methode `overShootOptimization()` optimiert (siehe Anhang B.11). Zudem sind diverse Setter- und Getter-Methoden Bestandteil dieser Model-Klasse.

`Controller`

Der `Controller` bildet die Oberklasse aller Faustformeln und der Phasengangmethode. Er beinhaltet die abstrakte Klasse `calculate()` sowie alle nötigen Setter- und Getter-Methoden um Werte zu setzten und auszulesen. Die Methode `utfController()`, deren Algorithmus in Anhang B.4 beschrieben ist, wird via die Methode `setUTFPoly()` hier ausgelöst.

`Chien20`

Die Klasse `Chien20` stellt die Algorithmen (siehe Anhang B.8) zur Berechnung der Reglerwerte gemäss der Faustformel Chien/Hrones/Reswick (20%) zur Verfügung, erbt von `Controller`, instanziiert und setzt eine Übertragungsfunktion.

ChienApper

Die Klasse `ChienApper` stellt die Algorithmen (siehe Anhang B.8) zur Berechnung der Reglerwerte gemäss der Faustformel Chien/Hrones/Reswick (aperiodisch) zur Verfügung, erbt von `Controller`, instanziiert und setzt eine Übertragungsfunktion.

Oppelt

Die Klasse `Oppelt` stellt die Algorithmen (siehe Anhang B.5) zur Berechnung der Reglerwerte gemäss der Faustformel Oppelt zur Verfügung, erbt von `Controller`, instanziiert und setzt eine Übertragungsfunktion.

Rosenberg

Die Klasse `Rosenberg` stellt die Algorithmen (siehe Anhang B.6) zur Berechnung der Reglerwerte gemäss der Faustformel Rosenberg zur Verfügung, erbt von `Controller`, instanziiert und setzt eine Übertragungsfunktion.

PhaseResponseMethod

Die Klasse `PhaseResponseMethod` stellt die Algorithmen zur Berechnung der Reglerwerte gemäss der Phasengangmethode zur Verfügung. Sie erbt von der Klasse `Controller` und bringt all deren Setter- und Getter-Methoden mit.

Über die überladene Methode `setData()` werden die Input-Werte gesetzt. Dabei wird die Methode `calculateOverShoot()`, die das für die Berechnung des korrekten Überschwingens benötigt Attribut `phiU` setzt, ausgelöst. Zusätzlich wird `calculate()` aufgerufen. `calculate()` berechnet anhand der Methode `createOmegaAxis()` die diskrete Frequenzachse in Abhängigkeit der Zeitkonstante der Regelstrecke. Die Übertragungsfunktion in s der Regelstrecke wird für alle Punkte von `omega` berechnet.

`calculateTnk()` wird ausgelöst und berechnet T_{nk} und T_{vk} unter Zuhilfenahme der diskreten Werte. `calculateTnk()` löst wiederum `calculateKrk()` zur Berechnung von K_{rk} aus und ruft zudem `calculateControllerConf()` und `setUTF()` auf. `calculateControllerConf()` transformiert die Werte in die reglerkonforme Darstellung (siehe Anhang B.3). `setUTF()` setzt die Übertragungs-Funktion des Reglers.

Path

Die Klasse `Path` berechnet in der Methode `calculate()`, unter Zuhilfenahme der Sani-Methode (`Calc.sani()`), die Strecke (`Path`). Sie besitzt eine Instanz der Klasse `UTF` sowie Setter- und Getter-Methoden zum Setzen und Auslesen von Werten.

UTF

Setzt und speichert die Übertragungsfunktion. Diese kann über diverse Getter- und Setter-Methoden ausgelesen bzw. geschrieben werden.

4.5 Benutzungs-Beispiel (Use Case)

Das Zusammenspiel der einzelnen Komponenten der Applikation wird im Folgenden anhand eines Beispiels erklärt. Die hier dargelegten Zusammenhänge können auch gut im Klassendiagramm (Anhang D) betrachtet werden.

Beim Programmstart werden durch das Model drei `closedLoops` (geschlossener Regelkreis) für die Phasengang-Methode sowie vier weitere für die Faustformeln erzeugt. Jeder `closedLoops` ist von Beginn an einem Berechnungstyp (Phasengang-Methode, Faustformel) zugewiesen. Er ist bereit, Daten aufzunehmen und zu verarbeiten.

Über die drei Eingabefelder 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 `GUIController` bestanden, fragt dieser zusätzlich die aktuellen Werte/Zustände 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 Optimierungs-Offset und weist die Daten den entsprechenden `closedLoops` der Phasengang-Methode sowie der Faustformeln mittels der Methode `setData()` zu. Jeder `ClosedLoop` leitet die Daten an den zugehörigen Controller weiter, der die Reglerwerte berechnet. Zu Beginn betrachten wir den Regler nach Oppelt genauer.

Die Klasse `Oppelt` erbt von der abstrakten Klasse `Controller` und besitzt somit alle Setter- und Getter-Methoden der Oberklasse. Als Input stehen die Informationen der Strecke sowie der Reglertyp (PI, PID) zur Verfügung. Je nach gewähltem Berechnungstyp werden die Reglerwerte reglerkonform berechnet und gespeichert. Weiter werden die Werte in die bodekonforme Darstellung umgerechnet und ebenfalls abgespeichert (mehr zu diesem Thema in Abschnitt 3.5).

Die Berechnungstypen Rosenberg, Chien/Hrones/Reswick (20%) sowie Chien/Hrones/Reswick (aperiod.) funktionieren analog dem Berechnungstyp Oppelt und werden nicht weiter ausgeführt.

Ein spezielles Augenmerk richten wir nun auf die Berechnung der Phasengang-Methode. Auch die Klasse `PhaseResponseMethod` erbt von der Klasse `Controller` und bringt die bereits erwähnten Setter- und Getter-Methoden mit. Die Input-Werte sind analog derjenigen der Faustformeln. Zusätzlich werden die Informationen zum Überschwingen, der Optimierung sowie T_p in die Berechnung miteinbezogen und die gesetzten Input-Werte werden den lokalen Attributen zugewiesen. `calculateOvershoot()` wird ausgelöst und setzt das Attribut `phiU`, welches für das korrekte Überschwingen benötigt wird. Darauf folgend wird `calculate()` aufgerufen. Diese Methode berechnet anhand der Methode `createOmegaAxis()` die diskrete Omega-Achse in Abhängigkeit der Zeitkonstanten der Regelstrecke. Der Wert der Übertragungsfunktion der Regelstrecke wird für alle Punkte der Frequenzachse ausgewertet. `calculateTnk()` wird ausgelöst. Diese Methode berechnet T_nk und T_vk unter Zuhilfenahme der diskreten Werte nach dem Prinzip der Phasengangmethode. Daraus resultiert die Übertragungsfunktion des Reglers. Falls $T_p = 0$ aus der Eingabe übergeben wurde, wird an dieser Stelle T_p berechnet. K_{rk} wird gemäss der Phasengangmethode mittels `calculateKrk()` berechnet. Zudem beinhaltet die Methode `calculateKrk()` den Aufruf von `calculateControllerConf()` und `setUTF()`. `calculateControllerConf()` transformiert die Werte in die reglerkonforme Darstellung. `setUTF()` setzt die Übertragungs-Funktion des Reglers.

Im `ClosedLoop` wird nun die Methode `calculate()` ausgeführt, welche mittels der Methode `calculateStepResponse()` die Schrittanwort des geschlossenen Regelkreises berechnet. Falls es

sich bei der Berechnungsmethode des Reglers um die Phasengang-Methode handelt, wird zusätzlich `overShootOptimization()` (dokumentiert in Anhang B.11) aufgerufen. Diese Methode ändert den Wert von K_{rk} so lange, bis das gewünschte Überschwingen erreicht wird.

Sobald die Berechnungen aller `ClosedLoops` abgeschlossen sind, wird im `Model` die Methode `notifyObserver()` ausgelöst. Hiermit wird die `View` darüber informiert, dass Änderungen im `Model` vorgenommen wurden und die Methode `update()` in den jeweiligen Unterklassen der Klasse `View`aufgerufen werden soll. Somit aktualisiert sich die `View` und die neu berechneten Reglerwerte. Die Streckenordnung wird ausgegeben und die Plot-Daten werden aktualisiert. Im Fall der Phasengangmethode werden auch Startwerte für T_p gesetzt.

Der Benutzer hat nun die Möglichkeit, die Resultate der Phasengangmethode weiter zu optimieren. Über das Panel `Optimierungen` stehen die Slider *Überschwingen* sowie *Optimierung* zur Verfügung. Das Überschwingen kann in vorgegebenen Schritten in Prozenten festgelegt werden. Die Optimierung schiebt φ_r in die positive sowie negative Richtung zugleich und wird mittels zwei separaten Plots dargestellt. Weiter können die Werte für T_p nachträglich für jede der drei Kurven individuell angepasst werden.

Sobald einer der drei Parameter (Überschwingen, Optimierung, T_p) verändert wird, wird über den `GUIController` die jeweilige Setter-Methode im `Model` aufgerufen. Das `Model` gibt die Daten an den jeweiligen `ClosedLoop` weiter, der diese wiederum den Methoden der Phasengang-Methode weiterleitet. Sobald die neu berechneten Werte vorliegen wird `notifyObservers()` aufgerufen und die `View` aktualisiert.

Die Plots sowie die Optimierungs-Schaltflächen sind nur dann sichtbar, wenn die CheckBox *erweitert* aktiviert ist. Durch Deaktivieren dieser CheckBox kann das Programm in einer Klein-Ansicht ohne grafische Ausgabe bedient werden. Die einzelnen Plots können über CheckBoxen unterhalb des Plot-Bereichs dazu- oder weggeschaltet werden.

Über den Button *Löschen* können alle Regler- sowie Plot-Daten gelöscht werden.

5 Verifikation

Damit die Software von Anfang an den Anforderungen des Kunden sowie dem Pflichtenheft entspricht, wurden die Berechnungen sowie die Funktionsweise der einzelnen Komponenten laufend mittels verschiedenster Varianten überprüft, welche nachfolgend erläutert werden.

5.1 Berechnungen

Die Berechnungen und Algorithmen wurden vorerst in Matlab geschrieben und durch ein weiteres Teammitglied überprüft. Die in Matlab berechneten Test-Resultate konnten mit dem Fachcoach/Auftraggeber abgeglichen werden. Die so entstandenen Matlab-Files dienten als direkte Vorlage für die Umsetzung in Java-Code. Zusätzlich konnten die berechneten Werte in Java direkt mit den Resultaten aus Matlab verglichen werden.

5.2 Intern (Team)

Um die Funktionalität und das Verhalten des GUI's zu testen wurde das Programm regelmässig durch das Team getestet. Dabei wurden alle Mitglieder dazu aufgefordert nebst den zu erwartenden Werten auch explizite Fehleingaben/-ausgaben zu provozieren.

Fehler wurden umgehend an den Programmier-Verantwortlichen weitergeleitet. Sobald der Fehler behoben wurde, wurde das Team darüber informiert und zur erneuten Verifikation aufgefordert.

5.3 Extern

Das Tool wurde auf Computer aussenstehender Personen auf Funktionalität und korrekte Darstellungsweise getestet. Dabei wurde ein spezielles Augenmerk darauf gelegt, dass die Testpersonen unterschiedliche Betriebssysteme und Bildschirmauflösungen verwenden.

5.4 Auftraggeber

Die Software wurde in enger Zusammenarbeit mit dem Auftraggeber entwickelt. So konnte er die Software immer wieder testen und direktes Feedback an die Entwickler geben. Eine möglichst kundenorientierte Lösung, die den Vorstellungen des Auftraggebers entspricht, konnte somit erarbeitet werden. Eine Vorabversion wurde dem Auftraggeber zur Überprüfung der Auflösung und Lauffähigkeit auf seinem System zugestellt.

6 Schlussfolgerungen

Das Projekt wurde erfolgreich abgeschlossen: Das Tool ist funktionstüchtig und alle Punkte des ursprünglichen Auftrages sowie einige der im Verlauf des Projekts optionalen Erweiterungen wurden erfüllt. Aus den Eingabewerten wird das dynamische Verhalten des geschlossenen Regelkreises berechnet und graphisch dargestellt. Die Erweiterungen ermöglichen das Dimensionieren des Reglers anhand der Phasengangmethode und die Optimierung der Reglerwerte.

Die im Pflichtenheft optional geplanten, graphischen Ausgaben des Amplitudengangs und der Strecke sowie die Schieberegler für das Verändern der Reglerwerte wurden weggelassen. Diese Zusätze wurden geplant, um das Dimensionieren des Reglers zu erleichtern und die Eingabewerte graphisch darzustellen. Der Grund für den Entscheid, diese zusätzlichen Features wegzulassen, war die Möglichkeit durch die hohe Rechenleistung den Regler in Echtzeit automatisiert zu optimieren, womit die Funktionen überflüssig wurden.

Alternativ kann die Schrittantwort mit zwei Schiebereglern manuell optimiert werden. Durch das Verändern des Überschwingens und das Einstellen der Optimierung kann die Kurve von Hand angepasst werden.

Eine möglicher Bereich, in dem zusätzliche Features implementiert werden könnten, ist die Auswertung und Analyse der Strecke. Es könnten die Kennzahlen der Übertragungsfunktion der Strecke zurückgegeben werden, zusammen mit einer graphischen Darstellung der Strecke.

Einen bedeutenden Zusatznutzen würde auch das automatische Einlesen der Streckenwerte aus einer graphischen Darstellung bringen. Das Tool würde an die eingelesene Kurve selbständig die Wendetangente legen und die zugeörigen Kennwerte auslesen. Das Ablesen könnte mit der Monte-Carlo-Studie verfeinert werden (ein Verfahren aus der Wahrscheinlichkeitsrechung, welches aus einer endlichen Schar von Kurven die numerisch wahrscheinlichste Lösung findet).

Das Vergleichen der Resultate von verschiedenen PI- und PID-Reglern im gleichen Plot ist momentan noch nicht möglich, wäre sicherlich aber auch nützlich.

Weiter wäre eine automatische Anpassung der Werte für φ_s denkbar (siehe dazu Gleichungen 3 und 15 auf Seite 18 respektive 23 sowie die zugehörigen Fussnoten [IV] und [VII]). Auch zusätzliche Reglertypen könnten noch implementiert 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: _____, _____

A 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:

$$\begin{aligned}
\omega_{pid} &= 0.6714\,\mathrm{s}^{-1} \\
T_{vk} &= \frac{\beta}{\omega_{pid}} = \frac{0.5}{0.6714\,\mathrm{s}^{-1}} = 0.7447\,\mathrm{s} \\
T_{nk} &= \frac{1}{\omega_{pid}\cdot\beta} = \frac{1}{0.6714\,\mathrm{s}^{-1}\cdot 0.5} = 2.9789\,\mathrm{s} \\
K_{rk} &= 1
\end{aligned} \tag{41}$$

Diese Werte eingesetzt in Gleichung 14 ergeben:

$$\begin{aligned}
H_{rpid}(j\omega) &= K_{rk}\cdot\left[\frac{(1+s\cdot T_{nk})(1+s\cdot T_{vk})}{s\cdot T_{nk}}\right] \\
&= 1\cdot\left[\frac{(1+j\omega\cdot 0.7447\,\mathrm{s})(1+j\omega\cdot 2.9789\,\mathrm{s})}{j\omega\cdot 2.9789\,\mathrm{s}}\right] \\
&= \frac{1+j\omega\cdot(2.9789\,\mathrm{s}+0.7447\,\mathrm{s})-\omega^2\cdot 0.7447\,\mathrm{s}\cdot 2.9789\,\mathrm{s}}{j\omega\cdot 2.9789\,\mathrm{s}} \\
&= \frac{1-2.2184\,\mathrm{s}^2\cdot\omega^2+j\omega\cdot 3.7236\,\mathrm{s}}{j\omega\cdot 2.9789\,\mathrm{s}} \\
&= \frac{-\omega\cdot 3.7236\,\mathrm{s}+j(1-\omega^2\cdot 2.2184\,\mathrm{s}^2)}{\omega\cdot 2.9789\,\mathrm{s}} \\
&= -1.250+j\cdot(\omega^{-1}\cdot 0.3357\,\mathrm{s}^{-1}-\omega\cdot 0.7450\,\mathrm{s})
\end{aligned} \tag{42}$$

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:

$$\begin{aligned}
\varphi(Re+j\cdot Im) &= atan\left(\frac{Im}{Re}\right)+\pi \qquad && Re<0\wedge Im\geq 0 \\
\varphi(Re+j\cdot Im) &= atan\left(\frac{Im}{Re}\right)-\pi \qquad && Re<0\wedge Im<0
\end{aligned} \tag{43}$$

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.

$$\begin{aligned}
\varphi(H_{rpid}(j\omega)) &= atan\left(\frac{\omega^{-1}\cdot 0.3357\,\mathrm{s}^{-1}-\omega\cdot 0.7450\,\mathrm{s}}{-1.250}\right)\pm\pi \\
&= atan\left(\omega^{-1}\cdot -0.2686\,\mathrm{s}^{-1}-\omega\cdot 0.5960\,\mathrm{s}\right)\pm\pi
\end{aligned} \tag{44}$$

Die Ableitung des Arkustangens ist:

$$\frac{d}{dx}atan(x) = \frac{1}{1+x^2} \tag{45}$$

Mit

$$x(j\omega) = \omega^{-1} \cdot -0.2686\,\mathrm{s}^{-1} - \omega \cdot 0.5960\,\mathrm{s} \tag{46}$$

folgt

$$\begin{aligned} \frac{d}{d\omega}\varphi(H_{rpid}(j\omega)) &= \frac{d}{dx}atan(x(j\omega)) \cdot \frac{d}{d\omega}x(j\omega) \\ &= \frac{0.5960 + \omega^{-2} \cdot 0.2686\,\mathrm{s}^2}{1 + (\omega \cdot 0.5960\,\mathrm{s} - \omega^{-1} \cdot 0.2686\,\mathrm{s}^{-1})^2} \\ &\approx 1.1920 \end{aligned} \tag{47}$$

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 42 bis 47 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 B.12 zu finden.

B Beschreibung der Algorithmen

B.1 Sani

Input

T_u	Verzugszeit
T_g	Anstiegszeit

Output

Array	Array mit den Zeitkonstanten des Polynoms der Übertragungsfunktion der Strecke

Algorithmus

1. Ungültige Eingaben warden 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. Interpolation für r und ω wird mittels `spline()` ausgeführt.
6. $T(n)$ wird aus $\omega \cdot T_g$ berechnet.
7. Umspeichern und 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;
```

B.2 Umrechnung von reglerkonformer in bodekonforme Darstellung

Input

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

Output

Array	Array mit Nachstellzeit T_{nk}, Vorhaltezeit T_{vk} und Verstärkungsfaktor K_{rk} des Reglers

Algorithmus

1. Wählt je nach Reglertyp die Umrechnungsformel.
2. Falls ein nicht implementierter Reglertyp gewählt wird, wird ein Fehler zurückgegeben.
3. Berechnung der Reglerwerte gemäss Tabelle 5.

Matlab-Code

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

if (reglertyp = 2) %PI-Regler
    t_nk=t_n;
    k_rk=k_r;
    t_vk=0;
elseif (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);
else
     error('!!!! Nicht implementierter Reglertyp !!!!');
end;
```

B.3 Umrechnung von bodekonformer in reglerkonforme Darstellung

Input

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

Output

Array	Array mit Nachstellzeit T_n, Vorhaltezeit T_v und Verstärkungsfaktor K_r des Reglers

Algorithmus

1. Wählt je nach Reglertyp die Umrechnungsformel.
2. Falls ein nicht implementierter Reglertyp gewählt wird, wird ein Fehler zurückgegeben.
3. PI-Regler: $T_n = T_{nk}$, $K_k = K_{rk}$, $T_v = 0$
4. Berechnung der Reglerwerte gemäss Tabelle 5.

Matlab-Code

```
function [ t_n, t_v, k_r ] = p2_reglerkonf(t_nk, t_vk, t_p, k_rk ,reglertyp )

if (reglertyp = 2) %PI-Regler
    t_n=t_nk;
    k_r=k_rk;
    t_v=0;
elseif (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;
else
     error('!!!! Nicht implementierter Reglertyp !!!!');
end;
```

B.4 Übertragungsfunktion des Controllers (`utfController()`)

Input

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

Output

Array	Array mit reellem Zähler und reellem Nenner

Algorithmus

1. Wählt je nach Reglertyp die korrekten Formeln, gibt Fehler zurück bei nicht implementiertem Regler.
2. PI-Regler: Zurückgabe der Koeffizienten des Zähler- und Nennerpolynoms in bodekonformer Darstellung gemäss Gleichung 36.
3. PID-Regler: Zurückgabe der Koeffizienten des Zähler- und Nennerpolynoms in bodekonformer Darstellung gemäss Gleichung 37.

Matlab-Code

```
1  function [ Zah_r, Nen_r ] = p2_UTFController(t_nk, t_vk, t_p, k_rk ,Reglertyp )
2
3
4  if (Reglertyp == 2) % PI Regler
5      Zah_r = k_rk*[t_nk 1];
6      Nen_r = [t_nk 0];
7  elseif (Reglertyp == 3) % PID-Regler
8      Zah_r = k_rk * p2_xdiskConv([t_vk 1],[t_nk 1]);
9      Nen_r = [t_nk 0];
10 else
11      error('!!!! Nicht implementierter Reglertyp !!!!');
12 end;
```

B.5 Faustformel Oppelt

Input

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

Output

Array	Array mit Proportionalitätsfaktor K_p, Nachstellzeit T_n und Vorhaltezeit T_v

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:

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

$$T_n = 2 \cdot T_u$$

$$T_v = 0.42 * T_u$$

B.6 Faustformel Rosenberg

Input

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

Output

Array	Array mit Proportionalitätsfaktor K_p, Nachstellzeit T_n und Vorhaltezeit T_v

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:

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

$$T_n = 2 \cdot T_u$$

$$T_v = 0.45 \cdot T_u;$$

B.7 Faustformel Ziegler

Input

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

Output

Array	Array mit Proportionalitätsfaktor K_p, Nachstellzeit T_n und Vorhaltezeit T_v

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:

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

$$T_n = 2 \cdot T_u$$

$$T_v = 0.5 \cdot T_u$$

B.8 Faustformel Chien

Input

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

Output

Array	Array mit Proportionalitätsfaktor K_p, Nachstellzeit T_n und Vorhaltezeit T_v

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:

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

$$T_n = 2.4 \cdot T_u$$

$$T_v = 0.42cdotT_u;$$

5. Für PID, 20% Überschwingen:

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

$$T_n = 2 \cdot T_u$$

$$T_v = 0.42 \cdot T_u;$$

B.9 Steigung einer Funktion

`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 }
```

B.10 Inverse Fast Fourier Transform

`schrittIfft()` berechnet die Schrittantwort im Zeitbereich einer Übertragungsfunktion.

Input

Zähler	Zähler der Übertragungsfunktion
Nenner	Nenner der Übertragungsfunktion
f_s	Abstastfrequenz
n	Granulierung der Frequenzachse

Output

Resultat	Zweidimensionales Array mit Zeitachse und zugehörigen Funktionswerten

Algorithmus

1. Array mit 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

```
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;
}
```

B.11 Optimieren des Überschwingens

`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. Die Optimierung des Überschwingens erfolgt in zwei Stufen: Eine rasche, aber grobe, erste Stufe, und eine langsamere, aber feinere, zweite Stufe. Dies erlaubt, rasch in den gewünschten Wertebereich zu kommen, ohne dass dabei das Endergebnis an Genauigkeit verliert. In beiden Stufen ist das Konzept identisch:
   (a) Falls zu starkes Überschwingen: Reglerverstärkung K_{rk} schrittweise reduzieren, bis gewünschtes Verhalten eingehalten wird.
   (b) Falls zu schwaches Überschwingen: Reglerverstärkung K_{rk} schrittweise erhöhen, bis gewünschtes Verhalten eingehalten wird.

Java-Code

```
1  private void overShootOptimization() {
2      PhaseResponseMethod phaseResponseMethod = (PhaseResponseMethod) controller;
3      double max = yt[0][Calc.max(yt[0])];
4
5      // Grobskalierung
6      int order = path.getT().length;
7      double maxSoll = controller.overShoot / 100 + 1.0;
8      double KrkNew;
9      double Krk = phaseResponseMethod.getControllerValues()[PhaseResponseMethod.KrkPOS];
10     int count = 0;
11     // Falls max kleiner Soll
12     if (maxSoll - max > 0.08) {
13         while (maxSoll - max > 0.08 & count < 10) {
14             count++;
15             KrkNew = Krk * maxSoll / max * (8.0 / order);
16             phaseResponseMethod.setKrk(KrkNew);
17             calculateStepResponse();
18             double maxNew = yt[0][Calc.max(yt[0])];
19
20             // Kontrolle ob erfolgreich vergroessert sonst Abbruch
21             if (maxNew > max) {
22                 max = maxNew;
23                 Krk = phaseResponseMethod.getControllerValues()[PhaseResponseMethod.KrkPOS];
24             } else {
25                 count = 100;
26             }
27         }
28         // Falls max groesser Soll
29     } else if (maxSoll - max < -0.08) {
30         while (maxSoll - max < -0.08 & count < 5) {
31             count++;
32             KrkNew = Krk * maxSoll / max * (order / 8.0);
33             phaseResponseMethod.setKrk(KrkNew);
34             calculateStepResponse();
35             double maxNew = yt[0][Calc.max(yt[0])];
36
37             // Kontrolle ob erfolgreich verkleinert sonst Abbruch
38             if (maxNew < max) {
39                 max = maxNew;
40                 Krk = phaseResponseMethod.getControllerValues()[PhaseResponseMethod.KrkPOS];
41             } else {
42                 count = 100;
43             }
44         }
45     }
46
47     int countMax = 100;
48     if (pointnumber < 8193)
49         countMax = 200;
50     // Feinskalierung mit dem Faktor 1.05
51     count = 0;
52     if (max > maxSoll) {
53         while (max > maxSoll & Krk > 1e-19 & count < countMax) {
54             count++;
55             phaseResponseMethod.setKrk(Krk / 1.05);
56             calculateStepResponse();
57             max = yt[0][Calc.max(yt[0])];
58             Krk = phaseResponseMethod.getControllerValues()[PhaseResponseMethod.KrkPOS];
59         }
60     } else {
61         while (max < maxSoll & Krk < 1e16 & count < countMax) {
62             count++;
63             phaseResponseMethod.setKrk(Krk * 1.05);
64             calculateStepResponse();
65             max = yt[0][Calc.max(yt[0])];
66             Krk = phaseResponseMethod.getControllerValues()[PhaseResponseMethod.KrkPOS];
67         }
68     }
69 }
```

B.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 A für die manuelle Rechnung). Dazu wird auch die in Anhang B.9 beschriebene Methode `diskDiff()` benutzt.

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, dPhiR, dPhiO;
6      for (int iteration = 0; iteration < 20; iteration++) {
7          beta = (beta_o - beta_u) / 2 + beta_u;
8          Tvk = 1 / (omegaController / beta);
9          Tnk = 1 / (omegaController * beta);
10         for (int i = 0; i < pointNumber; i++) {
11             Hr[i] = new Complex(1, omega[i] * Tnk).multiply(
12                     new Complex(1, omega[i] * Tvk)).divide(
13                     new Complex(0, omega[i] * Tnk));
14             Ho[i] = Hs[i].multiply(Hr[i]);
15         }
16         phiR = Calc.ComplexAngleUnwraped(Hr);
17         dPhiR = Calc.diskDiff(omega, phiR, omegaControllerIndex);
18         dPhiO = dPhiS + dPhiR;
19         if (dPhiO * omegaController < -0.5) { beta_u = beta; }
20         else { beta_o = beta; }}
21 }
```

C Matlab-File für schnelle inverse Fourier-Transformation

```
function [y, t] = schrittIfft(B,A,fs,N)
% function [y, t] = schritt(B,A,fs,N)
%
%Bsp.:
%fs = 50;
%N = 4*1024;
%B = [4.0812 1.1400];
%A = [7.4902 33.6613 46.4843 23.2772 7.6612 1.1400];
%[y1, t] = schrittIfft(B,A,fs,N);
%[y2, t] = step(tf(B,A),t);
%figure; plot(t, y1, 'g', t, y2, 'b', t, y1-y2, 'r'); grid on; legend('Via ifft()', 'Via
    step()', 'Abweichung');

T = 1/fs;
w = linspace(0, fs*pi, N/2);
H = freqs(B, A, w);
H = [H(1:N/2) 0 conj(H(N/2:-1:2))];
h = ifft(H);
y = conv(h, ones(1,N+1));
y = y(1:length(y)/2)';
t = linspace(0, (length(y)-1)*T, length(y))';
```

D Klassendiagramm

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.