Fachhochschule Nordwestschweiz
Hochschule für Technik

Fachbericht Easy-PID

Windisch, 1. Juni 2015

Easy-PID

Smart Controller Design

Auftraggeber:	Peter Niklaus
Betreuer:	Peter Niklaus (Elektrotechnik) Richard Gut (Objektorientierte Programmierung) Pascal Buchschacher (Projektmanagement) Anita Gertiser (Kommunikation)
Gruppe:	FS15 pro2E Team 3
Teammitglieder:	Yanick Frei (Projektleiter) Simon Wyss (Experte Elektrotechnik) Josua Stierli (Experte Programmierung) Alexander Murray (Stv. Projektleiter) Simon Sturm
Studiengang:	Elektro- und Informationstechnik

Abstract

Um in der Praxis einen geeigneten Regler zu dimensionieren ist oftmals viel Zeit und Erfahrung notwendig, was es schwierig macht, einen idealen Regler zu finden. Zwar gibt es zur Hilfe Faustformeln, diese liefern jedoch oftmals nur ungenügende Ergebnisse, sodass man schlussendlich auf grafischen Methoden mit Papier und Bleistift zurückgreifen muss. „Easy-PID“ ist ein Softwaretool, welches im Rahmen des Projekt 2 der Fachhochschule Nordwestschweiz entwickelt wurde, das die Reglerdimensionierung mit der grafischen Phasengangmethode von Zellweger sowie mehreren Faustformeln automatisch durchführt und so hilft, einen idealen Regler zu finden. Mit Hilfe der Kenngrössen der Regelstrecke sowie weiteren Einstellungen wie beispielsweise dem maximalen Überschwingen und der Wahl des Reglertyps kann „Easy-PID“ alle benötigten Reglerparameter bestimmen.

Zur richtigen Reglerdimensionierung mussten zuerst die Formeln hergeleitet werden. Die damit erstellten Berechnungsalgorithmen wurden in Matlab implementiert und mit den Werten von Herr Niklaus verglichen, um die Korrektheit zu überprüfen. Um „Easy-PID“ flexibel in Java programmieren zu können, wurde auf das Model-View-Controller Entwurfsmuster zurückgegriffen. Die von „Easy-PID“ gelieferten Ergebnisse wurden ausserdem mit Matlab verifiziert.
Das Java-Programmierung wurde anschliessend weiter optimiert, um ein ideales Verhältnis zwischen Geschwindigkeit und Genauigkeit zu finden. So wurden beispielsweise zur Berechnung der Schrittantworten zwei Vorgehensweisen implementiert: Residuen und IFFT. Dabei hat sich gezeigt, dass mit Residuen sowohl genauere als auch schnellere Resultate erzielt werden können.

„Easy-PID“ kann durchaus als erfolgreiches Projekt bezeichnet werden. Sämtliche vom Auftraggeber gewünschten Funktionen sowie einige optionale Funktionen wie beispielsweise der Export als PDF, eine Miniversion oder die nachträgliche Feinanpassung der Phasengangmethode sind fehlerfrei implementiert. Ausserdem kann der User mit einem Hilfe-Menü hilfreiche Links aufrufen, die auftretende Fragen zum Thema Regelungstechnik beantworten können.
Sämtliche Daten werden dabei in einem einfach zu bedienenden User-Interface dargestellt. So kann der Benutzer die Eingabeparameter in Textfeldern eingeben und mittels einem Dropdown-Menü den Reglertyp wählen. Die mit diesen Daten berechneten Regler werden tabellarisch dargestellt. Die Sprungantwort der geschlossenen Regelstrecke wird ausserdem grafisch dargestellt, wobei auch einzelne Graphen ausgewählt werden können.

Projekt P2 - Aufgabenstellung vom Auftraggeber (FS_2015)

Reglerdimensionierung mit Hilfe der Schrittantwort

1. Einleitung

In der Praxis werden die klassischen Regler (PI, PID, PD, …) oft mit sog. Faustformeln dimensioniert. Dazu benötigt man bestimmte Informationen der zu regelnden Strecke. Handelt es sich dabei um „langsame Strecken“ mit Zeitkonstanten im Bereich von Sekunden bis Minuten, so ist das Bestimmen und Ausmessen der Schrittantwort oft die einzige Möglichkeit zur Identifikation der Strecke. Typische Beispiele dafür sind Temperaturheizstrecken, welche meistens mit einem PTn-Verhalten modelliert werden können (Kaffeemaschine, Boiler, Raumheizungen, Lötkolben, Warmluftfön, usw.).

Die Schrittanwort wird mit Hilfe einer Wendetangente vermessen und die Kenngrössen Streckenbeiwert (K_s), Verzugszeit (T_u) und Anstiegszeit (T_g) werden bestimmt. Dies kann sowohl von Hand (grafisch) oder auch automatisiert durchgeführt werden, falls die Messdaten elektronisch vorliegen. Mit diesen drei Kenngrössen können mit Hilfe sog. Faustformeln PI- und PID-Regler dimensioniert werden (Ziegler/Nichols, Chien/Hrones/Reswick, Oppelt, Rosenberg). Die Faustformeln liefern zwar sehr schnell die Reglerdaten, aber die Schrittantworten der entspr. Regelungen sind teilweise weit vom "Optimum" entfernt und der Regelkreis kann sogar instabil werden. In der Praxis muss man diese "Startwerte" häufig nachoptimieren, damit die Schrittantwort der Regelung die Anforderungen erfüllt.

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

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

2. Aufgaben/Anforderungen an Tool

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

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

3. Bemerkungen

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

Literatur

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

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

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

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

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

16.02.2015
Peter Niklaus

Inhaltsverzeichnis

1 Einleitung

Das Berechnen von Reglern ist oftmals eine schwierige Angelegenheit, die sowohl Erfahrung als auch Zeit benötigt. Zur richtigen Dimensionierung helfen entweder Faustformeln oder auch grafische Methoden, welche jedoch von Hand ausgeführt werden müssen. Folglich ist die Berechnung entweder ungenau oder sehr aufwändig. Da es fragwürdig ist, ob solch manuelle Methoden in einer Zeit von Matlab und weiteren Berechnungstools noch zeitgemäss sind, wurde im Projekt 2 eine Software erstellt, welche Regler automatisch anhand der „Phasengangmethode zur Reglerdimensionierung“ von Jakob Zellweger dimensionieren kann.

Das Hauptziel des Projektes war es, ein funktionierendes Programm zu erstellen, welches die Reglerberechnung automatisch nach der Phasengangmethode von Zellweger sowie weiteren Faustformeln dimensioniert. Zur korrekten Funktion werden sowohl richtig implementierte Berechnungen benötigt als auch die richtige Darstellung der Ergebnisse, beispielsweise mittels sinnvoll skalierten Graphen. Ausserdem wurden einige optionale Ziele definiert. Die wichtigsten davon sind die Simulation der geschlossenen Regelstrecke, eine Miniversion des Programms sowie eine Möglichkeit zum Nachjustieren der Eingabeparameter.

Easy-PID ist ein Programm, mit welchem Regler anhand der Parameter der Schrittantwort automatisch dimensioniert werden können. Easy-PID besitzt ausserdem ein intuitiv zu bediendendes User-Interface, welches neben den Ein- und Ausgabeparemetern auch das dynamische Verhalten des Regelkreises, welche mit verschiedenen Methoden berechnet wurde, grafisch darstellt. Durch die verschiedenen Dimensionerungsmethoden und der Möglichkeit, Parameter in Echtzeit zu manipulieren, kann ein optimaler Regler gefunden werden.

Dieser Bericht beschreibt die Problemlösung aufgeteilt auf die drei Teilbereiche Elektrotechnik, Porgrammierung und Validierung. Der Elektrotechnische Teil behandelt die hergeleiteten Formeln für die Reglerdimensionierung und die Programmierung dieser Formeln in Matlab, während der zweite Teil den Aufbau der Benutzeroberfläche und die Umsetzung der Matlab-Programmierung in Java behandelt. Die Validierung beschreibt den Aufbau und den Vorgang der Tests sowie deren Ergebnisse.

2 Elektrotechnische Problemlösung

In diesem Kapitel wird das Vorgehen zur Elektrotechnischen Problemlösung genauer erläutert. Dabei wird näher auf die Grundlagen der Regelungstechnik, die Zellweger-Methode, einige Faustformeln und das Berechnen des geschlossenen Regelkreises eingegangen.

2.1 Regelungstechnische Grundlagen

Ein Regelkreis besteht aus einer Strecke und einem Regler.

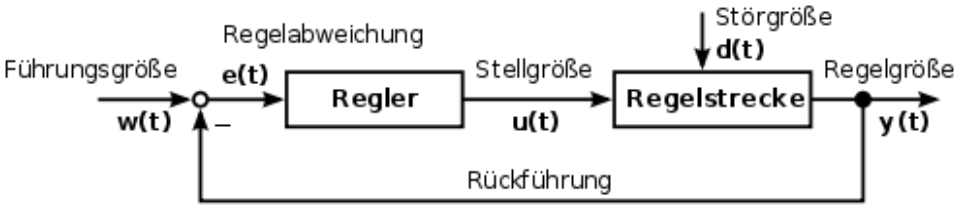

Abbildung 1: Regler und Regelstrecke [?]

Die Aufgabe des Reglekreises ist, es die Regelgrösse konstant auf dem Wert der sogenannten Führungsgrösse zu halten. Bei einer Heizung wären dies beispielsweise die momentane Temperatur, die auf die gewünschte Temperatur zu regeln ist. Die Strecke wäre dabei der zu heizende Raum und der Regler die Heizung selbst.

Um den Regelkreis beschreiben zu können braucht man zunächst die Schrittantwort der Strecke. Aus dieser kann man nun mithilfe der Wendetangentenmethode die Parameter Ks (Verstärkungsfaktor), Tu (Verzugszeit) und Tg (Ausgleichszeit) auslesen.

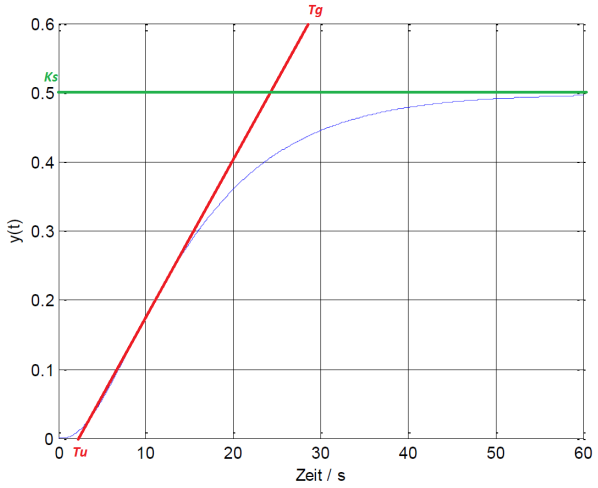

Abbildung 2: Regler und Regelstrecke [?]

Mithilfe dieser Werte und der Sani Methode, welche vom Auftraggeber in Form eines Matlab-Files zur Verfügung gestellt wurde, ist es einem nun möglich, die Ordnung der Strecke sowie die Zeitkonstanten $T1$ bis Tz zu berechnen, wobei Z die Ordnung der Strecke ist. Diese Zeitkonstanten werden schlussendlich dazu verwendet die Übertragungsfunktion $H(s)$ zu berechnen:

$$s = j \cdot w$$

$$H(s) = \prod_{i=1}^{Z} \frac{1}{(1 + s \cdot T_i)} \tag{1}$$

Für die Übertragungsfunktion eines PI/PID Reglers gilt es zu beachten, dass es zwei Darstellungsarten gibt, die Reglerkonforme, sowie die Bodekonforme. Diese sehen wie folgt aus:

PI Reglerkonform:

$$G_R(s) = K_R \left(1 + \frac{1}{s \cdot Tn}\right)$$

PI Bodekonform:

$$G_R(s) = K_R \left(\frac{1 + s \cdot Tn}{s \cdot Tn}\right)$$

PID Reglerkonform:

$$G_R(s) = K_R \left(1 + \frac{1}{s \cdot Tn} + \frac{s \cdot Tv}{1 + s \cdot Tp}\right)$$

PID Bodekonform:

$$G_R(s) = K_{RK} \left(\frac{(1 + s \cdot Tnk)(1 + s \cdot Tvk)}{s \cdot Tnk(1 + s \cdot Tp)}\right)$$

Auf die Verwendung dieser Funktionen sowie deren Herleitung wird im Kapitel 2.3, Zellweger Methode sowie auch im Anhang A.2 genauer eingegangen.

2.2 Faustformeln

Zum Dimensionieren von verschiedenen Reglertypen gibt es eine Vielzahl von fixen Einstellregeln. Diese sind einfach anzuwenden, da sie keine besonderen Vorkenntnisse benötigen, jedoch ist deren Genauigkeit oftmals ungenügend. Die für dieses Projekt relevanten Faustformeln sind:

- Chien/Hrones und Reswick
- Oppelt
- Rosenberg

Diese Faustformeln wurden gewählt, da sie ebenfalls die Werte Ks, Tu und Tg verwenden, welche mittels Wendetangenten aus der Schrittantwort herausgelesen werden können, die uns vom Auftraggeber zur Verfügung gestellt wurde.

Zur Verifizierung der Faustformeln wurden verschiedene Quellen verglichen und die Mehrheit wurde als richtig angesehen.

2.2.1 Chien/Hrones und Reswick

PI

- Aperiodischer Verlauf
  - Gutes Störverhalten

$$Kr = 0.6 \cdot \frac{Tg}{Ks \cdot Tu}$$

$$Tn = 4 \cdot Tu$$

  - Gutes Führungsverhalten

$$Kr = 0.45 \cdot \frac{Tg}{Ks \cdot Tu}$$

$$Tn = 1.2 \cdot Tg$$

- 20% Überschwingen
  - Gutes Störverhalten

$$Kr = 0.7 \cdot \frac{Tg}{Ks \cdot Tu}$$

$$Tn = 2.3 \cdot Tu$$

  - Gutes Führungsverhalten

$$Kr = 0.6 \cdot \frac{Tg}{Ks \cdot Tu}$$

$$Tn = Tg$$

PID

- Aperiodischer Verlauf

  - Gutes Störverhalten

$$Kr = 0.95 \cdot \frac{Tg}{Ks \cdot Tu}$$

$$Tn = 2.4 \cdot Tu$$

$$Tv = 0.42 \cdot Tu$$

  - Gutes Führungsverhalten

$$Kr = 0.6 \cdot \frac{Tg}{Ks \cdot Tu}$$

$$Tn = Tg$$

$$Tv = 0.5 \cdot Tu$$

- 20% Überschwingen
  - Gutes Störverhalten

$$Kr = 1.2 \cdot \frac{Tg}{Ks \cdot Tu}$$

$$Tn = 2 \cdot Tu$$

$$Tv = 0.42 \cdot Tu$$

  - Gutes Führungsverhalten

$$Kr = 0.95 \cdot \frac{Tg}{Ks \cdot Tu}$$

$$Tn = 1.35 \cdot Tg$$

$$Tv = 0.47 \cdot Tu$$

2.2.2 Oppelt

PI

$$Kr = 0.8 \cdot \frac{Tg}{Ks \cdot Tu}$$

$$Tn = 3 \cdot Tu$$

PID

$$Kr = 1.2 \cdot \frac{Tg}{Ks \cdot Tu}$$

$$Tn = 2 \cdot Tu$$

$$Tv = 0.42 \cdot Tu$$

2.2.3 Rosenberg

PI

$$Kr = 0.91 \cdot \frac{Tg}{Ks \cdot Tu}$$

$$Tn = 3.3 \cdot Tu$$

PID

$$Kr = 1.2 \cdot \frac{Tg}{Ks \cdot Tu}$$

$$Tn = 2 \cdot Tu$$

$$Tv = 0.44 \cdot Tu$$

2.3 Zellweger Methode

Der PI- oder PID-Regler wird aus dem Amplituden- und Phasengang mit der Phasengangmethode nach Zellweger berechnet. Der Amplituden- und Phasengang kann aus der Übertragungsfunktion $H(s)(1)$ berechnet werden. Folgend werden die dazu verwendeten Formeln aufgeführt:

$$Amplitudengang\ Strecke\ A_s(\omega) = Ks \cdot \prod_{m=1}^{Z} \frac{1}{\sqrt{1 + (\omega \cdot T_m)^2}}$$

$$Phasengang\ Strecke\ \phi_s(\omega) = -1 \cdot \sum_{m=1}^{Z} \arctan(\omega \cdot T_m)$$

Gemäss Zellweger wird nun nach folgendem Rezept vorgegangen:

1. Aus dem Phasengang ϕ_s wird die Frequenz bei einem bestimmten Winkel ϕ_x herausgesucht. Der Winkel ϕ_x ist abhängig davon, welcher Regler-Typ dimensioniert werden soll. Beim PI-Regler ist ϕ_{pi} -90°, beim PID-Regler ist ϕ_{pid} -135°.

Gemäss dem Skript „Regelkreise und Regelung“ [**?**] von Zellweger gibt es für die Regler zwei Darstellungsvarianten: Die reglerkonforme und die bodekonforme Darstellung.

Bei den weiteren Berechnungen des PID-Reglers muss für die Phasengang-Methode nach Zellweger in die bodekonforme Darstellung gewechselt werden (siehe Anhang Seite 38 Kapitel A.2).

Für PI- und PID-Regler unterscheidet sich nun das weitere Vorgehen. Es ist nun in die folgenden zwei Kapitel aufgeteilt.

Weiteres Vorgehen PI Regler:

Die gefundene Frequenz ω_{pi} bei phi_{pi} entspricht $1/Tn$.

$$Tn = \frac{1}{\omega_{pi}}$$

2. Mit dem gefundenen Tn und $Kr = 1$ wird der Amplituden- und Frequenzgang des offenen Regelkreises (Go) berechnet. Das heisst Regler (Gr) und Strecke (Gs) in Serie, aber ohne Rückkoppelung.

$$Übertragungsfunktion\ PI - Regler\ (reglerkonform)\ Gr = Kr \cdot \left[1 + \frac{1}{s \cdot Tn}\right]$$

$$Übertragungsfunktion\ PI - Regler\ (bodekonform)\ Gr = Kr \cdot \frac{1 + s \cdot Tn}{s \cdot Tn}$$

$$Phasengang\ PI - Regler\ \phi_r = \arctan(\omega \cdot Tn) - \frac{\pi}{2}$$

$$Amplitudengang\ PI - Strecke\ A_r = Kr \frac{\sqrt{1 + (\omega \cdot Tn)^2}}{\omega \cdot Tn}$$

$$Phasengang\ Regelkreis\ offen\ \ \phi_o = \phi_s + \phi_r$$

$$Amplitudentang\, Regelkreis\, offen \;\; A_o = A_s \cdot A_r$$

3. Jetzt wird abhängig vom gewünschten Überschwingen des Reglers ein Winkel auf dem Phasengang des offenen Regelkreises gesucht. Folgende Tabelle zeigt den Zusammenhang des für die Dimensionierung gewählten Überschwingens und des dazu gehörigen Winkels.

Dämpfmass d	ϕ_{rand}	ϕ_{st}	Überschwingen
0.42	45°	-135°	23.3%
0.5	51.5°	-128.5°	16.3%
0.7	65.5°	-114.6°	4.6%
1	76.3°	-103.7°	0%

Tabelle 1: Phasenwinkel abhängig vom gewünschten Überschwingen

Bei 0% Überschwingen muss beispielsweise der Winkel ϕ_{st} gleich -103.7° auf dem Phasengang gesucht werden. Bei der gefundenen Frequenz wird der dazu gehörige Wert aus dem Amplitudengang des offenen Regelkreises herausgelesen. Um diesen Faktor muss die Verstärkung des Reglers vermindert werden. Der aus dem Amplitudengang herausgelesene Wert mit (-1) multipliziert ergibt also den Regler-Parameter Kr.

Weiteres Vorgehen PID Regler:

Beim PID-Regler wird analog zum beim PI-Regler beschriebenen Vorgehen gerechnet. Die gefundene Frequenz ω_{pid} für ϕ_{pid} = -135° entspricht aber nicht direkt $1/Tn$ wie beim PI-Regler.

Die Zellweger'sche Phasengang-Methode berechnet beim PID-Regler Tnk, Tvk und Krk. Dies sind Parameter in der bodekonformen Darstellung. Diese Parameter können schliesslich wieder in die Parameter Tn, Tv und Kr der reglerkonformen Darstellung umgerechnet werden.

2. Für die Berechnung von Tnk und Tvk wird ein Parameter β benötigt. Folgend wird die Berechnung davon aufgeführt:

$$\frac{2 \cdot \beta}{1+\beta^2} - w_{PID} \cdot \sum_{m=1}^{n} \frac{T_m}{1+(w_{PID} \cdot T_m)^2} = -0.5$$

Folgendes wird mit Z substitutiert:

$$Z = \frac{2 \cdot \beta}{1+\beta^2}$$

$$Z = w_{PID} \cdot \sum_{m=1}^{n} \frac{T_m}{1+(w_{PID} \cdot T_m)^2} - 0.5$$

β kann nun durch folgende Gleichung beschrieben werden:

$$\beta = \frac{1}{Z} - \sqrt{\frac{1}{Z^2} - 1}$$

β muss zwischen 0 und 1 liegen. Damit es keine komplexen Lösungen gibt, wird β für Z >1 gleich 1. Für eine detaillierte Herleitung von β siehe Anhang A.1.

3. Tnk und Tvk können nun wie folgt mit β berechnet werden:

$$\frac{1}{Tnk} = \omega_{pid} \cdot \beta$$

$$\frac{1}{Tvk} = \frac{\omega_{pid}}{\beta}$$

4. Für die Berechnung des Dämpfungsfaktors Krk muss beim PID-Regler folglich die Formel der Übertragungsfunktion (1) des PID-Reglers verwendet werden. Der Regler-Parameter Tp wird benötigt, weil der Regler nicht über eine unendlich kurze Zeit differenzieren kann. $1/Tp$ muss ca. 1 Dekade höher als $1/Tvk$ gewählt werden. Beim PID-Regler wird die Bodekonforme Darstellung verwendet, weil Tnk und Tvk vorhanden sind, Tn und Tk jedoch noch nicht. Eine genauere Erläueterung dieser ist im Anhang Seite 38 Kapitel A.2 zu finden. Für die Berechnung des Amplitudenganges wird $Krk = 1$ verwendet:

$$\frac{1}{Tp} = 10 \cdot \frac{1}{Tnk}$$

$$\text{Übertragungsfunktion } PID - Regler \text{ (Reglerkonform) } Gr = Kr \cdot \left[1 + \frac{1}{s \cdot Tn} + \frac{s \cdot Tv}{1 + s \cdot Tp}\right]$$

$$\text{Übertragungsfunktion } PID - Regler \text{ (bodekonform) } Gr = Krk \cdot \left[\frac{(1 + s \cdot Tnk) \cdot (1 + s \cdot Tvk)}{s \cdot Tnk \cdot (1 + s \cdot Tp)}\right]$$

$$Phasengang\, PID - Regler\, \phi_r = \arctan(\omega \cdot Tnk) + arctan(\omega \cdot Tvk) - \arctan(\omega \cdot Tp) - \frac{\pi}{2}$$

$$Amplitudengang\, PID - Regler\, A_r = Krk \frac{\sqrt{1 + (\omega \cdot Tnk)^2} \cdot \sqrt{1 + (\omega \cdot Tvk)^2}}{\omega \cdot Tnk}$$

$$Phasengang\, Regelkreis\, offen \quad \phi_o = \phi_s + \phi_r$$

$$Amplitudentang\, Regelkreis\, offen \quad A_o = A_s \cdot A_r$$

5. Abhängig vom gewählten Überschwingen wird wiederum ein Winkel auf dem Phasengang ϕ_o gesucht (Siehe Tabelle 1 Seite 10). Die Amplitude bei diesem Winkel mit (-1) multipliziert ergibt Krk.

2.3.1 Numerisches Beispiel PI-Regler

Im folgenden Abschnitt wird ein Berechnungsbeispiel zum PI-Regler gemacht. Folgende Werte werden verwendet:

$Ks = 0.5$
$Tu = 2.5$
$Tg = 18.3$
ϕ_{st} = -114.6°(entspricht 4,6% Überschwingen)

Zunächst werden nun die Zeitkonstanten sowie die Ordnung der Strecke mithilfe der Sani Methode bestimmt. Bei unserer Strecke ergibt dies eine Strecke 3. Ordnung und die folgenden Zeitkonstanten:

$T1 = 1.0688s$

$T2 = 3.3484s$

$T3 = 10.4901s$

Nun wird der Frequenzgang aus der Übertragungsfunktion (1) berechnet und in einem Plot aufgezeichnet (siehe Abbildung 3). Danach wird der -90° Punkt auf dem Phasengang gesucht(mit blau gestrichelter Linie in Abbildung 3 dargestellt). Dies ergibt uns eine Frequenz ω_{pi} von $0.1415s^{-1}$. Daraus berechnet sich die Zeitkonstante $Tn = 7.0647s$.

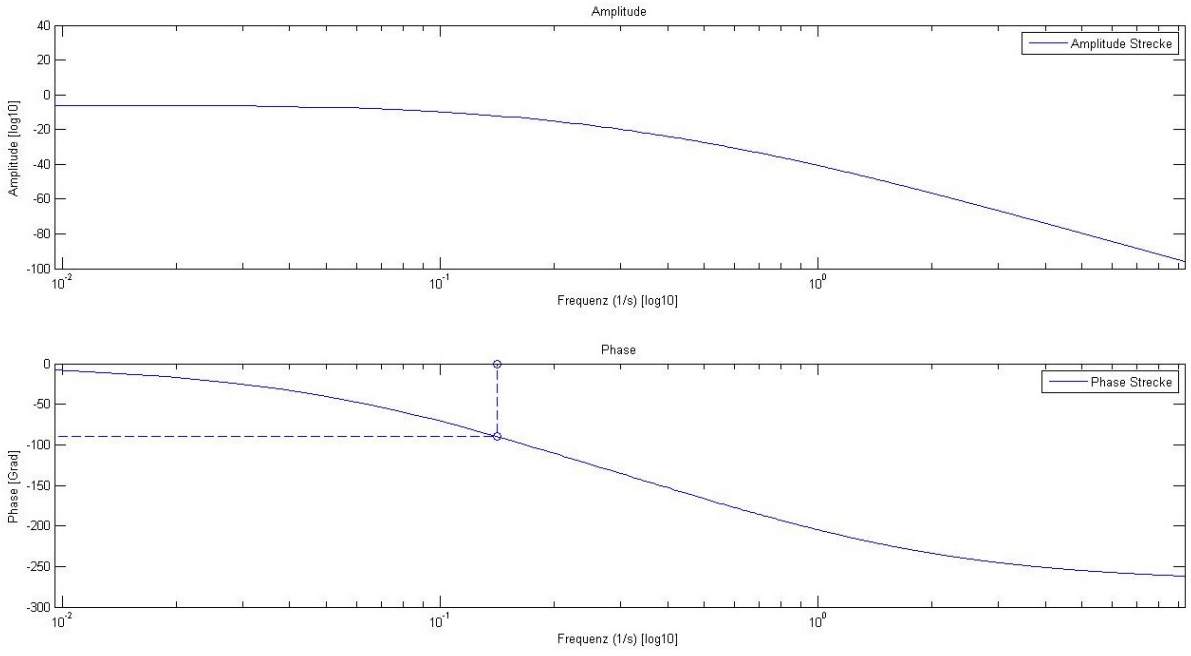

Abbildung 3: Amplituden- und Phasengang der Regelstrecke

Im nächsten Schritt berechnet man mithilfe von Tn und mit $Kr = 1$ den offenen Regelkreis (rot dargestellt in Abbildung 4)

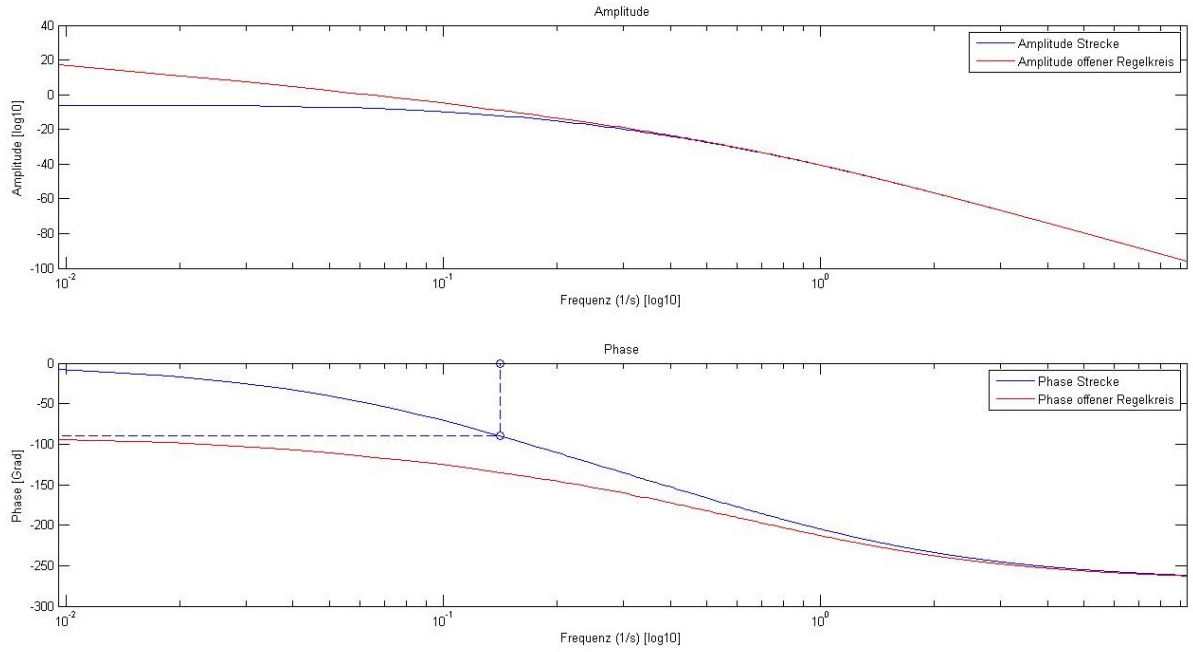

Abbildung 4: Frequenzgang der offenen Strecke

Auf dem offenen Regelkreis (Abbildung 5) wird nun die Frequenz bei ϕ_{st} mithilfe der Tabelle 1 auf Seite 10 gesucht (rot gestrichelte Linie). Die gefundene Frequenz ist $0.0611 s^{-1}$.

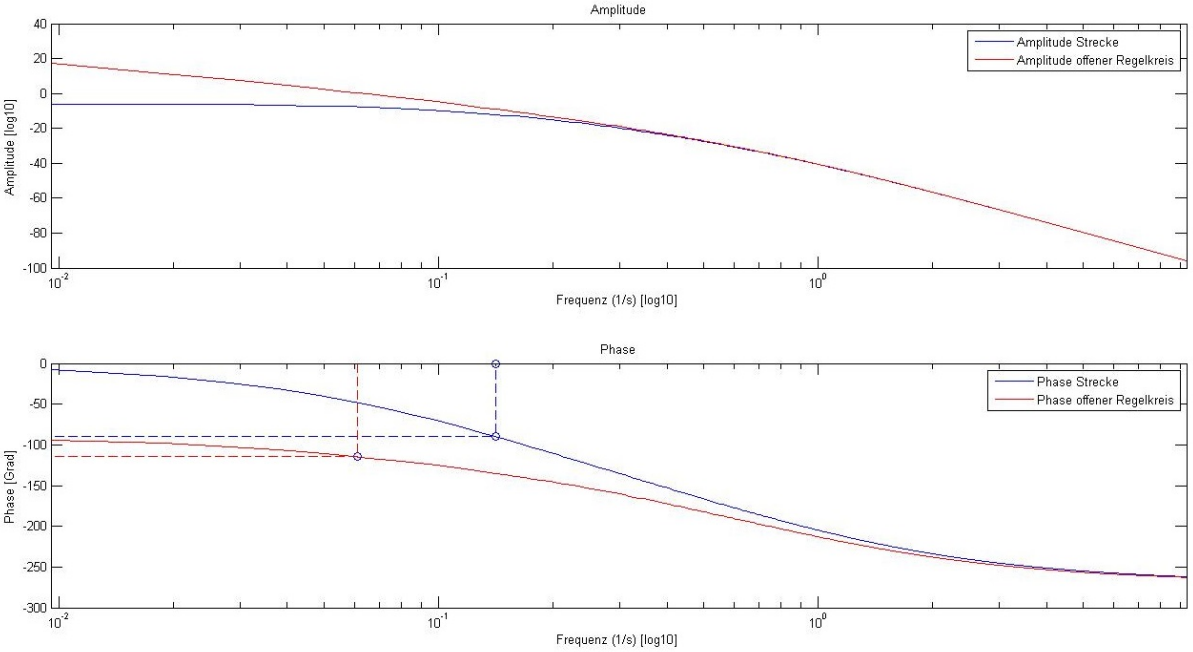

Abbildung 5: ω bei ϕ_{st}

Schlussendlich wird nun die entsprechenede Verstärkung zur gefundenen Frequenz aus dem offenen Amplitudengang herausgelesen (Abbildung 6). In unserem Beispiel ergibt dies eine Verstärkung von 1.0390. Um diesen Faktor wird die Verstärkung des Reglers gemindert, dies ergibt den letzten benötigten Parameter Kr.

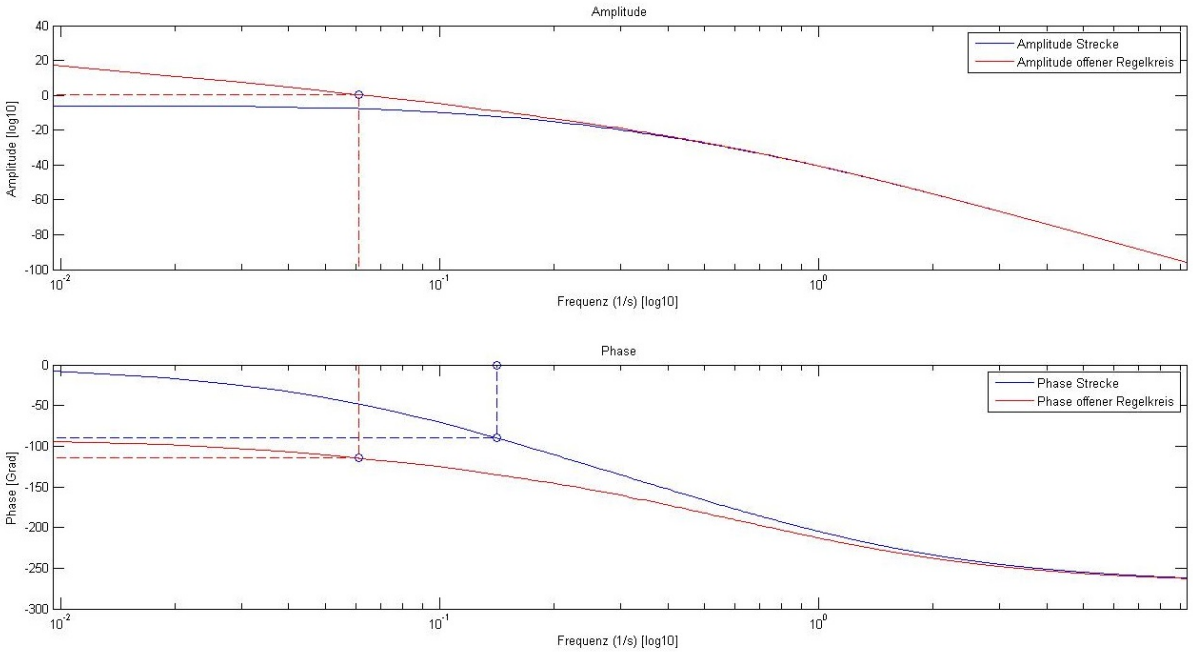

Abbildung 6: Bestimmung von Kr

Die Endresultate dieser Berechnung sind:

$Tn = 7.0647s$
$Kr = 0.9625$

2.3.2 Vorgehen PID-Regler mit Matlab

Die Berechnung eines PID-Reglers verläuft ähnlich wie die eines PI-Reglers. Folglich trifft dies auch in Matlab zu. Für das Beispiel werden wieder die gleichen Eingabeparameter verwendet wie schon beim PI-Regler:

$Ks = 0.5$
$Tu = 2.5$
$Tg = 18.3$
ϕ_{st}=-114.6°

Im Gegensatz zum PI-Regler sind alle Ausgabeparameter in der bodekonformen Darstellung. Eine genauere Erklärung dazu ist im Anhang, Sektor A.2, Seite 38 zu finden.

Zunächst werden nun die Zeitkonstanten sowie die Ordnung der Strecke mithilfe der Sani Methode bestimmt. Bei unserer Strecke ergibt dies eine Strecke 3. Ordnung und die folgenden Zeitkonstanten:

$T1 = 1.0688s$

$T2 = 3.3484s$

$T3 = 10.4901s$

Nun wird nach dem -135° Punkt auf dem Phasengang gesucht. Dies ist in Abbildung 7 ersichtlich und ergibt uns folgende Frequenz: $0.2987s^{-1}$.

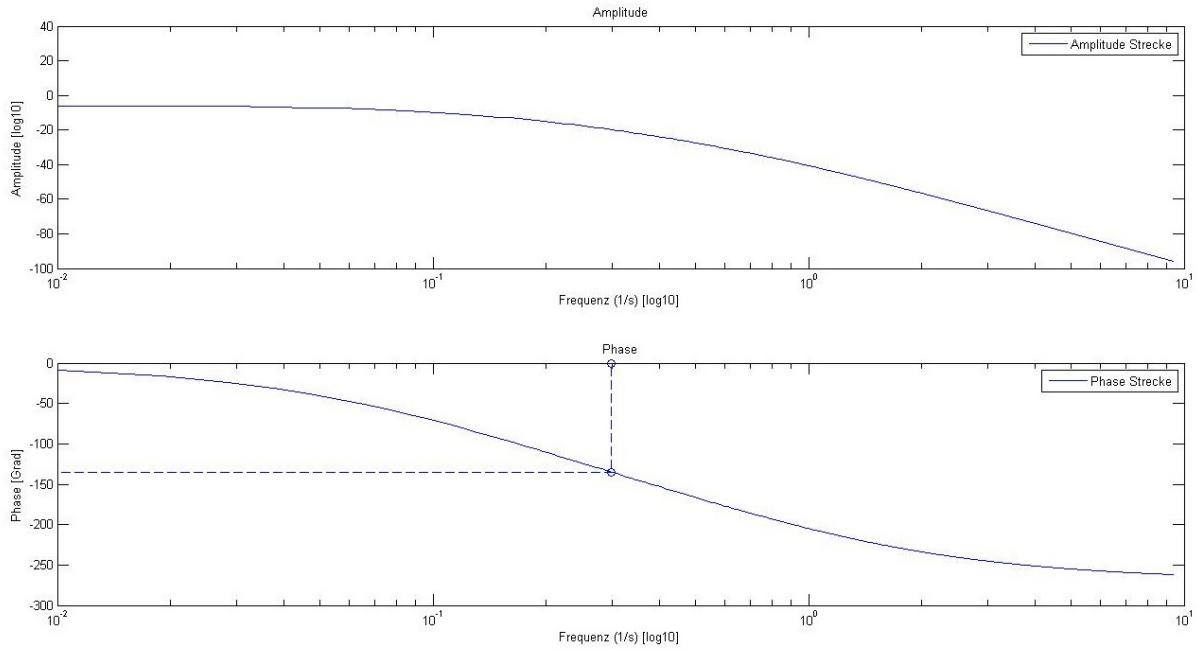

Abbildung 7: Amplituden- und Phasengang der Regelstrecke

Bevor nun jedoch der offene Frequenzgang berechnet wird, muss β bestimmt werden. Die genaue Berechnung dazu ist im Anhang, Kapitel A.1 auf Seite 36 ersichtlich. Nach Ausführen dieser Berechnung erhält man $\beta = 0.3192$.

Mit β ist es einem möglich, $Tnk = 10.4902s$ und $Tvk = 1.0688s$ zu berechnen. Weiter wird noch Tp bestimmt (siehe Kapitel 2.3, Seite 9). In diesem Beispiel wurde Tp eine Dekade kleiner als Tvk gewählt, folglich ist der Wert von $Tp = 0.107s$. Mit diesen Werten und $Krk = 1$ wird nun der offene Amplituden- und Phasengang berechnet (Abbildung 8).

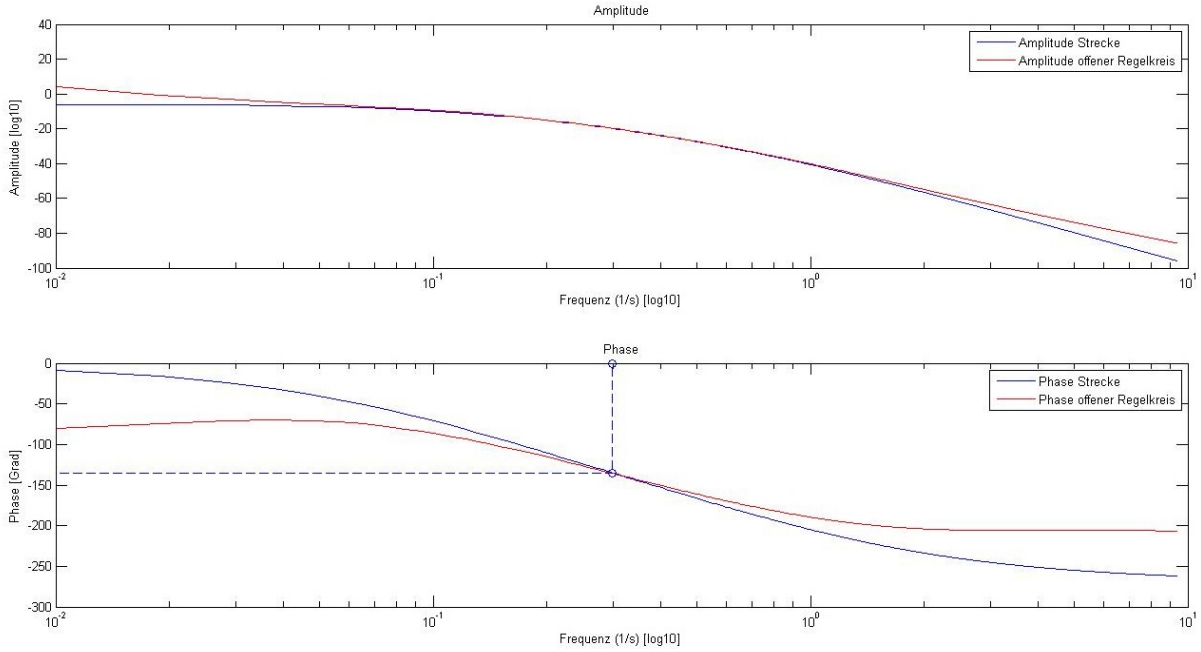

Abbildung 8: Frequenzgang der offenen Strecke

Als nächster Schritt wird wieder die Tabelle 1 auf Seite 10 zu Hilfe genommen. Daraus wird das gewünschte Überschwingen ausgewählt und die entsprechende Frequenz auf dem offenen Phasengang gesucht (siehe Abbildung 9). Die gefundene Frequenz lautet: $0.1317s^{-1}$.

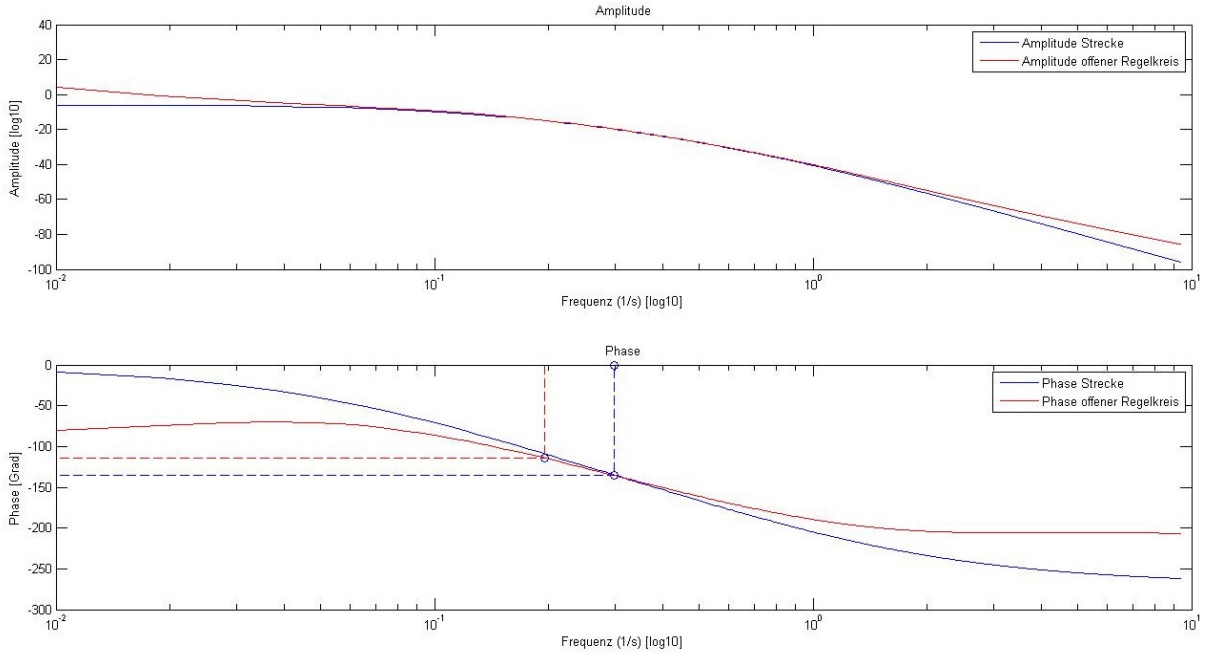

Abbildung 9: ω bei ϕ_{st}

Zu der gefunden Frequenz liest man nun die Verstärkung von 0.3312 heraus. Der Kehrwert davon ergibt den letzten Faktor $Krk = 3.0194$. (Abbildung 10)

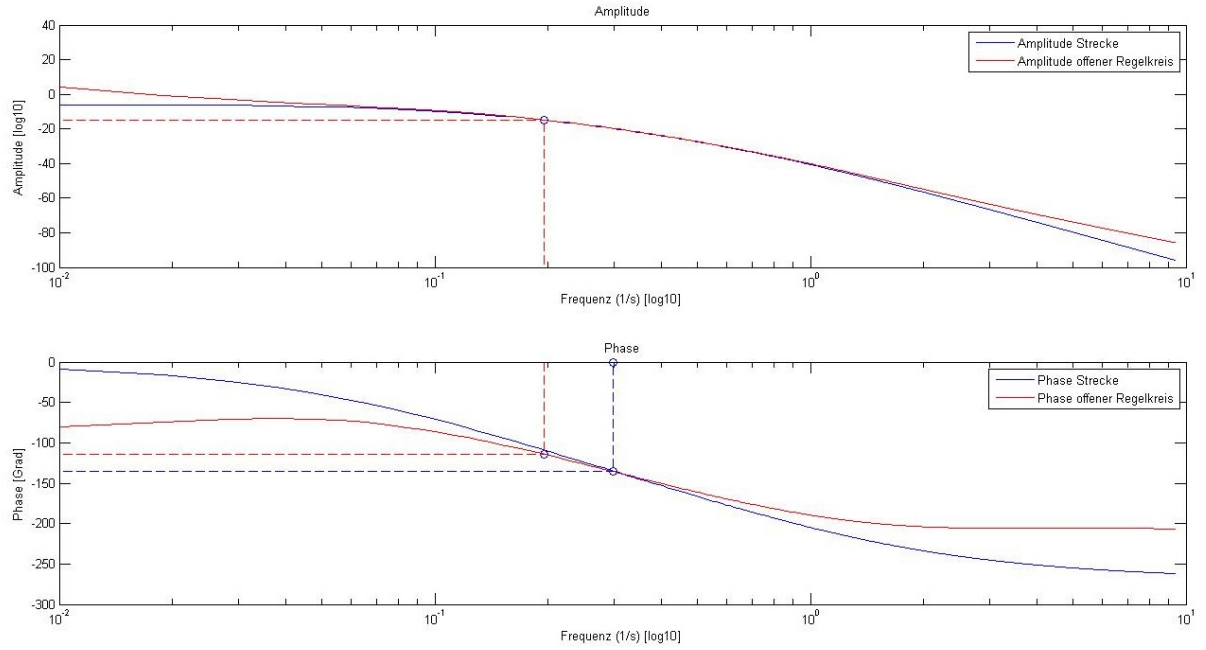

Abbildung 10: Bestimmung von Krk

Die Endresultate dieser Berechnung sind:

$Tnk = 10.4902$
$Tvk = 1.0688$
$Krk = 3.0194$

Wie schon erwähnt sind diese Angaben alle in der Bodekonformen Darstellung. Umgewandelt in die Reglerkonforme Darstellung ergibt dies folgende Werte:

$Tn = 11.4521$
$Tv = 0.8721$
$Kr = 3.2963$

Den genauen Verlauf für die Umrechnung zur Regelkonfomren Darstellung ist im Anhang, Sektor A.2, Seite 38 zu finden.

3 Software

Im folgenden Abschnitt wird der Aufbau der Software beschrieben. Die Software basiert auf dem Model-View-Controller Entwurfsmuster, welches für GUI-Applikationen aufgrund seiner hohen Flexibilität und Wiederverwendbarkeit als quasi-Standard gilt. Das Model-View-Controller Prinzip basiert auf den drei Hauptklassen $Model$, $View$ und $Controller$, welche kurz erläutert werden:

Model: Das $Model$ enthält die Daten und Berechnungsmethoden, welche für die Software benötigt werden. Diese Daten sind unabhängig von der grafischen Darstellung der Software. Falls Daten im $Model$ geändert werden, informiert das $Model$ die Klasse $View$, welches die Daten für den Benutzer darstellt.
View: Die $View$ beinhaltet und visualisiert die Benutzeroberfläche. Sie erhält dazu die Daten der Klasse $Model$, welche grafisch dargestellt werden, und leitet Eingaben an die Klasse $Controller$ weiter. In der Klasse $View$ werden keine Berechnungen durchgeführt. Das Model ist mit der $View$ mittels Observable-Observer Entwurfsmuster verknüpft, wobei das $Model$ als Observable und die $View$ als Observer agiert.
Controller: Der $Controller$ überprüft die Benutzereingaben und leitet diese, falls sie korrekt sind, zur weiteren Berechnung an das $Model$ weiter.

Im folgenden werden wiederholt Ausschnitte des Klassendiagramms gezeigt, das komplette Klassendiagramm findet sich im Anhang (**FEHLT NOCH!!!**).

3.1 Model

Das $Model$ ist das Herzstück sämtlicher Berechnungen. Dazu führt es folgende Funktionen aus:

- Die Koordination der unabhängigen und parallel verlaufenden Berechnungen der Reglertypen und deren Schrittantworten.
- Das Verwalten der dazu notwendigen Objekte, beispielsweise die Sani-Kurven.
- Die Implementation des iterativen Approximationsverfahren zur genauen Bestimmung des Überschwingens.

Abbildung 11: Das Klassendiagramm des Models

Als erstes wird die zu Grunde liegende Terminologie erläutert:

Unter einer *Calculation* versteht man den Prozess, einen Regler zu berechnen, ein geschlossener Regelkreis zu erstellen und davon die Schrittantwort zu berechnen. Es gibt meistens mehrere *Calculations* für jede *Simulation*. Eine *Calculation* wird von der inneren Klasse *CalculationCycle* implementiert.

Eine *Simulation* beinhaltet mehrere *Calculations* (nämlich genau soviele, wie es Reglertypen hat). Unter *Simulation* versteht man den Prozess, alle zu den entsprechenden Parametern gehörenden *Calculations* durchgeführt zu haben. Da die *Calculations* rechenintensiv, aber von einander unabhängig sind, implementiert die Klasse *CalculationCycle* das Interface *Runnable*, damit die Berechnungen mittels einem Thread-Pool parallel verlaufen können. Eine *Simulation* wird direkt von der *Model*-Klasse übernommen.

Der Prozess, die Schrittantwort eines geschlossenen Regelkreises zu berechnen, fängt mit der Erstellung eines *Plant*-Objektes, welche die zu regelnde Strecke beschreibt, an. Es gibt sehr viele Möglichkeiten und Verfahren, ein passender Regler für die Strecke zu berechnen, was auch von der komplexen Vererbungshierarchie der verschiedenen *AbstractControllerCalculator*-Klassen reflektiert wird. Was zwischen den Rechner-Klassen gemeinsam bleibt ist, dass sie als Eingabe ein *Plant*-Objekt entgegennehmen und als Ausgabe ein *Controller*-Objekt herausgeben. Es kann zum Beispiel mit Hilfe der *ZellwegerPID*-Klasse ein PID *Controller*-Objekt mittels der Zellweger Methodik erstellt werden. Mit dem *Controller*-Objekt kann zusammen mit dem *Plant*-Objekt ein geschlossener Regelkreis mit der Klasse *ClosedLoop* erstellt werden. Diese Klasse erlaubt das Berechnen einer Schrittantwort. Somit ist der Ablauf einer *Calculation* beschrieben.

Die *Model*-Klasse hat viele Methoden, die das Einstellen einer *Calculation* beziehungsweise einer *Simulation* steuern kann. Die wichtigsten Methoden sind folgend aufgezählt:

- *Model::setPlant()*: Erlaubt es dem Benutzer, die Strecke mit den Parametern T_u, T_g und K_r zu definieren.

- *Model::setRegulatorType()*: Mit dieser Methode kann der Reglertyp ausgewählt werden. Dabei kann zwischen I, PI und PID ausgewählt werden. Insgesamt unterstützt „Easy-PID“ 15 Methoden, einen Regler zu berechnen. Diese Methode filtert dabei die Methoden, so dass nur die Regler simuliert werden, die mit dem ausgewählten Reglertyp übereinstimmen.
- *Model::setOversing*: Erlaubt es dem Benutzer, das gewünschte Überschwingen in Prozent einzugeben. Dabei wird bei der Zellweger-Methode der Parameter K_r iterativ angepasst, bis das Überschwingen mit dem angegebenen Wert übebreinstimmt. Die Faustformeln werden von dieser Methode nicht beeinflusst.

Die Methode *Model::setPlant()* erlaubt es den Benutzer, die Strecke mit den Parametern T_u, T_g, und K_r zu definieren.

Ausserdem kann noch die parasitäre Zeitkonstante angegeben werden. Die Zellweger-Regler berechnen T_p, indem sie diesen Faktor mit $T_v k$ multiplizieren. Die Faustformeln berechnen T_p, indem sie diesen Faktor mit T_v multiplizieren. Gewöhnlich wird hier etwa 10% benutzt.

3.1.1 Regelstrecke

Eine Regelstrecke, im Programm vom Englischen als *Plant* benannt, beschreibt die zu regelnde Strecke anhand einer *TransferFunction* (Übertragungsfunktion), einer Liste von *timeConstants* (Zeitkonstanten) und natürlich anhand den Parametern T_u, T_g und K_r. Bei der Instanzierung eines *Plant*-Objektes werden die drei Parameter zusammen mit einer Instanz der Klasse *SaniCurves* übergeben.

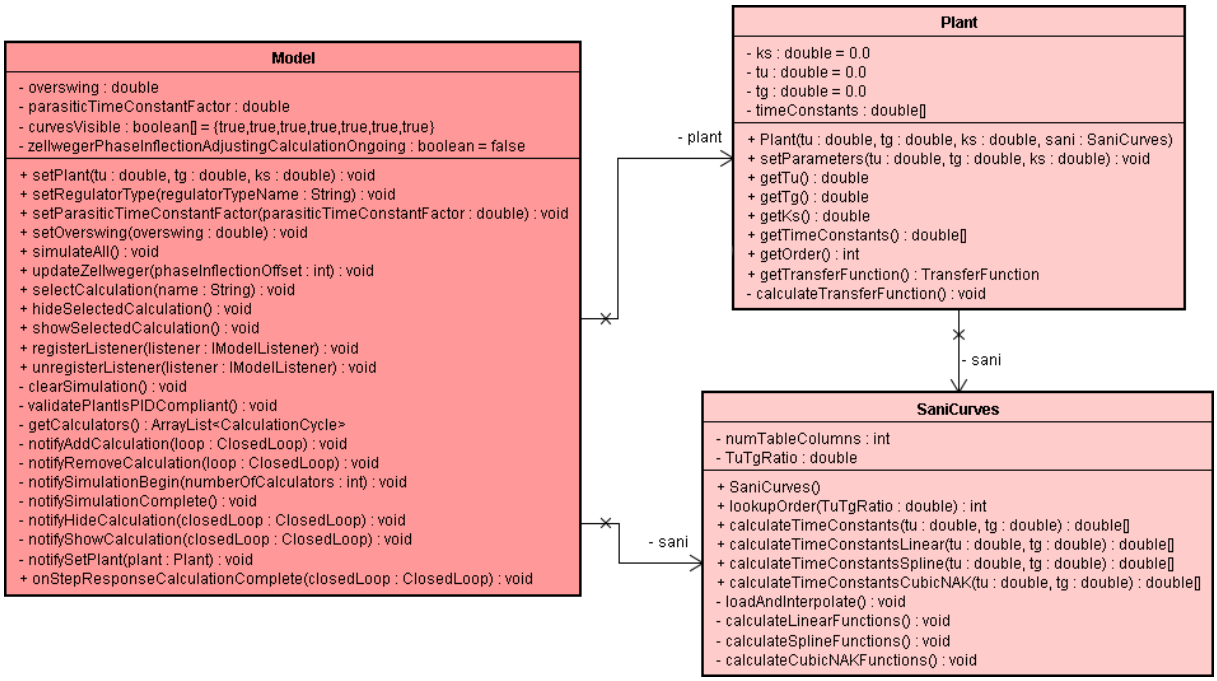

Abbildung 12: Das Klassendiagramm der Regelstrecke

Die Sani-Kurven sind die Grundlage zur Berechnung der *TransferFunction* und die *TransferFunction* ist die Grundlage zur späteren Berechnung der Schrittantwort.

Die Sani-Kurven werden nicht vom Programm selbst berechnet, sondern werden zu Programmbeginn von einer Text-Datei importiert, welche wiederum von Matlab exportiert wurde. Da die Kurven als diskrete Punkte gespeichert sind, werden sie beim Importieren in Java kubisch interpoliert, um eine höhere Auflösung zu ermöglichen. Das Interpolieren erfolgt mittels der

SplineNAK-Klasse, welches von C in Java übersetzt wurde. (http://www.pcs.cnu.edu/ bbradie/cpp/interp.C).
Da diese Interpolationsmethode mit der in Matlab verwendeten übereinstimmt, wird ein stabiler Benchmark für Tests ermöglicht.

Das *SaniCurves*-Objekt wird nur einmal im *Model* erstellt, um die Festplattenzugriffszeit zu minimieren. Das *Model*-Objekt hat zu jedem Zeitpunkt ein Referenz auf den aktuellen *Plant*.

Jedes mal, wenn sich mindestens einer der drei Parameter T_u, T_g oder K_r im *Plant*-Objekt ändert - sei es durch eine Instanzierung oder durch Aufrufen der Methode *Plant::setParameters()* - werden mittels *SaniCurves* die Zeitkonstanten der Strecke und die Übertragungsfunktion neu berechnet.

3.1.2 Regler

Der Regler, im Programm vom Englischen als *Controller* benannt, beschreibt den Regler der Strecke anhand einer *TransferFunction* (Übertragungsfunktion).

Nicht alle Regler haben gemeinsame Parameter. Bei den verschiedenen Reglertypen sind die Parameter anders benannt oder gar nicht vorhanden. Beispielsweise hat der PI-Regler die Parameter T_u und T_g, aber es Fehlen die PID-Parameter T_v und T_p. Ausserdem wird abhängig vom Reglertyp die Übertragungsfunktion anders berechnet. Es gibt aber auch Gemeinsamkeiten, wie der Name oder die Farbe des Reglers, und die Tatsache, dass die Parameter in einer Tabelle eingefügt werden müssen. Für den Rest des Programms ist ausserdem nciht ersichtlich, um was für einen Reglertyp es sich handelt. Für die Schrittantwort wird lediglich die Übertragungsfunktion verwendet.

Aus diesen Gründen arbeiten wir hier mit Vererbung. Die Klassen *ControllerI*, *ControllerPI* und *ControllerPID* implementieren jeweils die Details der entsprechenden Reglertypen. Die Superklasse *AbstractController* stellt eine gemeinsame Schnittstelle zur Verfügung.

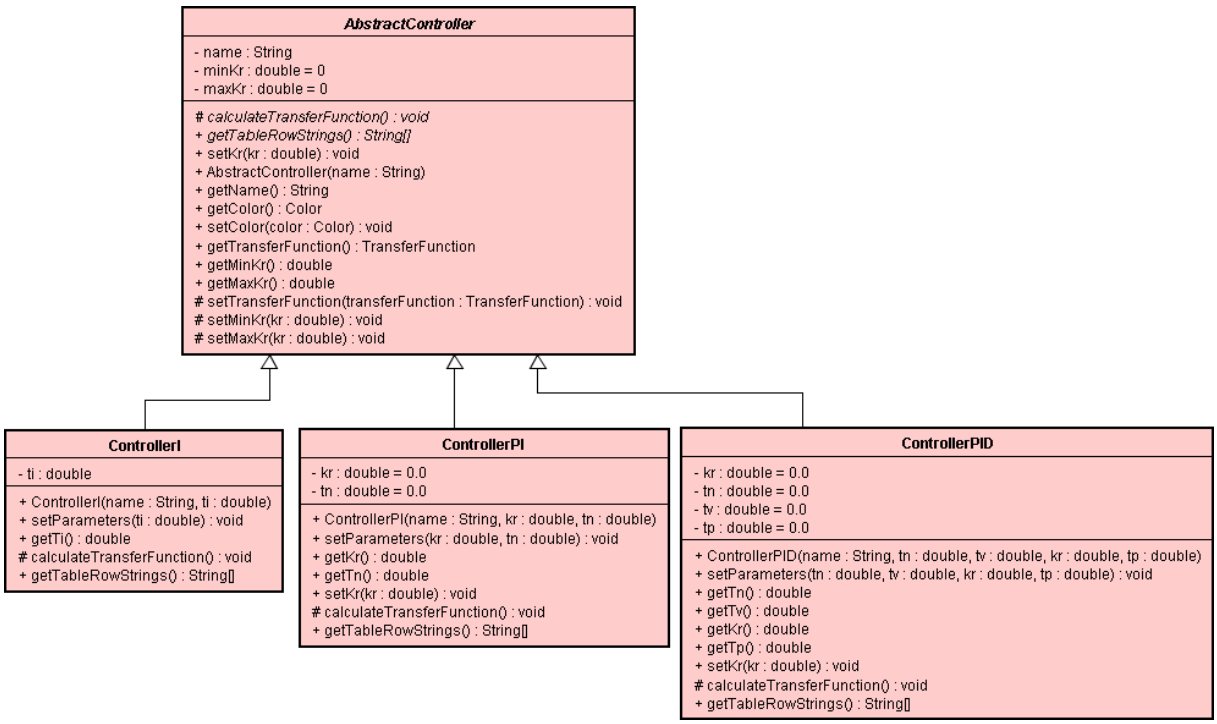

Abbildung 13: Das Klassendiagramm des Reglers

Immer wenn sich die Reglerparameter ändern - also dann wenn der Regler instanziert wird - wird seine Übertragungsfunktion berechnet. Diese ist mittels der Methode *AbstractController::getTransferFunction()* abrufbar.

Jeder Regler definiert eine Farbe und einen Namen für den Plot. Auf diese Informationen kann mittels der Methoden *AbstractController::getName()* und *AbstractController::getColor()* zugegriffen werden.

Die erbenden Klassen implementieren die Methode *getTableRowStrings()*, welche erlaubt, die Reglerparameter über die Superklasse in die Tabelle zu einzufügen ohne zu wissen, um was für einen Reglertypen es sich handelt.

Wurde der Regler mittels einer Zellwegermethode berechnet, so kann ein Fenster für den Minimal- und den Maximalwert des Parameters K_r über die Methoden *AbstractController::getMinKr()* und *AbstractController::getMaxKr()* geholt werden. Dieses Fenster wird für das iterative Approximieren des Überschwingens verwendet. Alle anderen Berechnungsmethoden, also die Faustformeln, berechnen diese Fenster nicht.

3.1.3 Reglerberechung

Die Reglerberechnung ist hierarchisch und technisch das komplizierteste Teilstück des Programms. Es gibt insgesamt 15 verschiedene Klassen, die auf 15 verschiedene Arten einen passenden Regler für eine Strecke berechnen können. Alle erben von der Klasse *AbstractControllerCalculator*.

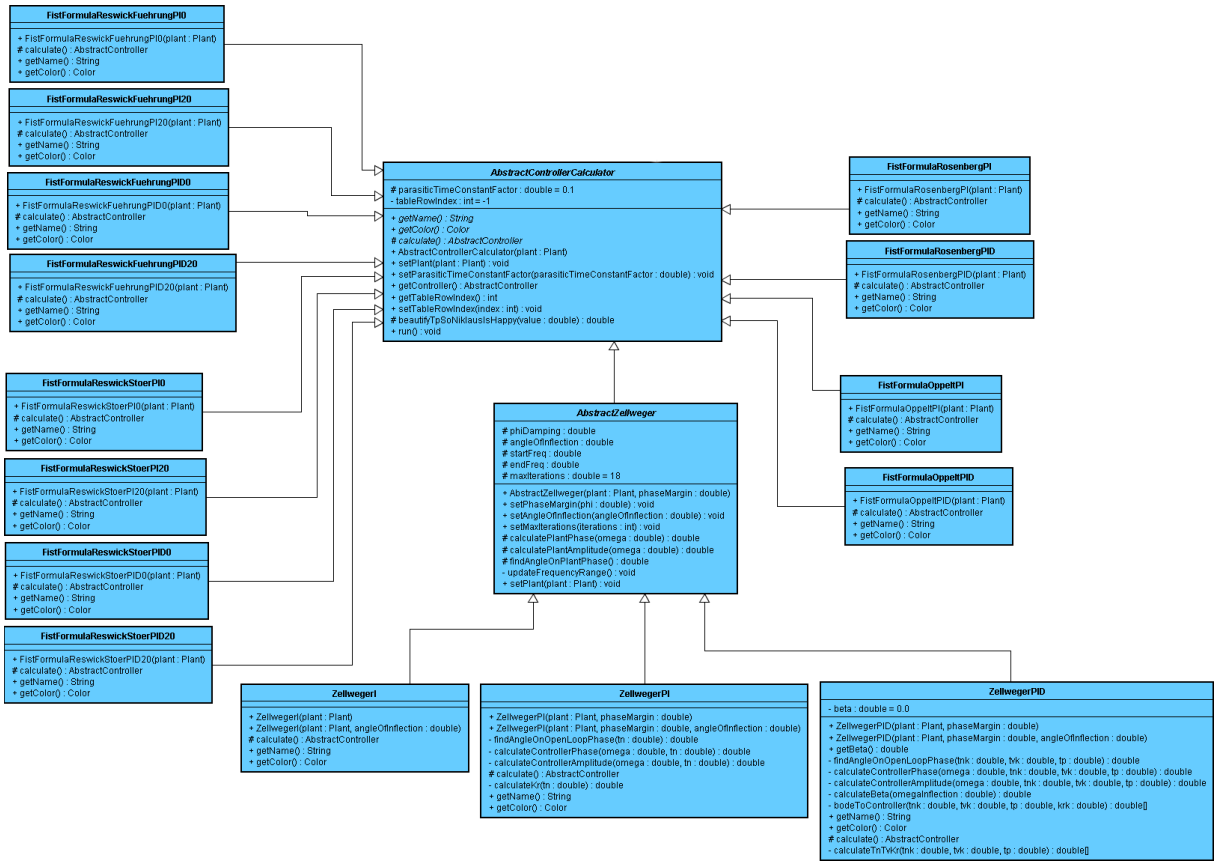

Abbildung 14: Das Klassendiagramm zur Reglerberechnung

Grundlegend kann ein neues *ControllerCalculator*-Objekt mit einem *Plant*-Objekt (die Strecke) instanziert werden, es kann *AbstractControllerCalculator::run()* aufgerufen werden, und es kann

mittels der Methode *AbstractControllerCalculator::getController()* der resultierende Regler als *AbstractController* geholt werden.

Wie der Regler berechnet wird und welchem Typ der Regler angehört ist dabei durch die Vererbung abstrahiert.

3.1.4 Regelkreis

Mit der Klasse *ClosedLoop* kann mit einem *Plant*-Objekt und einem *AbstractController*-Objekt ein geschlossener Regelkreis erstellt werden. Auch der *ClosedLoop* hat eine Übertragungsfunktion (ein *TransferFunction*-Objekt), welche aus den den Übertragungsfunktionen der Strecke und des Reglers berechnet wird.

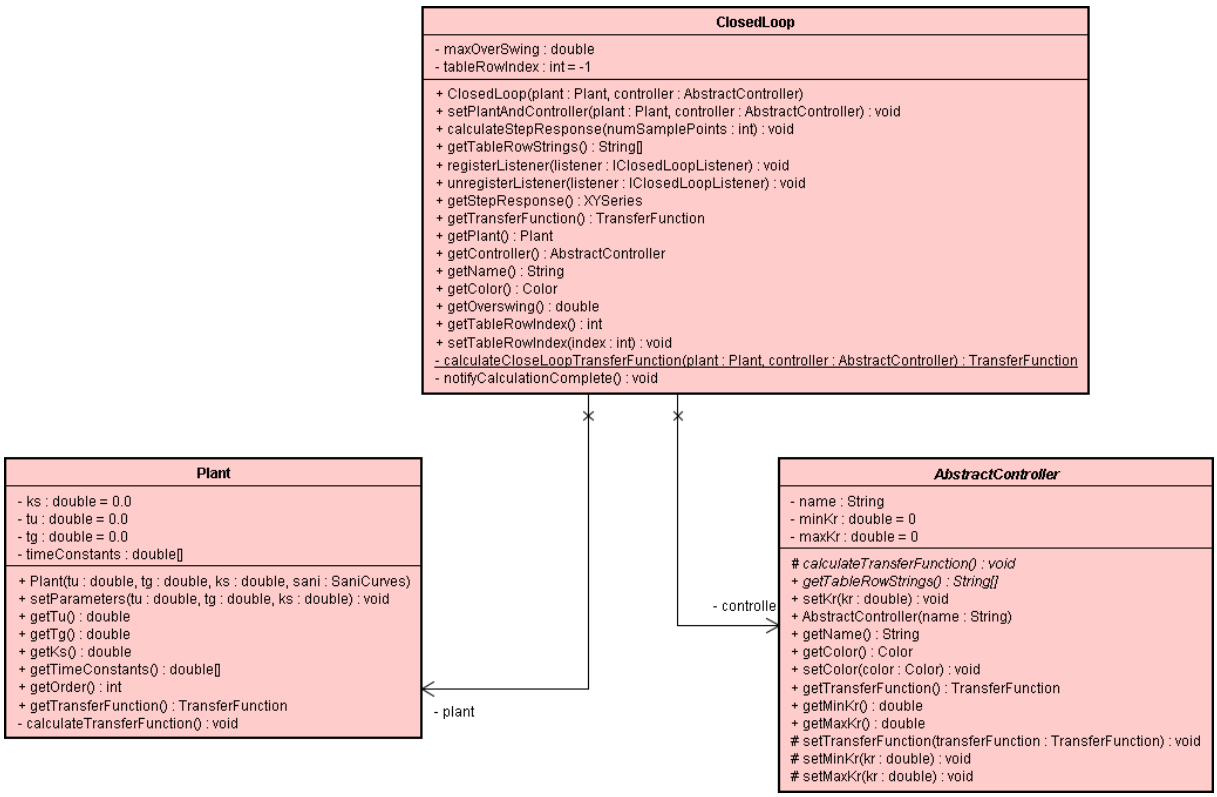

Abbildung 15: Das Klassendiagramm zum Regelkreis

Immer wenn sich das *Plant*-Objekt oder das *AbstractController*-Objekt ändern, sei es durch eine Instanzierung oder durch Aufrufen der Methode *ClosedLoop::setPlantAndController()*, wird aus den beiden Übertragungsfunktionen des *Plant*-Objektes und des *AbstractController*-Objektes die Übertragungsfunktion des geschlossenen Regelkreises berechnet.

Durch Aufrufen der Methode *ClosedLoop::calculateStepResponse()* wird dann die Sprungantwort mittels der Residuenmethode berechnet. Die resultierende Schrittantwort kann mit der Methode *ClosedLoop::getStepResponse()* als *XYSeires* für JFreeChart geholt werden.

Weiter wird bei der Berechnung der Sprungantwort das maximale Überschwingen gemessen. Dieser Wert kann mit der Methode *ClosedLoop::getOverswing()* geholt werden.

3.1.5 Resultate

Die Resultate der Berechnungen sind die Reglerparameter T_n, T_v und K_r, die parasitäre Zeitkonstante T_p, das Überschwingen, der Name der verwendeten Methode und die Schrittantwort.

Die Schrittantworten werden mit JFreeChart geplottet und die restlichen Parameter können mit der Methode *AbstractController::getTableRowStrings()* als String-Array geholt werden und werden im GUI in einer Tabelle gefügt.

3.2 View

Die Klasse *View* und ihre Unterklassen sind zuständig für die Darstellung der Benutzeroberfläche. Am oberen Fensterrand befindet sich die *MenuBar*, welche sich über die gesamte Breite erstreckt. Sie enthält Einstellungsmöglichkeiten und weiterführende Informationen. Auf der linken Seite befinden sich zwei Panels für die Ein- und Ausgabe. Im *InputPanel* können die Eingabeparameter eingetragen, der gewünschte Regler und das Überschwingen gewählt und die Simulation mit einem Button gestartet werden. Das *OutputPanel* stellt alle berechneten Graphen mit der Angabe ihrer Kennwerte in einer Tabelle dar und besitzt einen Trimmer für das manuelle justieren des Phasenwinkels. Auf der rechten Seite des Fensters werden die Graphen mit der Klasse *GraphPanel* angezeigt. Unter diesem Plot befindet sich das *GraphDisplayPanel* und das *GraphSettingPanel*, welche beide für für die Auswahl der darzustellenden Kurven verantwortlich sind. Im folgenden Abschnitt wird die Darstellung des GUIs (Graphical User Interface) genauer erläutert.

3.2.1 Anforderungen an die Benutzeroberfläche

Die Benutzeroberfläche wurde so umgesetzt, dass der Anwender das Tool intuitiv bedienen kann und es einen grafisch ansprechenden Eindruck hinterlässt. Des Weiteren wird nach Wunsch des Auftraggebers das Programm in einem einzelnen Fenster dargestellt. Dies ermöglicht es dem Anwender alles zu überblicken und mit einem einzigen Printscreen das gesamte Programm (Einstellungen, Graph) in einer Dokumentation festzuhalten oder mit anderen Methoden zu vergleichen. Dies führt zu einer einfacheren Handhabung der erstellten Simulationen.

Vom Auftraggeber wurden folgende Punkte für die Benutzeroberfläche gewünscht:

- Es soll lediglich der Graph der Schrittantwort dargestellt werden.
- Eine automatische Skalierung über einen sinnvollen Bereich für den Graphen soll implementiert sein. Falls dies nicht gelingt, sollten die Achsen manuell einstellbar sein.
- Das im Graphen ausgemessene Überschwingen soll in einer Tabelle dargestellt werden.
- Die Fenstergrösse soll veränderbar sein, zusätzlich sollte eine Miniversion enthalten sein.
- Alle Informationen sollten in einem Fenster angezeigt werden, sodass mit einem Blick alles erkannt werden kann.

3.2.2 Erstellung der Benutzeroberfläche

Für die Erstellung der Benutzeroberfläche wurden hauptsächlich die Swing-Klassen der Java Foundation Classes (JFC) verwendet. Sie enthalten die notwendigen Komponenten wie Textfelder, Buttons usw., mit denen grafische Oberflächen erstellt werden können. Erkennbar sind sie auch durch das „J“, welches dem Namen vorangestellt ist (beispielsweise JButton). Auf die ausführliche Beschreibung dieser Standardkomponenten wird bewusst verzichtet und es werden nur die zusätzlichen Komponenten erläutert, welche verwendet wurden.
Für die Platzierung der Komponenten gibt es in Java verschiedene Layoutmanager, um die Komponenten anzuordnen. Um einen kurzen Überblick über diese Manager zu geben, werden die Verwendeten in der folgenden Aufzählung kurz beschrieben:

1. *GridBagLayout*: Er ist der mächtigste LayoutMangager aus dem Java AWT-Package und ermöglicht es, die Komponenten in ein anpassbares Raster (Grid) zu setzen. Die einzelnen Spalten können dabei unterschiedlich breit und die einzelnen Zeilen unterschiedlich hoch sein. Die Komponenten werden in eine Zelle gesetzt und können von dort in X- und Y-Richtung weitere Zellen überlappen. Weiter gibt es viele zusätzliche Anpassungsmöglichkeiten.

2. *GridLayout*: Setzt die Komponenten in ein gleichmässiges Raster (Grid), bei dem alle Zellen die selbe Grösse haben.
3. *BorderLayout*: Besitzt fünf Bereiche (Norden, Süden, Osten, Westen, Zentrum), in welche Komponenten platziert werden können.
4. *WrapLayout*: Alle Komponenten werden nacheinander von links nach rechts eingefüllt. Beim verändern der Fenstergrösse gibt es Zeilenumbrüche, wenn die Breite des Panels nicht mehr ausreicht. Dieses Layout gehört nicht zu den Standard-Typen und wird bei der Erläuterung des *GraphDisplayPanels* genauer beschrieben.

Für weitere Informationen zu Layoutmanagern wird auf die Dokumentation von Oracle verwiesen(QUELLE https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html).

Aufbau der Benutzerfläche
Der Aufbau des GUIs wurde so erstellt, dass die gesamte grafische Benutzeroberfläche in einem JFrame platziert ist. Dieses Frame enthält jeweils eine Instanz der Klasse *MenuBar* und *View*. Die *View* besteht wiederum aus den vier JPanels *InputOutputPanel*, *GraphPanel*, *GraphDisplayPanel* und *GraphSettingPanel*. Das *InputOutputPanel* fasst das *InputPanel* und das *OutputPanel* zu einem gemeinsamen JPanel zusammen. Auf die Layoutmanager wird bei der Beschreibung der Einzelnen Panels genauer eingegangen.

In der folgenden Grafik wurde der gesamte Zusammenhang dargestellt:

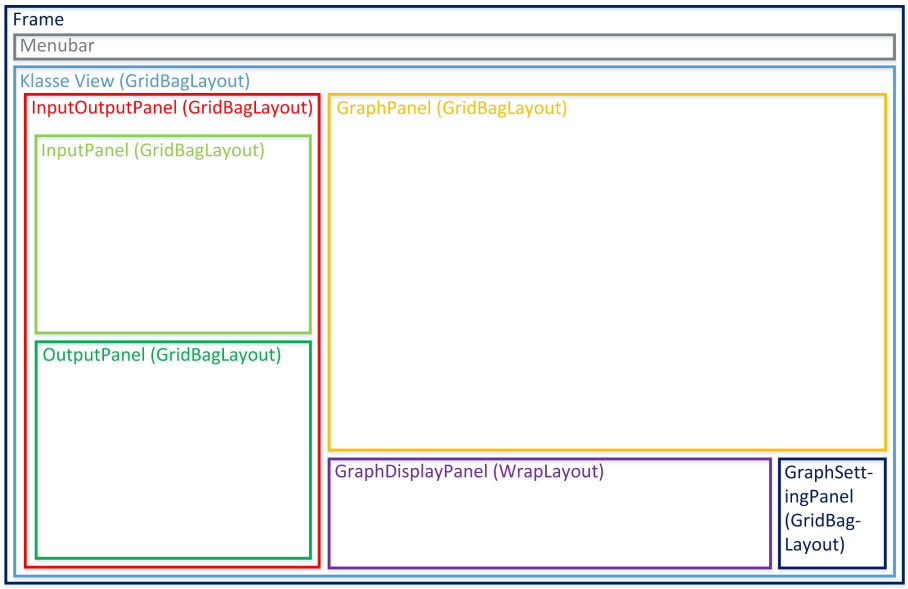

Abbildung 16: Aufbau des GUIs

3.2.3 Aufgabe der einzelnen Panels

Die Benutzeroberfläche beinhaltet, wie vorhin beschrieben, mehrere Panels. Das *InputPanel* dient zur Eingabe der Kenngrössen der Regelstrecke und das *OutputPanel* zur Anzeige der berechneten Reglerparameter und zur nachträglichen Manipulation des Phasenwinkels. Das *GraphPanel* visualisiert die Schrittantwort, das *GraphDisplayPanel* ermöglicht die Auswahl der anzuzeigenden Kurven und das *GraphSettingPanel* besitzt drei Buttons für die Einstellungen zum Graphen. Ein Grossteil der verwendeten Komponenten stammt aus der Swing Klasse von Java, alle zusätzlichen werden speziell erläutert. Folgend werden die Panels genauer beschrieben:

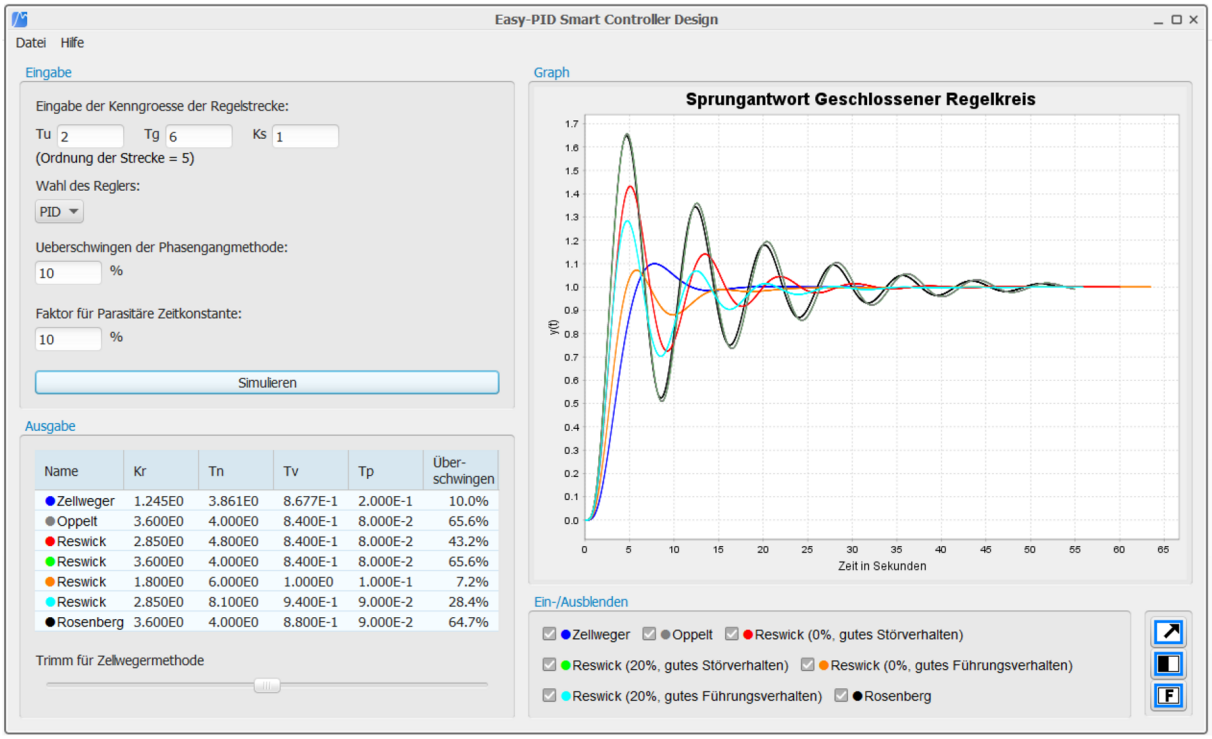

Abbildung 17: Übersicht über das GUI

InputOutputPanel: Die Aufgabe dieses im GridBagLayout erstellten Panels besteht einzig darin, das *InputPanel* und das *OutputPanel* in einem gemeinsamen Panel zu vereinen. Dieses Panel wurde erstellt, um die Handhabung im GridBagLayout der Klasse *View* zu verbessern.

InputPanel: Dieses Panel wurde im GridBagLayout erstellt und besitzt einen Rahmen mit dem Titel „Eingabe“. Es enthält zuoberst die drei Eingabefelder für die Parameter *Tu* (1), *Tg* (2) und *Ks* (3) der Regelstrecke. Anschliessend folgt eine leere Zeile, die zu Beginn keinen Inhalt hat. Sie zeigt nach einer Simulation den Grad der Berechnung an und bei fehlerhaften Eingaben rote Hinweise für den Anwender. Als nächstes folgt das Dropdown-Menü für die Auswahl des Reglers (4) (*I*, *PI* oder *PID*). In den nächsten beiden Textfeldern(4 und 5) kann das Überschwingen und die Parasitäre Zeitkonstante eingetragen werden. Diese beiden Eingaben sind abhängig von der Auswahl des Reglers und deshalb nicht für alle Regler aktiviert. Wenn die Felder deaktiviert sind, so werden sie ausgegraut und es können keine Werte eingetragen werden. Beim I-Regler werden beide und beim PI-Regler nur das Feld für die parasitäre Zeitkonstante deaktiviert. Beim PID-Regler sind beide aktiviert. Am Ende dieses Panels befindet sich über die gesamte Breite die Schaltfläche zum Starten der Simulation (7).

Alle Beschriftungen sind JLabels, alle Textfelder sind JFormatedDoubleTextfields, die Schaltfläche ist ein JButton und für die Wahl des Reglers wurde eine JComboBox verwendet. Weiter verfügen alle Textfelder über ToolTips (Zusatzinformationen bei Objekten), welche dem Nutzer zusätzliche Informationen bieten.

OutputPanel: Dieses Panel wurde im GridBagLayout erstellt und besitzt einen Rahmen mit dem Titel „Ausgabe“. In einer übersichtlichen Tabelle werden alle Kennwerte jeder simulierten Kurve aufgelistet. Diese Tabelle besteht aus den sechs Spalten mit den folgenden Inhalten:

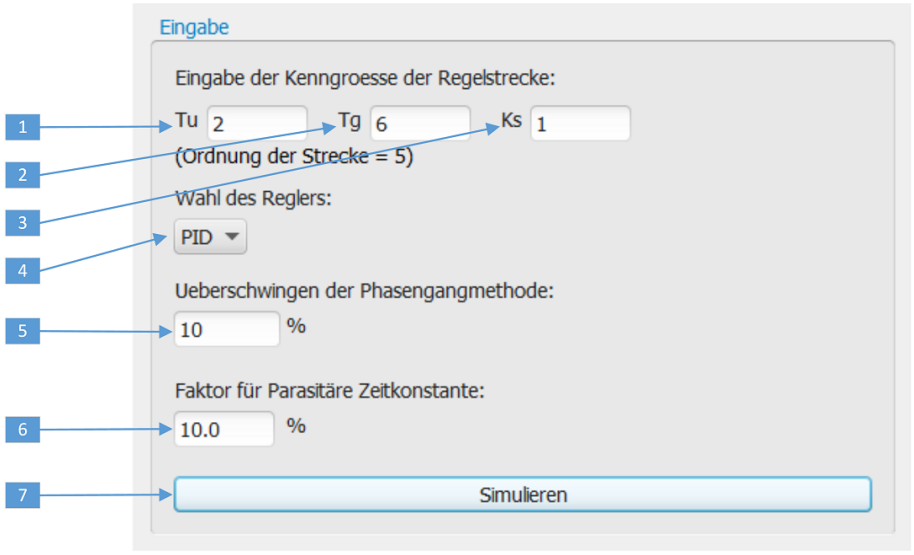

Abbildung 18: InputPanel

1. *Name*: Als erstes steht ein grosser farbiger Kreis in der Farbe der dazugehörigen Kurve. Dadurch kann der Anwender den Zusammenhang zwischen Tabelle, Graph und Auswahl der Kurven leicht erkennen. Nach dem Kreis folgt im selben Feld der Name des Reglers.
2. *Kr*: Enthält den berechneten Wert für die Reglerverstärkung.
3. *Tn*: Enthält den berechneten Tn-Reglerparameter
4. *Tv*: Enthält den berechneten Tv-Reglerparameter
5. *Tp*: Enthält den berechneten Tp-Reglerparameter
6. *Überschwingen*: Enthält das gemessene Überschwingen in Prozent

Für die nachträgliche Optimierung folgt unter der Tabelle ein Slider (*Trimm für Zellwegermethode*), mit dem der Phasenwinkel manuell verändert werden kann (7). Die Tabelle des *OutputPanels* ist eine JTable mit einem TableHeader und der Slider ein JSlider.

Abbildung 19: OutputPanel

GraphPanel: Das *GraphPanel* stellt die Schrittantworten der im *GraphDisplayPanel* ausgewählten Regler-Typen dar. Die Achsen werden dabei automatisch skaliert und falls im *GraphDisplayPanel* Anpassungen vorgenommen wurden, werden die einzelnen Kurven ein- oder ausgeblendet. Für das Anzeigen des Graphen wird ein XYPlot der Klasse *JFreeChart* verwendet, der im BorderLayout platziert wurde.

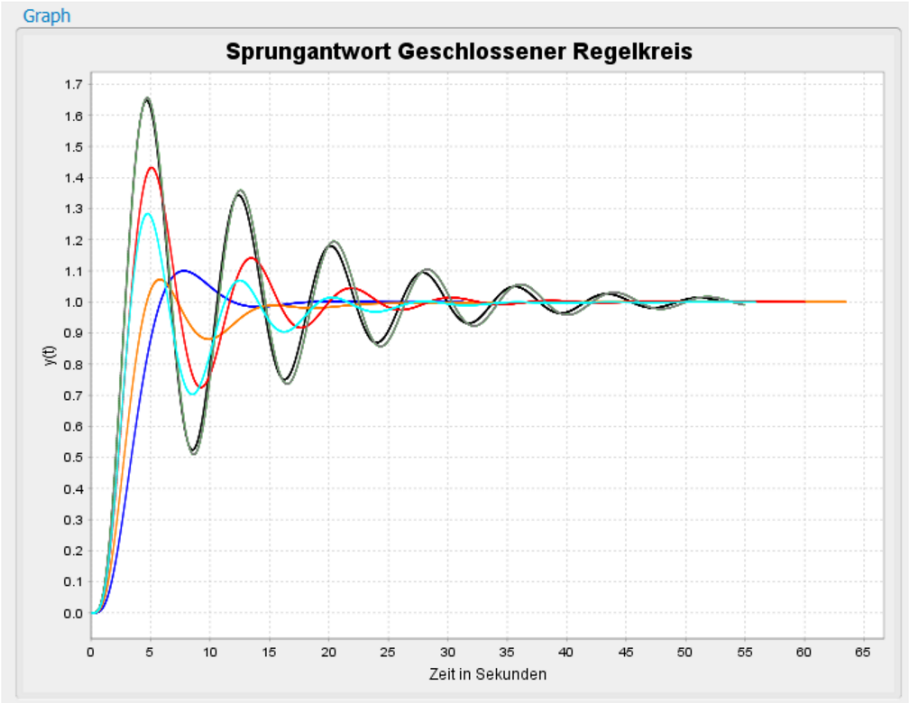

Abbildung 20: GraphPanel

GraphDisplayPanel und GraphSettingPanel: Im *GraphDisplayPanel* können mit Hilfe der Checkboxen die Regler ausgewählt werden, welche im *GraphPanel* angezeigt werden sollen (1). Rechts daneben wurde das *GraphSettingPanel* platziert. Dieses zusätzliche Panel verfügt über drei Buttons mit den folgenden Funktionen:

- (2) Zur original Ansicht wechseln (Zoom auf 100%).
- (3) Alle Kurven ein-/ausblenden.
- (4) Alle Faustformeln ein-/ausblenden.

Alle Checkboxen sind JCheckBoxen der Swing-Klassen und die drei Button JButtons mit selbst erstellten Icons.

Beim *GraphDisplayPanel* wurde der *WrapLayoutManager* verwendet. Dies ist kein standard Layoutmanager und wurde auf einem Web-Blog gefunden (QUELLE https://tips4java.wordpress.com/2008/11/06/wrap-layout/). Dieser Manager ist von der Funktion her vergleichbar mit dem FlowLayout und füllt alle Komponenten von links nach rechts in ein JPanel. Speziell an ihm ist jedoch, dass er beim Verändern der Fenstergrösse den Umbruch der Zeilen richtig handhabt, auch wenn das Panel in einem GridBagLayout platziert ist.

Ausserdem befindet sich am oberen Rand des Fensters die *MenuBar* mit den beiden Schaltflächen *Datei* und *Hilfe*. Sie wurde direkt ins Frame gesetzt und wurde nicht in der Klasse *View* hinzugefügt.

Das Menü *Datei* besteht aus den folgenden Menüeinträgen:

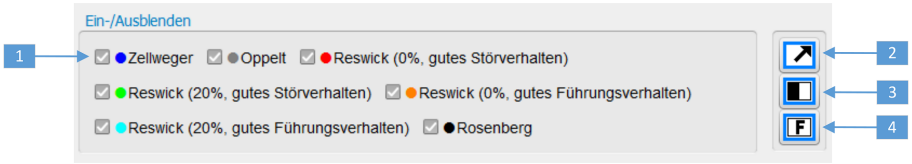

Abbildung 21: GraphDisplayPanel

- *Zur Mini-Version wechseln*: Wechselt von der Gesamtansicht zur Mini-Version. Elemente wie das *GraphPanel*, das *GraphDisplayPanel* und der Trimmer werden dabei ausgeblendet. Der Name dieses Menüeintrages ändert sich von Mini-Version zu Normal-Version.
- *Export als PDF*: Öffnet ein Fenster, mit dem der Pfad und der Dokumentname ausgewählt werden kann. Nach der Bestätigung mit dem Button Speichern wird ein PDF generiert, welches die gesamte Ansicht des Tools exportiert.
- *Schliessen*: Beendet die gesamte Applikation.

Das Menü *Hilfe* besteht aus den folgenden Menüeinträgen:

- *Info*: Es öffnet sich ein Fenster mit Informationen zum Projekt und den Autoren.
- *Hilfreiche Links*: Mit ihnen können Links und PDF-Dokumente geöffnet werden, welche weiterführende Information zur Reglerdimensionierung und der Zellweger-Methode bieten.

Das gesamte Menü wurde mit Komponenten der Swing-Klassen erstellt.

3.2.4 Funktionalität der Benutzeroberfläche

Eingabewerte Für alle Textfelder wurden JDoubleTextFields verwendet. Diese Klasse bekamen wir im Rahmen des Fachinputs durch Herrn Prof. Richard Gut empfohlen. Sie handhabt die unerlaubten Eingaben der Textfelder. Dabei wurden folgende Einstellungen vorgenommen:

1. Die Felder werden nicht gerundet, verfügen also über keinen Formater.
2. Es können nur Zahlen, keine Buchstaben und keine anderen Zeichen eingegeben werden (Ausnahme dabei ist das „e“ für Engineering-Eingaben).
3. Die Fehlermeldung direkt im Textfeld wurde deaktiviert. Fehler werden in diesem Tool auf einer extra Zeile dargestellt.

Fenstermanagement
Wenn das Tool gestartet wird, öffnet sich das Programm in der Normalansicht (alle Panels werden angezeigt) mit der kleinstmöglichen Fenstergrösse. Dazu wird das Frame so gepackt, dass alle Komponenten mit ihrer Minimalgrösse platziert werden. Um auch bei unterschiedlichen Bildschirmauflösungen das Tool verwenden zu können, wurden dafür keine absoluten Pixelwerte verwendet.
Wenn die Fenstergrösse durch den Anwender verändert wird, hat dies einen unterschiedlichen Einfluss auf die einzelnen Panels. Das *OutputPanel* wird nur in der Vertikalen und das *GraphDisplayPanel* und die *MenuBar* nur in der Horizontalen verändert. Weiter wird das *GraphPanel* in beide Richtungen gestreckt oder gestaucht. Das *InputPanel* und das *GraphSettingPanel* bleiben dabei unverändert. Zur Veranschaulichung wurde dieses Verhalten in der folgenden Grafik dargestellt. Die roten Pfeile zeigen dabei, in welche Richtungen sich die Panels verändern können.

Mini-Version
Durch eine Schaltfläche in der *MenuBar* kann zur Mini-Version gewechselt werden. Diese reduzierte Ansicht verzichtet auf den Graphen und die dazugehörigen Panels und wurde speziell für fortgeschrittene Anwender implementiert. Die folgende Grafik zeigt die Mini-Version des Tools.

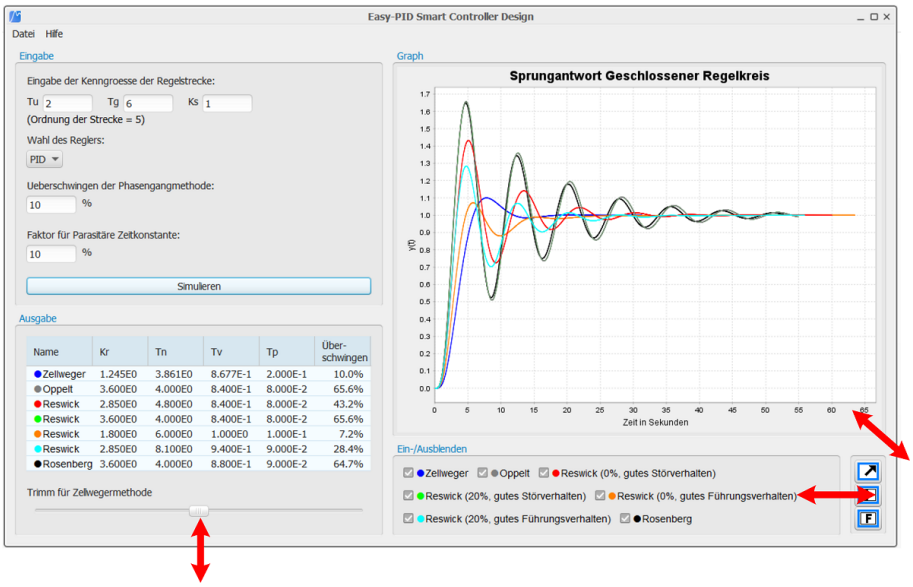

Abbildung 22: Verändern der Fenstergrösse

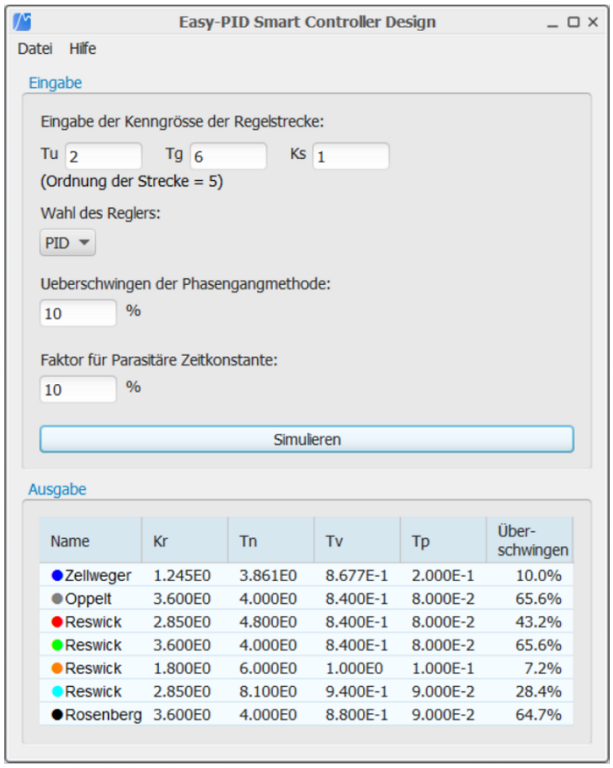

Abbildung 23: Ansicht der Mini-Version

3.3 Controller

Der *Controller* ist sehr schlank gehalten und ist ein Teil des Model-View-Controller-Pattern. Der Controller hat dabei mehrere Aufgaben:

- Beim Drücken des Simulationsschaltfläche im *InputPanel* werden die entsprechenden Methoden des Models aufgerufen, um die neuen Regler mit den neuen Eingabeparametern zu berechnen, die alten Simulationen zu löschen und neue Simulationen zu starten.
- Wenn der Trimm-Slider im *OutputPanel* verändert wird, so wird die entsprechende Methode im Model aufgerufen.
- Wenn der Anwender eine Checkbox im *GraphDisplayPanel* anklickt, so wird die entsprechende Methode im Model aufgerufen, um eine Kurve ein- oder auszublenden.

4 Validierung

Im folgenden Abschnitt wird die Validierung beschrieben. Die Validierung dient zum Auffinden und Beheben von Fehlern, um so eine korrekte Funktion des Programms garantieren zu können. Zum einen wurden die Berechnungen und Simulationen von Matlab überprüft, zum anderen die Software, aufgeteilt in die Teilbereiche Benutzeroberfläche und Berechnungen.

4.1 Matlab-Programmierung

Die Berechnung der Regler wurde zuerst mittels m-Files in Matlab implementiert. Zur Überprüfung der von Matlab gelieferten Werte wurden diese mit den Werten vom Fachcoach Peter Niklaus verglichen.

4.2 Matlab-Simulation

Mittels Simulink wurde der geschlossene Regelkreis simuliert.

4.3 Benutzeroberfläche

Um die einfache Handhabung der Software zu überprüfen wurde die Software im ganzen Projektteam sowie von mehreren unabhängigen Benutzern getestet. Die so gefundenen Unklarheiten oder Probleme wurden behoben.

4.4 Berechnungen Software

Die Plausibilität der Schrittantworten konnte relativ einfach optisch mit den Graphen überprüft werden, da das Verhalten der einzelnen Regler bekannt ist. Zur genauen Überprüfung der einzelnen Klassen wurden sämtliche Klassen des Models mittels JUnit mit den Ergebnissen von Matlab verglichen.

5 Schlussfolgerung

Easy-PID konnte durchaus erfolgreich abgeschlossen werden. Das Tool enthält dabei sämtliche Anforderungen, welche im Pflichtenheft als Sollziele festgelegt wurden. Des weiteren wurden einige Wunschziele, namentlich die Miniversion, die Simulation der geschlossenen Regelstrecke und der Export der Berechnung als PDF als Zusatzfunktionen im Programm implementiert. Eine Nachjustiermöglichkeit für die Reglerparameter wurde dabei bewusst nicht umgesetzt, jedoch kann mittels einem Schieberegler die Phasengangmethode optimiert werden.

Der Benutzer kann die Parameter der Regelstrecke in der grafischen Benutzeroberfläche eingeben, woraufhin „Easy-PID“ die Reglerparameter mittels verschiedener Methoden berechnet und die entsprechenden Sprungantworten grafisch darstellt. Die einzelnen Methoden können dabei auch ausgeblendet werden. Für die Phasengangmethode existiert ausserdem eine Trimm-Regler, mit welchem die Phasengangmethode angepasst werden kann. Mit diesen Funktionen ermöglicht es Easy-PID dem Benutzer, den idealen Regler zu dimensionieren.

Während der Umsetzung sind immer wieder Probleme aufgetreten, die jedoch allesamt gelöst werden konnten. So konnte beispielsweise der erste Entwurf des Klassendiagramms nicht übernommen werden, da das Konzept dafür noch nicht fertig erstellt war. So musste im späteren Projektverlauf ein neues Klassendiagramm mit sämtlichen Konzeptüberlegungen aufgestellt werden. Ein weiteres Problem war die Umsetzung von Matlab-Code in Java. Viele Funktionen, welche in Matlab bereits vorimplementiert sind, mussten für das Tool neu programmiert werden. Auch haben die vielen kleinen Unterschiede zwischen den beiden Programmiersprachen immer wieder zu Problemen geführt.

Am Ende konnte das Projekt jedoch termingerecht und korrekt funktionierend abgeliefert werden. Trotzdem gibt es viele Features, welche in zukünftigen Projekten noch realisiert werden könnten. Eine Möglichkeit wäre das Importieren und Exportieren von Projekten. Dank dem Model-View-Controller Entwurfsmuster kann Easy-PID auch problemlos für eine Mobile-App adaptiert werden.

A Anhang Elektrotechnik

A.1 Herleitung der Berechnung von β für die Zellweger Methode

Für den offenen Regelkreis Go gilt:

$$\phi_{RE} = PhaseRegler$$
$$\phi_S = PhaseStrecke$$
$$\phi_O = PhaseOffenerRegelkreis$$
$$\phi_{RE}(w_{PID}) + \phi_S(w_{PID}) = \phi_O(w_{PID})$$

Für die Steigung (Ableitung) gilt:

$$\frac{d\phi_{RE}(w_{PID})}{dw} + \frac{d\phi_S(w_{PID})}{dw} = \frac{d\phi_O(w_{PID})}{dw} \tag{2}$$

Der offene Regelkreis berechnet sich wie folgt:

$$Go = \frac{1}{s \cdot T0 \cdot (1 + s \cdot T1)}$$

Die Phase von O berechnet sich wie folgt:

$$Tk = \frac{1}{w_{PID}}$$

$$\phi_O(w_{PID}) = -\frac{\pi}{2} - \arctan\left(w \cdot \frac{1}{w_{PID}}\right) = -\frac{pi}{2} - \arctan\left(w \cdot Tk\right)$$

ϕ_O wird nach w abgeleitet und für Tk wird $\frac{1}{w_{PID}}$ eingesetzt. Da in w_{PID} abgeleitet wurde wird w gleich w_{PID} gesetzt:

$$\frac{d\phi_O(w_{PID})}{dw} = -\frac{Tk}{1 + w^2 \cdot Tk^2} = -\frac{Tk}{1 + {w_{PID}}^2 \cdot \left(\frac{1}{w_{PID}}\right)^2} = -\frac{1}{2}Tk = -\frac{1}{2 \cdot w_{PID}} \tag{3}$$

Aus der Ableitung der Phase des offenen Regelkreises (3) und der Phasengleichung (2) des offenen Regelkreises folgt:

$$\phi_{RE}(w_{PID}) + \phi_S(w_{PID}) = -\frac{1}{2 \cdot w_{PID}}$$

Mit w_{PID} multipliziert

$$w_{PID} \cdot \frac{d\phi_{RE}(w_{PID})}{dw} + w_{PID} \cdot \frac{d\phi_S(w_{PID})}{dw} = -0.5$$

Wie man einfach erkennt folgt daraus:

$$w_{PID} \cdot \frac{d\phi_{RE}(w_{PID})}{dw} = \frac{2 \cdot \beta}{1 + \beta^2}$$

Eingesetzt in die Phasengleichung:

$$\frac{2 \cdot \beta}{1 + \beta^2} + w_{PID} \cdot \frac{d\phi_S(w_{PID})}{dw} = -0.5$$

Für $\phi_S(w_{PID})$ wird eingesetzt:

$$\frac{2 \cdot \beta}{1+\beta^2} + w_{PID} \cdot (-\arctan(w_{PID} \cdot T1) - \arctan(w_{PID} \cdot T2) - \arctan(w_{PID} \cdot Tz)) = -0.5$$

Als Summe geschrieben:

$$\frac{2 \cdot \beta}{1+\beta^2} - w_{PID} \cdot \sum_{m=1}^{n} \arctan(w_{PID} \cdot T_m) = -0.5$$

$\phi_S(w_{PID})$ abgeleitet:

$$\frac{2 \cdot \beta}{1+\beta^2} - w_{PID} \cdot \left(\frac{T_1}{1+(w_{PID} \cdot T_2)^2} + \frac{T_2}{1+(w_{PID} \cdot T_2)^2} + \frac{T_m}{1+(w_{PID} \cdot T_m)^2} \right) = -0.5$$

Als Summe geschrieben:

$$\frac{2 \cdot \beta}{1+\beta^2} - w_{PID} \cdot \sum_{m=1}^{n} \frac{T_m}{1+(w_{PID} \cdot T_m)^2} = -0.5$$

Nun wird so umgeformt, dass alle β auf einer Seite stehen:

$$\frac{2 \cdot \beta}{1+\beta^2} = w_{PID} \cdot \sum_{m=1}^{n} \frac{T_m}{1+(w_{PID} \cdot T_m)^2} - 0.5$$

Die rechte Seite wird mit Z substituiert:

$$Z = w_{PID} \cdot \sum_{m=1}^{n} \frac{T_m}{1+(w_{PID} \cdot T_m)^2} - 0.5$$

$$Z = \frac{2 \cdot \beta}{1+\beta^2}$$

Nach β aufgelöst:

$$\beta = \frac{1}{Z} - \sqrt{\frac{1}{Z^2} - 1}$$

A.2 Herleitung der Bodekonformen und Reglerkonformen Darstellung

Regler können in verschieden Darstellungen abgebildet werden. Unterschieden werden hier die reglerkonforme und die bodekonforme Darstellung, wobei oftmals standartmässig die reglerkonforme Darstellung gewählt wird. Folgend sind beide Darstellungen von PI- und PID-Regler aufgeführt. Beim PID-Regler wird zudem die Beziehung zwischen den beiden Formen hergeleitet.

A.2.1 PI-Regler

Der PI-Regler hat die Parameter Kr und Tn.

Reglerkonform:

$$G_R(s) = K_R \left(1 + \frac{1}{s \cdot Tn}\right) \tag{4}$$

Bodekonform:

$$G_R(s) = K_R \left(\frac{1 + s \cdot Tn}{s \cdot Tn}\right) \tag{5}$$

A.2.2 PID-Regler

Reglerkonform:

Der PID-Regler hat in der reglerkonformen Darstellung die Parameter Kr, Tn, Tv und Tp.

$$G_R(s) = K_R \left(1 + \frac{1}{s \cdot Tn} + \frac{s \cdot Tv}{1 + s \cdot Tp}\right) \tag{6}$$

Umformung:

$$\begin{aligned} G_R(s) &= K_R \left(\frac{s \cdot Tn(1 + s \cdot Tp) + (1 + s \cdot Tp)(s \cdot Tn \cdot s \cdot Tv)}{s \cdot Tn(1 + s \cdot Tp)}\right) \\ &= K_R \left(\frac{s^2 \cdot Tn \cdot Tp + s^2 \cdot Tn \cdot Tv + s \cdot Tp + s \cdot Tn + 1}{s \cdot Tn(1 + s \cdot Tp)}\right) \\ &= K_R \left(\frac{s^2(Tn \cdot Tp + Tn \cdot Tv) + s(Tp + Tn) + 1}{s \cdot Tn(1 + s \cdot Tp)}\right) \\ &= K_R \left(\frac{s^2 \cdot Tn(Tp + Tv) + s(Tp + Tn) + 1}{s \cdot Tn(1 + s \cdot Tp)}\right) \end{aligned} \tag{7}$$

Bodekonform:

Der PID-Regler hat in der bodekonformen Darstellung die Parameter Krk, Tn, Tv und Tp.

$$G_R(s) = K_{RK} \left(\frac{(1 + s \cdot Tnk)(1 + s \cdot Tvk)}{s \cdot Tnk(1 + s \cdot Tp)}\right) \tag{8}$$

Umformung:

$$G_R(s) = K_{RK} \left(\frac{s^2(Tnk \cdot Tvk) + s(Tnk + Tnk) + 1}{s \cdot Tnk(1 + s \cdot Tp)}\right) \tag{9}$$

Koeffizienten-Vergleich:
Damit ein Koeffizienten-Vergleich zwischen der reglerkonformen und der bodekonformen Darstellung des PID-Reglers durchgeführt werden kann, müssen die beiden Formen gleichgesetzt werden (links reglerkonform, rechts bodekonform):

$$K_R\left(\frac{s^2 \cdot Tn(Tp+Tv)+s(Tp+Tn)+1}{s \cdot Tn(1+s \cdot Tp)}\right) = K_{RK}\left(\frac{s^2(Tnk \cdot Tvk)+s(Tnk+Tnk)+1}{s \cdot Tnk(1+s \cdot Tp)}\right)$$

s wird auf beiden Seiten im Nenner gekürzt:

$$K_R\left(\frac{s^2 \cdot Tn(Tp+Tv)+s(Tp+Tn)+1}{Tn(1+s \cdot Tp)}\right) = K_{RK}\left(\frac{s^2(Tnk \cdot Tvk)+s(Tnk+Tnk)+1}{Tnk(1+s \cdot Tp)}\right)$$

Lässt man die Frequenz w gegen Null gehen, strebt auch s gegen Null und die Terme lassen sich wie folgt vereinfachen:

$$K_R\left(\frac{1}{Tn}\right) = K_{RK}\left(\frac{1}{Tnk}\right)$$

An der oben stehenden Umformung erkennt man, dass man einen Koeffizientenvergleich machen kann. Folgend werden zuerst die Koeffizienten mit s, danach die s^2 und am Ende die Koeffizienten ohne s verglichen und nach den Parameter der reglerkonformen Darstellung umgeformt.

2. Koeffizient

$$\begin{aligned} s(Tn+Tp) &= s(Tnk+Tvk) \\ Tn+Tp &= Tnk+Tvk \\ Tn &= Tnk+Tvk-Tp \end{aligned} \tag{10}$$

Damit ist Tn bestimmt.

1. Koeffizient:

$$\begin{aligned} s^2(Tn \cdot Tp+Tn \cdot Tv) &= s^2(Tnk \cdot Tvk) \\ (Tn \cdot Tp+Tn \cdot Tv) &= (Tnk \cdot Tvk) \\ Tn(Tv+Tp) &= Tnk \cdot Tvk \end{aligned} \tag{11}$$

Obenstehende Gleichung wird nun nach Tv umgeformt:

$$\begin{aligned} Tv+Tp &= \frac{Tnk \cdot Tvk}{Tn} \\ Tv &= \frac{Tnk \cdot Tvk}{Tn}-Tp \end{aligned}$$

Die Variable Tn wird in der folgenden Gleichung durch die Beziehung von Formel 10 ersetzt:

$$Tv = \frac{Tnk \cdot Tvk}{Tnk+Tvk-Tp}-Tp \tag{12}$$

Damit ist Tv bestimmt. Kr wird wie folgt bestimmt:

3. Koeffizient:

$$\frac{Kr}{s \cdot Tn(1+s \cdot Tp)} = \frac{Krk}{s \cdot Tnk(1+s \cdot Tp)}$$

s und $(1 + s \cdot Tp)$ werden gekürzt und Tn ersetzt:

$$\frac{Kr}{Tnk + Tvk - Tp} = \frac{Krk}{Tnk}$$
$$Kr = \frac{Krk(Tnk + Tvk - Tp)}{Tnk}$$

Das im Verhältnis zu Tvk ca. zehn mal kleinere Tp kann vernachlässigt werden:

$$Kr = \frac{Krk(Tnk + Tvk)}{Tnk}$$
$$Kr = \frac{Krk(1 + Tvk)}{Tnk} \tag{13}$$

B CD-Rom