Aufgabenstellung 3

Ein Viewer für Bode-Diagramme

Autor: Marcin Maślanka
Matrikelnummer: 32107801

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Tabellenverzeichnis

1 Erarbeitungs- /Reflexionsphase

1.1 Github

Pfad zum Projekt:

https://github.com/marcinmaslanka/cpp-repo

1.2 Einführung

Der Quellcode des Bode-Diagramm-Viewer wurde in der Programmiersprache C++ geschrieben. Als integrierte Entwicklungsumgebung diente Visual Studio Code. Die folgenden externen Bibliotheken wurden im Projekt verwendet.

Tabelle 1 Bibliotheken. Eigene Darstellung.

```
#include<iostream>
#include<fstream>
#include<cmath>
#include<vector>
#include<complex>
```

Das übergeordnete Zeil beim Schreiben des Programmcodes bestand darin, eine maximale Übersichtlichkeit des Quellcodes sicherzustellen. Das Hauptprogramm *main()* enthält nur die Variablendeklarationen und die Hauptschleife *do-while*. Die drei wichtigsten variablen stehen am Anfang der Deklarationsliste, das sind *startFreq, endFreq* und *numPoints*. Abhängig von der Übertragungsfunktion können diese Variablen bei Bedarf angepasst werden. Die Anzahl der Punkte *numPoints* ist ein wichtiger Parameter bei der Berechnung Amplitudenreserve und Phasenreserve. Die Standardeinstellungen sind in der folgenden Tabelle aufgeführt.

Tabelle 2 Standardeinstellungen. Eigene Darstellung.

```
double startFreq = 0.01;
double endFreq = 100.0;
int numPoints = 1024;
```

Der Kern des *main()*-Programms stellt die *do-while* Schleife dar. Das Beenden der Schleife bendet auch das Programm. Zwischen *main()* und *#include* gibt es vier Unterprogramme: *evaluateTransferFunction, calculateMagnitued, calculatePhase* und *starteGnuplot*. Die ersten drei sind für die Berechnung der Übertragungsfunktion zuständig und der letzte dient als Schnittstelle zu Gnuplot.

1.3 Module des Programms

1.3.1 Hauptmenü

Das Hauptmenü in der Erarbeitungsphase im Vergleich zur Konzeptionsphase wurde leicht verändert. Der Punkt „Plot der Übertragungsfunktion“ wurde in zwei separate Punkte aufgeteilt: „Plot des Amplitudengangs“ und „Plot des Phasengangs“. Auf die Konvertierung in die JPG-Datei wurde verzichtet, stattdessen werden die Diagramme in der PNG-Datei gespeichert. Die Speicherung erfolgt nach Durchführung des Plots der Amplituden bzw. des Phasengangs.

Der Aufbau des Hauptmenüs wurde als *do-while*-Schleife implementiert. Im *do*-Bereich wird das Hauptmenü in der Konsole angezeigt und bleibt dauerhaft in der Konsole sichtbar. Der *while*-Abschnitt des Codes prüft, ob der Ausdruck in den Klammern wahr ist. Wenn ja, dann wird die *do-while*-Schleife immer wieder ausgeführt. Mit der Taster 5 wird der Ausdruck in Klammern zu falsch und das bedeutet Ende der Schleife und das Ende des Programms. Die *switch*-Funktion dient zur Auswahl einer bestimmten Funktionalität aus dem Hauptmenü. Im Programm sind ganze Zahlen von 1 bis 5 erlaubt. Andere Zahlen werden über das Default Anweisung aufgegriffen und geben dem Benutzer eine Meldung: „Falsche Eingabe. Probieren Sie noch einmal“.

Tabelle 3 Menü. Eigene Darstellung.

MENU: [1] – Eingabe der Übertragungsfunktion [2] – Plot der Amplitudengang [3] – Plot der Phasengang [4] – Berechnen der Amplitudenreserve und der Phasenreserve [5] – Beenden Geben Sie die Zahl 1-5 ein:

1.3.2 [1] Eingabe der Übertragungsfunktion

Der Benutzer gibt die Übertragungsfunktion mithilfe des Polynoms ein, genauer gesagt, indem er die Koeffizienten des Polynoms eingibt. Die Koeffizienten für den Zähler und die Koeffizienten für den Nenner werden getrennt eingegeben. Das ist eine Änderung gegenüber der Konzeptionsphase, in der der Benutzer die Übertagungsfunktion in Bode-Normalform eingeben sollte.

Nach der Auswahl der Taster 1 wird der Benutzer nach Koeffizienten des Polynoms gefragt. Im Zähler werden die Koeffizienten des Polynoms zweiten Grades und im Nenner die Koeffizienten des Polynoms dritten Grades eingetragen. Die eingegebenen Zahlen werden jeweils in einem Container namens Vector gespeichert. Der Vector für den Zähler heißt *numeratorCoefficients* und für den Nenner *denominatorCoefficients*. Erlaubt sind ganze Zahlen, Fließkommazahlen, positive und negative Zahlen. Die Eingegebenen Zahlen müssen durch ein Leerzeichen, einen Tabulator oder einer neuen Zeile voneinander getrennt werden. Sollte der Koeffizient auf null gesetzt werden, wird der betroffene Term, zum Beispiel *s^2* oder *s^1*, ebenfalls zu Null gesetzt und daher in weiteren Berechnungen als inaktiv betrachtet. Zum Beispiel, wenn Sie eine Übertragungsfunktion des Tiefpassfilters berechnen möchte. Die Koeffizienten für den Zähler sind wie folgt einzugeben: „0 0 1“ und die Koeffizienten für den Nenner: „0 0 1 1“. Die Zahlen werden in der gleichen Reihenfolge im Vektor des Zählers *numeratorCoefficient = {0, 0, 1}* und im Vektor des Nenners *denominatorCoeffcient = {0, 0, 1, 1}* in das Polynom eingetragen.

Um dies für den Benutzer zu bestätigen, wird die Übertragungsfunktion H(s) in der Konsole angezeigt. Dies wurde mit einer *for*-Schleife für den Zähler und den Nenner implementiert. Bei jeder Iteration der *for*-Schleife wird eine bestimmte *s*^-Variable hinzugefügt. Die einzelnen Terme werden durch ein Zeichen vom Typ-String „+“ voneinander getrennt.

Tabelle 4 Eingabe der Koeffizienten der Übertragungsfunktion. Eigene Darstellung.

```
Enter coefficients for the numerator (a,b,c): 0 0 1
Enter coefficients for the denominator (d,e,f,g): 0 0 1 1
H(s)= 0s^2 + 0s^1 + 1s^0) / (0s^3 + 0s^2 + 1s^1 + 1s^0)
```

1.3.3 [2] [3] Plot der Übertragungsfunktion

Erstellen des Diagramms der Übertragungsfunktion aus der Konzeptionsphase wurde in Plot des Amplitudengangs und in ein Plot des Phasengangs unterteilt. Die Darstellung des Amplitudengangs wird mit der Ziffer 2 im Hauptmenü gestartet. Der Ablauf dieser Funktion ist wie folgt. Zu Beginn werden die Datenpunkte generiert. Anschließend werden die Datenpunkte in einer Textdatei namens *bode_data_magnitude.txt* gespeichert. Im letzten Schritt wird Gnuplot aufgerufen und die Werte aus der Textdatei gelesen und geplottet.

Die Datenpunkte werden in einer *for*-Schleife generiert. Bei jeder Iteration wird ein Frequenzwert berechnet und in einem Vector *freq[i]* sowie in einer variablen *double frequency* zwischengespeichert. Der Grund für die doppelte Speicherung der Frequenzwerte bittet Flexibilität bei dem Umgang mit anderen Funktionen. Die Daten aus dem Vektor werden später

zum Speichern in der Textdatei verwendet und die Frequenzen aus der Variablen *double frequency* werden sofort in derselben *for*-Schleife zur Berechnung der Amplitude verwendet.

Die x-Achse des Amplitudengangs und des Phasengangs ist logarithmisch skaliert. Dazu müssen die Datenpunkte entsprechend aufbereitet werden. Ausgehend von der allgemeinen Gleichung der Potenzfunktion, erhält man den Ausdruck aus der Gleichung 2. Anschließend muss man noch den Exponenten über die Anzahl der Datenpunkte, in diesem Fall (*numPoints-1) normieren*. Dadurch wird der Graph genau in der Mitte zwischen den beiden Grenzfrequenzen *startFreq* und *endFreq* positioniert.

$$x_i = x_0 * r^i \qquad \text{Gl. 1}$$

$$x_i = x_0 * 10^{i * \log(x1/x0)} \qquad \text{Gl. 2}$$

$$freq[i] = startFreq * 10^{(i * \log(endFreq/startFreq)/(numPoints-1))} \qquad \text{Gl. 3}$$

Die zur Berechnung der Amplitude verwendete Funktion heißt *calculateMagnitude*. Im Argument der Funktion werden die Vektoren für Zähler und Nenner übergeben sowie die berechneten Frequenzen aus der variablen *double frequency*. Der Rückgabewert der Funktion wird im Vektor *magnitudeProb[i]* gespeichert.

Die Übertragung der Werte von einem eindimensionalen Vektor in eine Textdatei erfolgt über eine *for*-Schleife. Bei der ersten Iteration wird der Wert aus dem Vektor *freq[i]* mit der Adresse 0 gelesen und in der ersten Zeile der Textdatei positioniert, anschließend wird ein Tabulator eingefügt, damit die Spalten klar unterschieden werden können. Anschließend werden die Daten aus dem zweiten Vektor *magnituedProb[i]* mit Adresse 0 kopiert und die Zeile endet mit dem Befehl */n*. Dadurch werden zwei Spalten erstellt, die erste Spalte für die Frequenz und die zweite Spalte für die Amplitude. Die Anzahl der Zeilen wird mit der Variablen *numPoints* deklariert und begrenzt die maximale Anzahl von Iterationen. Nach der letzten Iteration wird die Datei geschlossen und die Konsole zeigt dem Benutzer eine Meldung an, dass die Datei erfolgreich gespeichert wurde. Eventuelle Fehler, die beim Schreiben in die Datei auftreten können, werden mit der *else statement* aufgegriffen.

Der Phasengang wird analog berechnet und geplottet. Zur Berechnung der Phase wird eine Funktion *calculatePhase* gestartet.

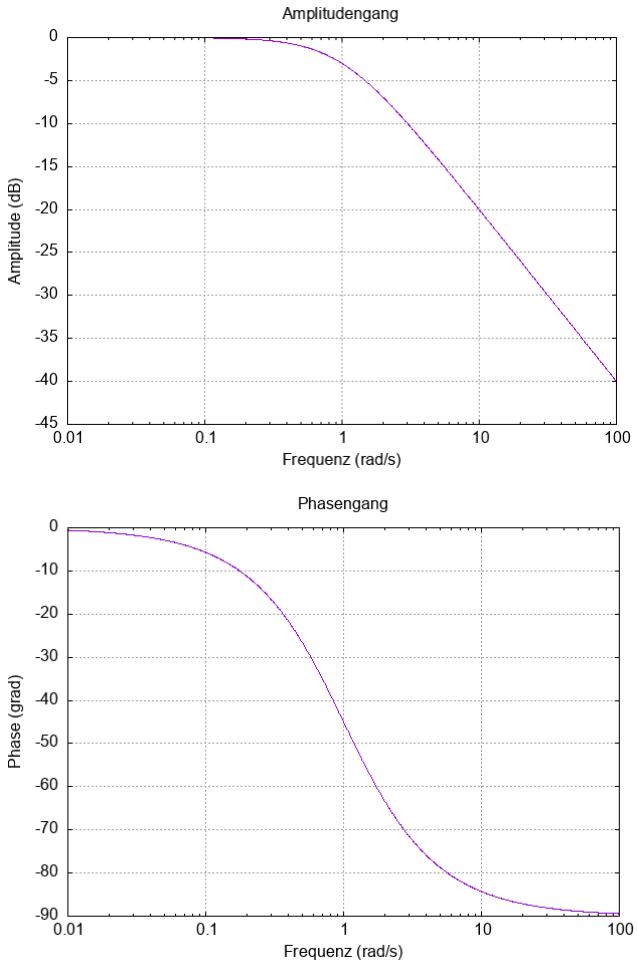


Abbildung 1 Plot der Übertragungsfunktion des Tiefpassfilters. Eigene Darstellung.

1.3.4 [4] Berechnung der Amplitudenreserve

Die Berechnung der Amplitudenreserve erfolgt mit einer *for*-Schleife. Die Anzahl der Iterationen wird mit der Variablen *numPoints* festgelegt. Je mehr Datenpunkte ausgewählt werden, desto näher kommt man an dem gewünschten Grenzpunkt. Bei der Anzahl von 1024 Datenpunkten wird der Wert der Phase kleiner oder gleich -179.1 Grad gesucht. Ist der Wert erreicht, werden Amplitude und Frequenz in die Hilfsvariablen, *gainAtPhaseCrossover* und *frequencyAtPhaseCrossover* kopiert. Der Wert von *gainMargin* wird dann mit -1 multipliziert, sodass das Endergebnis positiv ist. Die Ergebnisse werden in Form von Text in der Konsole ausgegeben.

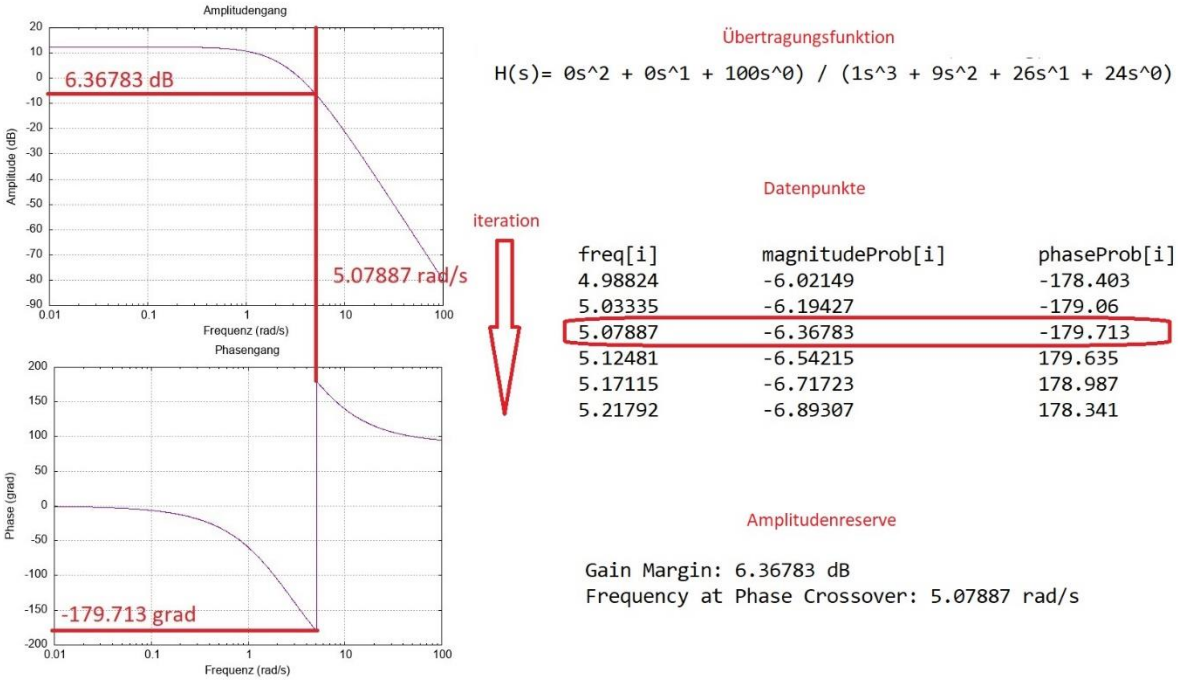
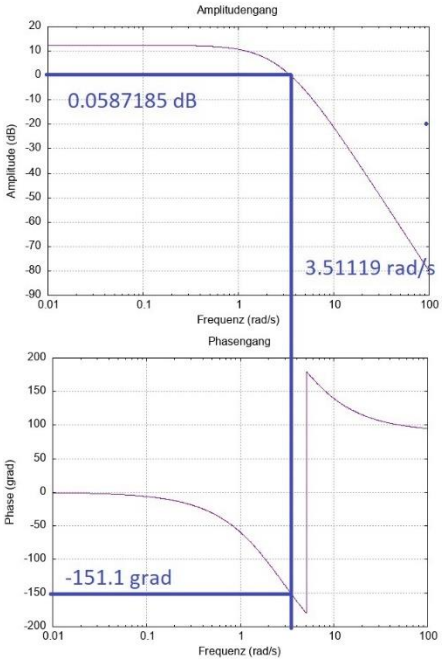


Abbildung 2 Amplitudenreserve. Eigene Darstellung.

1.3.5 [4] Berechnung der Phasenreserve

Zur Berechnung der Phasenreserve wird im Diagramm der Punkt gesucht, an dem der Amplitudengang die 0-dB-Linie schneidet. Aufgrund der begrenzten Anzahl an Iterationspunkten muss der gesuchte Schnittpunkt auf 0.1 gesetzt werden. Die Phasenreserve ist definiert als der Abstand zu -180 Grad des Phasengangs. Dann wird zu der Variable *phaseAtCrossover* +180 addiert, um den Phasenreservewert positiv zu erhalten.



Übertragungsfunktion

H(s)= 0s^2 + 0s^1 + 100s^0) / (1s^3 + 9s^2 + 26s^1 + 24s^0)

Datenpunkte

iteration

freq[i]	magnitudeProb[i]	phaseProb[i]
3.44853	0.333375	-149.633
3.47972	0.196523	-150.367
3.51119	0.0587185	-151.1
3.54295	-0.0800361	-151.831
3.57499	-0.219738	-152.562
3.60732	-0.360385	-153.292
3.63994	-0.501973	-154.02

Phasenreserve

Phase Margin: 28.9005 degrees
Frequency at Gain Crossover: 3.51119 rad/s

Abbildung 3 Phasenreserve. Eigene Darstellung.

1.4 Interaktion zwischen Modulen

1.4.1 Gnuplot ohne #include<gnuplot-iostream.h>

Die Möglichkeit, Gnuplot zu verwenden, ohne externe Bibliotheken anbinden zu müssen ist wie folgt. Zusätzlich zur Quellcodedatei und der exe-Datei sind zwei weitere Konfigurationsdateien erforderlich. Einmal für die Darstellung des Amplitudengangs und einmal für die Darstellung des Phasengangs. Die Konfigurationsdateien heißen amplitude.gp und phase.gp und enthalten Befehle aus dem Gnuplot-Programm. Zuerst der Logarithmische Skalierung der x-Achse, der Titel des Diagramms, die Achsenbeschriftungen und der Typ der erstellten Plotdatei. Der Plot wird als PNG-Datei mit den Titeln *amplitude.png* und *phase.png* gespeichert.

Das Verfahren ist dasselbe, als würde der Benutzer die Behle in die Kommandozeile der Gnuplot-Konsole selbst eingeben. Die Funktion *startenGnuplot* trägt den String *gnuplot amplitude.gp* in die Kommandozeile der Konsole ein, wodurch der Gnuplot gestartet wird und die Befehle aus der Konfigurationsdatei *amplitude.gp*, nacheinander automatisch in die Gnuplot-Konsole eingegeben und ausgeführt werden.

Wichtig ist, dass vor dem Start der Bode-Diagramm-Viewer, Gnuplot installiert wird und der Pfad zu Gnuplot in der Systemvariablen eingetragen werden muss.

Tabelle 5 Konfigurationsdatei amplitude.gp. Eigene Darstellung.

```
set logscale x
set grid
set title 'Amplitudengang'
set xlabel 'Frequenz (rad/s)'
set ylabel 'Amplitude (dB)'
set terminal png
set output 'amplitude.png'
unset key
plot 'bode_data_magnitude.txt' using 1:2 linecolor 1 with lines
```

1.4.2 Kalkulieren der Amplituden- /Phasengang

Die Berechnung der Amplitude bzw. der Phase erfolgt in zwei Schritten. Zunächst wird das Unterprogramm *calculateMagnitude* aufgerufen und dort eine Substitution durchgeführt. Der Parameter *s* entspricht der imaginären Zahl *i,* multipliziert mit der Variablen *double frequency*. Im zweiten Schritt wird das Unterprogramm *evaluateTransferFunction* mit den folgenden Argumenten, Vektoren der Koeffizienten und der imaginären Zahl *s* aufgerufen. Der Rückgabewert des Unterprograms evaluateTransferFunction ist die komplexe Zahl. Diese Zahl wird in einer Komplexen Variablen zwischengespeichert und für die weitere Berechnung zur Verfügung gestellt. Im Unterprogram *calculateMagnitude* wird die Amplitude berechnet und in Dezibel zurückgegeben. Die Berechnung der Phase erfolgt im Unterprogram *calculatePhase* nach dem gleichen Prinzip und das Endergebnis wird im Grad zurückgegeben. Beide Werte werden entsprechend im Vektor *magnitudeProb[i]* für die Amplitude und *phaseProb[i]* für die Phase gespeichert. Anschließend erfolgt die nächste Iteration, sodass die Vektoren *magnitudeProb[i]* und *phaseProb[i]* mit Werten gefüllt werden, bis die Anzahl der Datenpunkte den Wert *numPoints* erreicht.