



Hochschule Konstanz
Technik, Wirtschaft und Gestaltung

Signale, Systeme und Sensoren

Fourieranalyse und Akustik

T. Schoch, L. Stratmann

Konstanz, 12. Mai 2019

Zusammenfassung (Abstract)

Thema:	Fourieranalyse und Akustik	
Autoren:	T. Schoch	tobias.schoch@htwg-konstanz.de
	L. Stratmann	luca.stratmann@htwg-konstanz.de
Betreuer:	Prof. Dr. Matthias O. Franz Jürgen Keppler Christoph Kaiser	mfranz@htwg-konstanz.de juergen.keppler@htwg-konstanz.de christoph.kaiser@htwg-konstanz.de

In dem dritten Versuch der Versuchsreihe werden wir die Fourieranalyse auf akustische Signale und Systeme in Form von Musik durch eine Mundharmonika und Rückkopplung anwenden. Die Signale werden auf einem Oszilloskop angezeigt.

In den beiden Teilen des Versuchs werden wir akustische Signale mittels der Python Bibliothek TekTDS2000 aufnehmen und abspeichern. Im Anschluss erfolgt mittels Python die Auswertung der Messdaten.

Dabei werden die Techniken der Fouriertransformation und des Bode Diagramms angewendet.

Inhaltsverzeichnis

Abbildungsverzeichnis	III
Tabellenverzeichnis	V
Listingverzeichnis	VI
1 Einleitung	1
2 Versuch 1	2
2.1 Fragestellung, Messprinzip, Aufbau, Messmittel	2
2.2 Messwerte	5
2.3 Auswertung	6
2.4 Interpretation	9
3 Versuch 2	10
3.1 Fragestellung, Messprinzip, Aufbau, Messmittel	10
3.2 Messwerte	13
3.3 Auswertung	15
3.4 Interpretation	20
Anhang	21
A.1 Quellcode	21
A.1.1 Quellcode Versuch 1.1	21
A.1.2 Quellcode Versuch 1.2	22
A.1.3 Quellcode Versuch 1.3	22
A.1.4 Quellcode Versuch 2.1	24
A.1.5 Quellcode Versuch 2.2	26
A.2 Messergebnisse	30

Abbildungsverzeichnis

2.1	Versuchsaufbau Teil 1	3
2.2	Signal einer Mundharmonika	5
2.3	Das Amplitudenspektrum von der Fourieranalyse	7
3.1	Signal einer Mundharmonika	11
3.2	Einlesen des Signals durch das Oszilloskop und in die .csv Datei	13
3.2a	Die Ausgabe des Oszilloskopes	13
3.2b	Die Auswertung der .csv Datei durch Python	13
3.3	Vergleich zwischen dem kleinen und dem großen Lautsprecher bei einer Frequenz von 200Hz	14
3.3a	Die Spannung bei dem kleinen Lautsprecher und einer Frequenz von 200Hz	14
3.3b	Die Spannung bei dem großen Lautsprecher und einer Frequenz von 200Hz	14
3.4	Amplitude mit Dateinummer und zugehöriger Frequenz	15
3.4a	Amplitude mit der zugehörigen Dateinummer	15
3.4b	Amplitude mit der zugehörigen Frequenz	15
3.5	Beispiel einer Phasenverschiebung	16
3.6	Phasenverschiebung in einer .csv Datei	16
3.7	Phasengang mit der zugehörigen Frequenz	17
3.7a	Phasengang mit der zugehörigen Dateinummer	17
3.7b	Phasengang mit der zugehörigen Dateinummer	17
3.8	Bode Diagramm für die Frequenz	18
3.8a	Bode-Diagramm für die Amplitude mit Dateinummer	18
3.8b	Bode-Diagramm für die Amplitude mit Frequenz	18
3.9	Bode-Diagramm für den Phasengang	19
3.9a	Bode-Diagramm für den Phasengang mit Dateinummer	19

3.9b	Bode-Diagramm für den Phasengang mit Frequenz	19
4.10	Messergebnisse für Task 2	30

Tabellenverzeichnis

2.1 Zu berechnende Werte	6
2.2 Amplitude und Frequenz	8

Listingverzeichnis

4.1	Das Bild des Oszilloskopes einlesen und abspeichern	21
4.2	.csv Datei einlesen und in einem Plot graphisch wiedergeben	22
4.3	Fouriertransformation anwenden und ein Amplitudenspektrum ausgeben sowie Berechnung einiger Werte	22
4.4	Lautsprecherdaten graphisch ausgeben und einige Werte berechnen	24
4.5	Amplitude Phasenverschiebung und Bode-Diagramm berechnen	26

1

Einleitung

In dem dritten Versuch der Versuchsreihe werden wir die Fourieranalyse auf akustische Signale und Systeme in Form von Musik durch eine Mundharmonika und Rückkopplung anwenden. Die Signale werden auf einem Oszilloskop angezeigt.

In den beiden Teilen des Versuchs werden wir akustische Signale mittels der Python Bibliothek TekTDS2000 aufnehmen und abspeichern. Im Anschluss erfolgt mittels Python die Auswertung der Messdaten.

Im ersten Versuch wird die Fouriertransformation und einige Komplementaritäten behandelt bezüglich eines Tons einer Mundharmonika den wir aufnehmen. So müssen wir unter Anderem die Grundperiode, die Grundfrequenz, die Signaldauer, die Abtastfrequenz, die Signallänge und das Abtastintervall für die Aufnahme berechnen.

Dabei werden wir die Fouriertransformation und das Amplitudenspektrum berechnen und graphisch wiedergegeben. Im zweiten Versuch müssen wir mit einem Mikrofon und zwei unterschiedlichen Lautsprechern eine Rückkopplung verursachen und mit verschiedenen Frequenzen wieder Aufnahmen machen.

Die einzelnen Aufnahmen werden ausgewertet und daraus die Phasenverschiebung und die Amplitude berechnet. Auf die Ergebnisse der Berechnung wenden wir das Bode-Diagramm an um die Leistung der Lautsprecher zu analysieren.

Das und vieles mehr erwartet euch auf den folgenden Seiten unseres Protokolles.

2

Versuch 1

2.1 Fragestellung, Messprinzip, Aufbau, Messmittel

Im ersten Versuch der Versuchsreihe "Fourieranalyse und Akustik" werden wir die Fourieranalyse auf akustische Signale und Systeme anwenden. Dazu werden wir mittels einer Mundharmonika in ein Mikrofon blasen, dass wiederum an ein Oszilloskop angeschlossen ist und die Spannungen anzeigt.

Dabei ist es wichtig zu beachten, dass auf dem Oszilloskop mehrere Perioden abgebildet werden. Das Triggerlevel soll so im Oszilloskop eingestellt werden, dass das Signal nur bei genügend hoher Amplitude angezeigt wird.

Die Triggerung sollte dabei auch im Single Sequence Modus eingestellt werden um statische Daten zu erhalten die auch dem tatsächlichen Spannungswert entsprechen. Nachdem dies erledigt ist, haben wir das Python Skript task1.1.py geschrieben, um das Signal aus dem Oszilloskop bei der Einstellung der Triggerung SSingle Sequenzäuszulesen.

Dies wird mittels der Toolbox TekTDS2000 von M. Miller in eine .csv Datei gespeichert welche zum Beispiel mit Excel geöffnet werden kann.

So sieht der Versuchsaufbau des ersten Versuches aus. Dabei wird das Mikrofon an den Channel 1 des Oszilloskopes angeschlossen.

Nach der Kalibrierung und der Einstellung der beiden Sinuskurven haben wir mit dem Pythonskript das ausgegebene Signal des Oszilloskopes in eine .csv Datei eingelesen.

Danach hatten wir die Aufgabe folgende Werte zu berechnen.

- Grundperiode (in ms)
- Grundfrequenz (in Hz)
- Signaldauer (in s)
- Abtastfrequenz (in Hz)
- Signallänge M (Anzahl der Abtastzeitpunkte)
- Abtastintervall Δt (in s)



Abbildung 2.1: Versuchsaufbau Teil 1

Im Anschluss mussten wir mithilfe der Funktion `numpy.fft.fft()` die Fouriertransformation des Signals berechnen. Daraus sollten wir das Amplitudenspektrum bestimmen und grafisch darstellen.

Dabei ist zu beachten, dass die x-Achse mit der Frequenz nicht in Hertz sondern in der Anzahl Schwingungen innerhalb der gesamten Signaldauer definiert ist. Die Frequenz f in Hertz lässt sich jedoch folgendermaße berechnen.

$$f = \frac{n}{M * \Delta t}$$

Als letzten Teil der ersten Aufgabe sollen wir die Grundfrequenz im Spektrum berechnen und damit die Frequenz in Hertz identifizieren. Folgende Materialien wurden benötigt:

- Oszilloskop
- Mikrofon
- Mundharmonika
- Signalkabel
- Python auf einem Computer

2.2 Messwerte

Auf dem Bild ist das Signal abgebildet, dass wir mittels einer Mundharmonika in das Mikrofon geblasen haben.

Durch die Toolbox TekTDS2000 können wir das Signal nun in eine .csv Datei umwandeln und abspeichern.

Diese Datei haben wir mittels Numpy und der Funktion `np.loadtxt` ausgelesen. Die Ergebnisse wurden in die 2 verschiedene Funktionen `x` und `y` geschrieben.

Danach wurden diese mittels matplotlib graphisch dargestellt.

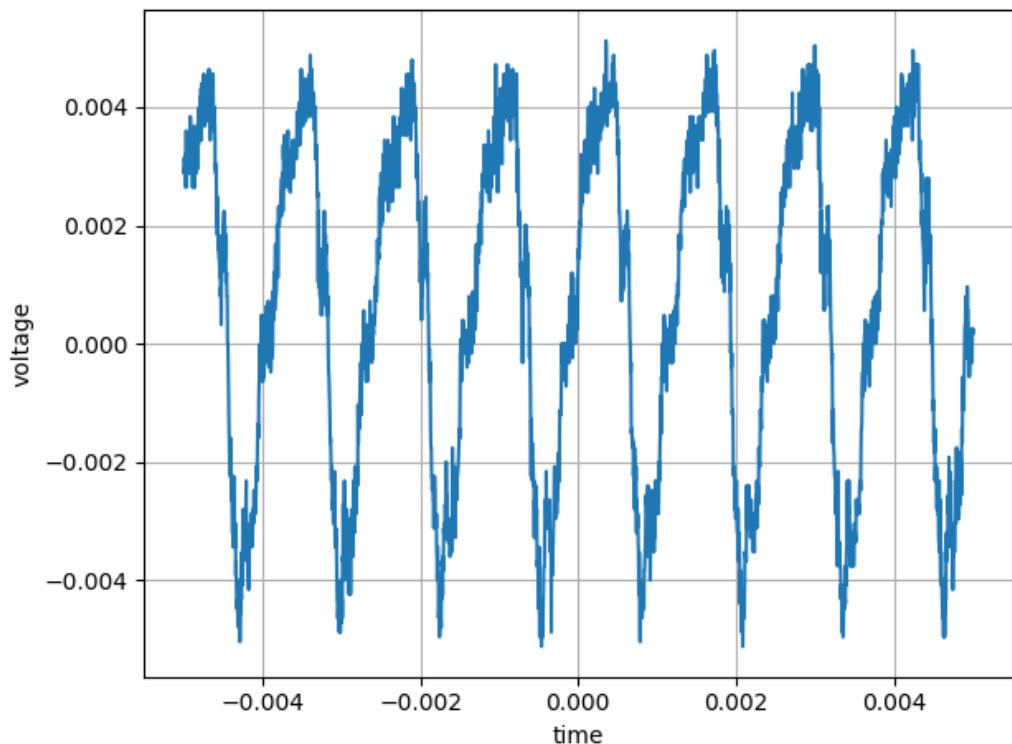


Abbildung 2.2: Signal einer Mundharmonika

2.3 Auswertung

Auf dem Spannungsverlauf sieht man das aufgenommene Signal der Mundharmonika. Durch einen Pythonfunktion die wir aus der Toolbox TekTDS2000 erhalten haben, konnten wir den Spannungsverlauf aus dem Oszilloskop einlesen und in eine .csv Datei schreiben.

Dieses Format wird auch häufig für Excel Dateien verwendet. Durch das Drücken des Knopfes Single Sequence haben wir auf dem Oszilloskop ein Standbild erhalten.

Diese Datei haben wir mittels Numpy und der Funktion `np.loadtxt` ausgelesen. Die Ergebnisse wurden in die 2 verschiedene Funktionen `x` und `y` geschrieben.

Danach wurden diese mittels matplotlib graphisch dargestellt.

Die Grundperiode und die Grundfrequenz haben wir uns mittels der Toolbox TekTDS2000 durch `getFreq(1)` und `getPeriod(1)` ausgeben lassen. Die Signallänge M haben wir durch `len(file.readlines())` erfahren.

Durch die absolute Addition von den beiden Maxima- und Minimawerten erhalten wir gerundet die Signaldauer. Wenn man nun die Signaldauer dividiert durch die Signallänge erhält man das Abtastintervall Δt mit dem Wert $4\mu s$.

Mit dem Abtastintervall erhält man auch die Abtastfrequenz. Diese entspricht 250000 Hertz.

Im folgenden sind die Ergebnisse der Berechnungen aufzufinden:

Plot	Wert
Grundperiode (in ms)	1.275ms
Grundfrequenz (in Hertz)	784.31Hz
Signaldauer	0,01s
Abtastfrequenz (in Kilohertz)	250KHz
Signallänge M (Anzahl der Abtastzeitpunkte)	2500
Abtastintervall Δt (in μs)	$4\mu s$

Tabelle 2.1: Zu berechnende Werte

Das Amplitudenspektrum erhält man indem man die Spannung absolut fouriertransformiert mit der Numpy Funktion `np.fft.fft()` und als Amplitude auf der y Achse ausgibt.

Doch bisher wird die x-Achse in der Einheit *Anzahl Schwingungen innerhalb der gesamten Signaldauer* und nicht in Hertz angegeben.

Um dies jedoch zu tun müssen wir die Frequenz in f folgendermaßen berechnen:

$$f = \frac{n}{M * \Delta t}$$

Die jeweilige Schwingung n wird geteilt durch die Signallänge und das Abtastintervall. Dadurch erhält man folgenden Graphen, welcher bis zu dem von uns festgelegten Frequenzwert von 20000 geplottet wird.

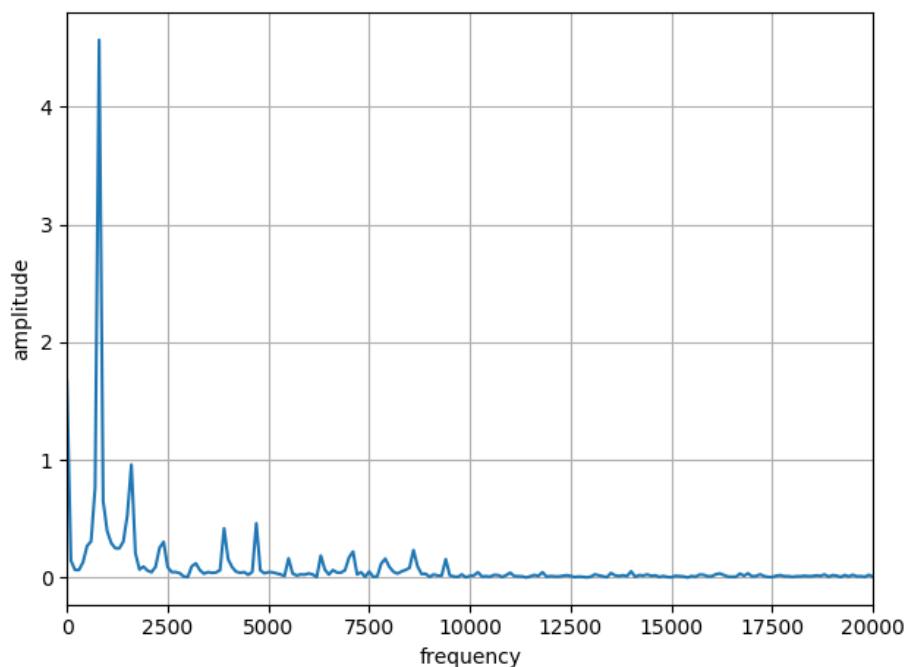


Abbildung 2.3: Das Amplitudenspektrum von der Fourieranalyse

Mittels der Numpy Funktion `np.max(spektrum)` können wir den maximalen Amplitudenwert abfragen. Mittels einer for-Schleife fragen wir nun den Wert bis zum 1250sten Signal ab der das Maxima hat.

Dadurch erhalten wir die Frequenz des maximalen Amplitudenwerts. Wir fragen genau bis zur Hälfte der Werte ab, da der Graph sich spiegelt und es so 2 Maxima gibt. Da wir nur den ersten Wert wollen müssen wir dies festlegen.

Im folgenden sind die Ergebnisse der Berechnungen aufzufinden:

Plot	Wert
Maximaler Amplitudenwert	4.5675
Frequenz des Maxima	800Hz

Tabelle 2.2: Amplitude und Frequenz

2.4 Interpretation

Da das Signal der Mundharmonika schön gleichmäßig ist, kann man feststellen, dass der Ton sehr gleichmäßig gespielt wurde.

Durch die Beschaffenheit des Funktionsweise und die laute Gegebenheiten zum Zeitpunkt der Messung, kann man gut beobachten, dass die Amplitude nach so gut wie jeder periodischen Wiederholung zunimmt und damit die Spannung auch höher wird. Im großen Ganzen jedoch gibt es keine Unstimmigkeiten.

Für die Tabelle 2.1 kann man folgende Schlussfolgerungen ziehen:

- Die Signaldauer ist komplementär mit der Signallänge und dem Abtastintervall
- Das Abtastintervall ist komplementär zur Abtastfrequenz
- Frequenz und Zeit sind komplementär zueinander

Bei der Abbildung 2.3 ist eine schöne Maximalamplitude zu erkennen, die sich von den anderen Amplituden durch ihre Höhe absetzt. Dieser maximale Amplitudenausschlag mit der Höhe 4.5675V ist der Ton.

Der Ton den wir produziert haben liegt also demnach bei 800Hz. Da wir bis auf eine Öffnung die Anderen zugeklebt haben, wissen wir dass wir einen Ton mit der Frequenz von 800Hz gespielt haben.

Die anderen kleinen Harmonischen sind die Obertöne. Wir wissen durch das Amplitudenspektrum auch, dass wir keinen Klang gespielt haben und die Mundharmonika richtig zugeklebt haben.

Wenn man die Fouriertransformierte für alle 2500 Abtastungen graphisch darstellt, dann stellt sich eine y-Achsenspiegelung dar. Da wir jedoch keine Achsenspiegelung wollen sondern ledigliche bis Frequenzen von 20000Hz kürzen wir die x-Achse um nur die wichtigen Werte zu erlangen.

3

Versuch 2

3.1 Fragestellung, Messprinzip, Aufbau, Messmittel

In dem zweiten Versuch war es die Aufgabe für jeden der beiden Lautsprecher die Amplitude und die Phasenverschiebung des akustischen Ausgangssignales zu ermitteln, indem man sowohl die Eingangs- als auch das Mikrofonsignal auf dem Oszilloskop darstellt.

Eines der beiden Diagramme haben wir abgespeichert welches in der Abbildung 3.1 einsehbar ist. Zudem berechnen wir jeden der 17 verschiedenen Frequenzen für jeweils den großen und den kleinen Lautsprecher die Amplitude und die Phasenverschiebung. Die 17 verschiedenen Frequenzen sind folgende:

100Hz, 200Hz, 300Hz, 400Hz, 500Hz, 700Hz, 850Hz, 1000Hz, 1200Hz, 1500Hz, 1700Hz, 2000Hz, 3000Hz, 4000Hz, 5000Hz, 6000Hz, 10000Hz

Diese geben wir im Laufe des Versuches in einer Tabelle aus. Wichtig ist es hierbei bei beiden Lautsprechern den selben Abstand zu haben. Zudem sollten beide Kanäle auf AC Coupling gestellt sein, da wir den Gleichanteil nicht benötigen.

Wie bei dem ersten Versuch verwenden wir wieder den Single Sequence Mode um ein Standbild des Signals zu erhalten um dieses besser zu capturen.

Nach der Aufnahme des Signals ist es wichtig für die nächste Frequenz wieder den Single Sequence Mode zu deaktivieren um ein neues Bild zu erhalten.

Das Eingangssignal soll konstant auf 1.5V eingestellt sein. Mit jeder Veränderung der Frequenz muss die Amplitude wieder angepasst werden. Danach sollen wir die Daten der Amplitude und der Phasenverschiebung in Arrays transferieren und graphisch darstellen.

Für beide Lautsprecher werden wir mit matplotlib ein Bode-Diagramm darstellen, indem wir beide Diagramme mit der Funktion semilogx() halblogarithmisch darstellen und den Phasenwinkel dazu berechnen mit der Formel:

$$\text{Amplitude} = 1 / \text{Amplitude}$$

$$\text{Amplitude} = 20\log_{10} * \text{Amplitude}$$

$$\text{Phasenwinkel} = -\Delta t * f * 360$$

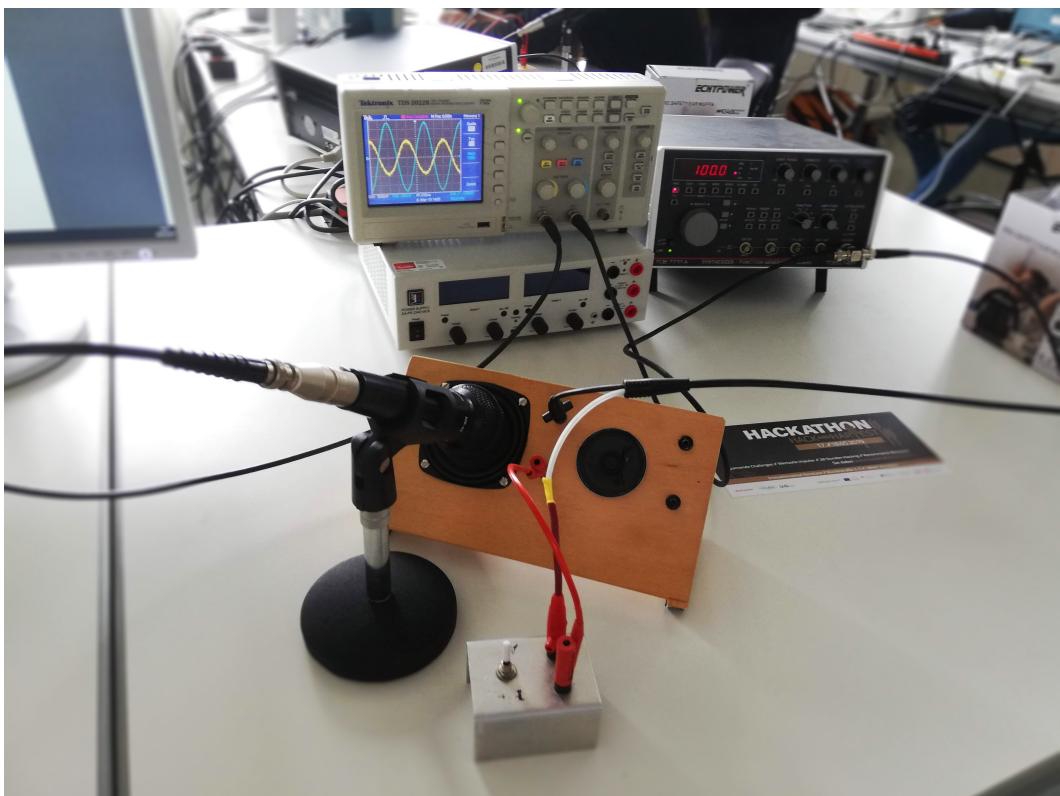


Abbildung 3.1: Signal einer Mundharmonika

In dem zweiten Versuch schließen wir das Mikrofon erneut an den Channel 1 des Oszilloskopes. Der Lautsprecher wird mit dem Pluspol an den An- und Ausschalter angeschlossen. Der Schalter wiederrum wird wiederrum mit dem Negativkabel verbunden und in den Verstärker geleitet.

Den Schalter benötigen wir, um den nervtötenden Ton direkt wieder zu eliminieren, sobald man eine Single Sequence besitzt. Der Verstärker wiederrum geht in den Channel 2 des Oszilloskopes. Der Verstärker wird wiederrum an den Output des Oszilloskopes angehängt.

Der Abstand vom Mikrofon zum Lautsprecher soll bei beiden Lautsprechern gleich sein.

Folgende Materialien wurden benötigt:

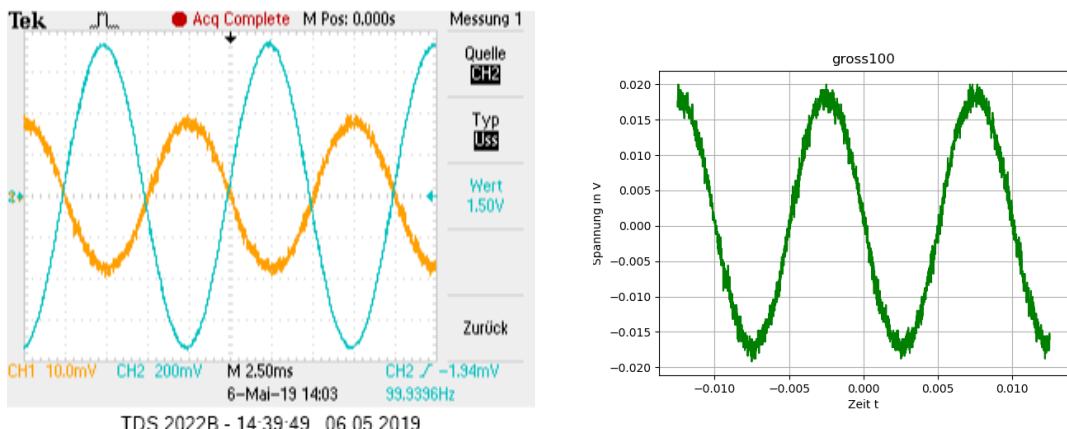
- Oszilloskop
- Verstärker
- 2 verschiedene Lautsprecher
- Schalter
- Mikrofon
- Mundharmonika
- Signalkabel Schwarz
- Signalkabel Rot
- Signalkabel, dass Plus mit Minus verbindet und beide weiterleitet
- Adapter für Verstärker
- Python auf einem Computer

3.2 Messwerte

Auf der linken Seite ist die Single Sequence des Oszilloskopes bei einer Frequenz von 100Hz. Dabei stellt das gelbe Signal den Input beziehungsweise das Mikrofon dar, während das blaue Signal vom Channel 2 stammt und der Output für den Lautsprecher ist.

Bei beiden Signalen sehen wir eine gleichmäßige Sinuskurve. Die beiden Signale wurden in .csv Dateien gespeichert.

Der Input von Channel 1 wurde dabei nochmals von uns eingelesen in einem Pythonskript und wurde daraufhin graphisch ausgegeben.



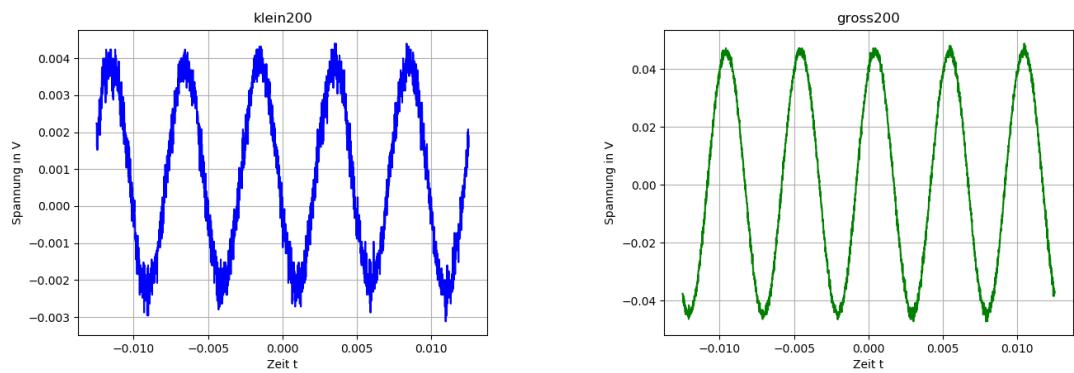
(b) Die Auswertung der .csv Datei durch Python

(a) Die Ausgabe des Oszilloskopes

Abbildung 3.2: Einlesen des Signals durch das Oszilloskop und in die .csv Datei

Wir haben für jede Frequenz und sowohl für Input als auch Output jeweils eine graphische Darstellung. Da diese jedoch den Rahmen sprengen würde und zudem das Dokument viel zu unübersichtlich wäre, reduziere ich die Darstellung auf die Frequenz von 200Hz.

In der Abbildung 3.3 haben wir die abgebildete Rückkopplung für den kleinen und den großen Lautsprecher bei jeweils 200Hz. Links ist der kleine Lautsprecher abgebildet und rechts der große Lautsprecher.



- (a) Die Spannung bei dem kleinen Lautsprecher
und einer Frequenz von 200Hz (b) Die Spannung bei dem großen Lautsprecher
und einer Frequenz von 200Hz

Abbildung 3.3: Vergleich zwischen dem kleinen und dem großen Lautsprecher bei einer Frequenz von 200Hz

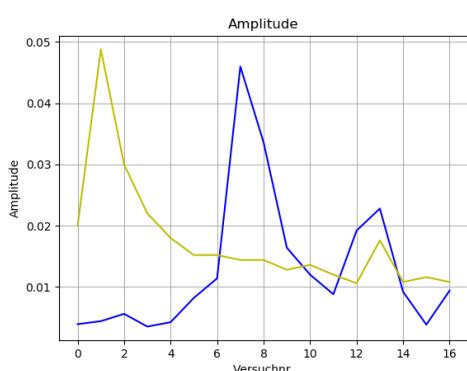
3.3 Auswertung

Auf dem Spannungsverlauf beziehungsweise den Spannungsverläufen sieht man die Spannung die bei einer Rückkopplung entsteht bei bestimmten Frequenzen. Diese wurden auf dem Oszilloskop angezeigt und durch ein von uns mithilfe der Toolbox TekTDS2000 erstelltes Python Skript in eine .csv Datei geschrieben.

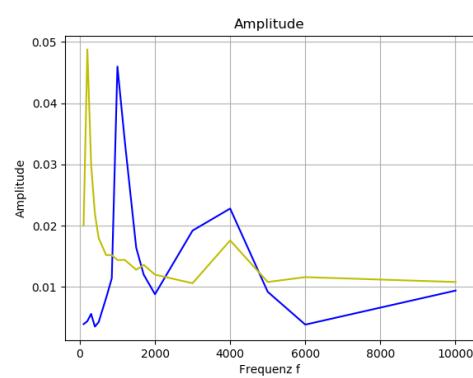
Dabei haben wir jeweils eine .csv Datei erstellt für Channel 1 (Mikrofon als Input) und für Channel 2 (Lautsprecher als Output). Die .csv Dateien werden von uns in einem Python Skript wieder eingelesen und die Werte in Arrays gespeichert.

Aus den Daten stellen wir wiederrum die Amplitude und den Phasengang graphisch dar.

Mit der Numpy Funktion `np.max(np.abs())` erhalten wir den maximalen absoluten Wert des Signals beziehungsweise die Amplitude. Dies machen wir mit sämtlichen 17 unterschiedlichen Frequenzen. Das Signal das gelb abgebildet ist, beschreibt die Amplitude des großen Lautsprechers, während das blaue Signal die Amplitude des kleinen Lautsprechers darstellt. Daraus erhalten wir die folgenden graphischen Darstellungen.



(a) Amplitude mit der zugehörigen Dateinummer



(b) Amplitude mit der zugehörigen Frequenz

Abbildung 3.4: Amplitude mit Dateinummmer und zugehöriger Frequenz

Die Phasenverschiebung zu berechnen ist etwas komplizierter als erst gedacht. Zuerst schauen wir uns den Plot an der uns für die jeweiligen Frequenzen ausgegeben wird.

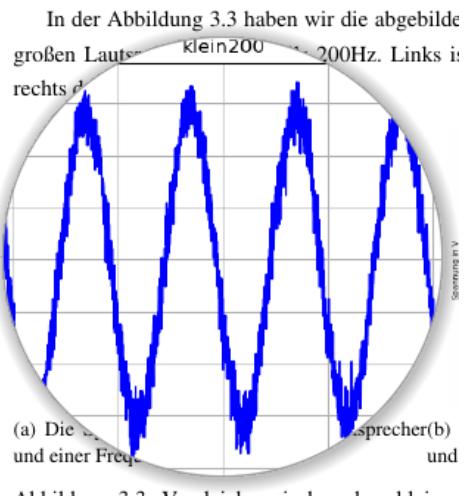


Abbildung 3.5: Beispiel einer Phasenverschiebung

Der Achsenschnittpunkt befindet sich in der Mitte. Dort sollte also auch der Anfang der Sinuskurve sein. Da der Abstand zum nächsten Nullpunkt auf der rechten Seite des Achsenschnittpunktes ist, müssen wir nun in der .csv Tabelle den nächsten Nullpunkt finden. Der Achsenschnittpunkt liegt bei der Zeile 1250. Deshalb müssen wir nun tiefer suchen nach einem Nullpunkt.

1434	0.00183999999999415,-0.0004
1435	0.001849999999994146,-0.0009600000000000001
1436	0.001859999999994142,-0.0010400000000000001
1437	0.001869999999994138,0
1438	0.001879999999994134,-8e-05
1439	0.00188999999999413,-0.00024000000000000003
1440	0.001899999999994126,-0.0004
1441	0.001909999999994122,8e-05

Abbildung 3.6: Phasenverschiebung in einer .csv Datei

Hier haben wir den Nullpunkt in der Zeile 1437. Deshalb rechnen wir nun $1437 - 1250 = 187$ Signallänge

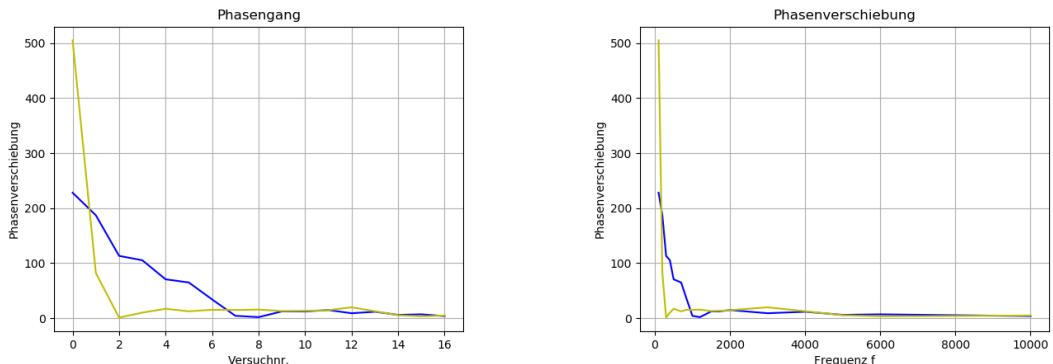
Die Singallänge von 87 multiplizieren wir nun mit dem Abtastintervall. Das Abtastintervall erhalten wir durch die Subtraktion zweier benachbarter Werte, die daraufhin auf 5 Nachkommastellen gerundet werden.

$$187 \text{ Signallänge} * 1 \mu\text{s} = 187 \mu\text{s}$$

Nun haben wir die Phasenverschiebung berechnet. An dem Beispiel für den kleinen Lautsprecher mit einer Frequenz von 200Hz beträgt die Phasenverschiebung also $187\mu\text{s}$.

Nachdem wir sämtliche Phasenverschiebungen berechnet haben, geben wir diese graphisch in einem Plot wieder.

Das gelbe Signal ist erneut der große Lautsprecher, während das blaue Signal der kleine Lautsprecher ist. Im linken Bild ist der Phasengang in Abhängigkeit zur Dateinummer und im rechten Bild ist der Phasengang in der Abhängigkeit zur Frequenz.



(a) Phasengang mit der zugehörigen Dateinummer (b) Phasengang mit der zugehörigen Dateinummer

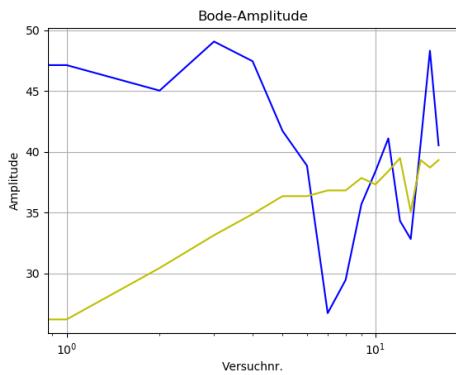
Abbildung 3.7: Phasengang mit der zugehörigen Frequenz

Das Bode Diagramm für die Amplitude wird folgendermaßen berechnet:

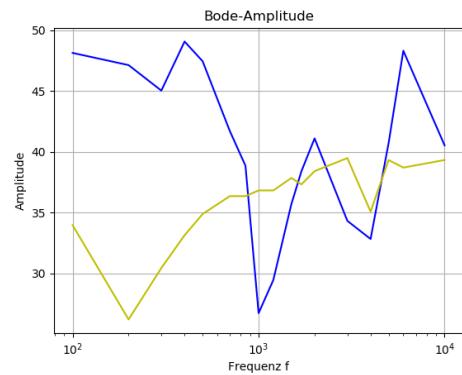
$$amplitude = 1 / amplitude$$

$$amplitude = 20 * \log_{10} amplitude$$

Nun muss man nur noch bei sämtlichen plots die Numpy Funktion `np.semilogx()` anwenden um die Diagramme halblogarithmisch dazustellen.



(a) Bode-Diagramm für die Amplitude mit Datei-
nummer



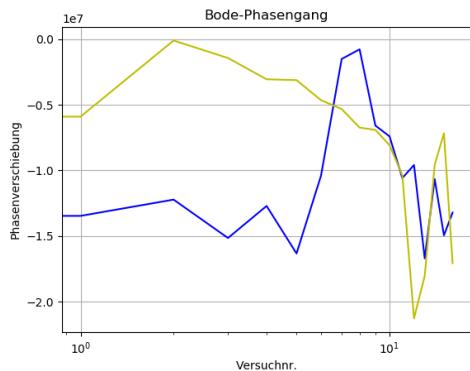
(b) Bode-Diagramm für die Amplitude mit Fre-
quenz

Abbildung 3.8: Bode Diagramm für die Frequenz

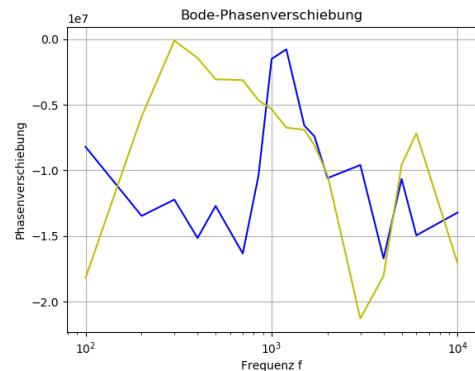
Das Bode Diagramm für die Phasenverschiebung wird folgendermaßen berechnet:

$$\text{Phasenwinkel} = -\Delta t * f * 360$$

Nun muss man nur noch bei sämtlichen plots die Numpy Funktion `np.semilogx()` anwenden um die Diagramme halblogarithmisch dazustellen.



(a) Bode-Diagramm für den Phasengang mit Da-
teinummer



(b) Bode-Diagramm für den Phasengang mit Fre-
quenz

Abbildung 3.9: Bode-Diagramm für den Phasengang

3.4 Interpretation

In der Abbildung 3.3 kann man die gut die Unterschiede zwischen dem kleinen und dem großen Lautsprecher erkennen. Der kleine Lautsprecher hat ein sehr ungenaues und unfeines Signal.

Der große Lautsprecher hingegen hat klare Linien und Kurven. Das lässt darauf deuten, dass der große Lautsprecher mehr Watt hat und bei den meisten Frequenzen eine bessere Leistung besitzt.

Damit lässt sich beweisen, dass der große Lautsprecher qualitativ besser ist.

In dem linken Schaubild in der Abbildung 3.7 divergiert die Phasenverschiebung mit zunehmender Frequenz immer mehr Richtung 0.

Da das Mikrofon Frequenzen von 70Hz bis 13KHz aufnehmen kann, stellen die von uns gemessenen Frequenzen keine Probleme dar in unseren Abbildungen.

Während das Bode-Daigramm bei der Amplitude den Betrag der Übetragsfunktion liefern kann, so kann es bei der Phasenverschiebung den Winkel der Übetragsfunktion berechnen.

Der große Lautsprecher hat eine größere Leistung bei den meisten der Frequenzen. Vor allem bei den niedrigen und basshaltigen Frequenzen wird dieser Unterschied sichtbar. Bei den mittleren bis hohen Frequenzen hat oft der kleine Lautsprecher eine bessere Leistung.

So kann man durch das Bode-Phasen Diagramm gut erkennen, dass der große Lautsprecher vor allem bei basslastigen und niedrigen Frequenzen sehr gut funktioniert. Der kleine Lautsprecher hingegen ist gut bei mittleren Frequenzen und hohen Frequenzen im Vergleich zu den großen Lautsprechern.

Anhang

A.1 Quellcode

A.1.1 Quellcode Versuch 1.1

```
1 from TekTDS2000 import *
2
3 scope = TekTDS2000()
4
5 # Einlesen vom Channel 1 und vom Channel 2
6 scope.saveCsv(filename='versuch3/kleinerLautsprecher/100.csv', ch=1)
7 scope.saveCsv(filename='100_2.csv', ch=2)
8
9 frequency = scope.getFreq(1)
10 period = scope.getPeriod(1)
11
12 print("Frequenz", frequency)
13 print("Periode", period)
```

Listing 4.1: Das Bild des Oszilloskopes einlesen und abspeichern

A.1.2 Quellcode Versuch 1.2

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 # Einlesen der .csv Datei
5 x, y = np.loadtxt('data/eins.csv', delimiter=',', unpack=True)
6
7 # Darstellung des Signals unserer Mundharmonikaaufnahme
8 plt.plot(x * 1000, y * 1000, 'b')
9 plt.ylabel('Spannung in mV')
10 plt.xlabel('Zeit in ms')
11 plt.grid(True)
12 plt.show()
```

Listing 4.2: .csv Datei einlesen und in einem Plot graphisch wiedergeben

A.1.3 Quellcode Versuch 1.3

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 # Einlesen der .csv Datei
5 data = np.genfromtxt('data/eins.csv', delimiter=',', skip_header=0, skip_footer=0)
6 # 2 Zeilen aus der linken Spalte in time schreiben
7 time = data[:, 0]
8 # Der zweite Wert wird absolut minus den ersten absoluten wert gerechnet um später den Wert
9 difference = np.abs(time[1] - time[0])
10 # Die zweite Spalte der .csv Datei wird Fouriertransformiert
11 fourier = np.fft.fft(data[:, 1])
12 # Die Fouriertransformierte Frequenz wird absolutiert, so dass kein negativer Wert mehr vorzufinden ist
13 spektrum = np.abs(fourier)
14 # Formel um die Anzahl der Schwingungen in die Freqeuenz umzurechnen – f = n / (M * t)
15 freq = range(0, 2500, 1) / (difference * 2500)
16
17 # Darstellung des Amplitudenspektrums
18 plt.plot(freq, spektrum)
19 plt.grid()
20 plt.xlabel('Frequency in Hz')
21 plt.ylabel('Amplitude in V')
22 plt.xlim(0, 20000)
23 plt.show()
```

```

24
25 # Einlesen der Signallänge
26 file = open("data/eins.csv")
27 signallaenge = len(file.readlines())
28 # Sekundenumwandlung so wird 0,000001s zu 1s
29 sek = 1000000
30 # Das Abtastintervall in s anzeigen und runden
31 abtastintervall = round((difference * sek), 2)
32
33 # For-Schleife um den passenden Frequenzwert zu erlangen der zu der maximalen Amplitude gehört
34 for x in range(0, 1250):
35     if round(spektrum[x], 4) == 4.5675:
36         frequency = freq[x]
37
38 # Berechnung der größten Amplitude
39 print("Grundperiode: 0.001275 s", )
40 print("Grundfrequenz: 784.31 Hz", )
41 print()
42 print("Signaldauer: ", (abtastintervall * signallaenge) / sek, " s")
43 print("Abtastfrequenz: ", 1 / abtastintervall * sek, " Hz")
44 print("Signallänge M: ", signallaenge)
45 print("Abtastintervall t:", abtastintervall, " s")
46 print()
47 print("Maximalster Amplitudenausschlag", round(frequency, 1))
48 print("Frequenz mit der größten Amplitude", np.max(spektrum))

```

Listing 4.3: Fouriertransformation anwenden und ein Amplitudenspektrum ausgeben sowie Berechnung einiger Werte

A.1.4 Quellcode Versuch 2.1

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 # Array um die einzelnen .csv Dateien einzulesen des kleinen Lautsprechers
5 klein = ["klein100", "klein200", "klein300", "klein400", "klein500", "klein700", "klein850", "klein1000",
6         "klein1200", "klein1500", "klein1700", "klein2000", "klein3000", "klein4000", "klein5000",
7         "klein6000", "klein10000"]
8 # Array um die einzelnen .csv Dateien einzulesen des großen Lautsprechers
9 gross = ["gross100", "gross200", "gross300", "gross400", "gross500", "gross700", "gross850", "gross1000",
10        "gross1200", "gross1500", "gross1700", "gross2000", "gross3000", "gross4000", "gross5000",
11        "gross6000", "gross10000"]
12
13 #for-Schleife für die 17 unterschiedlichen .csv Dateien
14 for a in range(0, 17):
15     # Einlesen der kleinen Dateien – dabei wird die linke Spalte als x und die rechte als y definiert
16     x, y = np.loadtxt('data/' + klein[a] + '.csv', delimiter=',', unpack=True)
17     # Einlesen der großen Dateien – dabei wird die linke Spalte als u und die rechte als v definiert
18     u, v = np.loadtxt('data/' + gross[a] + '.csv', delimiter=',', unpack=True)
19
20     # Die einzelnen .csv Dateien des kleinen Lautsprechers werden in jeweils einem Plot dargestellt
21     plt.plot(x, y, 'b')
22     plt.title(klein[a])
23     plt.ylabel('Spannung in V')
24     plt.xlabel('Zeit t in s')
25     plt.grid(True)
26     plt.savefig(str(klein[a]) + '.png')
27     plt.show()
28
29     # Die einzelnen .csv Dateien des großen Lautsprechers werden in jeweils einem Plot dargestellt
30     plt.plot(u, v, 'g')
31     plt.title(gross[a])
32     plt.ylabel('Spannung in V')
33     plt.xlabel('Zeit t in s')
34     plt.grid(True)
35     plt.savefig(str(gross[a]) + '.png')
36     plt.show()
37
38     # Die ersten beiden Zeiten für den kleinen Lautsprecher werden in time1 geschrieben
39     time1 = x[:2, ]
40     # Die ersten beiden Zeiten für den großen Lautsprecher werden in time1 geschrieben
```

```

41 time2 = u[:2, ]
42
43 # Die zwei Zeiten des kleinen Lautsprechers werden subtrahiert und multipliziert
44 # für eine mikrosekunden Darstellung
45 # Zudem wird die berechnete Dauer gerundet
46 timing1 = round((time1[1] - time1[0]) * 100000, 5)
47 # Die zwei Zeiten des großen Lautsprechers werden subtrahiert und multipliziert
48 # für eine mikrosekunden Darstellung
49 # Zudem wird die berechnete Dauer gerundet
50 timing2 = round((time2[1] - time2[0]) * 100000, 5)
51
52 # Ausgeben einer Tabelle im LaTeX Format
53 print("\hline")
54 # Der maximale Amplitudenwert des kleinen Lautsprechers für die jeweilige Frequenz
55 print("Amplitude:", klein[a], np.max(np.abs(y)))
56 # Der maximale Amplitudenwert des großen Lautsprechers für die jeweilige Frequenz
57 print("Amplitude:", gross[a], np.max(np.abs(v)))
58 print("ms", klein[a], timing1)
59 print("ms", gross[a], timing2)
60
61 print("\hline")

```

Listing 4.4: Lautsprecherdaten graphisch ausgeben und einige Werte berechnen

A.1.5 Quellcode Versuch 2.2

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 # Array um die einzelnen .csv Dateien einzulesen des kleinen Lautsprechers
5 klein = ["klein100", "klein200", "klein300", "klein400", "klein500", "klein700", "klein850", "klein1000",
6         "klein1200", "klein1500", "klein1700", "klein2000", "klein3000", "klein4000",
7         "klein5000", "klein6000", "klein10000"]
8 # Array um die einzelnen .csv Dateien einzulesen des großen Lautsprechers
9 gross = ["gross100", "gross200", "gross300", "gross400", "gross500", "gross700", "gross850", "gross1000",
10        "gross1200", "gross1500", "gross1700", "gross2000", "gross3000", "gross4000",
11        "gross5000", "gross6000", "gross10000"]
12
13 # Die für jeweils den großen und den kleinen Lautsprecher gemessenen Zeiten
14 zeit = [100, 200, 300, 400, 500, 700, 850, 1000, 1200, 1500, 1700, 2000, 3000, 4000, 5000, 6000, 10000]
15 # Die für den kleinen Lautsprecher von Hand berechneten Phasenverschiebungen in s
16 kleinphase = [228, 187, 113.2, 105.2, 70.6, 64.8, 34, 4.2, 1.8, 12.2, 12.1, 14.7, 8.88, 11.60, 5.92, 6.92, 3.67]
17 # Die für den großen Lautsprecher von Hand berechneten Phasenverschiebungen in s
18 grossphase = [505, 82, 1, 10, 17, 12.4, 15.2, 14.8, 15.6, 12.8, 13.2, 14.4, 19.7, 12.52, 5.32, 3.32, 4.74]
19 # Die einzelnen Abtastintervalle für die einzelnen Frequenzen des großen Lautsprechers
20 grossintervall = [1, 1, 0.4, 0.4, 0.4, 0.4, 0.4, 0.2, 0.2, 0.1, 0.1, 0.1, 0.04, 0.04, 0.04, 0.02, 0.01]
21 # Die einzelnen Abtastintervalle für die einzelnen Frequenzen des kleinen Lautsprechers
22 kleinintervall = [1, 1, 1, 1, 1, 0.4, 0.4, 0.4, 0.4, 0.2, 0.2, 0.2, 0.1, 0.04, 0.04, 0.04, 0.02]
23 # Ein Vektor in dem später für den kleinen Lautsprecher die Amplituden gespeichert werden
24 kleinamp = np.zeros(17)
25 # Ein Vektor in dem später für den großen Lautsprecher die Amplituden gespeichert werden
26 grossamp = np.zeros(17)
27
28 # For Schleife von 0–16 um die einzelnen .csv Dateien einzulesen und auszuwerten
29 for a in range(0, 17):
30     # Einlesen der kleinen Dateien – dabei wird die linke Spalte als x und die rechte als y definiert
31     x, y = np.loadtxt('data/' + klein[a] + '.csv', delimiter=',', unpack=True)
32     # Einlesen der großen Dateien – dabei wird die linke Spalte als u und die rechte als v definiert
33     u, v = np.loadtxt('data/' + gross[a] + '.csv', delimiter=',', unpack=True)
34
35     # Berechnung für jede Datei des kleinen Lautsprechers den maximalen absoluten Amplitudenwert
36     kleinamp[a] = np.max(np.abs(y))
37     # Berechnung für jede Datei des großen Lautsprechers den maximalen absoluten Amplitudenwert
38     grossamp[a] = np.max(np.abs(v))
39
40 # Darstellung der Amplitudenmaxima für beide Lautsprecher im Verhältnis zur Dateinr.
```

```

41 plt.plot(kleinamp, 'b')
42 plt.plot(grossamp, 'y')
43 plt.title("Amplitude")
44 plt.ylabel('Amplitude in V')
45 plt.xlabel('Versuchnr.')
46 plt.grid(True)
47 plt.savefig('amplitudeanzahl.png')
48 plt.show()
49
50 # Darstellung der Amplitudenmaxima für beide Lautsprecher im Verhältnis zur Frequenz
51 plt.plot(zeit, kleinamp, 'b')
52 plt.plot(zeit, grossamp, 'y')
53 plt.title("Amplitude")
54 plt.ylabel('Amplitude in V')
55 plt.xlabel('Frequenz f in Hertz')
56 plt.grid(True)
57 plt.savefig('amplitudefrequenz.png')
58 plt.show()
59
60 # Darstellung des Phasengangs für beide Lautsprecher im Verhältnis zur Dateinr.
61 plt.plot(kleinphase, 'b')
62 plt.plot(grossphase, 'y')
63 plt.title("Phasengang")
64 plt.ylabel('Phasenverschiebung in s')
65 plt.xlabel('Versuchnr.')
66 plt.grid(True)
67 plt.savefig('phasenanzahl.png')
68 plt.show()
69
70 # Darstellung des Phasengangs für beide Lautsprecher im Verhältnis zur Frequenz
71 plt.plot(zeit, kleinphase, 'b')
72 plt.plot(zeit, grossphase, 'y')
73 plt.title("Phasenverschiebung")
74 plt.ylabel('Phasenverschiebung in s')
75 plt.xlabel('Frequenz f in Hz')
76 plt.grid(True)
77 plt.savefig('phasenfrequenz.png')
78 plt.show()
79
80 #for Schleife zum Berechnen des Bode Diagramms
81 for a in range(0, 17):
82     # 20 log10 zur Berechnung des Bode-Diagramms für den großen Lautsprecher

```

```

83 grossamp[a] = 1 / grossamp[a]
84 grossamp[a] = 20 * np.log10(grossamp[a])
85 # 20 log10 zur Berechnung des Bode-Diagramms für den großen Lautsprecher
86 kleinamp[a] = 1 / kleinamp[a]
87 kleinamp[a] = 20 * np.log10(kleinamp[a])
88 # Berechnung des Phasenwinkels mit t*f*360 für den großen Lautsprecher
89 grossphase[a] = (grossphase[a] * -1) * zeit[a] * 360
90 # Berechnung des Phasenwinkels mit t*f*360 für den kleinen Lautsprecher
91 kleinphase[a] = (kleinphase[a] * -1) * zeit[a] * 360
92
93 # Darstellung des Phasengangs für beide Lautsprecher im Verhältnis zur Dateinr.
94 plt.plot(kleinamp, 'b')
95 plt.plot(grossamp, 'y')
96 plt.title("Bode-Amplitude")
97 plt.ylabel('Amplitude in V')
98 plt.xlabel('Versuchnr.')
99 plt.grid(True)
100 plt.semilogx()
101 plt.savefig('bodeamplitudeanzahl.png')
102 plt.show()
103
104 # Darstellung des Phasengangs für beide Lautsprecher im Verhältnis zur Frequenz
105 plt.plot(zeit, kleinamp, 'b')
106 plt.plot(zeit, grossamp, 'y')
107 plt.title("Bode-Amplitude")
108 plt.ylabel('Amplitude in V')
109 plt.xlabel('Frequenz f in Hz')
110 plt.grid(True)
111 plt.semilogx()
112 plt.savefig('bodeamplitudefrequenz.png')
113 plt.show()
114
115 # Darstellung des Phasengangs für beide Lautsprecher im Verhältnis zur Dateinr.
116 plt.plot(kleinphase, 'b')
117 plt.plot(grossphase, 'y')
118 plt.title("Bode-Phasengang")
119 plt.ylabel('Phasenverschiebung in s')
120 plt.xlabel('Versuchnr.')
121 plt.grid(True)
122 plt.semilogx()
123 plt.savefig('bodephasenanzahl.png')
124 plt.show()

```

```
125  
126 # Darstellung des Phasengangs für beide Lautsprecher im Verhältnis zur Frequenz  
127 plt.plot(zeit, kleinphase, 'b')  
128 plt.plot(zeit, grossphase, 'y')  
129 plt.title("Bode–Phasenverschiebung")  
130 plt.ylabel('Phasenverschiebung in s')  
131 plt.xlabel('Frequenz f in Hz')  
132 plt.grid(True)  
133 plt.semilogx()  
134 plt.savefig('bodephasenfrequenz.png')  
135 plt.show()
```

Listing 4.5: Amplitude Phasenverschiebung und Bode-Diagramm berechnen

A.2 Messergebnisse

	klein ms	Abstand intervall	Phasen- verschiebung	ms	Abstand- intervall	Phasen- verschiebung
100	25ms	1 ms	228 μs	25ms	1 ms	505 μs
200	25ms	1 ms	185 μs	25ms	1 ms	82 μs
300	10 ms	0,4 ms	113,2 μs	25ms	1 ms	1 μs
400	10 ms	0,6 ms	105,2 μs	25ms	1 ms	10 μs
500	10 ms	0,4 ms	70,6 μs	25ms	1 ms	12,4 μs
700	10 ms	0,4 ms	64,8 μs	10 ms	0,4 ms	17 μs
950	10ms	0,4ms	34 μs	10 ms	0,4 ms	15,2 μs
1000	5 ms	0,2 ms	4,2 μs	10 ms	0,4 ms	14,8 μs
1200	5ms	0,2ms	1,8 μs	10ms	0,4 ms	15,6 μs
1500	2,5ms	0,1 ms	1,2,2 μs	5 ms	0,2 μs	12,3 μs
1700	2,5ms	0,1 ms	12,1 μs	5 ms	0,2 μs	13,2 μs
2000	2,5ms	0,1 ms	14,7 μs	5 ms	0,2 μs	14,4 μs
3000	1 ms	0,04 ms	8,88 μs	2,5ms	0,1 ms	19,7 μs
4000	1 ms	0,04 ms	11,60 μs	1ms	0,04 ms	12,52 μs
5000	1ms	0,04ms	5,92 μs	1ms	0,04 ms	5,32 μs
6000	500μs	0,02 μs	6,92 μs	1ms	0,04 ms	3,32 μs
10000	250μs	0,01 μs	3,67 μs	0,5ms	0,02 μs	4,74 μs

Kleine
Lautsprecher
Große
Lautsprecher

Abbildung 4.10: Messergebnisse für Task 2