Hochschule Konstanz
Technik, Wirtschaft und Gestaltung

Signale, Systeme und Sensoren

Fourieranalyse und Akustik

D. Wollmann, V. Bratulescu

Konstanz, 14. Dezember 2020

Zusammenfassung (Abstract)

Thema:	Fourieranalyse und Akustik	
Autoren:	D. Wollmann	da161wol@htwg-konstanz.de
	V. Bratulescu	vl161bra@htwg-konstanz.de
Betreuer:	Prof. Dr. Matthias O. Franz	mfranz@htwg-konstanz.de
	Jürgen Keppler	juergen.keppler@htwg-konstanz.de
	Mert Zeybek	me431zey@htwg-konstanz.de

In diesem Versuch wird die Fourieranalyse auf einen akustischen Ton angewendet. Dieser wird in Form eines Keyboards produziert und mithilfe von Python verarbeitet. Dabei wird die Technik der Fouriertransformation angewendet.

Inhaltsverzeichnis

Abbildungsverzeichnis **III**

Tabellenverzeichnis **IV**

Listingverzeichnis **V**

1 Einleitung **1**

2 Versuch 1: Bestimmung der Tonhöhe eines akustischen Signals **2**

2.1 Fragestellung, Messprinzip, Aufbau, Messmittel 2

2.2 Messwerte 4

2.3 Auswertung 5

2.4 Interpretation 7

Anhang **8**

A.1 Quellcode 8

A.1.1 Quellcode Versuch 1 8

Abbildungsverzeichnis

2.1 Versuchsaufbau 2
2.2 Ausschnitt vom Signal des Keyboards 4
2.3 Gesamtes Amplitudenspektrum bis zur Hälfte 4
2.4 Gesamtes Amplitudenspektrum bis zur Hälfte 5
2.5 Ausschnitt des Amplitudenspektrums 6

Tabellenverzeichnis

2.1 Ergebnisse 5
2.2 Ergebnisse 6

Listingverzeichnis

3.1 Skript Versuch 1a 8
3.2 Skript Versuch 1b 9
3.3 Skript Versuch 1c 10

Kapitel 1

Einleitung

In dem dritten Versuch wird die Fourieranalyse auf ein akustisches Signal angewendet. Das akustische Signal wird durch ein Keyboard erzeugt, welches einen konstanten Ton abspielt, der periodisch ist. Der abgespielte Ton mithilfe der pyaudio Bibliothek in Python aufgenommen und weiterverarbeitet. Dabei werden die Grundperiode, Signaldauer und weitere Merkmale berechnet. Anschließend wird durch die Fouriertransformation das Amplitudenspektrum berechnet und graphisch dargestellt.

Kapitel 2

Versuch 1: Bestimmung der Tonhöhe eines akustischen Signals

2.1 Fragestellung, Messprinzip, Aufbau, Messmittel

In diesem Versuch wird die Fourieranalyse auf ein akustisches Signal angewendet. Dazu wird mit einem Keyboard ein konstanter Ton (C4) erzeugt und mittels eines Mikrofons aufgenommen. Der aufgenommene Ton soll eine genügend hohe Amplitude haben und gleichmäßig wiederholende Perioden aufweisen. Der Ton wird über das Mikrofon mittels der pyaudio Bibliothek aufgenommen und anschließen in einer .csv Datei abgespeichert. Das verwendete Mikrofon wurde in der Nähe der Ausgabelautsprecher des Keyboards platziert, sodass der abgespielte Ton gut aufgenommen werden kann.

Abbildung 2.1: Versuchsaufbau

Des Weiteren ist die Aufgabe mehrere Perioden des Signals graphisch darzustellen und anhand dieses Plots die Grundfrequenz und Grundperiode zu ermitteln. Zu dem soll die Signaldauer, Abtastfrequenz, Signallänge und Abtastintervall berechnet werden. Im Anschluss soll die Fouriertransformation mithilfe der Python Funktion numpy.fft.fft() angewendet und daraus das Amplitudensprektrum bestimmt und graphisch dargestellt werden. Dabei ist zu beachten, dass die x-Achse mit der Frequenz nicht in Hertz sondern in Anzahl Schwingungen innerhalb der gesamten Signaldauer definiert ist. Die zugehörige Frequenz f in Hertz kann jedoch durch die Formel

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

berechnet werden. Durch das Amplitudenspektrum kann anschließend die ursprüngliche Grundfrequenz identifiziert werden.

2.2 Messwerte

In Abbildung 2.2 ist ein Ausschnitt einzelner Perioden des aufgenommenen Signals dargestellt, welches wir durch ein Keyboard erzeugt haben. Die x-Achse represäntiert den Zeitpunkt in Millisekunden von der Aufnahme der jeweiligen Messung. Die jeweiligen Zeitpunkte haben wir berechnet und den Daten in der .csv Datei hinzugefügt. Um den jeweiligen Zeitpunkt zu berechnen, haben wir das Abtastintervall mit dem jeweiligen Index des Messwertes multipliziert.

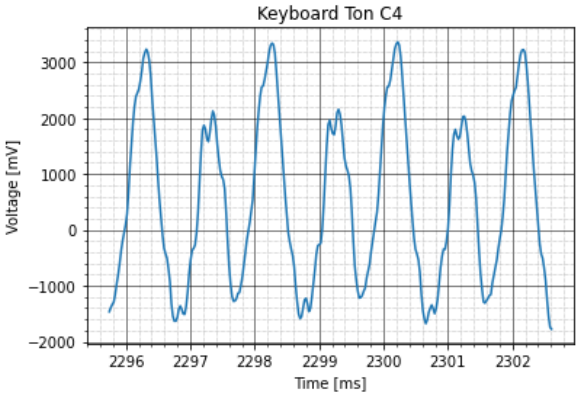

Abbildung 2.2: Ausschnitt vom Signal des Keyboards

In der nachfolgenden Abbildung 2.3 ist das gesamte aufgenommene Signal zu sehen. Da wir den Ton schon vor Beginn der Aufnahme abgespielt haben, ist kein Einschwingvorgang oder ähnliches zu erkennen.

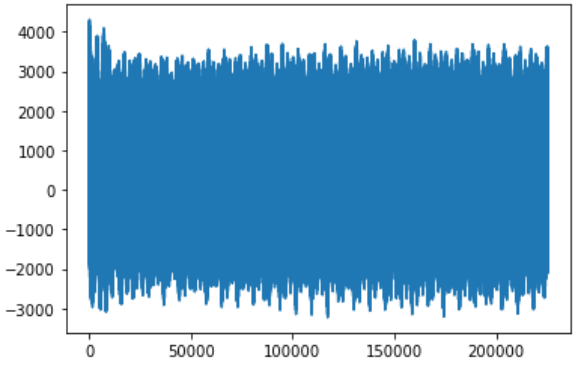

Abbildung 2.3: Gesamtes Amplitudenspektrum bis zur Hälfte

2.3 Auswertung

Aus der Abbildung 2.2 kann die Grundperiode in Millisekunden abgelesen werden, welche 2ms beträgt. Dadurch kann die Grundfrequenz von 500Hz abgeleitet werden. Die Signaldauer, Abtastfrequenz, Signallänge und Abtastintervall können durch einfache Berechnungen ermittelt werden. Die Ergebnisse sind in der Tabelle 2.1 zu sehen.

Beschreibung	**Wert**
Grundperiode [ms]	2
Grundfrequenz [Hz]	500
Signaldauer [sec]	5.171
Abtastfrequenz [Hz]	43558.89
Signallänge [Anzahl der Abtastzeitpunkte]	225280
Abtastintervall [sec]	2.295e-05

Tabelle 2.1: Ergebnisse

Das Amplitudenspektrum wird durch den Betrag der numpy Funktion np.fft.fft() berechnet. Wie zuvor erwähnt ist die x-Achse Beschriftung jedoch in der Einheit Anzahl Schwingungen innerhalb der gesamten Signaldauer und nicht in Hertz eingegeben. Deshalb muss die oben genannte Formel angewandt werden. Außerdem wird das Amplitudenspektrum nur bis zur Hälfte angezeigt. Dieses wird in Abbildung 2.4 dargestellt.

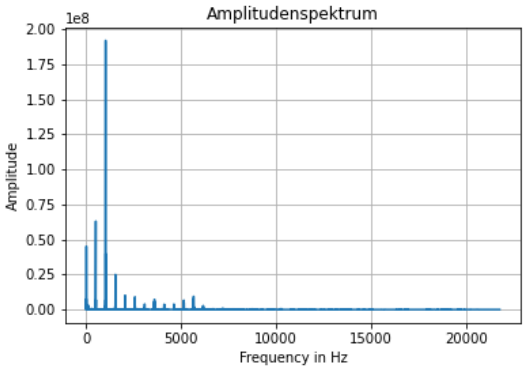

Abbildung 2.4: Gesamtes Amplitudenspektrum bis zur Hälfte

In Abbildung 2.5 ist ein Ausschnitt des Amplitudenspektrums zu sehen.

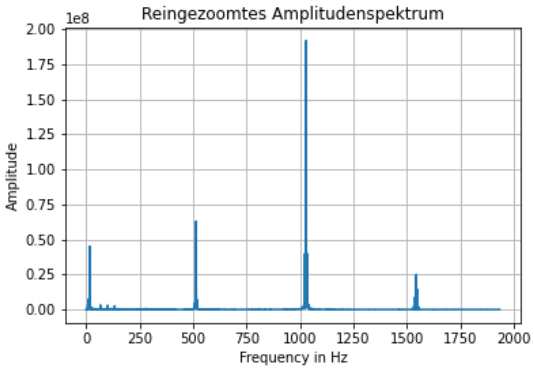

Abbildung 2.5: Ausschnitt des Amplitudenspektrums

Der größte Amplitudenausschlag innerhalb des Spektrums liegt bei der Frequenz 1029Hz. Außerdem ist die Wellenzahl k zu berechnen, die wir mit der Formel

$$k = \frac{1}{\lambda}$$

ermittelt haben. Wobei λ die Wellenlänge ist, die in unserem Fall 2ms beträgt. Dies kann in der Abbildung 2.2 abgelesen werden. Dadurch ergibt sich die Wellenzahl von 500Hz.

Die Frequenz mit dem maximalen Amplitudenwert kann mithilfe einer einfachen for Schleife, die über das Spektrum iteriert, ermittelt werden.

Beschreibung	**Wert**
Maximaler Amplitudenausschlag	191860000.7
Frequenz mit der größten Amplitude [Hz]	1029.22

Tabelle 2.2: Ergebnisse

2.4 Interpretation

Wie zu erwarten, ist in Abbildung 2.2 sehr gut ersichtlich, dass der Ton gleichmäßig und mit konstanter Tonhöhe gespielt wurde.

Aus der Tabelle 2.1 lässt sich schließen, dass die Signaldauer das Ergebnis der Multiplikation von der Signallänge mit dem Abstastintervall ist. Des Weiteren ist das Abtastintervall komplementär zur Abtrastfrequenz, da die Frequenz zunimmt, wenn das Intervall abnimmt. Analog gilt das auch für die Grundperiode und Grundfrequenz.

In Abbildung 2.5 sind die harmonischen vielfachen der Grundfrequenz sehr deutlich zu sehen. Des Weiteren ist sehr schön zu sehen, dass nicht so viele Obertöne vorhanden sind, was darauf zurückzuführen ist, dass der Ton durch ein digitales Keyboard erzeugt wurde und nicht durch ein klassisches Klavier.

Anhang

A.1 Quellcode

A.1.1 Quellcode Versuch 1

```
# −∗− coding: utf−8 −∗−

import time
import pyaudio
import numpy
import matplotlib.pyplot as plt
FORMAT = pyaudio.paInt16
SAMPLEFREQ = 44100
FRAMESIZE = 1024
NOFFRAMES = 220

start = time.time()

p = pyaudio.PyAudio()
print("running")
stream = p.open(format=FORMAT,channels=1,rate=SAMPLEFREQ,input=True,frames_per_buffer=FRAMESIZE)
data = stream.read(NOFFRAMES∗FRAMESIZE)
decoded = numpy.fromstring(data, "Int16");
stream.stop_stream()
stream.close()
p.terminate()

end = time.time()

abtastintervall = ((end − start) ∗ 1000) / (NOFFRAMES∗FRAMESIZE)

plt.plot(decoded)
plt.show()
```

```

decodedWithTimestamp = numpy.zeros((decoded.size,2))

for i in range(decoded.size):
    decodedWithTimestamp[i,0] = decoded[i]
    decodedWithTimestamp[i,1] = abtastintervall * (i)

#Aufnahme Sound 111
#Lautstärke 6
numpy.savetxt("decoded.csv", decodedWithTimestamp, delimiter=",")
```

Listing 3.1: Skript Versuch 1a

```
# -*- coding: utf-8 -*-

import numpy as np
import matplotlib.pyplot as plt

data, time = np.genfromtxt("decodedRichtig.csv", delimiter=",", unpack=True, skip_header=0)

plt.title('Keyboard Ton C4')
plt.xlabel('Time [ms]')
plt.ylabel('Voltage [mV]')

plt.minorticks_on()
plt.grid(which='major', linestyle='-', linewidth='0.5', color='black')
plt.grid(which='minor', linestyle=':', linewidth='0.5', color='gray')

plt.plot(time[100000:100300], data[100000:100300],  "-")
plt.savefig('Keyboard.png', dpi=900)
plt.show()

#Abgelesen von Plot:
#Eine Schwingung geht von 2296 1,5 bis 2298 1,5
#Grundperiode => 2ms
#Grundfrequenz => 500Hz

print("Grundperiode: 2ms")
print("Grundfrequenz: 500Hz")

#kann auch durch Abtastintervall * Signallänge berechnet werden
gemesseneZeit = 5.17184853553772 #sekunden

```

```
NOFFRAMES = 220
FRAMESIZE = 1024

abtastintervall = (gemesseneZeit / (NOFFRAMES*FRAMESIZE))
signaldauer = gemesseneZeit
signallaenge = NOFFRAMES*FRAMESIZE
abtastfrequenz = 1/(abtastintervall)

print("Signaldauer: " + str(signaldauer) + "sec")
print("Abtastfrequenz: " + str(abtastfrequenz) + "Hz")
print("Signallänge (Anzahl Abtastzeitpunkte): " + str(signallaenge))
print("Abtastintervall: " + str(abtastintervall) + "sec")
```

Listing 3.2: Skript Versuch 1b

```
# -*- coding: utf-8 -*-

import numpy as np
import matplotlib.pyplot as plt

data, time = np.genfromtxt("decodedRichtig.csv", delimiter=",", unpack=True, skip_header=0)

abtastintervall = 0.022957424252209337 / 1000 #sec
signallaenge = 225280

data = data # / 1000 #von mV in V

fourier = np.fft.fft(data)

spektrum = np.abs(fourier)

haelfte = int(signallaenge / 2)

print(abtastintervall * signallaenge)

freq = np.zeros(signallaenge)

# Formel um die Anzahl der Schwingungen in die Freuquenz umzurechnen - f = n / (M * (delta)t)
for n in range(0, signallaenge, 1):
    freq[n] = n/(abtastintervall * signallaenge)

# Darstellung des Amplitudenspektrums
plt.plot(freq[:haelfte], spektrum[:haelfte], "-")
```

```
plt.grid()
plt.title("Amplitudenspektrum")
plt.xlabel('Frequency in Hz')
plt.ylabel('Amplitude')
plt.savefig('AmplitudenspektrumBreit.png')
plt.show()

# Darstellung des Amplitudenspektrums
plt.plot(freq[:10000], spektrum[:10000], "-")
plt.title("Reingezoomtes Amplitudenspektrum")
plt.grid()
plt.xlabel('Frequency in Hz')
plt.ylabel('Amplitude')
plt.savefig('AmplitudenspektrumSchmal.png')
plt.show()

maxValue = 0
maxFreq = 0

for i in range(0, haelfte):
    if(maxValue < spektrum[i]):
        maxValue = spektrum[i]
        maxFreq = freq[i]

print("Maximalster Amplitudenausschlag", round(maxValue, 1))
print("Frequenz mit der größten Amplitude", maxFreq)
```

Listing 3.3: Skript Versuch 1c