Assignment 3: Fourier Transformation in Matlab

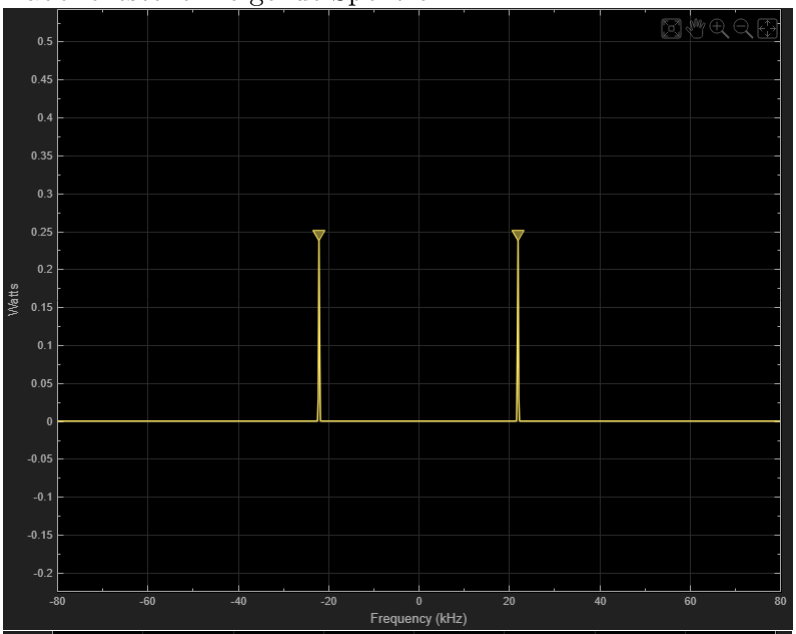
Angewandte Modellierung 25

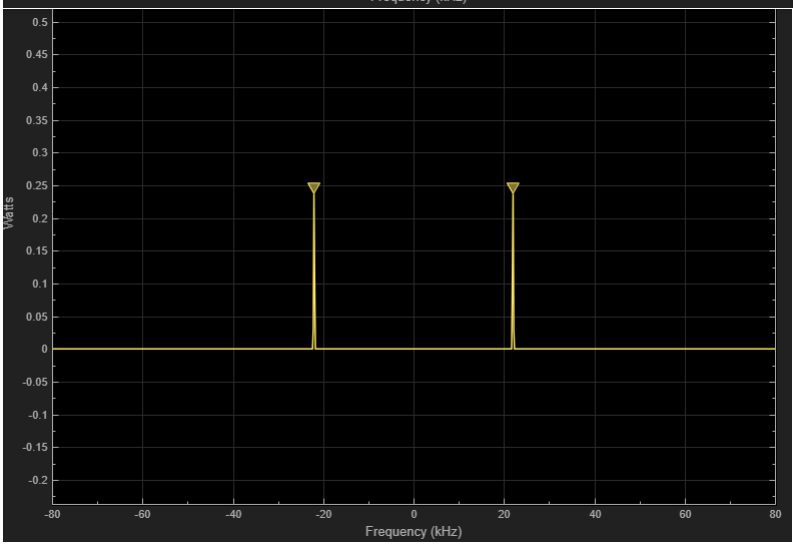
Carl Colmant

May 17, 2025

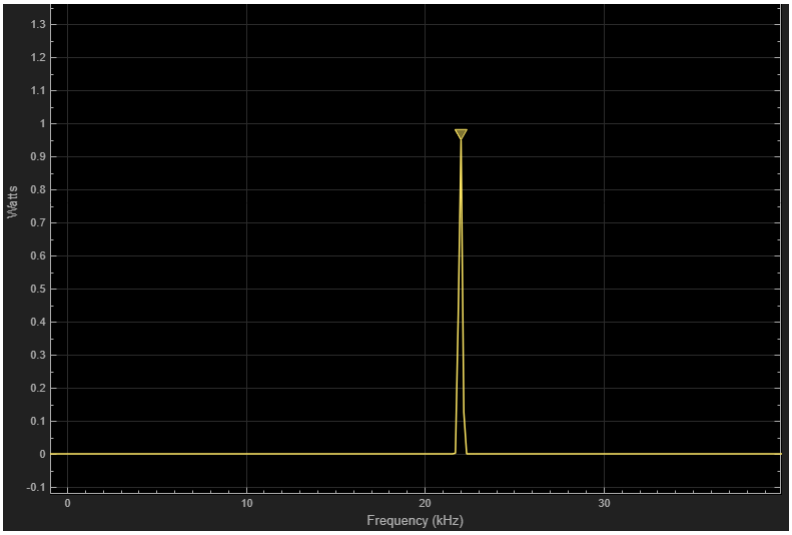
Exercize 1. Modellierung von Sinussignalen

Zu erst habe ich die Blöcke wie in der Aufgabe beschrieben erstellt (meine letzte Ziffer der matrikelnr. ist 2).
Dabei entstehen folgende Spektren:





Und das Spectrum der Summe der Signale:

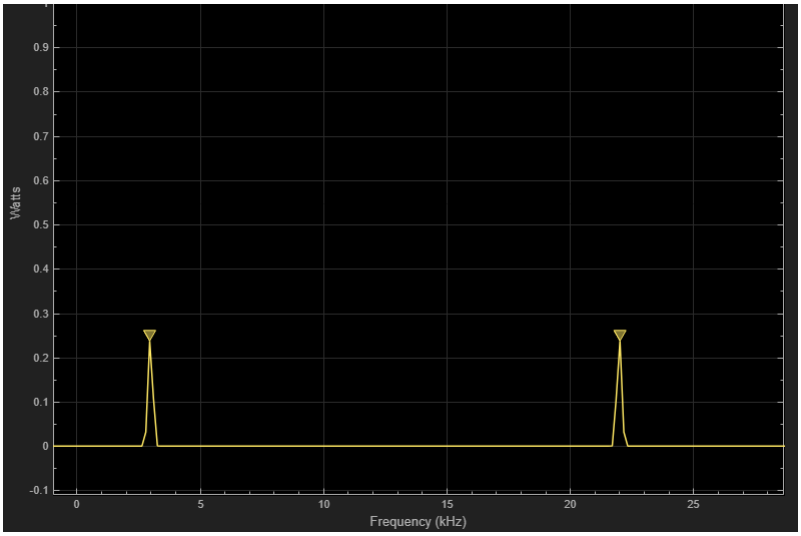


Die beiden Sinus funktionen haben die Frequenz 22000Hz die erste wird um $\frac{\pi}{2}$ verschoben heißt also $sin(22000 \cdot t + \frac{\pi}{2})$ und die zweite ist $sin(22000 \cdot t)$. Weil die beiden Signale die gleiche Frequenz haben, bekommen wir im Spectrum Analyzer den selben Peak bei 22000Hz und -22000Hz.
Die zweite Sinus Funktion wird vor dem Addieren noch mit j multipliziert. Das heißt wir addieren $cos(22000t) + j * sin(22000t) = e^{j*22000t}$ und das ist die Darstellung der komplexen Exponentialfunktion. Die nur eine Spectrallinie hat bei 22000Hz.

Das kann auch in der Simulation gesehen werden.

Frequenz Veränderung

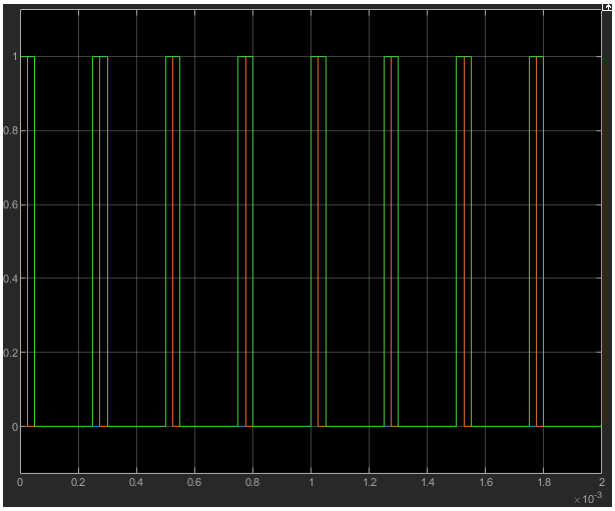
Die Frequenz der Sinus Signale wird in Block Sine Wave 1 auf $2\pi * 3000$ verringert. Nun haben wir keine gleichen Frequenzen der beiden Signale das heißt es entsteht ein überlagertes komplexes Signal, mit 2 SPectral linien.



Phasen Verschiebung

Wenn man die Phase im zweiten Sine wave Block auf 4 setzt ergibt sich kein Unterschied im Spectrum. Das liegt daran das sich die Phase nicht auf die Frequenz auswirkt.

Exercize 2: Analyse von puls Generator Spektren



Wenn man sich den Scope ansieht sieht man das der dritte und vierte Puls Block die doppelte Frequenz hat wie der erste und zweite. Nun ist die Pulse width im ersten block 5% und im zweiten 10%. Der dritte block hat die doppelte Pulse width wie der erste Block also auch 10% und damit die selbe width. Der vierte Block hat die doppelte width wie der zweite Block (20%) und damit die selbe width. Das liegt daran dass die

breite des Puls mit der Periode wächst und der Dritte und vierte Block die hälfte der Periodenzeit haben wie der erste und zweite Block. Die Frequenz unterschiede sind auch auf die unterschiedlichen Periodenzeiten zurückzuführen($freq = \frac{1}{periode}$)

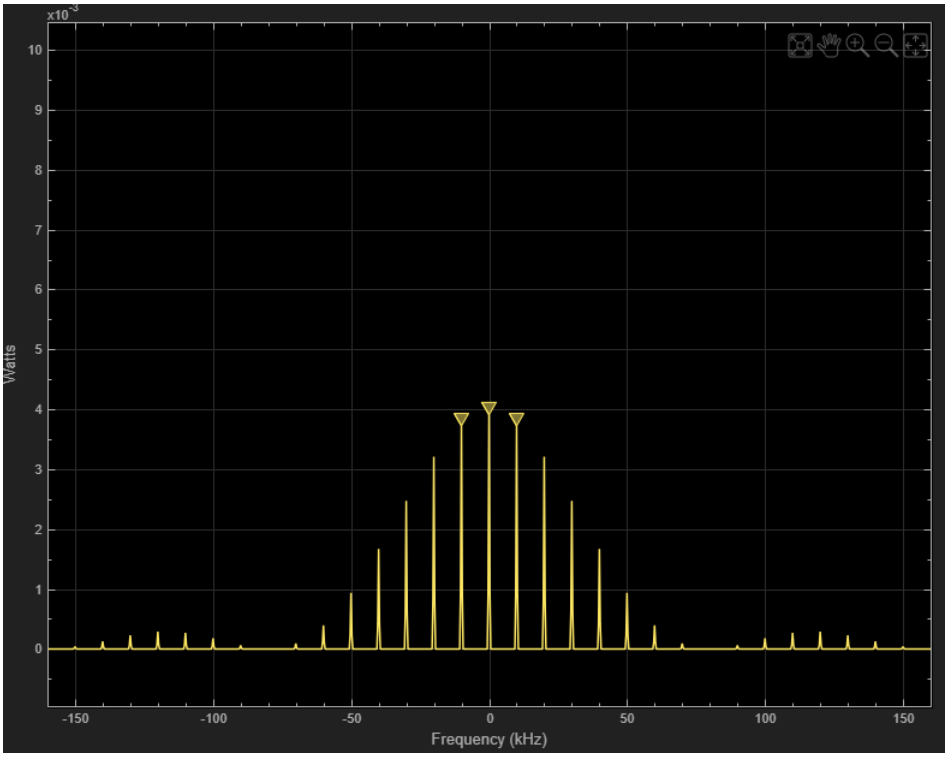
Der Intervall zwischen zwei Spektrallinien sollte nach der Formel 2kHz sein, das lässt sich so ungefähr auch im Spectrum Analyzer ablesen mithilfe von dem Data cursor.
Der erste Nullpunkt des Pulses ist abhängig von der Pulse width und der Periodenzeit. So kleiner die Perioden Zeit desto größer die Puls Größe(also desto größer der Nullpunkt), genauso beeinflusst auch die Pulse width die Pulse Größe so kleiner sie ist desto später kommt der Nullpunkt.

Exercize 3: Period signal generation

Um ein bestimmtes Signal mit einem bestimmten Spectrum zhu erzeugen, brauchen wir die Grund Frequenz den ersten Nullpunkt des Spectrums und müssen dann die Periode und die Pulse width berechnen. AUf dem Bild kann man erkennen das die Grund Frequenz 10 kHz beträgt.
Dazu Habe ich folgendes Matlab Script geschrieben:

```
%the first null in the Envelope is at 80kHz
% f_0 = 2kHz
% T = 1/f_0

T = 1/10000;

% for the Pulse width:
% f = 1/roh
% f = 80kHz
roh = 1/80000;

%period width percent = rho/period *100

w = roh/T * 100
```

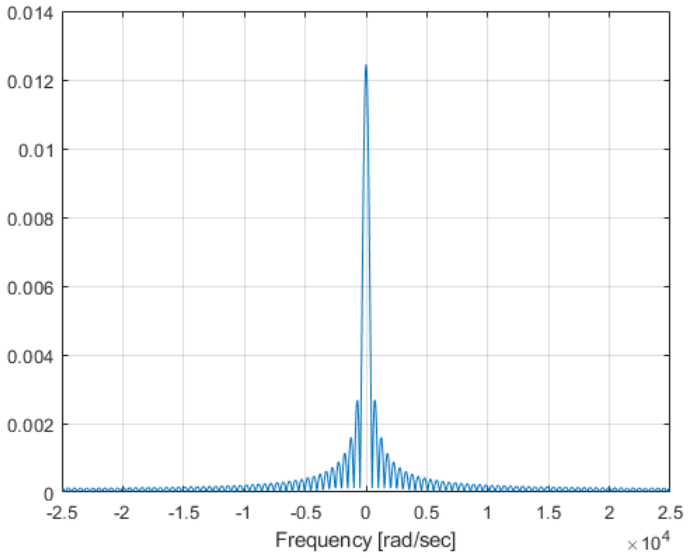
Das Scrip berechnet die Periode als $1e-4$ und die Puls width als $1.25e-5$ damit die prozentuale Pulse width mit $12,5$.
Am ende muss man noch die Amplitude anpassen, so dass man das richtige Spectrum bekommt. Mit der Amplitude von $0,5$ bekommt man ein sehr ähnliches Spectrum:


x10^-3
10
9
8
7
6
5
4
3
2
1
0
Watts
-150
-100
-50
0
50
100
150
Frequency (kHz)

Exercize 4: Fourier Transformation

```
% wir lassen w von -25000 bis 25000 mit 50er schritten gehen.
for w=-25000:50:25000

    % Das ist an sich eine Fourier Transformation nur das das integral
    % durch eine Riemann summe ausgetauscht wurde/approximiert wird. Die
    % rectpuls funktion agiert hier als ein Ersatz für den dirac Impuls der
    % nur Mathematisch existiert. Die breite des rectpuls entscheidet über
    % die genauig keit der Fourier Transformation.
    Fw=sum(rectpuls(t_vector, 0.025).*exp(-1i*w*t_vector))*t_step;

    %Die Werte werden gespeichert damit sie nach dem For loop geplottet
    %werden können.
    w_vector=[w_vector w];
    Fw_vector=[Fw_vector Fw];

end
plot(w_vector,abs(Fw_vector))
xlabel('Frequency [rad/sec]')
grid
```

Dann Erhält man folgendes Spectrum:

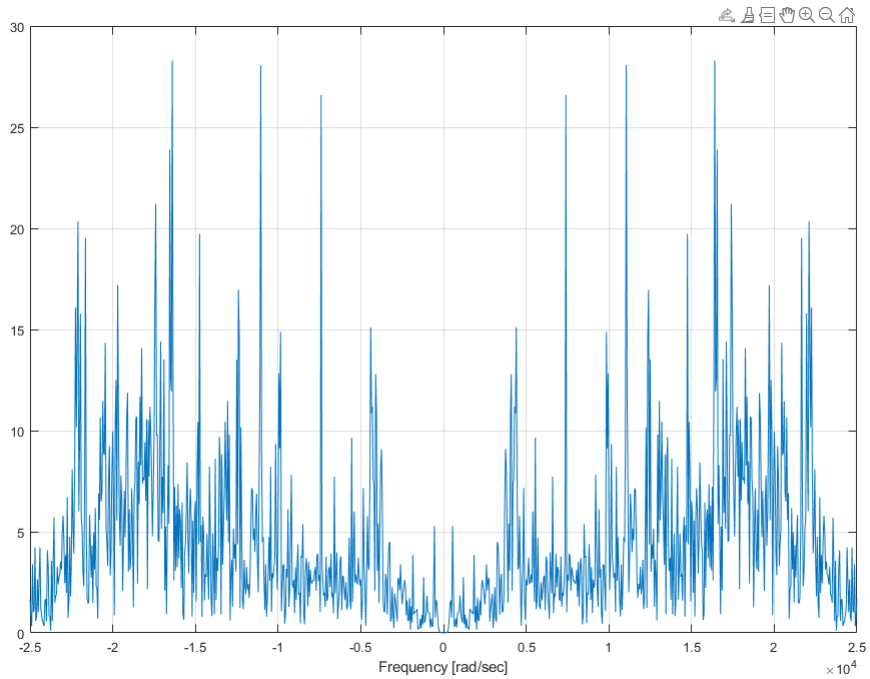


Das menschliche Ohr kann Frequenzen von 20Hz bis 20kHz hören. Normlaerweise sind in Aufnahmen deshalb Frequenzen von bis zu 40 kHz zu sehen weil die Digitalisierung mit doppelter Samplerate erfolgt. Da wir Frequenzen von bis zu 2.5 kHz sehen kann davon ausgegangen werden, dass die Daten zumindest hörbare Signale sind.
Wenn man diese nun mit den in der Aufgabe genannten Funktionen abspielt, hört man tatsächlich etwas, es klingt wie eine Aufnahme von einem Konzert. Die Funktion soundsc resultiert in einer eher schlechten Audioqualität da es mehrfach übersteuert und die Lautstärke stark erhöht ist. Die sound Funktion hingegen klingt sehr viel besser und eher wie eine Moderne Aufnahme.

Exercize 5: High pass filter with differentiation

```
clear
load for_ps3.mat
t_step = t_vector(2) - t_vector(1);

% Länge des ft Vectors
L = length(ft_vector);

% Berechung der Ableitung von den Werten des ft_vectors
diff_ft = (ft_vector(2:L) - ft_vector(1:(L-1))) / t_step;

% Den Zeit vector (t_vector) an die Länge der Ableitung anpassen weil wir
% die Ableitung numerisch ausgerechnet haben zwischen zwei wert Paren fehlt
% hinten ein wert.
t_vector = t_vector(1:(L-1));

Dw_vector = [];
w_vector = [];

% wir lassen w von -25000 bis 25000 mit 50er schritten gehen.
for w = -25000:50:25000


    % Die Fourier Transformation für die Ableitung ausrechnen.
    % die Fehlenden Terme waren diff_ft, w und der neue t_vector. Die
    % Ableitung wird wie eine Art Fenster benutzt.
    Dw = sum(diff_ft .* exp(-j * w * t_vector)) * t_step;

    % Den Wert der Ableitung dem DW_Vector hinzufügen
    Dw_vector = [Dw_vector Dw];
    % w wert zum w vector hinzufügen.
    w_vector = [w_vector w];
end

% Plot
plot(w_vector, abs(Dw_vector));
xlabel('Frequency [rad/sec]');
grid on;
```

Mit dem resultierenden Spectrum:



Wenn man sich den ft_Vector anhört hört man relativ klar den Bass im hintergrund. Wenn man nun aber die Ableitung des Signals berechnet wie oben und sich den resultierenden Vector anhört hört man den Bass fast nicht mehr. Die Ableitung fungiert also tatsächlich wie ein Hochpassfilter. Das kann man auch am Spectrum erahnen hier wurden vor allem die hohen Frequenzen stark verstärkt während die niedrigen Frequenzen gerade die um Null rum stark gedämpft werden.

Github

Wie immer sind alle meine benutzten Dateien auf meinem Github zu finden.