Octave 3.3

Gegeben ist das System:

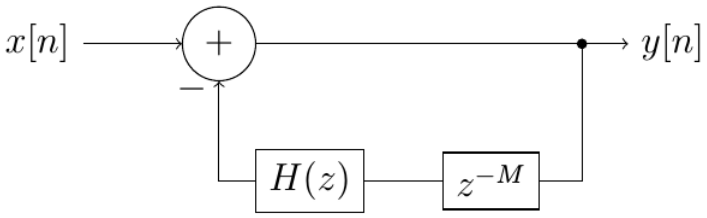

Figure 1: System

Bei $H(z)$ spricht man von der z-Transformierte der Impulsantwort$h[n]$ und z^{-M} ist die Verzögerung um M Samples.

a) Problem: Finde eine Differentialgleichung, die das System beschreibt und bestimmen der $H_1(z) = \frac{Y(z)}{X(z)}$.

Nach Umformung von der Abbildung 1 erhalten wir folgende Differenzengleichung:

$$y[n] = x[n] - h[n-m]$$

Die zugehörige Übertragungsfunktion sieht folgendermaßen aus:

$$\begin{aligned} H_1(z) &= \frac{Y(z)}{X(z)} \\ Y(z) &= X(z) - H(z)z^{-M}Y(z) \\ Y(z)\left(H(z)z^{-M}+1\right) &= X(z) \\ H_1(z) &= \frac{Y(z)}{X(z)} = \frac{1}{1+H(z)z^{-M}} \end{aligned}$$

$H_1(z)$ werden wir bei späteren Berechnungen brauchen.

b) Problem: Eine Audiodatei einlesen und den Absolutbetrag plotten.

Durchführung in Matlab

Algorithm 1 Berechnung des Frequenzvektors

```
[x, fs] = audioread('Data/sample1.wav');
N = length(x);
f = (0:(N-1))*fs/N;
xdft = fft(x);
[~,index] = max(abs(xdft(1:length(x)/2+1)));
freq = 0:(fs/length(x)):fs/2;
```

Wir haben ein rein reelwertiges Signal, dadurch wissen wir dass das DFT Spektrum symmetrisch wird. Bei max() brauchen wir nur die Hälfte des Spektrum betrachten müssen, weil es eben symmetrisch ist, um das Maximum zu finden. Mit den Index bestimme ich beim Plot die Frequenz, wo sie nicht 0 ist, sondern das Maximum (also wo die Amplitute max ist).

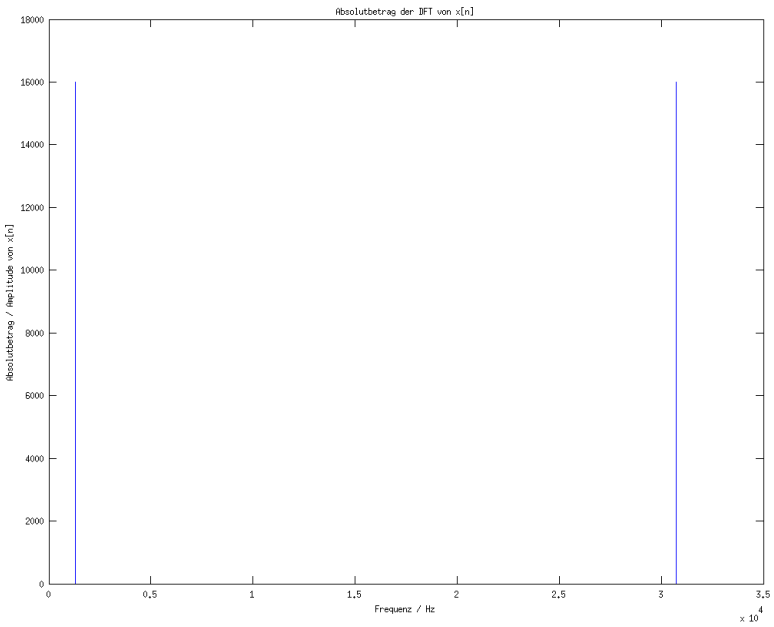

Figure 2: Grundfrequenz f_0 des zugrunde liegenden Signals - Absolutbetrag von xdft

c) Problem: Plot der ersten Periode des Signals $x[n]$

Es soll ein Vektor x_1 erstellt werden, der die erste Periode der Länge T_0 des Signals $x[n]$ enthält. In Punkt a wurde die Grundfrequenz $f_0 = 640$ ermittelt. Mit dieser kann die Periodendauer T_0 ermittelt werden:

$$T_0 = \frac{1}{f_0}$$

Jetzt wissen wir nur die Periodendauer T_0, aber nicht wieviele Samples dabei sein sollen. Dazu muss die Samplefrequenz mit der größten Amplitude dividiert werden.

$$samples = \frac{f_s}{f[index]} = 25 \tag{1}$$

Durchführung in Matlab

Algorithm 2 Berechnung des Frequenzvektors

```
1 T_0 = 1/f(index);
2 cnt_samples = fs / f(index);
3 fig = figure(201);
4 y_axis = (0:1/fs:T_0);
5 x1 = x(1:cnt_samples);
6 plot(y_axis(1:cnt_samples), x(1:cnt_samples));
7 title('Signal x1');
8 ylabel('x1[n]');
9 xlabel('n');
```

Wie in der Problemstellung erklärt wird die Periodendauer benötigt, jedoch die Variable *index* wurde schon früher ermittelt, siehe Listing 1. In der 6 Zeile vom Algorithmus 2 wird die erste Periode in die Variable $x_1$gespeichert, wobei die Formel 1 benötigt wurde um auf die Anzahl der Samples zu kommen.

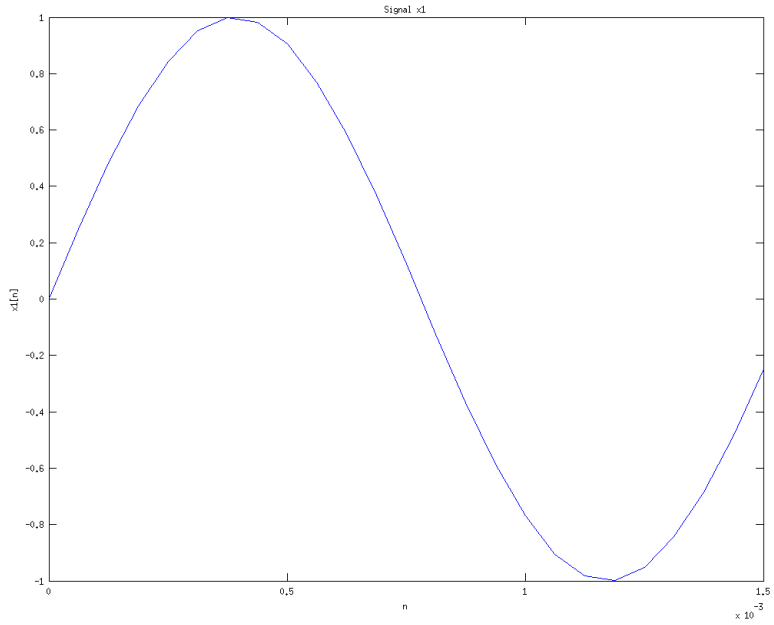

Figure 3: 1 Periodendauer des Signals $x[n]$

d) Welchen Wert muss α annehmen, dass $H_1(z)$ stabil ist?

Wir wissen aus vorherigen Berechnungen, dass

- $M = T_0 \cdot fs = 25$
- $H(z) = \alpha, \quad \alpha \in \mathbb{C}$

Was muss α erfüllen, damit $H_1(z)$ stabil ist?

Wir wissen, dass $H_1(z) = \frac{1}{1+z^{-M}\alpha}$.

Damit ein System stabil ist, muss der Einheitskreis im ROC sein. Das bedeutet, dass alle Pole im Einheitskreis liegen müssen. Der Pol von $H_1(z)$ errechnet sich durch

$$1 + z^{-M} \cdot \alpha = 0$$

Wir wissen dass Pole die From von $z - z_0$ haben und können die obrigen Term umformen:

$$z^{-M} + \frac{1}{\alpha} = 0$$

Nun muss α so gewählt werden, dass der Betrag des Bruchs $\frac{1}{\alpha} < 1$ bleibt, damit die Pole im Einheitskreis bleiben. Daraus folgt, dass $|\alpha| > 1$ sein muss.

e) Problem: KS-algorithm

Eine Funktion $ksalgorithm(x, \alpha, M, Nout)$ soll in einer eigenen Datei (bei uns heisst sie ksalgorithm.m) erstellt werden, der ermöglicht den Klang einer gezupften Saite zu synthesieren.

Durchführung in Matlab

Algorithm 3 Berechnung des Frequenzvektors

```
function [ y ] = ksalgorithm( x, alpha, M, Nout )
    x1 = zeros('like',x);
    x1(1:M) = x(1:M);
    a = [1 zeros(1,M-1) -alpha];
    tmp = filter(1,a,x);
    for n = M:Nout
        x1(n) = x(n) + tmp(n-M+1);
    end
    y = x1;
end
```

Die ersten M Werte werden von x nach ich x1 kopiert, wobei die restlichen Wert von x1 0 sind. Die *filter(1, a, x1)* Funktion wird mit der Transferfunktion gestopft. Die M ersten Samples bleiben gleich wie beim Original. Bei den restlichen Wert von M bis Nout werden immer wiederkehren die filter Samples dazuaddiert. Anmerkung: Hilfestellung für die Polenaufstellung wurde unter folgenden Link: dsp.stackexchange.com geholt.

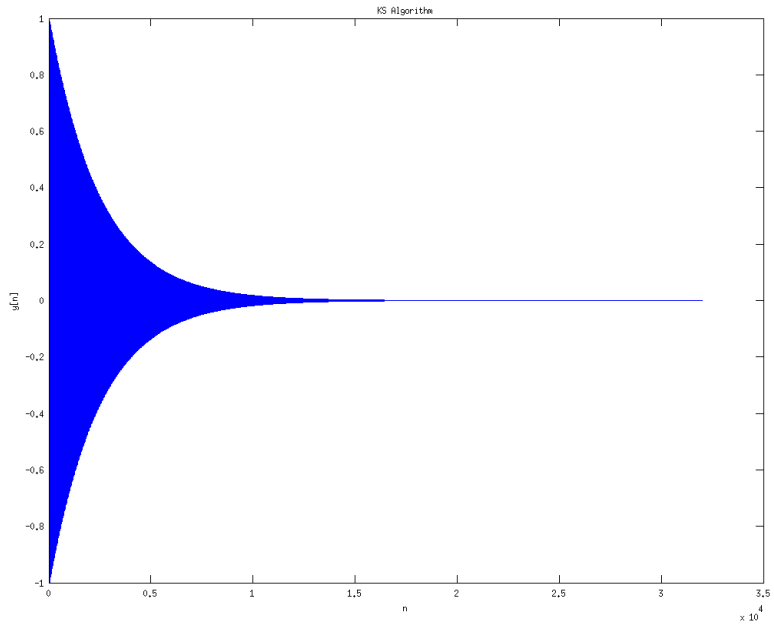

Figure 4: KS-Algorithmus mit x_1

In Abbildung 4 ist zu sehen, dass das Signal abklingt, was auch so sein soll, weil $\alpha < 1$ ist somit das System stabil.

f) Problem: KS-Algorithmus anwenden bei verschieden Verschiebungen M

Es soll nun bei diesem Problem die Audiodateien gespeichert/abgespielt werden und ein Plot erstellt werden. M soll folgende Werte annehmen:

- round(M/2)
- 2M

Hinweis: Die wav-Dateien sind im selben Ordner wie die Matlabdateien mit den Namen *3_3f_25m.wav*, *3_3f_13m.wav* und *3_3f_50m.wav* hinterlegt.
Interpretation der folgenden Dateien:

1. *3_3f_25m.wav: hoher Klang. Kurzer Ausklang.*
2. *3_3f_13m.wav: noch höherer Klang. Noch kürzer Ausklang.*
3. *3_3f_50m.wav: tieferer Klang als die anderen beiden, wobei ein längeres ausklingen deutlich hörbar ist.*

Zusammenfassend lässt sich sagen, dass bei kleinerem M die Frequenz höher ist, wobei die Abklingzeit zugleich auch kürzer ist. Der KS-Algorithmus belässt die ersten M Samples und die Addition (siehe Algorithmus 4) kommt erst nachher zu $x[n > M]$ dazu.

Durchführung in Matlab

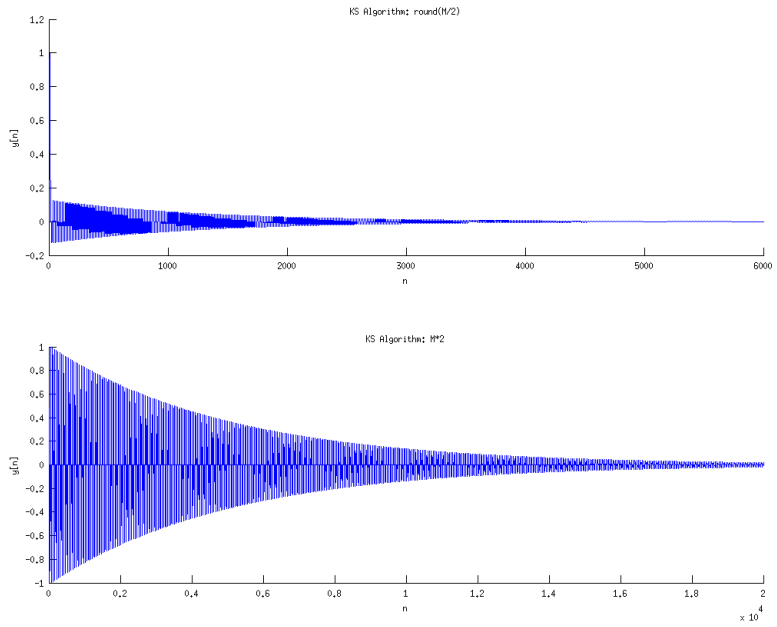

Figure 5: KS-Algorithmus bei $\lceil \frac{M}{2} \rceil$ und $2M$

Bei $\lceil \frac{M}{2} \rceil$ sind viel weniger Samples gebraucht bis der KS-Algorithmus konvergiert. Hingegen, wenn $2M$ verwendet wird, dann braucht er weitaus mehr Samples bis der KS-Algorithmus konvergiert.

g) Problem: Wie beeinflusst α das Ausgangssignal?

Es werden mehrere Subplots erstellt bei verschiedenen $\alpha = \{0.99, 0.5, 1.00, 1.01\}$ *und danach verglichen.*

Durchführung in Matlab

Algorithm 4 Ausgangssignale mit verschiedenen α berechnet.

```
y1 = ksalgorithm(x_input, alpha, 25, Nout);
y2 = ksalgorithm(x_input, 0.5  , 25, Nout);
y3 = ksalgorithm(x_input, 1    , 25, Nout);
y4 = ksalgorithm(x_input, 1.01 , 25, Nout);
```

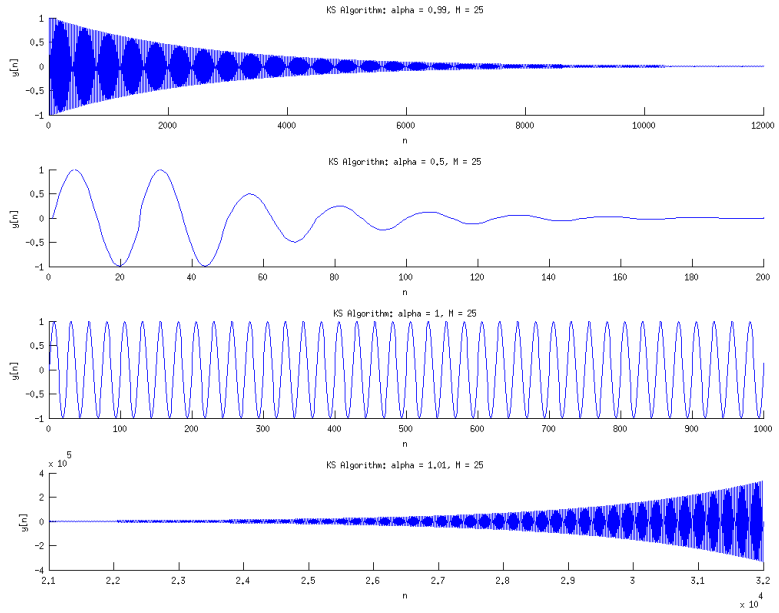

Figure 6: KS Algorithmus mit $\alpha = \{0.99, 0.5, 1.00, 1.01\}$

Bei einem $|\alpha| > 1$ ist das System Stabil, was man erkennen kann dass die beiden oberen Plots gegen Unendlich sich an 0 nähern. Weiters kann gesagt werden je kleiner das α, desto schneller klingt es ab. Die unteren beiden Plots zeigen das Gegenteil, der dritte Plot oszilliert und der vierte Plot divergiert, und somit sind für $\alpha = 1$ und $\alpha = 1.01$ das System instabil.

h) [Bonus] Problem:

Es soll ein neuer Vektor x_2 generiert werden, der die Länge von x_1 besitzt, jedoch mit random Werte befüllt wurde.

Durchführung in Matlab

Algorithm 5 Erzeugung von x_2 und die Ausgangssignale x_2 und y2

```
y_normal = ksalgorithm(x1, 0.99, 25, Nout);
x2 = rand(size(x1));
x2(26:end) = 0;
y2 = ksalgorithm(x2, 0.99, 25, Nout);
```

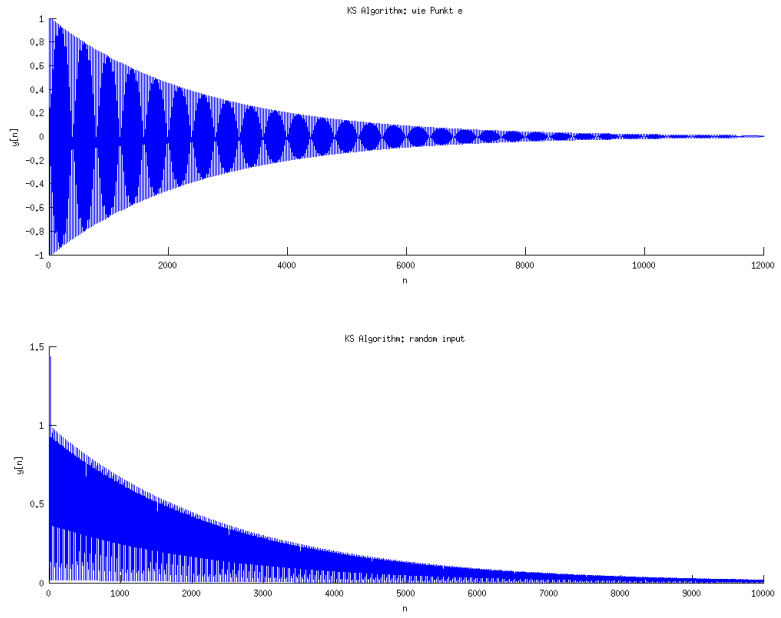

Figure 7: Vergleich von x_1 und x_2

Bei x_2 ist bei den ersten M samples ein größerer Ausschlag erkennbar, was wohl die Werte von der Standardfunktion rand() sind. Außerdem sind die Randomwerte nur positiv. Der KS-Algorithmus lässt sich davon nicht beeinflussen und konvergiert wie bei x_1.

i) [Bonus] Problem:

Ein Messdatensatz steht bereit mit den ersten $\frac{N_{FFT}}{2}$ Punkte einer N_{FFT} langen DFT. Das ist unser Vektor **X**. Das zugehörige Zeitsignal x[n] soll neben Amplitude, Phase, Realteil, Imaginärteil geplotted werden.

Durchführung in Matlab

Algorithm 6 Laden und Vorbereiten der Messdaten

```
X = load('Data/Measurement.mat');
b = cell2mat(struct2cell(X));
br = reshape(b, [1,2048]);
X2 = fliplr(br);
X = [br X2];
x = ifftreal(X);
f = (0:length(X)-1)*100/length(X);
phase = radtodeg(angle(X));
```

Der Algorithmus für ifftreal(X) hat vom Formelzettel Seite 2 die Gleichung $X[k] = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{N}nk}$ implementiert.

Algorithm 7 ifftreal(X)

```
function [ x ] = ifftreal( X )
    N = length(X);
    tmp = zeros(N,N);
    for n = 0:N-1
        for k = 0:N-1
            tmp(k+1,n+1) = exp(1j*2*pi*n*k/N);
        end
    end
    x = X * (1/N .* tmp);
end
```

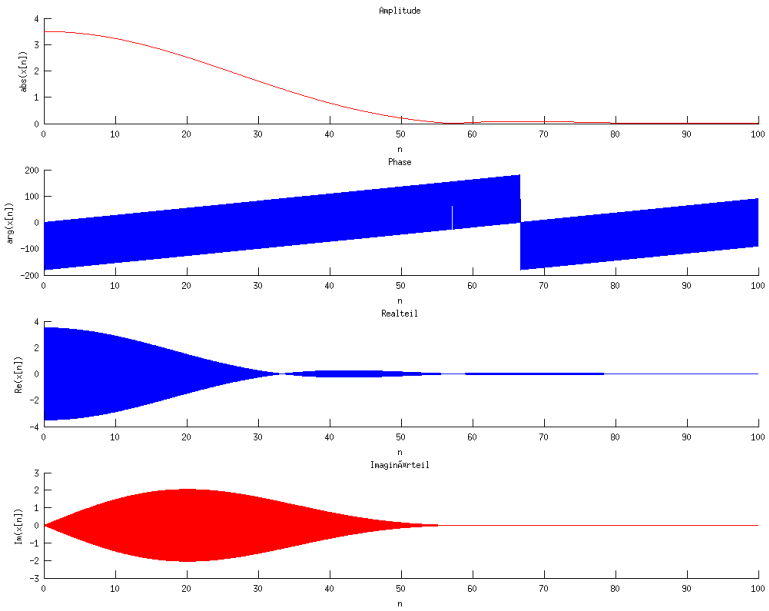

Figure 8: Amplitute, Phase, Realteil und Imaginärteil der N_{FFT}-Punkte langen DFT von $x[n]$

Wie verhalten sich Realteil und Imaginärteil (bzw. Amplitute und Phase)?

Phase passt so, weil durch den Absolutbetrag Phasenspruenge entstehen. Zeiger am Einheitskreis (EK) bewegen sich nur mehr auf der rechter seite des EK, da Realteil durch Absolutbetrag positiv bleibt. Eigentlich gehen sie auf linker weiter, was einen Phasensprung bewirkt.

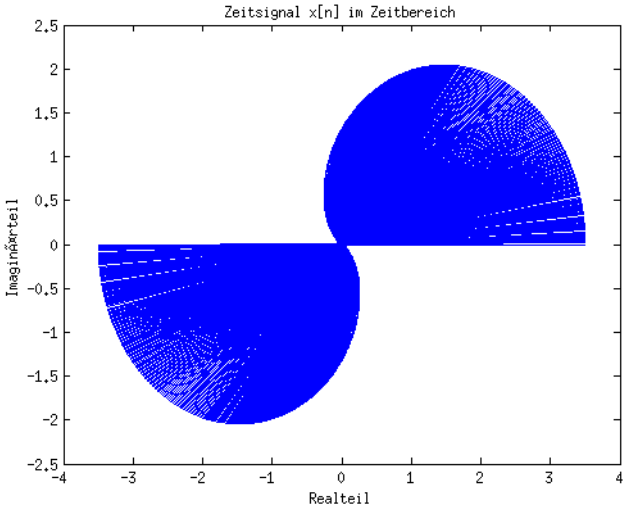

Figure 9: Zeitsignal x[n]