1. Aufgabe: Abtastung

a. Simulieren Sie mit Matlab zwei Cosinussignale der Länge 1 s mit den Frequenzen 1 kHz und 7 kHz. Tasten Sie die beiden Signale mit einer Abtastfrequenz von 8 kHz ab und vergleichen Sie die Abtastfolgen. Wie lässt sich das Ergebnis erklären?

```
% 1a)

% Abtastfrequenz und Wertevektor
fs8 = 8000;
n8 = 0:fs8;

% Abtastung zweier Signale mit Frequenz 1 kHz bzw. 7 kHz
f1 = 1000;
y1 = cos(n8*2*pi*f1/fs8);

f2 = 7000;
y2 = cos(n8*2*pi*f2/fs8);


%% Plot über eine Periode des 1-kHz-Signals
index1 = (fs8/f1);
figure(1)
subplot(2,1,1), stem(n8(1:index1), y1(1:index1))
xlabel('n'), ylabel('y[n]')
title('Abtastfolge 1-kHz-Cosinussignals, fs = 8 kHz')
subplot(2,1,2), stem(n8(1:index1), y2(1:index1))
title('Abtastfolge 7-kHz-Cosinussignals, fs = 8 kHz')
xlabel('n'), ylabel('y[n]')

%%
sound(y1,fs8)
pause(1)
sound(y2,fs8)
```

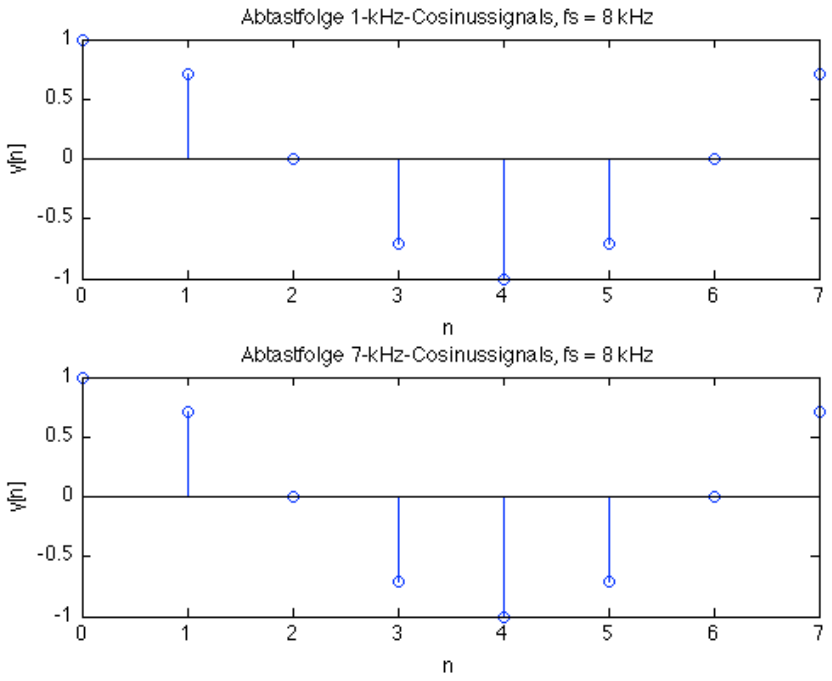


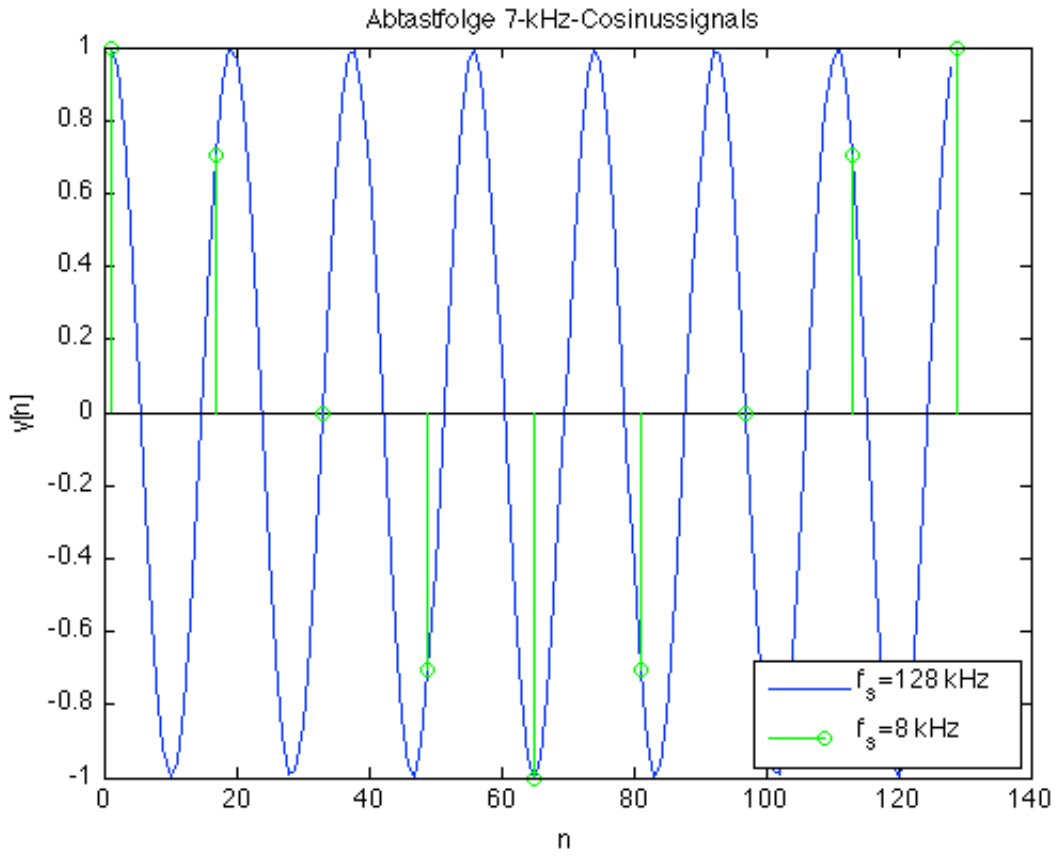
Abbildung 1

Es zeigt sich, dass die beiden Abtastfolgen nicht zu unterscheiden sind. Dies liegt daran, dass das 7-Hz-Signal mit 8 kHz unterabgetastet wird. Das Abtasttheorem $f_S > 2\ f_{max}$ wird nicht eingehalten. Die Spiegelfrequenz, welche durch die Abtastung entsteht, liegt exakt bei 8 kHz - 7 kHz = 1 kHz. Die Uneindeutigkeit zeigt sich auch beim Probehören.

b. Veranschaulichen Sie den bei der Abtastung in Aufgabe a. entstandenen Fehler, indem Sie einen 7 kHz Sinus mit einer Abtastfrequenz von 128 kHz und mit 8 kHz in die selbe Grafik plotten.

```
%% 1b)

% Abtastfrequenz und Wertevektor
fs128 = 16*fs8;
n128 = 0:fs128;

% Abtastung mit neuer Abtastfrequenz
y128 = cos(n128*2*pi*f2/fs828);


figure(2)
plot(y128(1:16*index1))

% y2 in die Grafik von y128 plotten. Weil y128 aber mit dem Faktor
16 Ueberabgetastet ist, muessen die Werte von y2 an jedes 16. Sam-
ple geplottet werden.
hold on
xlabel('n'), ylabel('y[n]')
title('Abtastfolge 7-kHz-Cosinussignals')
for k = 1:9
    stem((k-1)*16+1, y2(k), 'Color', 'g')
```

```
    legend('f_{s}=128 kHz', 'f_{s}=8 kHz', 'location', 'south-
east')
    pause(2)
end
hold off
```



Durch die Verletzung des Abtasttheorems wird die 7 kHz Schwingung zu einer 1 kHz Schwingung.

c. Stellen Sie die Betragsspektren der Signale aus a. im Bereich von $0 < f < f_s$ dar. Wie sehen die Betragsspektren für die selben Signalfrequenzen aber mit $f_s = 9$ kHz bzw. $f_s = 44{,}1$ kHz aus?

```
%% 1c)

% neue Abtastfrequenz = 9 kHz und zugehöriger Zeitvektor
fs9 = 9000;
n9 = 0:fs9;

% Abtastung zweier Signale mit Frequenz 1 kHz bzw. 7 kHz
y3 = cos(n9*2*pi*f1/fs9);
y4 = cos(n9*2*pi*f2/fs9);

% neue Abtastfrequenz = 44,1 kHz und zugehöriger Zeitvektor
fs44 = 44100;
n44 = 0:fs44;

% Abtastung zweier Signale mit Frequenz 1 kHz bzw. 7 kHz
```

```
y5 = cos(n44*2*pi*f1/fs44);
y6 = cos(n44*2*pi*f2/fs44);

% Plot über zwei Perioden des 1-kHz-Signals
index2 = (fs9/f1)*2;
index3 = round((fs44/f1)*2);
figure(3)
subplot(2,2,1), stem(n9(1:index2),y3(1:index2))
xlabel('t [s]'), ylabel('y(t)')
title('Abtastfolge eines 1-kHz-Cosinussignals, fs = 9 kHz')
subplot(2,2,3), stem(n9(1:index2),y4(1:index2))
xlabel('t [s]'), ylabel('y(t)')
title('Abtastfolge eines 7-kHz-Cosinussignals, fs = 9 kHz')
subplot(2,2,2), stem(n44(1:index3),y5(1:index3))
xlabel('t [s]'), ylabel('y(t)')
title('Abtastfolge eines 1-kHz-Cosinussignals, fs = 44,1 kHz')
subplot(2,2,4), stem(n44(1:index3),y6(1:index3))
xlabel('t [s]'), ylabel('y(t)')
title('Abtastfolge eines 7-kHz-Cosinussignals, fs = 44,1 kHz')


%%
sound(y3,fs9)   % 1 kHz mit 9 kHz abgetastet
pause(1)
sound(y4,fs9)   % 7 kHz mit 9 kHz abgetastet
pause(1)
sound(y5,fs44)   % 1 kHz mit 44,1 kHz abgetastet
pause(1)
sound(y6,fs44)   % 7 kHz mit 44,1 kHz abgetastet


%%

% Normierte Darstellung im Frequenzbereich von 0 bis fs
Y1=abs(fft(y1))/max(abs(fft(y1)));
Y2=abs(fft(y2))/max(abs(fft(y2)));
Y3=abs(fft(y3))/max(abs(fft(y3)));
Y4=abs(fft(y4))/max(abs(fft(y4)));
Y5=abs(fft(y5))/max(abs(fft(y5)));
Y6=abs(fft(y6))/max(abs(fft(y6)));
N1 = length(Y1);
N2 = length(Y3);
N3 = length(Y5);
f_index1 = 0:fs8/N1:fs8-fs8/N1;
f_index2 = 0:fs9/N2:fs9-fs9/N2;
f_index3 = 0:fs44/N3:fs44-fs44/N3;

figure(4)
subplot (2,3,1), plot(f_index1,20*log10(abs(Y1)))
xlabel('Frequenz [Hz]'), ylabel('Betrag [dBFS]')
title('Betragsspektrum eines 1-kHz-Cosinussignals, fs = 8 kHz')
axis([0 8000 -50 10])
subplot (2,3,4), plot(f_index1,20*log10(abs(Y2)))
xlabel('Frequenz [Hz]'), ylabel('Betrag [dBFS]')
```

```
title('Betragsspektrum eines 7-kHz-Cosinussignals, fs = 8 kHz')
axis([0 8000 -50 10])
subplot (2,3,2), plot(f_index2,20*log10(abs(Y3)))
xlabel('Frequenz [Hz]'), ylabel('Betrag [dBFS]')
title('Betragsspektrum eines 1-kHz-Cosinussignals, fs = 9 kHz')
axis([0 9000 -50 10])
subplot (2,3,5), plot(f_index2,20*log10(abs(Y4)))
xlabel('Frequenz [Hz]'), ylabel('Betrag [dBFS]')
title('Betragsspektrum eines 7-kHz-Cosinussignals, fs = 9 kHz')
axis([0 9000 -50 10])
subplot (2,3,3), plot(f_index3,20*log10(abs(Y5)))
xlabel('Frequenz [Hz]'), ylabel('Betrag [dBFS]')
title('Betragsspektrum eines 1-kHz-Cosinussignals, fs = 44,1 kHz')
axis([0 44100 -50 10])
subplot (2,3,6), plot(f_index3,20*log10(abs(Y6)))
xlabel('Frequenz [Hz]'), ylabel('Betrag [dBFS]')
title('Betragsspektrum eines 7-kHz-Cosinussignals, fs = 44,1 kHz')
axis([0 44100 -50 10])
```

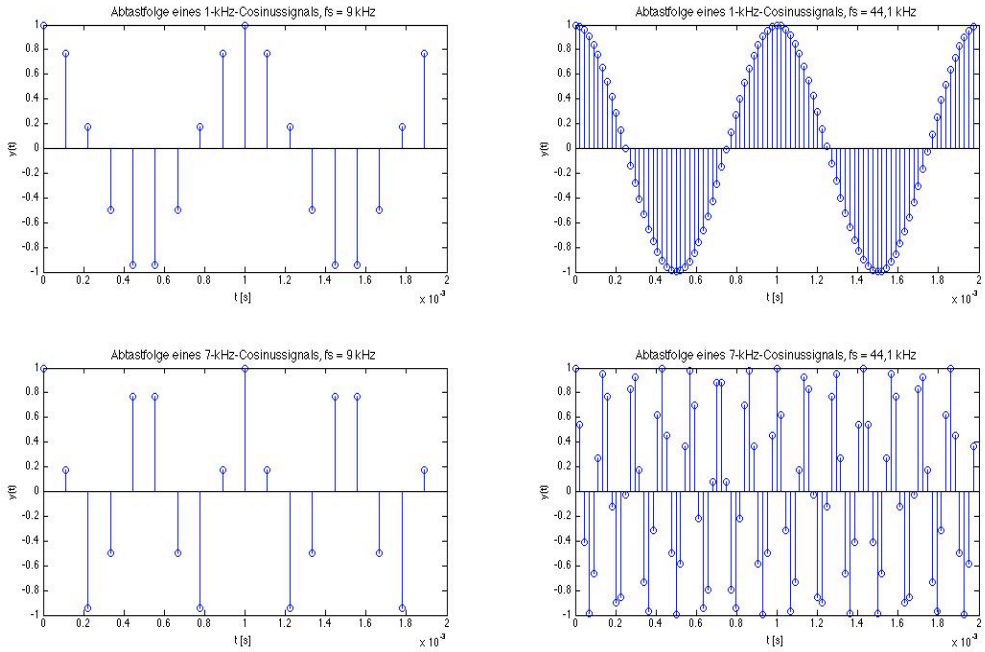


Abbildung 2

Die beiden Abtastfolgen unterscheiden sich zwar bei einer Abtastung mit 9 kHz, das 7-kHz-Signal ist aber immer noch unterabgetastet. Erst bei einer Abtastrate größer als die doppelte Signalfrequenz (in diesem Fall 44,1 kHz) können beide Signale eindeutig rekonstruiert werden (Abtasttheorem). Eine Hörkontrolle macht dies deutlich.

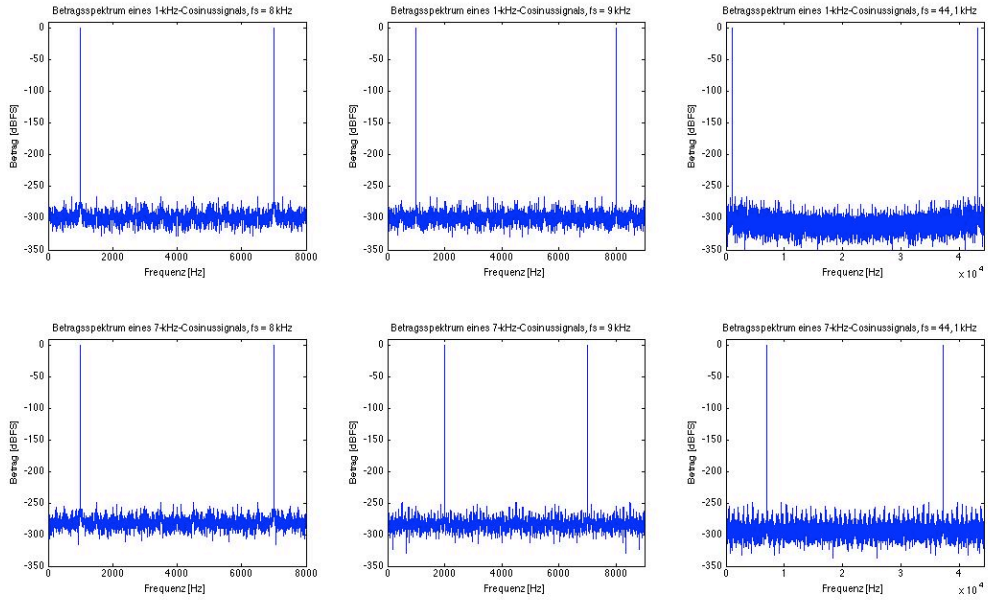


Abbildung 3

Die Betragspektren zeigen, was bereits im Zeitbereich bzw. beim Hören deutlich wurde: Die Spektren aus Aufgabe a) unterscheiden sich nicht. Werden die Signale mit 9 kHz abgetastet, so ergeben sich zwar unterschiedliche Spektren, beim 7-kHz-Signal erscheint aber trotzdem eine Spiegelfrequenz unter der Signalfrequenz (9 kHz - 7 kHz = 2 kHz), sodass diese als Grundton aufgefasst wird. Um dieses Aliasing zu vermeiden, muss vor der Abtastung ein Tiefpassfilter eingesetzt werden (Anti-Aliasing-Filter), das Signalanteile oberhalb $f_S/2$ (Nyquistfrequenz) unterdrückt.
Bei einer Abtastung mit 44,1 kHz ergeben sich zwar auch Spiegelfrequenzen bei $f_S - f$, allerdings erscheinen diese oberhalb der höchsten Nutzfrequenz, und sie sind nicht mehr hörbar. Für die in diesem Fall durch das Hörvermögen hervorgerufene Filterung sorgt im Allgemeinen ein Rekonstruktionstiefpassfilter mit einer Grenzfrequenz bei $f_S/2$.
Es zeigt sich, dass die Rauschpegel der im Verhältnis zur Nutzfrequenz höher abgetasteten Signale leicht tiefer sind. Dies liegt an der Verteilung der Fehlerenergie über das gesamte Spektrum (siehe Oversampling).

2. Aufgabe: Quantisierung

a. Erzeugen Sie ein Sinussignal mit f = 500 Hz der Länge 1 s, tasten Sie es mit einer Abtastrate f_S = 44,1 kHz ab. Schreiben Sie eine Funktion mit folgenden Parametern, die das Signal quantisiert.

xQ = xquant(x, nbits, method)

„method“ soll dabei ein Parameter für die Art der Quantisierungskennlinie sein (mid-tread bzw. mid-rise), „nbit“ die zur Quantisierung benutzte Wortbreite und „x“ das Signal selbst.
Überlegen Sie zunächst wie die zu omplementierenden Quantisierungskennlienen zu beschreiben sind.

```
% 2a)
% Signal erzeugen
```

```
fs = 44100;
f1 = 500;
t = 0:1/fs:1-1/fs;
y = sin(2*pi*f1*t);
```

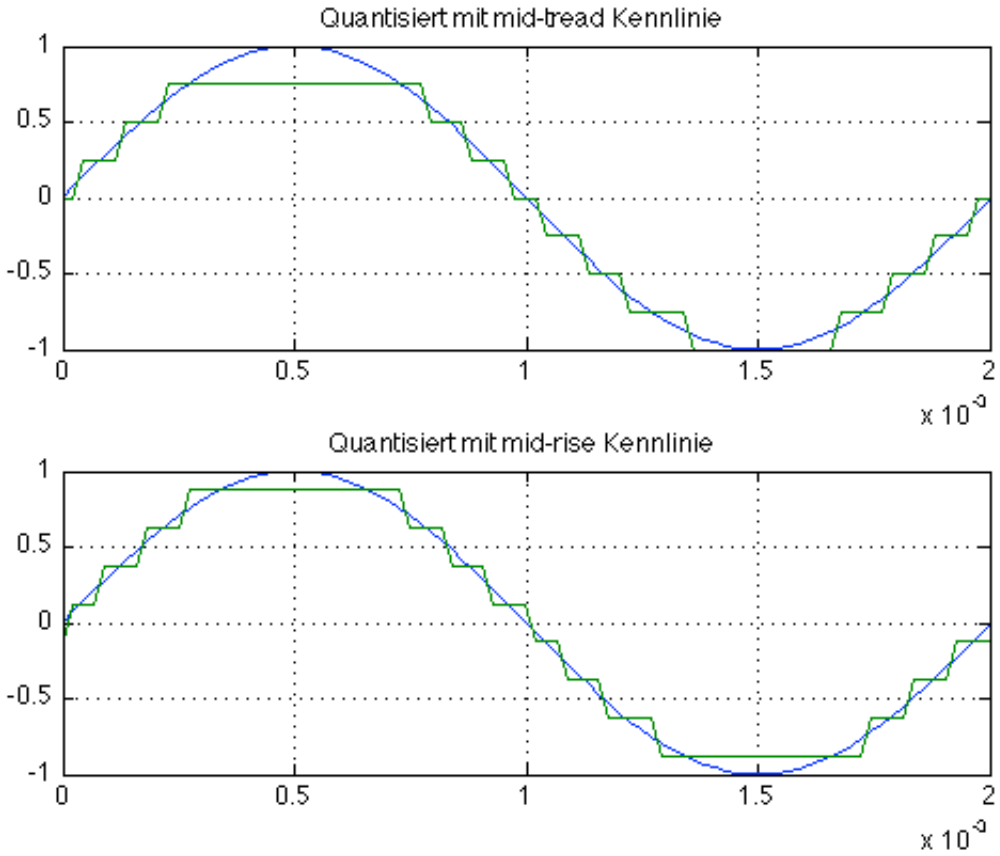
b. Quantisieren Sie das Signal mit einer Wortbreite von 3 bit. Verwenden Sie hierfür sowohl eine mid-tread, als auch eine mid-rise Kennlinie. Plotten Sie das Originalsignal und die quantisierten Signale im Zeit- und Frequenzbereich.

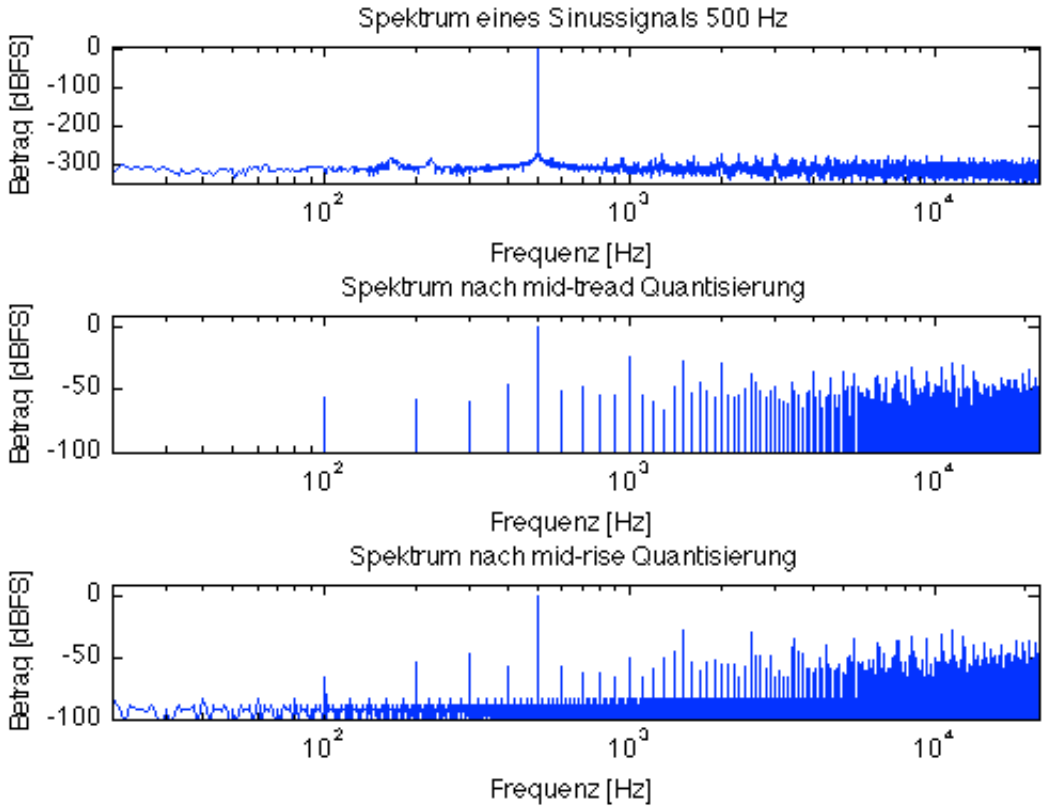
```
%% 2b)
% Signal mit 3 bit quantisieren
yQtread = xquant(y, 3, 'mid-tread');
yQrise  = xquant(y, 3, 'mid-rise');
```

Die Funktion xquant befindet sich im Anhang. Die Eingangssignale müssen MONO sein! Sollten mit der integrierten Plot-Funktion Probleme auftauchen schickt mir den Code, mit dem ihr das Eingangssignal generiert habt, und ich werde versuchen das zu beheben.

```
% Plot im Zeitbereich ueber 1 Periodenlängen
index = ceil(fs/f1);

figure(1)
subplot(2,1,1)
    plot(t(1:index), y(1:index), t(1:index), yQtread(1:index))
    title('Quantisiert mit mid-tread Kennlinie')
    grid on
subplot(2,1,2)
    plot(t(1:index), y(1:index), t(1:index), yQrise(1:index))
    title('Quantisiert mit mid-rise Kennlinie')
    grid on

%% plot im Frequenzbereich
Y       = fft(y) / max(fft(y));
YQtread = fft(yQtread)/ max(fft(yQtread));
YQrise  = fft(yQrise) / max(fft(yQrise));

f = 0:fs/length(t):fs-1/length(t);

subplot(3,1,1)
    semilogx(f, 20*log10(abs(Y)));
    grid on
    axis([0 fs/2 -100 10])
    title('Spektrum eines Sinussignals mit f = 500 Hz')
    xlabel('f in Hz')
subplot(3,1,2)
    semilogx(f, 20*log10(abs(YQtread)));
    grid on
    axis([0 fs/2 -100 10])
    title('Spektrum nach mid-tread Quantisierung')
    xlabel('f in Hz')
    ylabel('Amplitude in dB')
subplot(3,1,3)
```

```
semilogx(f, 20*log10(abs(YQrise)));
grid on
axis([0 fs/2 -100 10])
title('Spektrum nach mid-rise Quantisierung')
xlabel('f in Hz')
ylabel('Amplitude in dB')
```





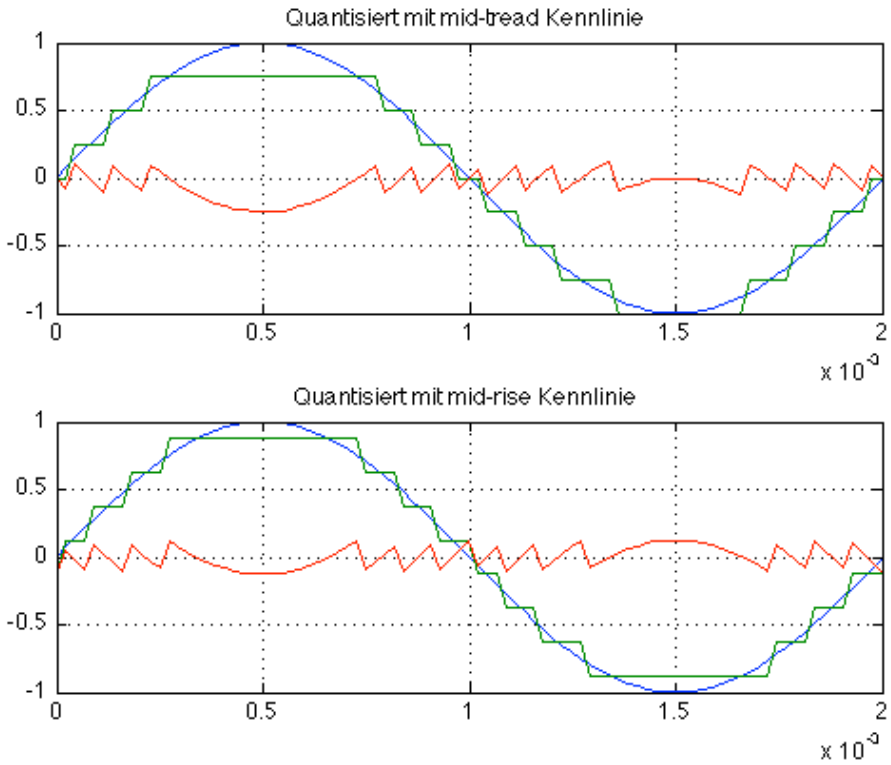
Im Zeitbereich ist zu sehen, dass die mid-rise Kennlinie keinen Quantisierungswert für die 0 aufweist, weswegen sie für die Quantisierung von Audioinhalten, bei denen Amplituden um Null öfter auftreten als andere, ungebräuchlich ist. Die mid-rise Kennlinie hingegen, hat den Nachteil, dass im positiven Amplitudenbereich eine Quantisierungsstufe weniger zu Vergügung steht, als für den negativen.

Das quantisierte Signal weist sehr starke nicht-lineare Verzerrungen auf, die als extrem störend wahrgenommen werden (Spektrum, Hörkontrolle).
Die Frequenzen unterhalb der Grundfrequenz kommen durch eine überlagerte Schwingung, hervorgerufen von der Abtastfrequenz, zustande. Es kommen $\frac{f_S}{f} = \frac{44100 \text{ samples/s}}{500 \text{ s}^{-1}} = 88{,}2 \text{ samples}$ in einer Periode des Signals vor. Die erste ganze Zahl von samples tritt nach 5 Perioden des Signals auf und beträgt 441 samples. Dies ergibt eine Frequenz von $\frac{44100 \text{ samples/s}}{441 \text{ samples}} = 100 \text{ Hz}$, die als tiefste Frequenz im Spektrum auftritt.

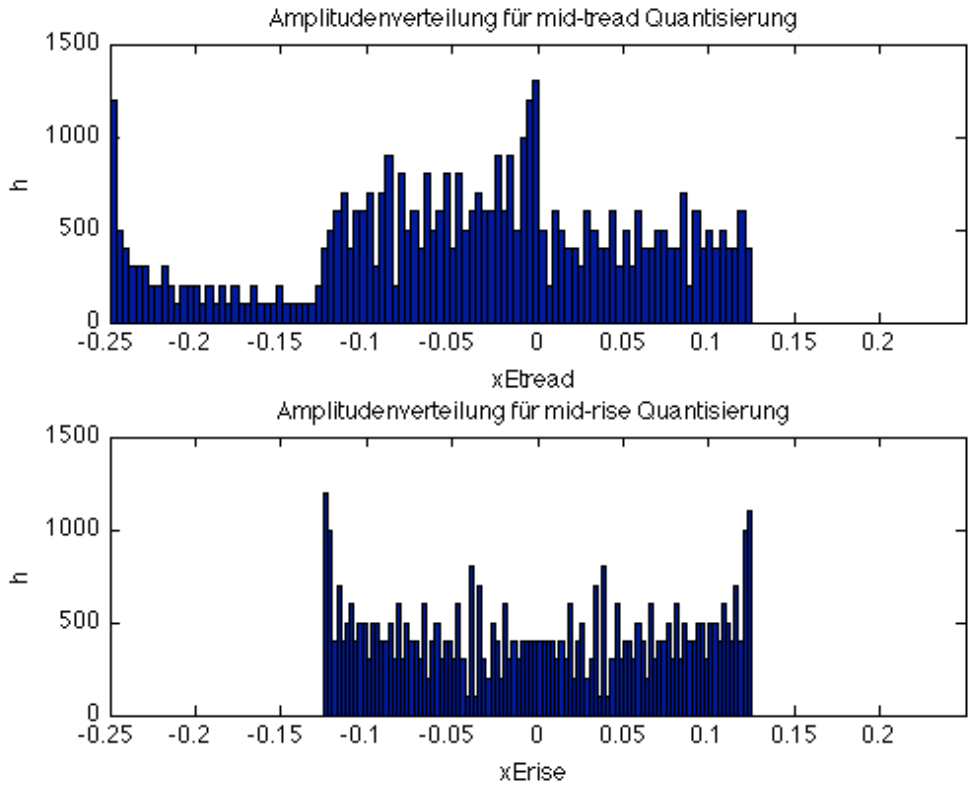
c. Erweitern sie die Funktion xquant um die Berechnung des Quantisierungsfehlers und plotten Sie dessen zeitlichen Verlauf und die Amplitudenverteilung als Histogram.

```
%% 2c)
% Erneutes Quantisieren, diesmal mit Quantisierungsfehler als
% Rueckgabeparameter
[yQtread, yEtread] = xquant(y, 3, 'mid-tread');
[yQrise, yErise]   = xquant(y, 3, 'mid-rise');
```

```
% Plot der Zeitsignale ueber eine Periode
figure(3)
subplot(2,1,1)
    plot(t(1:index), y(1:index), t(1:index), yQtread(1:index),
t(1:index), yEtread(1:index))
    title('Quantisiert mit mid-tread Kennlinie')
    grid on
subplot(2,1,2)
    plot(t(1:index), y(1:index), t(1:index), yQrise(1:index),
t(1:index), yErise(1:index))
    title('Quantisiert mit mid-rise Kennlinie')
    grid on

%% Histogram des Quantisierungsfehlers
figure(4)
subplot(2,1,1)
    hist(yEtread, 100)
    xlim([-.25 .25])
    title('Amplitudenverteilung für mid-tread Quantisierung')
    xlabel('xEtread')
    ylabel('h')
subplot(2,1,2)
    hist(yErise, 100)
    xlim([-.25 .25])
    title('Amplitudenverteilung für mid-rise Quantisierung')
    xlabel('xErise')
    ylabel('h')
```



Es wird deutlich das der rot eingezeichnete Quantisierungsfehler ein selbst ein periodisches Signal ist, wodurch die im Spektrum entstandenen Verzerrungen erklärt werden. Für die mid-tread-Quantisierung ist er an einigen Stellen zudem Größer als LSB/2, was an den bereits angesprochenen Eigenschaften der Kennlinie liegt.



Die Wahrscheinlichkeitsdichtefunktion des Quantisierungsfehlers ist nicht gleichverteilt, wie dies bei gut ausgesteuerten Quantisierern und hohen Wortbreiten der Fall ist.

3. Anhang

```
% [xQ, xE, codebook, xD] = xquant(x, nbits, method, dither,
doPlot)
%
% quantizes a single row or column vektor
%
%
% input arguments
%
% x         : input signal
% nbits     : depth
% method    : 'mid-tread' or 'midrise'.
% dither    : 'rect' (rectangular, .0 lsb peak amplitude)
%             'tri' (triangular, 1 lsb peak amplitude)
%             'none'
% doPlot    : 0 = no plot
%             1 = plot time signals
%             2 = plot frequency Response of quantized signal and
histogramm
%                 of quantzation error
%
%
% output arguments:
%
% xQ        : quantised signal
% xE        : quantisation error (xQ-x)
% codebook  : values used for quantisation
% xD        : dithered signal
%
%
%
% author: fabian brinkmann


function [xQ, xE] = xquant(x, nbits, method)


% check right input format
if size(x, 1) > size(x, 2)
    x = x';
end

% least significant bit
lsb = 2^-(nbits-1);


% ---------------------- quantize signal ---------------------- %
switch method

    % mid-tread quantisation, with a value fr zero but one quanti-
sation
    % value less for values >0
    case 'mid-tread'
```

```
        % partition defines borders used for quantisation
        partition = -1+.5*lsb:lsb:1-1.5*lsb;
        % codebook defines values used for quantisation
        codebook  = -1:lsb:1-lsb;

    % mid-rise quantisation. No value for zero but evan spacing of
values
    % around zero
    case 'mid-rise'
        partition = -1+lsb:lsb:1-lsb;
        codebook  = -1+.5*lsb:lsb:1-.5*lsb;
end

% quantise the signal using 'quantiz'
[index, xQ] = quantiz(x,partition,codebook);

% calculate quantisation error
xE = xQ - x;
```