AM- und FM-Signale

1 Übersicht

2 Aufwärmübungen

2.1 Generierung eines FM-Signals mit Matlab

2.1 a) Folgender Matlab-Code generiert ein Chirp-Signal

```
fsamp = 8000;
dt = 1/fsamp;
dur = 1.8;
tt = 0:dt:dur;
psi = 2*pi*(100+200*tt+500*tt.*tt); %
xx = real(7.7*exp(j*psi));
sound(xx,fsamp);
```

Frequenzbereich der oberen Funktion

Die Funktion $\psi(t) = 2\pi(100 + 200t + 500t^2)$ wird nach der Zeit abgeleitet und es ergibt sich mit

$$f_i(t) = \frac{1}{2\pi}\frac{d}{dt}\psi(t) = 200 + 1000t$$

Zum Zeitpunkt t=0s ist die Frequenz also f_0=200Hz was der Grundfrequenz des Signals entspricht. Da hier die Zeit bis t=1,8s läuft ist die maximale erreichte Frequenz f_{max}=2000Hz

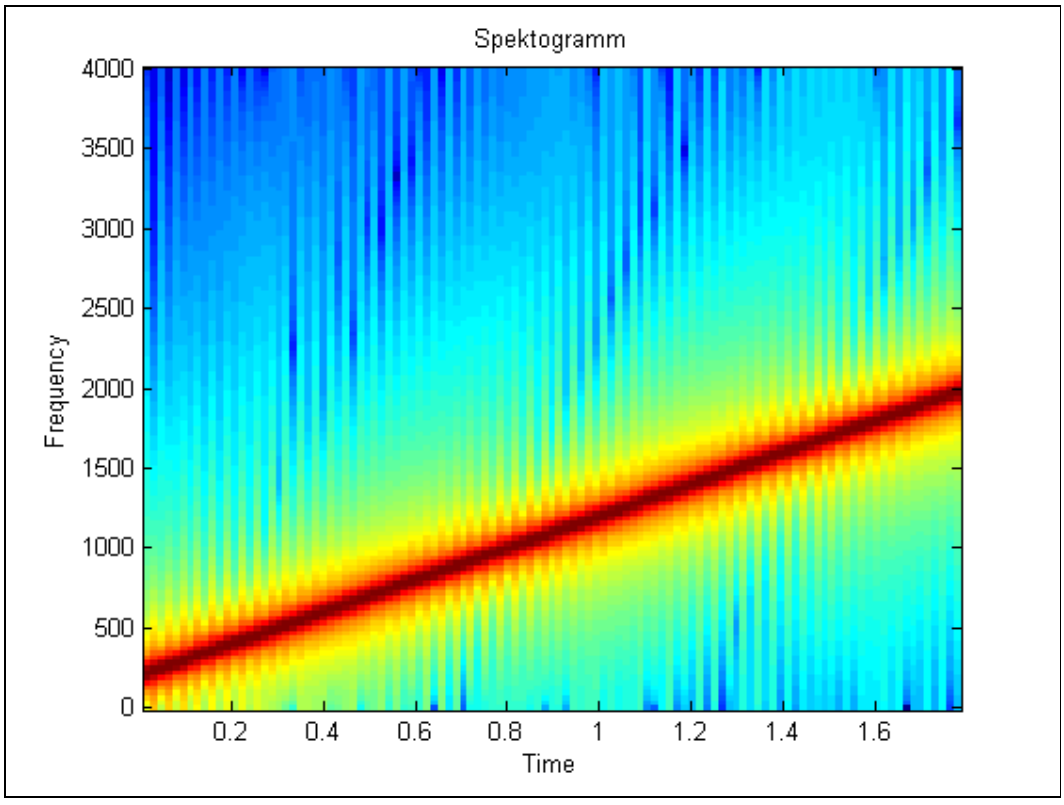
Die minimale Frequenz die man hört ist 200Hz und die Maximale 2000Hz.

2.1 b) Das Programm mychirp.m

```
function   xx = mychirp(f1,f2, dur, fsamp)
%MYCHIRP      generate a lenear-FM chirp signal
%
%  usage:   xx = mychirp(f1,f2, dur, fsamp)
%
%       f1 = starting qrequency
%       f2 = ending frequency
%      dur = total time duration
%    fsamp = sampling frequency (OPTIONAL: default is 8000)
%
if (nargin < 4)
    fsamp = 8000;
end
tt=[0:1/fsamp:dur];
ff=(f2-f1)/(2*dur);
psi=2*pi*(100+f1*tt+ff*tt.*tt);
xx=real(7.7*exp(j*psi));
sound(xx,fsamp);
specgram(xx,[],fsamp);
```

Benutzt man dieses M-File nun um das vorgegebene Chirp-Signal zu erzeugen, wird das Signal akustisch und als Spektogramm ausgegeben.

```
mychirp(200,2000,1.8)
```



3 Chirp und Beat Signale

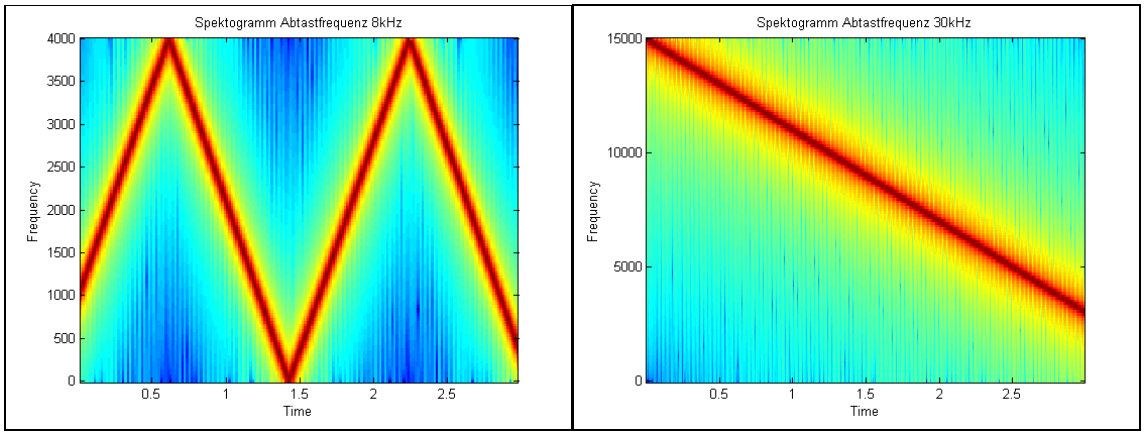
3.1 Generiere ein Chirp Signal

Der Aufruf des Programms mychirp sieht folgendermaßen aus:

```
mychirp(15000,300,3,8000)
```

Den angegebenen Frequenzen nach, würde man einen Ton erwarten der von 15kHz auf 300Hz linear fallen würde. Da die Abtastfrequenz aber nur mit 8kHz gewählt wurde, ist die höchste Frequenz die man zu hören bekommt 4kHz. Um das Signal korrekt darstellen zu können müsste man also eine Abtastfrequenz von mindestens 30kHz benutzen.
Das Audiosignal das man hört, hat einen immer wieder steigenden und fallenden Ton.

Folgend sind die Spektogramme mit der vorgegebenen Abtastfrequenz und der minimal zulässigen (nach Abtasttheorem) Abtastfrequenz gegenüber gestellt



3.2 Schwebungen

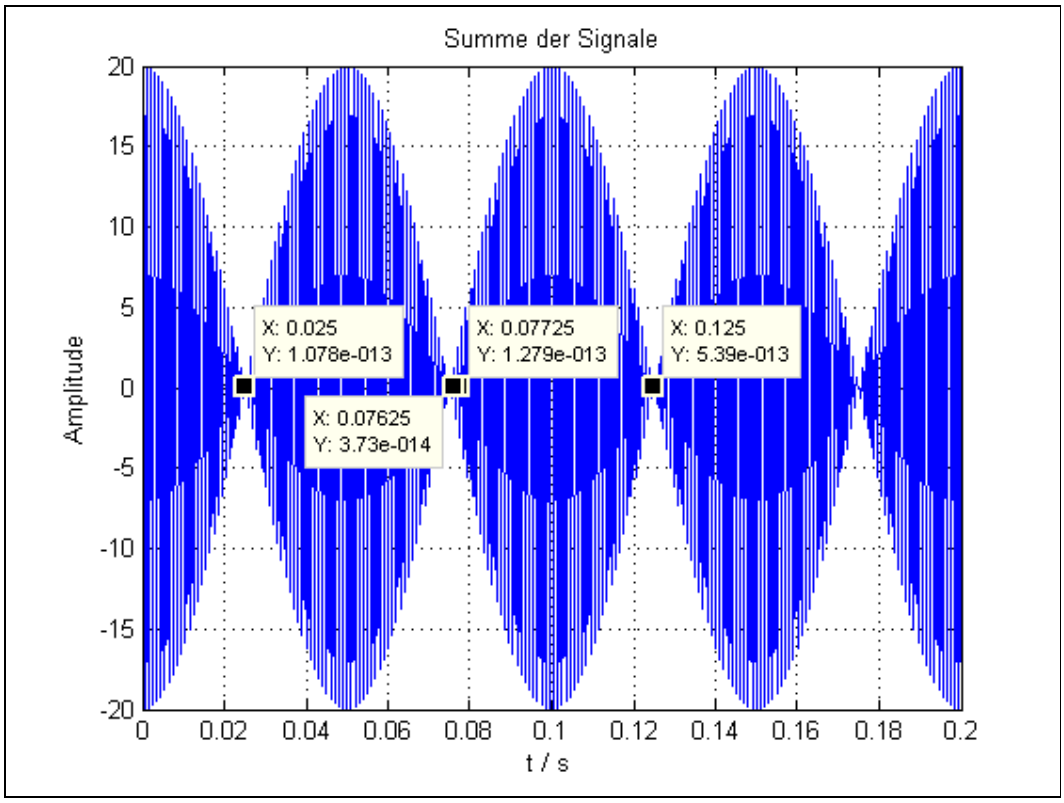
3.2 a) M-File beat.m

```
function  [xx, tt] = beat(A, B, fc, delf, fsamp, dur)
%Kommentare wurden entfernt
tt=(0:1/fsamp:dur);
xx=A*cos(2*pi*(fc-delf)*tt)+B*cos(2*pi*(fc+delf)*tt);
```

3.2 b) Schwebung

Erstellen einer Schwebung:

```
>> [xx,tt] = beat(10,10,1000,10,8000,0.2);
>> plot(tt,xx), grid on, title('Summe der Signale'), xlabel('t / s'), ylabel('Amplitude')
```



Mit Hilfe der Data-Cursors lässt sich nun leicht die Frequenz des Einhüllenden und des Hochfrequenten Signals bestimmen:

T = 0,125s - 0,025s = 0,1s
f = 10 Hz
T = 0,07725s - 0,07625s = 0,001s
f = 1000 Hz

3.2 c) Eigenschaften des Signals

Hört man sich das Signal mit f_{Δ}=10Hz an, kann man ein Signal mit f=1kHz erkennen was abwechselnd lauer und leiser wird, da sich die Amplitude wie auf obigem Plot erkennen lässt, zehn mal in der Sekunde auf 0 absenkt.

4 Frequenzmodulation von Musikinstrumenten

4.1 Generierung der Einhüllende für Glocken-Ton „bellenv.m“

```
function  yy = bellenv(tau, dur, fsamp);
%Kommentar entfernt
tt=0:1/fsamp:dur;
yy=exp(-tt/tau);
```

4.2 Parameter für den Glocken-Ton „bell.m“

```
function  xx = bell(ff, Io, tau, dur, fsamp)
%Kommentar entfernt
tt=0:1/fsamp:dur;
At=bellenv(tau, dur, fsamp);
It=Io*bellenv(tau, dur, fsamp);
xx=At.*cos(2*pi*ff(1)*tt+It.*cos(2*pi*ff(2)*tt-pi/2)-pi/2);
```

4.3 Glocken-Ton

Das Programm `bell.m` wurde mit folgenden Fällen ausgeführt.

```
>> case1 = bell([110 220], 10, 2, 6, 11025);
>> case2 = bell([220 440], 5, 2, 6, 11025);
>> case3 = bell([110 220], 10, 12, 3, 11025);
>> case4 = bell([110 220], 10, 0.3, 3, 11025);
>> case5 = bell([250 350], 5, 2, 5, 11025);
>> case6 = bell([250 350], 3, 1, 5, 11025);
```

4.3 a) Die Töne anhören

4.3 b) Fundamentalfrequenz

Ausgerechnet ist die Fundamentalfrequenz der größte gemeinsame Teiler von f_c und f_m. Dabei ist f_c die Trägerfrequenz und f_m die so genannte „modulating“ Frequenz, die die Frequenz der instantanen Frequenz beschreibt. In den Fällen 1-4 ist die Fundamentalfrequenz folglich gleich der „carrier frequency“ f_c. Bei Fall 5 und 6 ist es demnach 50Hz.

4.3 c) Beschreibung

Man hört, dass die Anzahl der beteiligten Frequenzen mit zunehmender Zeit schnell abnimmt. Dies würde der exponentiellen umhüllenden I(t) Funktion entsprechen.
Für $f_i(t)$ gilt:

$$f_i(t) = f_c - I(t) f_m \sin(f_m t + \phi_m) + \frac{dI}{dt}\cos(2\pi f_m t + \phi_m)$$

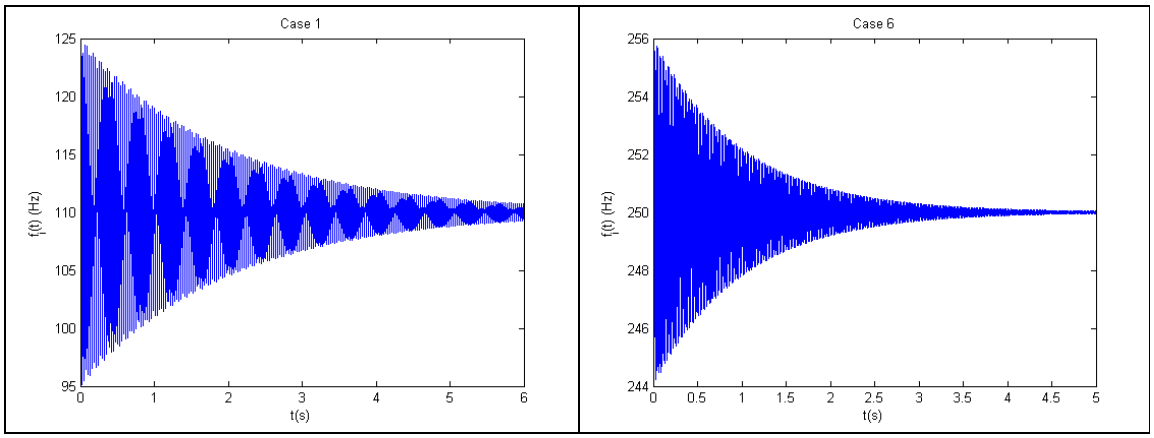
$$\frac{dI}{dt} = \frac{d}{dt} I_0 e^{-\frac{t}{\tau}}$$

$$\Rightarrow \frac{dI}{dt} = -\frac{I_0}{\tau} e^{-\frac{t}{\tau}}$$

$$f_i(t) = f_c - I_0 e^{-\frac{t}{\tau}} \sin(f_m t + \phi_m) - \frac{I_0}{\tau} e^{-\frac{t}{\tau}} \cos(2\pi f_m t + \phi_m)$$

Programm zum Plotten von $f_i(t)$

```
function xx = lab4_43c(ff, Io, tau, dur, fsamp)
t=0:1/fsamp:dur;
xx=ff(1)-(Io*exp(-t/tau).*sin(ff(2)*t-pi/2))-((Io/tau)*exp(-t/tau).*cos(2*pi*ff(2)*t-pi/2));
plot(tt, xx);
xlabel('t(s)')
ylabel('f_i(t) (Hz)')
```



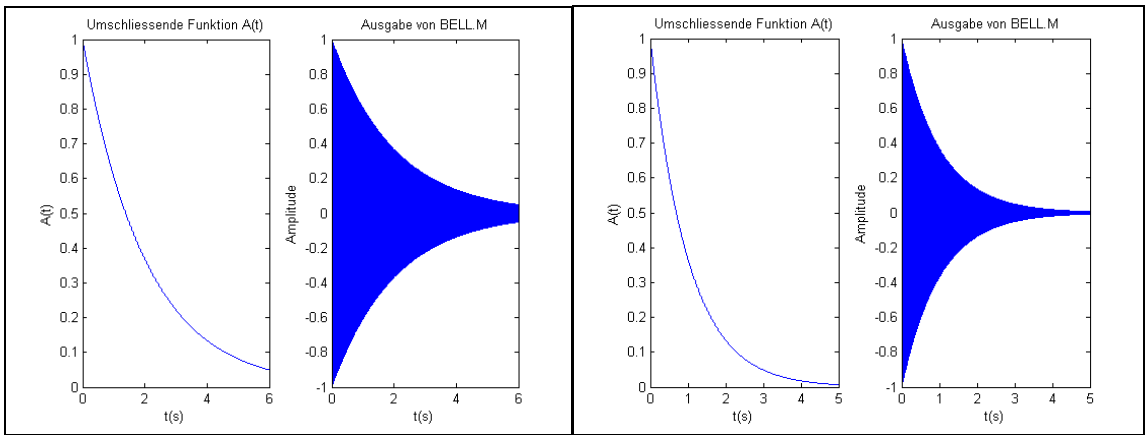
4.3 d) Spektrogramm

Diese Teilaufgabe wurde weggelassen.

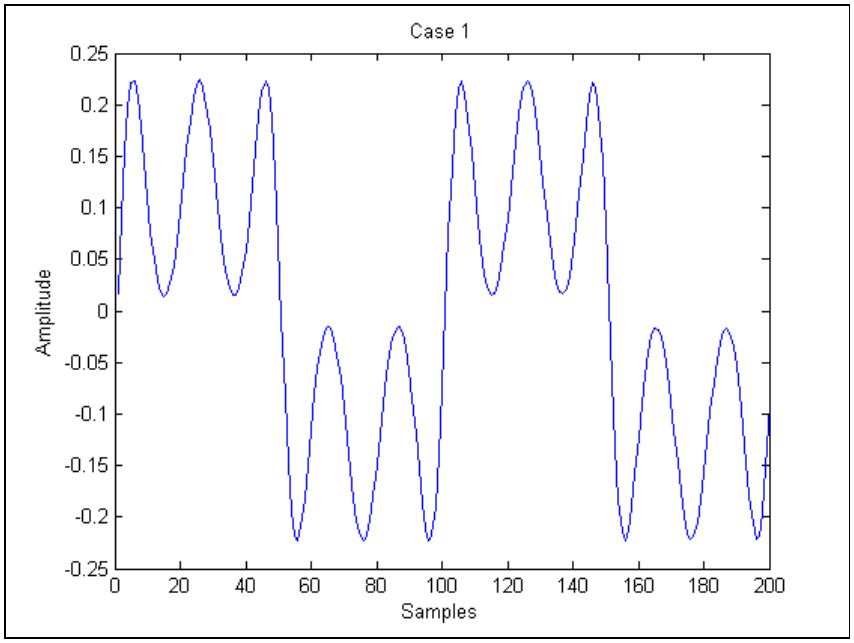
4.3 e) Signalabhängigkeit von A(t)

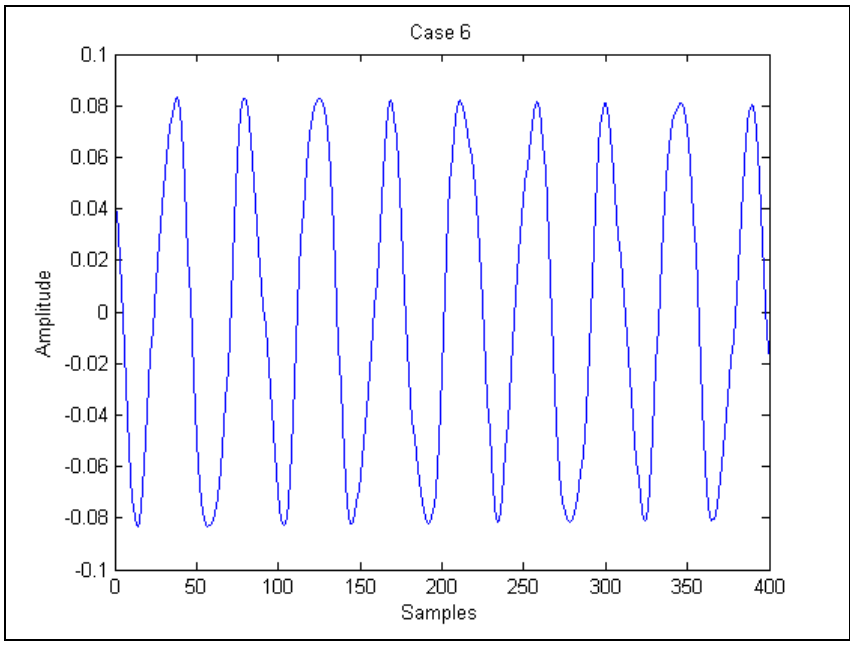
Die Funktion A(t) sieht ähnlich aus wie die I(t), außer, dass die A(t) von 1 bis nach 0 läuft und nicht wie I(t) mit dem Faktor 10 bzw. 3 multipliziert wird.

Vergleicht man also die Funktion A(t) und den „Bell Sound“, so kann man sagen, dass die Funktion A(t) die Amplitude des „Bell Sound“ bestimmt.



4.3 f) **Plot-Ausschnitt**





In beiden Fällen ändert sich die Frequenz in dem Bereich kontinuierlich. Die Änderung der Frequenz scheint Periodisch zu sein.

4.4 Kommentare zu den Glockentönen

Verhältnis 1:2 :

Fundamentale Frequenz: 100 Hz
```
>> sound(bell([100,200], 8, 5, 6, 11025), 11025);
```
Fundamentale Frequenz: 250 Hz
```
>> sound(bell([250,500], 8, 3, 6, 11025), 11025);
```
Fundamentale Frequenz: 500 Hz
```
>> sound(bell([500,1000], 8, 1, 6, 11025), 11025);
```

Mit zunehmender Frequenz wird der Glockenton und die zusätzlichen überlagernden Frequenzen höher.

Wird die Abfallrate τ kleiner gewählt, verschwinden die zusätzlichen Frequenzen schneller und der Grundton der Glocke klingt früher alleine.

Verhältnis 5:7 :

Fundamentale Frequenz: 20 Hz
```
>> sound(bell([100,140], 8, 2, 6, 11025), 11025);
```
Fundamentale Frequenz: 10 Hz
```
>> sound(bell([50,70], 8, 3, 6, 11025), 11025);
```
Fundamentale Frequenz: 50 Hz
```
>> sound(bell([250,350], 8, 3, 6, 11025), 11025);
```