Algorithmen in Akustik und Computermusik

Skriptum zur Übung

letzte Änderung: 19. Oktober 2019

Institut für Elektronische Musik und Akustik

Piotr Majdak
piotr@majdak.com

Inhaltsverzeichnis

1. Generierung von Signalen

1.1. Mathematische Zusammenhänge

Anzahl der Samples und Signallänge: $N = T \cdot f_s$

Annahme: periodische Schwingung: $s_i = A_i \cos(\Theta_i)$

mit: s_i ... Signal
Θ_i ... Phase
A_i ... Amplitude

Anzahl der Perioden der Schwingung: $M = N \cdot \frac{f}{f_s} = T \cdot f$

mit: N ... Anzahl der Samples
M ... Anzahl der Perioden im Signal
T ... Länge des Signals in s
f_s ... Sampling-Frequenz in Hz
f ... Grundfrequenz des Signals bei harmonischen Schwingungen in Hz

Ausdehnung und Auflösung des Signals:

Phase für eine Periode: $\Theta_1 = [0, 2\pi)$

Phase für M Perioden: $\Theta = [0, 2\pi \cdot M) = [0, 2\pi \cdot T \cdot f) = [0, 2\pi \cdot N \cdot \frac{f}{f_s})$

Auflösung der Phase: $d\Theta = \frac{2\pi M}{N} = 2\pi \cdot \frac{f}{f_s}$

1.2. Signalgenerierung

- Es sollen 2 Vektoren generiert werden, die folgende Signale beinhalten:

s_1: Cosinus-Signal, $f = 1000\text{Hz}$
s_2: Überlagerung von 2 Cosinus-Signalen, $f_1 = 1000\text{Hz}$, $f_2 = 4410\text{Hz}$

- Die Signale sollten die Länge von 2.268ms bei einer Sampling Frequenz von $f_s = 44100\text{Hz}$ haben. Die Amplituden der einzelnen Signale sollen $A_i = 1$ betragen.
- Beide Signale sollten gemeinsam in einem Zeit-Diagramm unterschiedlich gefärbt dargestellt werden, wobei das Diagramm korrekt beschriftet werden sollte.

Setzen der Konstanten

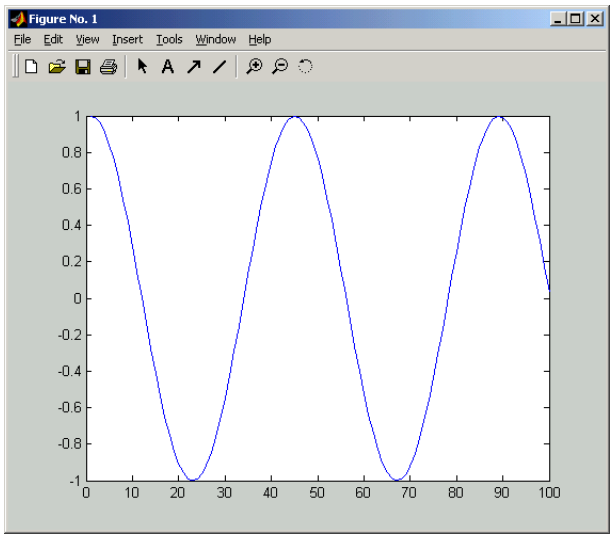
```
>> fs=44100

fs =

       44100

>> f=1000;

>> N=2.268e-3*fs

N =

  100.0188  --> falsch!

>> N=round(2.268e-3*fs)

N =

   100      --> 100 Samples: richtig!
```

Berechnung der Hilfsgrößen

```
>> dTH=2*pi*f/fs;
>> TH1=0:dTH:2*pi*N*f/fs-dTH;
```

Berechnung des Signals

```
>> s1=cos(TH1);
>> plot(s1);
```

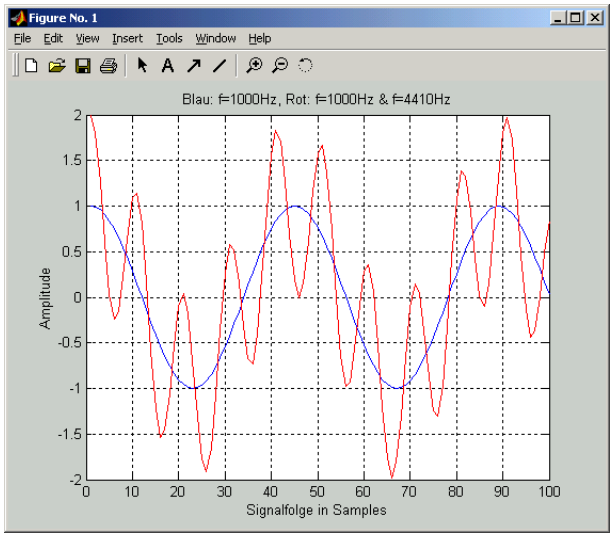


Berechnung des zweiten Signals

```
>> f=4410;
```

```
>> dTH=2*pi*f/fs;
>> TH2=0:dTH:2*pi*N*f/fs-dTH;

>> s2=s1+cos(TH2);
```

Darstellung




```
>> hold on
Current plot held
>> plot(s2,'r')
>> title('Blau: f=1000Hz, Rot: f=1000Hz & f=4410Hz');
>> xlabel('Signalfolge in Samples');
>> ylabel('Amplitude');
>> grid;
```

1.3. Beispiel

- Auf der Grundlage der ersten Übung sollen 2 weitere Vektoren generiert werden:

s_3: Überlagerung von 3 Cosinus-Signalen, $f_1 = 1000\text{Hz}$, $f_2 = 4410\text{Hz}$, $f_3 = 13000\text{Hz}$, $\varphi_3 = 45°$, $A_3 = 0.5$

s_4: Überlagerung eines Cosinus-Signales $f = 1000\text{Hz}$ mit einem Rauschsignal. Die Werte des Rauschsignals sollten im Intervall $[-0.25, 0.25]$ liegen.

- Die Signale sollten die Länge von 2.268ms bei einer Sampling Frequenz von $f_s = 44100\text{Hz}$ haben. Wenn nicht anders angegeben sollten die Amplituden der einzelnen Signale $A_i = 1$ sein.
- Alle 4 Signale sollten gemeinsam in einem Zeit-Diagramm unterschiedlich gefärbt dargestellt werden, wobei das Diagramm korrekt beschriftet werden sollte.
- Die Signale sollen gespeichert werden – sie werden in weiteren Übungen benötigt.

Fragen:
- Berücksichtigung der zusätzlichen Phase?
- Veränderung der Amplitude?

- Erzeugung von Rauschen?

```
>> help rand

 RAND   Uniformly distributed random numbers.
    RAND(N) is an N-by-N matrix with random entries, chosen from
    a uniform distribution on the interval (0.0,1.0).
    RAND(M,N) and RAND([M,N]) are M-by-N matrices with random entries.
...
>> rand(10,1)

ans =

    0.9501
    0.2311
    0.6068
    0.4860
    0.8913
    0.7621
    0.4565
    0.0185
    0.8214
    0.4447
```

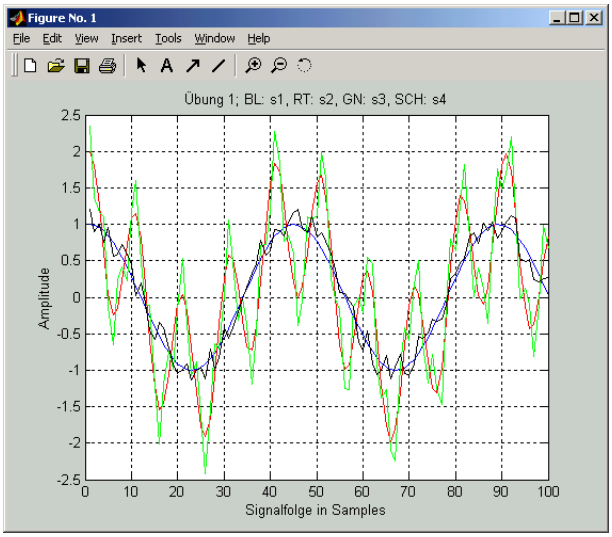
- Signale speichern

```
>> save filename var1
```

1.4. Übersicht der verwendeten Befehle

`help xxx`	Hilfe über xxx
`st:res:en`	Erzeugt ein Vektor von `st` bis `en` mit der Schrittweite `res`
`cos`	Cosinusfunktion (in radiant)
`plot`	Zeichnet ein 2-D-Diagramm
`rand`	Zufallsgenerator, liefert Werte im Intervall `[0,1]`
`hold [on|off]`	Schaltet die Beibehaltung der Plots in einem Diagramm (`figure`) ein/aus
`title`	Beschriftung eines Diagramms
`xlabel, ylabel`	Beschriftung der X-/Y-Achsen
`grid`	Schaltet Gitter in einem Diagramm ein/aus
`save`	Speichert angegebene Variablen in einer Datei

1.5. Eine (mögliche) Lösung



2. Analyse der Signale

2.1. Mathematische Zusammenhänge

Diskrete Fourier Transformation: $X_p(k) = \sum_{n=0}^{N-1} x_p(n) \cdot e^{-j(2\pi/N) \cdot n \cdot k}$

mit: $X_p(k)$ ... Transformationskoeffizienten, Representation im Frequenzbereich
$x_p(n)$ ... Eingangssignal
N ... Länge des Signals (in Samples), gleichzeitig Anzahl der Transformationskoeffizienten (bins)

2.2. Darstellung und Analyse

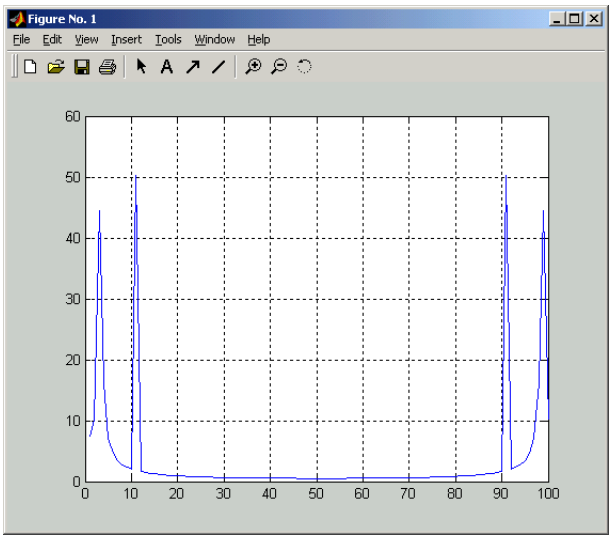
- Das in der Übung 1 generierte Signal s_2 sollte im Frequenzbereich dargestellt und diskutiert werden.

Laden der Signale:

```
>> load cossignals
```

Berechnung der DFT:

```
>> f2=fft(s2);
>> plot(abs(f2));
>> grid;
```

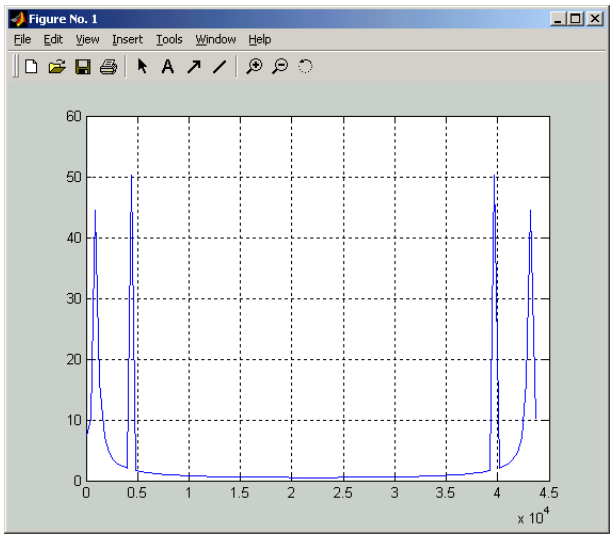


Auflösung im Frequenzbereich: $\Delta f = \frac{f_s}{N}$

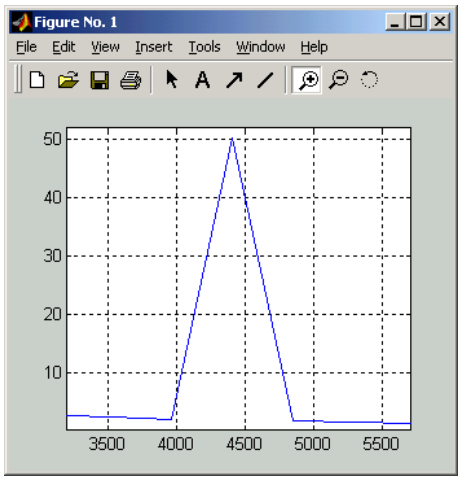
mit: Δf ... Auflösung im Frequenzbereich (in Hz)
f_s ... Sampling Frequenz (in Hz)
N ... Länge des Signals (in Samples), gleichzeitig Anzahl der Transformationskoeffizienten (bins)

Korrekte Darstellung der X-Achse:

```
>> fax=0:fs/N:fs-fs/N;
>> plot(fax,abs(f2));
>> grid;
```

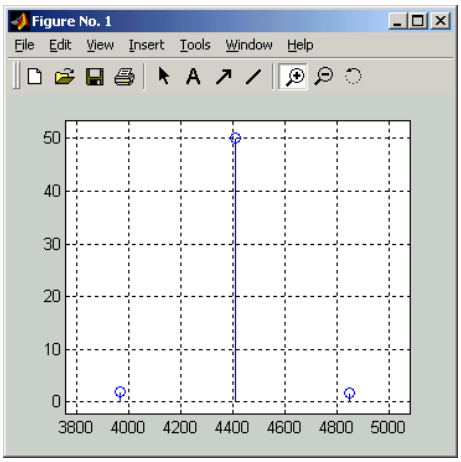


Untersuchung des Peaks mit höherer Frequenz:

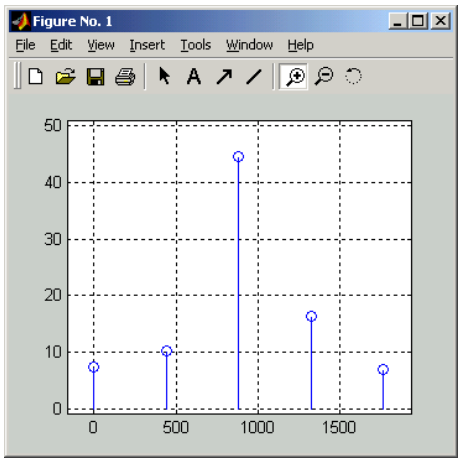


Abhilfe zur korrekten Darstellung:

```
>> stem(fax,abs(f2))
>> grid
```

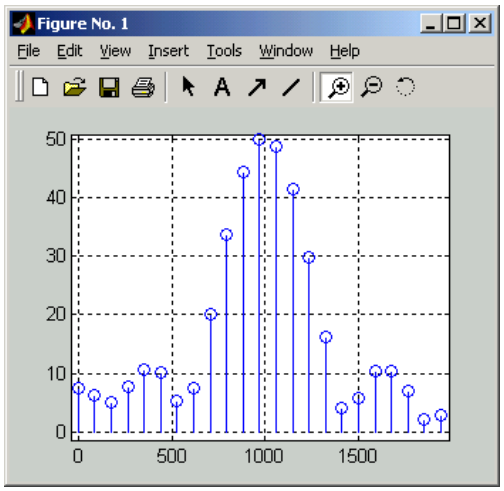


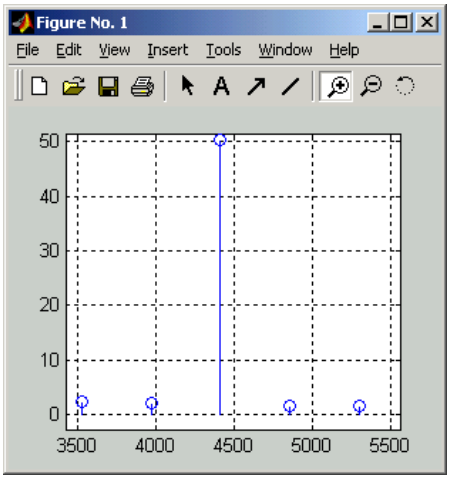
Untersuchung des zweiten Peaks:



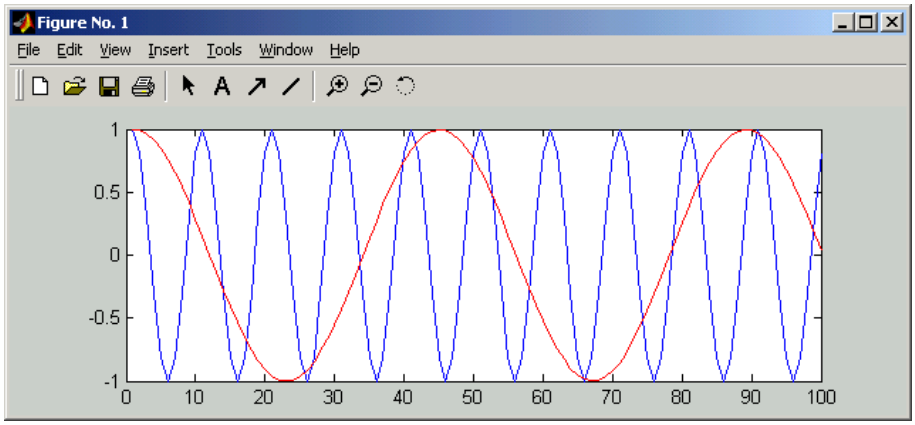
Genauere Untersuchung - Erhöhung der Auflösung:

```
>> fax5=0:fs/N/5:fs-fs/N/5;
>> f25=fft(s2,5*N);
>> stem(fax5, abs(f25));
>> grid
```



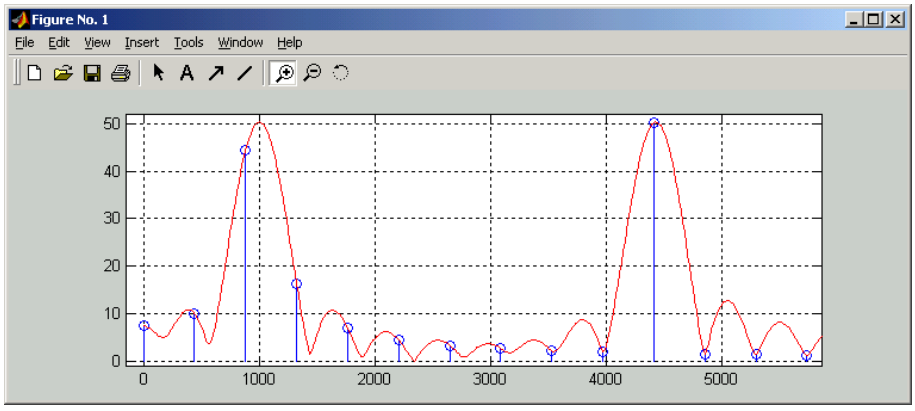


Fensterung (windowing) im Zeitbereich [hier: mit Rechteckfenster]...



...ist Faltung im Frequenzbereich [hier: mit SINC-Funktion]:

```
>> stem(fax,abs(f2))
>> hold
>> fax30=0:fs/N/30:fs-fs/N/30;
>> f230=fft(s2,30*N);
>> plot(fax30,abs(f230),'r');
>> grid
```

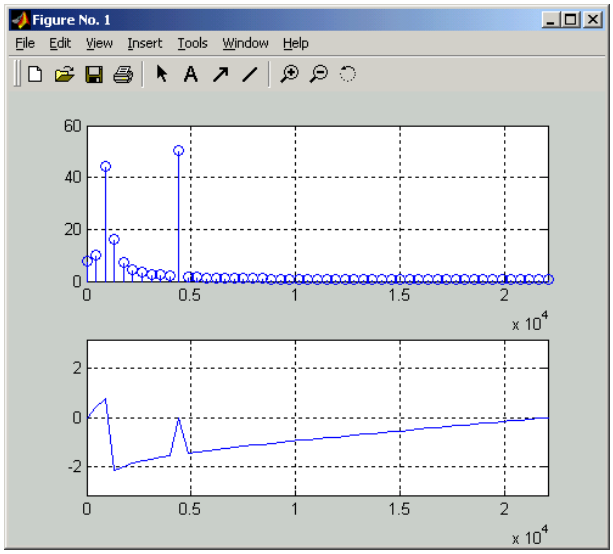


Fensterung eines Signals: $y(n)=x(n)\cdot w(n) \;\Leftrightarrow\; Y_p(\Theta)=X_p(\Theta)\otimes W_p(\Theta)$

mit: $y(n), Y_p(\Theta)$ ... Ausschnitt des Signals
$x(n), X_p(\Theta)$ ... das ursprüngliche Signal
$w(n), W_p(\Theta)$ ... Fensterfunktion

Darstellung des gesamten Frequenzganges:

```
>> subplot(2,1,1);
>> stem(fax,abs(f2));
>> axis([0 22050 0 60])
>> grid;
>> subplot(2,1,2);
>> plot(fax,unwrap(angle(f2)));
>> axis([0 22050 -pi pi])
>> grid;
```



- Erkärung des Phasengangs?

2.3. Beispiel

- Der Frequenzgang des Signals s_3 sollte in einem Diagramm dargestellt und beschriftet werden. Weiters sollten die Phasen und die Amplituden der Signale aus dem Frequenzgang abgelesen werden.
- Die Signale sollen mit verschiedenen Fenstern (Rechteck, Hann, Hamming, Dreieck, Kaiser) analysiert werden. Dabei soll die Breite der Hauptkeule und die Höhe der Nebenkeulen verglichen werden.

Hinweise:

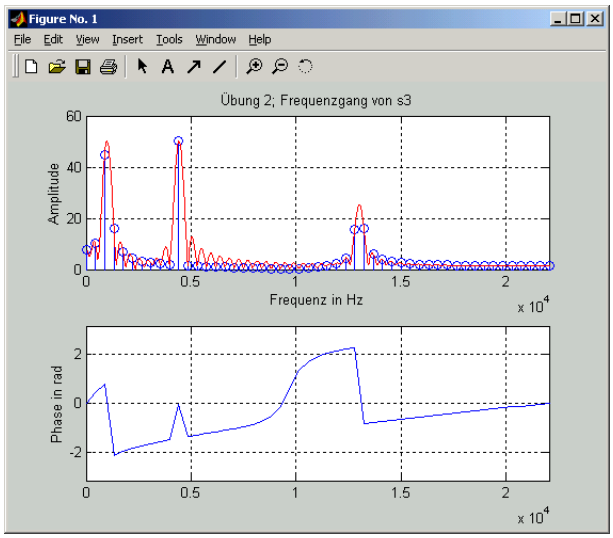
- Fensterfunktionen:
```
rectwin, hann, kaiser(N,0.8), triang
```

- Operationen „Element für Element“:
```
a .* b
a ./ b
a .^ b
```

- Matrix transponieren:
```
b = a.';   a=[1x100], b=[100x1]
```

- Frequenzgang anzeigen (http://piotr.majdak.com/download/mlib/index.php):
```
spect(fft(x), fs, flags);
```

2.4. Eine (mögliche) Lösung



3. Systemidentifikation

3.1. Impulsanregung

Anregungssignal:

- Ein Einheitsimpuls:
  - Amplitude: 1
  - Länge: 1 Sample (+ digitale Stille für die Systemantwort)

Vorteile:

- sehr einfache Messung
- Messung „as is“; keine weiteren Einflüsse

Nachteile:

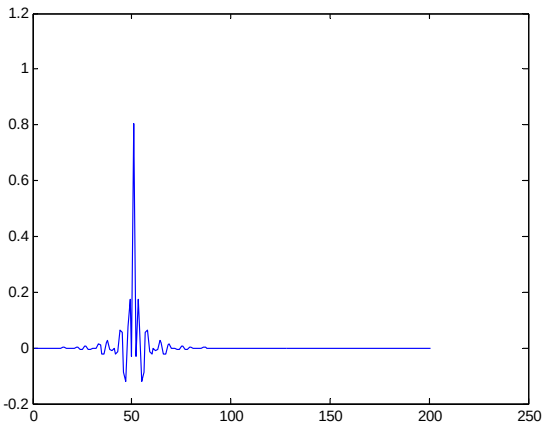
- hoher Crest-Faktor; wenig Energie im Anregungssignal
- schlechte Immunität gegenüber transienten Störungen

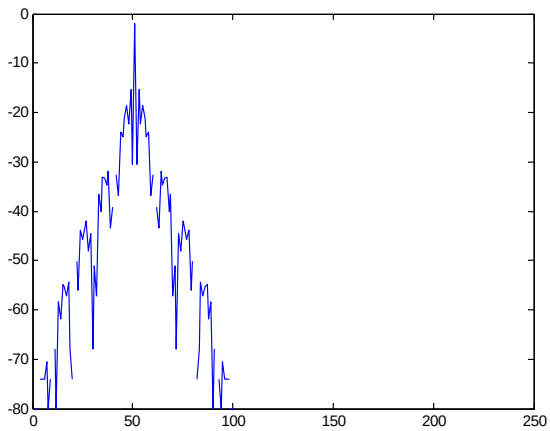
Erzeugung eines Impulses:

```
imp=[1; zeros(200,1)];
```

Anregung eines Systems und Ermittlung der Impulsantwort:

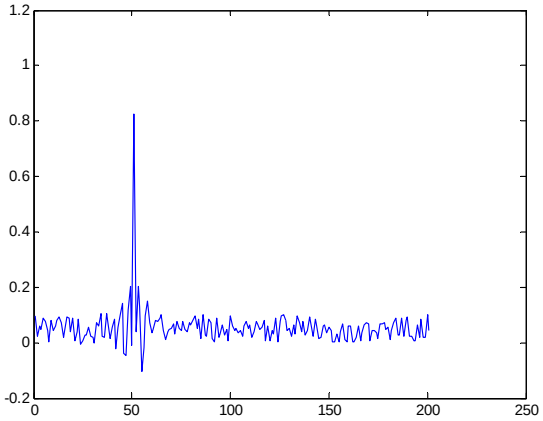
```
out=afilter(imp);
plot(out);
plot(etc(out));
```

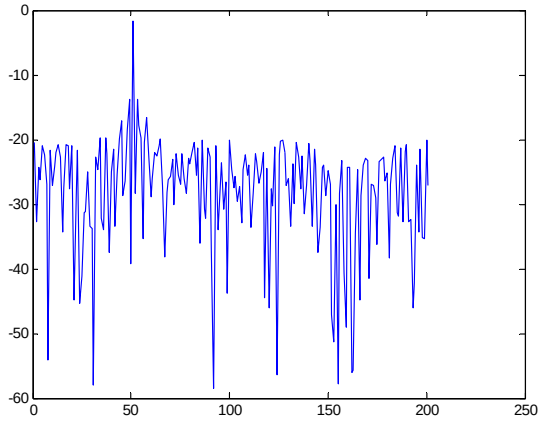




Systemidentifikation unter realistischen Bedingungen (mit Störrauschen):

```
out=afilter(imp,-20);     % -20dB Rauschen
```





3.2. Maximum Length Sequence

Anregungssignal:

- binäre Pseudozufallsfolge mit einer maximalen Periode
  - Amplitude: 1
  - Länge: $2^N - 1$ Samples (+ digitale Stille für die Systemantwort) mit N...Ordnung der MLS

Vorteile:

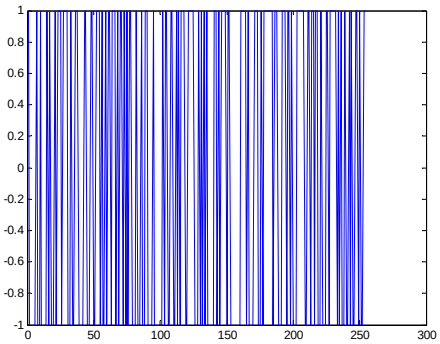
- sehr hohe Energie im Anregungssignal; sehr gute Rauschunterdrückung
- robust gegenüber transienten Störungen
- robust gegenüber leichten Abweichungen im System während der Messung

Nachteile:

- Empfindlich gegenüber nichtlinearen Verzerrungen im System

Erzeugung von MLS:

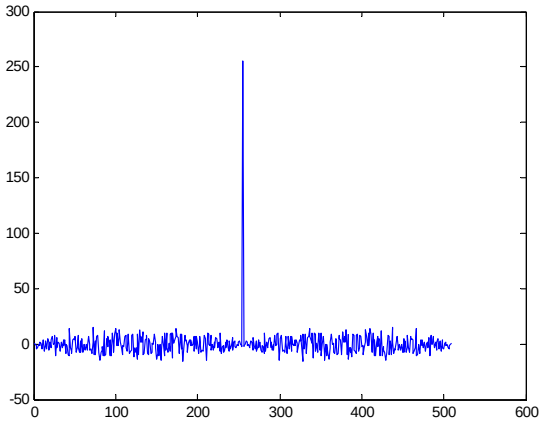
```
m=mls(8);
```



Berechnung der Impulsantwort, allgemein:

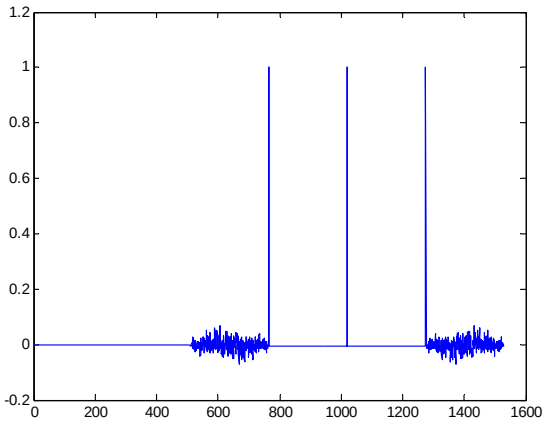
Autokorrelation von MLS: $r_{xx}(n) = \delta(n) - \frac{1}{L+1}$

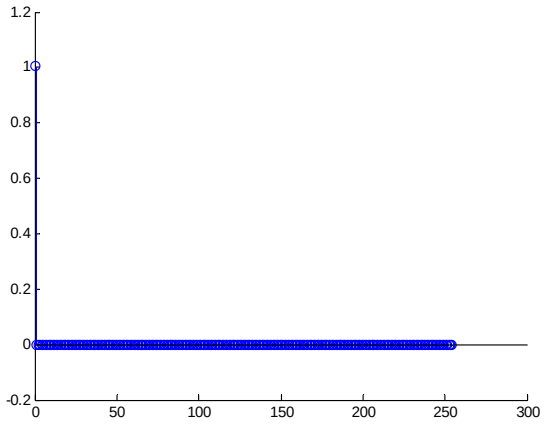
```
d=xcorr(m, m);
```



Problem: fehlende Überlappung der Signale bei der Berechnung der Korrelation.
Abhilfe: MLS öfters wiederholen

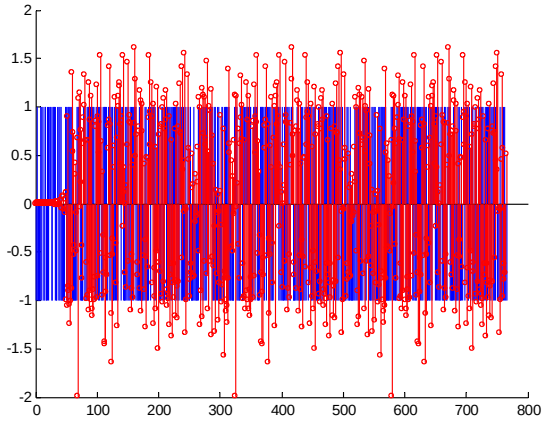
```
d=xcorr([m;m;m], m)/length(m);   % Normalisierung gilt nur für MLS
stem(d(4*length(m):5*length(m)-1));
```





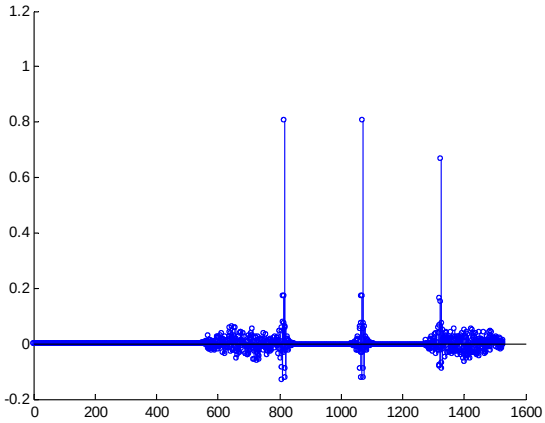
Anregung eines Systems:

```
y=afilter([m; m; m]);
```

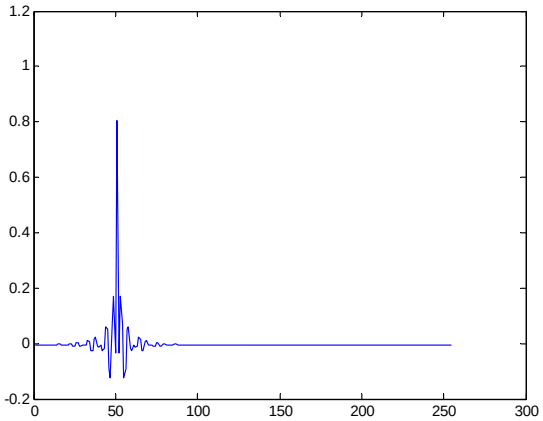


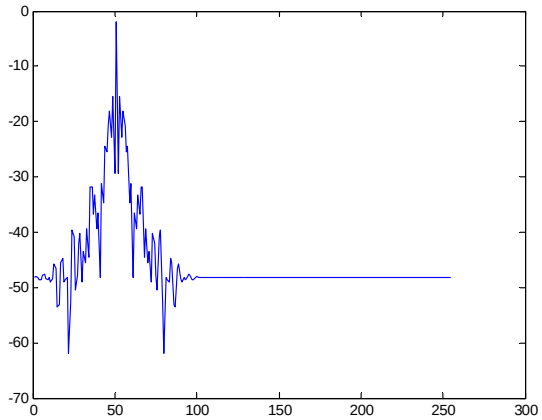
Berechnung der Impulsantwort:

```
hr=xcorr(y, m)/length(m);
```



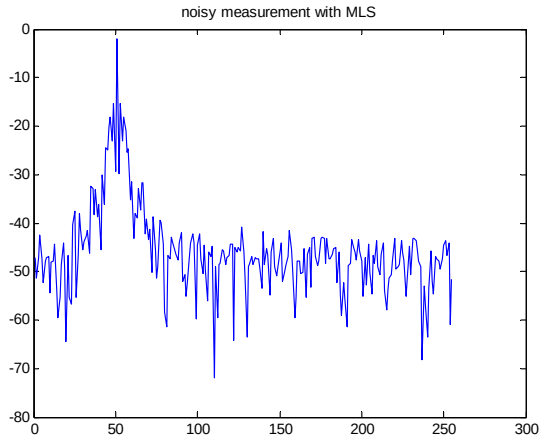

```
hr=hr(4*length(m):5*length(m)-1);
plot(hr);
plot(etc(hr));
```

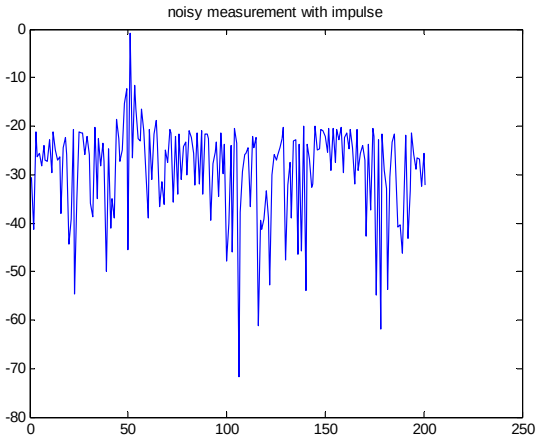




Systemidentifikation unter realistischen Bedingungen (mit Störrauschen):

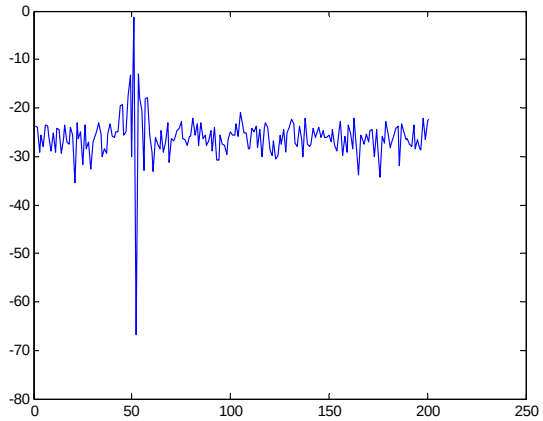
```
yn=afilter([m; m; m],-20);
hn=xcorr(yn, m)/length(m);
hn=hn((4*length(m):5*length(m)-1));
plot(etc(hn)); title('noisy measurement with MLS');
plot(etc(out)); title('noisy measurement with impulse');
```





Fairer Vergleich: (gleiche Messdauer – PIE mit 4 Wiederholungen)

```
for ii=1:4
  o(:,ii)=afilter(imp,-20);
end
plot(etc(mean(o')));
```



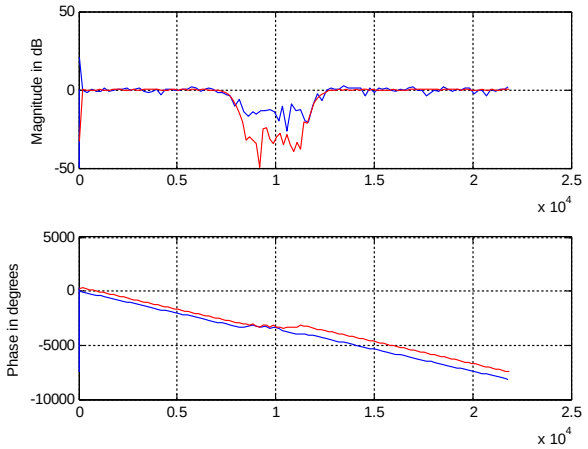
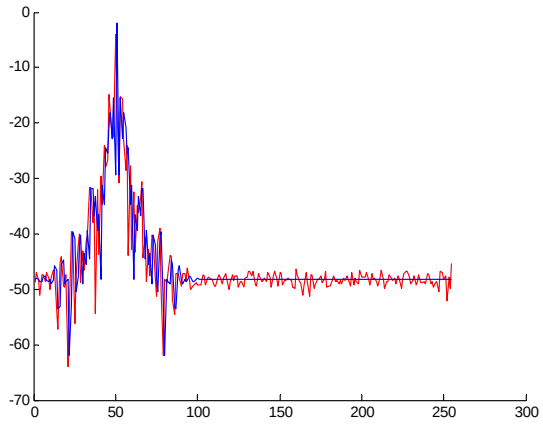
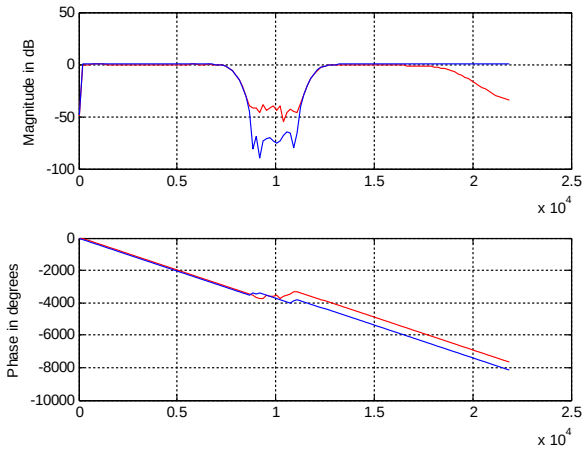


Illustration 1: blau: PIE, rot: MLS

Systemidentifikation eines leicht nichtlinearen Systems:

```
ynl=afilter([m;m;m],[],-20);
hnl=xcorr(ynl, m)/length(m);
hnl=hnl(4*length(m):5*length(m)-1);
plot(etc(hnl),'r'); hold on;
plot(etc(hr));
```





3.3. Exponentielle Sweeps

Anregungssignal:

- frequenzmodulierter Träger: $x(t)=\sin\left[A\left(e^{t/\tau}-1\right)\right]$ mit $A=\frac{T\,\omega_1}{\ln\left(\omega_2/\omega_1\right)}$,

$$\tau=\frac{T}{\ln\left(\omega_2/\omega_1\right)}$$

  - ω_1 : Startfrequenz in Hz
  - ω_2 : Endfrequenz in Hz
  - T : Signaldauer in s

Vorteile:

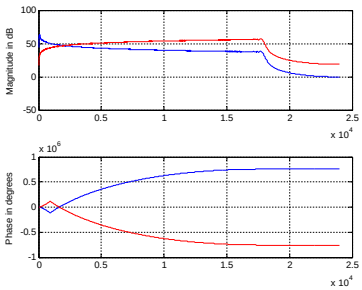
- Einfach zu erzeugendes Anregungssignal
- Hohe Energie im Anregungssignal
- Trennung der Impulsantworten der nichtlinearen Systemteile
- Frequenzbereich frei wählbar

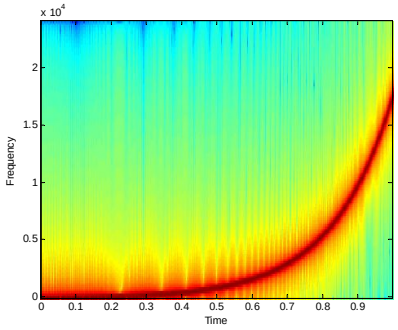
Nachteile:

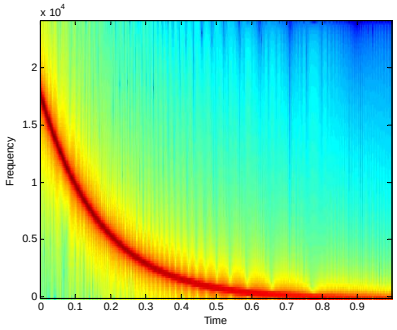
- Empfindlich gegenüber transienten Störungen
- Bandbegrenzte Messung
- Nicht robust bei leicht zeitvarianten Systemen

Erzeugung des Sweeps:

```
[sw, isw] = expsweep(50,18000,1,48000);
spect(fft(sw),48000);

spectrogram(sw,256,128,256,48000,'yaxis');
        % bis Matlab 2017: specgram(sw,256,48000);
spectrogram(isw,256,128,256,48000,'yaxis');
```







Berechnung der Impulsantwort, allgemein:

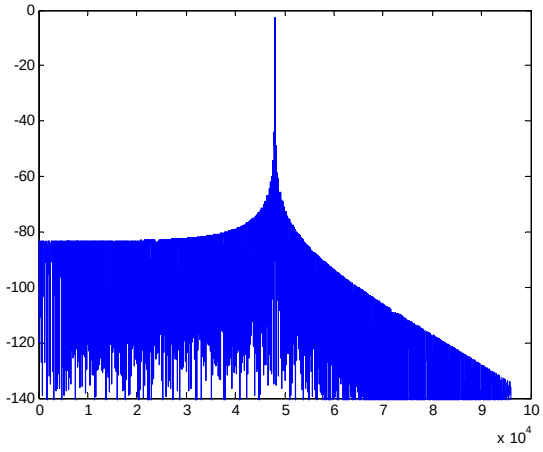
$Y(f)=X(f)\cdot H(f) \rightarrow H(f)=Y(f)\cdot X^{-1}(f)$ mit

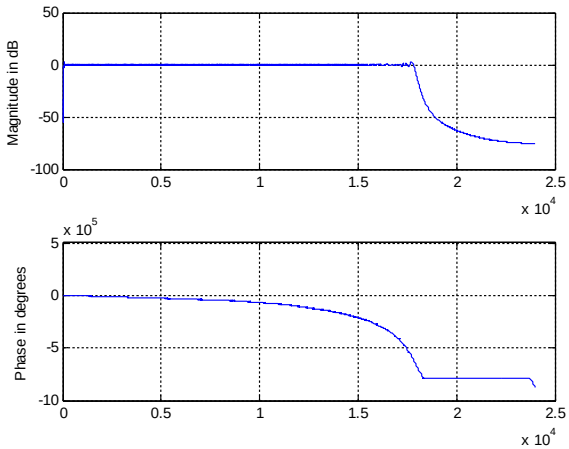
- $X(f)$: Spektrum des Anregungssignals
- $Y(f)$: Systemantwort
- $H(f)$: Übertragungsfunktion des Systems
- $X^{-1}(f)$: Inverses Spektrum des Anregungssignals

$$X^{-1}(f)=\frac{1}{|X(f)|}\cdot\frac{\mathcal{F}\{x(-t)\}}{|X(f)|}=\frac{\mathcal{F}\{x(-t)\}}{X(f)^2}$$

$X(f)\cdot X^{-1}(f)$:

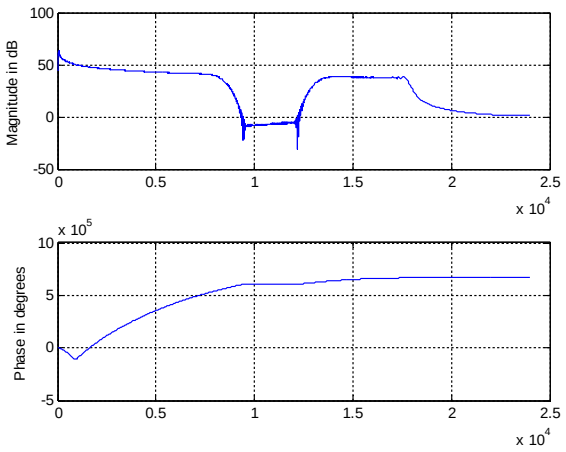
```
nges=length(sw)+length(isw)-1;
d=real(ifft( fft(sw,nges).*fft(isw,nges) ));
```

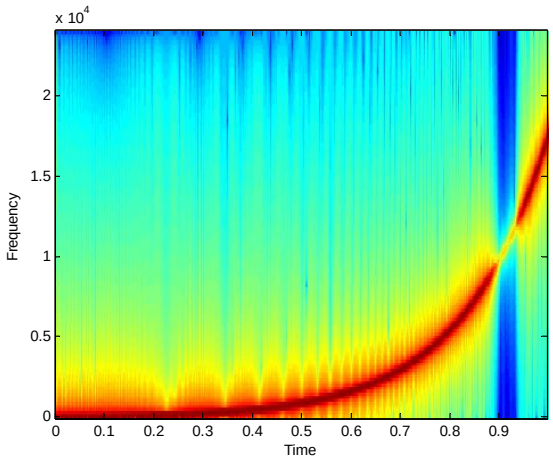




Anregung eines Systems:

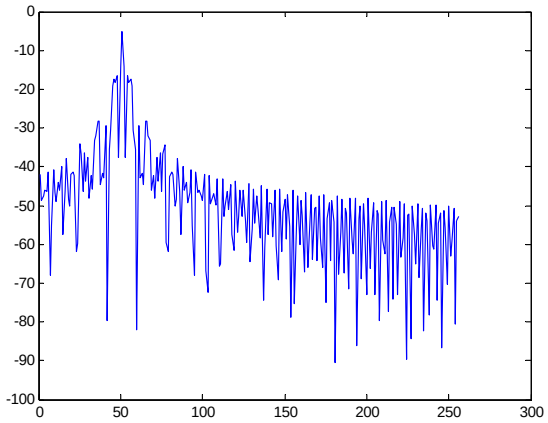
```
trail=255;
y=afilter([sw; zeros(trail,1)]);
spectrogram(y,256,128,256,48000,'yaxis');
```

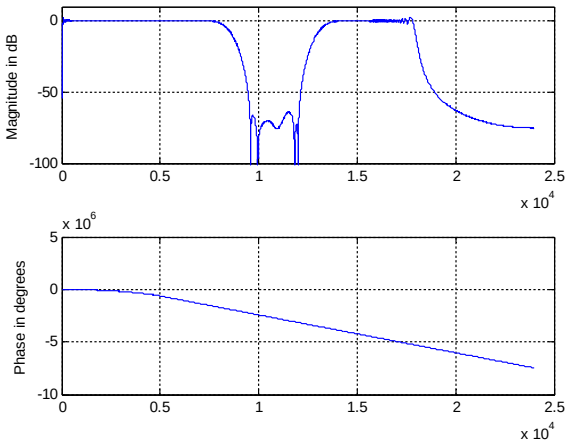




Berechnung der Impulsantwort:

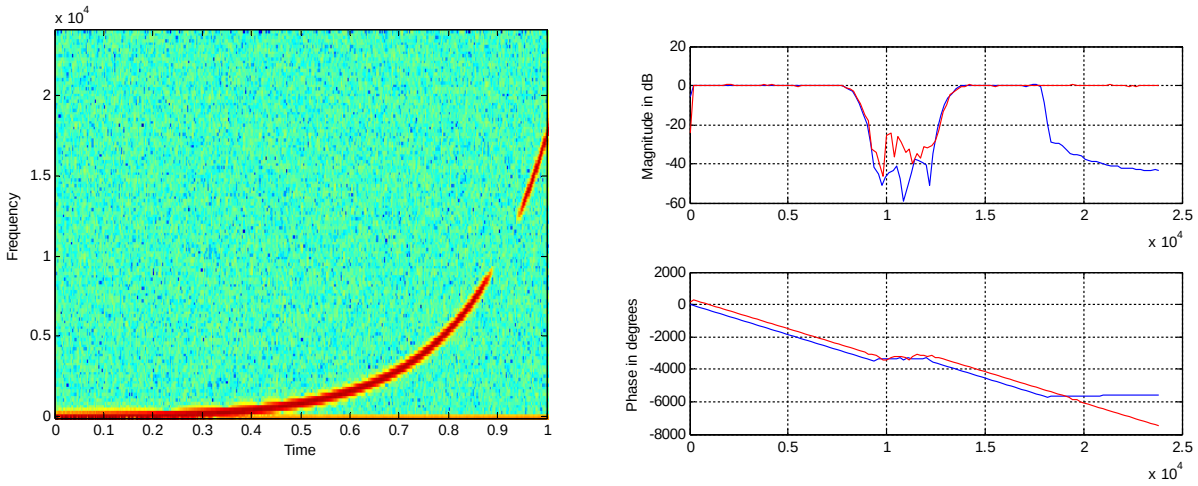
```
nges=length(sw)+length(isw)+trail-1;
h=real(ifft(fft(y,nges).*fft(isw,nges)));
h=h(length(sw):length(sw)+255);
```





Systemidentifikation unter realistischen Bedingungen (mit Störrauschen):

```
yn=afilter([sw; zeros(trail,1)],-20);
hn=real(ifft(fft(yn,nges).*fft(isw,nges)));
hn=hn(length(sw):length(sw)+255);
```



Blau: Log Sweep; Rot: MLS (255 Samples)

Fairer Vergleich: gleiche Messdauer!

- Sweep: 1 Sekunde = 48000 Samples
- MLS: 3 Wiederholungen á 16384 Samples = 49152 Samples → MLS 14-ter Ordnung

```
m=mls(14);
yn=afilter([m; m; m],-20);
hn=xcorr(yn, m)/length(m); % raw IR
hn=hn((4*length(m):5*length(m)-1));
hn=hn(1:255);
```

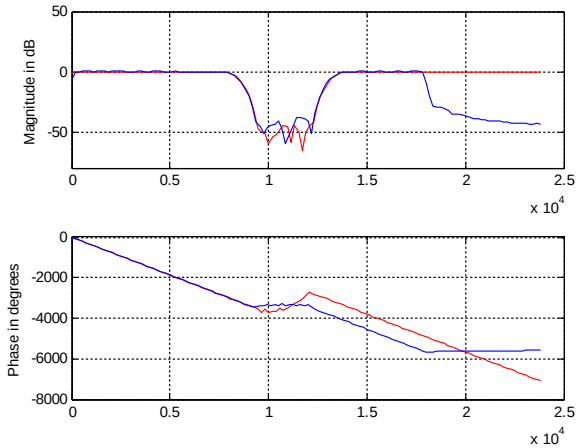
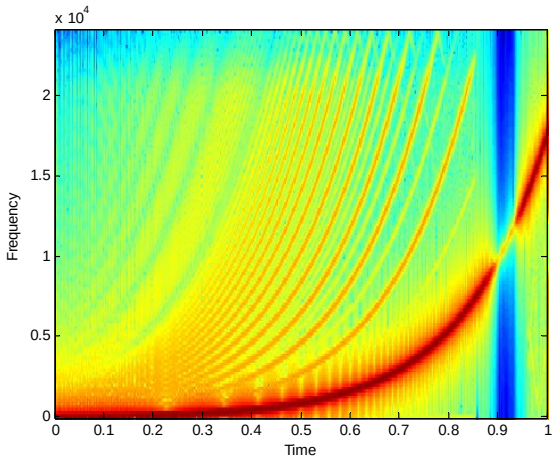


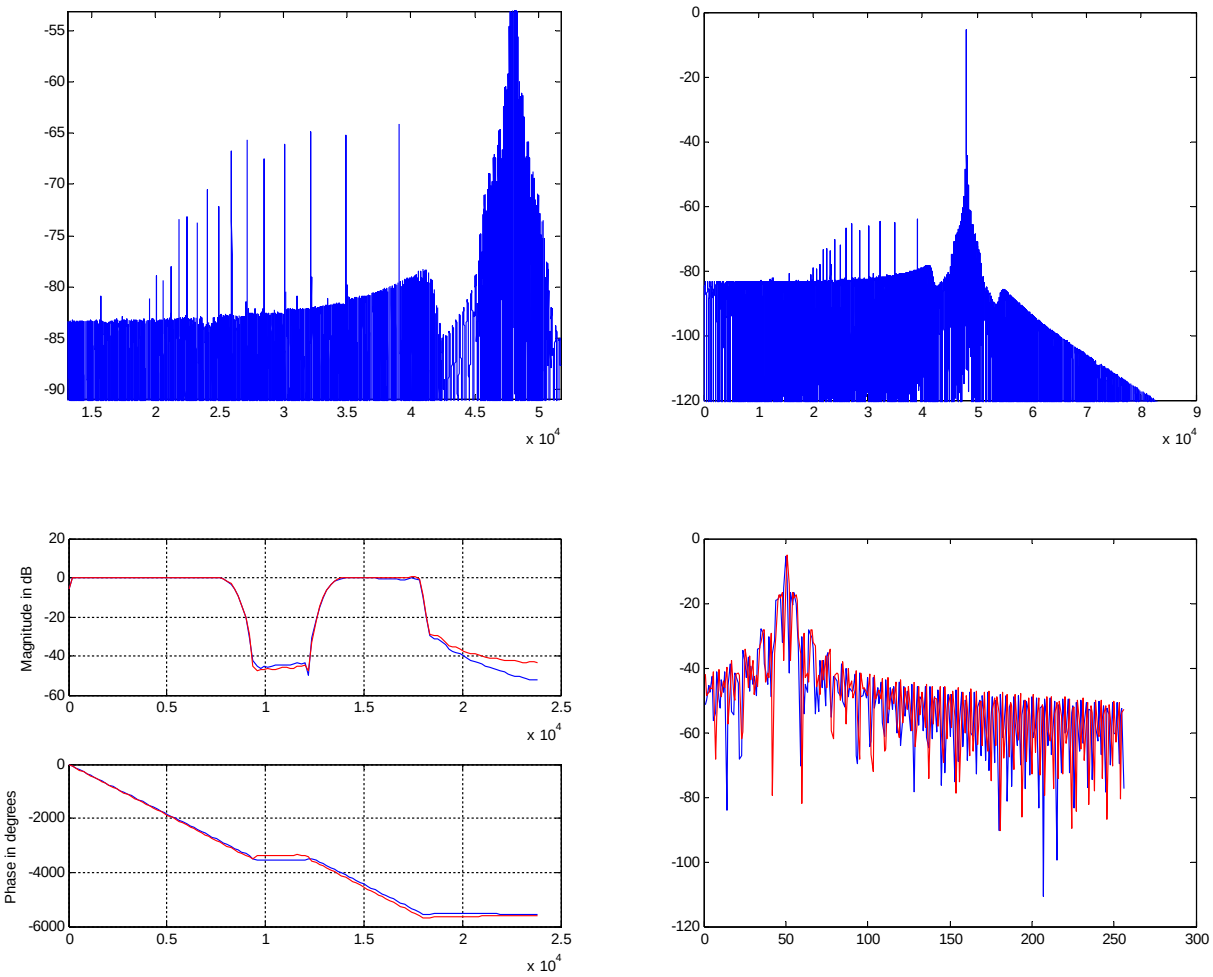
Illustration 2: blau: Log Sweep (1 Sekunde), rot: MLS (3 mal 14-Ordnung)

Systemidentifikation eines leicht nichtlinearen Systems:

```
ynl=afilter([sw; zeros(trail,1)],[], -40);
```

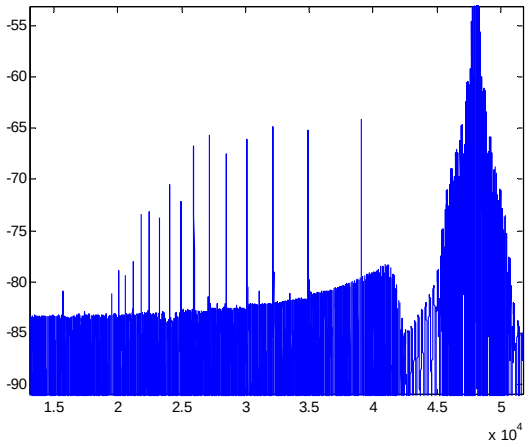



```
trail=255;
nges=length(sw)+length(isw)+trail;
hnl=real(ifft(fft(ynl,nges).*fft(isw,nges)));
hnls=hnl(length(sw):length(sw)+trail);
plot(etc(hnl));
```



Rot: Log Sweep, Lineares System *Blau: Log Sweep, Nicht lineares System;*

Analyse der gesamten Impulsantwort:



Beginn einer harmonischen Impulsantwort:

$$t_N = T - T \cdot \frac{\ln(N)}{\ln\left(\frac{f_2}{f_1}\right)}$$ mit:

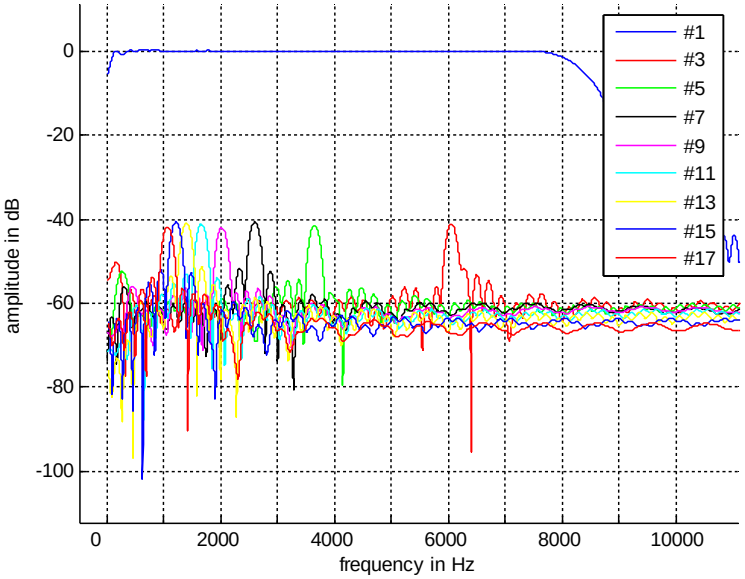
- t_N: Beginn der N-ten harmonischen Impulsantwort
- T: Dauer des Sweeps
- N: Nummer der Harmonischen
- f_2, f_1: End- und Startfrequenz (in Hz oder rad)

```
nmax=19;
n=round(T*fs-T*fs*log(1:nmax)/log(f2/f1));
hm=zeros(trail+1,nmax);
hamp=zeros(fs/2,nmax);
for ii=1:nmax
  hm(:,ii) = hnl(n(ii):n(ii)+trail);
  htemp=20*log10(abs(fft(hm(:,ii),fs)));
  hamp(:,ii) = htemp(1:fs/2);
end
clear htemp;
```

Klirrfaktor:

$$k = \sqrt{\frac{\sum_{n=2} A_n^2}{A_1^2 + \sum_{n=2} A_n^2}}$$ mit: A_n...Amplitude der n-ten Harmonischen

Ablesen des Klirrfaktors für z.B.: 1 kHz:



- A_1 : 0dB → 1
- A_3 , A_5 , ... A_{17} : 8 mal -60dB → 8 mal 0.001
- $k = \sqrt{\frac{\sum_{n=2} A_n^2}{A_1^2 + \sum_{n=2} A_n^2}}$ = 0.0028 → 2.8 %

3.4. Beispiel 1: asystem()

- Der Frequenzgang des Moduls `asystem()` soll ermittelt werden und diskutiert werden. Dabei sollen verschiedene Konfigurationen des Moduls getestet werden:
  - ideale Messung: `asystem(anregungssignal)`
  - Messung mit Störrauschen von -40dB: `asystem(anregungssignal,-40)`
  - Messung mit Störrauschen von -10dB: `asystem(anregungssignal,-10)`
  - Messung mit Nichtlinearitäten von -20dB: `asystem(anregungssignal,[],-20).`
- Befehle: `asystem, spect, etc, mls, expsweep`
  - http://piotr.majdak.com/download/mlib/index.php
- Ziele:
  - System erklären können (welcher Filter ist `asystem`?)
  - Welche Messmethoden führt zu besten Resultaten bei einer Messung unter idealisierten Bedingungen (ohne Rauschen, lineares System)? Durch Diagramme begründen.
  - Ist die Messung mit Sweeps tatsächlich immun gegenüber Nichtlinearitäten? Resultate sollten das widerspiegeln.
  - Wie können sinnvolle Resultate unter stark verrauschten Messbedingungen erzielt werden?
  - Haben Sie für faire Vergleiche zwischen den Messmethoden gesorgt?

3.5. Beispiel 2: Soundkarte

Der Frequenzgang der Soundkarte des Rechners soll mithilfe des Loop-Back-Kabels gemessen werden.

- Zuerst ein Signal über Line-Out abspielen und über Line-In (oder Mic-In) aufnehmen. Die Einstellungen der Soundkarte beachten. Clipping vermeiden.
- Danach die Systemidentifikation-Prozedur anwenden

Handhabung der Audiosignale in MATLAB

```
r = audiorecorder(FS, 16, CH); % create recorder object with CH channels
p = audioplayer(OUT, FS);      % create player object
record(r);                     % start recording
playblocking(p);               % play OUT and wait until done
stop(r);                       % stop recording
Y=getaudiodata(r);             % retrieve the recorder data to Y
```

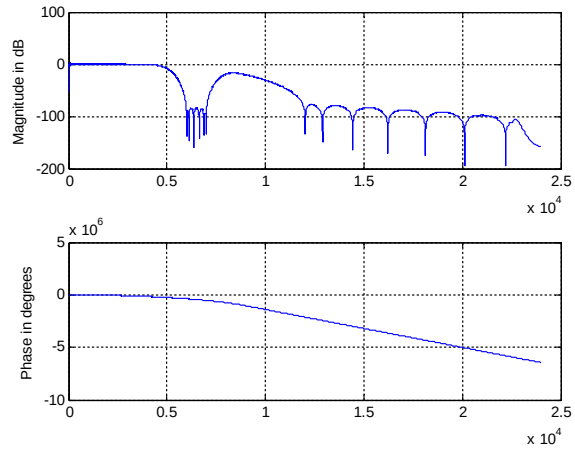
Hinweise:

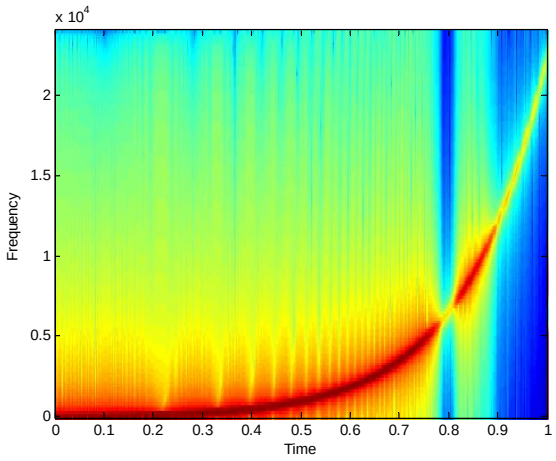
- Mono- oder Stereo-Signale?
- Wertebereich für Audioausgabe in Matlab?
- Test der Soundkarte: das ausgespielte und aufgenommene Signal anschauen (plot, spectrogram)!
- Latenz der Soundkarte berücksichtigen: Lange genug aufnehmen!
- Audiopfade beachten (Mic? Line-in? Headphones? Line-out?)
- Seit Windows 7: Audiodevices werden erst nach Anstecken eines Kabels freigeschaltet. Sollte Matlab das nicht registrieren (audiodevinfo liefert leere Input/Ouput Geräte): Matlab neustarten.

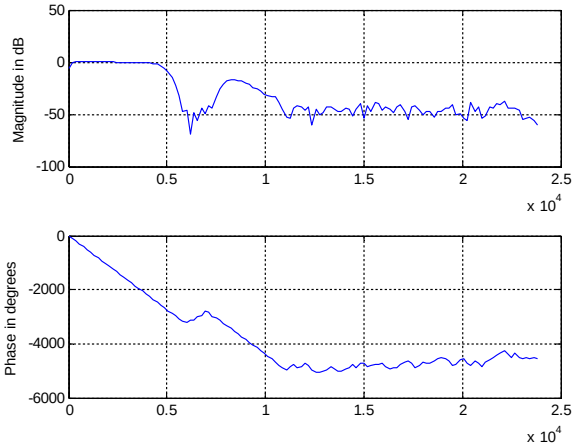
3.6. Beispiel 3: Unbekanntes System

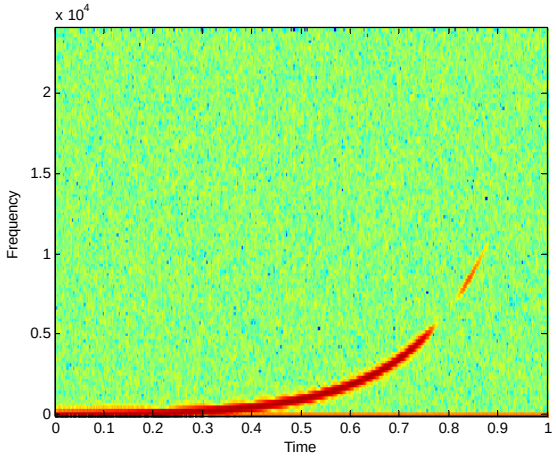
Der Frequenzgang eines unbekannten Systems (modifizierter Loop-Back-Kabel) soll identifiziert werden.

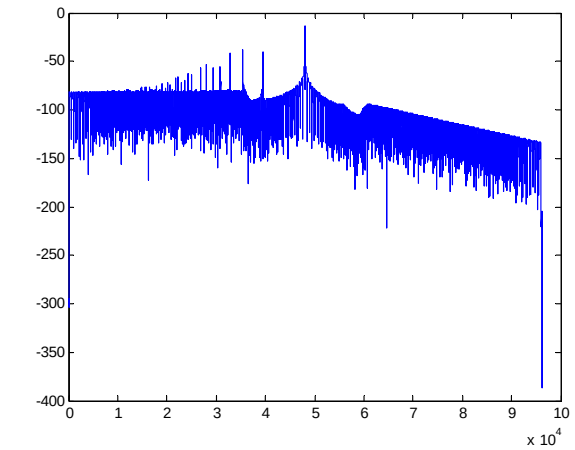
- Ein Signal über Line-Out abspielen und übe Mic-In aufnehmen.
- Signal visuell analysieren und eine geeignete Systemidentifikation-Methode auswählen
- die ausgewählte Systemidentifikation-Prozedur anwenden
- Ergebnisse diskutieren

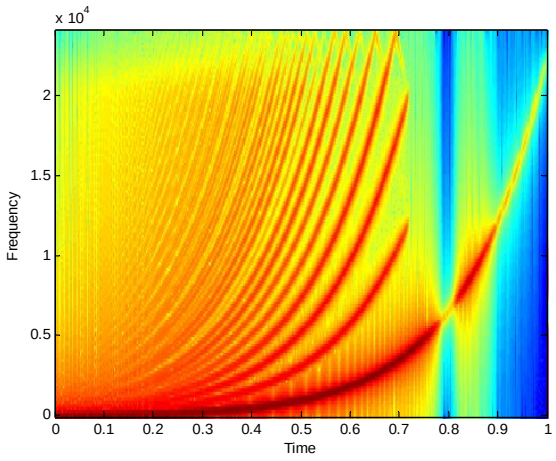

Magnitude in dB
Phase in degrees
x 10^4
x 10^6


x 10^4
Frequency
Time


Magnitude in dB
Phase in degrees
x 10^4

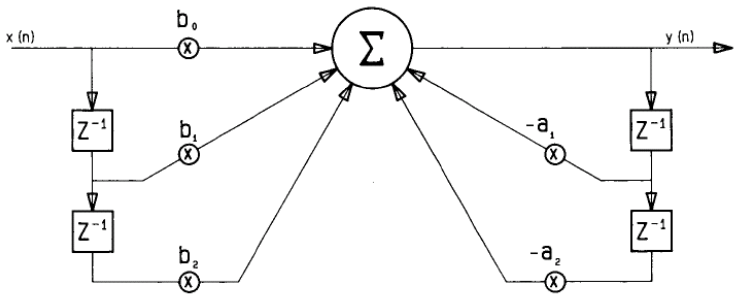

x 10^4
Frequency
Time


x 10^4


x 10^4
Frequency
Time

4. Filterstrukturen – IIR-Filter

4.1. Biquad-Filter



Differenzengleichung

$$y(n)=b_0 x(n)+b_1 x(n-1)+b_2 x(n-2) \\ -a_1 y(n-1)-a_2 y(n-2)$$

Übertragungsfunktion

$$H(z)=\frac{b_0+b_1 z^{-1}+b_2 z^{-2}}{1+a_1 z^{-1}+a_2 z^{-2}}$$

Berechnung der Koeffizienten:

- Filterdesign-Tool:

  `fdatool` → `Num` und `Den`, entspricht `B` und `A`

- Direkte Berechnung:

  IIR: `butter, cheby1, cheby2, ellip` → `B` und `A`
  FIR: `firls` → `B`, entspricht der Impulsantwort

- Tabellen/Unterlagen, hier: Tiefpass-Filter:

$$b_0=1/(1+2\cdot d\cdot F+F^2) \\ b_1=2b_0 \\ b_2=b_0 \\ a_1=2b_0(1-F^2) \\ a_2=b_0(1-2\cdot d\cdot F+F^2)$$

mit $F=\frac{1}{\tan(\pi f_c/f_s)}$

f_c ... analoge Grenzfrequenz
f_s ... Sampling Frequenz
d ... Dämpfungsfaktor

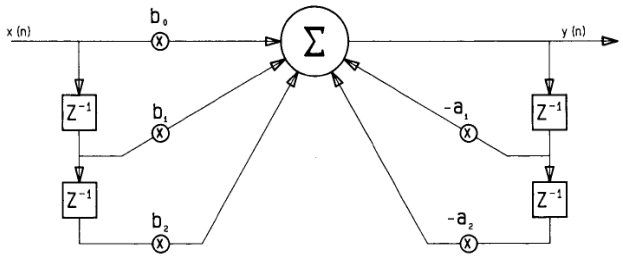
Implementation in MATLAB

Möglichkeit 1:

```
B=[ b0 b1 b2];
A=[ 1  a1 a2];
out=filter(B,A,in);
```

Möglichkeit 2 (ausprogrammierte Struktur):

```
function out=biquad_lp(in,fc,d,fs)

% biquad_lp         - low pass filter, canonical bi-quad form
% out=biquad_lp(in,fc,d,[fs])
%
% Implementation of low pass biquad filter in canonical form.
%
% input parameters:
% in:   input signal
% fc:   cut-off frequency [Hz]
% d:    damping factor
% fs:   sampling frequency

    % calculate all coefficients
F=1/tan(pi*fc/fs);
b0=1/(1+2*d*F+F*F);
b1=2*b0;
b2=b0;
a1=2*b0*(1-F*F);
a2=b0*(1-2*d*F+F*F);
    % initialize vectors
len=length(in)+2;    % get the length + overhead
y=zeros(len,1);      % define y and x
x=zeros(len,1);
x(3:len)=in;       % merge to: zeros . in . zeros
    % main loop
for n=3:len
    y(n)=b0*x(n)+b1*x(n-1)+b2*x(n-2)-a1*y(n-1)-a2*y(n-2);
end
    % set the output
out=y(3:end);
```



$$y(n)=b_0x(n)+b_1x(n-1)+b_2x(n-2)$$
$$-a_1y(n-1)-a_2y(n-2)$$

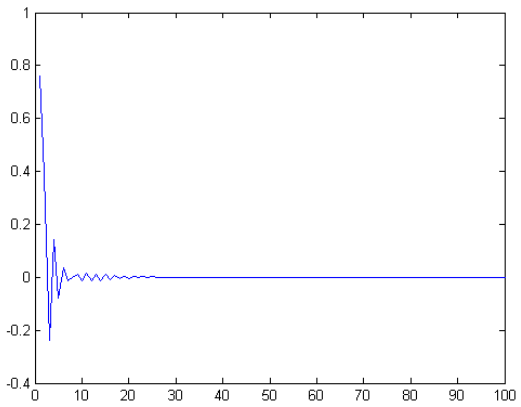
Debugging des Scripts:

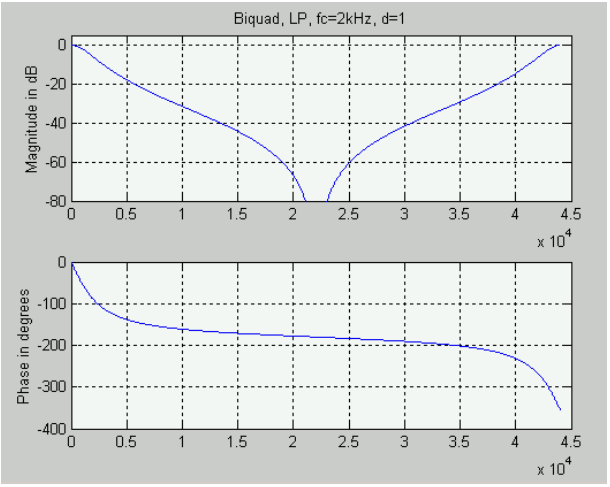
Systemidentifikation über die Impulsantwort:

```
imp=[1; zeros(1024,1)];
out=biquad_lp(imp,2000,1,44100);
```

Auswertung:

```
plot(out(1:100));
spect(fft(out),44100,'w');
```

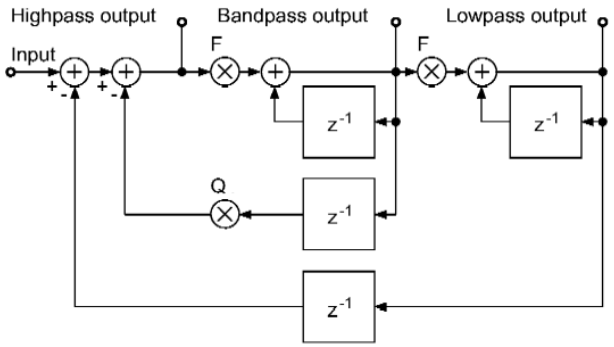




Eigenschaften der Struktur:

- Vorteile: einfache Implementation
- Nachteil: komplizierte Berechnung der Koeffizienten, viele Koeffizienten für wenige Parameter

4.2. State Variable Filter



- Tiefpass (Ausgang $y_l(n)$)
- Hochpass (Ausgang $y_h(n)$)
- Bandpass (Ausgang $y_b(n)$)

Die Differenzgleichungen:

$$y_l(n) = F \cdot y_b(n) + y_l(n-1)$$
$$y_b(n) = F \cdot y_h(n) + y_b(n-1) \qquad \text{mit} \quad F = 2\sin(\pi f_c / f_s) \quad Q = 2 \cdot d$$
$$y_h(n) = x(n) - y_l(n-1) - Q \cdot y_b(n-1)$$

Eigenschaften der State-Variable-Struktur:

Vorteile: einfache Zuordnung der Filtereigenschaften zu den Parametern

Nachteile: komplexere Berechnung, mehr Speicher notwendig.

Anwendung: Umschaltung zw. zwei extremen Settings, Filter-Glissandi, Synthesizer

4.3. Beispiel

Das State-Variable-Filter soll als Funktion `statevar` implementiert werden:

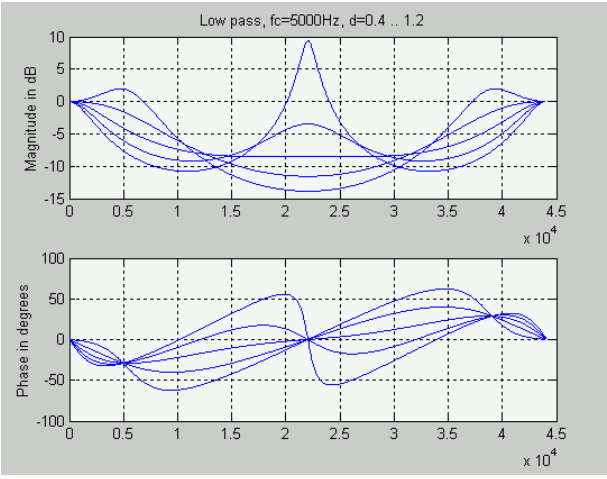
```
function out=statevar(in,fc,d,fs,typ)

% statevar        - digital state variable filter
% out=statevar(in,fc,d,fs, [typ])
%
% input parameters:
% in:   input signal
% fc:   cut-off frequency [Hz]
% d:    damping factor
% fs:   sampling frequency
% typ:  type of filter ('lp', 'bp', 'hp'), optional (default='lp')
```

Die Funktion sollte mit folgenden Parametern überprüft werden $(f_s = 44100\text{Hz})$:

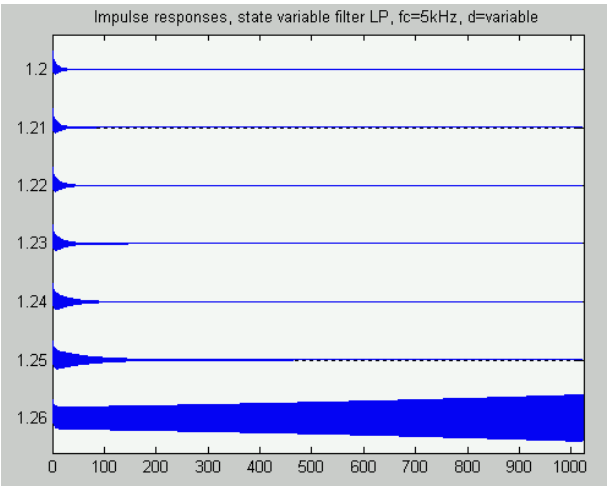
- Tiefpass, $f_c = 5000\,Hz, d = 0.4$ bis 1.4 in 0.2-Schritten
- Bandpass, f_c = 2000 Hz bis 5000 Hz in 1000 Hz-Schritten
- Hochpass, $f_c = 4000, d = 0$ bis 0.3 in 0.1-Schritten

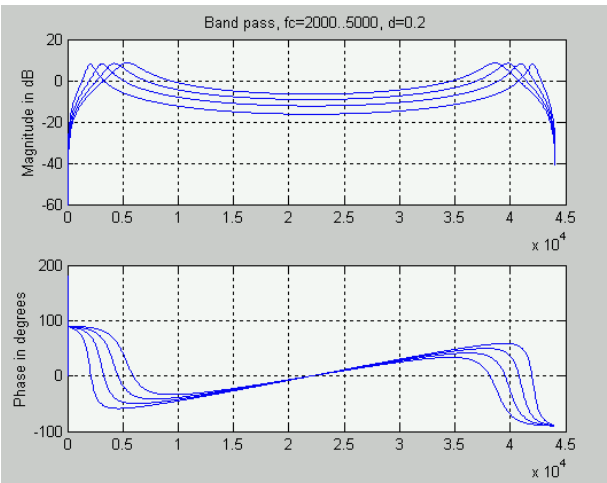
Was passiert bei einer sehr kleinen Dämpfung? Was passiert bei sehr grossen Dämpfungen und hohen Grenzfrequenzen?

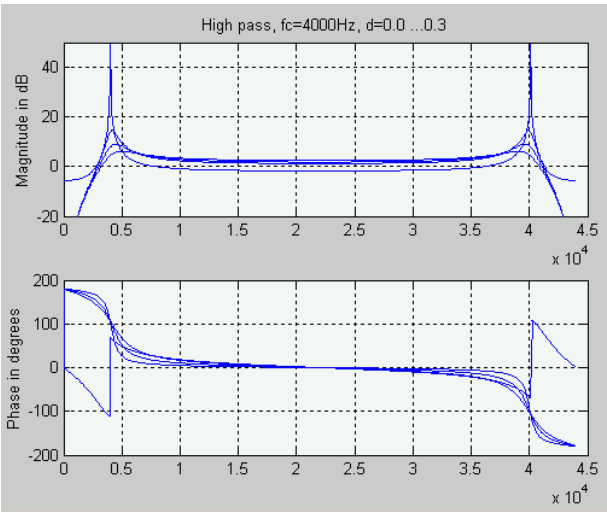
4.4. Lösungen



Stabilitätskriterium: $F < 2 - Q$







weiteres Stabilitätskriterium: $Q > 0$

4.5. Parametrische Filter

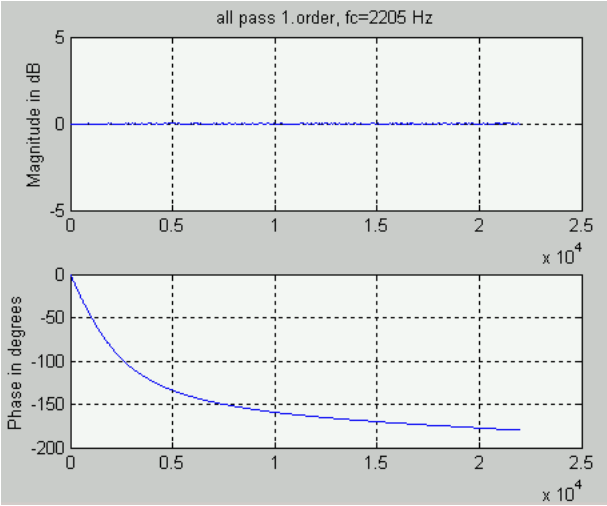
Ziel: mit einem Parameter eine gezielte Veränderung des Verhaltens des Filters ermöglichen

Allpass erster Ordnung

Übertragungsfunktion: $A_1(z)=\frac{z^{-1}+c}{1+c\cdot z^{-1}}$ mit $c=\frac{\tan(\pi f_c/f_s)-1}{\tan(\pi f_c/f_s)+1}$

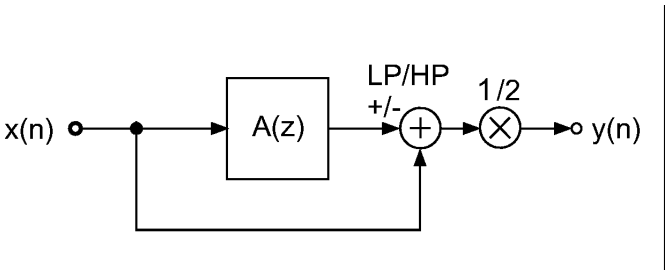
Differenzengleichung: $a(n)=c\,x(n)+x(n-1)-c\,a(n-1)$

Frequenzgang:

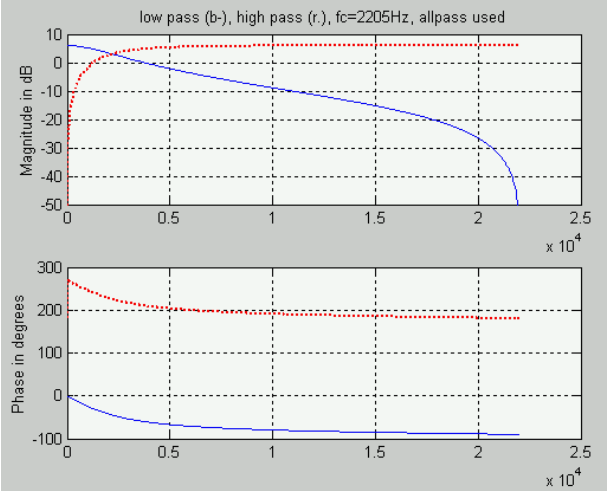


Tiefpass und Hochpass erster Ordnung

Differenzengleichung Tiefpass: $y(n)=1/2\left[x(n)+a(n)\right]$
Differenzengleichung Hochpass: $y(n)=1/2\left[x(n)-a(n)\right]$



Frequenzgang:



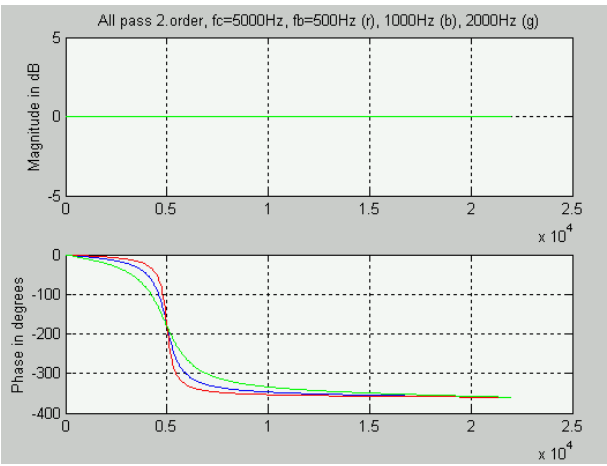
Allpassfilter zweiter Ordnung

Übertragungsfunktion: $A_2(z)=\dfrac{z^{-2}+c(1-b)z^{-1}-b}{1+c(1-b)z^{-1}-bz^{-2}}$

mit $b=\dfrac{\tan(\pi f_b/f_s)-1}{\tan(\pi f_b/f_s)+1}$ und $c=-\cos(2\pi f_c/f_s)$, (f_c ... Grenzfrequenz, f_b ... Bandbreite)

Differenzengleichung: $a(n)=-bx(n)+c(1-b)x(n-1)+x(n-2)-c(1-b)a(n-1)+ba(n-2)$

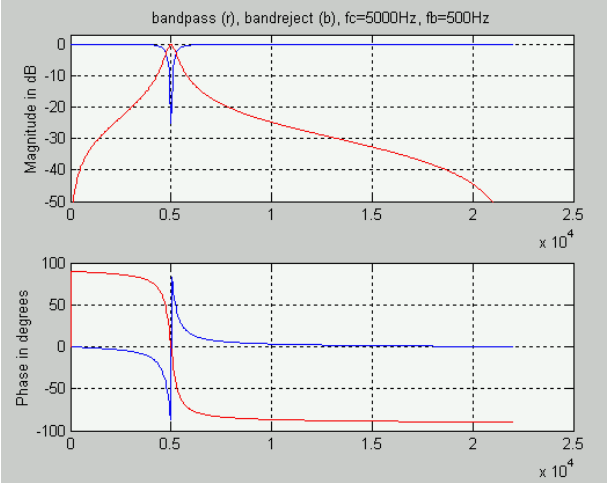
Frequenzgang:



Bandpass und Bandsperre zweiter Ordnung

Übertragungsfunktion Bandsperre: $y(n)=1/2\left[x(n)+a_2(n)\right]$
Übertragungsfunktion Bandpass: $y(n)=1/2\left[x(n)+a_2(n)\right]$

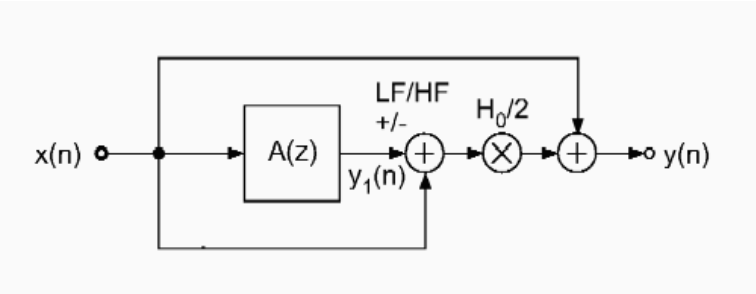
Frequenzgang:



4.6. Equalizer

Shelving Filter

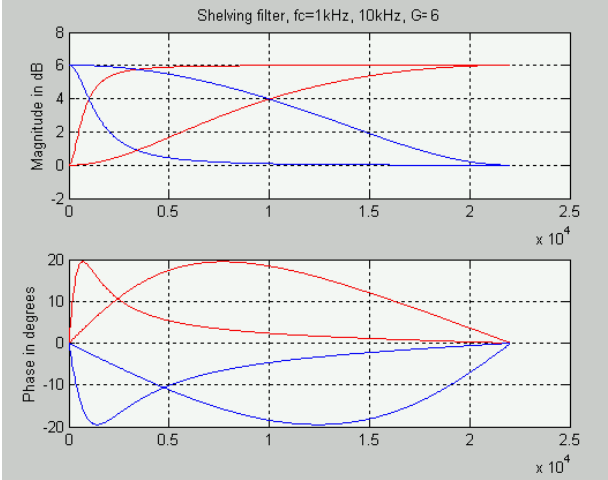
Übertragungsfunktion: $H(z)=1+\frac{H_0}{2}\cdot[1\pm A(z)]$ mit LF (+), HF (-) und $H_0=G_c-1$, wobei G_c die Verstärkung ist.



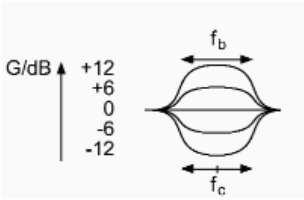
Differenzgleichungen: $y_1(n)=cx(n)+x(n-1)-cy_1(n-1)$

$$y(n)=\frac{H_0}{2}\left[x(n)\pm y_1(n)\right]+x(n)$$

Frequenzgang:



Peak filter / Notch filter

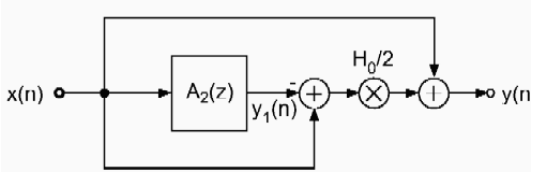


Übertragungsfunktion: $H(z)=1+\frac{H_0}{2}\cdot\left[1-A_2(z)\right]$

mit $A_2(z)=\frac{z^{-2}+c(1-b)z^{-1}-b}{1+c(1-b)z^{-1}-bz^{-2}}$, $b=\frac{\tan(\pi f_b/f_s)-1}{\tan(\pi f_b/f_s)+1}$, $c=-\cos(2\pi f_c/f_s)$,

(f_c ...Center-Frequenz, f_b ... Bandbreite) und $H_0=G_c-1$, wobei G_c die Verstärkung des Peaks bei der Center-Frequenz darstellt.

Struktur:

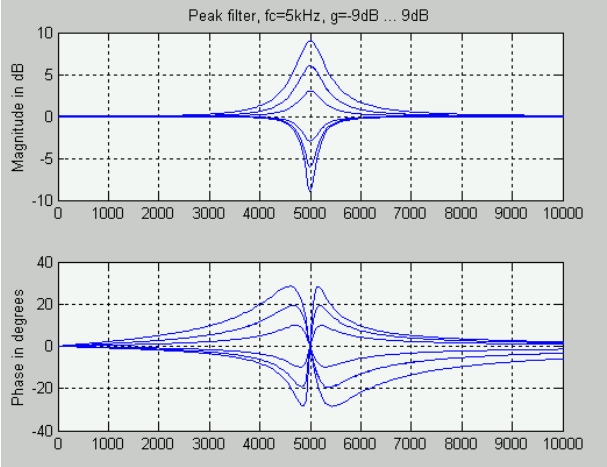


Differenzengleichungen:

$$y_1(n)=-bx(n)+c(1-b)x(n-1)+x(n-2)-c(1-b)y_1(n-1)+by_1(n-2)$$

$$y(n)=\frac{H_0}{2}\left[x(n)-y_1(n)\right]+x(n)$$

Frequenzgang:



4.7. Beispiel: Shelving und Peak-Filter

- Ein Shelving-Filter erster Ordnung soll implementiert werden. Die Parameter werden über folgenden Header übergeben:

```
function out=shelvfilt(in,fc,G,fs,typ)

% shelvfilt         - shelving filter, first order
% out=shelvfilt(in,fc,g,fs)
%
% input parameters:
% in:   input signal
% fc:   cut-off frequency [Hz]
% G:    gain [dB]
% fs:   sampling frequency [Hz]
% typ:  type of filter ('lp', 'hp'), optional (default='lp')
```

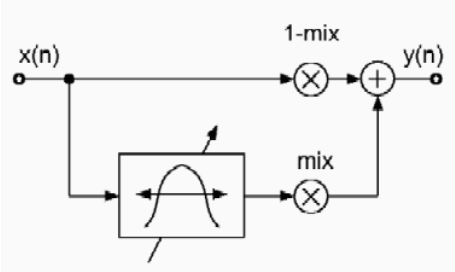
- Ein Peak-Filter zweiter Ordnung soll implementiert werden. Folgender Header ist zu verwenden:

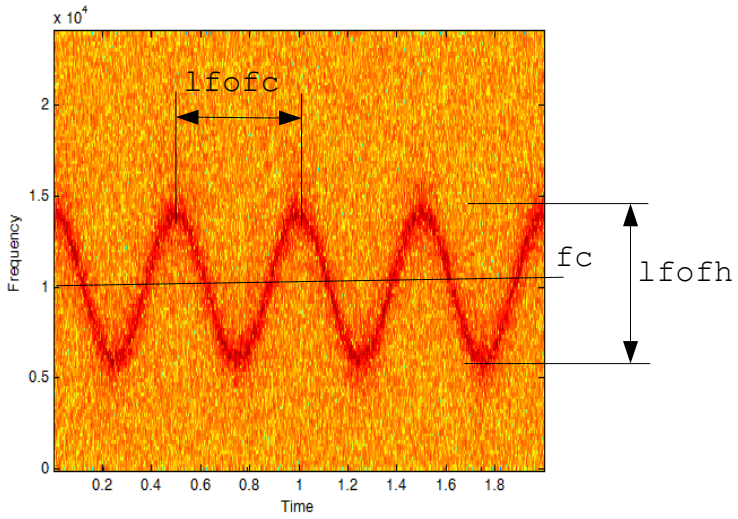
```
function out=peakfilt(in,fc,fb,G,fs)

% peakfilt         - peak filter, second order
% out=shelv(in,fc,fb,g,fs)
%
% input parameters:
% in:   input signal
% fc:   cut-off frequency [Hz]
% fb:   bandwidth [Hz]
% G:    gain [dB]
% fs:   sampling frequency [Hz]
```

Beide Filter sollen überprüft werden (Impulsantworten, Frequenzgänge).

5. Zeitvariante Filter

5.1. Auto-wah-Effekt





Ein Auto-wah-Effekt soll auf der Basis eines zeitvarianten Bandpassfilter implementiert werden.

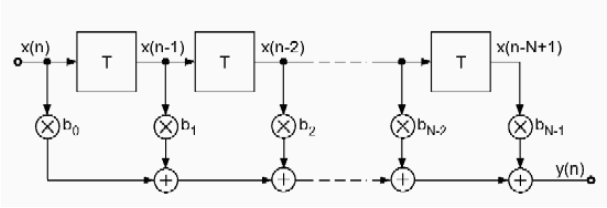
```
function out = autowah(in, fc, fb, G, lfofc, lfofh, mix, fs)

% input parameters:
%  in:          input
%  fc:          center frequency [Hz]
%  fb:          bandwidth [Hz]
%  G:           gain [dB]
%  lfofc:       LFO center frequency [Hz]
%  lfofh:       LFO frequency hub
%  mix:         ratio of dry (0) to wet (1)
%  fs:          sampling frequency [Hz]
```

Hinweise:

- Erweiterung eines Peakfilters: Modulation der fixen Mittenfrequenz des Peakfilters mit Filterkoeffizient `c` als Vektor
- Schrittweise debuggen:
  - Impulsantwort und Spektrum des zeit-invarianten Filters (`lfofh=0`)
  - Spektrogramm des zeitvariantes Filters für Rauschen (`randn`) als Eingangssignal
- Parameterwahl beim debuggen beachten, z.B., hohe Verstärkung, Wet, hohe Mittenfrequenz, großer Hub, langsame Variation der Mittenfrequenz, langes Eingangssignal.

6. FIR-Filter

6.1. Allgemein



Übertragungsfunktion: $H(z)=\sum_{k=0}^{k=L-1} b_k \cdot z^{-k}$

Differenzengleichung: $y(n)=\sum_{k=0}^{k=L-1} b_k \cdot x(n-k)$

Impulsantwort: $h(n)=\sum_{k=0}^{k=L-1} b_k \cdot \delta(n-k)$ und entspricht den Gewichten in der Übertragungsfunktion.

6.2. Berechnung in MATLAB

Annahme:
`sig`:Signal, `ir`: Impulsantwort

```
N=length(sig);
M=length(ir);
```

Möglichkeit 1: Faltung:

```
out=conv(sig, ir);        % sehr langsam bei längeren IR's!!!
plot(out(1:N));
```

Möglichkeit 2: `filter` – Befehl

```
out=filter(ir,1,sig);     % langsam bei längeren IR's!!!
plot(out);
```

Möglichkeit 3: über die Fouriertransfomation (schnelle Faltung):

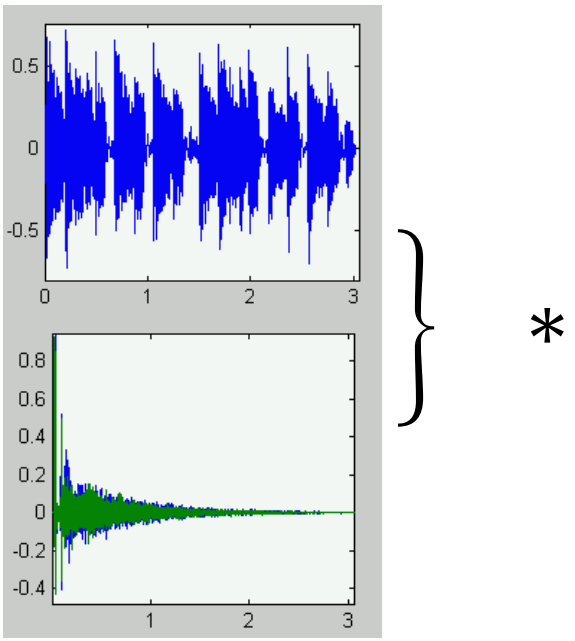
```
sigf=fft(sig,N+M);
irf=fft(if,N+M);
outf=sigf.*irf;
out=real(ifft(outf);
plot(out(1:N));
```

Möglichkeit 4: Fouriertransformation mit Overlap-and-Add-Methode:

```
out=fftfilt(ir,[sig; zeros(M,1)]);
plot(out);
```

6.3. Beispiel: Virtueller Hall

Eine Aufnahme, durchgeführt in einem trockenen Studioraum (=Freifeldaufnahme), soll virtuell in einem reell vorhandenen Raum positioniert werden. Dadurch soll der Eindruck einer in diesem Raum durchgeführten Aufnahme entstehen. Als Raumübertragungsfunktion stehen mehrere Impulsantworten zur Verfügung. Bei Raumimpulsantworten im Stereo-Format, soll als Ergebnis ebenfalls eine Stereo-WAV-Datei entstehen.



6.4. Beispiel: Virtual Sound Positioning

- Eine Aufnahme soll virtuell um den Kopf im Bereich $\pm 90°$ rotiert werden. Als Head-Related Transfer Functions (HRTFs) stehen KEMAR-Impulsantworten in 5°-Auflösung für die Horizontalebene für beide Ohren zur Verfügung. Die gerendete virtuelle Schallquelle soll von 0° bis +90° (nach links) ausgelenkt, danach umkehren und bis -90° rotiert werden, um zum Schluss bei der Position 0° stehen zu bleiben, wobei die Änderung des Azimuts linear erfolgen soll. Die für die Aktualisierung der Filterkoeffizienten der HRTFs notwendige Interpolation soll vernachlässigt werden.
- Implementation: siehe vsp.m
- Debugging: siehe bsp_vsp.m
- Wie wirkt sich die Kopfbewegung (=Geschwindigkeit der Filterveränderung) auf die korrekte Filterung?

7. Zeitverzögerungsglieder

7.1. Allgemein

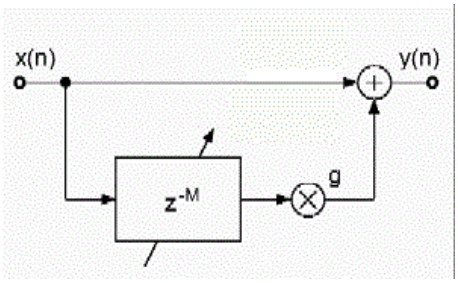
- Schallausbreitung
- Auswirkung im Zeitbereich
- Auswirkung im Frequenzbereich
- Wiederholte Reflexionen

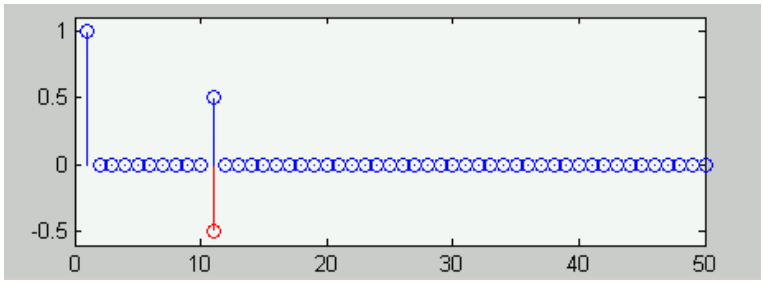
7.2. FIR-Kammfilter

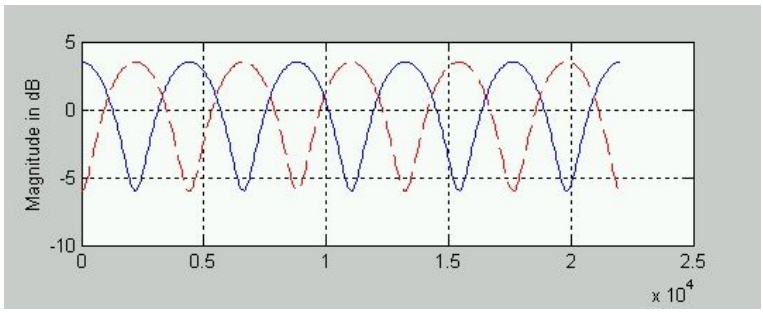
Entstehung: einfache Raumreflexionen

$y(n)=x(n)+g\cdot x(n-M)$ mit $M=\tau\cdot f_s$

$H(z)=1+g\cdot z^{-M}$





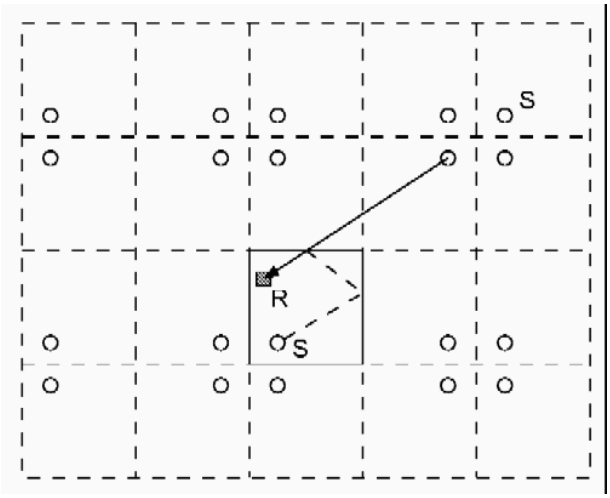


MATLAB:

```
dline=[zeros(M,1); in];
in=[in; zeros(M,1)];
out=in + gain.*dline;
```

7.3. Hallalgorithmen

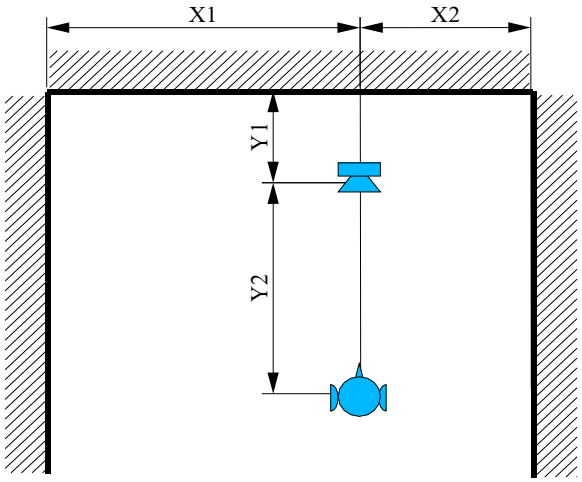
Spiegelquellenprinzip



Kodierung der Distanz für eine Kugelwelle: Schalldruck nimmt mit $1/r$ ab.

Beispiel: Hallsimulation (FIR, Spiegelquellen erster Ordnung)

Hallsimulation einer Audioquelle in folgender Aufstellung:



Schallharte Wände ohne Dämpfung, Frequenzverhalten: Tiefpasswirkung 1. Ordnung, f_c = 2 kHz. Schallgeschwindigkeit: c = 340 m/s, Abtastfrequenz: 44.1kHz. Zusätzlich sollte die Distanz mit 1/r berücksichtigt werden. Geometrie: X1 = 6m; X2 = 4m; Y1 = 2m; Y2 = 4m

Vorgehensweise

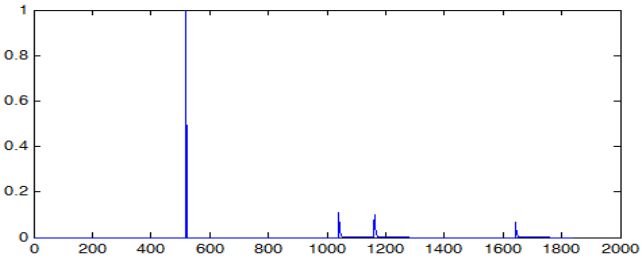
- Positionen der Spiegelquellen ermitteln
- Jede Spiegelquelle verzögern und filtern, anschließend der Distanz entsprechend abschwächen
- Reflexionen und Direktschall überlagern
- Berücksichtigung der Distanz: Faktor: `r0/r` mit `r0` als Abstand zum Direktschall und `r` als Abstand zur Spiegelquelle

Lösungen

Koeffizienten, in `filter`-kompatibler Form, aus fdatool oder TP-Filter in Allpassform:

```
A=[1 -0.75];
B=[0.125 0.125];
```

Impulsantwort:

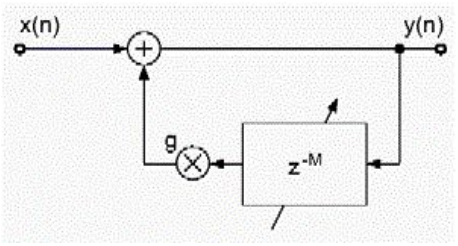


7.4. IIR-Kammfilter

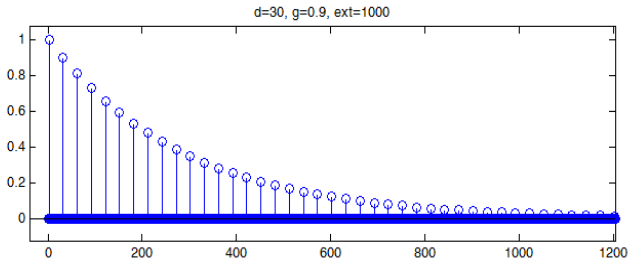
Anwendung: Zum Beispiel: Entstehung rekursiver Raumreflexionen

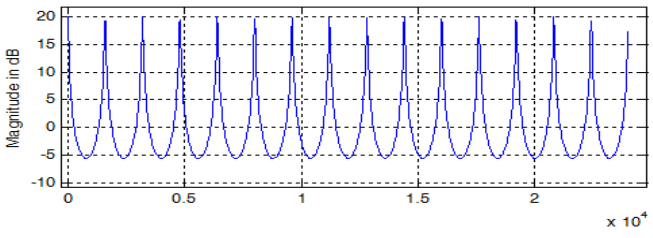
$$y(n)=c\cdot x(n)+g\cdot y(n-M) \text{ mit } M=\tau\cdot f_s$$

$$H(z)=\frac{c}{1-g\cdot z^{-M}}$$



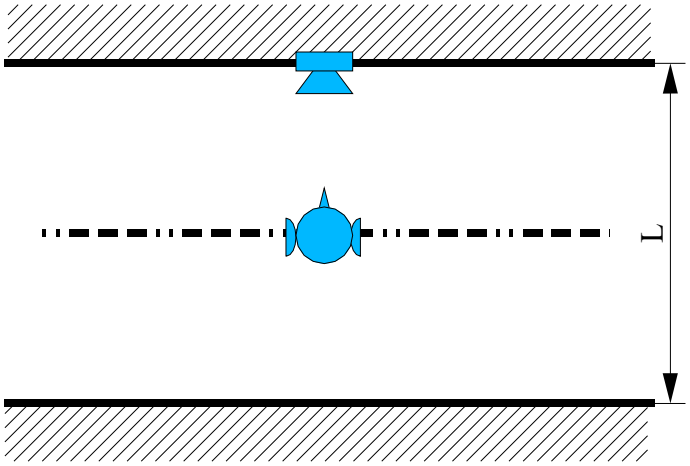

```
function out = iircomb(in, d, g, ext)
% in: input vector
% d:  delay [samples]
% g:  gain in the recursive path (linear)
% ext: extension of the input signal [samples]
% out: output vector

in=[zeros(d+1,1); in; zeros(ext,1)];
y=zeros(length(in),1);
yh=zeros(length(in),1);
for n=d+1:length(in)
    yh(n) = g*y(n-d);
    y(n)=in(n)+yh(n);
end
out=y(d+1:end);
```





Beispiel: Hallsimulation (IIR)

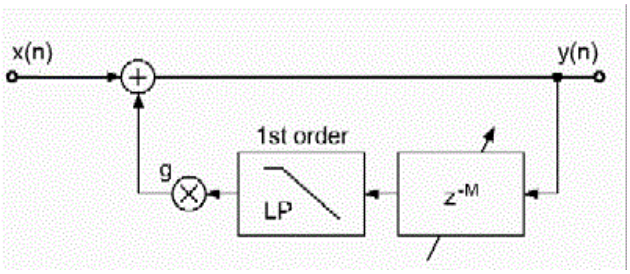
Hallsimulation einer Audioquelle in folgender Aufstellung:



Abstand zwischen den schallharten Wänden: L = 17 m. Dämpfung einer Wand: g = -3dB, Frequenzverhalten: Tiefpasswirkung 1.Ordnung, f_c = 2 kHz. Schallgeschwindigkeit: c = 340 m/s, Abtastfrequenz: 44.1kHz

Vorgehensweise

- Auswahl und Anpassung der Struktur
- Filterentwurf:
  - Auswahl des Tiefpassfilters
  - Berechnung der Koeffizienten
  - Einbeziehung des Dämpfungsfaktors
  - Überprüfung des Filters (Impulsantwort, `filter`)
- Berechnung der Zeitverzögerung
- Überprüfung des Hallalgorithmus (Impulsantwort)
- Verhallung eines Audiosignals

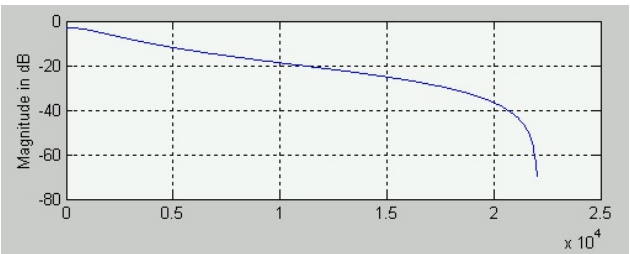


Lösungen

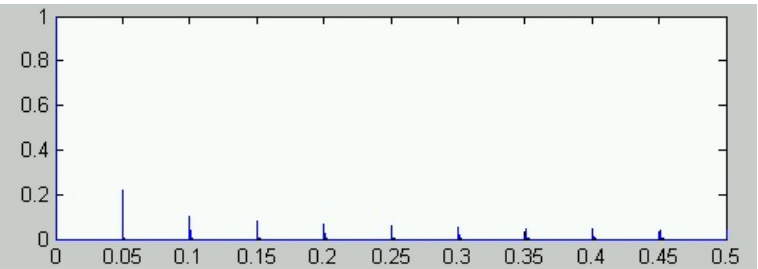
Koeffizienten (Tiefpass+Dämpfung), in `filter`-kompatibler Form, aus fdatool oder TP-Filter in Allpassform:

```
A=[1 -0.75];
B=[0.125 0.125]/1.4125;
```

Amplitudengang des Filters:

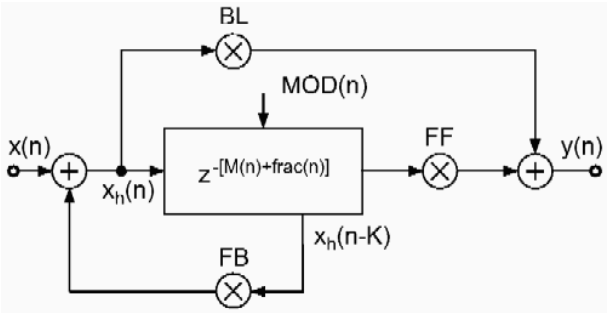


Impulsantwort des Kammfilters:



7.5. Delay Line – Effects

Parametrierung:



Effekt	*BL*	*FF*	*FB*	*Delay [ms]*	*MOD(n)*
Resonator	-	0 < x < 1	-	0..20	-
Slapback	-	0 < x < 1	-	20..50	-
Echo	-	0 < x < 1	-	> 50	-
Vibrato	-	1	-	0	0.1-5 Hz, sinus
Flanger	0.7	0.7	0.7	0	0.1-1 Hz, sinus
Chorus	0.7	1	(-0.7)	2..30	LP-Rauschen
Doubling	0.7	0.7	-	30..100	LP-Rauschen

8. Homomorphe Signalverarbeitung

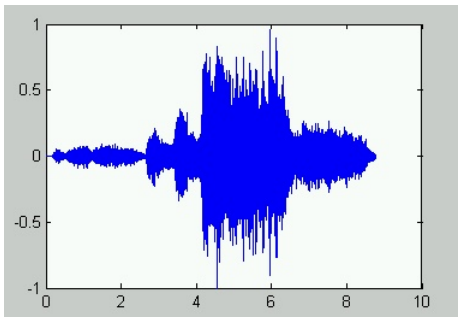
8.1. Änderung des Dynamikumfanges

Multiplikation → Addition

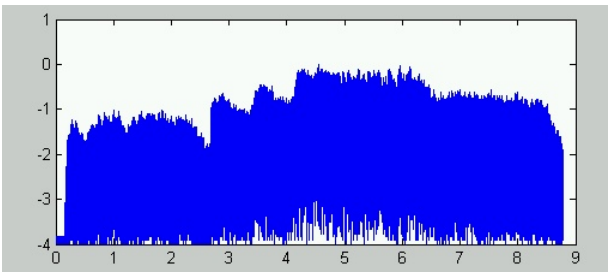
$$y(n)=s(n)\cdot h(n)\rightarrow \tilde{y}=\tilde{s}(n)+\tilde{h}(n)$$
$$y(n)=s(n)\cdot h(n) \quad | \ \log$$
$$\log y(n)=\log s(n)+\log h(n)$$

Berechnungsvorgang

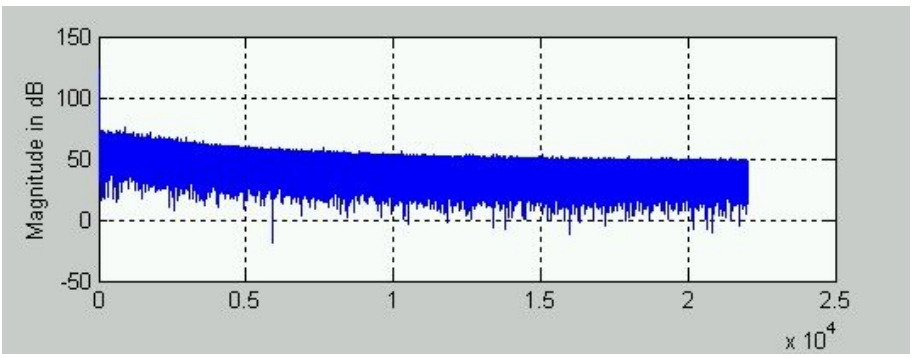
Signal: $y(n)=s(n)\cdot d(n)$



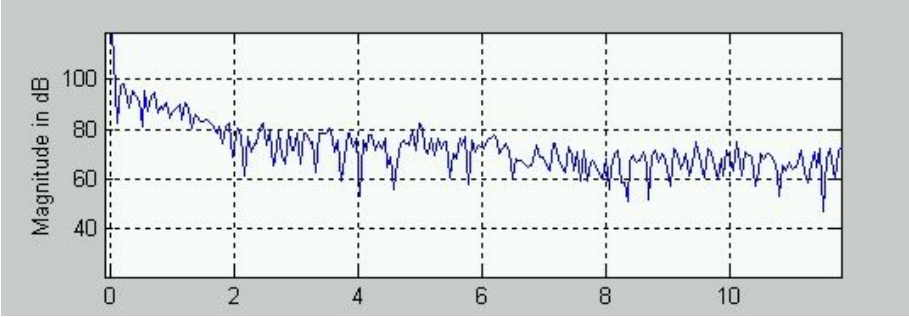
Logarithmieren: $\log y(n)=\log s(n)+\log d(n)$



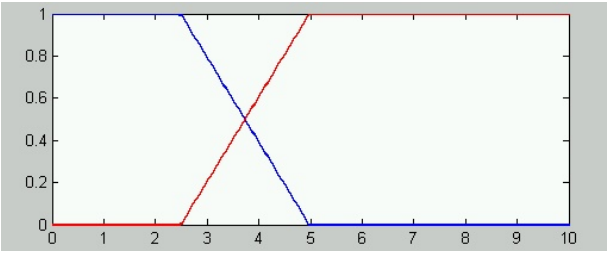
Ergebnis: $\tilde{y}(n)=\tilde{s}(n)+\tilde{d}(n)$
Dynamik $d(n)$: Tieffrequenter Anteil
Signal (dynamiklos) $s(n)$: Hochfrequenter Anteil
Trennung: Transformation in den Frequenzbereich: $\tilde{Y}(k)=\tilde{S}(k)+\tilde{D}(k)$



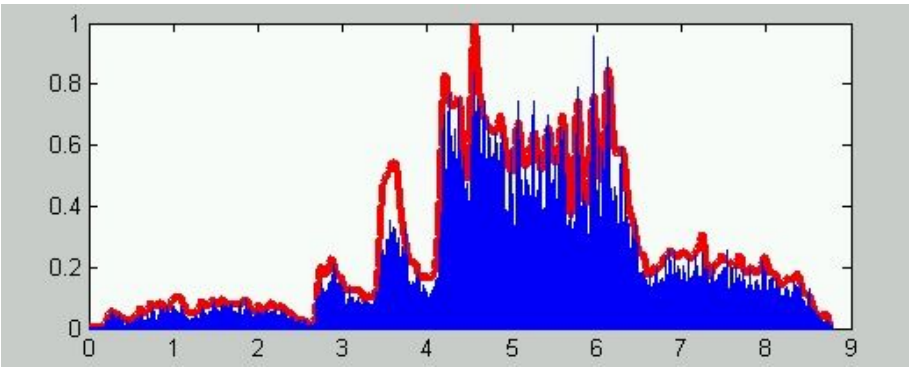
Zoomed:



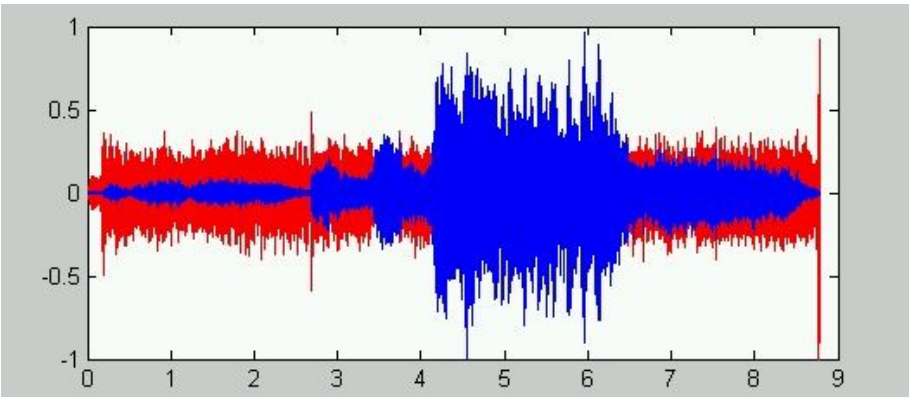
Trennung: Fensterung mit Tiefpass/Hochpass-Fenster: $\tilde{D}(k)=\tilde{Y}(k)\cdot W_{LP}(k)$ und $\tilde{S}(k)=\tilde{Y}(k)\cdot W_{HP}(k)$



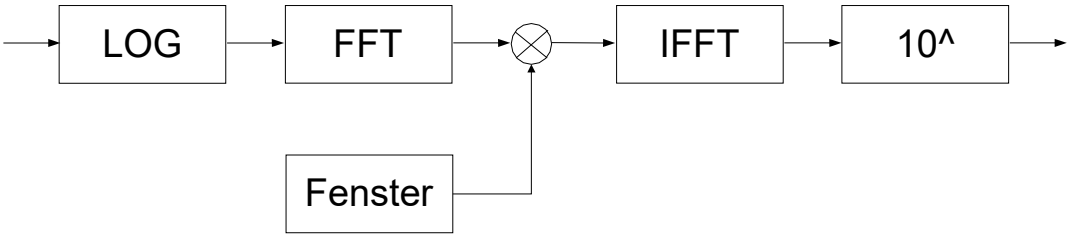
Rückrechnung in die Zeitdomäne ergibt a) den Verlauf der Dynamik: $d(n)=10^{IFFT[\tilde{D}(k)]}$



und b) das dynamiklose Signal: $s(n)=10^{IFFT[\tilde{S}(k)]}$



Gesamtstruktur:



8.2. Cepstrum

Ziel: 2 Signale, über Faltung verbunden, zu trennen:

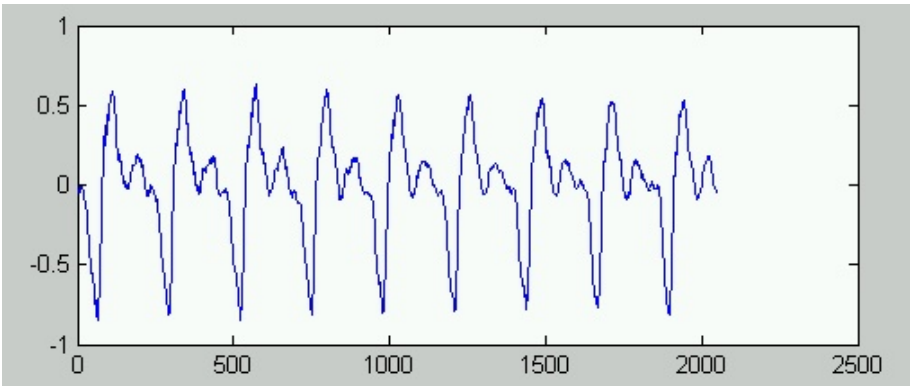
$$y(n)=s(n)*h(n)\rightarrow\tilde{y}=\tilde{s}(n)+\tilde{h}(n)$$

Definition

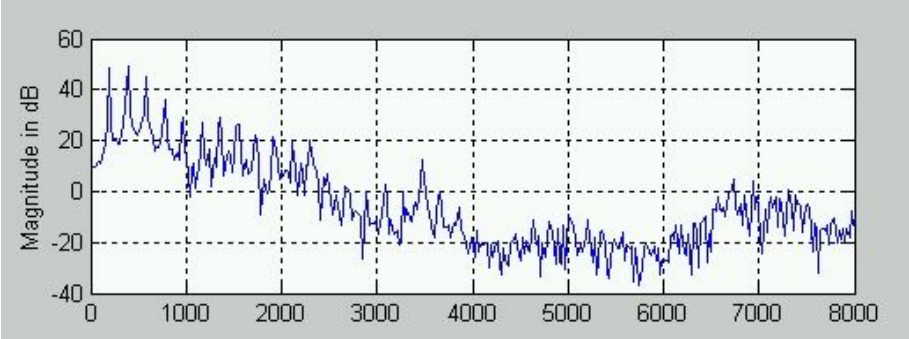
$$\begin{aligned} y(n)&=s(n)*h(n) && \mid FFT\\ Y(k)&=S(k)\cdot H(k) && \mid \log\\ \log Y(k)&=\log S(k)+\log H(k) && \mid IFFT\\ \tilde{y}(n)&=\tilde{s}(n)+\tilde{h}(n) \end{aligned}$$

Beispiel: Sprachanalyse

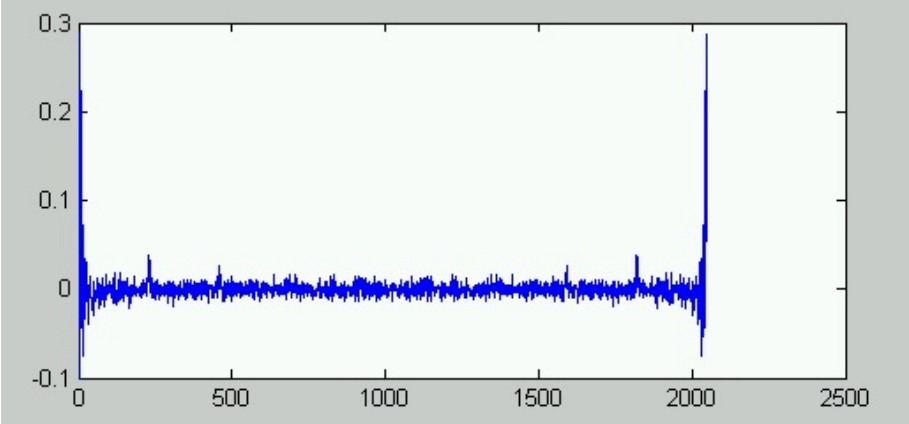
Ausschnitt im Zeitbereich: $y(n)=s(n)*h(n)$



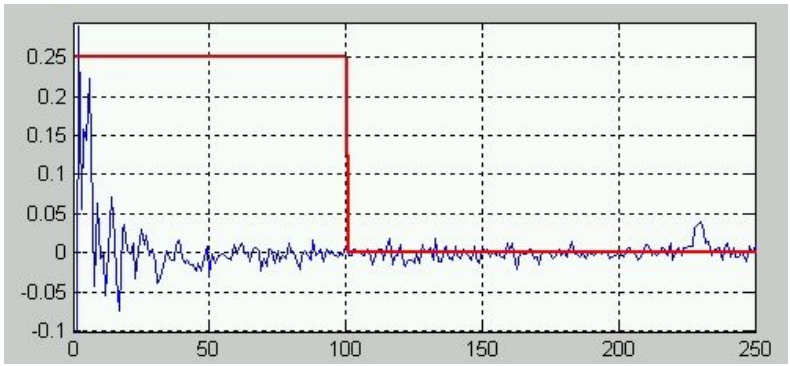
Ausschnitt im Frequenzbereich: $Y(k)=S(k)\cdot H(k)$



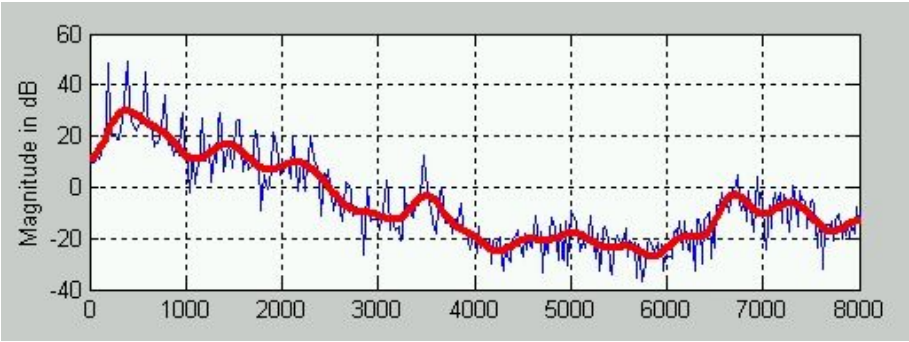
Nach der Transformation in den Cepstrum-Bereich: $\tilde{y}(n)=\tilde{s}(n)+\tilde{h}(n)$



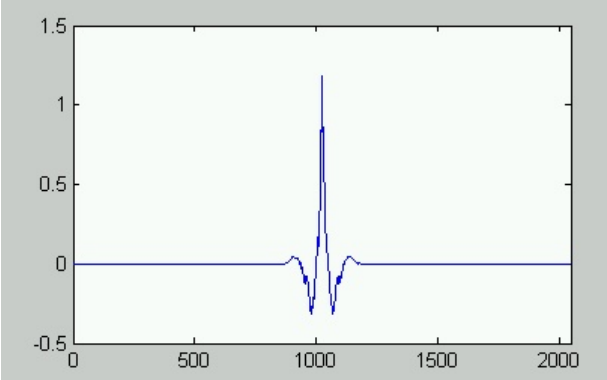
Fensterung: $\tilde{s}(n)=\tilde{y}(n)\cdot w(n)$



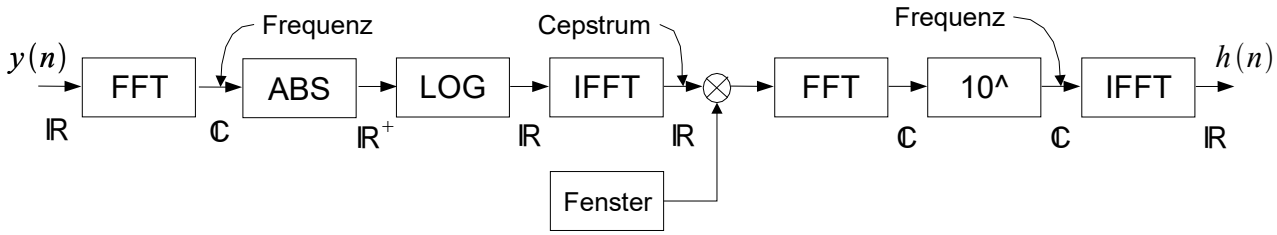
Nach der Fensterung im Frequenzbereich: $S(k)=10^{IFFT[\tilde{s}(n)]}$



Rekonstruktion des Filter als Impulsantwort: $s(n)=IFFT[S(k)]$

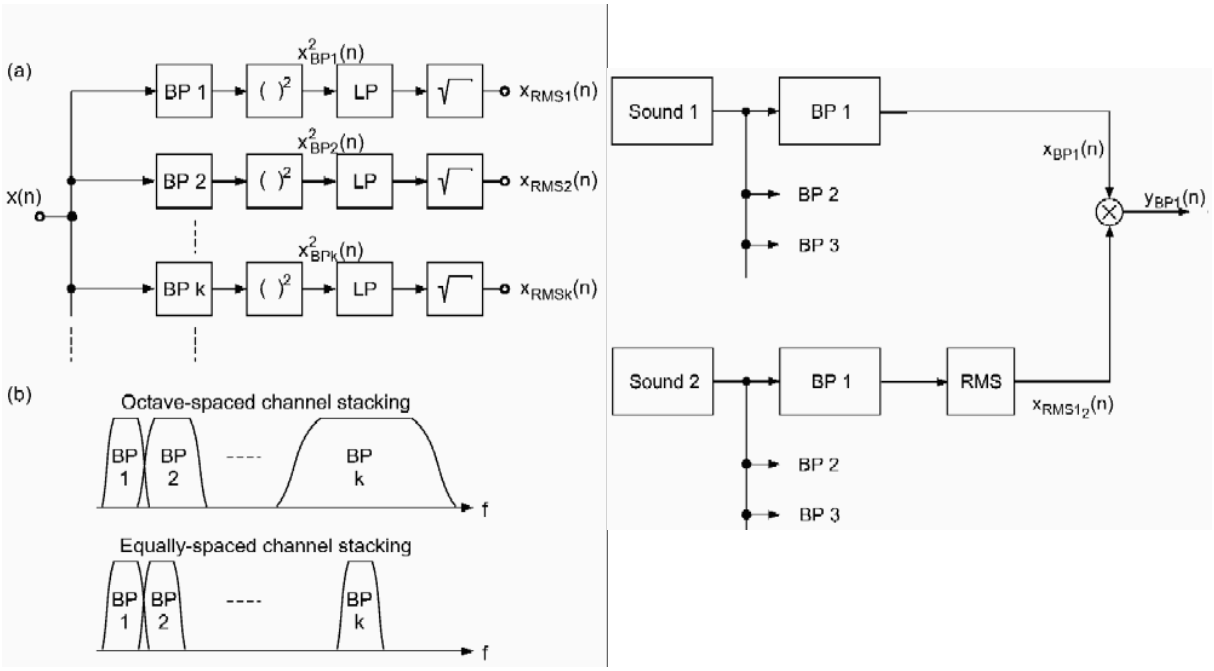


Reelles Cepstrum



8.3. Channel vocoder

Ziel: Spektrum-Einhüllende eines Signals einem zweiten Signal aufprägen

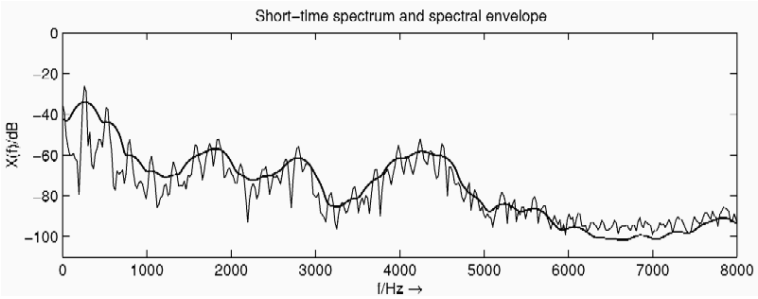


Vorgehensweise

- Berechnung der Einhüllenden des Filtersignals
- Gewichtung des Hauptsignals nach der Einhüllenden des Filtersignals

Vorgehensweise für Systeme mit Filtern gleicher Bandbreite:
1. Berechnung von RMS
2. Summe aller Energie in einem Frequenzband
3. Filterung des Spektrums $Y(k)$ mit der Filtereinhüllenden $H(k)$: $E(k)=Y(k)*H(k)$:
4. Vereinfachung der Filterung über FFT: $E(k)=IFFT\{FFT\{Y(k)\}\cdot FFT\{H(k)\}\}$

Beispiel



8.4. Beispiele

- Berechnung des Dynamikverlaufes eines Signals mit homomorpher Signalverarbeitung.
  - Signal mit starken Dynamikveränderungen verwenden (zB.: figaro1.wav)

- Veränderung eines Signals über die Extraktion der Hüllkurve (Cepstrum, LPC, Channel-Vocoder).
  - Beispiel: weisslich.m
  - Skript: overlapandadd.m erweitern