Einführung in die Digitale Signalverarbeitung

Prof. Dr. Wolfgang Fohl

Department Informatik
HAW Hamburg

Sommer 2010

Homepage:
http://www.informatik.haw-hamburg.de/fohl.html

E-Mail: fohl@informatik.haw-hamburg.de

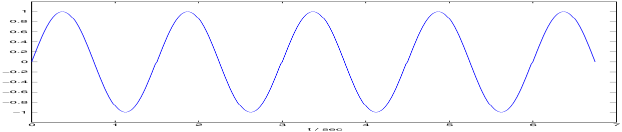
Übersicht

Einführung

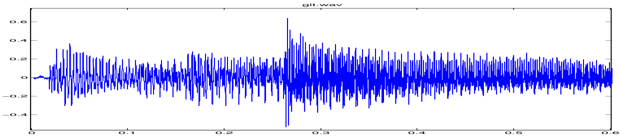
- Kontinuierliche Signale
- Wiederholung: Fourier-Reihe
- Wiederholung: Fourier-Synthese
- Wiederholung / Exkurs: Komplexe Zahlen
- Zeitdiskrete Signale
  - Abtastung, Aliasing, Nyquist-Theorem
  - Quantisierung, Clipping

Beispiele für Klangsignale

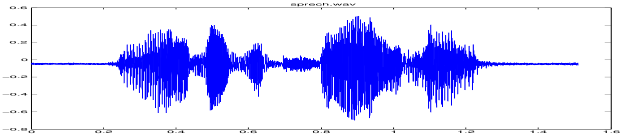
- Sinuston



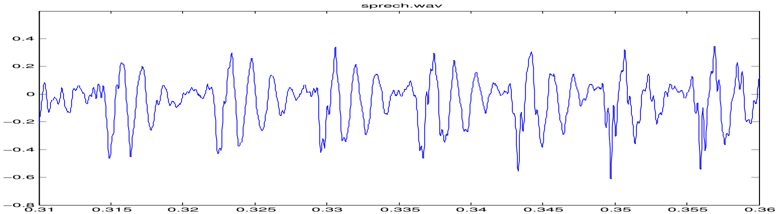
- Klang einer Gitarre



- Sprache



Periodische Signale



Der Ausschnitt aus dem Sprachsignal zeigt:

- Klänge haben einen quasi-periodischen Zeitverlauf

Ein periodisches Signal wiederholt sich nach der Zeit T

$$x(t) = x(t + nT) \qquad n \in \mathcal{Z}$$

$T = 1/\nu = 2\pi/\omega$	Periode
$\nu = 1/T$	Frequenz
$\omega = 2\,\pi\,\nu$	Kreisfrequenz

Das Fourier-Theorem

Sinus- und Kosinussignale sind die einfachsten Repräsentanten periodischer Signale.

Fourier-Synthese

Alle[1] periodischen Signale können durch die Überlagerung von Sinus- und Kosinussignalen dargestellt werden:

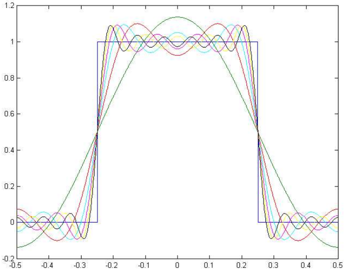
$$x(t) = a_0 + a_1 \cos(\omega_0 t) + b_1 \sin(\omega_0 t) + a_2 \cos(2\omega_0 t) + b_2 \sin(2\omega_0 t) + \cdots$$

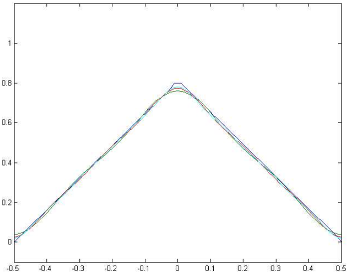
Die Frequenzen der Sinus- und Kosinussignale sind ganzzahlige Vielfache der Frequenz des Signals $x(t)$

Die Spektralkoeffizienten $\{a_i\}$ und $\{b_i\}$ sind eine gleichwertige Darstellung des Zeitsignals $x(t)$

[1] Naja, fast alle!

Beispiele für die Fourier-Synthese





Je schärfere Ecken ein Signal hat, desto höhere Frequenzen müssen bei der Überlagerung aus Sinus- und Kosinussignalen berücksichtigt werden.

Die richtige Mischung

Frage: Welchen Wert haben denn die Spektralkoeffizienten?

Antwort: Um so größer, je ähnlicher das untersuchte Signal der entsprechenden Oberwelle ist.

Auf mathematisch:

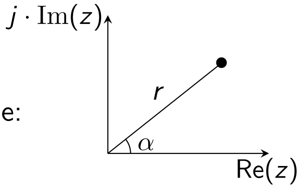
Fourier-Analyse

$$a_k = \frac{2}{T_0}\int_0^{T_0} x(t)\cos(k\omega_0 t)dt \qquad k = 0\dots\infty$$

$$b_k = \frac{2}{T_0}\int_0^{T_0} x(t)\sin(k\omega_0 t)dt \qquad k = 1\dots\infty$$

Ganz schön komplex ...

Komplexe Zahlenebene:



$$e^{j\alpha} = \cos\alpha + j\sin\alpha$$

Andersrum:

$$\cos\alpha = \frac{1}{2}\left(e^{j\alpha} + e^{-j\alpha}\right), \quad \sin\alpha = \frac{1}{2j}\left(e^{j\alpha} - e^{-j\alpha}\right)$$

Betrag und Phase

Man kann eine komplexe Zahl durch die reellen und imaginären Koordinaten beschreiben:

$$c = a + jb$$

a ist der Realteil von c
a ist der Imaginärteil von c

Alternativ:
Beschreibung durch Entfernung vom Nullpunkt und Winkel zur reellen Achse

$$c = re^{j\alpha}$$

r ist der Betrag von c
α ist die Phase von c

Exponentialfunktion statt Sinus und Kosinus

Komplexe Spektralkoeffizienten

Synthese:

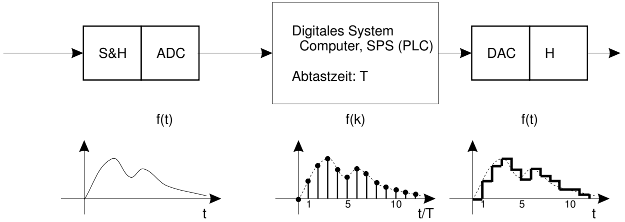
$$x(t) = \sum_{k=-\infty}^{+\infty} c_k e^{jk\omega_0 t}$$

Analyse

$$c_k = \frac{1}{T_0} \int_0^{T_0} x(t) e^{-jk\omega_0 t} dt$$

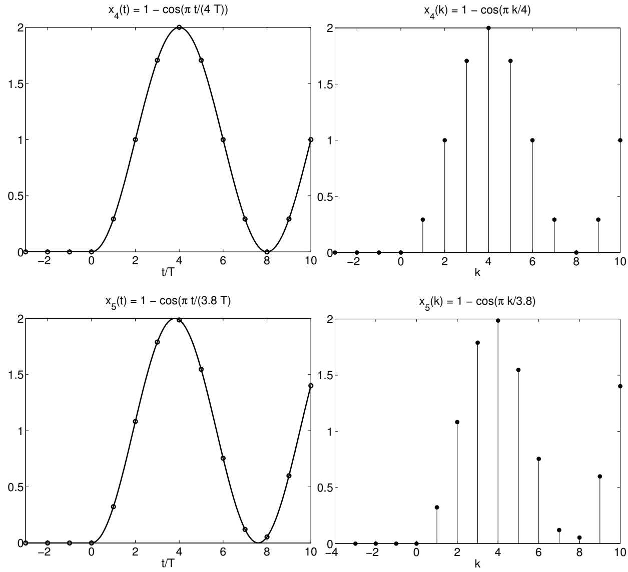
Hier sind die c_k komplexe Zahlen

Zeitdiskrete Signale



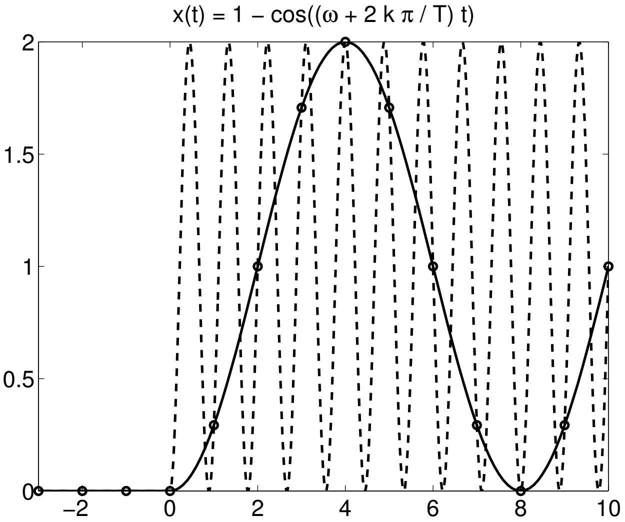
In Systemen zur digitalen Signalverarbeitung werden die Signale mit der Abtastfrequenz $f_a = 1/T_a$ abgetastet und im Wertebereich quantisiert

Abtastung



$x_4(t) = 1 - \cos(\pi\, t/(4\, T))$
t/T
$x_4(k) = 1 - \cos(\pi\, k/4)$
k
$x_5(t) = 1 - \cos(\pi\, t/(3.8\, T))$
t/T
$x_5(k) = 1 - \cos(\pi\, k/3.8)$
k

Aliasing



Die Nyquist-Frequenz

Vermeiden von Aliasing

Aliasing tritt nicht auf, wenn alle Frequenzen, die das analoge Eingangssignal enthält kleiner sind als die halbe Abtastfrequenz

$$f_{\mathrm{Nyquist}} = \frac{1}{2} f_a$$

Für CD-Audio ist $f_a = 44100\ \mathrm{Hz}$.
Die Nyquist-Frequenz von 22050 Hz entspricht der Grenze des menschlichen Hörvermögens.

Zeitbegrenzte diskrete Signale

Betrachte N Abtastwerte, Abtastzeit ist T_a

Frage: Spektrale Zusammensetzung dieses Signals?

Antwort: Übertrage Fourier-Transformation in den diskreten Bereich.
Implizit: periodische Fortsetzung

Konsequenz: Frequenzauflösung $= \frac{1}{NT_a}$

Der diskrete Charme der Fourier-Transformation

Diskrete Fourier-Transformation DFT

$$F(k) = \sum_{n=0}^{N-1} f(n) e^{-j2\pi \frac{k}{N} n}$$

Inverse Diskrete Fourier-Transformation IDFT

$$f(n) = \frac{1}{N} \sum_{k=0}^{N-1} F(k) e^{j2\pi \frac{n}{N} k}$$

FFT: Ein schneller Algorithmus zur Berechnung der (I)DFT

Frequenzauflösung

Den N Werten der DFT entspricht der Frequenzbereich von 0 bis $1/T_a$. Die Frequenzauflösung ist also:

$$\Delta f = \frac{f_A}{N}$$

Beispiel: Frequenzbestimmung auf einen Halbton genau

$$f_a = 44100 \text{ Hz}, \quad f = 100 \text{ Hz} \quad \Delta f = 5 \text{ Hz}$$

Für die Bestimmung des Spektrums muss die Anzahl N der Abtastwerte so gross sein, dass mindestens eine volle Periode einer 5-Hz-Schwingung erfasst wird:

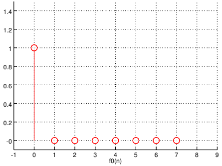
$$N = \frac{f_a}{\Delta f} = \frac{44100}{5} \approx 8192$$

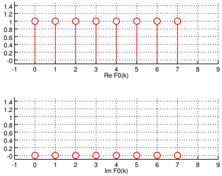
$\Rightarrow$ Kompromiss zwischen Messzeit und Frequenzauflösung

Spektrum eines Impulses

$$f_0(n) = \delta_{0\,n} := \begin{cases} 1 & \text{für} \qquad n = 0 \\ 0 & \text{sonst} \end{cases}$$

$$F_0(k) = 1$$





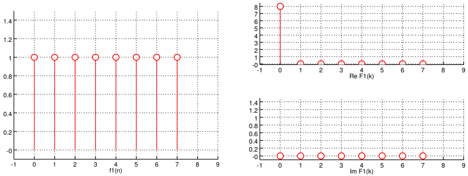
Spektrum eines konstanten Signals

$$f_1(n) = 1$$

$$F_1(k) = \delta_{0\,k}$$

denn $F(0)$ ist 1 und für $k \neq 0$ gilt die Formel für die geometrische Reihe $\sum_{n=0}^{N-1} a^n = \frac{1-a^N}{1-a}$

$$F(k) = \sum_{n=0}^{N-1} e^{j2\pi \frac{k}{N} n} = \frac{1 - e^{j2\pi \frac{k}{N} N}}{1 - e^{j2\pi \frac{k}{N}}} = 0 \quad \text{da der Zähler Null ist}$$



Verschobener Impuls

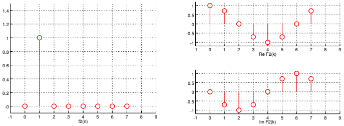
$$f_2(n) = \delta_{0\,(n-N_0)}$$

$$F_2(k) = e^{j2\pi \frac{k}{N} N_0}$$

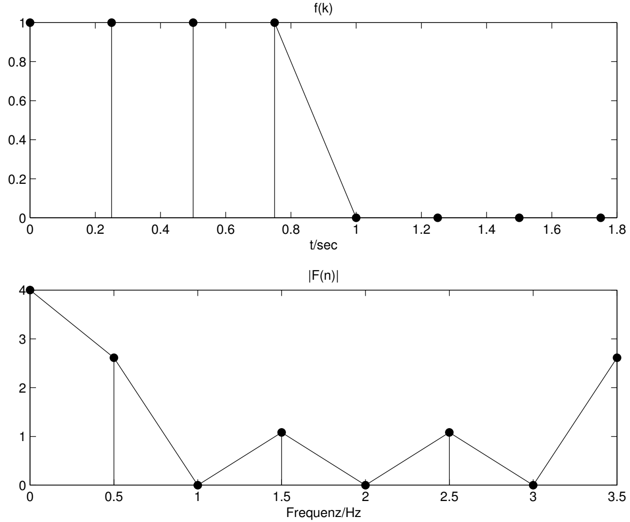
das kann man auch so schreiben, dann gilt es immer:

$$f_0(n) \overset{\mathcal{DFT}}{\Rightarrow} F_0(k)$$

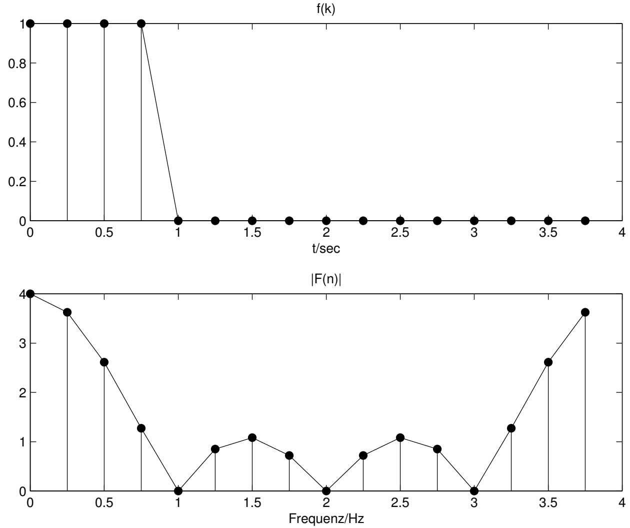
$$f_2(n) = f_0(n - N_0) \overset{\mathcal{DFT}}{\Rightarrow} F_2(k) = e^{j2\pi \frac{k}{N} N_0} \cdot F_0(k)$$



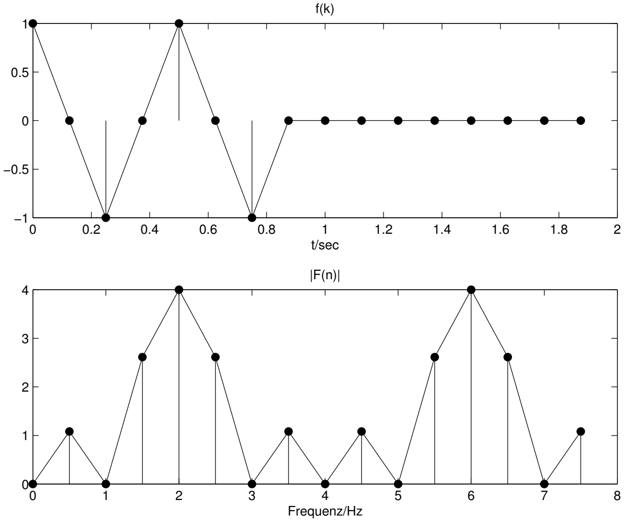
Spektrum eines Rechtecksignals



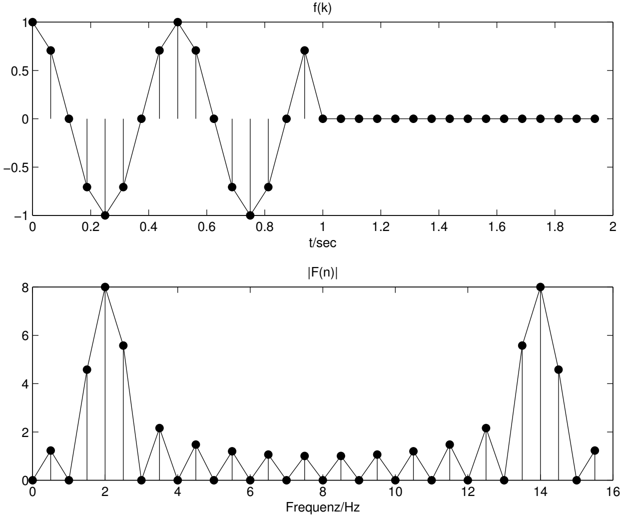
Bessere Frequenzauflösung durch zero-padding



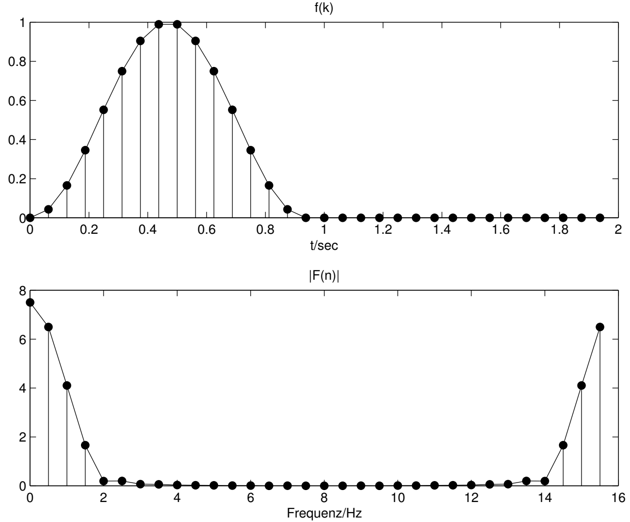
Begrenztes Kosinus-Signal



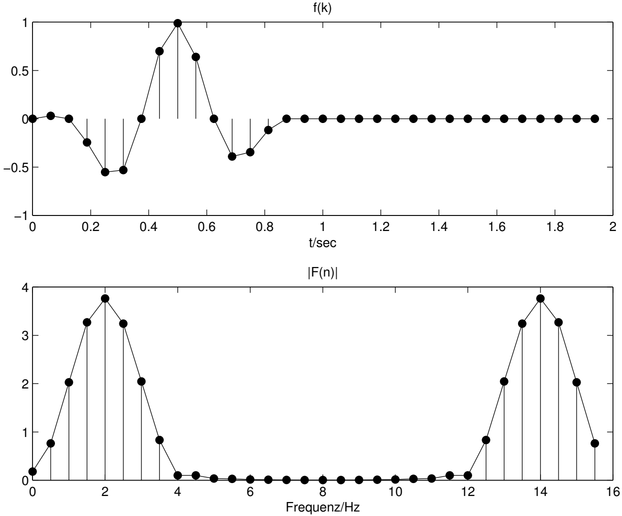
Verdoppeln der Abtastrate: Erhöhen der Grenzfrequenz



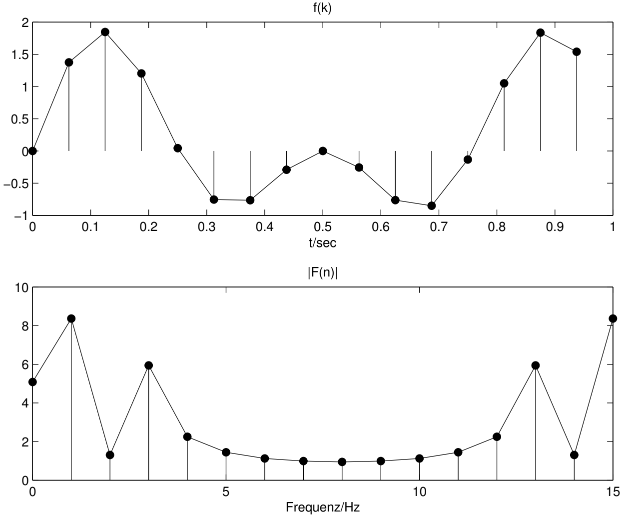
Hann-Fenster: weniger Artefakte bei hohen Frequenzen



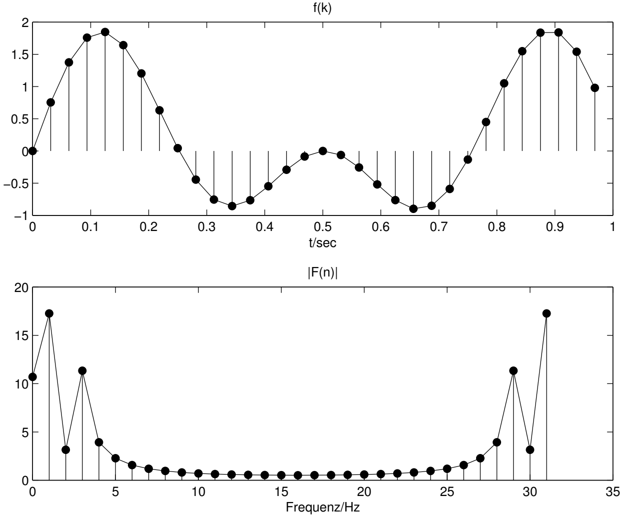
Kosinus-Signal mit Hann-Fenster: bessere Lokalisierung im Frequenzbereich



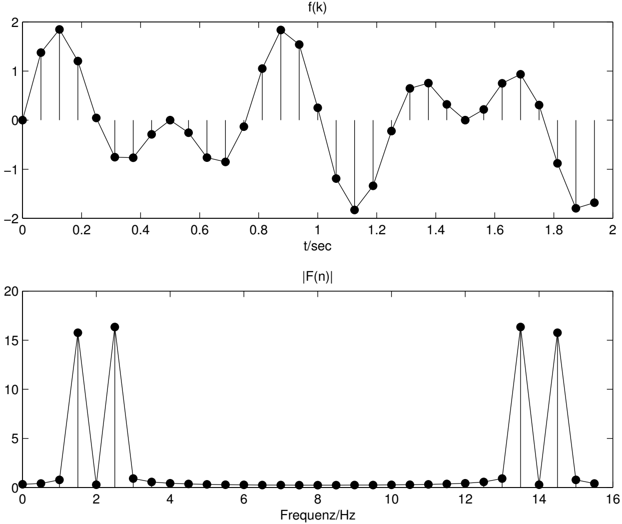
Überlagerung von zwei Sinussignalen mit $f_1 = 1,5$ Hz und $f_2 = 2,5$ Hz



Zwei Sinussignale: Halbierung der Abtastzeit ergibt keine bessere Frequenzauflösung



Zwei Sinussignale: Verdoppelung der Signaldauer verdoppelt die Frequenzauflösung



Filter

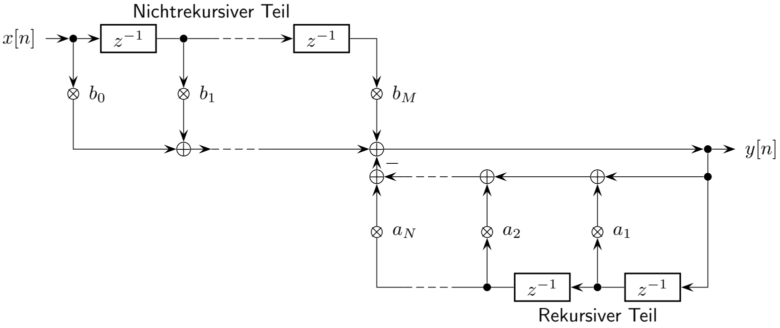
Wichtigste Klasse: lineare, zeitinvariante und kausale Filter
Differenzengleichung:

$$y[n] = b_0 x[n] + b_1 x[n-1] + \cdots + b_M x[n-M] - \\ a_1 y[n-1] - a_2 y[n-2] - \cdots - a_N y[k-N]$$

Aufgrund der Kausalität muss gelten: $M \geq N$
Gleichbedeutend: Übertragungsfunktion $H(z)$ des Filters:

$$H(z) = \frac{b_0 + b_1 z^{-1} + \cdots + b_M z^{-M}}{1 + a_1 z^{-1} + \cdots + a_N z^{-N}}$$

Filterstruktur: Direktform 1



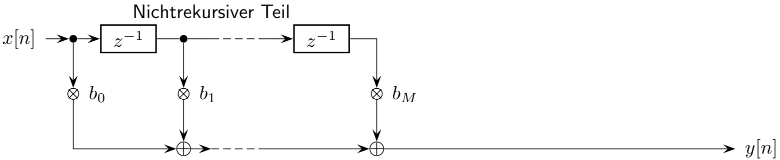
FIR / IIR

Wenn alle Filterkoeffizienten $a_1 \dots a_N$ im Nenner gleich Null sind, ist die Impulsantwort des Filters endlich und durch die Koeffizienten b_i gegeben:

FIR Finite Impulse Response

IIR Infinite Impulse Response

FIR-Filterstruktur: Direktform 1



Praktische Anwendungen: Matlab

- Audiosignal laden, speichern und abspielen
- Filterfunktion definieren
  - Mittelwertbildung
  - Tiefpass mit Design Toolbox
  - Aus Frequenzgang mit `remez`
- Filtern
- Frequenzgang anzeigen

Audiodaten bearbeiten

```
%% audio.m: Verarbeiten von Audiodateien
%% 1.) Audio einlesen
fn = 'Sounds/santa_maria.wav';
[y,fs,nbits] = wavread(fn);
fs, nbits %% Werte anzeigen
whos y    %% Dimension anzeigen
%% 2.) Audio abspielen
soundsc(y, fs);
%% 3.) Audiodaten modifizieren:
%%     linker Kanal: *0.9
%%     rechter Kanal: Rampe von 0 bis 1
tmp = y(:,1); %% linker Kanal
tmp = tmp * 0.9;
rampe = linspace(0,1, length(y));
whos rampe
%% Zeilenvektor in Spaltenvektor umwandeln
rampe = rampe.'; whos rampe
%% Elementweise multiplizieren
  %% Operator: ''.*'' !!
y(:,2) = y(:,2) .* rampe;
soundsc(y, fs);
%% Resultat ausgeben
fnout='Sounds/santa_maria_neu.wav';
wavwrite(y, fs, nbits, fnout);
ls Sounds/*.wav
```

Filter: Mittelwertbildung

Der aktuelle Ausgangswert soll der Mittelwert der N letzten Eingangswerte sein.
Das ist ein einfaches FIR-Filter mit den Koeffizienten

$$b_i = 1/N \quad \text{für } i = \{0, \ldots, N-1\}$$

Der erzeugte Plot zeigt, dass die hohen Frequenzen gedämpft werden.

FIR-Mittelwertfilter mit Matlab

```
%% mwfilt.m: Mittelwertfilter
fn = 'Sounds/santa_maria.wav';
[y,fs,nbits] = wavread(fn);
%% FIR-Filterkoeffizienten
N = 30; b = 1/N * ones(N,1);
%% Filtern
yfilt = filter(b, 1, y);
%% Vergleich der Betragsspektren
%% Stereo -> Mono
ymono = sum(y, 2); %% oder y(:,1) + y(:,2)
yfmono = sum(yfilt,2);

Y = abs(fft(ymono));
YF = abs(fft(yfmono));
f = fs *[0:length(Y) - 1]/length(Y); %% Frequenz
rng = 1:ceil(length(Y)/2) + 1;
Y = Y(rng); YF = YF(rng); f = f(rng); %% Untere Haelfte
plot(f, Y, f, YF); xlabel('f/Hz'); ylabel('rel. Amplitude');
legend('Ungefiltert', 'Gefiltert');
ax=axis; ax(1:2) = [0 8000]; axis(ax); %% Geht auch mit der Maus
%% Alternative Darstellung:
subplot(2,1,1); %% 2 Zeilen, 1 Spalte, 1. Feld
plot(f, Y); xlabel('f/Hz'); ylabel('rel. Amplitude'); title('Ungefiltert');
ax=axis; ax(1:2) = [0 8000]; axis(ax);
subplot(2,1,2); %% 2 Zeilen, 1 Spalte, 2. Feld
plot(f, YF); xlabel('f/Hz'); ylabel('rel. Amplitude'); title('Gefiltert');
ax=axis; ax(1:2) = [0 8000]; axis(ax);
subplot(1,1,1); %% Normalmodus
```

FIR-Filterentwurf im Frequenzbereich

Im nächsten Beispiel soll ein Filter mit einem vorgegebenen Frequenzgang entworfen werden.

- Dazu dient die Matlab-Funktion `fir2`
- Das Resultat wird mit Hilfe der Funktion `freqz` überprüft.
- Es gibt natürlich auch grafische Tools: `fdatool`, `filterbuilder`
- Die mit `fir2` berechneten Filter sind linearphasig. Die Matlab-Funktion `cfirpm` erlaubt es auch, die Phase explizit vorzugeben.

Filterentwurf mit `fir2`


```
%% fdesign.m: Filterentwurf im Frequenzbereich
 [y,fs,nbits] = wavread('Sounds/santa_maria.wav');
ord = 511; %% Filterordnung = Anzahl Koeffizienten - 1
%% Frequenzgang-Vorgabe
fv = [0 200 800 1000 2000 6000 fs ]/fs; %% Frequenzen
av = [5 100 500  800 1000 1000 800]/1000;%% Amplituden
b = fir2(ord, fv, av);
%% Frequenzgang
[h, w] = freqz(b); %% Ist-Frequenzgang
plot(w * fs/pi, abs(h), fv*fs, av );
%% Halblogarithmische Darstellung
semilogy(w * fs/pi, abs(h), fv*fs, av );
%% Doppeltlogarithmische Darstellung
loglog(w * fs/pi, abs(h), fv*fs, av );
%% Filtern
y = sum(y,2); %% Stereo -> Mono
yf = filter(b, 1, y);
```