

## Allgemein

Die **Digitale Signalverarbeitung (DSV)** spielt in den meisten elektronischen Geräten der Consumer-Elektronik, Telekommunikation, Regelungstechnik, Messtechnik, Audio/Video-Technik, ... sowie in vielen Sensor- bzw. Signal-bezogenen Software-Applikationen eine wichtige Rolle.

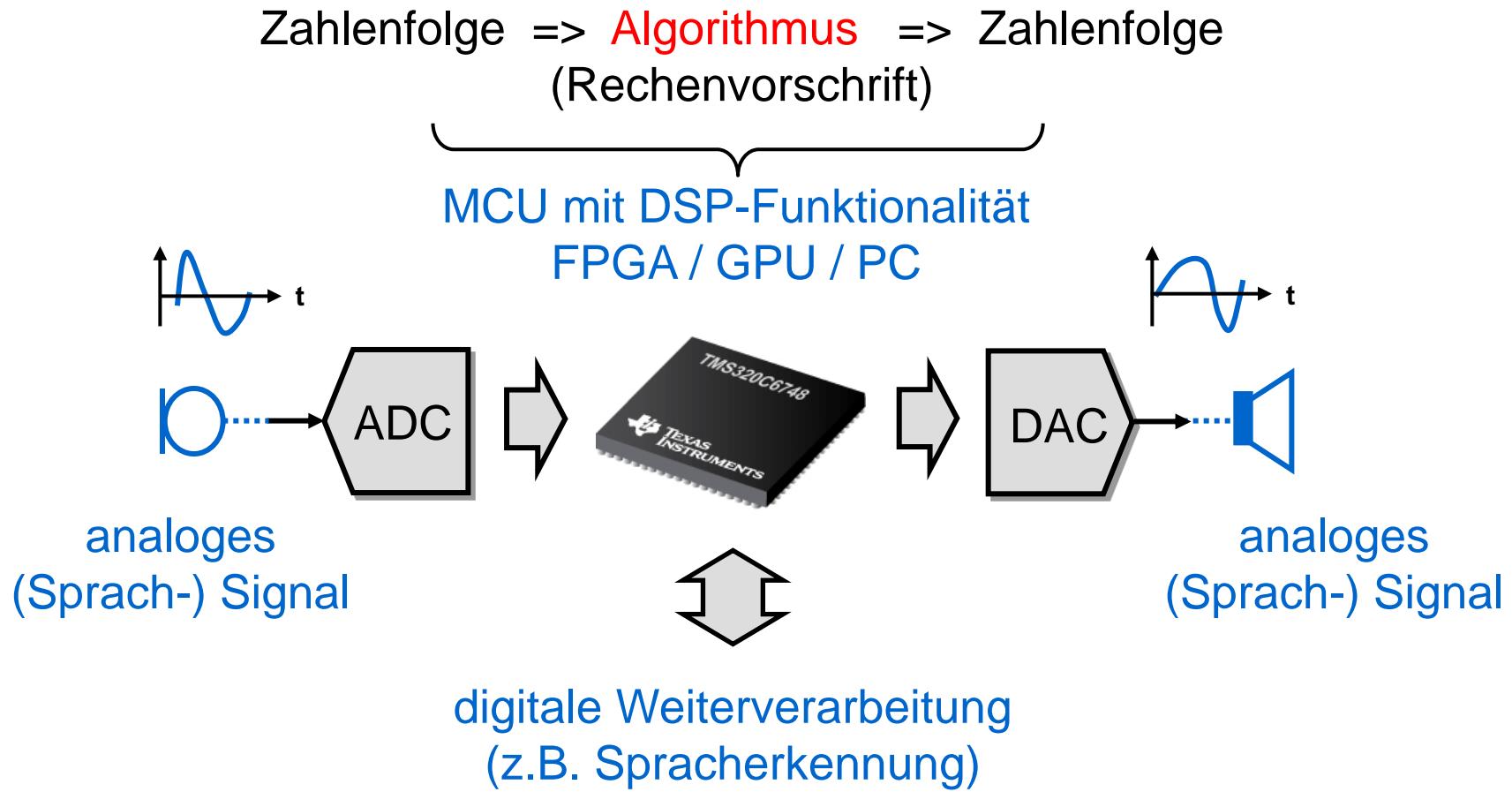
Die DSV dient auch der Vorverarbeitung (z.B. feature extraction) für Machine Learning, siehe z.B. [Beispiel 1](#).

Die DSV hat viele Vorteile. Dazu gehören sicherlich: Flexibilität, Reproduzierbarkeit, Realisierbarkeit von komplexen Algorithmen, HW-mässige und applikative Integrierbarkeit, meist gutes Kosten/Nutzen-Verhältnis, ...

## Bücher

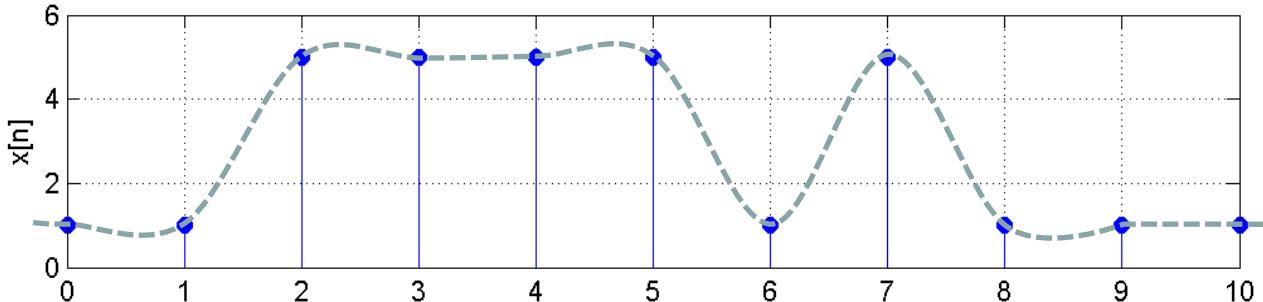
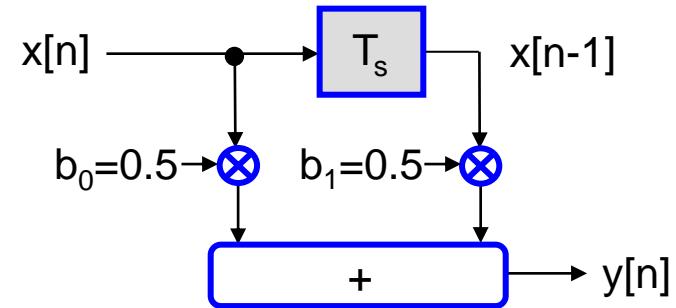
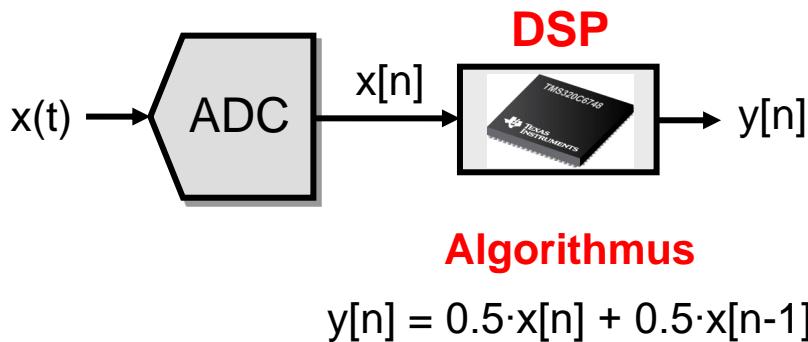
- [1] D.G. Manolakis, V.K. Ingle, "[Applied Digital Signal Processing](#)", Cambridge University Press, 2011.
- [2] A.V. Oppenheim, R.W. Schafer: „Discrete-Time Signal Processing“, 3rd Edition, Pearson, 2010. Oppenheim und Schafer sind DSP-Pioniere.

# Vorgang der digitalen Signalverarbeitung

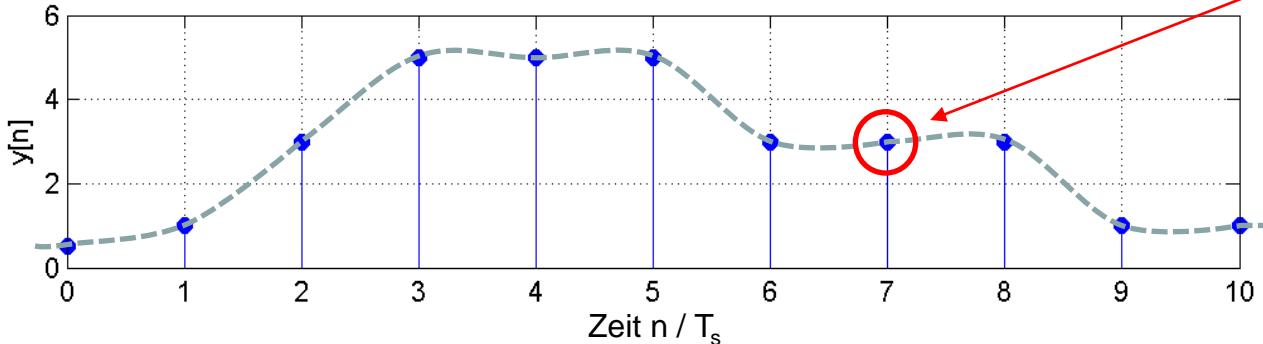


In diesem Kurs lernen wir die beiden Tätigkeiten  
**(Matlab-) Algorithmen-Entwurf** und **(DSP-) Realisierung** kennen.

# DSV-Beispiel (I)

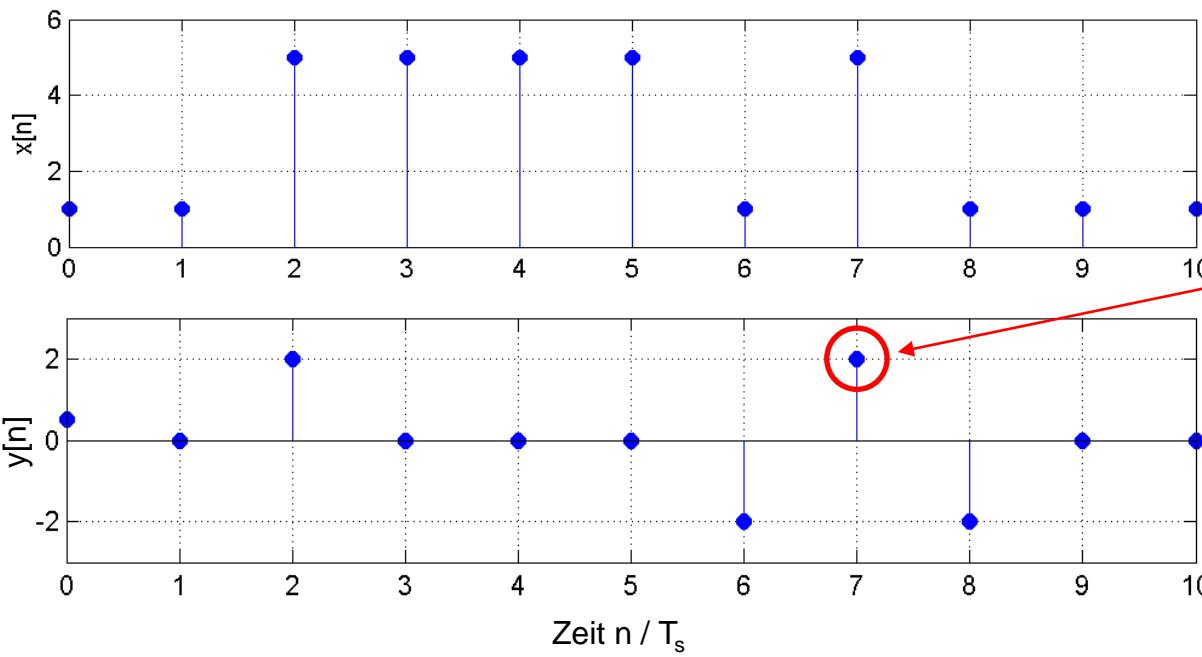
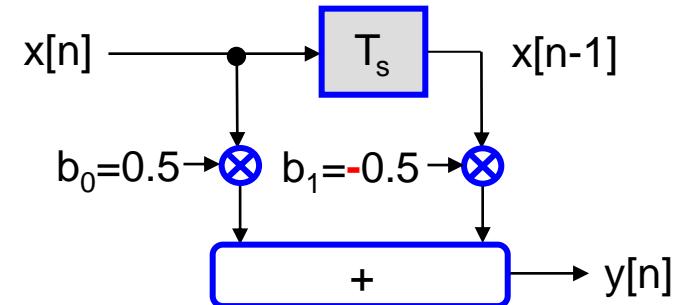
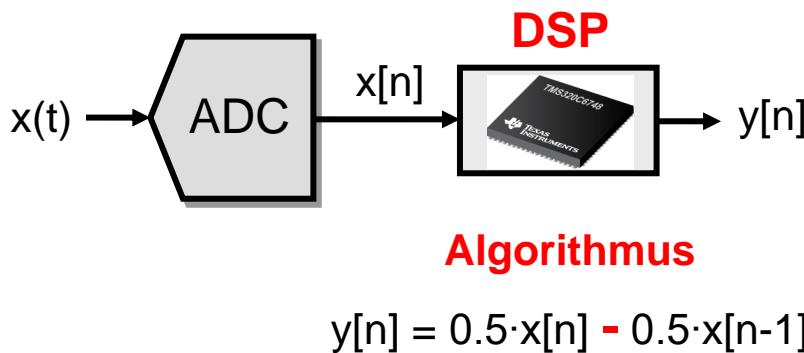


$$\begin{aligned} y[7] &= (x[7] + x[6]) / 2 \\ &= (5+1)/2 = 3 \end{aligned}$$



wobei  $T_s$  das Abtastintervall darstellt

# DSV-Beispiel (II)



$$\begin{aligned} y[7] &= (x[7] - x[6]) / 2 \\ &= (5 - 1) / 2 = 2 \end{aligned}$$

wobei  $T_s$  das Abtastintervall ist

## Weitergehende Fragestellungen

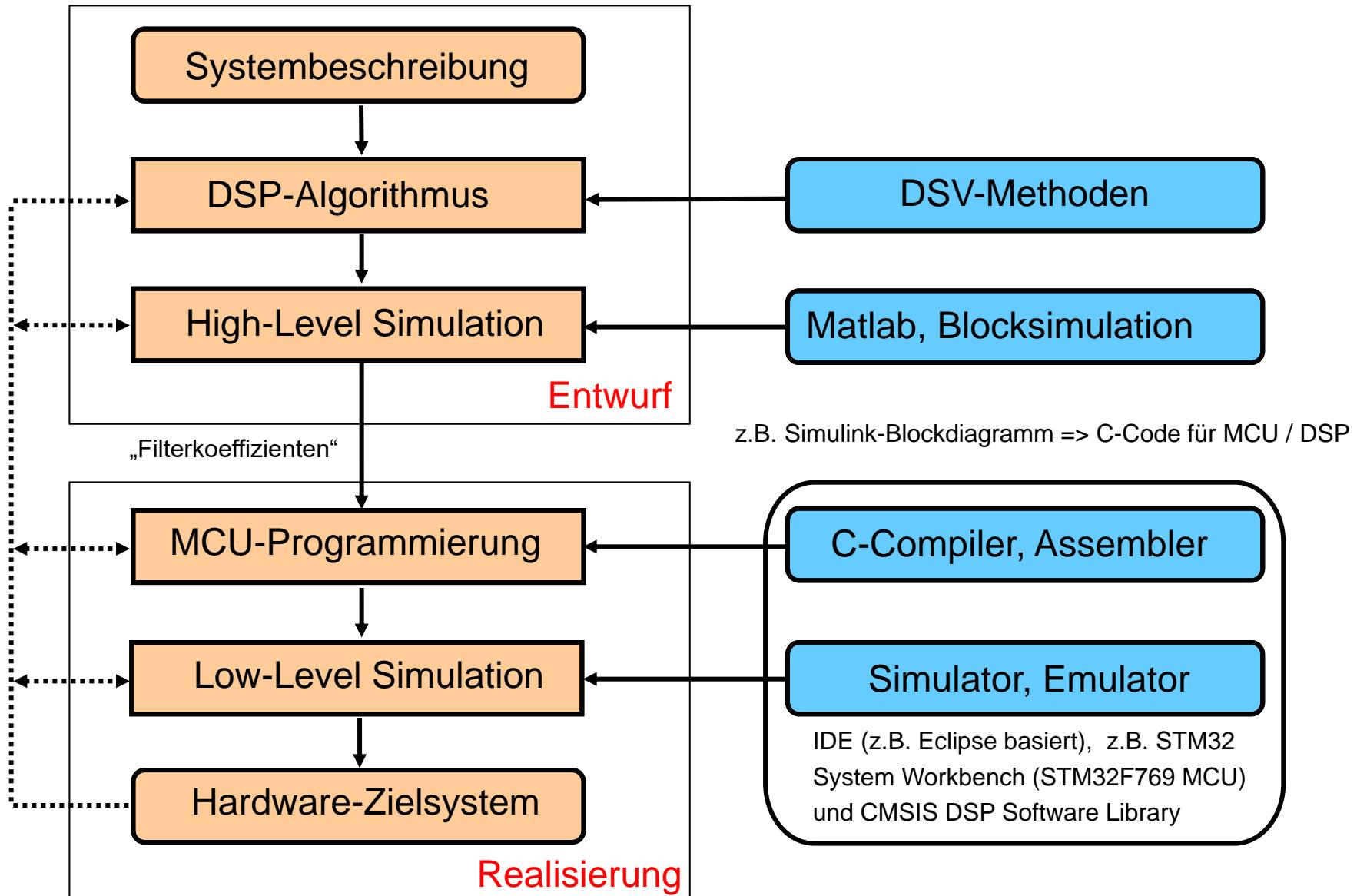
- Was machen die beiden Algorithmen genau (im **Zeit-** und im **Frequenzbereich**)?
- Welche Wirkung hätten andere **Systemparameter**  $b_0$  und  $b_1$ , und wie findet man sie?
- Gibt es auch **rekursive** Systeme, bei denen der Ausgangswert  $y[n]$  nicht nur von alten Eingangswerten  $x[n-1]$ , sondern auch von alten Ausgangswerten abhängt, z.B.  $y[n] = 0.1 \cdot x[n] + 0.9 \cdot y[n-1]$ ?

Welche Vor- und Nachteile haben (nicht-) rekursive Systeme?

- Welche Systeme lassen sich mit **Differenzengleichungen**, d.h. nur mit **Verzögerungselementen**, **Multiplikationen** und **Additionen**, beschreiben? Wie unterscheiden sich Differenzen- und Differentialgleichungen?

# DSV-Entwicklungsphasen

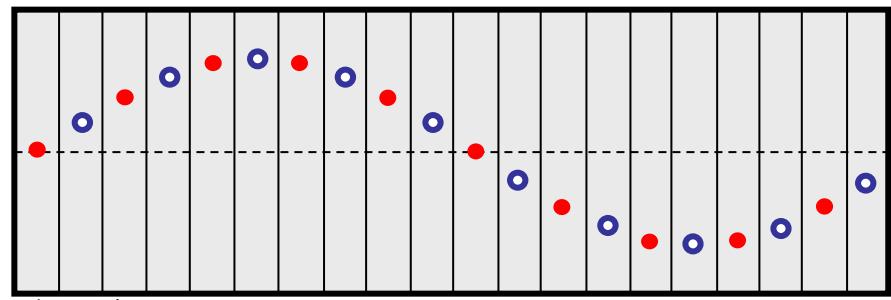
Quelle: G. Doblinger, „Signalprozessoren“, J. Schlembach Fachverlag, 2000.



# Direkte Digitale Synthese (DDS)

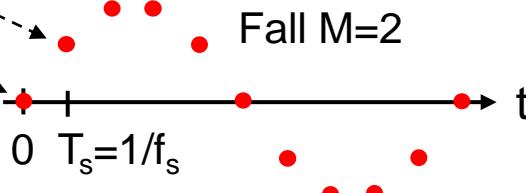
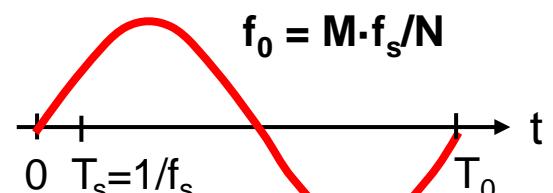
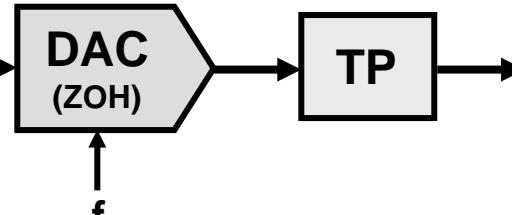
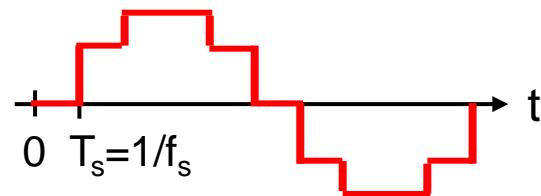
## Numerical Controlled Oscillator (NCO)

Lookup-Tabelle mit  
N Werten einer Sinus-Periode



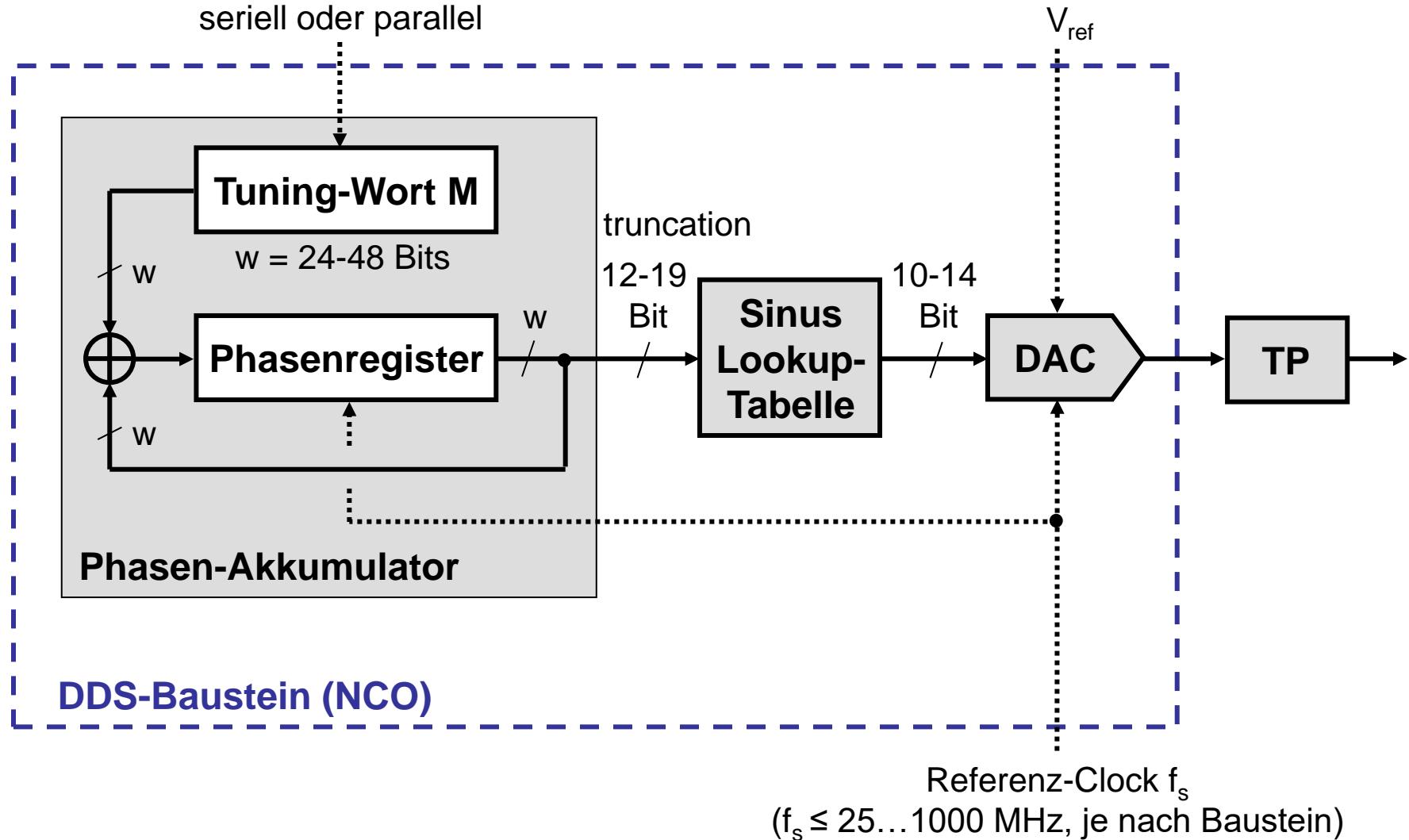
$$\text{Phase}[n] = (\text{Phase}[n-1] + M) \bmod N$$

(Phase entspricht Adresse)

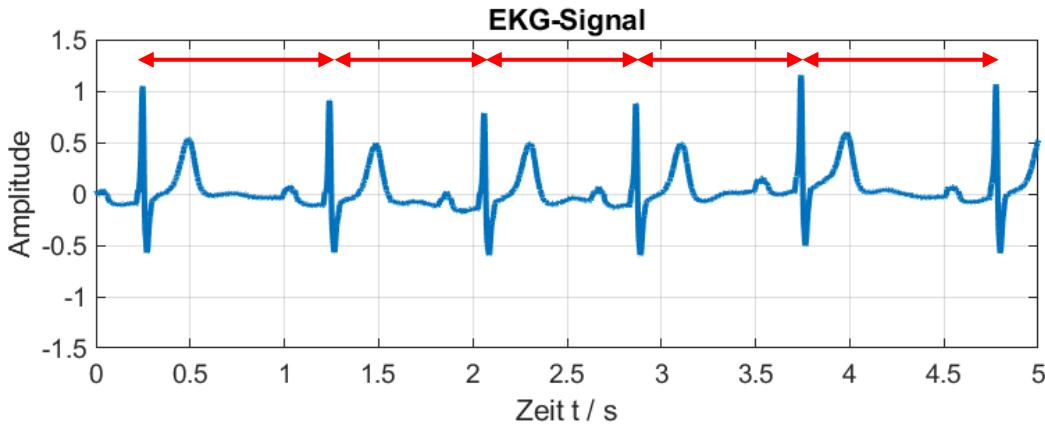


Fall  $M=2$

# DDS-Blockdiagramm



# DSV-Beispiel: EKG



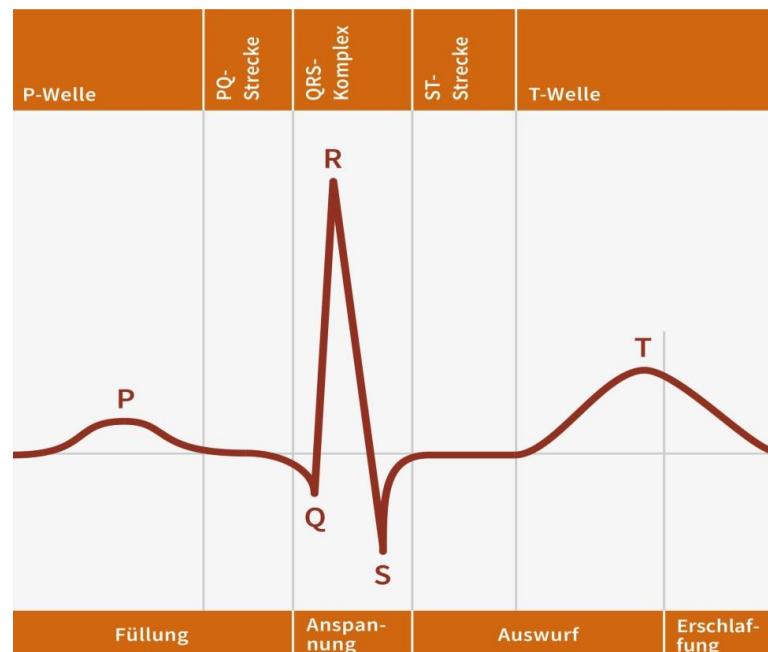
gesunde erwachsene Person,  
Ruhezustand ( $f_s = 500$  Samples/s)

↔ RR- / Herzschlag-Intervalle

Aus dem mittleren Abstand der R-Zacken  
kann die **Herzfrequenz** bestimmt werden.

Hier genügt ein einfacher Schwellwert-  
Detektor, aber was ist, wenn die T-Welle  
eine höhere Amplitude als die R-Zacke  
aufweist?

**Signalanteile / Merkmale**  
(im gesunden EKG)



# DSV-Beispiel: Hörgeräte

Quelle: Dr. S. Wyrsch

## Kennzahlen:

- DSP/MCU typ. 1 mW Verbrauch (ca. 1 mA an ca. 1.2V)
- DSP-Taktraten im Bereich von einigen MHz bis ca. 50 MHz
- Ausgangsfrequenzbereich bis ca. 16 kHz
- Verzögerung durch gesamtes System < 10 ms

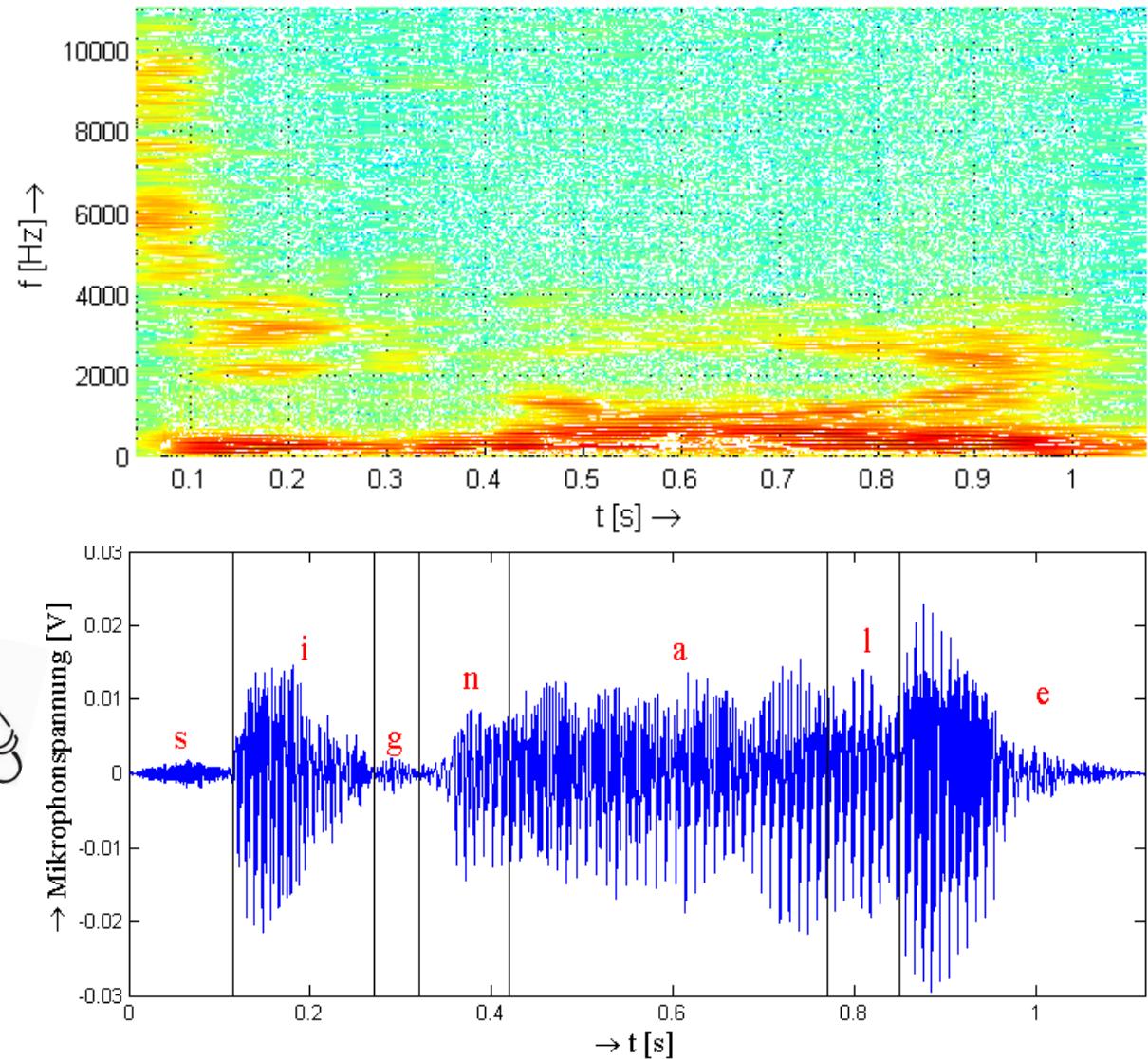


Bilder: Bernafon & Phonak

# DSV-Beispiel: Spektrogramm Sprache

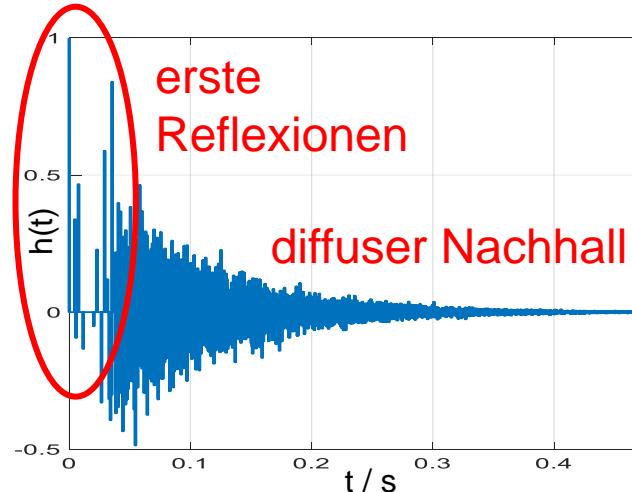
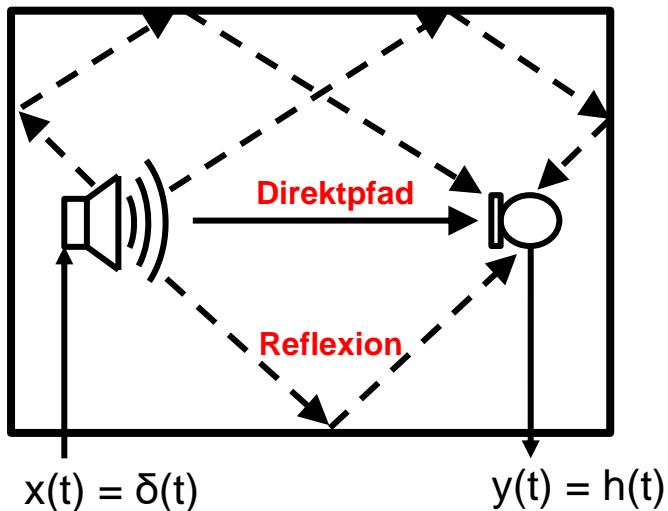


Sprecher/Quelle:  
Dr. S. Wyrtsch, ZHAW



# DSV-Beispiel: Hall-Filterung

- 1) Messung der Raum-Impulse-Response (RIR) z.B. mit einem Knall.



- 2) Studioaufnahme (trocken, reflexionsarm):  
=> Suzanne Vega singt im Studio «Tom's Diner»



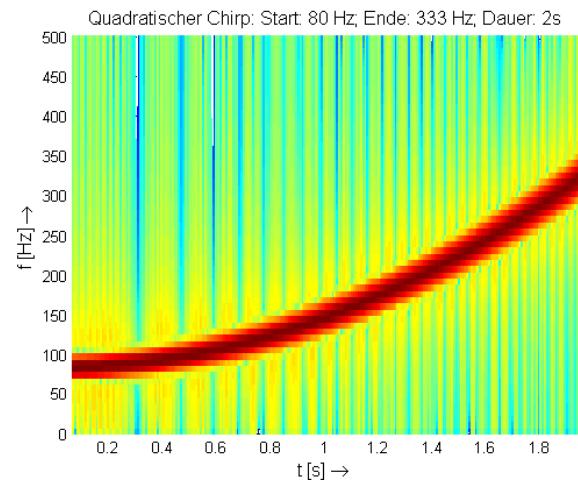
- 3) Filterung Studioaufnahme mit RIR  
Suzanne Vega singt (vermeintlich)  
in der Kirche.



Die Diskrete Fourier-Transformation (DFT) spielt in der Digitalen Signalverarbeitung (DSV) eine zentrale Rolle, weil viele Probleme elegant im Frequenzbereich gelöst werden können, und weil es den schnellen FFT-Algorithmus zur Berechnung der DFT-Werte gibt, der den Aufwand von  $N^2$  auf  $N \cdot \log_2(N)$  komplexe Multiplikationen reduziert.

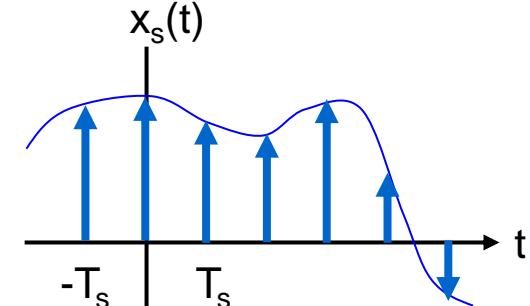
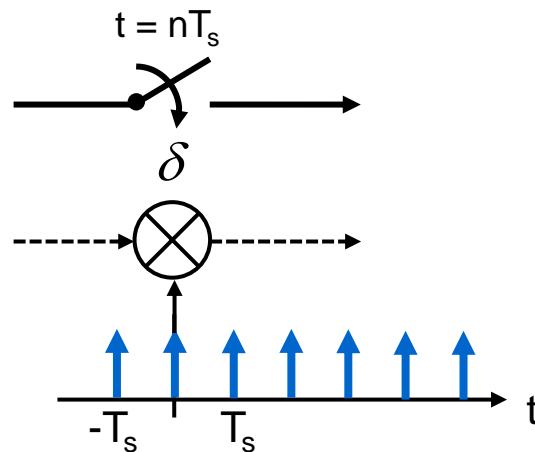
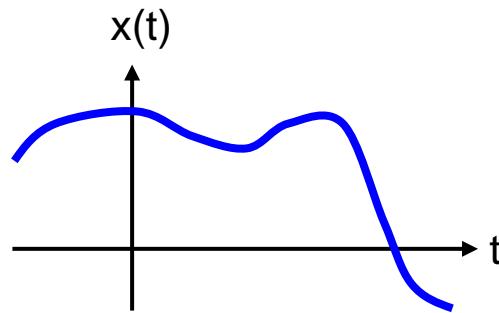
## Inhalt

- Repetition Abtastung
- DFT und Eigenschaften
- Verwandtschaft DFT mit Fourier-Reihe
- Leakage und Fensterung (Windowing)
- Filterfunktion der DFT
- Schnelle Fourier-Transformation (FFT)
- FFT von reellwertigen Signalen
- Zero Padding
- Spektrogramme



# Repetition der idealen Abtastung

siehe auch Anhang «Repetition ADC/DAC»



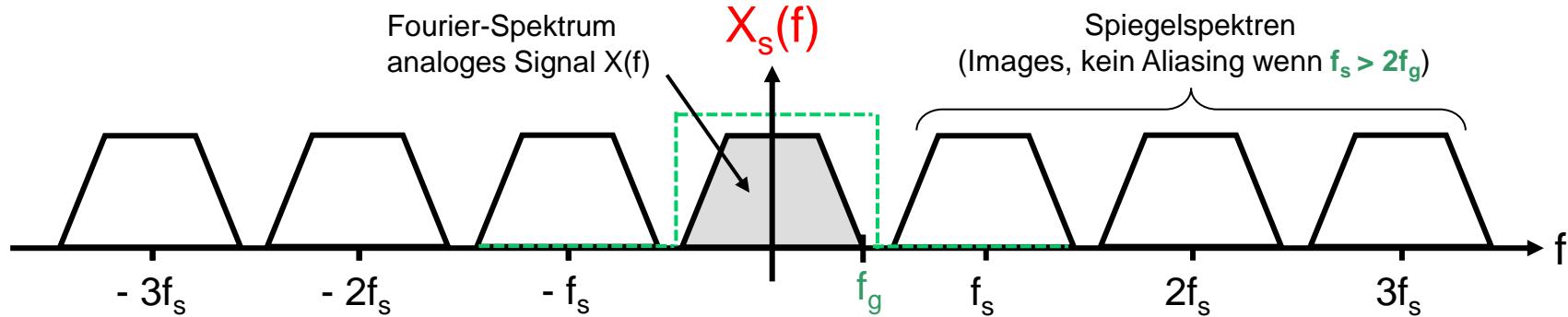
## Zeitdiskretes Signal

$$x_s(t) = \sum_{n=-\infty}^{\infty} x(nT_s) \cdot \delta(t - nT_s)$$



## Fourier-Spektrum

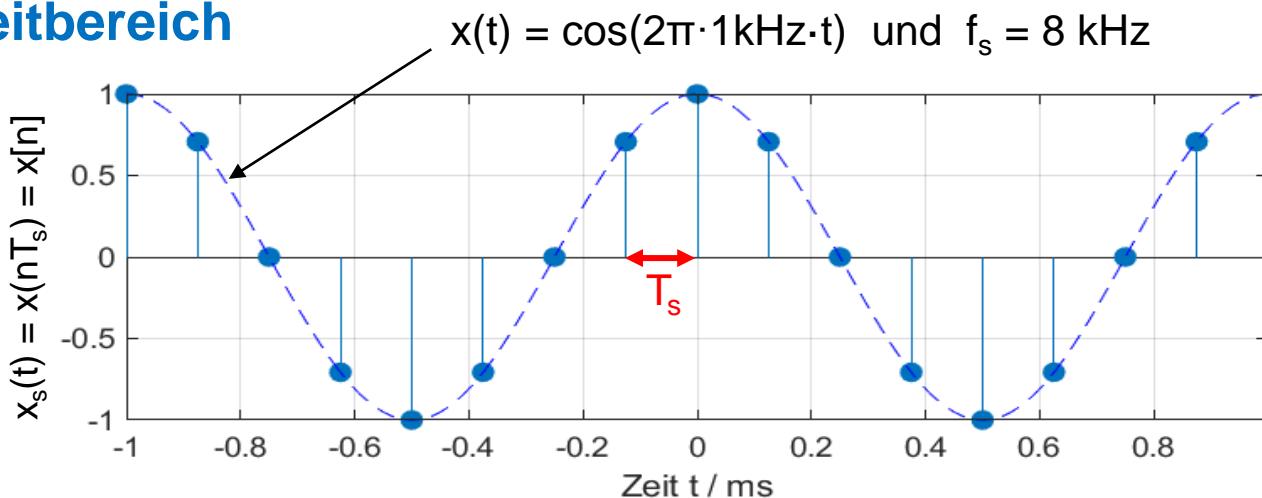
$$X_s(f) = \frac{1}{T_s} \cdot \sum_{n=-\infty}^{\infty} X(f - nf_s)$$



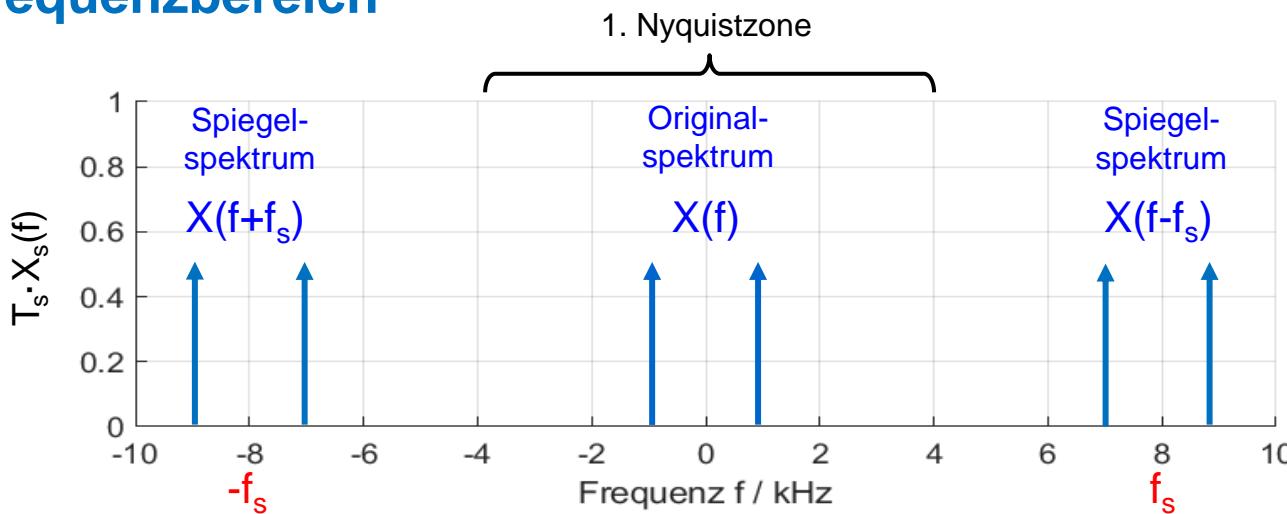
# Repetition ideale Abtastung: Beispiel

siehe auch Anhang «Repetition ADC/DAC»

## Zeitbereich



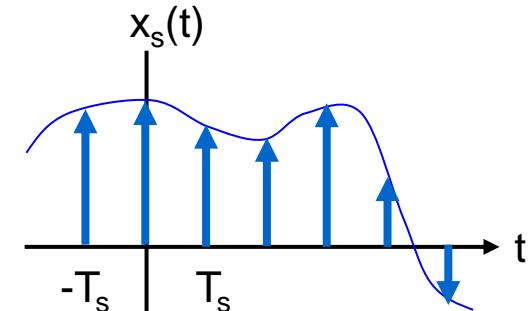
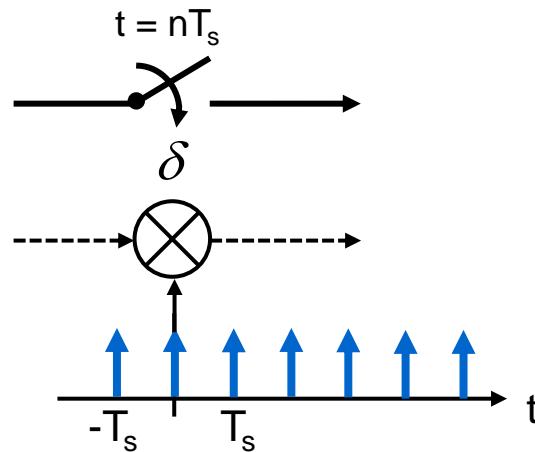
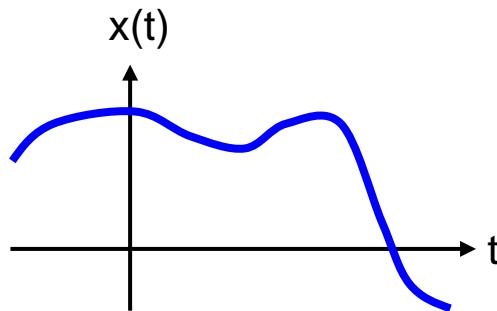
## Frequenzbereich



...

# Ideale Abtastung: Alternativer Ausdruck

siehe auch Anhang «Repetition ADC/DAC»



## Zeitdiskretes Signal

$$x_s(t) = \sum_{n=-\infty}^{\infty} \underbrace{x(nT_s)}_{x[n]} \cdot \delta(t - nT_s)$$

FT  
Linearität  
Zeitverschiebung  
 $\text{FT}\{\delta(t)\} = 1$

## Fourier-Spektrum

$$X_s(f) = \sum_{n=-\infty}^{\infty} x[n] \cdot e^{-jn2\pi f T_s}$$

periodisch mit der  
Abtastfrequenz  $f_s$

Ideal abgetastetes Signal

$$x_s(t) = \sum_{n=-\infty}^{\infty} x(nT_s) \cdot \delta(t - nT_s)$$

Fourier-Spektrum zeitdiskretes Signal

$$X(f) = \sum_{n=-\infty}^{\infty} x[n] \cdot e^{-jn2\pi f T_s}$$

Approximation mit N Samples im Zeitfenster  $[0, NT_s]$      $X(f) \approx \sum_{n=0}^{N-1} x[n] \cdot e^{-jn2\pi f T_s}$

N äquidistante Spektralwerte  $X(f = m \cdot f_s / N)$  im Frequenzbereich  $[0, f_s]$

Diskrete Fourier Transformation

$$X[m] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j \frac{2\pi}{N} mn} \quad m = 0, 1, \dots, N-1$$

**DFT**

$$X[m] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j \frac{2\pi}{N} mn} \quad m = 0, 1, \dots, N-1$$

**IDFT**

inverse

$$x[n] = \frac{1}{N} \cdot \sum_{m=0}^{N-1} X[m] \cdot e^{j \frac{2\pi}{N} mn} \quad n = 0, 1, \dots, N-1$$



Die DFT berechnet aus

N Abtastwerten  $x[n]$  im Zeitfenster der Länge  $T_{DFT} = NT_s$

N äquidistante (komplexe) Spektralwerte im f-Bereich  $[0, f_s)$

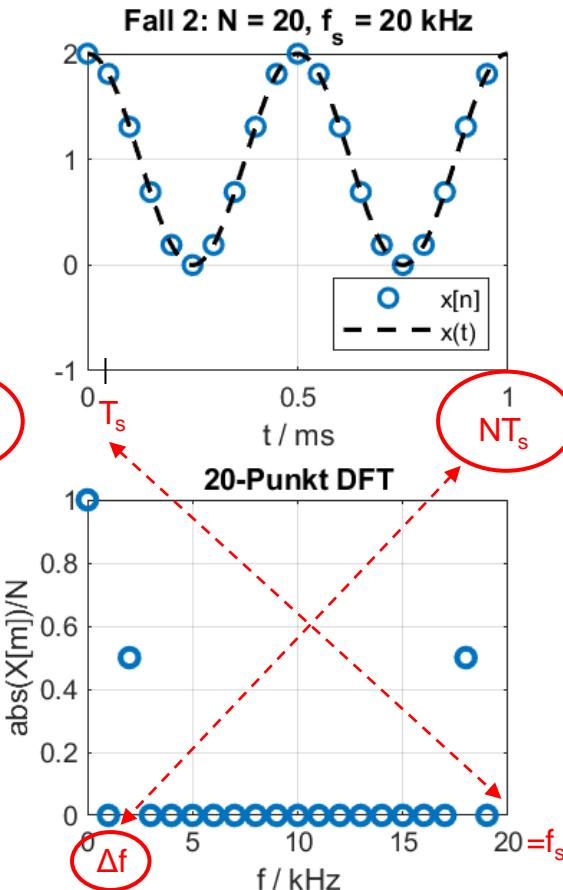
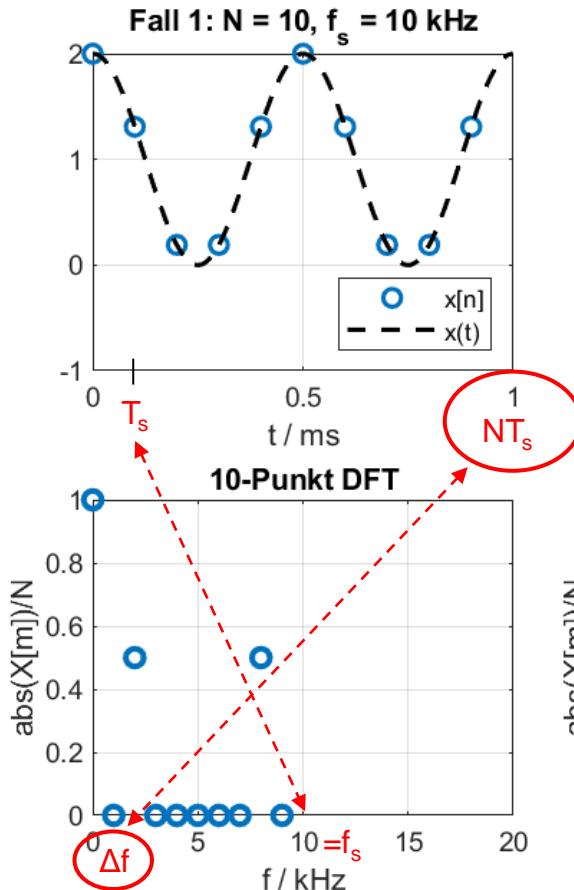
Die Frequenzauflösung  $\Delta f = f_s / N = 1 / NT_s = 1 / T_{DFT}$   
ist umgekehrt proportional zur Länge des Zeitfensters  $T_{DFT} = NT_s$

«Beobachtungszeit  $NT_s$ » mal «Frequenzauflösung  $f_s/N$ » = 1

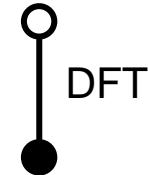
# DFT und Frequenzauflösung: Beispiel

Normiertes DFT-Betragsspektrum  $|X[m]|/N$  des Signals

$$x[n] = 1 + \cos(2\pi \cdot f_0 \cdot n T_s), \quad n = 0, \dots, N-1, \quad f_0 = 2 \text{ kHz}, \quad \text{wenn}$$



N Samples im  
Zeitfenster  $[0, T_{DFT} = 1 \mu\text{s}]$   
 $\text{ms}$



N DFT-Spektralwerte  
im Frequenzbereich  $[0, f_s]$

Die Frequenzauflösung  
 $\Delta f = f_s / N = 1 / NT_s = 1 / T_{DFT}$   
ist in beiden Fällen  $\Delta f = 1 \text{ kHz}$ ,  
weil  $T_{DFT} = 1 \mu\text{s}$ .  
 $\text{ms}$

# Berechnung DFT mit Matlab

siehe auch Matlab\_Basics\_in\_90min.pdf

t = [0:N-1]\*Ts; % diskreter Zeitvektor [0\*Ts, 1\*Ts, ..., (N-1)\*Ts]

x = cos(2\*pi\*f0\*t); % zeit-diskretes Signal x[n], hier cos-Signal,  
% bzw. Vektor [x(0), x(Ts), x(2\*Ts), ..., x((N-1)\*Ts)]

f = [0:N-1]\*fs/N; % diskreter Frequenzvektor [0 (DC), df, ..., (N-1)\*df]  
% mit Frequenzauflösung df = fs/N  
% Alternative: linspace(0,N-1,N)\*fs/N

X = fft(x)/N;  
ergibt N komplexe Werte % normiertes DFT-Spektrum  
% bzw. Vektor abs([X[0], ..., X[N-1]]) / N  
% (schnell) berechnet mit der FFT, siehe später

XdB = 20\*log10(abs(X)); % normiertes DFT-Betragsspektrum

phi\_X = unwrap(angle(X)); % DFT-Phasenspektrum in rad  
% «ohne» Phasensprünge (unwrap)

plot(f,XdB); % plot DFT-Betragsspektrum in dB vs. Frequenz

# Bedeutung der DFT-Komponenten

Definition DFT

$$X[m] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j \frac{2\pi}{N} mn} \quad m = 0, 1, \dots, N-1$$

"DC"-Komponente @  $0 \cdot f_s/N$

$$X[0] = \sum_{n=0}^{N-1} x[n] \cdot 1$$

"AC"-Komponente @  $1 \cdot f_s/N$

$$X[1] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j \frac{2\pi}{N} n} = \sum_{n=0}^{N-1} x[n] \cdot e^{-j \cdot 2\pi \frac{f_s}{N} n T_s}$$

Frequenz

Zeit

"AC"-Komponente @  $2 \cdot f_s/N$

$$X[2] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j \frac{2\pi}{N} 2 \cdot n}$$

oft betrachtet man die normierten DFT-Frequenzkomponenten  $X[m]/N$   
(dann entspricht die DFT-DC-Komponente dem DC-Wert des Signals)

# Matrix-Darstellung der DFT

Beispiel N=4 Punkt DFT:  $X[m] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j\frac{2\pi}{N}mn}$   $m = 0, 1, \dots, N-1$

Spektral- werte	DFT-Transformations-Matrix	Zeit- werte
$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \end{bmatrix}$	$= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & e^{-j\frac{2\pi}{4}1} & e^{-j\frac{2\pi}{4}2} & e^{-j\frac{2\pi}{4}3} \\ 1 & e^{-j\frac{2\pi}{4}2} & e^{-j\frac{2\pi}{4}4} & e^{-j\frac{2\pi}{4}6} \\ 1 & e^{-j\frac{2\pi}{4}3} & e^{-j\frac{2\pi}{4}6} & e^{-j\frac{2\pi}{4}9} \end{bmatrix} \cdot \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \end{bmatrix}$	

$X[0]$  = Projektion des Zeitvektors  $x$  auf DC-Signal [1 1 1 1] (Skalarprodukt)

$X[1]$  = Projektion Zeitvektor  $x$  auf komplexe Schwingung der Frequenz  $f_s/4$

$X[2]$  = Projektion Zeitvektor  $x$  auf komplexe Schwingung der Frequenz  $f_s/2$

$X[3]$  = Projektion Zeitvektor  $x$  auf komplexe Schwingung der Frequenz  $-f_s/4$

# Matrix-Darstellung der DFT

Quelle: Dr. S. Wyrsc

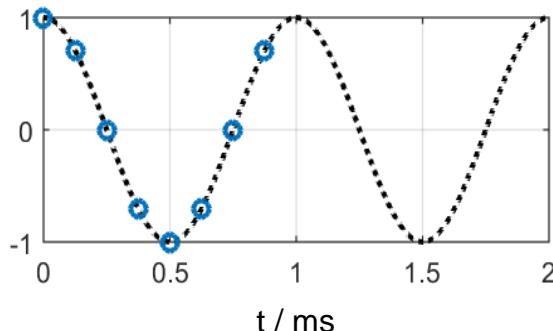
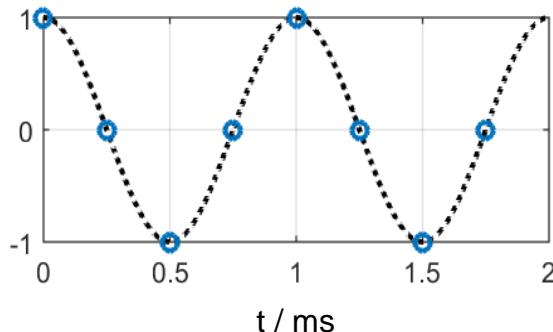
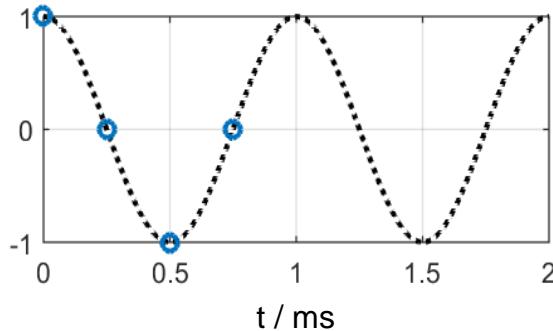
## Beispiel 8-Punkt DFT (mit komplexen Zeigern)

$$X[m] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j \frac{2\pi}{N} mn} \quad m = 0, 1, \dots, N-1$$

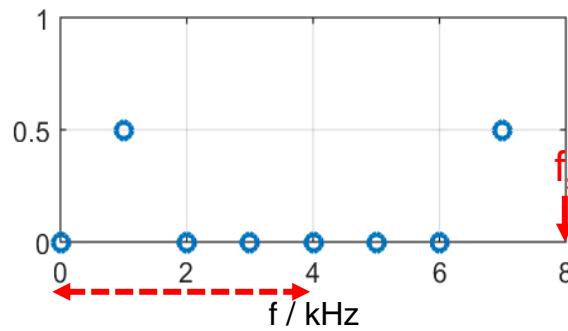
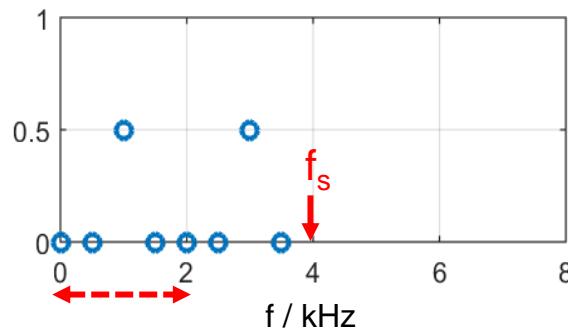
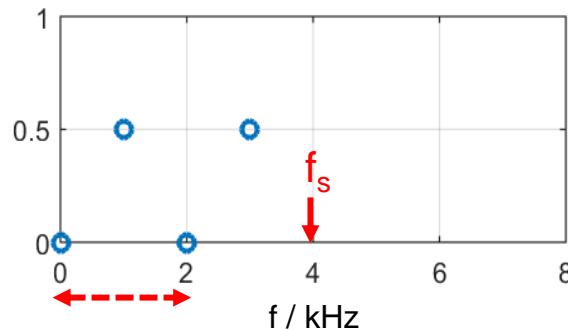
$$\begin{bmatrix} X[0] \\ X[1] \\ \vdots \\ X[7] \end{bmatrix} = \begin{bmatrix} \text{circle with arrow} & \text{circle with arrow} \\ \text{circle with arrow} & \text{circle with arrow} \\ \text{circle with arrow} & \text{circle with arrow} \\ \text{circle with arrow} & \text{circle with arrow} \\ \text{circle with arrow} & \text{circle with arrow} \\ \text{circle with arrow} & \text{circle with arrow} \\ \text{circle with arrow} & \text{circle with arrow} \\ \text{circle with arrow} & \text{circle with arrow} \end{bmatrix} \cdot \begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[7] \end{bmatrix}$$

# DFT-Spektrum: Beispiele

Zeitsignale (gegeben)



DFT-Spektren (gesucht)



wichtig

$$N = 4, f_s = 4 \text{ kHz}$$

$$\Delta f = f_s / N = 1 \text{ kHz} \text{ bzw.}$$

$$\Delta f = 1 / T_{\text{DFT}} = 1/(1 \text{ ms})$$

$$N = 8, f_s = 4 \text{ kHz}$$

$$\Delta f = f_s / N = 0.5 \text{ kHz} \text{ bzw.}$$

$$\Delta f = 1 / T_{\text{DFT}} = 1/(2 \text{ ms})$$

$$N = 8, f_s = 8 \text{ kHz}$$

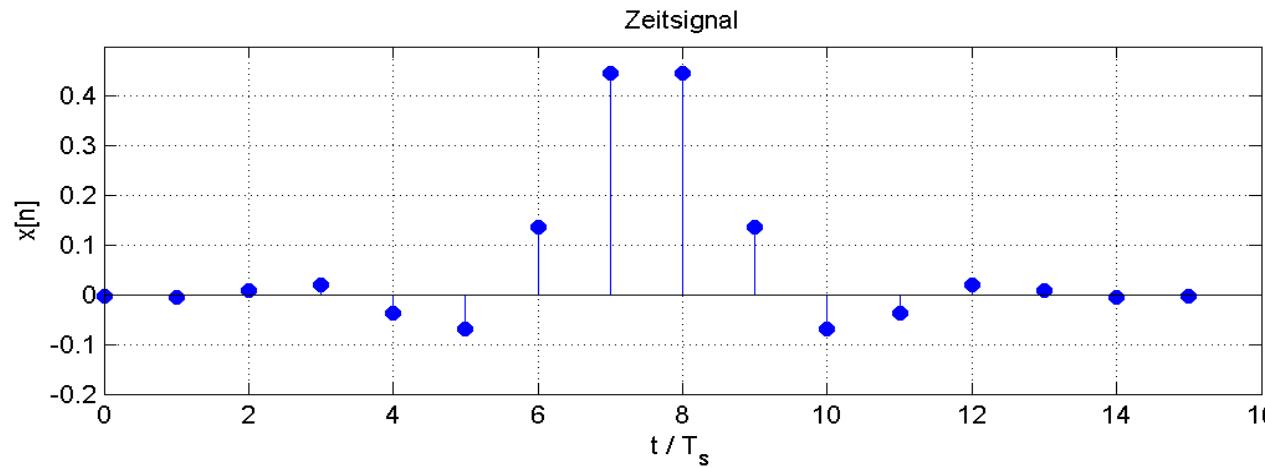
$$\Delta f = f_s / N = 1 \text{ kHz} \text{ bzw.}$$

$$\Delta f = 1 / T_{\text{DFT}} = 1 / (1 \text{ ms})$$

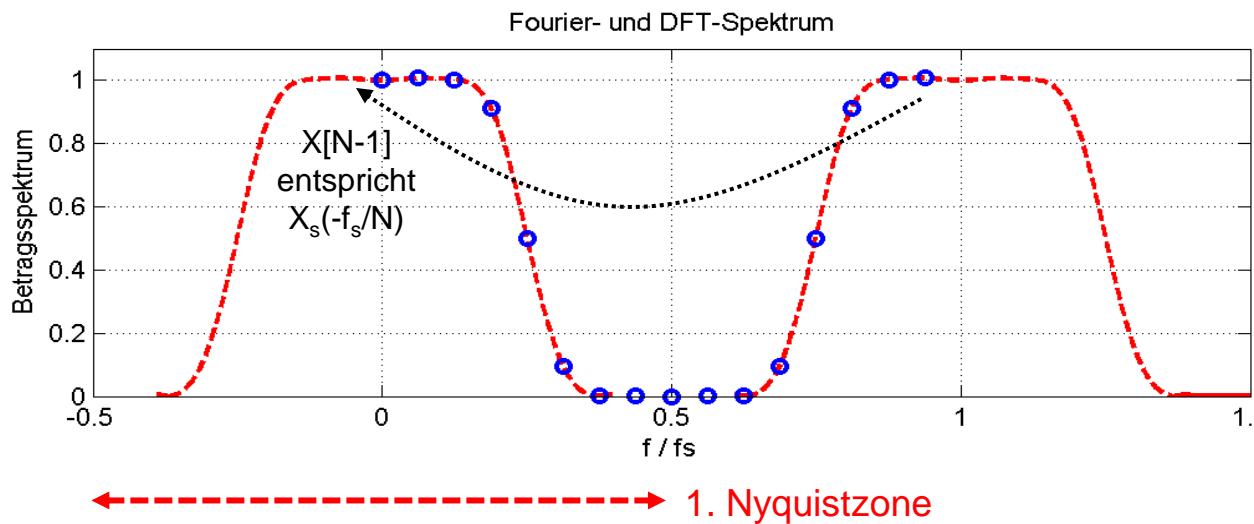
↔ 1. Nyquistzone

# DFT-Approximation des Fourier-Spektrums von einem zeitdiskreten Signal

Das **DFT-Spektrum  $X[m]$**  approximiert das **Fourier-Spektrum  $X_s(f)$**  des zeitdiskreten Signals  $x[n]$  an  $N$  äquidistanten Punkten im Frequenzbereich  $[0, f_s]$



«Puls»  
mit  $N = 16$   
Samples



-- (normiertes)  
Fourier(betrags-)  
spektrum  $T_s \cdot X_s(f)$

○ 16-Punkt DFT-  
Spektrum  $X[m]$   
( $\Delta f = f_s / 16$ )

Jedes periodische Signal  $x(t)$  kann als komplexe Fourierreihe (FR) wie folgt dargestellt werden:

$$x(t) = \sum_{k=-\infty}^{\infty} c_k \cdot e^{j2\pi k f_0 t}$$

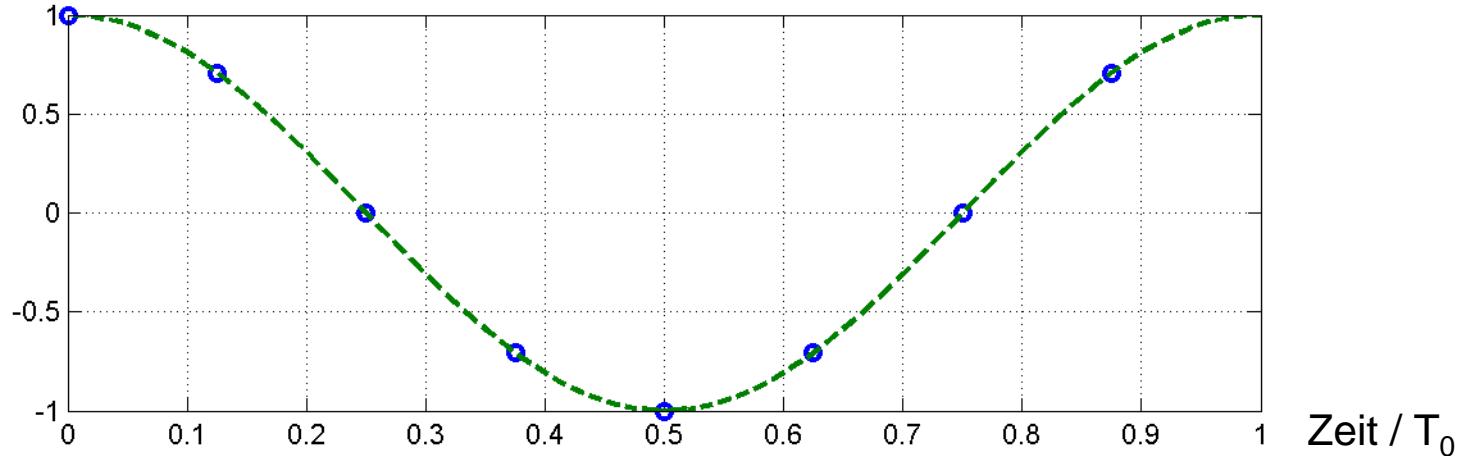
Wenn  $x(t)$  keine Harmonischen  $> f_s/2$  aufweist, d.h. beim Abtasten kein Aliasing entsteht, können mit einer N-Punkt DFT alle  $c_k$ -Koeffizienten «aufs Mal» wie folgt berechnet werden:

- Nehme N Samples von 1 Periode  $T_0 = 1/f_0$  von  $x(t)$   
=> Abtastintervall  $T_s = T_0/N$  bzw. Abtastfrequenz  $f_s = N \cdot f_0 = 1/T_s$
- Erhalte mit DFT N äquidistante Spektralwerte  $X[m]$  im Bereich  $[0, f_s]$   
=> Frequenzauflösung  $\Delta f = f_s/N = f_0$   
  
Die normierten DFT-Spektralwerte  $X[m]/N$ ,  $m = 0 \dots N/2$ , stimmen mit den komplexen  $c_m$ -Koeffizienten der FR von  $x(t)$  überein, d.h.

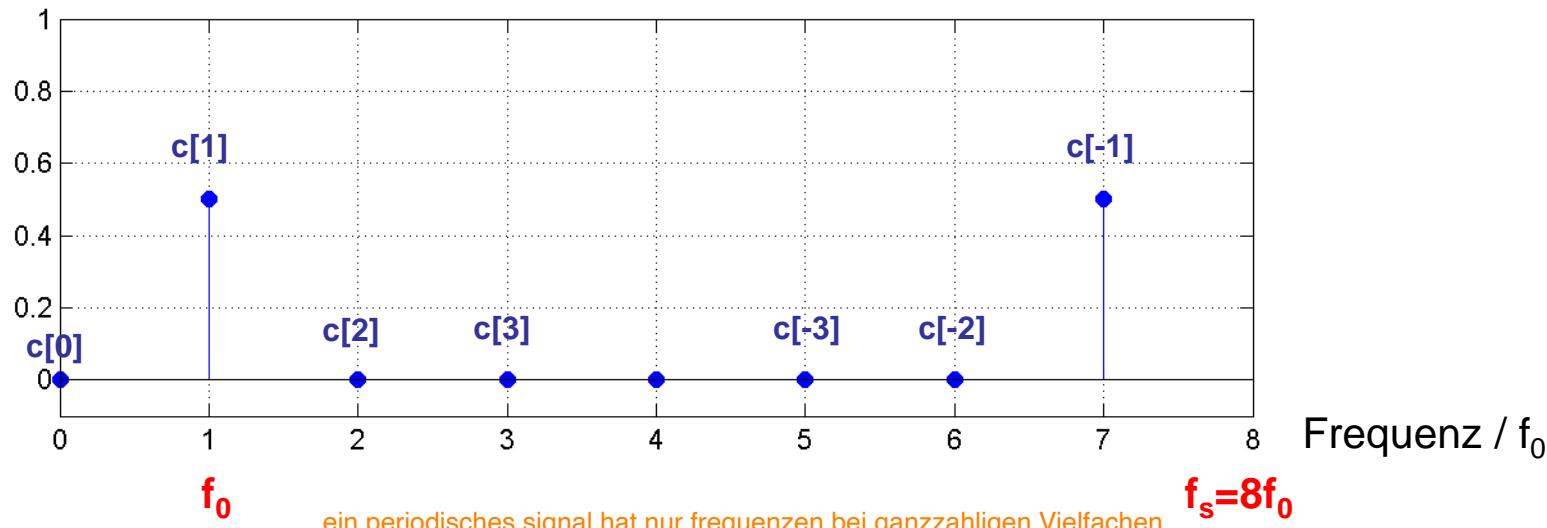
$$c_m = X[m]/N$$

# Approximation der FR mit der DFT

Zeitbereich:  $N = 8$  Abtastwerte  $x[n]$  von 1 Periode eines cos-Signals



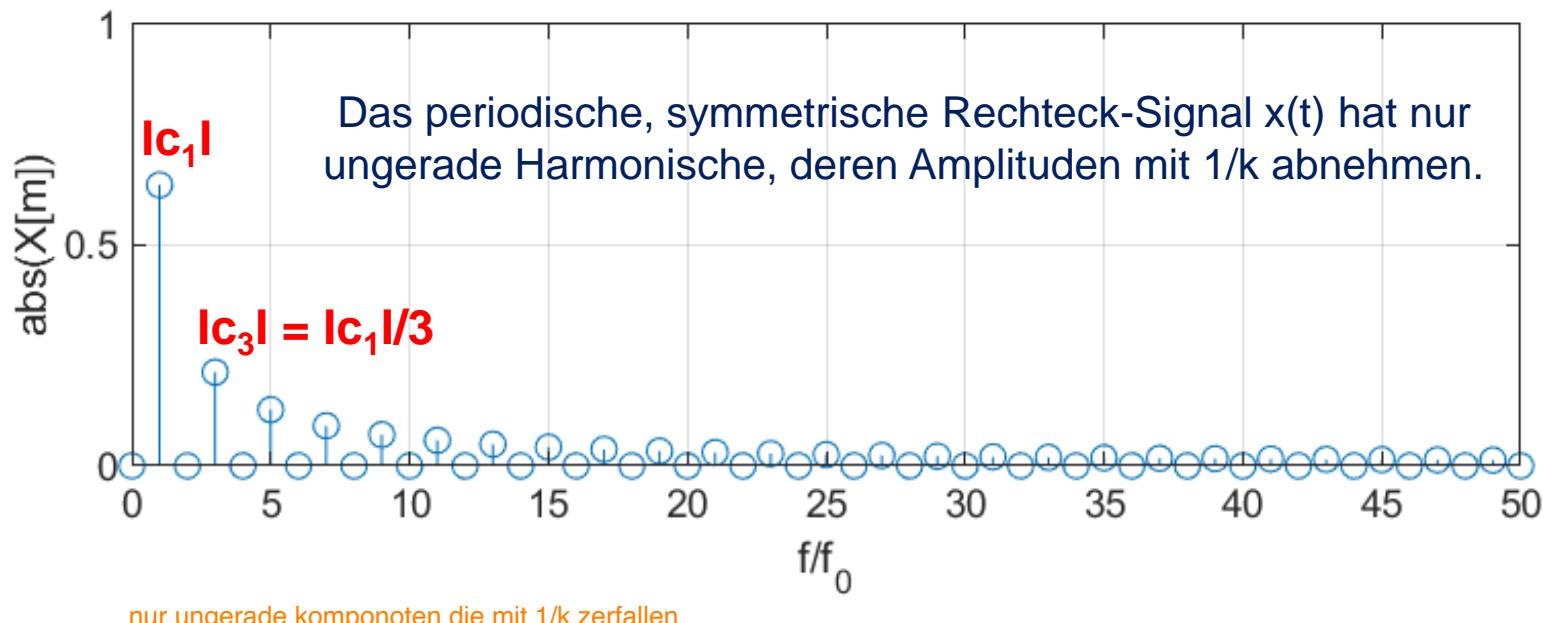
Frequenzbereich:  $N = 8$  Spektralwerte  $X[n]/N$  im Bereich  $[0, f_s]$



# Approximation der FR mit der DFT

Beispiel periodisches, symmetrisches Rechteck-Signal:

```
N=8192; % Anzahl Samples  
  
x=[ones(1,N/2) -ones(1,N/2)]; % 1 Periode Rechteck-Signal  
  
X=fft(x)/N; % FFT bzw. DFT  
  
subplot(2,1,1); stem([0:N-1],abs(X)); grid;  
  
xlabel('f/f_0'); ylabel('abs(X[m])'); axis([0 50 0 1])
```



Zeitbereich

diskret (abgetastet)

periodisch

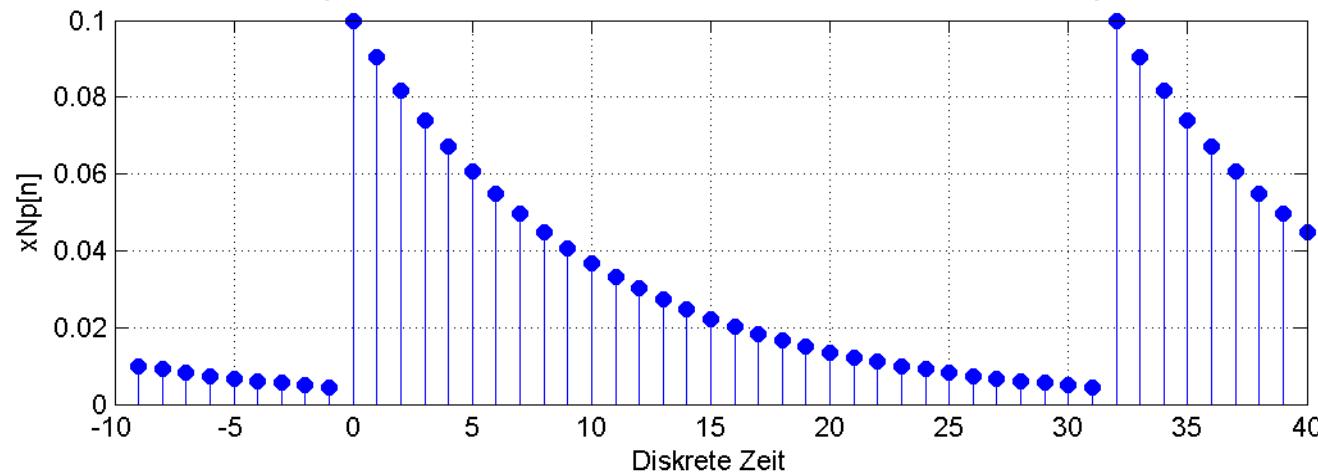
Frequenzbereich

periodisch

diskret



Zeitfenster-Sequenz

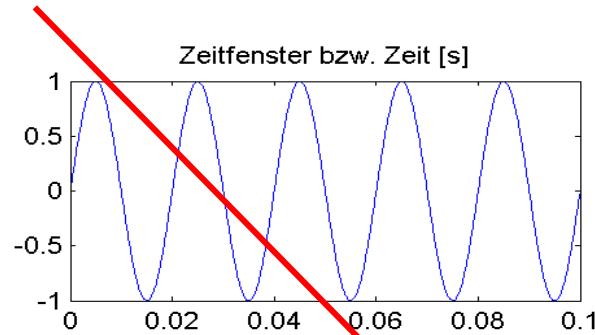


Die DFT berechnet das Spektrum des periodisch fortgesetzten, diskreten Zeitfenster-Signals.

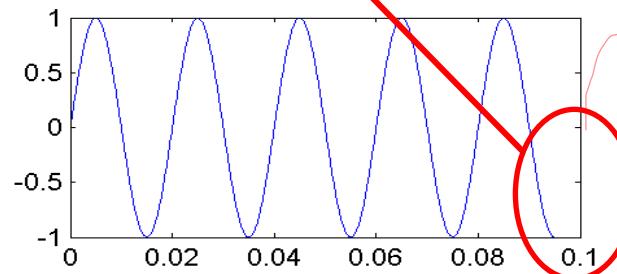
## Unpassende Fensterlänge

=> Sprungstellen bei periodischer Fortsetzung (die im realen Signal nicht vorhanden ist)

**5 x 50Hz-Perioden**

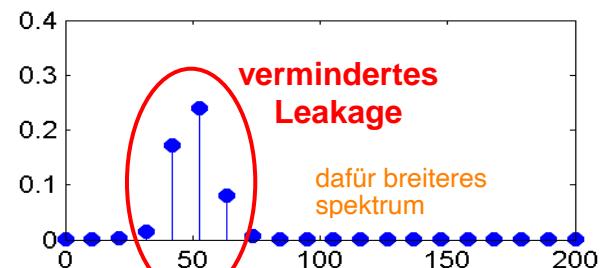
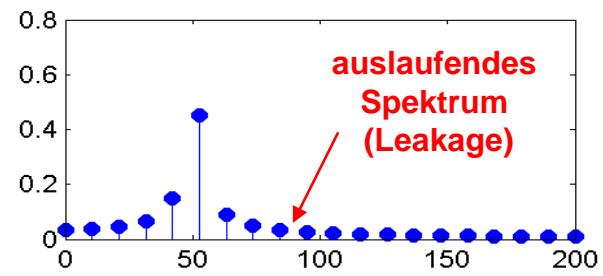
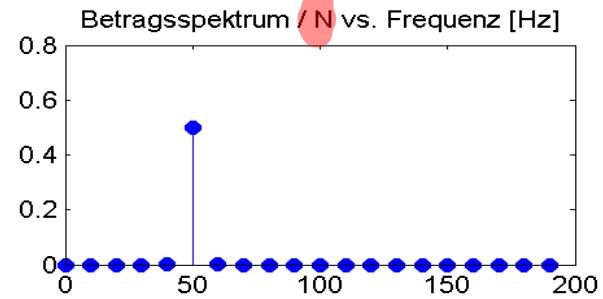
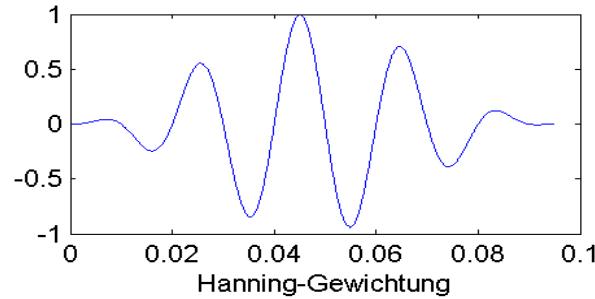


**4.75 x 50Hz-Perioden**



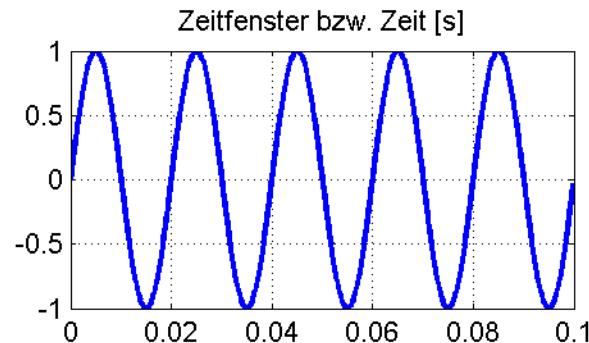
**auslaufendes  
Zeitfenster**

**4.75 x 50Hz-Perioden**

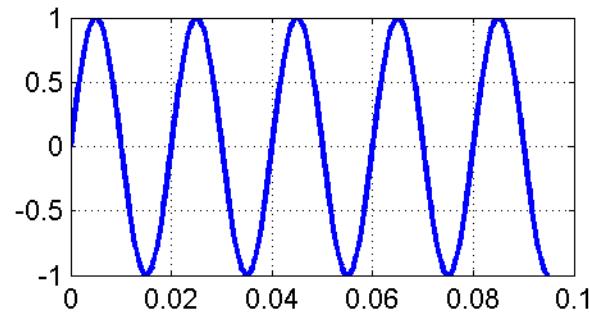


# Leakage

5 x 50Hz-Perioden

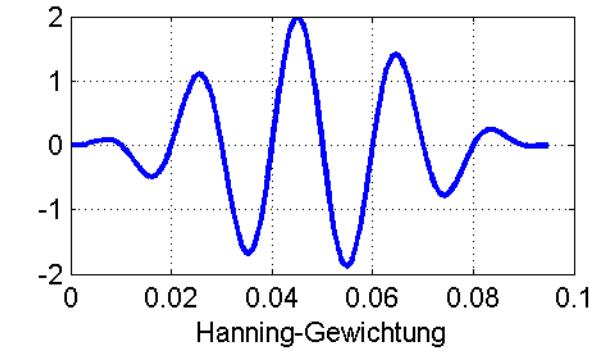


4.75 x 50Hz-Perioden

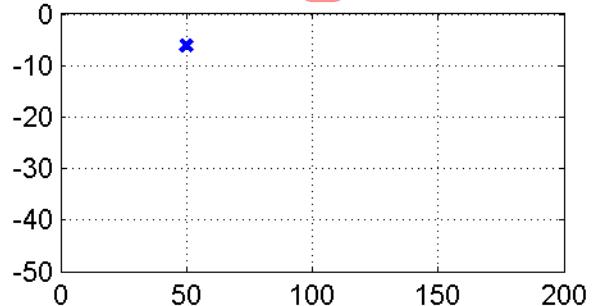


auslaufendes  
Zeitfenster

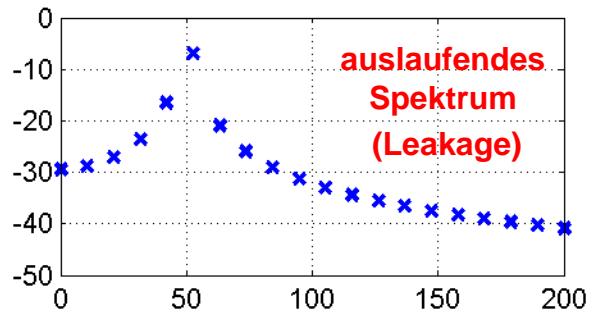
4.75 x 50Hz-Perioden



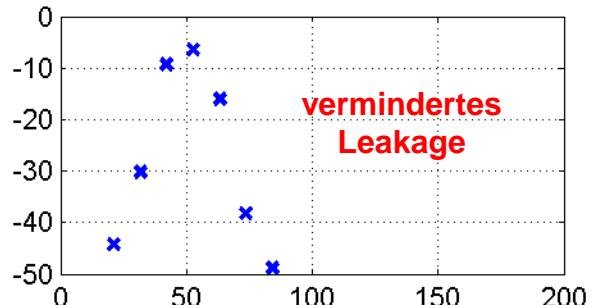
Betragsspektrum / N [dB] vs. Frequenz [Hz]



auslaufendes  
Spektrum  
(Leakage)



vermindertes  
Leakage

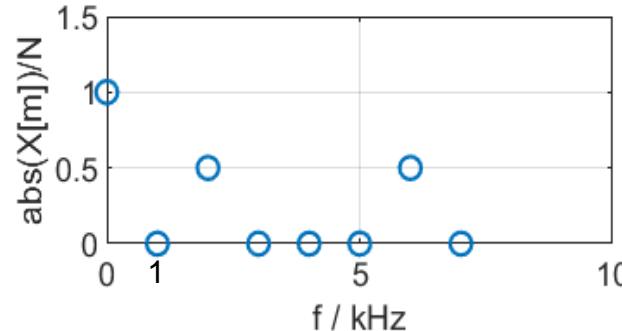
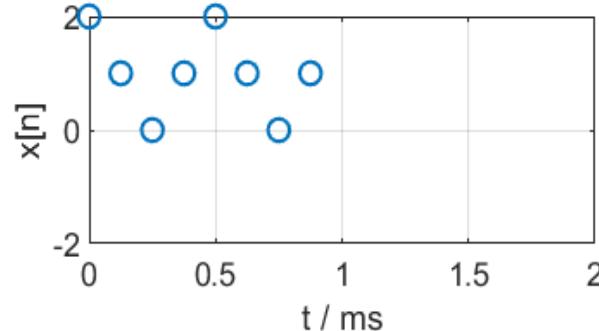


# Auswirkung der DFT-Fensterung

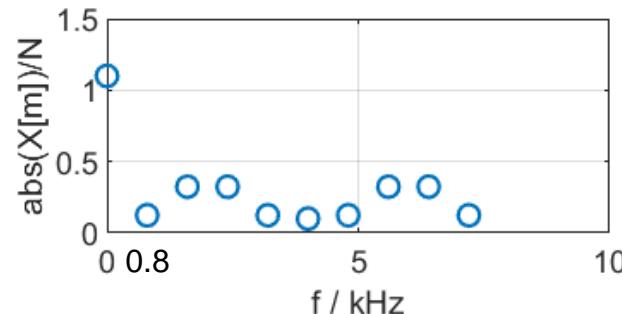
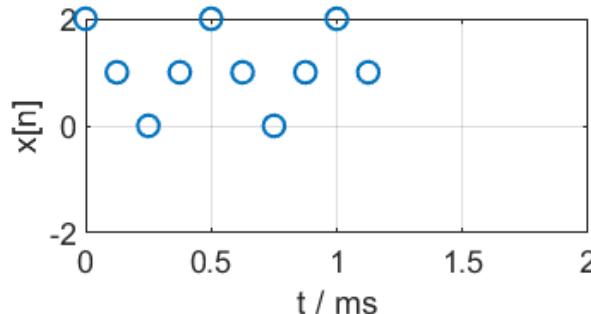
gegeben:  $x[n] = 1 + \cos(2\pi \cdot f_0 \cdot n T_s)$ , wobei  $f_0 = 2000$  Hz und  $n = 0, \dots, N-1$

gesucht: zugehöriges, normiertes DFT-Betragsspektrum  $|X[m]|/N$ , wenn

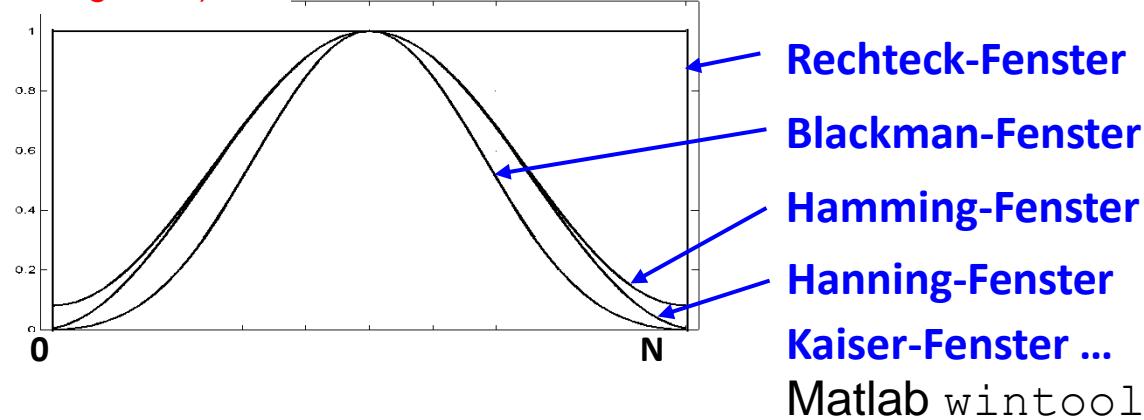
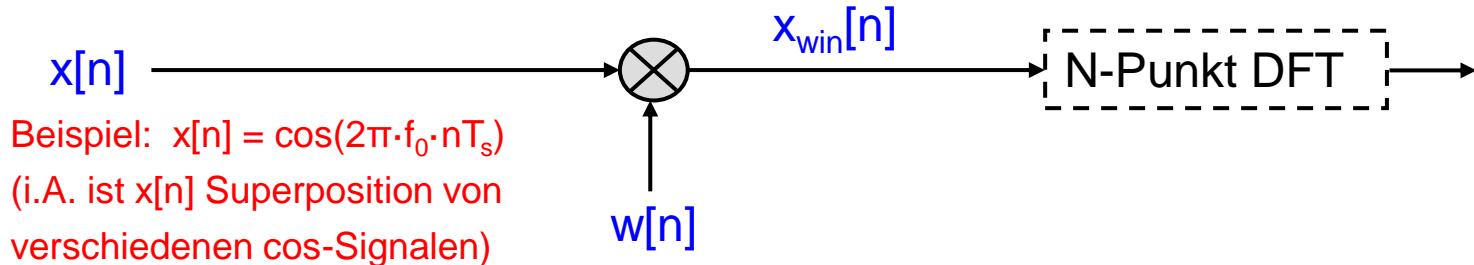
a) Blocklänge **N = 8** und Abtastfrequenz **f<sub>s</sub> = 8 kHz**:



b) Blocklänge **N = 10** und Abtastfrequenz **f<sub>s</sub> = 8 kHz**:



## Fensterung der Zeitfolge mit Fenster $w[n]$ (mit Spektrum $W(f)$ )



Beispiel

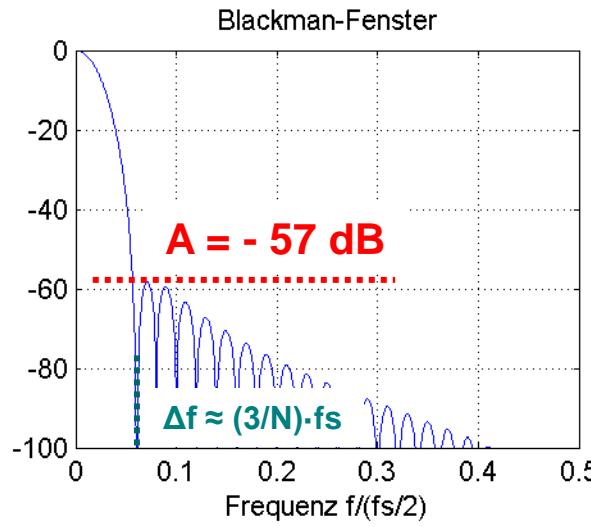
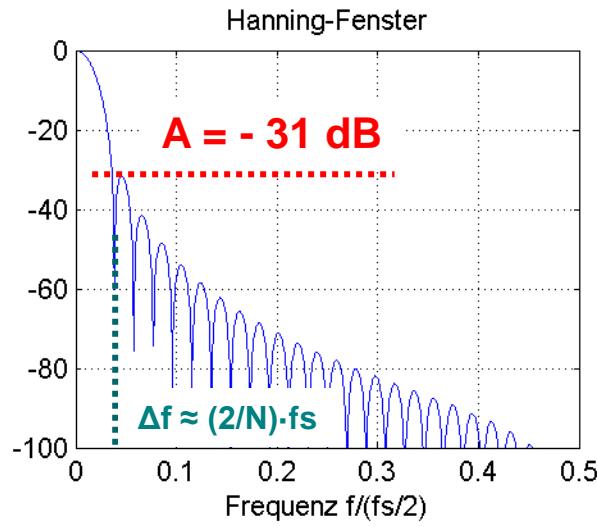
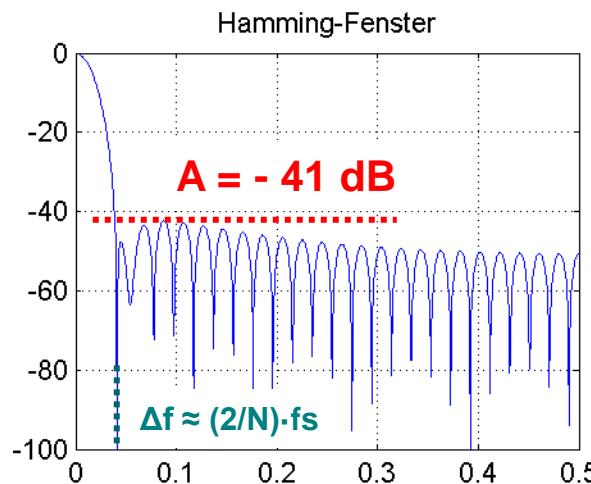
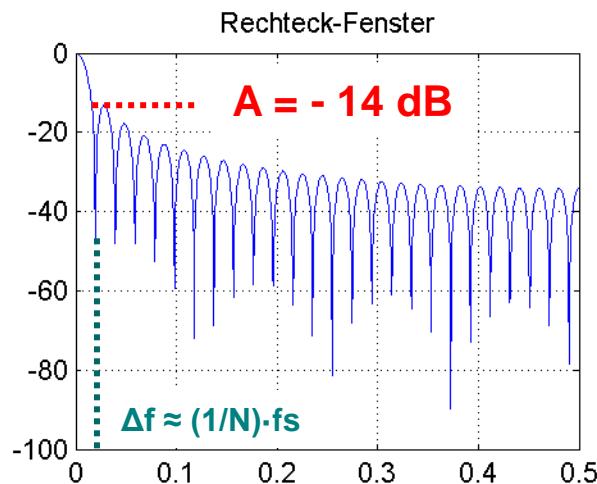
$$\text{Spektrum abgetastetes Signal: } X_{\text{win}}(f) = 0.5 \cdot W(f-f_0) + 0.5 \cdot W(f+f_0)$$

Multiplikation mit cos-Signal  $\rightarrow$  Frequenztranslation um  $f_0$

$$\text{DFT-Spektrum } X_{\text{win}}[m] = X_{\text{win}}(f=m \cdot f_s / N)$$

# DFT-Windowing

**schmalste Hauptkeule  
größte Nebenkeulen**

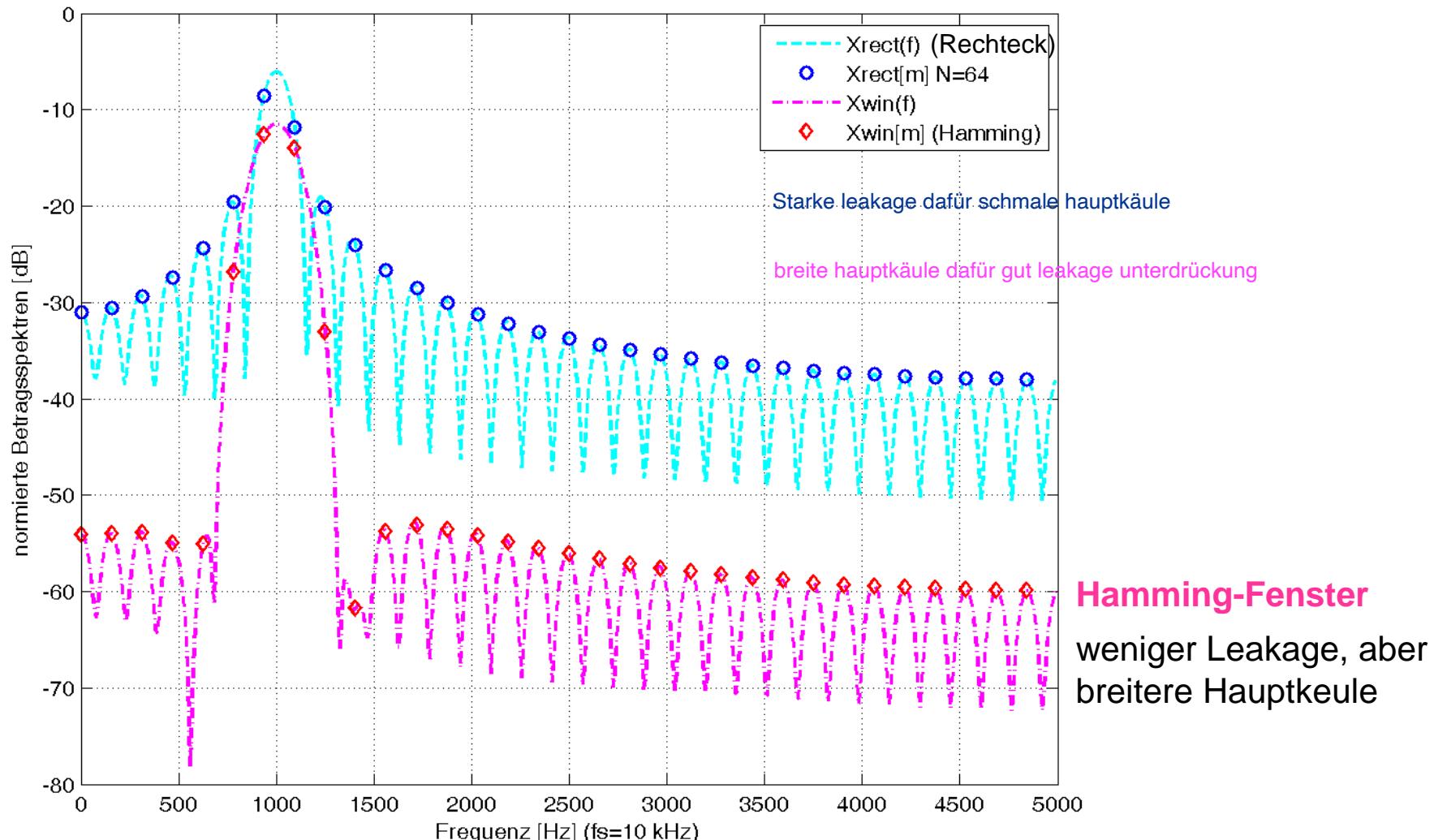


**breiteste Hauptkeule  
kleinste Nebenkeulen**

"leakage" stark reduziert,  
aber Verbreiterung der Spektrallinien

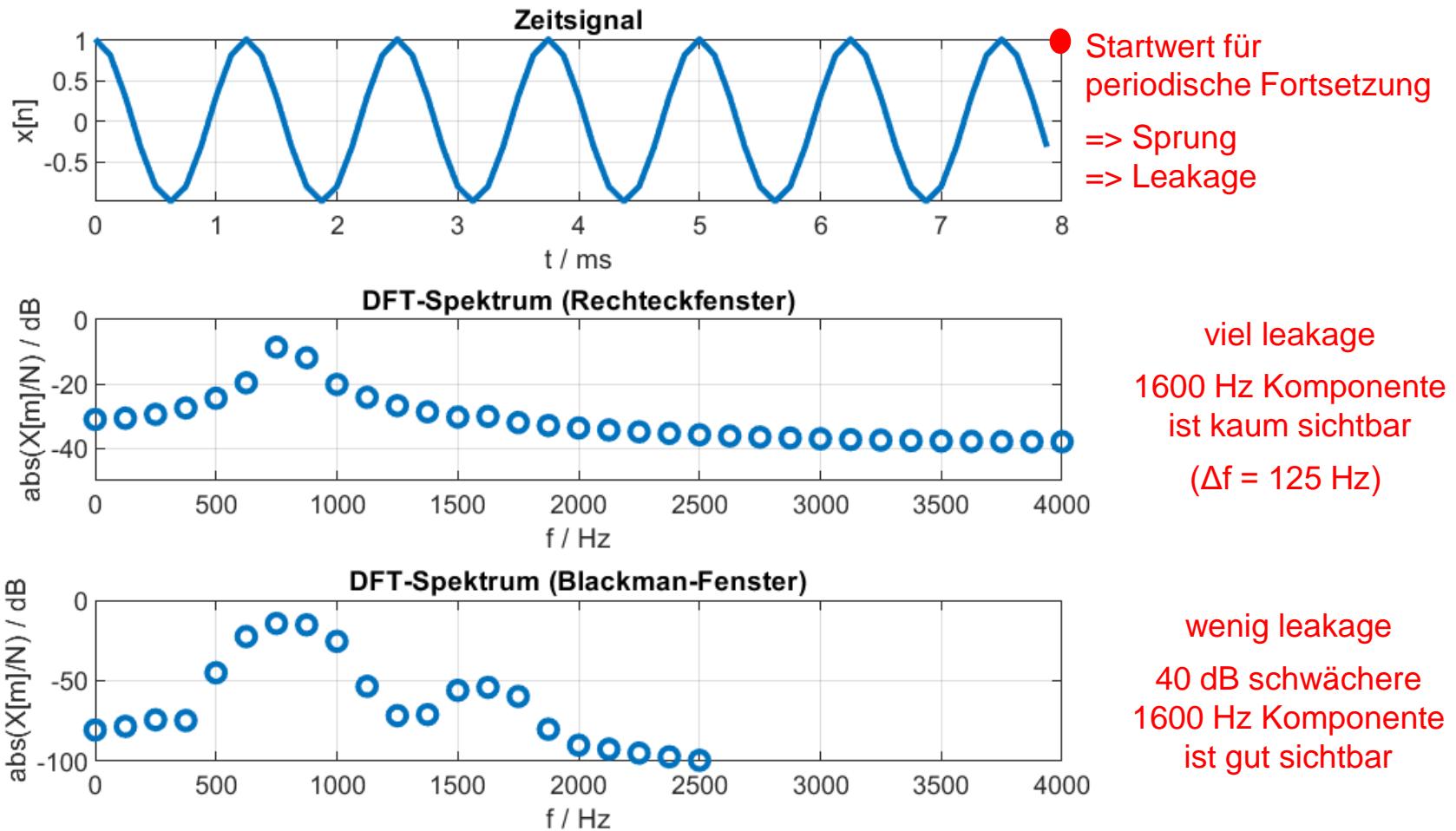
Fenster reduzieren "leakage" unterschiedlich stark, verbreitern aber Spektrallinien

N = 64 Punkt DFT-Spektrum eines 1000 Hz cos-Signals ( $f_s = 10 \text{ kHz}$ )



# DFT-Windowing

Signal  $x[n] = \cos(2\pi \cdot 800 \cdot nT_s) + 0.01 \cdot \cos(2\pi \cdot 1600 \cdot nT_s)$ ,  $f_s = 8$  kHz,  
und (un-) gefenstertes, normiertes 64-Punkt DFT-Betragsspektrum:



DFT und FT besitzen ähnliche Eigenschaften, z.B. Linearität

Verschiebungssatz:  $x[(n-k) \bmod N] \rightarrow X[m] \cdot e^{-j2\pi \frac{mk}{N}}$

zeitliche (zyklische) Verschiebung  $\rightarrow$  Phasendrehung im f-Bereich

**Beispiel:**  $x[n]=[1 2 3 4] \rightarrow X[m]$

$X_1[m]=X[m] \cdot e^{-j2\pi \frac{m \cdot 2}{4}}$   $\rightarrow$  entspricht Verschiebung (zyklisch) um 2 (nach rechts).

$X_1[m] \rightarrow x_1[n] = x[(n-2) \bmod 4] = [3 4 1 2]$

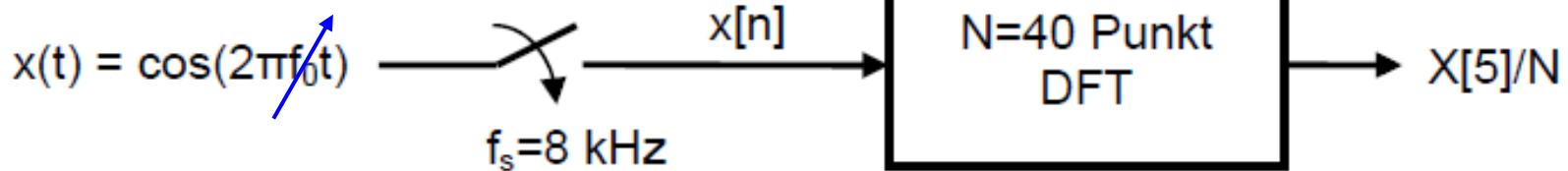
**Parseval (Bessels-Theorem)**  $\sum_{n=0}^{N-1} |x[n]|^2 = \frac{1}{N} \cdot \sum_{m=0}^{N-1} |X[m]|^2$

**Beispiel:**  $x[n]=[0 1 0 -1] \rightarrow X[m] = [0 -2j 0 2j] \rightarrow$  Energie  $E = 2$

DFT-Spektrum  $X[m]$  eines reellwertigen Zeitsignals  $x[n]$ :  $X[N-m] = X^*[m]$   
komplex konjugiert

Die Berechnung eines DFT-Werts  $X[m]$  entspricht einer **Filterung** mit einem **Bandpass** mit der **Bandbreite  $\Delta f$** .

## Beispiel

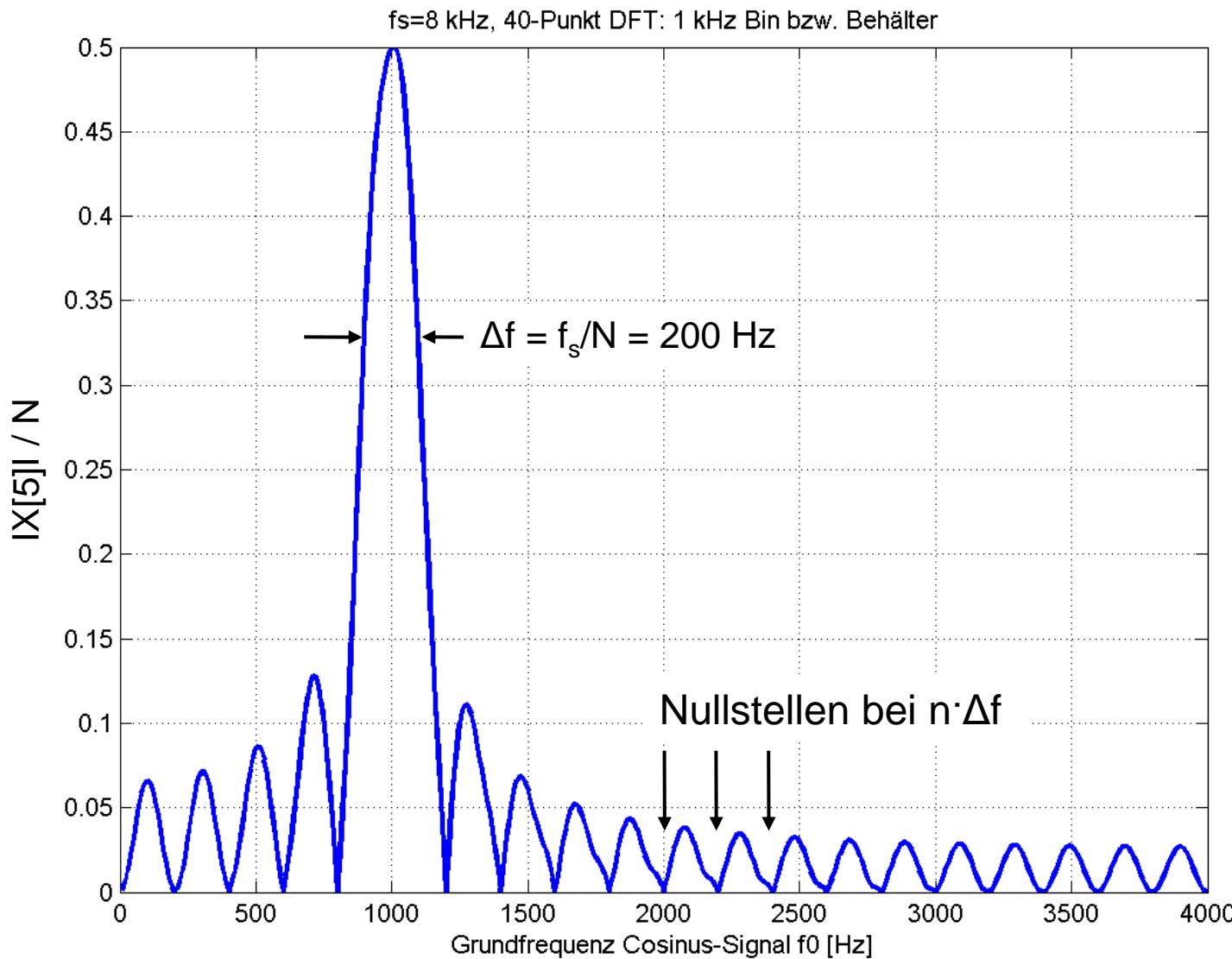


- $\Delta f = f_s/N = 8000/40 \text{ Hz} = 200 \text{ Hz}$
- $X[5]$  entspricht der Spektralkomponente bei  $5 \cdot \Delta f = 1 \text{ kHz}$

## Resultat (siehe nächste Folie)

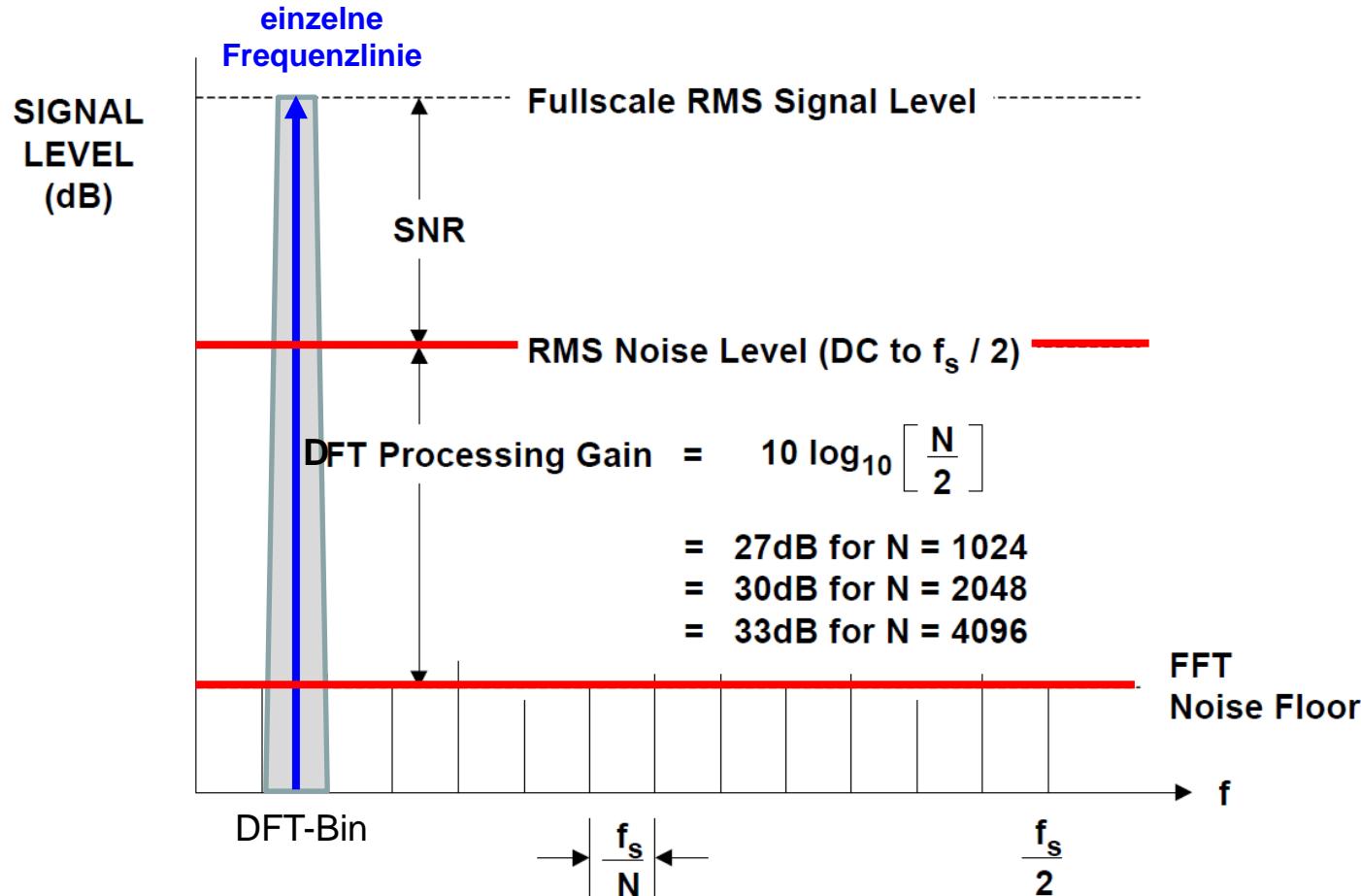
- $|X[5]| = 0.5$  wenn  $f_0 = 1 \text{ kHz}$  (wie erwartet)
- $|X[5]| = 0.5/\sqrt{2}$  wenn  $f_0 = 1.1 \text{ kHz}$
- $|X[5]| = 0.05$  wenn  $f_0 = 1.7 \text{ kHz}$  (leakage!)
- mit windowing:  
Filterfunktion hätte breitere Hauptkeule und kleinere Nebenkeulen

# Filterfunktion einer DFT-Komponente



# DFT Processing Gain

Quelle: Dr. S. Wyrsch, ZHAW



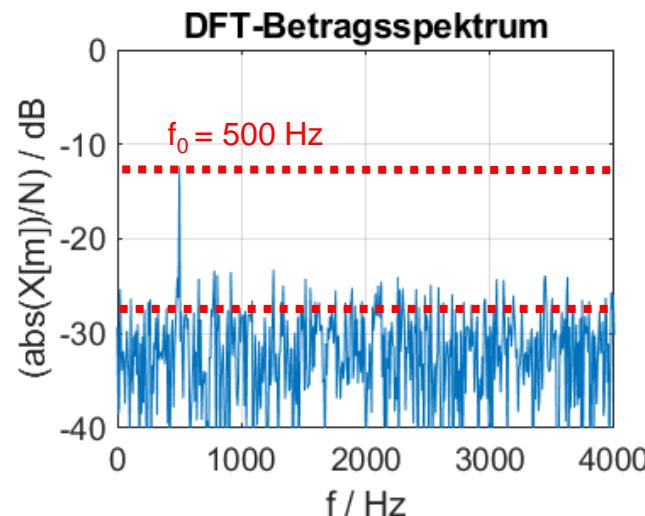
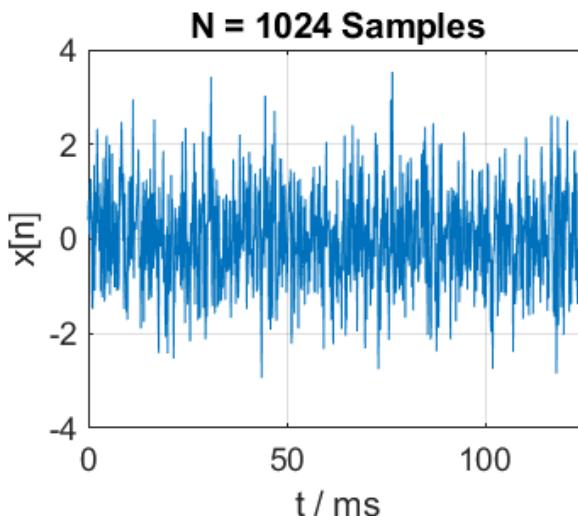
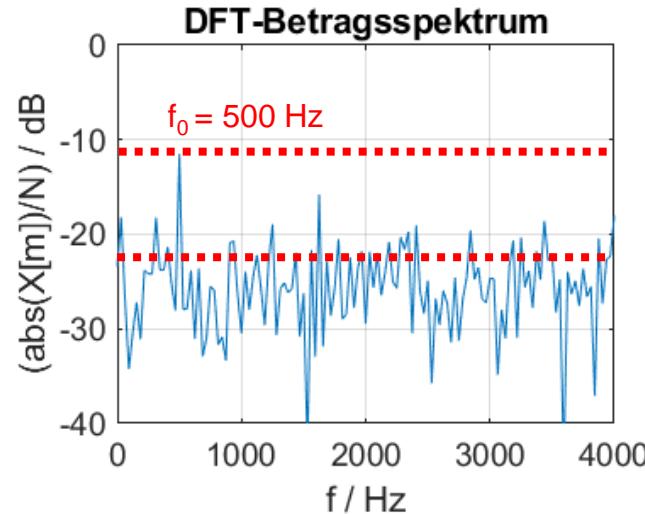
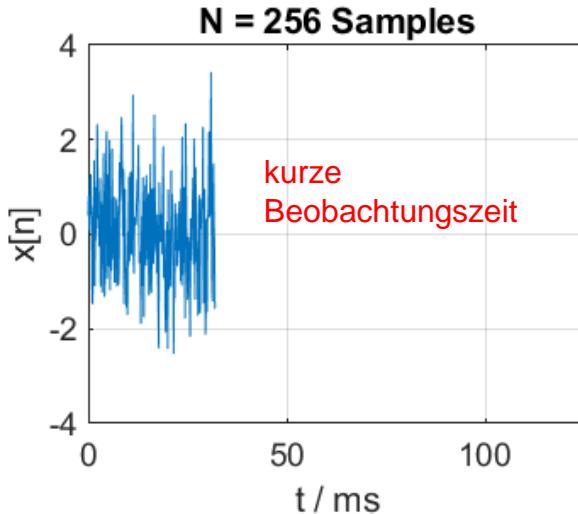
Bei einer Verdopplung der DFT-Länge N nimmt das SNR um 3 dB zu.

Die DFT wird oft eingesetzt, um schmalbandige Signalkomponenten in breitbandigem Rauschen zu detektieren.

- viele Rauschphänomene können als Additive White Gaussian Noise (AWGN) modelliert werden
- AWGN besitzt eine konstante Rauschleistungsdichte  $N_0 = P_{\text{noise}} / (f_s/2)$  im ganzen Nyquist-Frequenzband von  $[0, f_s/2]$
- DFT-Bin ist eigentlich ein Bandpass mit Bandbreite  $\Delta f = f_s/N$  ( $\Delta f$  ist invers proportional zur Blocklänge N bzw. Beobachtungszeit)
- einzelner (DFT-Bin-) Bandpass lässt das schmalbandige Signal passieren, filtert aber das breitbandige Rauschen und «reduziert» damit die breitbandige Rauschleistung  $P_{\text{noise}}$  um Faktor  $N/2$   
 $\Rightarrow$  Rauschleistung im Bin =  $\Delta f \cdot N_0 = \Delta f \cdot P_{\text{noise}} / (f_s/2) = (2/N) \cdot P_{\text{noise}}$
- DFT vergrössert also das SNR (Signal-to-Noise-Ratio) um  $10 \cdot \log_{10}(N/2)$  dB im Bin. Effekt wird oft als **DFT Processing Gain** bezeichnet.

# DFT Processing Gain: Beispiel

$x[n] = A \cdot \sin(2\pi \cdot f_0 \cdot nT_s) + \text{Rauschen}$ , SNR = -10 dB,  $f_s = 8 \text{ kSps}$



DFT-  
Processing-Gain

DFT-  
Processing-Gain

# Recheneffizienz DFT & FFT

Berechnung **1** DFT-Werts erfordert N komplexe Multiplikationen (=4N reelle Mults)

- Aufwand mit Goertzel-Algorithmus ist nur ca.  $\frac{1}{4}$  des DFT-Aufwands

DFT

$$X[m] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j\frac{2\pi}{N}mn} \quad m = 0, 1, \dots, N-1$$

Ein einziger Wert  $X[i]$  braucht N Multiplikationen

Berechnung aller **N** DFT-Werte

- erfordert  $N^2$  komplexe (oder  $4N^2$  reelle) Multiplikationen
- mit FFT-Algorithmus (Cooley, Tukey, 1965) sind nur  **$N \cdot \log_2(N)$**  komplexe Multiplikationen erforderlich, der Aufwand ist also praktisch proportional mit N (statt quadratisch)!
- Zahlenbeispiel mit  $N = 1024$ 
  - ohne FFT: ca. 1 Mio. komplexe Multiplikationen
  - mit FFT: nur ca. 10'000 komplexe Multiplikationen

# Herleitung FFT-Algorithmus (N=2<sup>x</sup>)

DFT

$$X[m] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j\frac{2\pi}{N}mn} \quad m = 0, 1, \dots, N-1$$

N-Punkt DFT kann in zwei N/2-Punkt DFTs über die geraden und die ungeraden Samples aufgeteilt werden (usw.)

$$X[m] = \sum_{n=0}^{N/2-1} x[2n] \cdot W_N^{m2n} + \sum_{n=0}^{N/2-1} x[2n+1] \cdot W_N^{m(2n+1)} \quad m = 0, 1, \dots, N-1$$

Der **twiddle factor**  $W_N = e^{-j2\pi/N}$  weist die Symmetrie  $W_N^{2mn} = W_{N/2}^{mn}$  auf!

$$X[m] = \sum_{n=0}^{N/2-1} x[2n] \cdot W_{N/2}^{mn} + W_N^m \cdot \sum_{n=0}^{N/2-1} x[2n+1] \cdot W_{N/2}^{mn} \quad m = 0, 1, \dots, N-1$$

# Beispiel mit N = 4

$$X[m] = \sum_{n=0}^1 x[2n] \cdot e^{-j \frac{2\pi}{2} mn} + e^{-j \frac{2\pi}{4} m} \cdot \sum_{n=0}^1 x[2n+1] \cdot e^{-j \frac{2\pi}{2} mn}$$

$$X[m] = \sum_{n=0}^1 x[2n] \cdot (-1)^{mn} + (-j)^m \cdot \sum_{n=0}^1 x[2n+1] \cdot (-1)^{mn}$$

$$m = 0: X[0] = (x[0] + x[2]) + 1 \cdot (x[1] + x[3]) = X[0]$$

$$m = 1: X[1] = (x[0] - x[2]) + (-j) \cdot (x[1] - x[3])$$

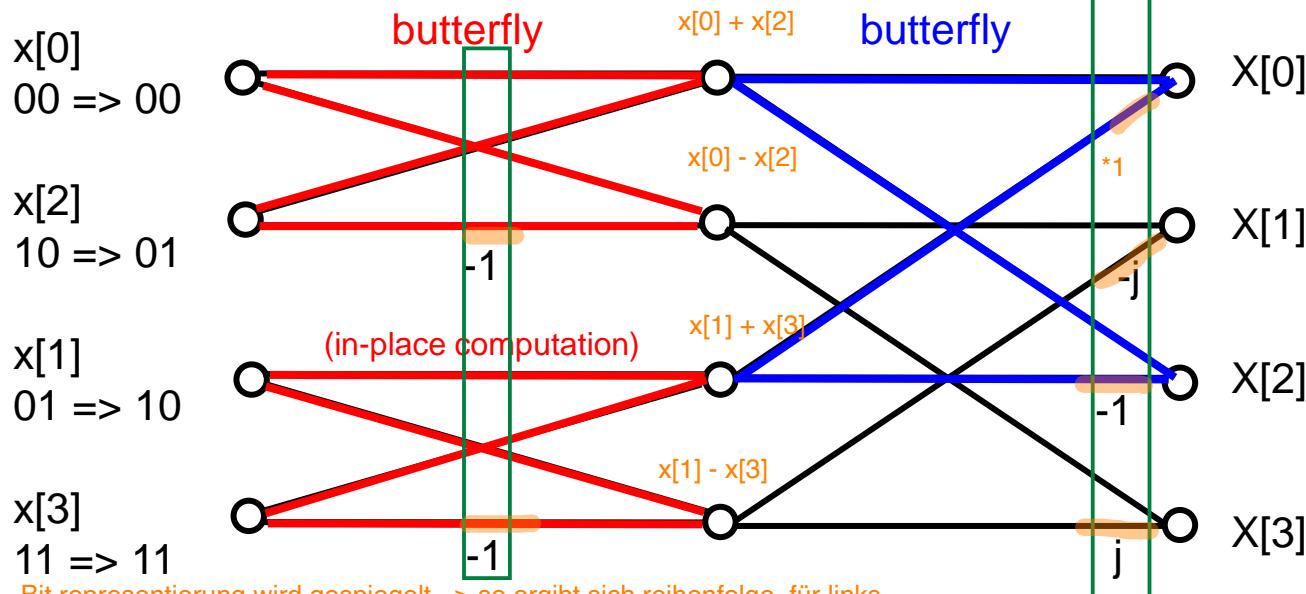
$$m = 2: X[2] = (x[0] + x[2]) + (-1) \cdot (x[1] + x[3])$$

$$m = 3: X[3] = (x[0] - x[2]) + j \cdot (x[1] - x[3])$$

roots of unity  
=> immer \*2 schritte mehr im  
gegenuhzeigersinn  
Hier z.B. 4 Schritte  
> 1, -j, -1, j

links 2 schritte -> 1, -1, 1, -1

Bit-reversed-Adressierung



Bit representierung wird gespiegelt. -> so ergibt sich reihenfolge für links

Farben unten sind  
nicht die gleiche  
Bedeutung wie oben

bei punkten wird  
addiert

diese Seite ist  
richtig sortiert

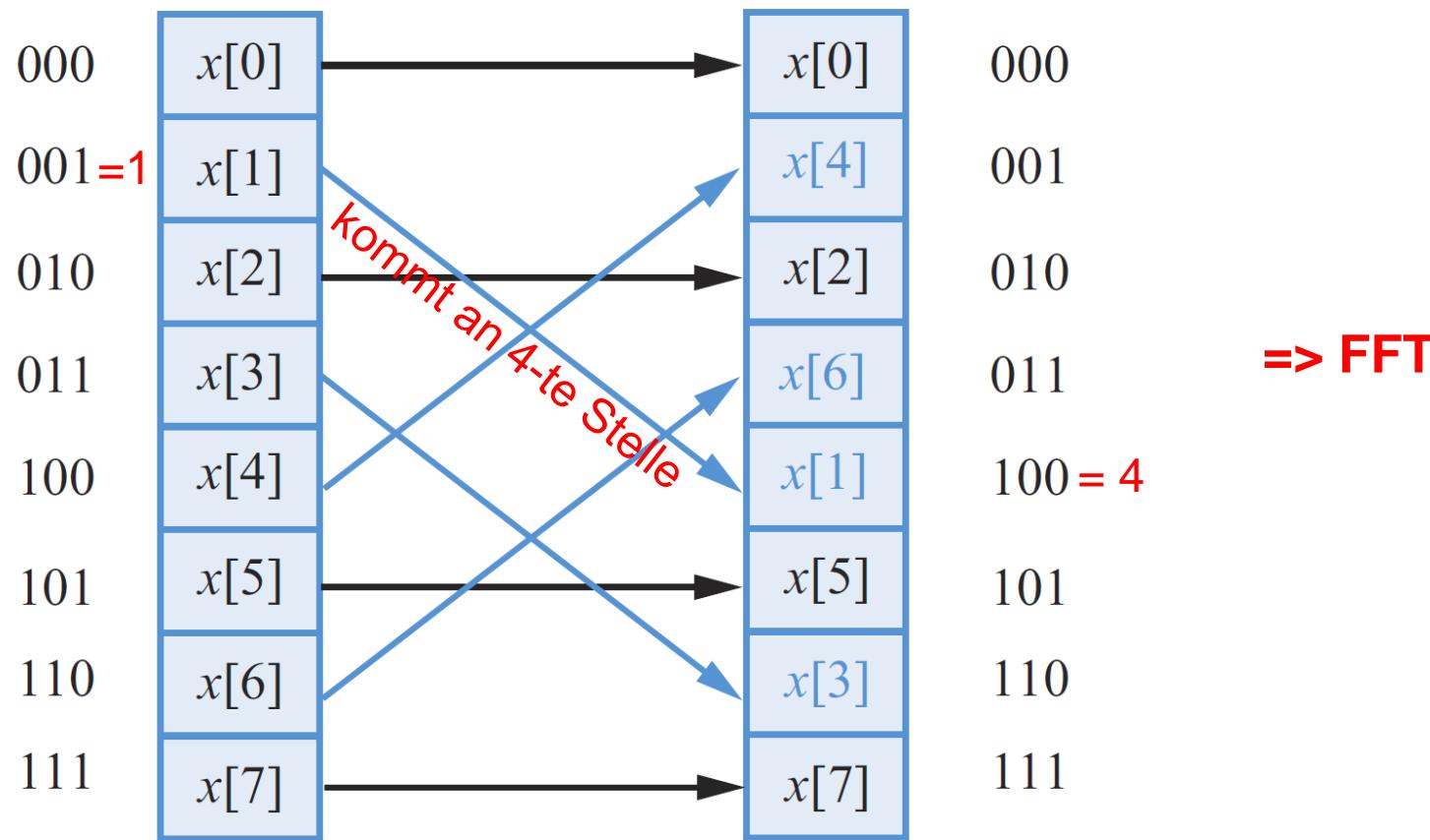
- N-Punkt DFT lässt sich in  $\log_2(N)$  Stufen zerlegen (wenn Blocklänge N eine 2er-Potenz ist, radix 2)
- pro Stufe fallen **N** komplexe Multiplikationen und Additionen an
- Gesamtaufwand reduziert sich von  **$N^2$**  auf  **$N \cdot \log_2(N)$**  komplexe Multiplikationen und Additionen.
- es ist nur 1 Vektor mit N komplexen Speicherwerten erforderlich (**in-place computation**)!
- Wenn die **Zeitfolge  $x[n]$**  in immer kürzere Teilfolgen aufgeteilt wird, spricht man von **decimation-in-time FFT**.
- Wenn die **Spektralfolge  $X[m]$**  in immer kürzere Teilfolgen aufgeteilt wird, spricht man von **decimation-in-frequency FFT**.
- Alternative Formen: z.B. **Radix-4-Algorithmus**  
(Blocklänge N ist eine Potenz von 4, butterflies haben 4 Datenpfade, 25% weniger Multiplikationen erforderlich, Prog-Code aufwendiger)

# Bit-reversed-Adressierung

für Decimation-in-time FFT (radix 2): Umsortierung der Eingangswerte

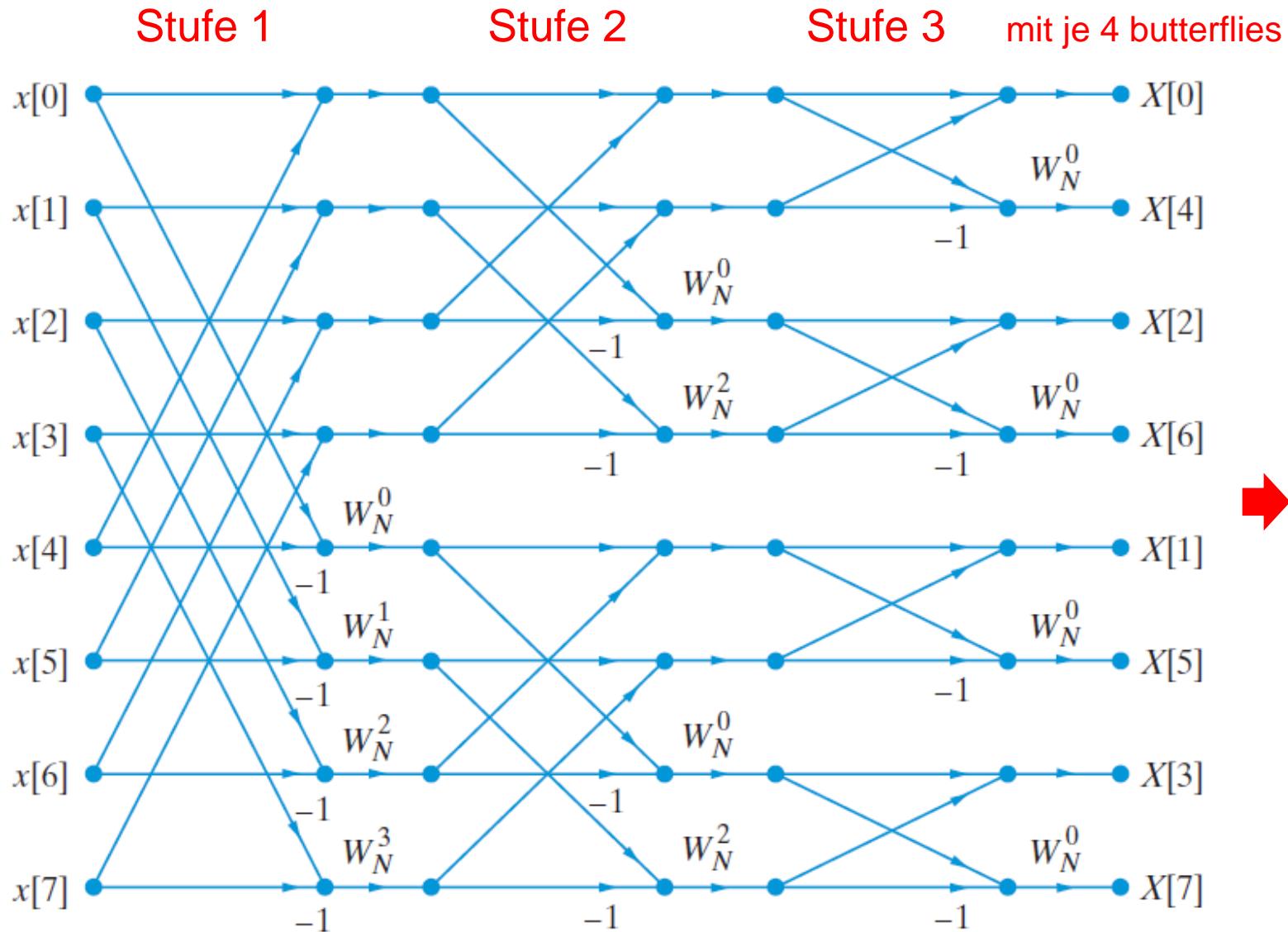
Zeitindex n binär darstellen, rückwärts lesen => Speicherplatz von x[n]

Beispiel wenn N=8:



# 8-Punkt FFT (decimation-in-frequency, radix 2)

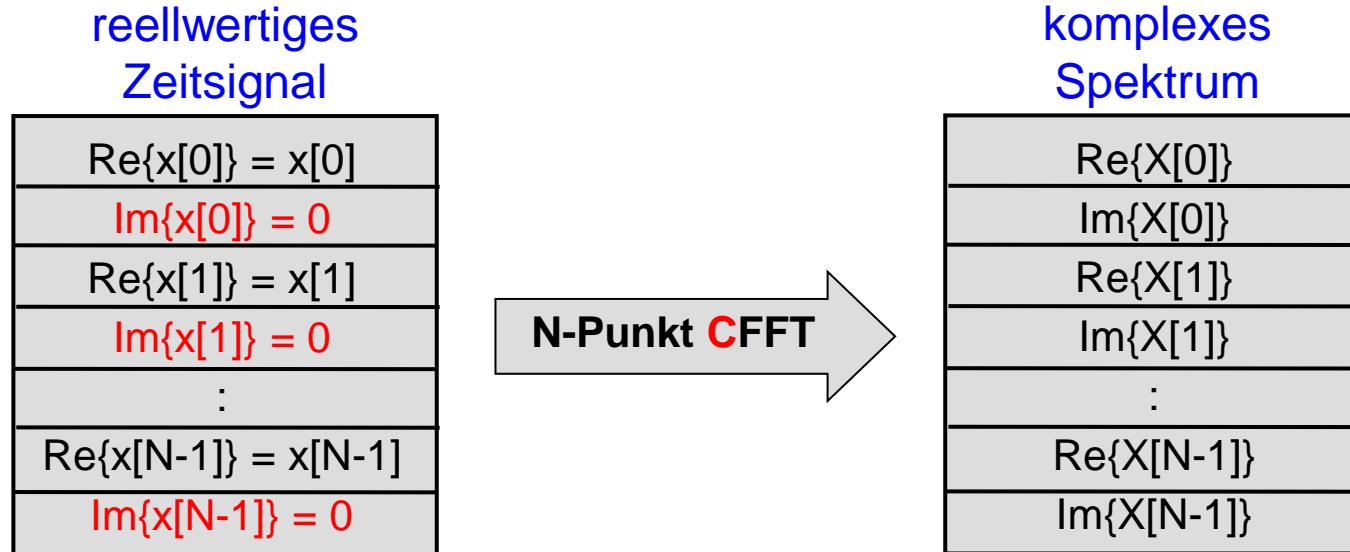
3 Stufen =  $\log_2(8\text{-punkt})$



Umsortierung (Bit-reversed-Adressierung)

Eine N-Punkt **complex FFT** transformiert N **komplexe** Zeitwerte in N **komplexe** Spektralwerte.

Reell-wertiges Zeitsignal: Imaginärteile im Speicher **mit Null initialisieren!**



**Es gibt 2 Ansätze, effizienter zu arbeiten.**

# Ansatz 1: Eine N-Punkt FFT von zwei reellwertigen Zeitsignalen mit je N Abtastwerten

**Input:** **zwei** reelle Zeitsignale  $x_1[0], \dots, x_1[N-1]$  &  $x_2[0], \dots, x_2[N-1]$

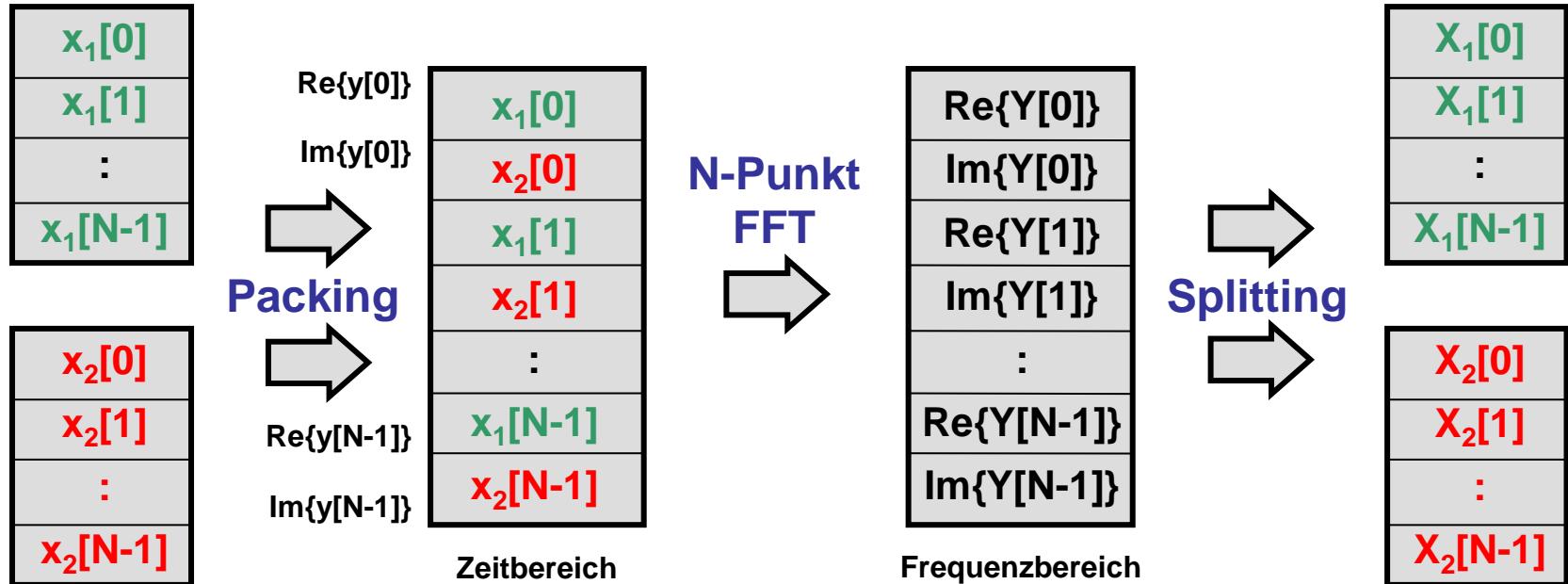
**Schritt 1:** Bildung komplexer Input-Vektor:  $y[n] = x_1[n] + j \cdot x_2[n]$

**Schritt 2:** **eine** N-Punkt FFT  $\Rightarrow Y[m]$

**Schritt 3:** Trennung der beiden Spektren

**Output:**  $X_1[m] = (Y[m] + Y^*[N-m])/2, m=0, \dots, N-1$

$X_2[m] = (Y[m] - Y^*[N-m])/2j, m=0, \dots, N-1$



# Ansatz 2: Eine N/2-Punkt FFT eines reellwertigen Zeitsignals mit N Abtastwerten

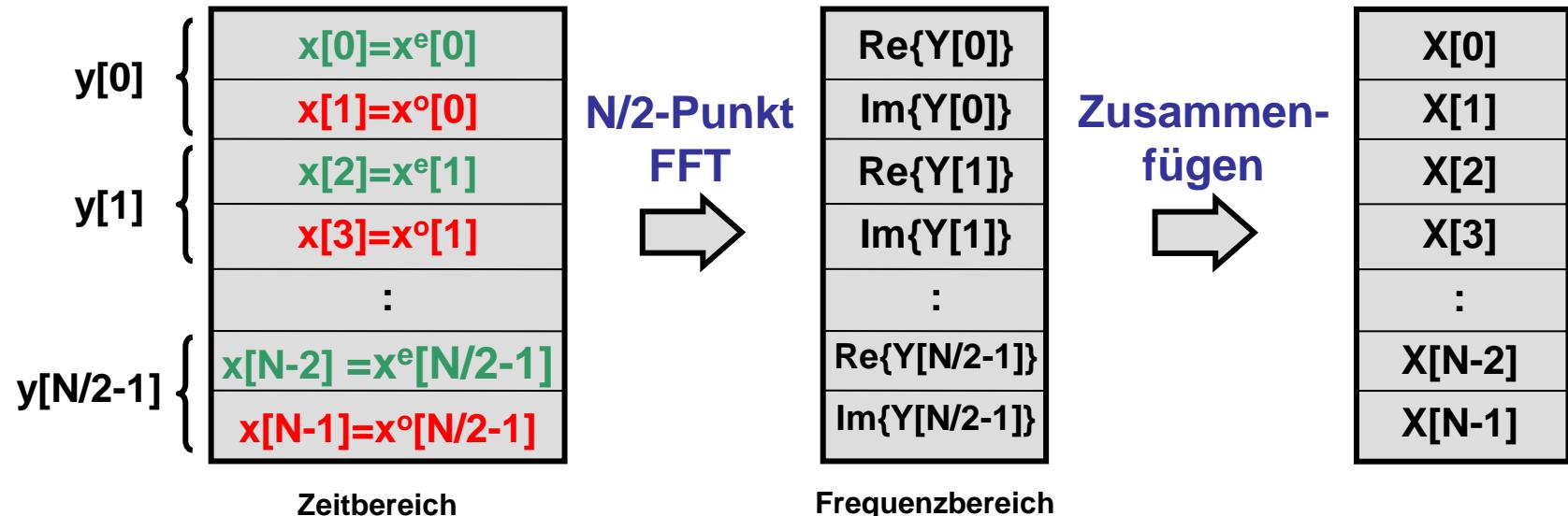
**Input:** ein reelles Zeitsignal  $x_1[0], \dots, x_1[N-1]$  der Länge **N**

**Schritt 1:** Bildung komplexer Input-Vektor:  $y[n] = x[2n] + j \cdot x[2n+1]$

**Schritt 2:** **N/2**-Punkt FFT  $\Rightarrow Y[m] = X^e[m] + j \cdot X^o[m]$

**Schritt 3:**  $X[m] = X^e[m] + e^{j2\pi m/N} \cdot X^o[m], m=0, \dots, N-1$

$$X[m] = 0.5 \cdot (Y[m] + Y^*[N/2-m]) - 0.5j \cdot (Y[m] - Y^*[N/2-m]) \cdot e^{j2\pi m/N}$$



Real FFT ist eine optimierte CFF für reellwertige input signale  $\rightarrow$  es kann mit N/2 punkt gearbeitet werden.

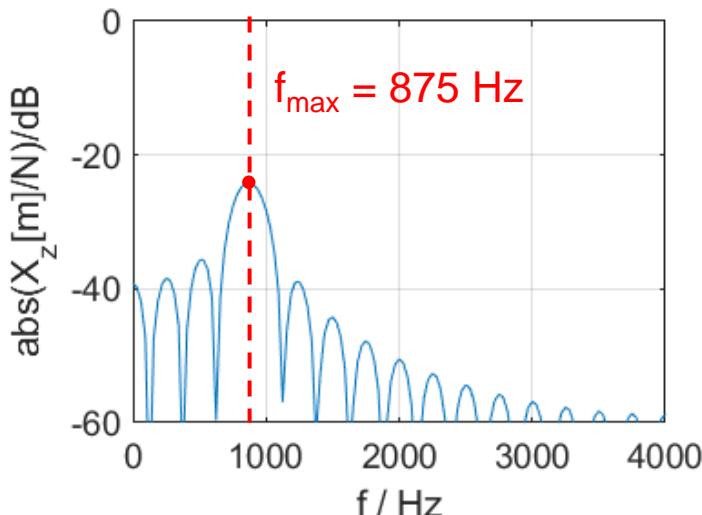
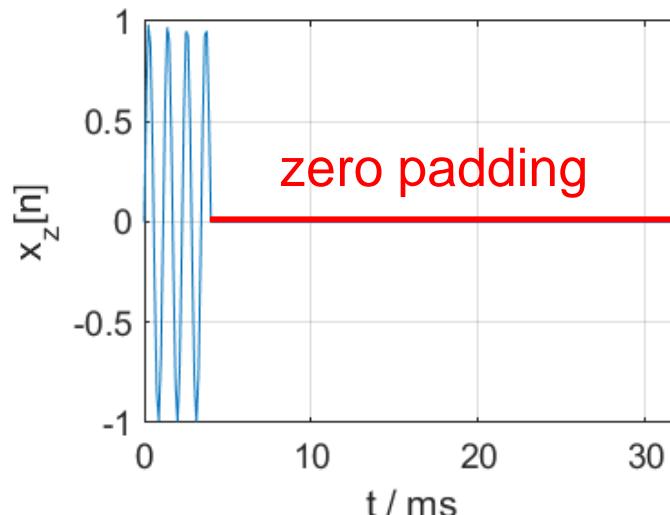
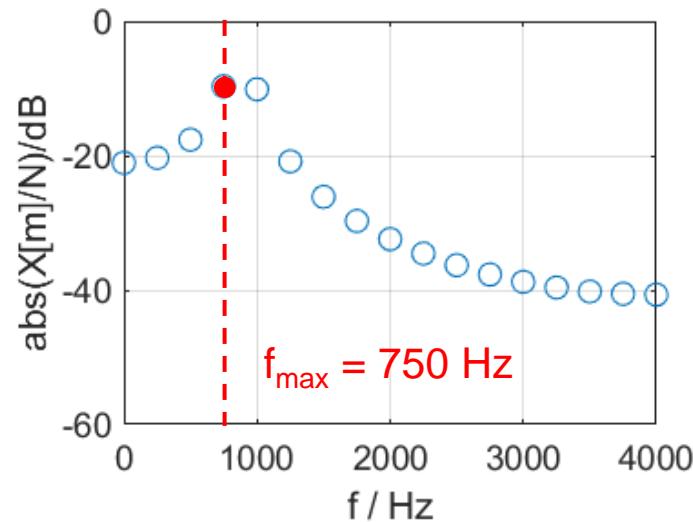
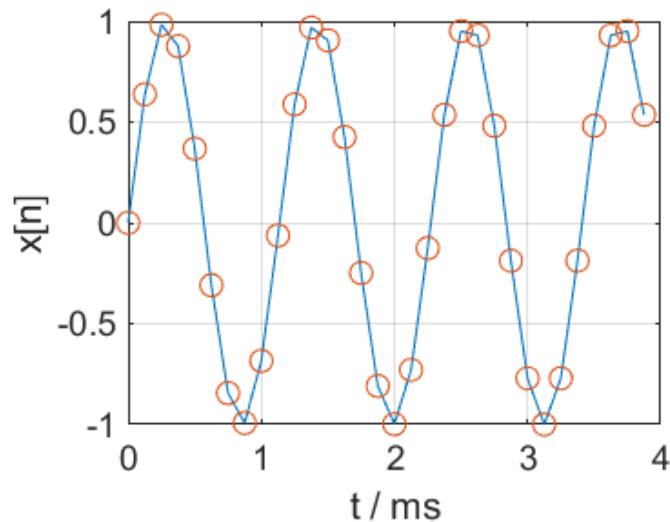
**Nullen anhängen**  $x_{\text{zero\_padded}} = [x \text{ zeros}(1, N - \text{length}(x))]$  wird verwendet

- um ein Zeitsignal auf die **Länge** einer **2-er Potenz** zu **ergänzen**, für eine (radix 2) FFT.
- um die **spektrale Auflösung** von "kurzen" Mess-Signalen zu **erhöhen** (zero-Padding verändert das Spektrum nicht).

## Beispiel

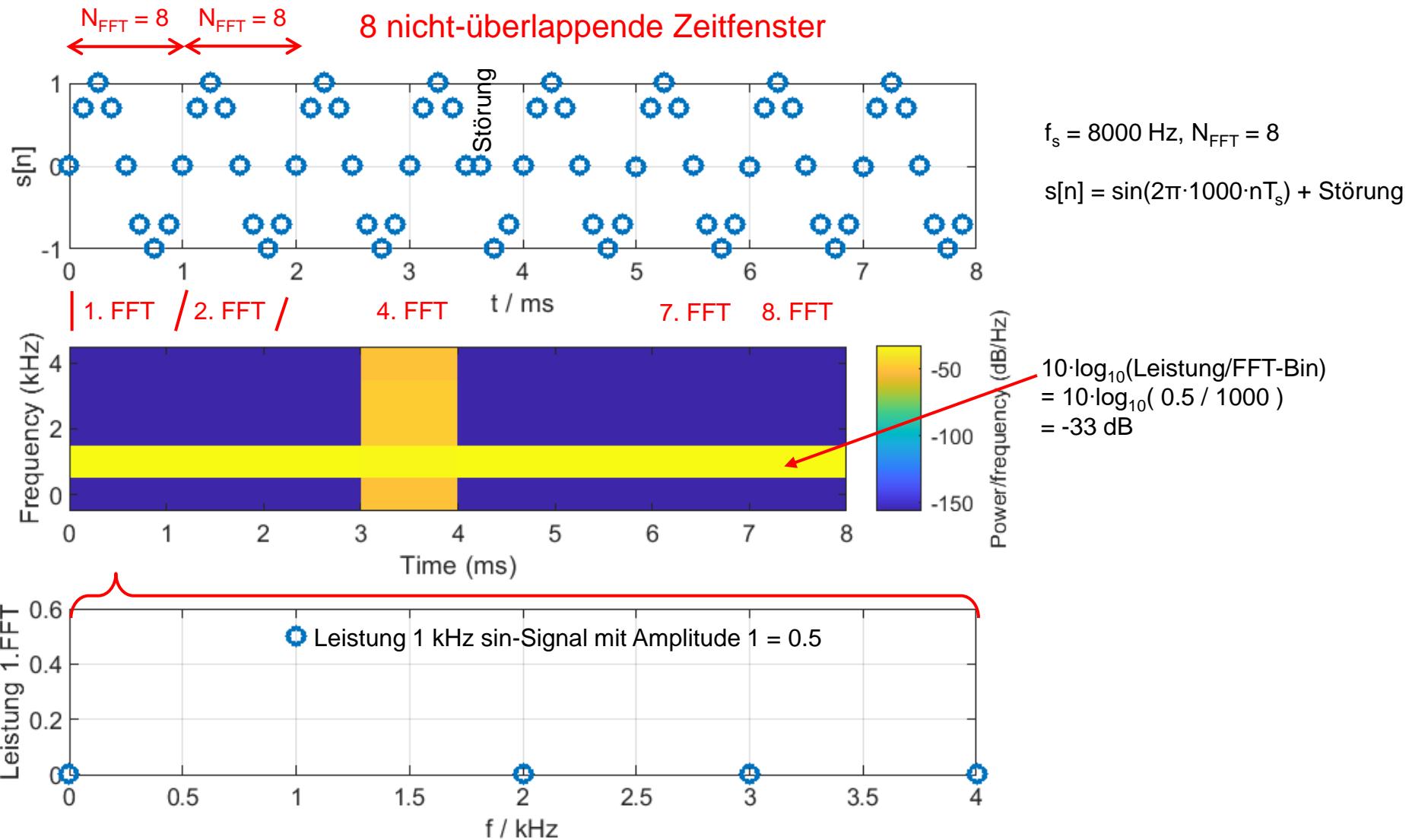
- kurzes Signal  $x[n] = \sin(2\pi \cdot f_0 \cdot n T_s)$ ,  $f_0 = 880$  Hz,  $n = 0 \dots 31$ ,  $f_s = 8$  kSps
- 32-Punkt FFT hat eine Frequenzauflösung von  $\Delta f = 250$  Hz  
=> Betragsspektrum maximal @  $f_{\max} = 750$  Hz statt 880 Hz
- zero-Padding  $x_z = [x \text{ zeros}(1, 512 - 32)]$  auf Länge  $N = 512$
- 512-Punkt FFT hat eine Frequenzauflösung von  $\Delta f = 15.625$  Hz  
=> Betragsspektrum maximal @  $f_{\max} = 875$  Hz statt 880 Hz

# Zero-Padding



- Ein Spektrogramm zeigt den **zeitlichen Verlauf des Spektrums** eines Signals **als Bild** (Achsen: Zeit und Frequenz, Pixel: Intensität meist farbcodiert).
- Zeitvariante Darstellung der Frequenzverteilung durch Aneinander-Reihen von (überlappenden) **Kurzzeit-FFT-Betragsspektren** (im Quadrat, eigentlich Leistung).  
mehrere FFT über kurze Zeitspannen  
-> somit kann zum Beispiel gesehen werden ob eine Frequenz absterbt
- Alternative Bezeichnung: **Short Time Fourier Transform (STFT)**
- Spektrogramme werden verwendet um zu analysieren, wie sich der Frequenzgehalt von **nicht-stationären** Signalen im Laufe der Zeit verändert.
- Eine typische Anwendung ist die **Audiosignalverarbeitung** z.B. als Ausgangspunkt (Feature Extraction) für
  - Soundklassifikation mittels Convolutional Neural Network (CNN)
  - Musikerkennung mit Shazam, ...

# Spektrogramm: Matlab-Beispiel



% Parameter

```
fs = 8000; f0 = 1000; N = 64; NFFT = 8;
```

% Signal

```
t=[0:N-1]/fs; s = sin(2*pi*f0*t); s(30) = 0;  
subplot(3,1,1); plot(1000*t,s,'LineWidth',2.0);  
grid; xlabel('t / ms'); ylabel('s[n]');
```

% Spectrogram, Alternative siehe STFT(.)

```
window = rectwin(NFFT); % Rechteck-Fenster, kein Leakage  
no_overlapping_samples = 0; % Zeitfenster nicht-überlappend  
subplot(3,1,2);  
spectrogram(s,window,no_overlapping_samples,NFFT,fs,'yaxis')
```

% Leistung

```
S2 = spectrogram(s,window,no_overlapping_samples,NFFT) /NFFT; % 5 x 8 Matrix  
f = [0:NFFT/2]*fs/NFFT; % nur positive Frequenzen  
subplot(3,1,3); plot(f/1000,abs(S2(:,1)),'o','LineWidth',2.0);  
grid; xlabel('f / kHz'); ylabel('Leistung 1.FFT');
```

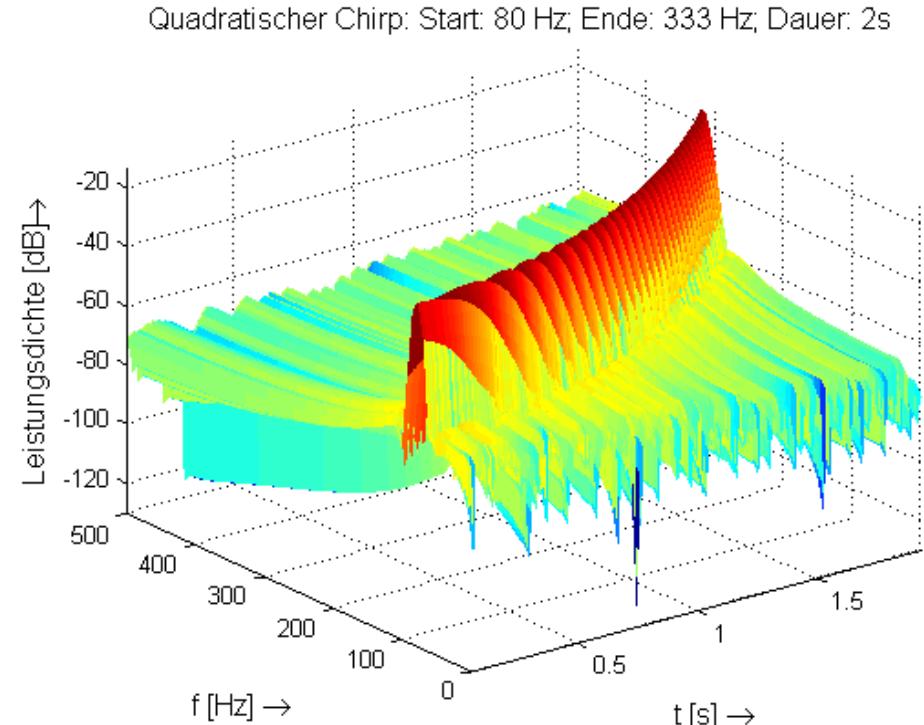
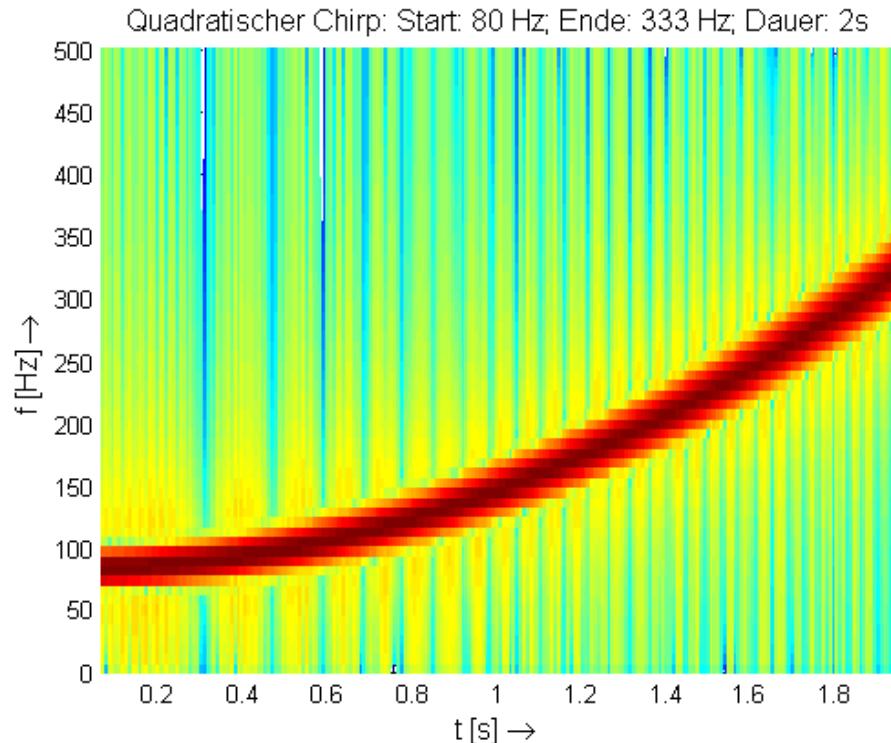
# Spektrogramm: Beispiel

Quelle: Dr. S. Wyrsch, ZHAW

Ein Chirp-Signal durchläuft während 2s die Frequenzen 80-333 Hz. Die Kurzzeit-FFTs haben eine Blocklänge von  $N = 128$  und überlappen mit 120 Samples.



Darstellung der Intensität als 2D & 3D-Plot. Abtastfrequenz  $f_s = 1000$  Hz. MATLAB-Befehl: spectrogram



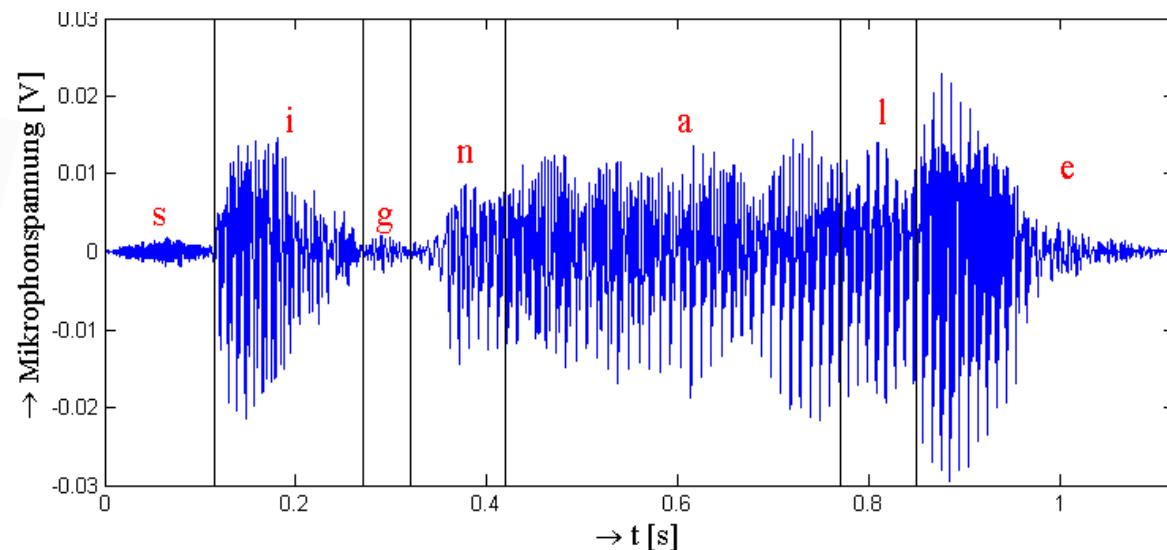
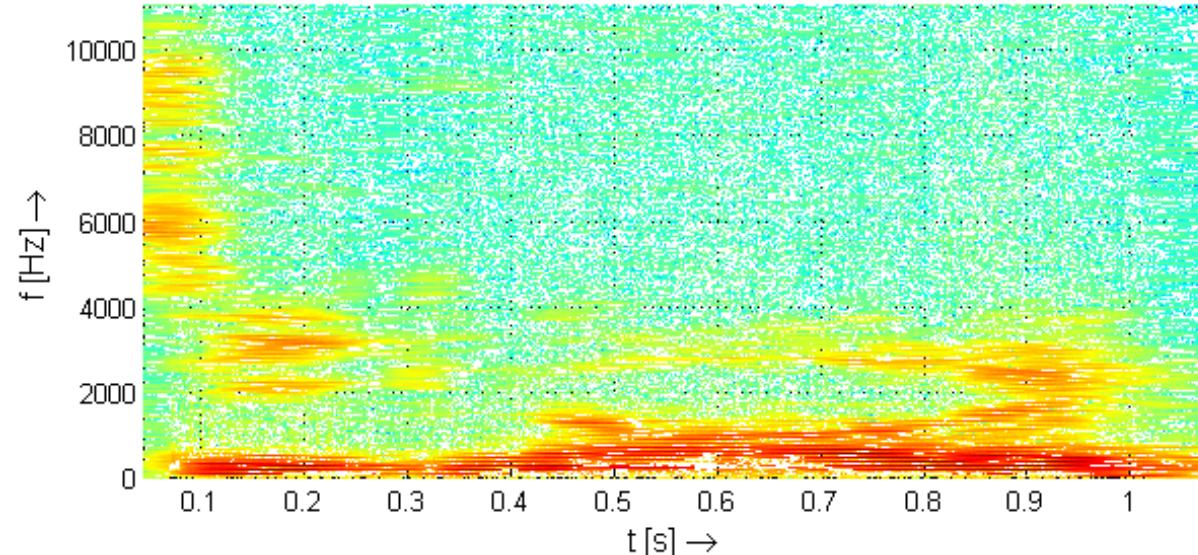
# Spektrogramm: Beispiel

Spektrogramm mit  $N=2048$  FFTs (Überlappung 2000 Samples).  $f_s = 22050$  Hz.

Wie «gross»  
ist ein Pixel?



Sprecher/Quelle:  
Dr. S. Wyrsh, ZHAW



Mit der **DFT** können nur Spektralkomponenten  $X[m]$  analysiert werden, die im **äquidistanten Frequenzraster**  $f = m \cdot f_s / N$ ,  $m = 0, \dots, N-1$  liegen.

Manchmal interessiert aber eine Spektralkomponente  $X(f_0)$  bei einer fixen Frequenz  $f_0$ , die nicht auf dem DFT/FFT-Frequenzraster liegt.

Die Spektralkomponente  $X(f_0)$  des mit  $f_s$  abgetasteten Signals  $x[n]$  kann im Zeitfenster  $[0, NT_s)$  wie folgt approximiert werden:

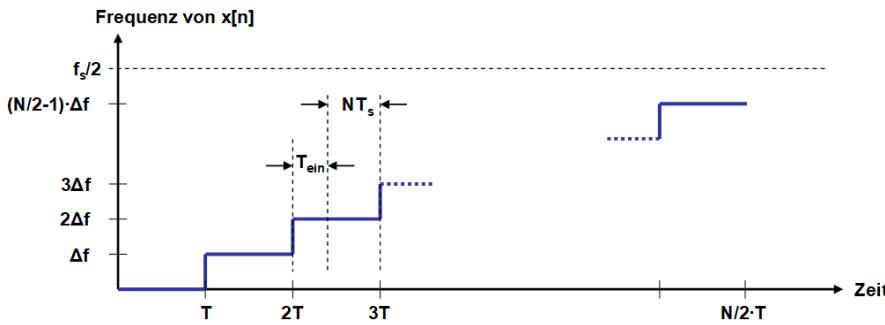
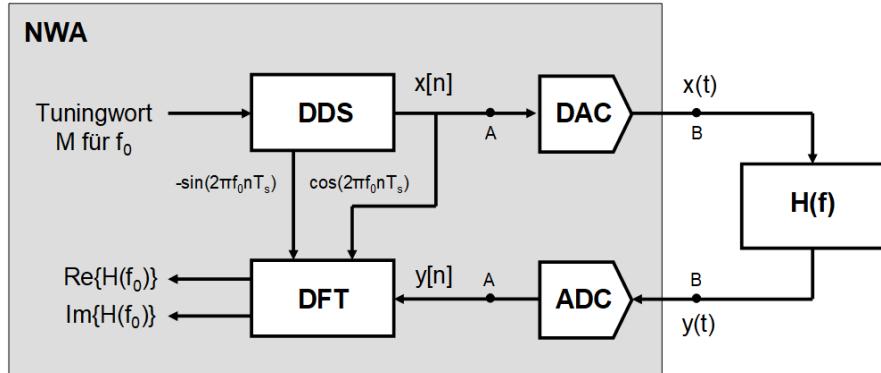
$$X(f_0) \cong \sum_{n=0}^{N-1} x[n] \cdot e^{-j2\pi \cdot f_0 \cdot n T_s}$$

Der Frequenz-Bin  $X(f_0)$  hat natürlich auch die Bandbreite  $\Delta f = f_s / N$  (mit windowing breiter).

Mit der Euler-Formel gilt alternativ:

$$X(f_0) \cong \sum_{n=0}^{N-1} x[n] \cdot \cos(2\pi f_0 n T_s) - j \cdot \sum_{n=0}^{N-1} x[n] \cdot \sin(2\pi f_0 n T_s)$$

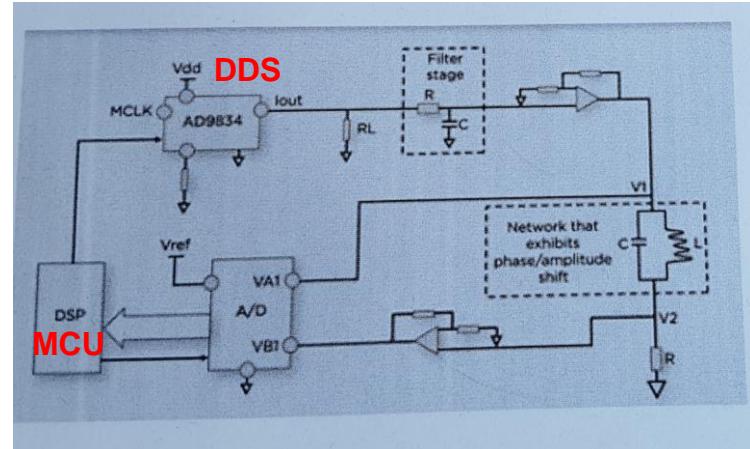
# Beispiel Networkanalyzer (NWA)



$$H[m] = \frac{Y[m]}{X[m]} = \frac{Y[m]}{1/2} \approx 2 \cdot \sum_{n=0}^{N-1} y[n] \cdot e^{-j2\pi f_0 n T_s} = 2 \cdot \sum_{n=0}^{N-1} y[n] \cdot e^{-j\frac{2\pi}{N} mn}$$

$$H[m] \approx 2 \cdot \underbrace{\sum_{n=0}^{N-1} y[n] \cdot \cos\left(\frac{2\pi}{N} mn\right)}_{\text{Re}\{H[m]\}} - j \cdot 2 \cdot \underbrace{\sum_{n=0}^{N-1} y[n] \cdot \sin\left(\frac{2\pi}{N} mn\right)}_{\text{Im}\{H[m]\}} .$$

## Beispiel Impedanzmessung



## Inhalt

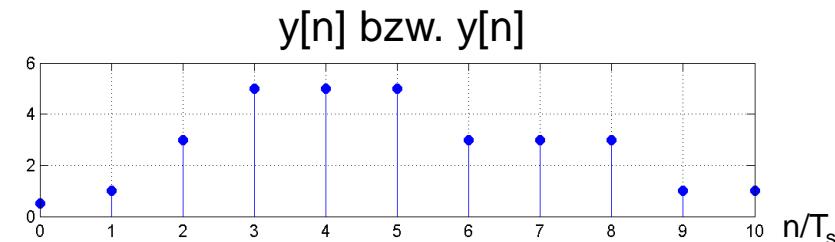
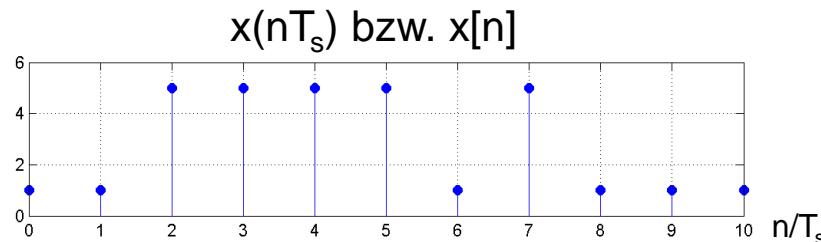
1. Einleitung
2. Diskrete LTI-Systeme (LTD-Systeme)
3. Differenzengleichung
4. Impulsantwort und Faltungssumme
5. z-Transformation
6. Übertragungsfunktion (UTF)
7. Zusammenfassung LTD-Systeme

## Bücher

- [1] A.V. Oppenheim, R.W. Schafer: „Discrete-Time Signal Processing“, 3rd Edition, Pearson, 2010. Oppenheim und Schafer sind DSP-Pioniere.
- [2] D.G. Manolakis, V.K. Ingle, "Applied Digital Signal Processing", Cambridge University Press, 2011, Kapitel 3.

# 1. Einleitung

Ein **digitales System** transformiert eine diskrete Eingangszahlenfolge  $x[n]$  mit einem Algorithmus in eine diskrete Ausgangszahlenfolge  $y[n]$ .



**Vor- und Nachteile** gegenüber analogen Systemen:

- + Reproduzierbarkeit, Flexibilität, Vorhersagbarkeit, Komplexität, Integrierbarkeit, Erweiterbarkeit ...
- ± Geschwindigkeit, Preis von «einfachen»-Lösungen

## 2. LTD-Systemen und Eigenschaften

Betrachten nur "linear, time-invariant, discrete" **LTD**-Systeme.

Die Beschreibung im Zeit- und Frequenzbereich ist eng verwandt mit der Beschreibung der analogen LTI-Systeme.

Für ein **lineares**, zeitdiskretes System gilt das Superpositionsprinzip:

- wenn  $x_1[n] \rightarrow y_1[n]$  und  $x_2[n] \rightarrow y_2[n]$   
dann  $k_1 \cdot x_1[n] + k_2 \cdot x_2[n] \rightarrow k_1 \cdot y_1[n] + k_2 \cdot y_2[n]$

linearkombination der inputs folgt linearkom der outputs  
=> linearität

Für ein **zeit-invariantes**, zeitdiskretes System gilt:

- wenn  $x[n] \rightarrow y[n]$  dann  $x[n-k] \rightarrow y[n-k]$  für alle k

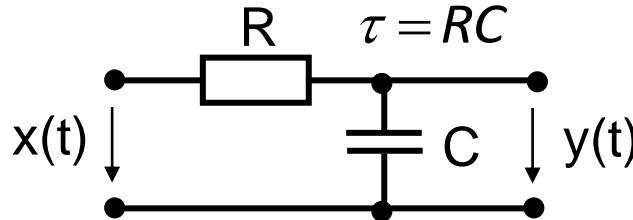
Ein LTD-System ist **kausal**, wenn der n-te Eingangswert  $x[n]$  nur den aktuellen und die zukünftigen Ausgangswerte  $y[n], y[n+1], \dots$  beeinflusst.

Ein LTD-System ist **(BIBO) stabil**, wenn es beschränkte Eingangsfolgen ( $|x[n]| < \infty$  für alle n) in beschränkte Ausgangsfolgen transformiert.

### 3. Differenzengleichung



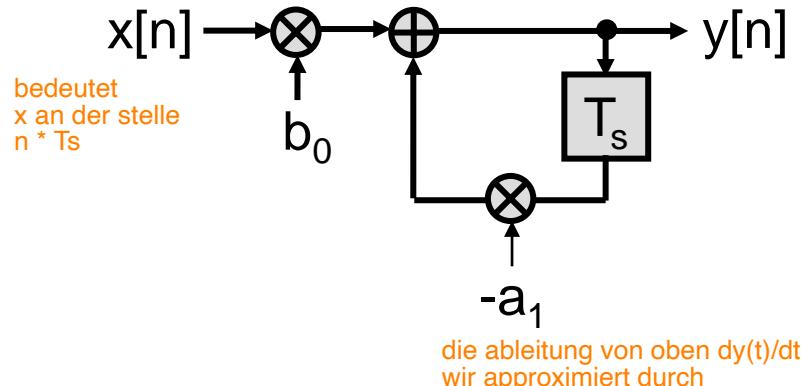
Analoge LTI-Systeme können mit Differentialgleichungen beschrieben werden



$$y(t) = x(t) - \tau \cdot dy(t)/dt$$

LTD-Systeme können mit Differenzengleichungen beschrieben werden.

Die b,a-Systemkoeffizienten charakterisieren das LTD-System vollständig.



$$y[n] = b_0 \cdot x[n] - a_1 \cdot y[n-1]$$

Mit der Approximation  
 $dy(t)/dt \approx [y(nT_s) - y((n-1) \cdot T_s)] / T_s$   
folgt  $b_0 = T_s / (T_s + \tau)$  und  $a_1 = -\tau / (T_s + \tau)$

$$y(t) = x(t) - \overbrace{\frac{dy(t)}{dt}}^{\text{approximation}} \quad \left. \frac{y[n] - y[n-1]}{T_s} \right)$$



$$y[n] = x[n] - \frac{T}{T_s} (y[n] - y[n-1])$$

$$y[n] \underbrace{\left(1 + \frac{T}{T_s}\right)}_{\text{approximation}} = x[n] + \frac{T}{T_s} y[n-1]$$

$$y[n] = \underbrace{\frac{1}{1 + \gamma/T_s} x[n]}_{b_0} + \underbrace{\frac{\gamma/T_s}{1 + \gamma/T_s} y[n-1]}_{a_1}$$

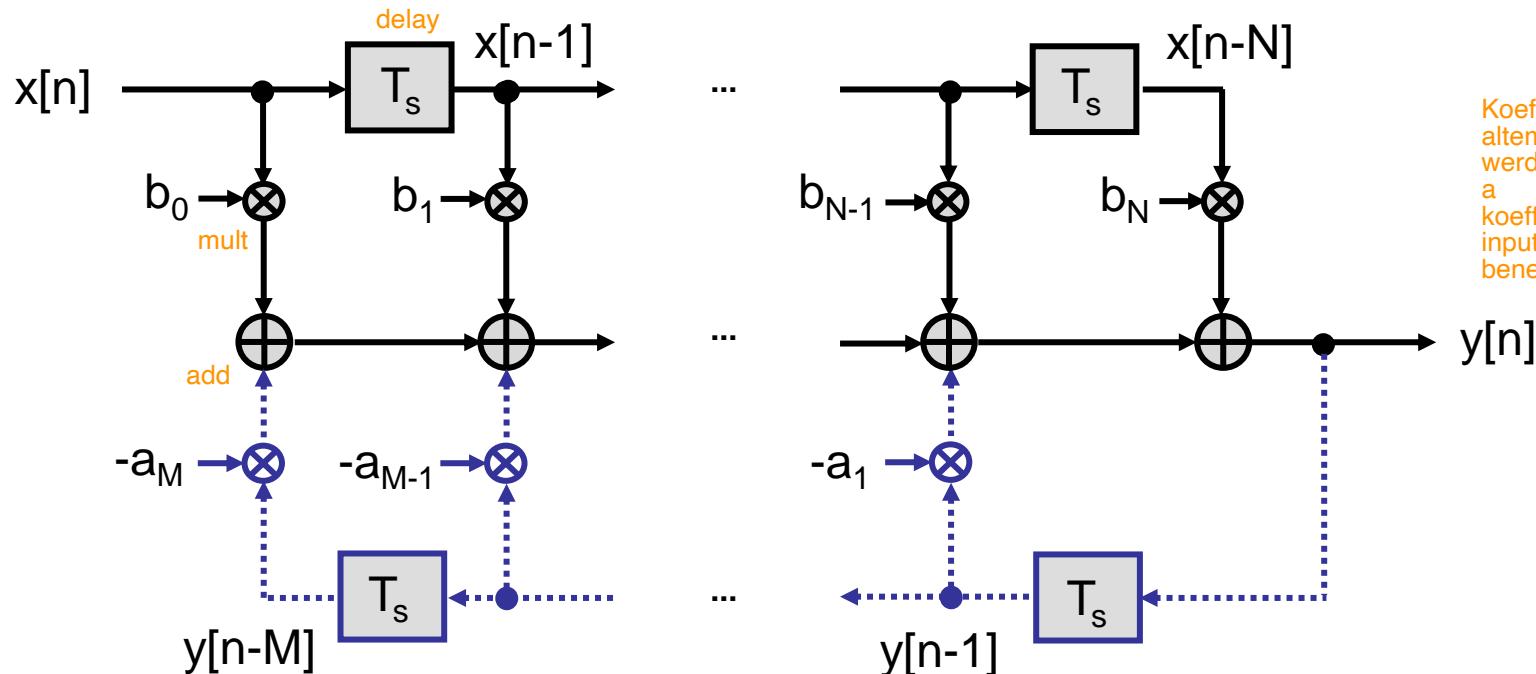
# 3. Differenzengleichung

wichtige Folie

## Allgemeine Form der Differenzengleichung eines LTD-Systems

$$y[n] = \sum_{k=0}^N b_k \cdot x[n - k] - \sum_{k=1}^M a_k \cdot y[n - k]$$

LTD-System besteht nur aus Elementen für Addition, Multiplikation und Zeitverzögerung

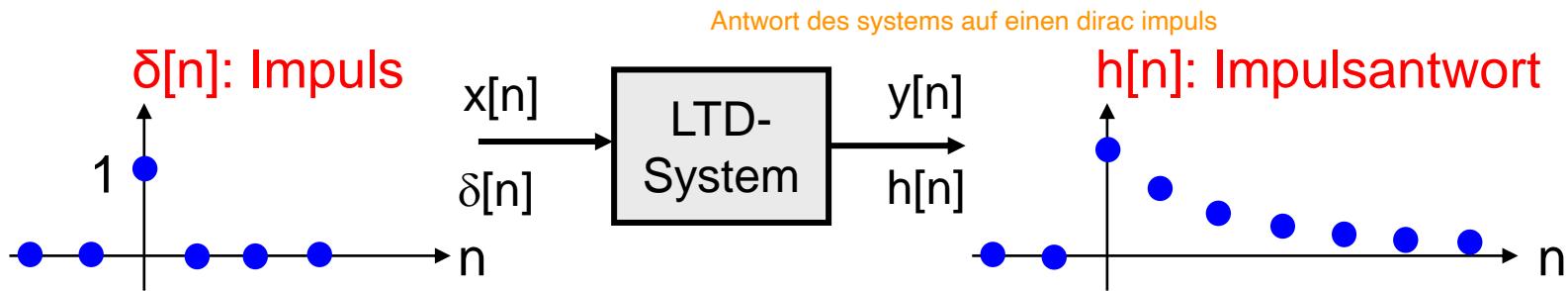


Koeffizienten welche mit altem signal verrechnet werden nennt man immer a  
koeffizienten welche mit input verrechnet werden benannt man b

FIR = Finite impulse response filter

- Nicht-rekursives Teil-System (**FIR**-/Transversalfilter, Tapped-Delay-Line) ohne blauen teil
- ..... Rekursives Teil-System (Bestandteil eines **IIR**-Filters) mit blauem teil

## 4. Impulsantwort und Faltungssumme



Bestimmung der Ausgangsfolge  $y[n]$  für beliebige Eingangsfolgen  $x[n]$

Zerlegung in Einzelimpulse:  $x[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot \delta[n-k]$  immer null außer bei  $n=k$

Antwort auf Impuls bei  $n=0$ :  $x_0[n] = x[0] \cdot \delta[n] \rightarrow y_0[n] = x[0] \cdot h[n]$

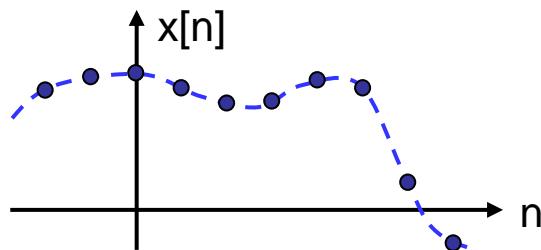
Antwort auf Impuls bei  $n=k$ :  $x_k[n] = x[k] \cdot \delta[n-k] \rightarrow y_k[n] = x[k] \cdot h[n-k]$

Summe der Antworten:  $y[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[n-k] = \sum_{k=-\infty}^{\infty} h[k] \cdot x[n-k]$

**Faltungssumme**

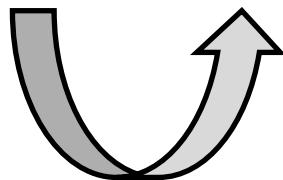
MATLAB: conv, filter

## 4. Impulsantwort und Faltungssumme



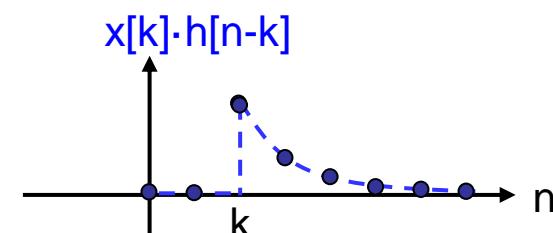
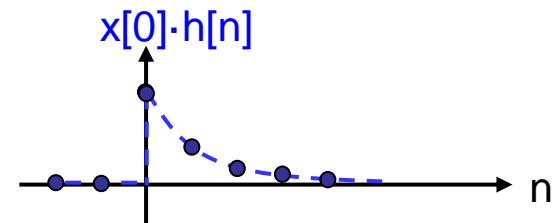
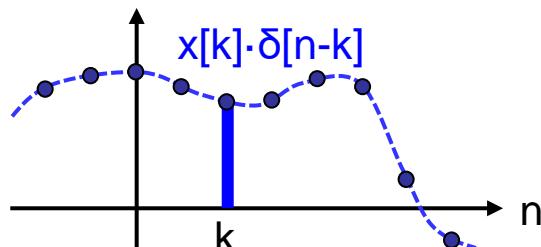
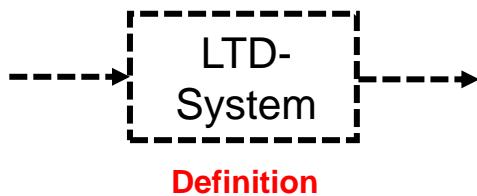
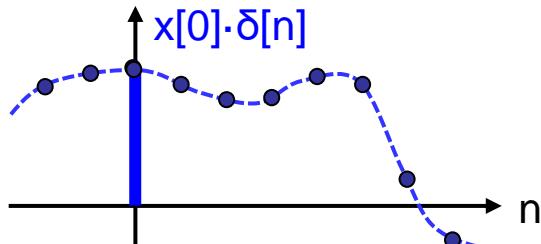
$$y[n] = x[n] * h[n] = h[n] * x[n]$$

$x[n]$  = Superposition von zeit-verschobenen und gewichteten Dirac-Impulsen



Faltungssumme

$$y[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[n-k]$$



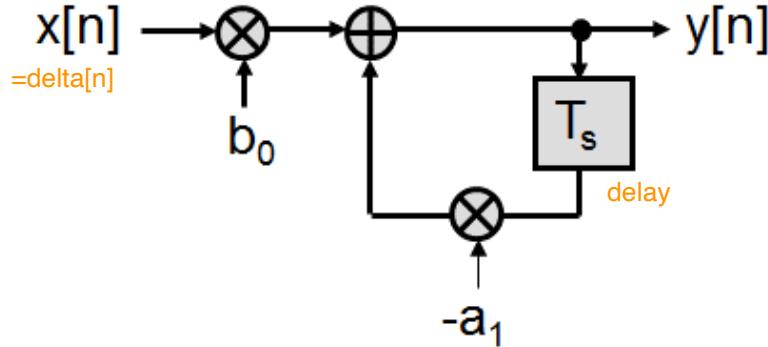
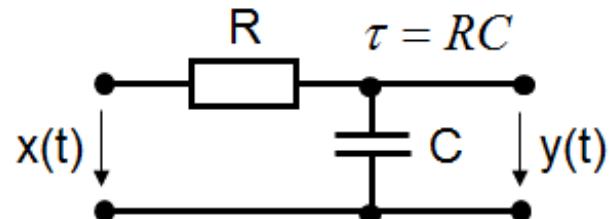
## 4. Beispiel zur Faltung

### Berechnung der Schrittantwort eines Systems 1. Ordnung

- Approximation RC-Tiefpass mit  $\tau = RC = 1\text{s}$ ,  $f_s = 1/T_s = 10\text{ Hz}$

$$\rightarrow b_0 = T_s/(T_s + \tau) = 0.1/1.1 = 0.0909$$

$$\rightarrow a_1 = -\tau/(T_s + \tau) = -1/1.1 = -0.9091$$



Input: 1, 0 , 0, 0, 0, 0,

- Impulsantwort  $h[n] = \{b_0, -b_0 \cdot a_1, b_0 \cdot a_1^2, -b_0 \cdot a_1^3, \dots\}$  für  $n \geq 0$ .  
numerisch:  $h[n] = b_0 \cdot (-a_1)^n = 0.0909 \cdot (0.9091)^n$  für  $n \geq 0$

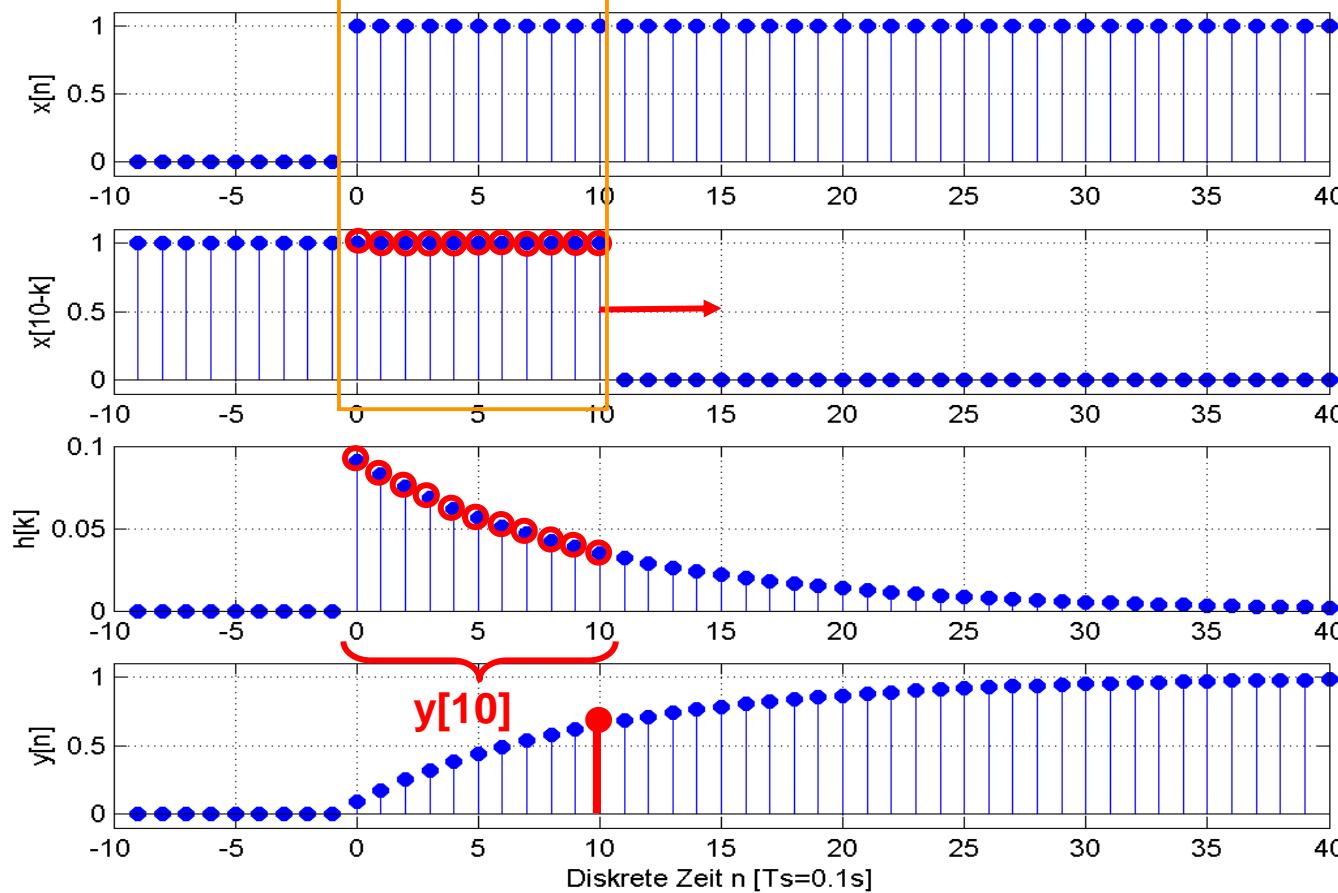
Impulsantwort ist wichtig weil man dann weiß wie das System auf einen impuls reagiert.

somit kann der ausgang berechnet werden anhand des eingangs.

# 4. Beispiel zur Faltung

## Berechnung der Schrittantwort eines Systems 1. Ordnung

Faltungssumme:  $y[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[n-k] = \sum_{k=-\infty}^{\infty} h[k] \cdot x[n-k]$



```
b = 0.0909; a = [1 -0.9091]; x = ones(1,40); y = filter(b,a,x); stem([0:39],y)
```

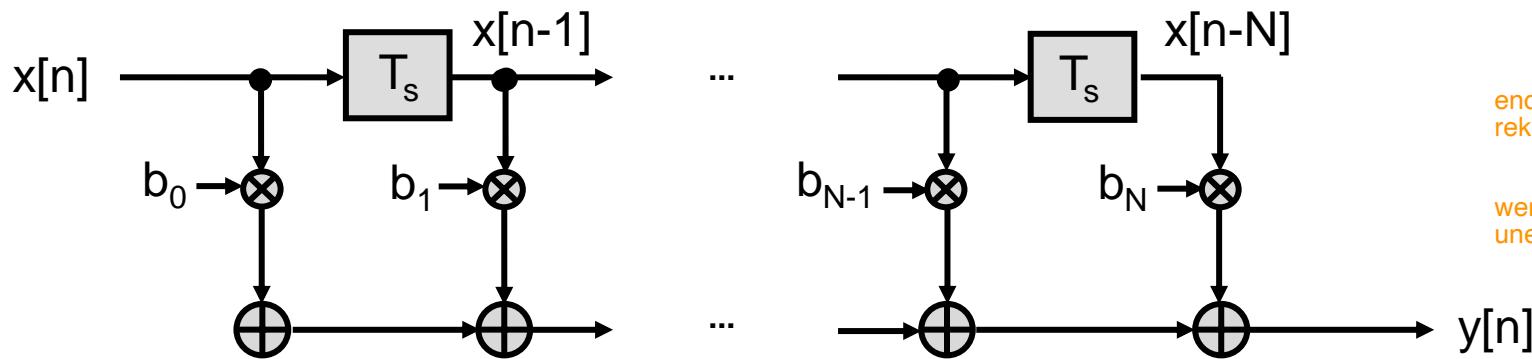
## 4. Impulsantwort eines FIR-Filters

### FIR-Filter bzw. nicht-rekursives LTD-System (Tapped-Delay-Line)

hängt nicht von alten Werten ab.

$$y[n] = \sum_{k=0}^N b_k \cdot x[n - k] - \sum_{k=1}^M a_k \cdot y[n - k]$$

Impulsantwort bedeutet immer Input =  
1 0 0 0 0 0 0 0 ....



endlich lang für nicht  
rekursive...

wenn rekursiv ist sie  
unendlich lang..

Die Impulsantwort stimmt mit den b-Filterkoeffizienten überein

$$h_{\text{FIR}}[n] = \{b_0, b_1, \dots, b_N\} \quad \text{für } n = 0, \dots, N$$

und ist endlich lang (**finite impulse response bzw. FIR-Filter**).

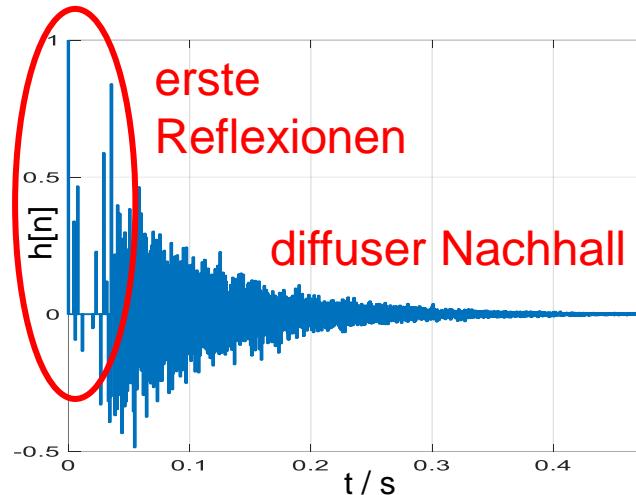
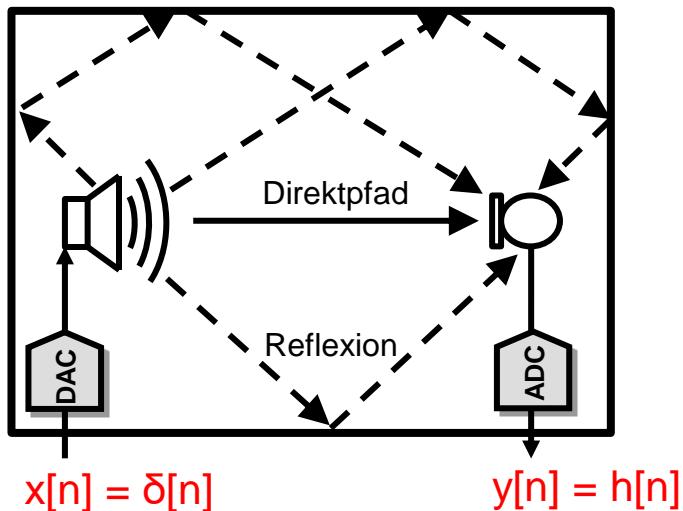
FIR-Filter realisieren die **endlich-lange** Faltungssumme

Es kann zum Beispiel mit einem Klatschen die impuls antwort eines  
gebäudes gemessen werden. Jetzt kann man eine Sängerin denn Hall des  
gebäudes adden -weil die Impulsantwort bekannt ist.

$$y[n] = \sum_{k=0}^N h[k] \cdot x[n - k]$$

## 4. Akustisches Beispiel zur Faltung

- 1) Messung der Raum-Impulse-Response (RIR) z.B. mit einem Knall.



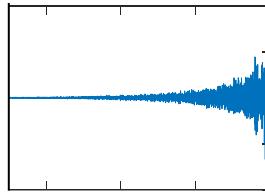
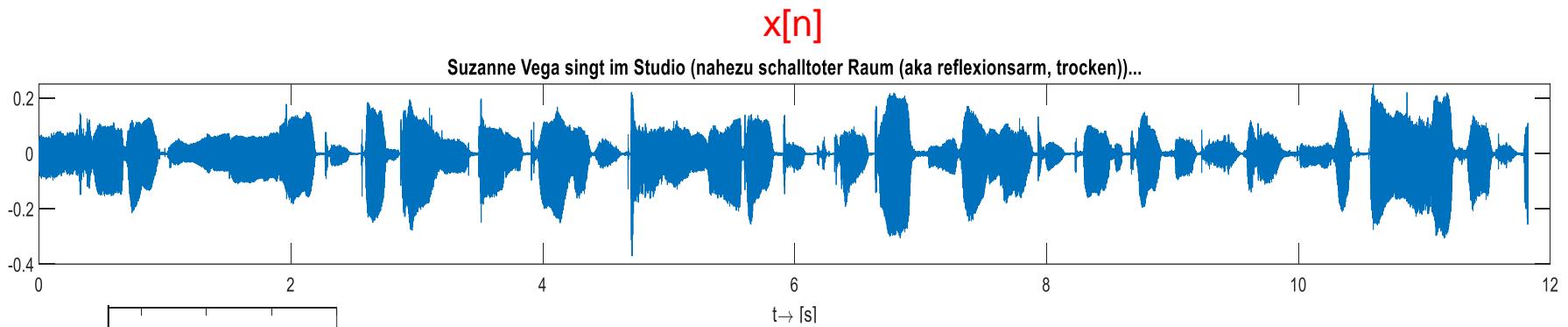
- 2) Studioaufnahme (trocken, reflexionsarm):  
=> Suzanne Vega singt im Studio «Tom's Diner»



- 3) Filterung Studioaufnahme mit RIR  
Suzanne Vega singt (vermeintlich)  
in der Kirche.

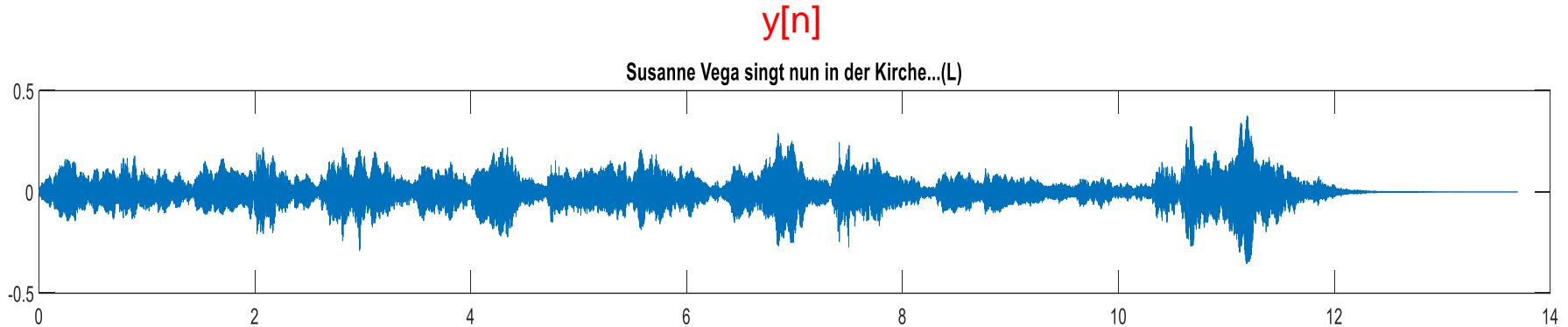


# 4. Akustisches Beispiel zur Faltung



→ Faltung bzw.  
FIR-Filterung ( $b_k = h_{\text{Kirche}}[k]$ )

$$y[n] = \sum_{k=0}^N b_k \cdot x[n - k]$$



# 5. z-Transformation

**Laplace-Transformation** diskretes Signal

=> fourier trasnformation s mit  $j \cdot 2 \cdot \pi \cdot f$  ersetzen

$$X(s) = \sum_{n=-\infty}^{\infty} x[n] \cdot e^{-nsT_s}$$

**Substitution**

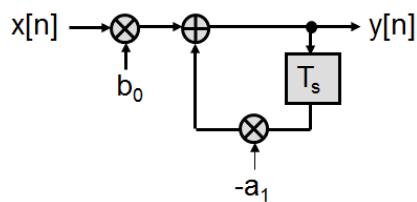
$$z = e^{sT_s}$$

**z-Transformierte** von diskretem Signal

$$X(z) = \sum_{n=-\infty}^{\infty} x[n] \cdot z^{-n}$$

**Beispiel:**

LTD-System 1. Ordnung



$$h[n] = \{b_0, -b_0 \cdot a_1, b_0 \cdot a_1^2, -b_0 \cdot a_1^3, \dots\} \quad \text{für } n \geq 0.$$

$$H(z) = b_0 - b_0 \cdot a_1 \cdot z^{-1} + b_0 \cdot (a_1)^2 \cdot z^{-2} - + \dots$$

Reminder geometrische Reihe:  $1 + q + q^2 + \dots = 1/(1-q)$

$$H(z) = b_0 \cdot (1 - a_1 \cdot z^{-1} + a_1^2 \cdot z^{-2} - + \dots) = \frac{b_0}{1 + a_1 \cdot z^{-1}}$$

## 5. z-Transformation

Fourier-/Laplace-Transformation:  $X(f) = X(s = j2\pi f)$

Laplace-/z-Transformation:  $X(s) = X(z = e^{sT_s})$

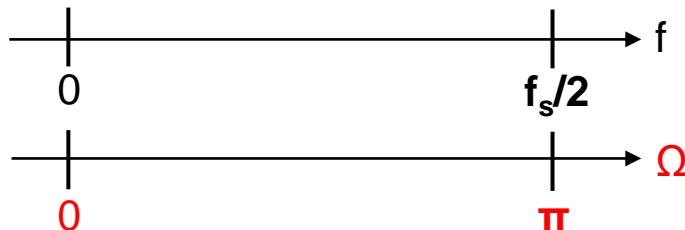
von z- zu Fourier-Transformation:  $X(f) = X(z = e^{j2\pi f T_s})$

Das Spektrum  $X(f)$  des diskreten Signals  $x[n]$  findet man durch Ersetzen von  $z$  durch  $\exp(j2\pi \cdot f / f_s)$  in der z-Transformierten  $X(z)$ !

DSV-Literatur

Verwendung der **normierten Frequenz**  $\Omega = 2\pi \cdot f / f_s = \pi \cdot f / (f_s/2)$

$$H(f) = H(z = e^{j2\pi f T_s}) = H(z = e^{j2\pi f / f_s}) = H(z = e^{j\Omega})$$



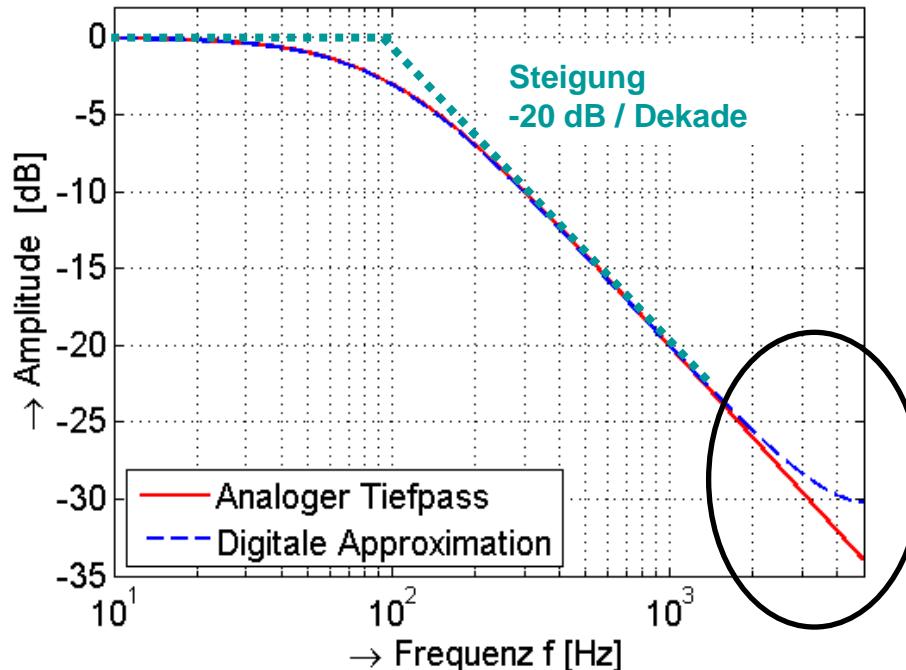
## 5. z-Transformation

### Beispiel LTD-System 1. Ordnung

Approximation RC-Tiefpass mit  $f_g = 1/(2\pi \cdot \tau) = 100 \text{ Hz}$ ,  $f_s = 10 \text{ kHz}$

=> LTD-System mit Koeffizienten  $b_0 = 0.06$  und  $a_1 = -0.94$

$$=> \text{Frequenzgang } H(f) = \frac{b_0}{1 + a_1 \cdot e^{-j2\pi f T_s}}$$



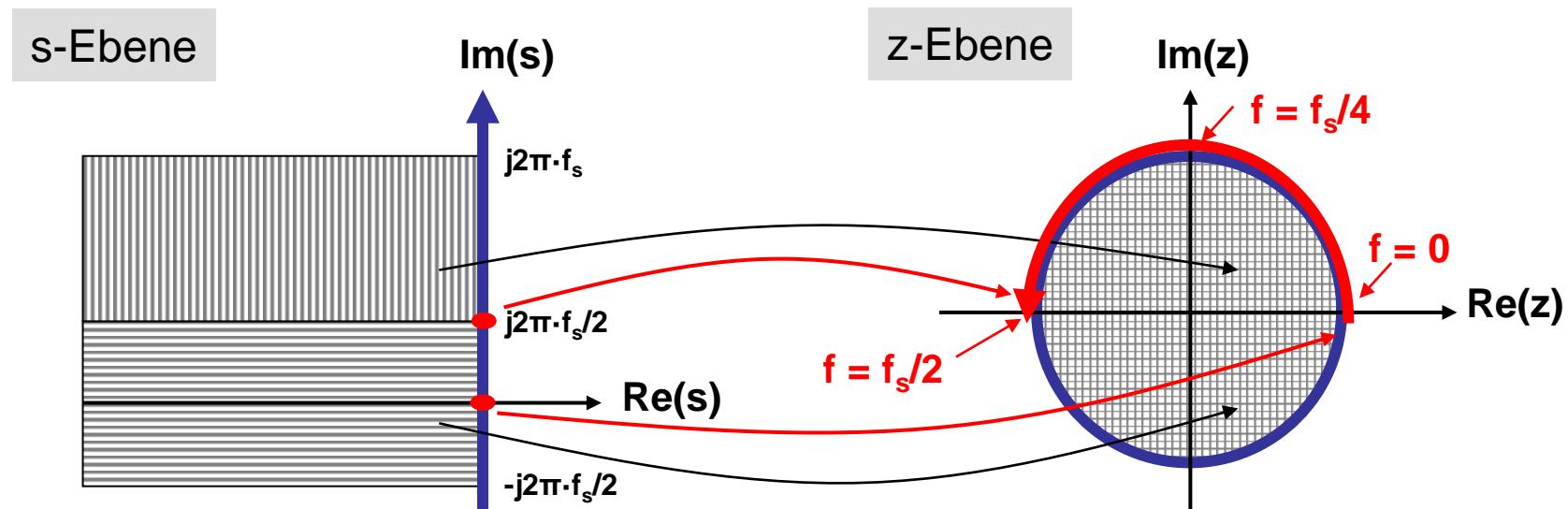
..... Bode-Approximation

Können Sie diese  
Abweichung erklären?

weil im digitalen spektrum die frequenz periodisch ist.

## 5. z-Transformation der s-Ebene

Substitution:  $z = e^{sT_s}$  → konforme Abbildung von  $s$  auf  $z$



- Imaginäre  $s$ -Achse wird mehrfach auf  $z$ -Einheitskreis abgebildet
- Linke  $s$ -Halbebene wird mehrfach in den  $z$ -Einheitskreis abgebildet
- Rechte  $s$ -Halbebene wird mehrfach um den  $z$ -Einheitskreis abgebildet

# 5. Eigenschaften der z-Transformation

auch d transformation gennant (für delay)

Linearität (!)

$$a \cdot x_1[n] + b \cdot x_2[n] \xrightarrow{\hspace{1cm}} a \cdot X_1(z) + b \cdot X_2(z)$$

Zeitverschiebung (!)

wichtig

$$x[n-k] \xrightarrow{\hspace{1cm}} z^{-k} \cdot X(z)$$

Faltung (!)

wichtig

$$x[n] * h[n] \xrightarrow{\hspace{1cm}} X(z) \cdot H(z)$$

Multiplikation mit Exponentialfolge

$$a^n \cdot x[n] \xrightarrow{\hspace{1cm}} X(z/a)$$

Multiplikation mit der Zeit

$$n \cdot x[n] \xrightarrow{\hspace{1cm}} -z \cdot dX(z)/dz$$

Zeitumkehr

$$x[-n] \xrightarrow{\hspace{1cm}} X^*(1/z^*)$$

Anfangswerttheorem für einseitige z-Trafo

$$x[0] = \lim_{z \rightarrow \infty} X(z)$$

Endwerttheorem für einseitige z-Trafo

$$\lim_{n \rightarrow \infty} x[n] = \lim_{z \rightarrow 1} (z - 1) \cdot X(z)$$

# 5. Inverse z-Transformation

nicht so wichtig nur der vollständigkeitshalber

**Inverse z-Transformation** hat in der Praxis nur wenig Bedeutung

$$x[n] = \frac{1}{2\pi j} \oint_C X(z) \cdot z^{n-1} dz$$

Das **C** in der Gleichung steht für „contour“ und stellt eine **geschlossene Kurve** um den Nullpunkt dar, die im Gegenuhrzeigersinn durchlaufen werden muss.

## Einfachere Alternativen

- Tabellenverfahren
- Integral-Inversionsformel, ...
- Partialbruchzerlegung, z.B. für kausale Systeme

$$H(z) = A_0 + \sum_{k=1}^N \frac{A_k \cdot z}{z - p_k} \longrightarrow h[n] = A_0 \cdot \delta[n] + \sum_{k=1}^N A_k \cdot p_k^n \quad (n \geq 0)$$

- Fortlaufende Division (z.B. rekursives LTD-System 1. Ordnung)

$$(b_0 \cdot z) : (z+a_1) = b_0 - b_0 a_1 \cdot z^{-1} + b_0 \cdot a_1^2 \cdot z^{-2} \dots$$

$$\begin{array}{r} \underline{b_0 z + b_0 a_1} \\ -b_0 a_1 \\ \underline{-b_0 a_1 - b_0 a_1^2 \cdot z^{-1}} \\ b_0 a_1^2 \cdot z^{-1} \end{array}$$

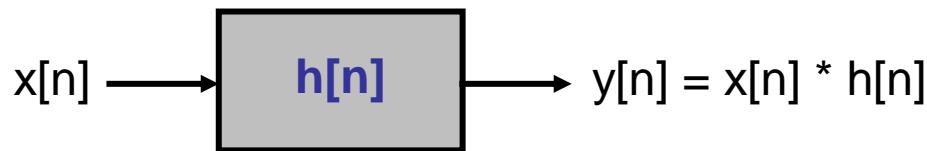
# 6. Übertragungsfunktion (UTF)

Das Ausgangssignal  $y[n]$  eines LTD-Systems kann durch Faltung des Eingangssignals  $x[n]$  mit der Impulsantwort  $h[n]$  gebildet werden.

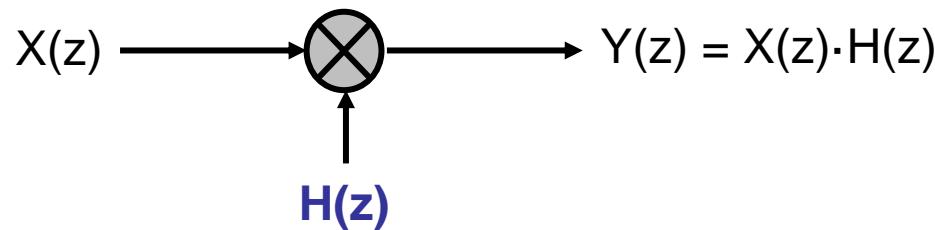
Faltung im Zeitbereich

 Multiplikation im z-Bereich  
z-Trafo

Zeitbereich



z-Bereich



Impulsantwort  $h[n]$

 Übertragungsfunktion  $H(z) = \frac{Y(z)}{X(z)}$   
z-Trafo

# 6. Übertragungsfunktion (UTF)

Der Zusammenhang zwischen dem Ein- und Ausgangssignal  $x[n]$  und  $y[n]$  eines LTD-Systems kann mit einer **Differenzengleichung** beschrieben werden.

## Differenzengleichung

$$y[n] = \sum_{k=0}^N b_k \cdot x[n-k] - \sum_{k=1}^M a_k \cdot y[n-k]$$

z-Trafo

hier minus

$$Y(z) = \sum_{k=0}^N b_k \cdot z^{-k} \cdot X(z) - \sum_{k=1}^M a_k \cdot z^{-k} \cdot Y(z)$$

## UTF

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^N b_k \cdot z^{-k}}{1 + \sum_{k=1}^M a_k \cdot z^{-k}} = \frac{\overbrace{b_0 + b_1 \cdot z^{-1} + \dots + b_N \cdot z^{-N}}^{\text{Zähler-Polynom } B(z)}}{\underbrace{1 + a_1 \cdot z^{-1} + \dots + a_M \cdot z^{-M}}_{\text{Nenner-Polynom } A(z)}}$$

da plus

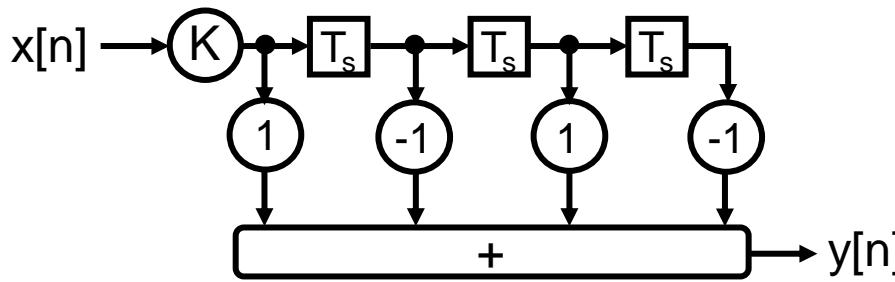
Nenner-Polynom  $A(z)$

# 6. Übertragungsfunktion (UTF)

## Pol-Nullstellen-Darstellung der UTF

(Polynom N-ter Ordnung hat N Nullstellen wenn Polynom-Koeffizienten reell sind, sind die Nullstellen reell oder treten in konjugiert-komplexen Paaren auf)

### Beispiel



$$\text{Impulsantwort } h[n] = \{K, -K, K, -K\}$$

$$H(z) = K \cdot (1 - z^{-1} + z^{-2} - z^{-3})$$

$$H(z) = K \cdot (z^3 - z^2 + z - 1) / z^3$$

$$H(z) = K \cdot (z-1) \cdot (z^2 + 1) / z^3$$

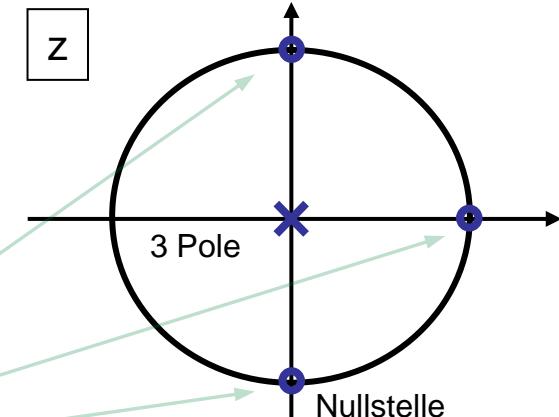
hat Wurzeln bei 1 und +/- i

$$H(z) = b_0 \frac{\prod_{k=1}^N (z - z_k)}{\prod_{k=1}^M (z - p_k)} \cdot z^{M-N}$$

↑ Nullstelle der UTF  
↑ Pol der UTF

Hier werden die Polynome der beiden Funktionen der UTF als Produkt der Nullstellen dargestellt

$$H(z) = K \cdot (z-1) \cdot (z+j) \cdot (z-j) / z^3$$



# 6. Übertragungsfunktion (UTF)

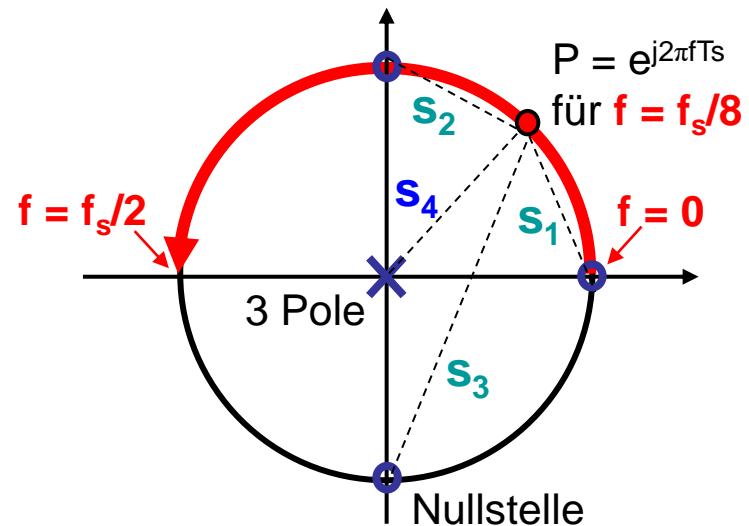
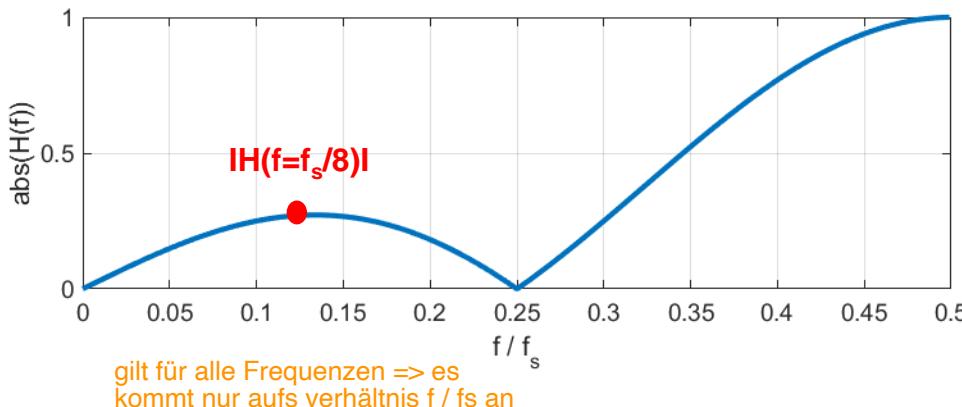
Den **Frequenzgang**  $H(f)$  erhält man durch Auswerten der UTF  $H(z)$  auf dem Einheitskreis, d.h.  $H(f) = H(z = e^{j2\pi f T_s})$ , bzw. dem oberen Halbkreis.

$$\text{Amplitudengang } |H(f)| = K \frac{\prod_{k=1}^N |e^{j2\pi f T_s} - z_k|}{\prod_{k=1}^M |e^{j2\pi f T_s} - p_k|}$$

Abstand Punkt  $P = e^{j2\pi f T_s}$  auf dem Einheitskreis zum Pol  $p_k$

## Beispiel (Fortsetzung)

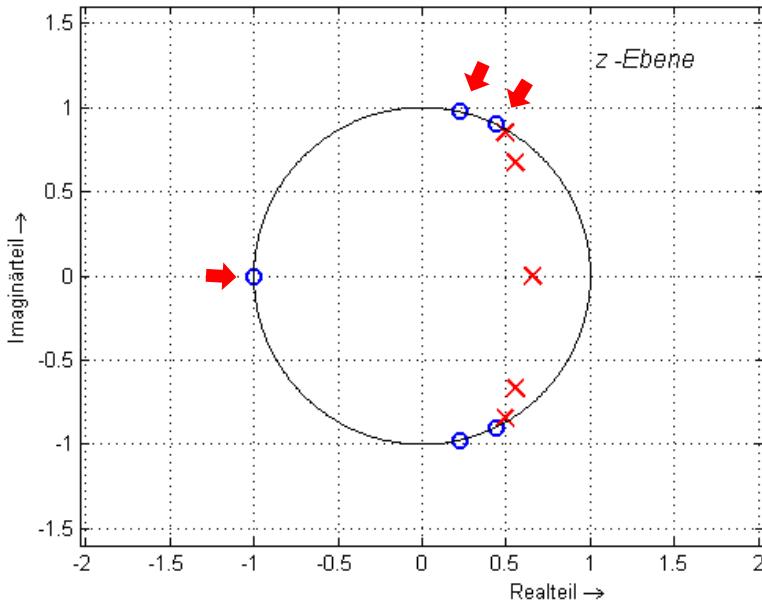
$$\text{UTF } H(z) = (1/4) \cdot (z-1) \cdot (z+j) \cdot (z-j) / z^3$$



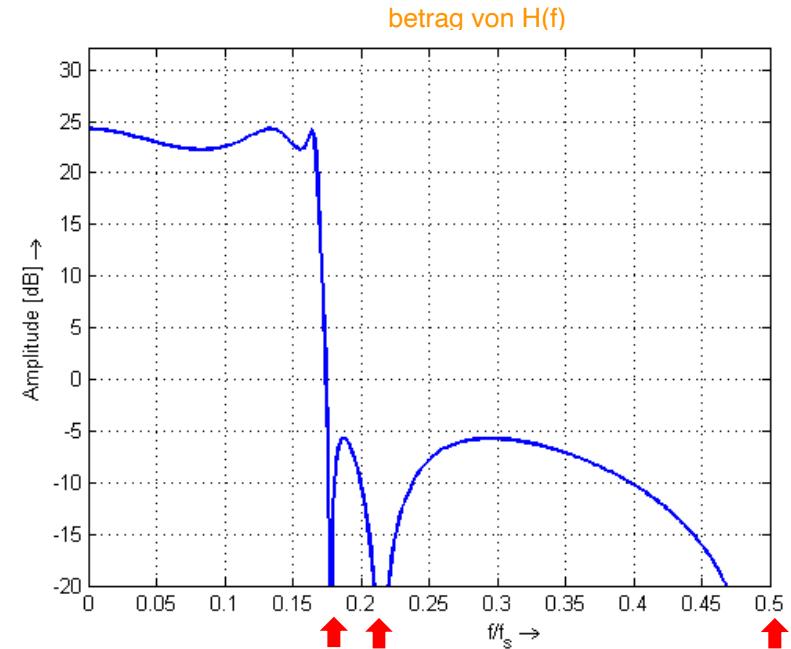
$$|H(f=f_s/8)| = K \cdot s_1 \cdot s_2 \cdot s_3 / s_4^3$$

# 6. Übertragungsfunktion (UTF)

PN-Darstellung von  $H(z)$



$\Leftrightarrow$  Amplitudengang  $|H(f)|$



Lage der Pole/Nst. (ohne Beweis, siehe analoge Systeme)

- Pole (Nullstellen) sind reell oder kommen in konjugiert-komplexen Paaren vor
- Die **Pole eines stabilen LTD-Systems** liegen **innerhalb des Einheitskreises**.  
=> alle FIR systeme sind stabil
- Die **Nullstellen** können grundsätzlich **überall in der z-Ebene** liegen.
- FIR-Systeme besitzen nur Nullstellen (bzw. alle Pole sind im Ursprung)  
=> daher division / 1

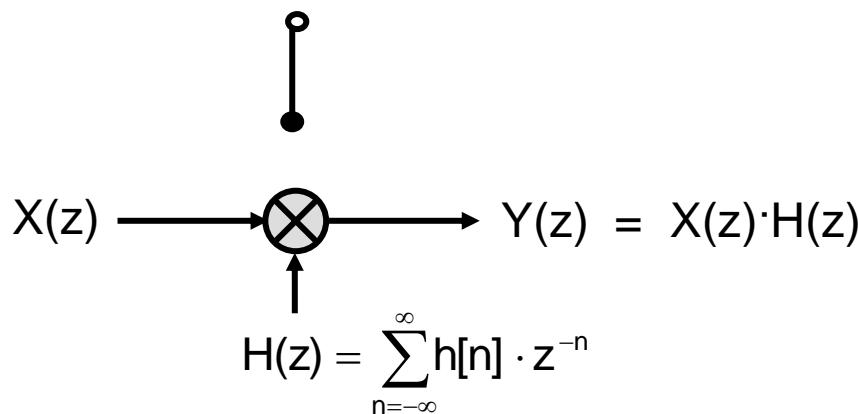
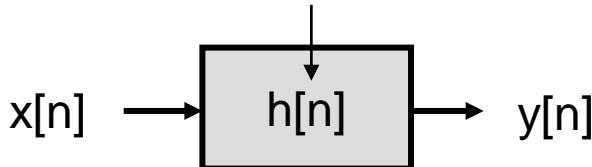
# Zusammenfassung LTD-Systeme

wichtigste Folie

LTD = linear time discrete

Impuls antwort =&gt; 1 und dann alles nullen im eingang

## Impulsantwort



Übertragungsfunktion

$$\text{UTF} \quad H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + \dots + b_N z^{-N}}{1 + a_1 z^{-1} + \dots + a_M z^{-M}}$$

## Frequenzgang

$$H(f) = H(z = e^{j2\pi f T_s}) = \sum_{n=-\infty}^{\infty} h[n] \cdot e^{-jn2\pi f T_s}$$

z setzen wir als  $e^{(i * 2\pi f * T_s)}$  =>  
dann erhalten wir frequenzgang

## Faltungssumme

$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{\infty} h[k] \cdot x[n-k]$$

## Differenzengleichung ( $a_0=1$ )

$$y[n] = \sum_{k=0}^N b_k \cdot x[n-k] - \sum_{k=1}^M a_k \cdot y[n-k]$$

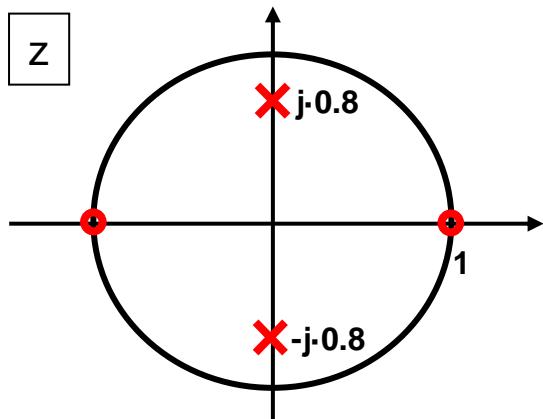
fängt bei 1 an! ->  
a0 komponente ist nicht dabei

## Pol-Nullstellen-Darstellung

$$H(z) = b_0 \frac{\prod_{k=1}^N (z - z_k)}{\prod_{k=1}^M (z - p_k)} \cdot z^{M-N}$$

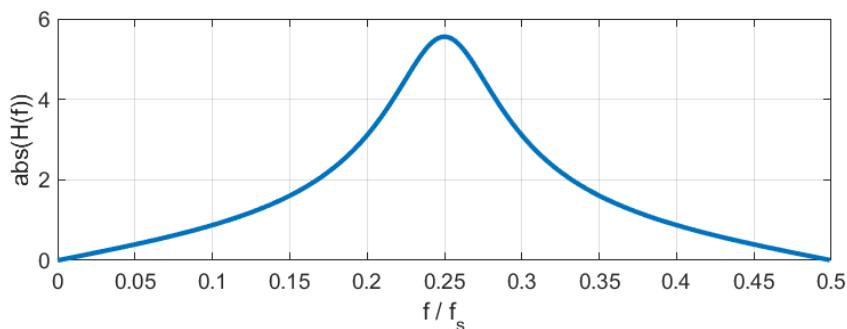
# Beispiel LTD-System

## PN-Darstellung von $H(z)$



wenn ein x und ein o genau auf dem gleichen punkt landen streichen sie sich weg

## Frequenzgang $H(f) = H(z = e^{j2\pi \cdot f \cdot T_s})$



## UTF $H(z)$ wenn $K = 1$

$$H(z) = \frac{(z - 1) \cdot (z + 1)}{(z - j \cdot 0.8) \cdot (z + j \cdot 0.8)}$$
$$H(z) = \frac{z^2 - 1}{z^2 + 0.64} = \frac{1 - z^{-2}}{1 + 0.64 \cdot z^{-2}}$$

1, 0, -1  
b werte  
a werte  
 $1 * z^0, 0 * z^{-1}, 0.64 * z^{-2}$

## Differenzengleichung

$$y[n] = x[n] - x[n-2] - 0.64 \cdot y[n-2]$$

```
b = [1 0 -1]; a = [1 0 0.64];
% zplane(b,a)
[H,w]=freqz(b,a); % w=normierte Frequenz
subplot(2,1,1);
plot(0.5*w/pi, abs(H), 'LineWidth', 2.0);
grid;
xlabel('f / f_s'); ylabel('abs(H(f))');
```

## **filter()**

- Berechnung des Ausgangssignals eines LTD-Systems
- Beispiel: **y = filter(b,a,x)**  
wobei  $b = [b_0 \ b_1 \ \dots]$  und  $a = [a_0 \ a_1 \ \dots]$  die Filterkoeffizienten und  $x = [x[0] \ x[1] \ \dots \ x[N-1]]$  und  $y = [y[0] \ y[1] \ \dots \ y[N-1]]$  das Ein- und das Ausgangssignal darstellen

## **freqz()**

- Berechnung des Frequenzgangs eines LTD-Systems
- Beispiel: **[H,f] = freqz(b,a,N\_f,fs)**  
wobei der Vektor H die komplexen Werte des Frequenzgangs an  $N_f$  äquidistanten Frequenzwerten im Intervall  $[0,fs/2)$  im Vektor f enthält (fs stellt die Abtastfrequenz dar)

## Digital Filter Analysis

Magnitude response, phase response, impulse response, pole-zero plot, group delay

### Functions

angle	Phase angle
fdatool	Open Filter Design and Analysis Tool
filternorm	2-norm or infinity-norm of digital filter
filtord	Filter order
firttype	Type of linear phase FIR filter
freqz	Frequency response of digital filter
fvtool	Open Filter Visualization Tool
grpdelay	Average filter delay (group delay)
impz	Impulse response of digital filter
impzlength	Impulse response length
info	Information about digital filter
phasedelay	Phase delay of digital filter
phasez	Phase response of digital filter
residuez	z-transform partial-fraction expansion
stepz	Step response of digital filter
zerophase	Zero-phase response of digital filter
zplane	Zero-pole plot

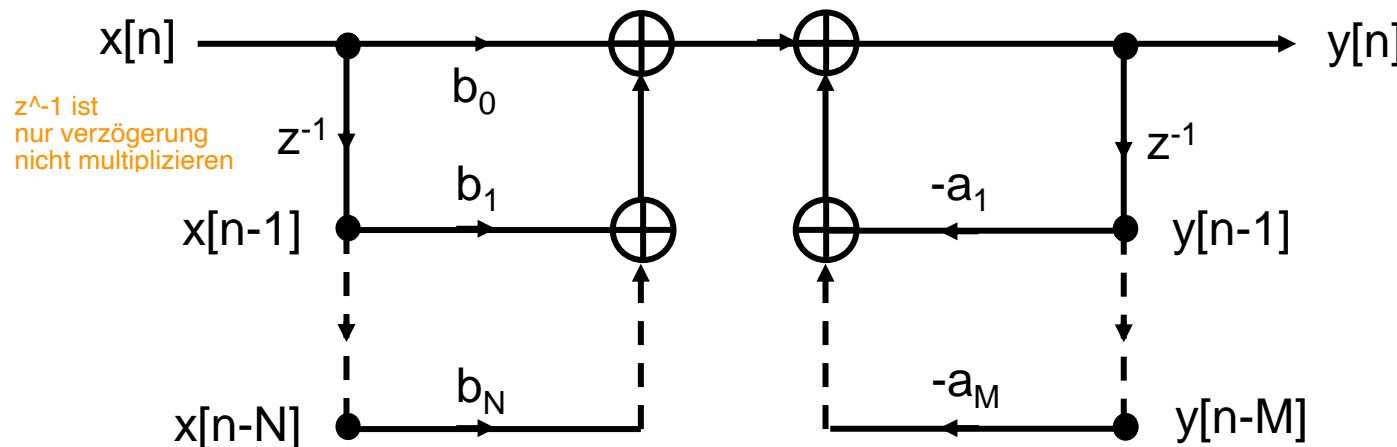
## Kapitel 3.2: Realisierungsstrukturen

Im Idealfall sind die folgenden Realisierungsstrukturen alle gleichwertig.

Im Realfall mit endlich genauen Filterkoeffizienten bzw. beschränkten Wortlängen verhalten sie sich aber unterschiedlich.

Direktstruktur 1: Direkte Umsetzung der Differenzengleichung

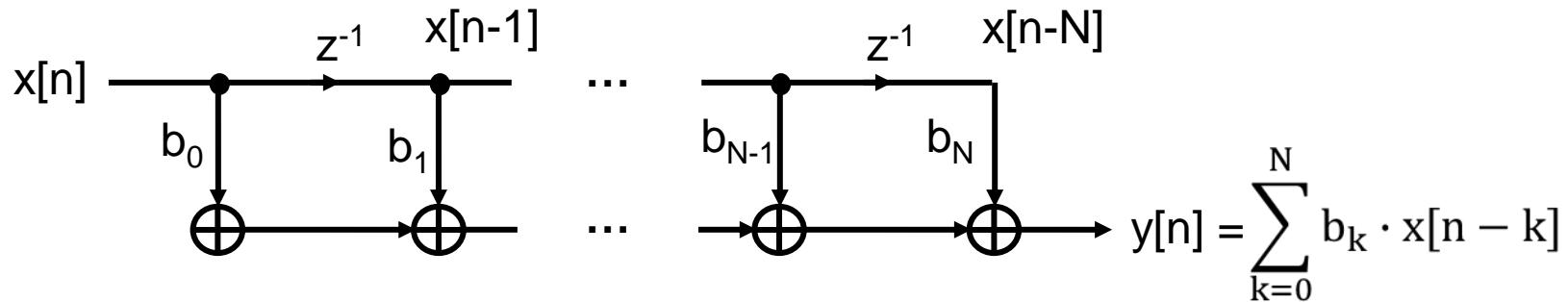
$$y[n] = \sum_{k=0}^N b_k \cdot x[n - k] - \sum_{k=1}^M a_k \cdot y[n - k]$$



FIR-Filter werden normalerweise in Direktstruktur 1 realisiert, IIR aber nie!

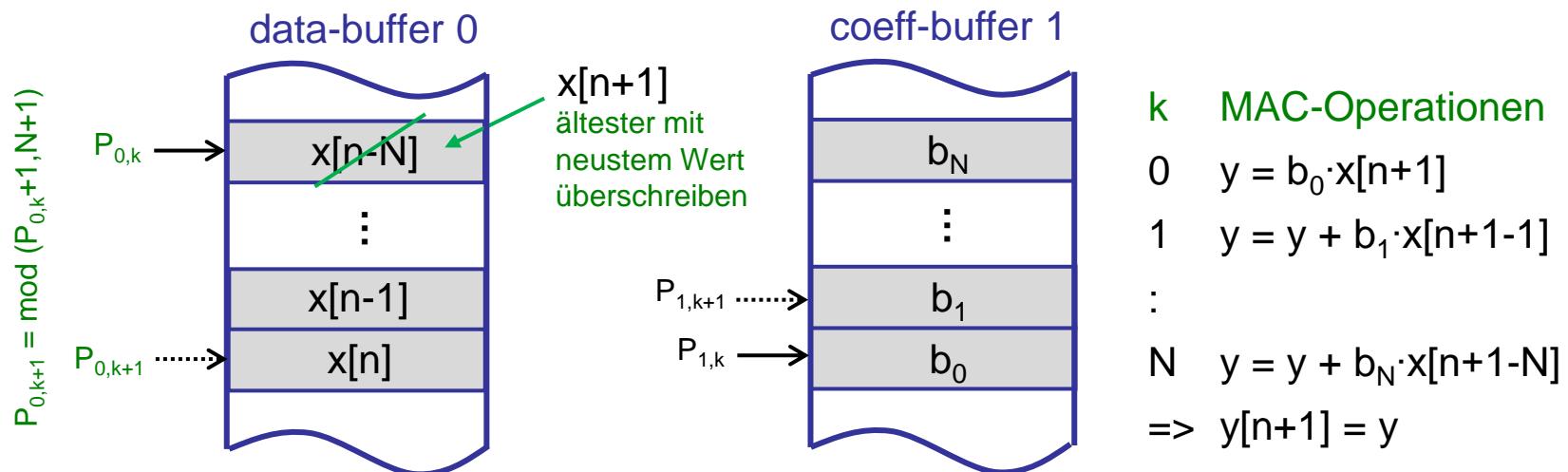
# FIR-Filter N-ter Ordnung in Direktstruktur 1

N Multiply-and-ACcumulate- (MAC-) Operationen pro Output  $y[n]$



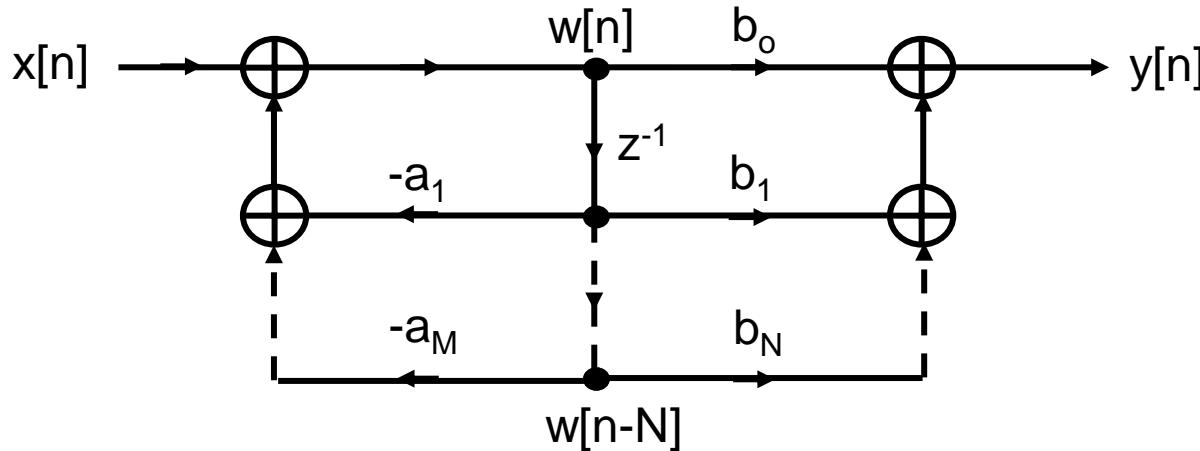
«schnelle» DSP-Unit rechnet 1 MAC-Operation ( $y = y + b \cdot x$ ) pro Clock-Zyklus

Implementierung mit (Ring-) Buffern & modulo-Pointer-Inkrementierung



# Direktstruktur 2

Reihenfolge der Teil-UTF in der Direktstruktur 1 kann vertauscht werden  
=> Direktstruktur 2 mit nur noch 1 Verzögerungskette



**Schritt 0:** (zirkulärer) w-Buffer  $\{w[n-1], \dots, w[n-N]\}$  mit Nullen initialisieren

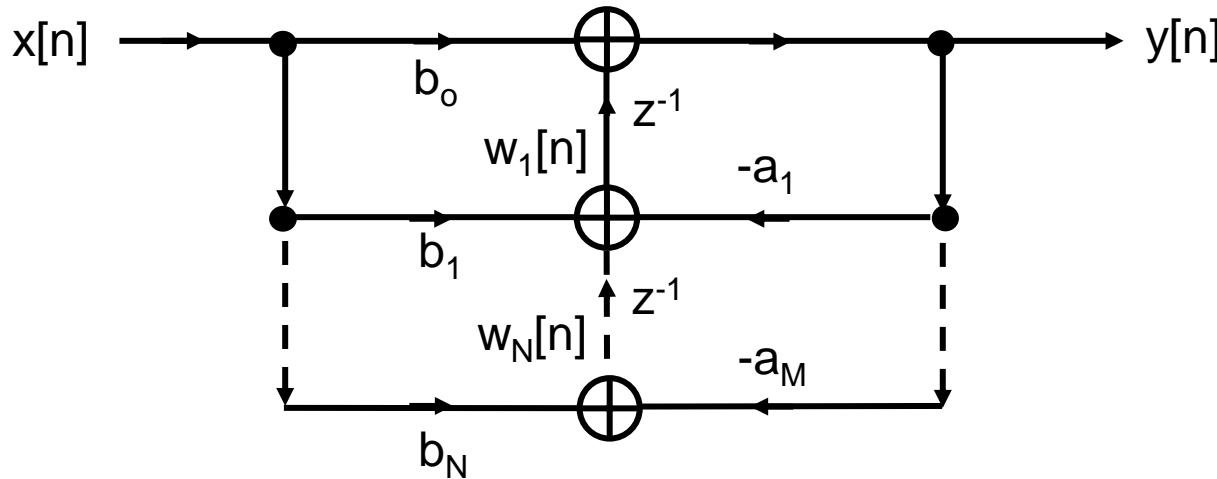
**Schritt 1:**  $w[n] = x[n] - a_1 \cdot w[n-1] - \dots - a_M \cdot w[n-M]$

**Schritt 2:**  $y[n] = b_0 \cdot w[n] + \dots + b_N \cdot w[n-N]$  ausgeben

**Schritt 3:** w-Buffer schieben,  $w[n]$  speichern  
(besser: im Ringbuffer ältesten Wert  $w[n-N]$  mit  $w[n]$  überschreiben)

**Schritt 4:** neuen Eingangswert  $x[n+1]$  lesen und mit Schritt 1 weiterfahren

# Transponierte Direktstruktur 2



$$y[n] = b_0 \cdot x[n] + w_1[n-1]$$

$$w_1[n] = b_1 \cdot x[n] - a_1 \cdot y[n] + w_2[n-1]$$

$$w_{N-1}[n] = b_{N-1} \cdot x[n] - a_{M-1} \cdot y[n] + w_N[n-1]$$

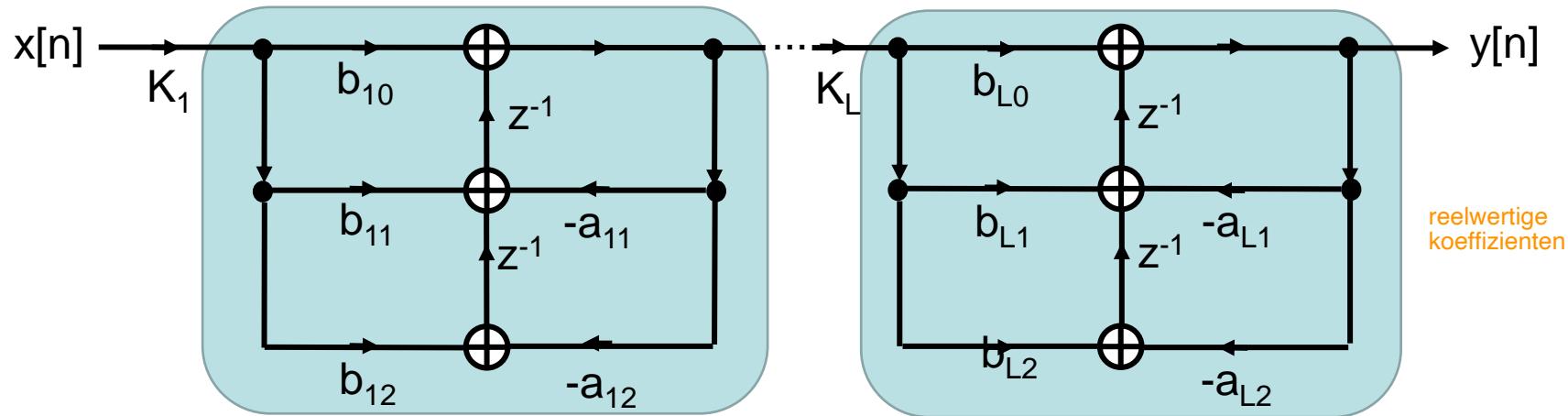
$$w_N[n] = b_N \cdot x[n] - a_M \cdot y[n]$$

# Kaskadierung von Biquads

UTF  $H(z)$  in Second-Order-Sections-Kaskaden-Darstellung (Matlab: `tf2sos`)

$$H(z) = K \cdot \frac{(z - z_1) \cdot (z - z_1^*)}{(z - p_1) \cdot (z - p_1^*)} \cdots \frac{(z - z_L) \cdot (z - z_L^*)}{(z - p_L) \cdot (z - p_L^*)} = K \cdot H_1(z) \cdots H_L(z)$$

$*$  = konjugiert komplex



Pol-Nullstellenpaarung (Normalfall):

sollte für jeden IIR so gemacht werden:  
Grund: Wenn eine Komponente nicht genau umgesetzt werden kann dann beeinflusst er nur 2 Komponentenpaare anstatt alle wie in Direktstruktur 1

- a) letzter Biquad enthält komplexes Polpaar am nächsten beim Einheitskreis und dazu nächstgelegenes konjugiert komplexes Nullstellenpaar.
- b) übrig gebliebenen Pole und Nullstellen werden nach Regel a) kombiniert.

IIR-Filter sollten «immer» in einer Kaskade von Biquads je in Direktstruktur 2 (transponiert) realisiert werden. Die Gefahr ist dann am kleinsten, dass das IIR-Filter instabil wird, weil Pole nahe am Einheitskreis durch Quantisierungseffekte ausserhalb des Einheitskreises zu liegen kommen.

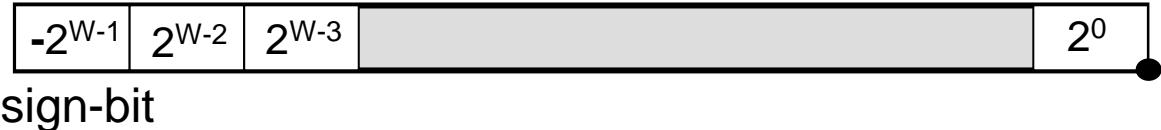
# Zahlendarstellung

## Festkomma-Zahlendarstellung (Zweierkomplement)

$b_{W-1}$

$b_0$

signed integer



unsigned integer



signed fractional



unsigned fractional



## Gleitkomma (IEEE 754/854)

$$F = (-1)^s \cdot M \cdot 2^{E-127}$$

$s$  Exponent  $1 \leq E \leq 254$

Mantisse  $1 \leq M < 2$



hidden 1

# Festkomma-Zahlendarstellung

**Beispiel:** Worte der Länge W=3 und entsprechende Festkomma-Zahlenwerte

[ $b_2 \ b_1 \ b_0$ ]	signed integer	unsigned integer	signed fractional	unsigned frac.
[ 0 0 0 ]	0	0	0.00	0.000
[ 0 0 1 ]	1	1	0.25	0.125
[ 0 1 0 ]	2	2	0.50	0.250
[ 0 1 1 ]	3	3	0.75	0.375
[ 1 0 0 ]	-4	4	-1.00	0.500
[ 1 0 1 ]	-3	5	-0.75	0.625
[ 1 1 0 ]	-2	6	-0.50	0.750
[ 1 1 1 ]	-1	7	-0.25	0.875

Binäre Zahl  $b$  kann negiert werden, indem alle Bits von  $b$  invertiert und 1 LSB addiert wird.

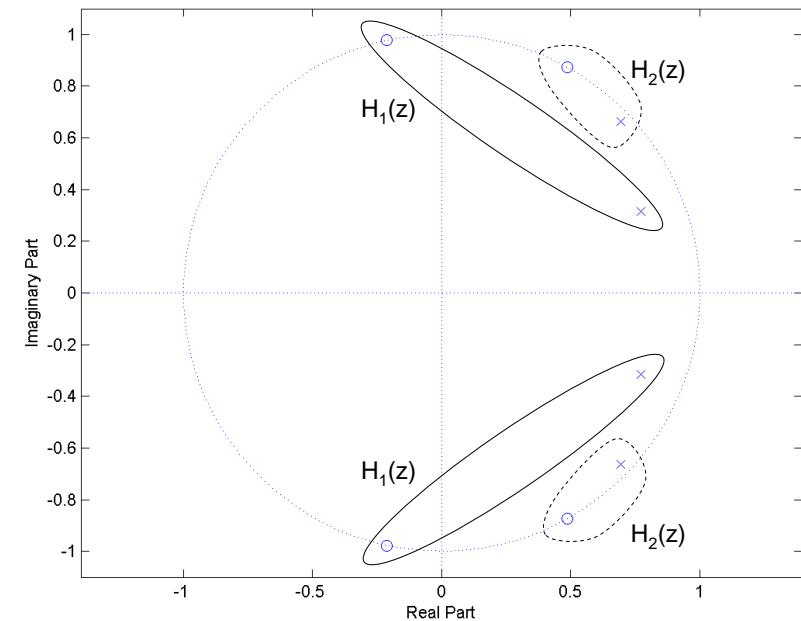
Verhinderung eines Überlauf vom grössten zum kleinsten Wert wird mit Hilfe von sättigender Arithmetik (kein Standard-C => Compiler Intrinsics oder Assembler)

# Festkomma-Filter (Beispiel)

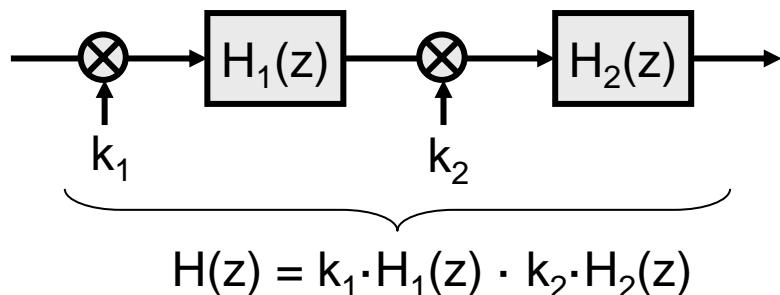
## Filterspezifikation

Filterart	(elliptisches) TP
Filterordnung	N=4
Abtastfrequenz	$f_s = 8000$ Hz
Eckfrequenz DB	$f_{DB} = 1000$ Hz
Eckfrequenz SB	$f_{SB} = 1300$ Hz
max. Rippel im DB	$R_p = 3$ dB
min. Rippel im SB	$R_s = 40$ dB
Wortbreite	W=8 Bit

## PN-Darstellung UTF



## Biquad-Kaskade

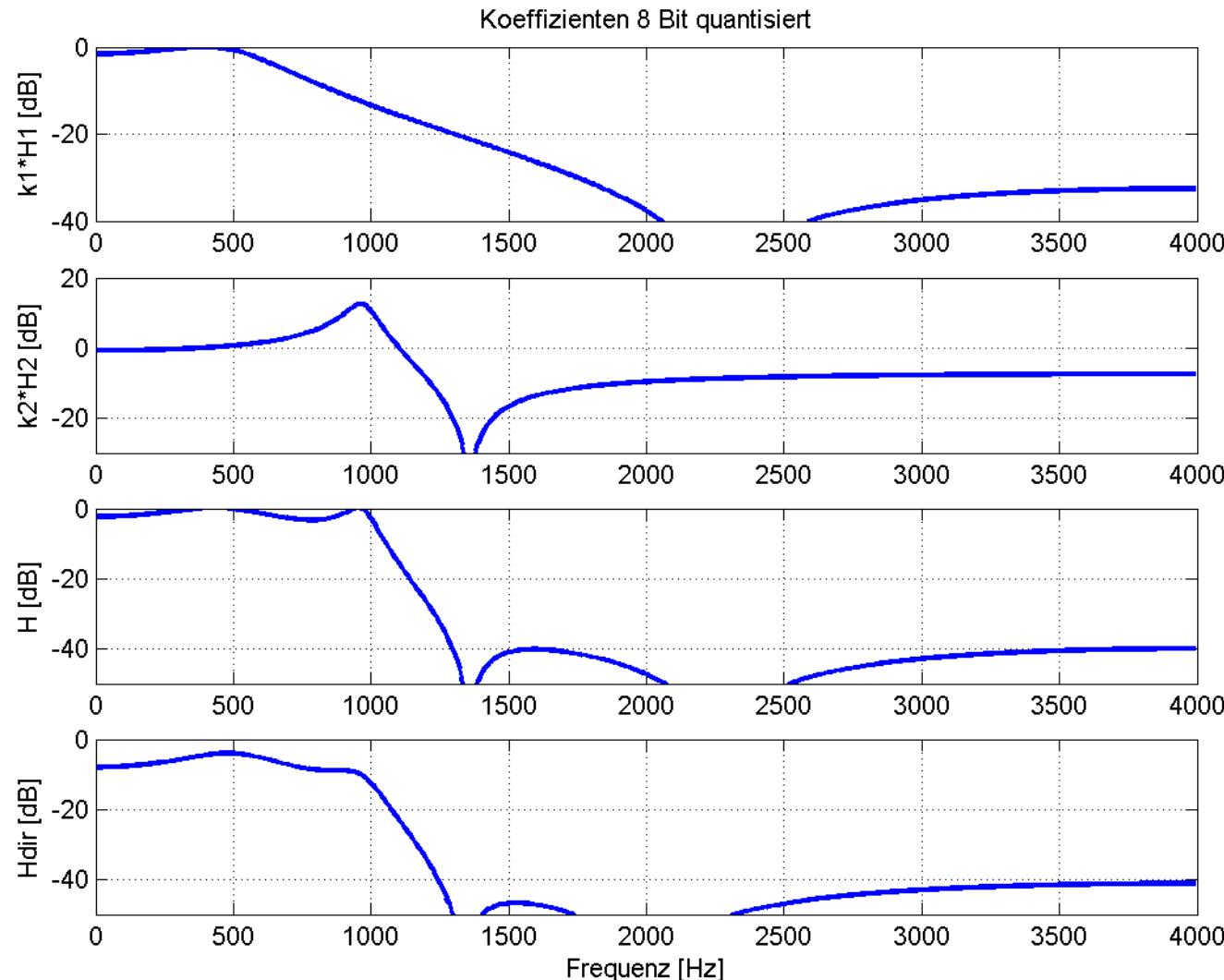


## Skalierung

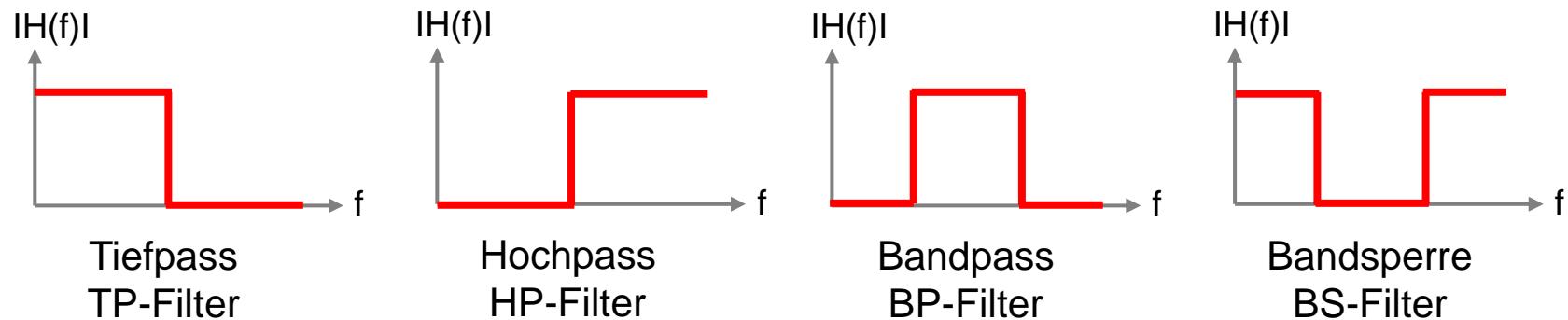
$$k_1 \cdot \max(|H_1(f)|) < 1$$

$$k_1 \cdot k_2 \cdot \max(|H_1(f) \cdot H_2(f)|) < 1$$

# Festkomma-Filter (Beispiel)



Wir betrachten in diesem Kapitel zuerst die **Approximation** der folgenden nicht-realisierten **idealen (brick wall) Filter**.



Diese Filter werden im Frequenzbereich spezifiziert.

Die Digitalfilter werden eingeteilt in zwei grossen Klassen:

- nichtrekursive bzw. **FIR-Filter** (FIR: Finite Impulse Response)
- rekursive bzw. **IIR-Filter** (IIR: Infinite Impulse Response)

## FIR-Filter sind nichtrekursive LTD-Systeme

werden meistens in Transversalstruktur (Direktform 1) realisiert

- + linearer Phasengang ist realisierbar (keine Signalverzerrungen im Durchlassbereich)
- + sind immer stabil (alle Pole im Ursprung) => instabil wenn Pol ausserhalb einheitskreises wäre
- + sind toleranter gegenüber Quantisierungseffekten als IIR-Filter
- weisen viel höhere Filterordnungen als vergleichbare IIR-Filter auf
- die Zeitverzögerung bzw. Gruppenlaufzeit ist relativ gross

## IIR-Filter sind rekursive LTD-Systeme

werden meistens als Biquad-Kaskade realisiert

- + kleine Filterordnung (Aufwand) dank Pol-Selektivität
- + kleine Zeitverzögerung
- linearer Phasengang für kausale Filter nicht realisierbar
- mehr Probleme mit Quantisierungseffekten als bei FIR-Filter

# Review Frequenzgang des LTD-Systems

## H(z): Übertragungsfunktion (UTF)

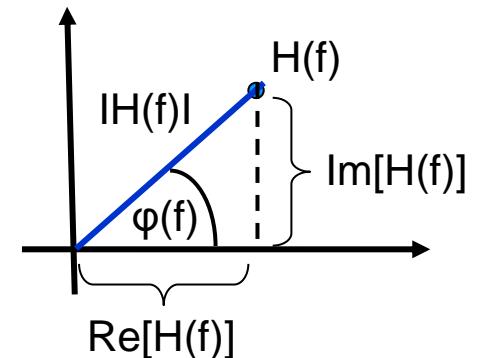
z-Transformierte der Impulsantwort  $h[n]$

## H(f): Frequenzgang

$$H(f) = H(z=e^{j2\pi f T_s})$$

Fourier-Transformierte von  $h[n]$

Polarcoordinatendarstellung =>



$$H(f) = |H(f)| e^{j\varphi(f)}$$

## |H(f)|: Amplitudengang

meistens in dB, d.h.  $20 \cdot \log_{10}(|H(f)|)$

gerade Funktion, d.h.  $|H(f)| = |H(-f)|$

$$H^*(-f) = |H(-f)| e^{-j\varphi(-f)}$$

wenn  $h[n]$  reell:

$$H(f) = H^*(-f)$$

## phi(f): Phasengang

ungerade Funktion, d.h.  $\varphi(f) = -\varphi(-f)$

$$\varphi(f) = \arctan(\operatorname{Im}[H(f)] / \operatorname{Re}[H(f)])$$

## Bedeutung des Amplituden- und Phasengangs

$$\cos(2\pi f_0 \cdot nT_s) \rightarrow \boxed{H(f)} \rightarrow IH(f_0)| \cdot \cos[2\pi f_0 \cdot nT_s + \varphi(f_0)] \\ = IH(f_0)| \cdot \cos[2\pi f_0 \cdot (nT_s - \tau_p)]$$

$t_p$  = Zeitverzögerung

Eine einzelne Frequenzkomponente bei  $f_0$  wird

- in der Amplitude mit  $|H(f_0)|$  gewichtet und erleidet eine
- Phasenverschiebung  $\varphi(f_0)$  bzw. Zeitverzögerung  $\tau_P(f_0) = \frac{-\varphi(f_0)}{2\pi f_0}$

**Linearer Phasengang**  $\varphi(f) = -K \cdot f$  wobei K eine Konstante ist

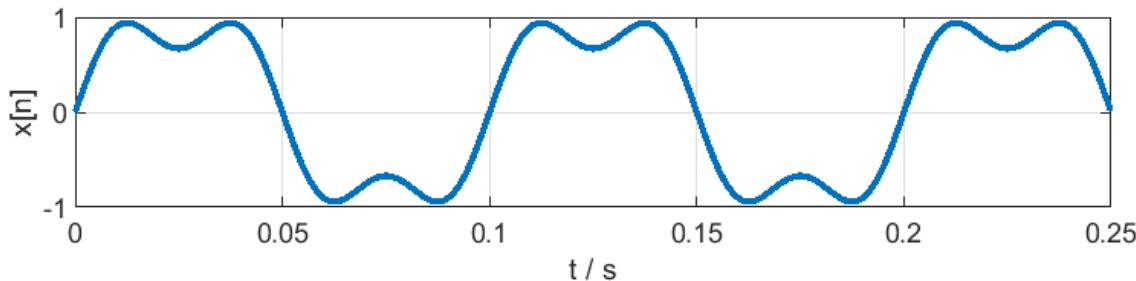
Zeitverzögerung ist für alle Frequenzkomponenten gleich

bzw. die Gruppenlaufzeit  $\tau_G(f_0) = \frac{-\partial \varphi(f_0)}{2\pi \partial f_0}$  ist konstant

d.h. keine Verzerrungen der Signalform im Durchlassbereich!

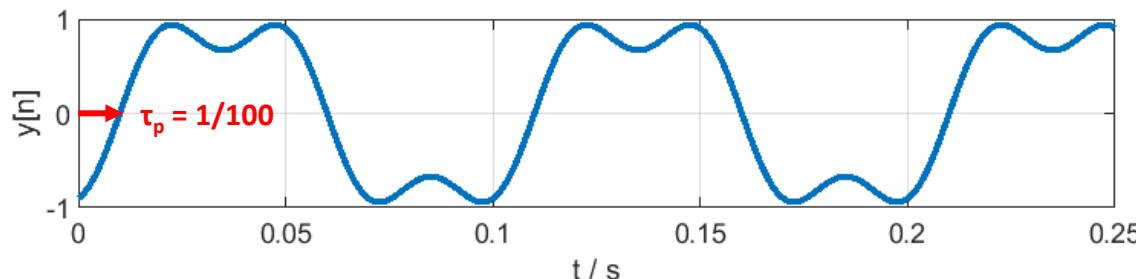
# Beispiel (nicht-) linearer Phasengang

Symmetrisches, periodisches Rechtecksignal: 10 Hz nur 1. und 3. Harmonische



$$x(t) = \sin(2\pi \cdot f_0 \cdot t) + \frac{1}{3} \cdot \sin(2\pi \cdot 3f_0 \cdot t)$$

Filterung von  $x(t)$  mit linear-phasigem Allpass:  $|H(f)| = 1$ ,  $\phi_H(f) = -K \cdot f$ ,  $K = \pi/50 \text{ rad} \cdot \text{s}$

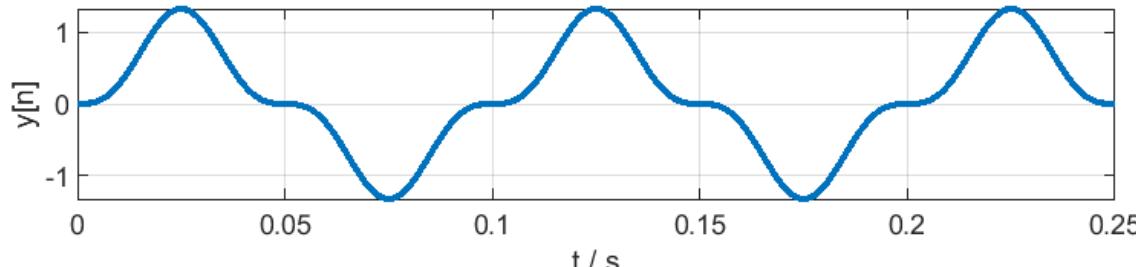


$$y(t) = 1 \cdot \sin(2\pi \cdot f_0 \cdot t - \frac{\pi}{5}) + \frac{1}{3} \cdot \sin(2\pi \cdot 3f_0 \cdot t - \frac{3\pi}{5})$$

$\Rightarrow$  Zeitverzögerung  $\tau_p = 10 \text{ ms}$

Keine Verzerrung! :D

Filterung von  $x(t)$  mit nicht-linear-phasigem Allpass:  $|H(f)| = 1$ ,  $\phi_H(10) = 0$ ,  $\phi_H(30) = -\pi$

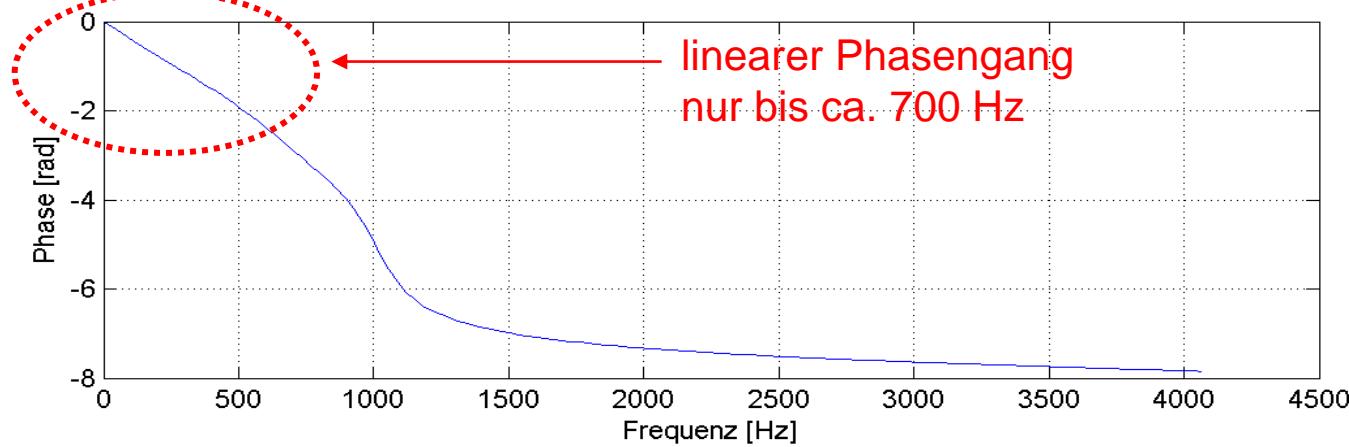
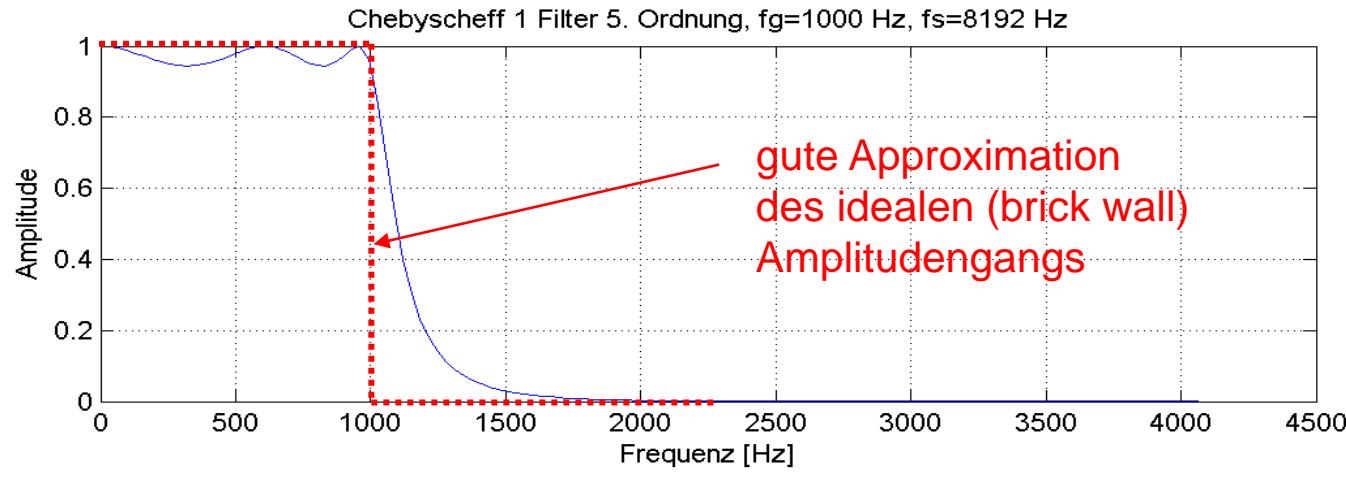


$$y(t) = 1 \cdot \sin(2\pi \cdot f_0 \cdot t - 0) + \frac{1}{3} \cdot \sin(2\pi \cdot 3f_0 \cdot t - \pi)$$

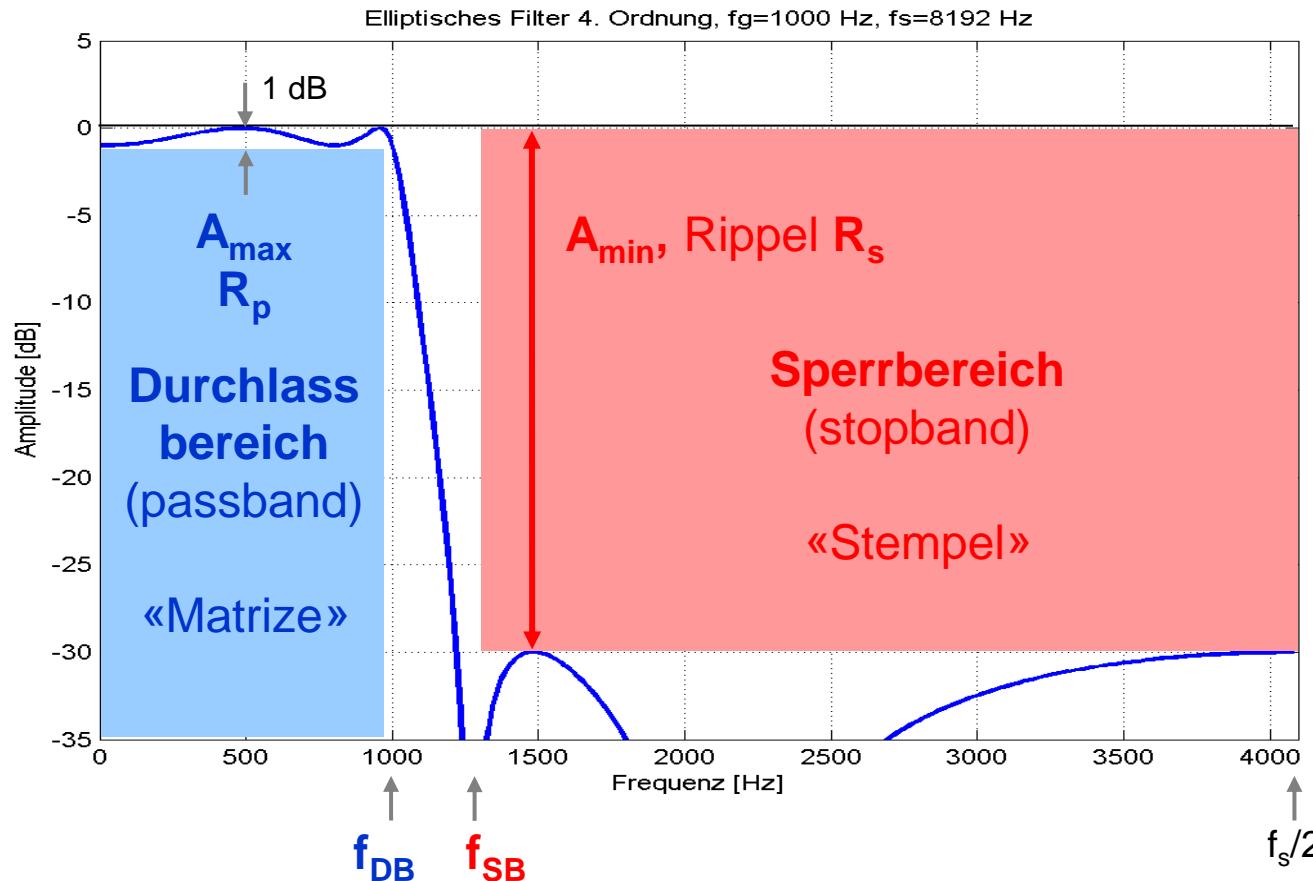
$\Rightarrow$  Signalverzerrung !

Amplitude bleibt gleich, Phase verschiebt sich

## Typischer Amplituden- und Phasengang eines IIR-Filters



Der Amplitudengang wird oft mit einem Stempel-Matrizen-Schema spezifiziert.



In diesem Beispiel:

$$f_{DB} = 1 \text{ kHz}$$

$$f_{SB} = 1.3 \text{ kHz}$$

$$A_{\max} = R_p = 1 \text{ dB}$$

$$A_{\min} = R_s = 30 \text{ dB}$$

Die Filterordnung bzw. der Rechenaufwand hängt von der Filtersteilheit im Übergangsbereich ab.

- Filterordnung und Realisierungsaufwand hängen primär von der Steilheit der Filterflanke zwischen Durchlass- und Sperrbereich ab.

Vergleiche z.B. 4. mit 6. Ordnung mit Matlab filterDesigner

- Bandpässe bzw. Bandsperren werden entsprechend spezifiziert, wobei der Durchlass- bzw. der Sperrbereich mit 2 Eckfrequenzen am unteren und oberen Rand spezifiziert werden muss.
- Der Phasengang wird meistens durch den Grad der Linearität (im Durchlassbereich) und/oder durch die maximale Zeitverzögerung spezifiziert.
- Wenn die Linearität des Phasengangs wichtig ist und die Zeitverzögerung nicht allzu klein sein muss, wählt man normalerweise ein (linearphasiges) FIR-Filter.

## FIR-Filter der Ordnung N haben einen linearen Phasengang

wenn die Filterkoeffizienten **symmetrisch** oder **antisymmetrisch** sind, d.h. wenn  $b_n = b_{N-n}$  oder  $b_n = -b_{N-n}$  für  $n = 0 \dots N$ . Die Zeitverzögerung beträgt dann für alle Frequenzkomponenten  $\tau_G = (N/2) \cdot T_s$ .

## 4 Typen linearphasiger FIR-Filter und Restriktionen im H(f)

Typ	Symmetrie	Ordnung N	H(0)	H(f <sub>s</sub> /2)	Filter realisierbar
1	sym.	gerade	-	-	alle
2	sym.	ungerade	-	Nullstelle	TP, BP
3	anti-sym.	gerade	Nullstelle	Nullstelle	BP
4	anti-sym.	ungerade	Nullstelle	-	HP, BP

**Beispiel:** FIR-Filter  $H(z) = b_0 \cdot (1+z^{-1})$

Die Ordnung N=1 und die Filterkoeffizienten sind symmetrisch => Typ 2.

Der Frequenzgang  $H(f) = 2b_0 \cdot \cos(\pi f T_s) \cdot e^{-j\pi f T_s}$  hat eine Nullstelle bei  $f = f_s/2$ .

Der Phasengang  $\phi(f) = -\pi \cdot f \cdot T_s$  ist linear bzw. die Zeitverzögerung  $\tau_G = T_s/2$ .

$$\sin x = \frac{e^{ix} - e^{-ix}}{2i} \quad \cos x = \frac{e^{ix} + e^{-ix}}{2}$$

Der obere Teil des Bruches wird rein multipliziert  
der untere wird umgewandelt zu - da  $1/\dots$  das gleiche ist.

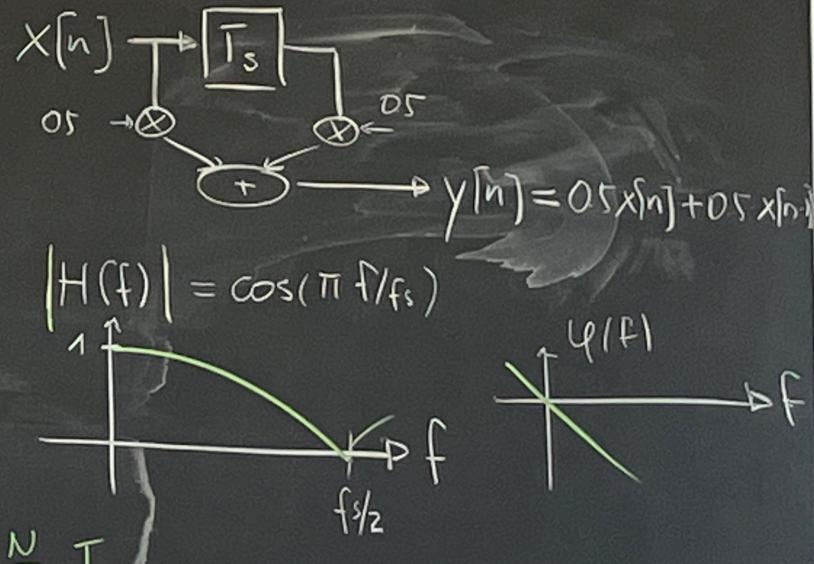
Bsp:  $h[n] = \{0.5, 0.5\}$

$$\begin{aligned} H(z) &= 0.5 + 0.5 z^{-1} \\ H(f) &= H(z=e^{j2\pi f T_s}) \\ &= 0.5 \cdot (1 + e^{-j2\pi f/f_s}) \cdot \frac{e^{j\pi f/f_s}}{e^{-j\pi f/f_s}} \\ &= 0.5 \cdot \left( \frac{e^{j\pi f/f_s} + e^{-j\pi f/f_s}}{2} \right) \cdot e^{-j\pi f/f_s} \\ &= \left| \cos(\pi f/f_s) \right| e^{-j\pi f/f_s} \end{aligned}$$

erweitern mit 2

$$\varphi(f) = -\pi f/f_s$$

$$\tau(f) = \frac{-\varphi(f)}{2\pi f} = \frac{\pi f T_s}{2\pi f} = \frac{T_s}{2} - \frac{N}{2} \cdot T_s$$



**Ziel:**  $b_n = h[n]$  so bestimmen, dass  $H(f)$  die Spezifikationen erfüllt

## Fenstermethode

1. Analoge Referenzstossantwort abtasten:  $h_d[n] = T_s \cdot h(t=nT_s)$

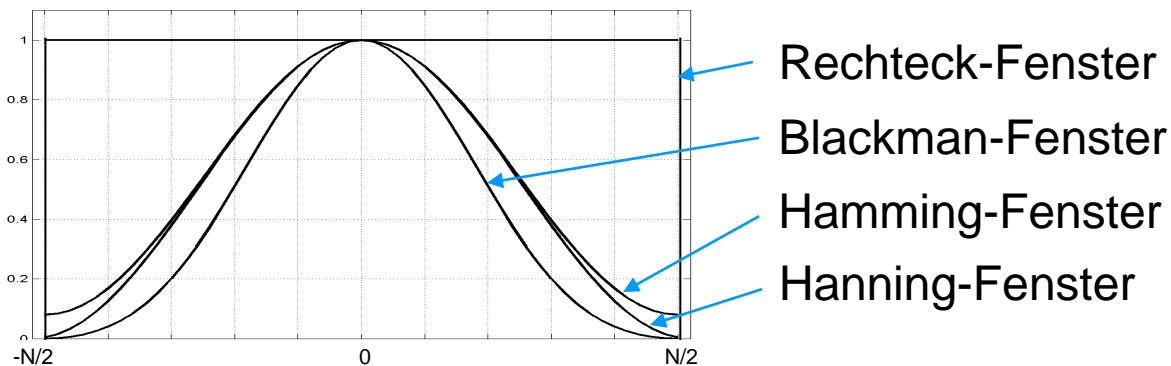
z.B. idealer TP:

$$h_d[n] = \frac{\sin(n\pi f_{DB}/(f_s/2))}{n\pi} \quad -\infty < n < \infty$$

die Zahlen von - bis + berechnen für die H Komponenten

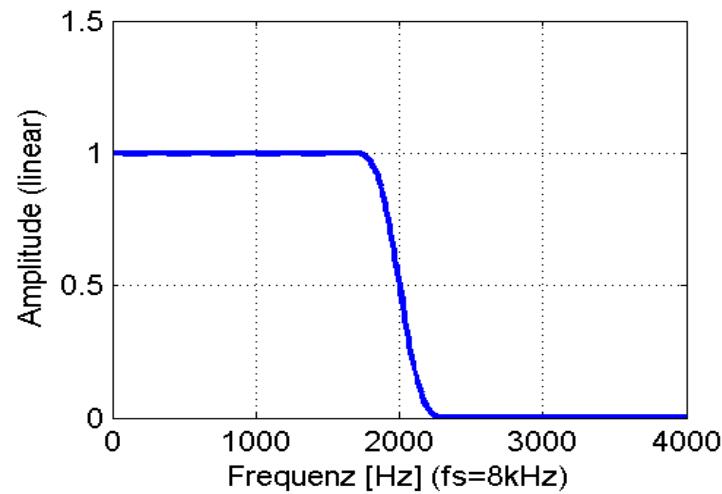
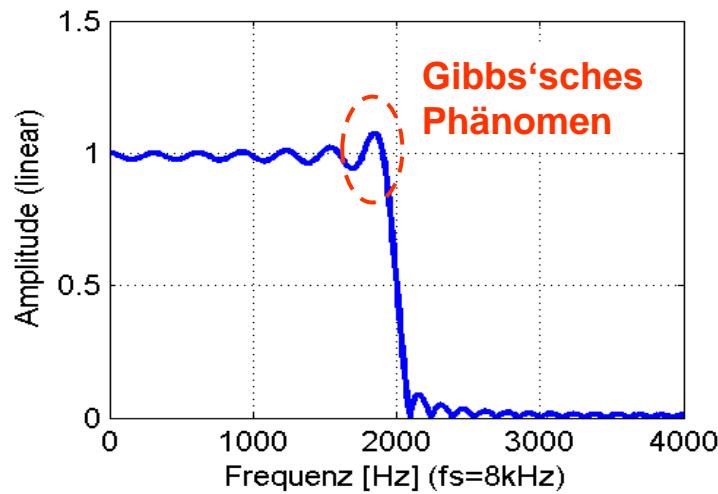
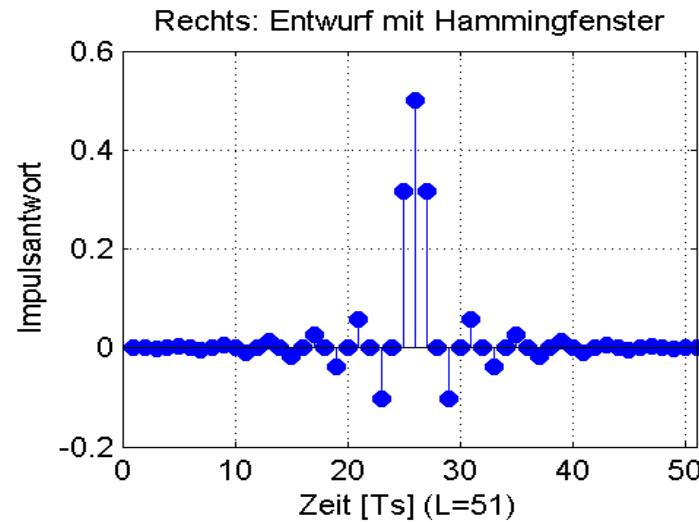
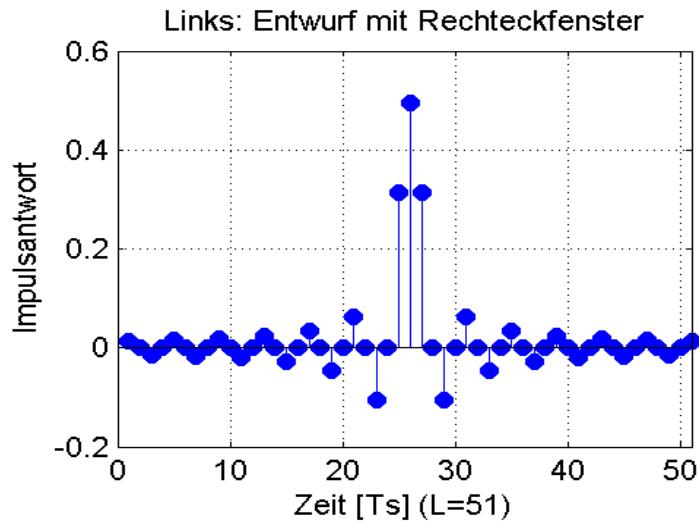
2. relevanten Anteil ausschneiden:  $h_c[n] = w[n] \cdot h_d[n]$  für  $-N/2 \leq n \leq N/2$

Fenster  $w[n]$ :



3. FIR-Filter mit Zeitverschiebung kausal machen:  $h[n] = h_c[n-N/2]$

# Beispiel zum Windowing

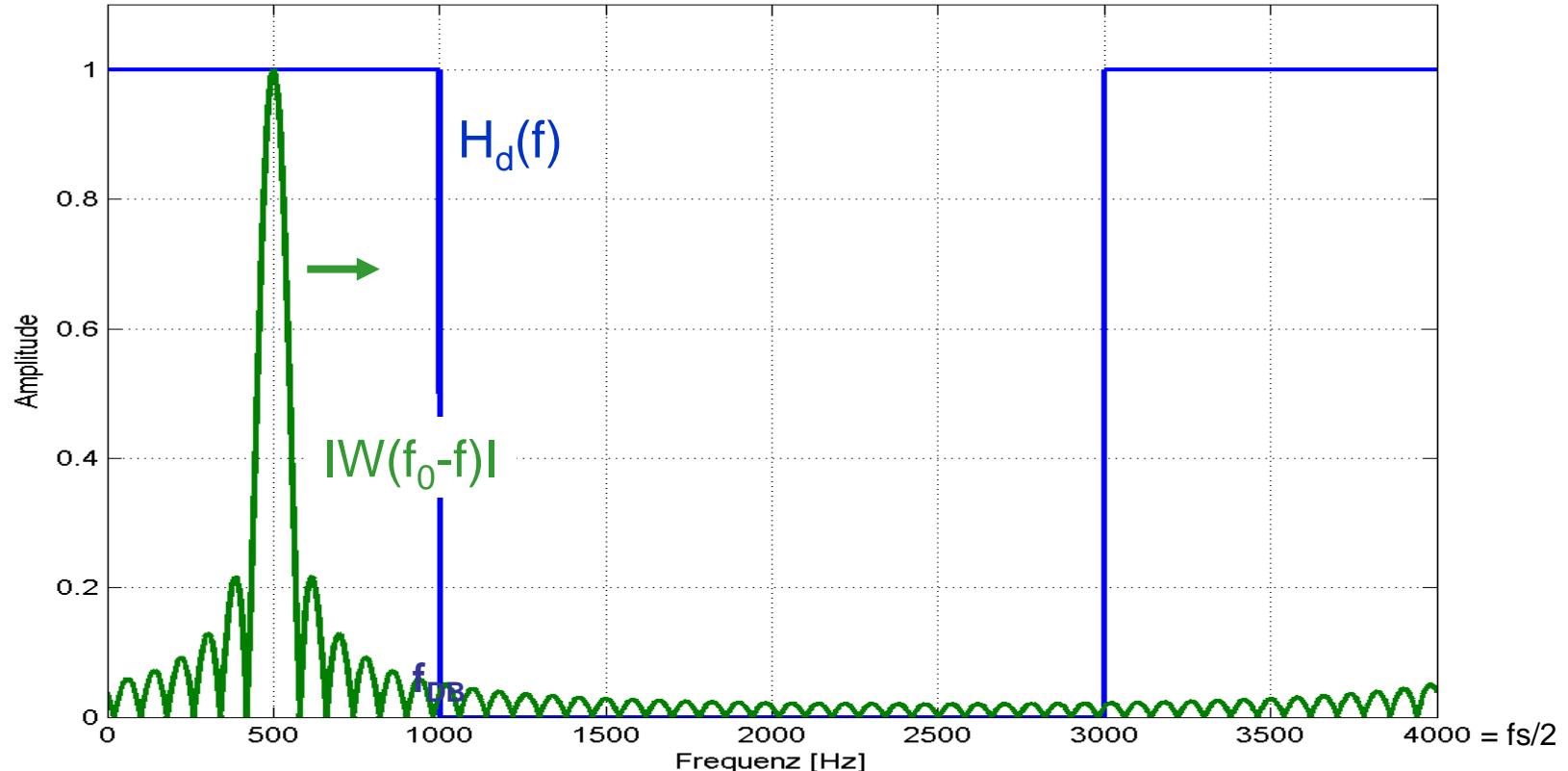


Matlab:  $N=50$ ;  $fs=8e3$ ;  $fDB=2e3$ ;  $b=fir1(N, fDB/(fs/2))$ ;  $freqz(b, 1, 1000, fs)$   
oder filterDesigner

fir1 = matlab befehl für Fir design mit Hamming Windowing methode  
!! auf  $fs/2$  bezogen

# Einfluss des Fensters

$$h_{\text{FIR}}[n] = w[n] \cdot h_d[n] \quad \bullet \bullet \quad H_{\text{FIR}}(f) = W(f) * H_d(f)$$

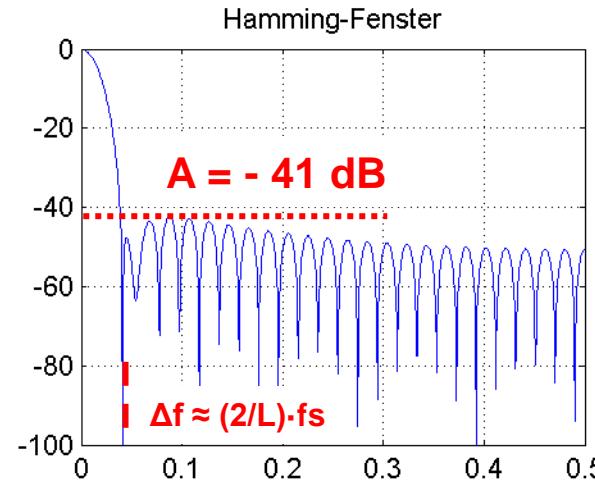
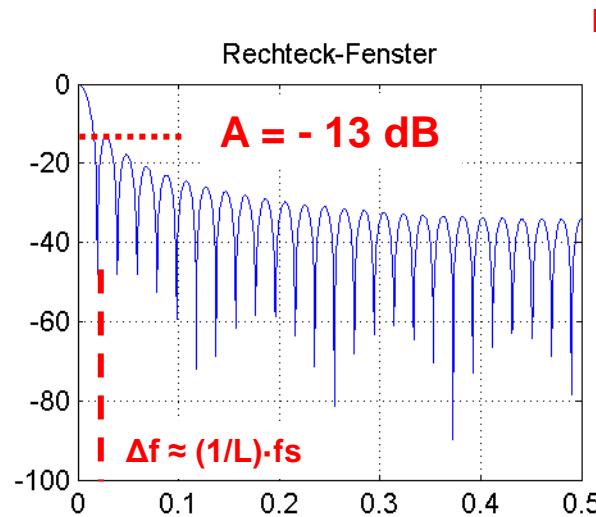


Gibbs'sches Phänomen: Überschwingen von  $H_{\text{FIR}}(f_0 \approx f_{\text{DB}})$

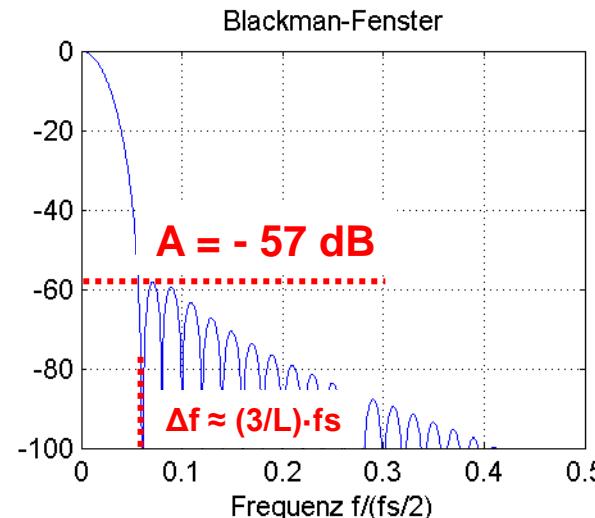
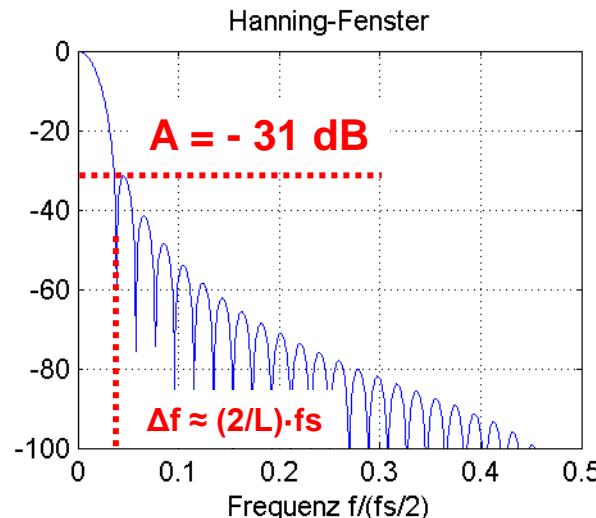
Nebenkeule von  $W(f)$  klein => Überschwingen von  $H_{\text{FIR}}(f)$  klein

Hauptkeule von  $W(f)$  schmal => Übergangsbereich von  $H_{\text{FIR}}(f)$  steil

# Spektren verschiedener Fenster



Je kleiner die Nebenkeulen desto grösser die Sperrdämpfung (siehe auch Folie 11).



Je schmäler Δf desto steiler die Filterflanke (siehe auch Folie 11).

# TP – BP/BS/HP-Transformationen

**Ziel:** Erhalt der linearen Phase

## TP-BP-Frequenzverschiebung

$$\text{Typ 1,2: } b_{BP}[n] = 2 \cdot \cos(\omega_0 \cdot n T_s) \cdot b_{TP}[n]$$

$$\text{Typ 3,4: } b_{BP}[n] = 2 \cdot \sin(\omega_0 \cdot n T_s) \cdot b_{TP}[n]$$

Verschiebung des Tiefpassfilters um  $\omega_0$  ->  
wird somit zum Bandpass

## TP-HP-Frequenzverschiebung

$$\text{TP-BP-Trafo mit } f_0 = f_s/2: \quad b_{HP}[n] = (-1)^n \cdot b_{TP}[n]$$

alternierende Multiplikation mit 1 und -1

## BP-BS-Transformation

Bandpass zu Bandsperre

$$\text{BS ist Komplementärfilter zu BP: } H_{BP}(z) + H_{BS}(z) = z^{-N/2}$$

wäre 1 aber mit  
Zeitverzögerung  $z^{-(N/2)}$

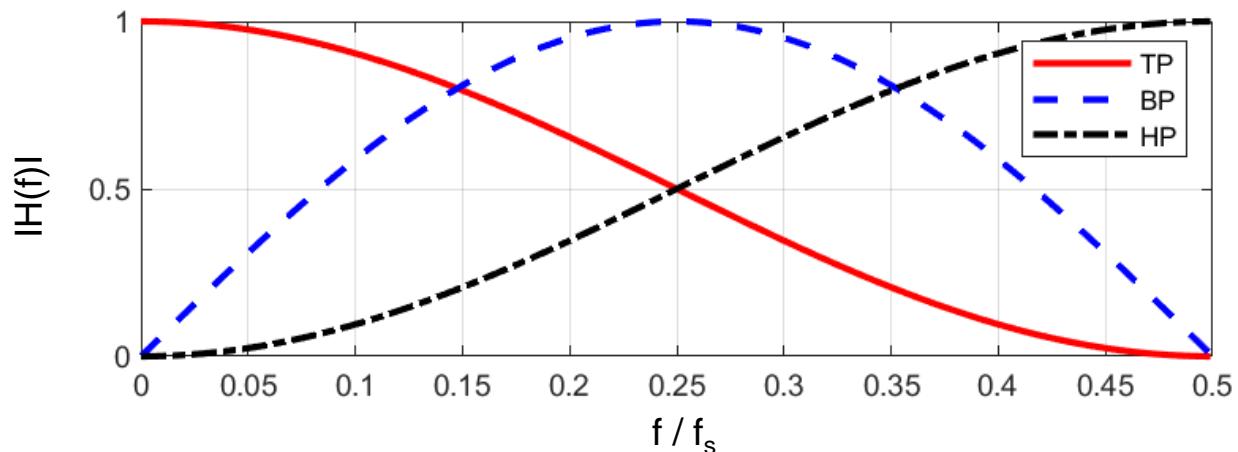
$$b_{BS}[n] = \delta[n-N/2] - b_{BP}[n]$$

**Matlab:** linearphasiges FIR-HP-Filter 10. Ordnung mit  $f_{DB} = f_s/6$

```
N=10; b=fir1(N,1/3,'high'); freqz(b,1,1000,12000)
```

## Beispiel

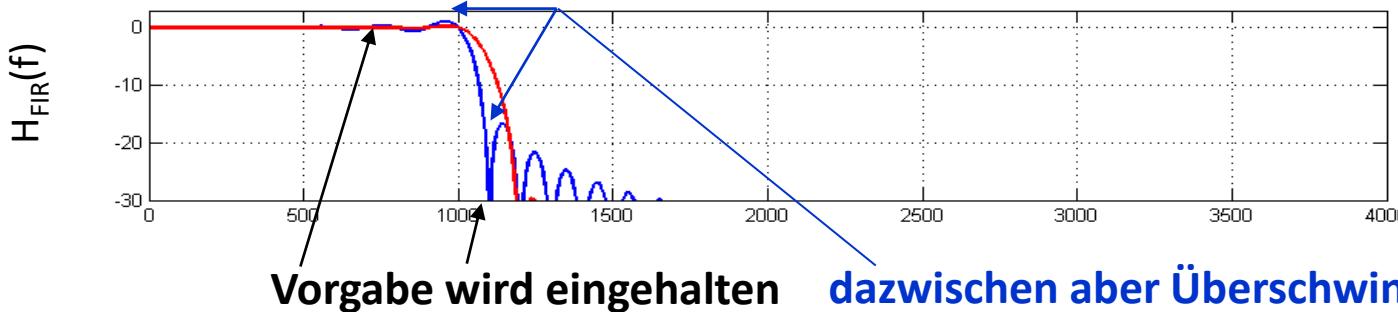
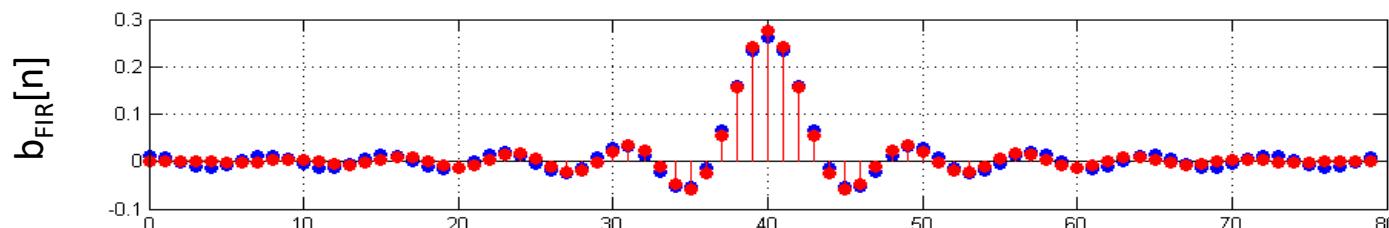
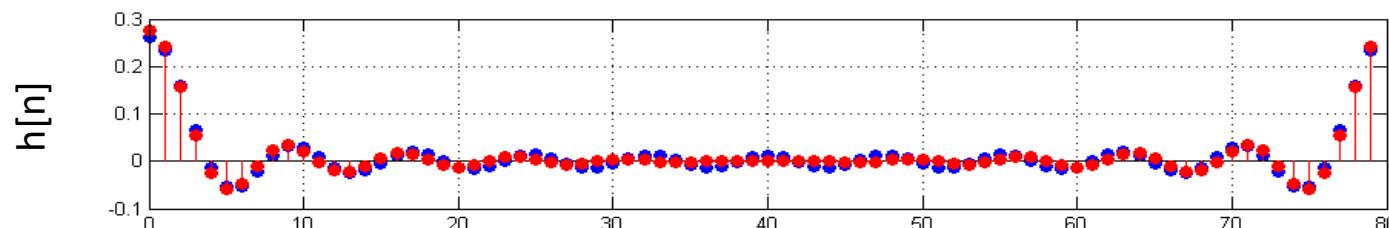
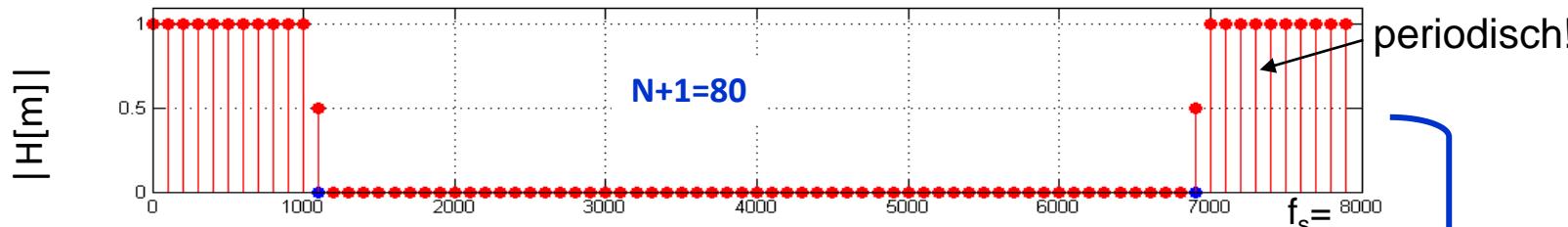
- gegeben: **FIR-TP Typ 1**  $b_{TP} = [ 0.25 \ 0.5 \ 0.25 ]$   
 $\Rightarrow$  Ordnung  $N=2$ , sym. Koeffizienten bzw. linearer Phasengang
- gesucht: **FIR-BP** mit Mittenfrequenz  $f_0=f_s/4$ :  $\omega_0 = 2\pi \cdot f_0$   
 $\Rightarrow b_{BP}[n] = 2 \cdot \cos(n \cdot \pi/2) \cdot b_{TP}[n] \Rightarrow b_{BP} = [ 0.5 \ 0 \ -0.5 ]$
- gesucht FIR-HP:  
 $\Rightarrow b_{HP}[n] = (-1)^n \cdot b_{TP}[n] \Rightarrow b_{HP} = [ 0.25 \ -0.5 \ 0.25 ]$
- Resultat:



- Bei dieser Entwurfsmethode werden **N+1** äquidistante Abtastpunkte des gewünschten **Frequenzgangs H(f)** vorgegeben.
- Mit Hilfe der **inversen diskreten Fourier-Transformation IDFT** bzw. der IFFT kann dann die **Impulsantwort h[n]**, n,=0, ..., N, des FIR-Filters bestimmt werden.
- Der **Frequenzgang** des FIR-Filters hält die **Vorgabepunkte** natürlich genau ein, dazwischen kann es aber „**Überschwinger**“ geben, vor allem im Übergang zwischen dem Durchlass- und dem Sperrbereich.
- Mit dem Frequenzabtastverfahren können „**beliebige**“ Formen des Frequenzganges vorgegeben werden.

# FIR-Filterentwurf: Frequenzabtastung

1. Vorgabe  $N+1$  äquidistante Abtastwerte von  $H(f)$  im Bereich  $[0, f_s]$



2. IFFT

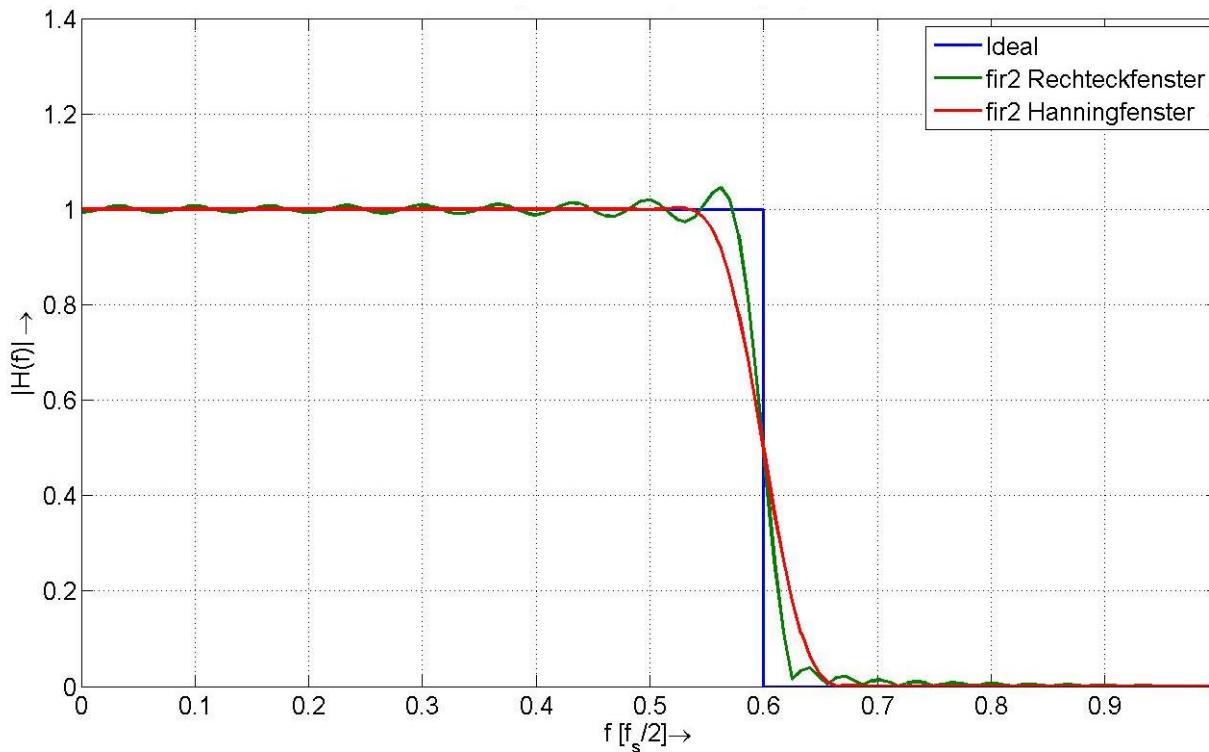
3. Zeit-  
verschiebung

Verbesserungen:  
Vorgabe weniger  
steil (siehe ● )  
oder Windowing

fir2() FIR arbitrary shape filter design using the frequency sampling method

Beispiel: linearphasiges FIR-TP-Filter (mit & ohne Fenster)

```
f = [0 0.6 0.6 1]; m = [1 1 0 0]; N=60;  
b = fir2(N,f,m,hanning(N+1)); oder  
b = fir2(N,f,m,rectwin(N+1));
```



computer aided design

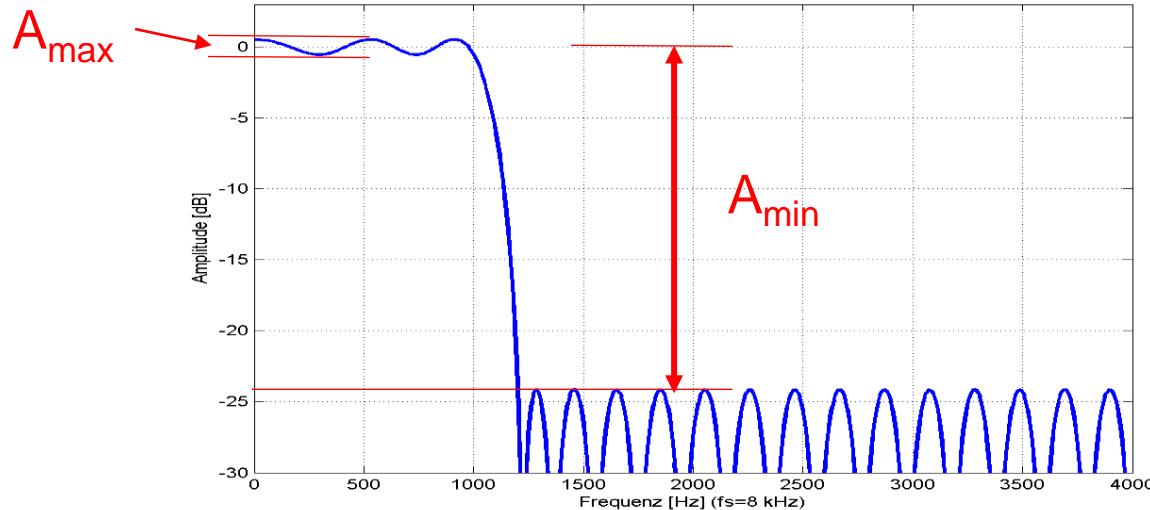
## Iterative Optimierungsverfahren (CAD)

- am bekanntesten: Remez-Algorithmus (Parks-McClellan): `firpm()`

Vorgabe Stempel-Matrise (auch Multiband) => Minimax-Optimierung

maximale abweichung wird minimiert

**Equiripple** im Durchlass- und Sperrbereich => kleinste Ordnung für  $A_{\min}$  &  $A_{\max}$   
welligkeit im durchlass und sperrbereich

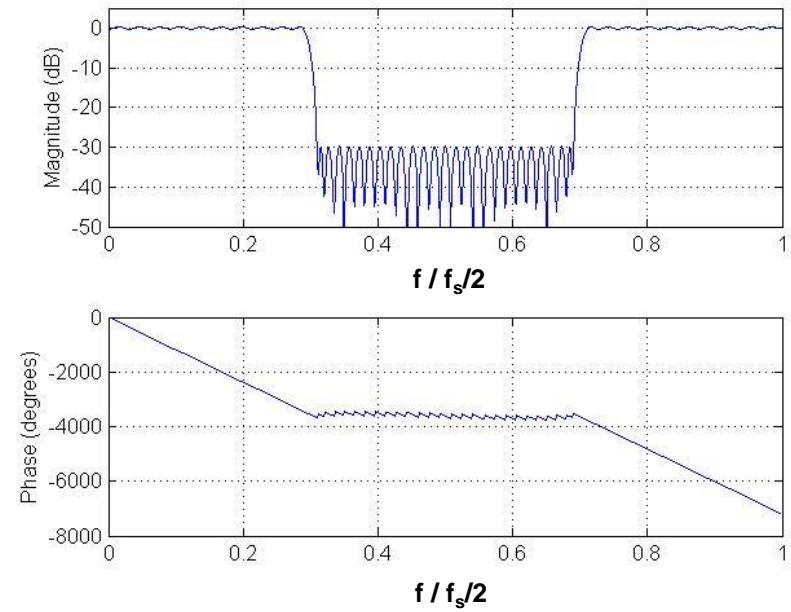
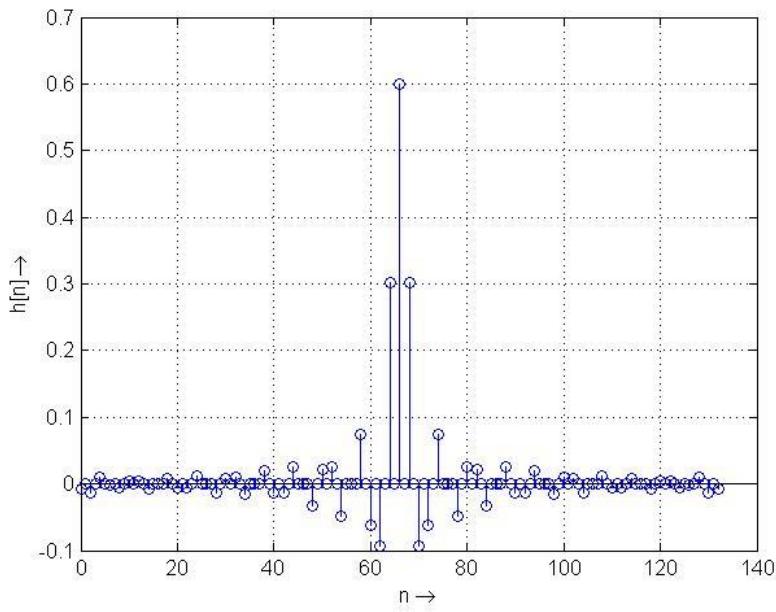


- Least-Square Optimierungsverfahren: `firls()`

`firpm()`: Parks-McClellan optimal equiripple linear-phase FIR filter design

Beispiel: linearphasiges FIR-BS-Filter der Ordnung N=132

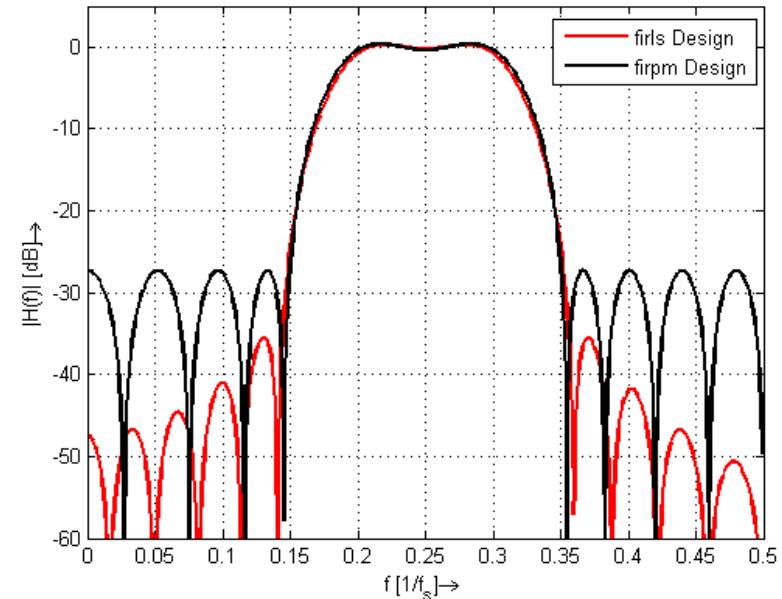
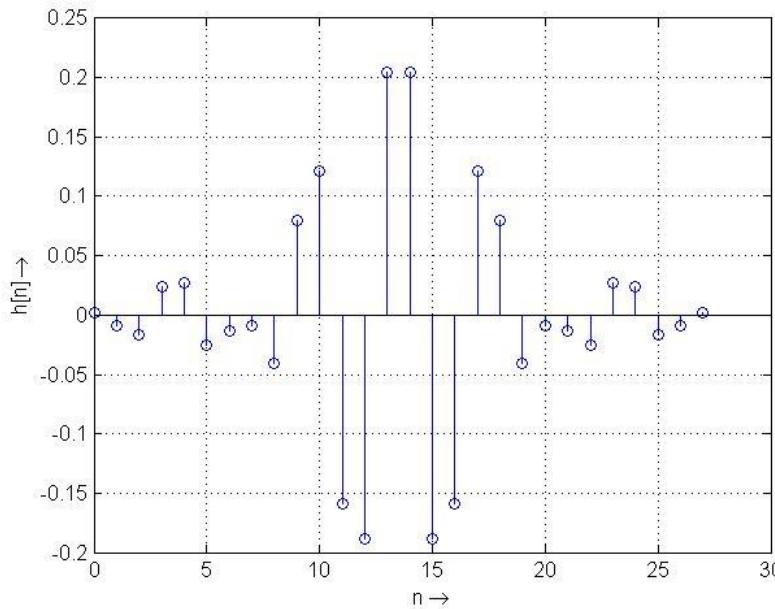
```
F=[0, 0.29, 0.31, 0.69, 0.71, 1]; A=[1, 1, 0, 0, 1 1];  
N=132; b = firpm(N, F, A);
```



`firls()`: Linear-phase FIR filter design using least-squares error minimization

Beispiel: linearphasiges FIR-BP-Filter der Ordnung N=27

```
Freq=[0, 0.3, 0.4, 0.6, 0.7, 1]; Amp=[0, 0, 1, 1, 0, 0];  
N=27; b=firls(N,Freq,Amp);
```



- Die **Approximation** von idealen «Brick-Wall»-Filtern ist im Analogen für verschiedene **Optimierungskriterien** durchgeführt worden.
- Die **Optimierungskriterien** sind z.B.:
  - a) möglichst kleiner Aufwand (**kleine Ordnung**)
  - b) möglichst konstante Gruppenlaufzeit (**absolut konstant nur FIR-Filter**)
  - c) möglichst **monotoner Verlauf** (in Durchlassbereich)
- Im **Digitalen** greifen wir auf diese **analogen Filter** zurück.
- Designschritte:
  - a) Analoger Prototyp-Tiefpass
  - b) eventuell BP,BS,HP-Transformation im Analogen
  - c) **Transformation ins digitale**  
(meist **bilinear** oder seltener impuls-invariant)
- Design-Tools führen diese Schritte direkt oder in Teilschritten aus.

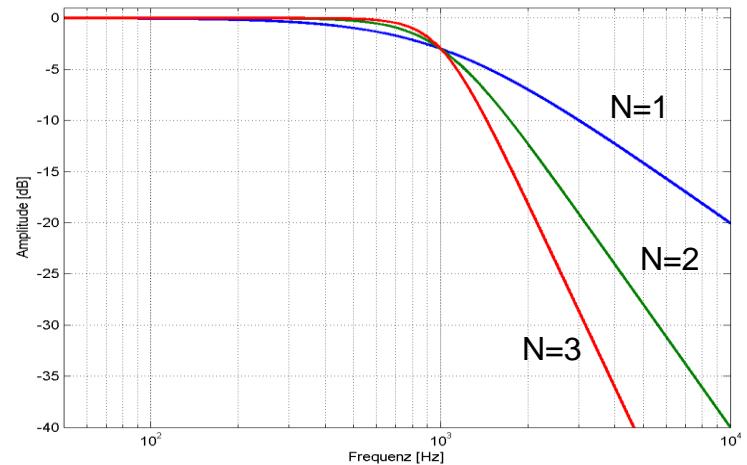
## Approximation von ‘Brickwall‘-Filtern ist im Analogen gelöst

Beispiel: Butterworth-TP N. Ordnung

$H(s)$  mit Amplitudengang

$$|H(f)| = \frac{1}{\sqrt{1 + (f/f_{DB})^{2N}}}$$

$f_{DB}$  = grenzfrequenz , N = filterordnung



## IIR-Filterentwurf

sz-Trafo  
(bilinear)

$H_{TP}(s)$  [ =>  $H_{BP}(s)$  ] =>  $H(z)$  => b-,a-Filterkoeffizienten

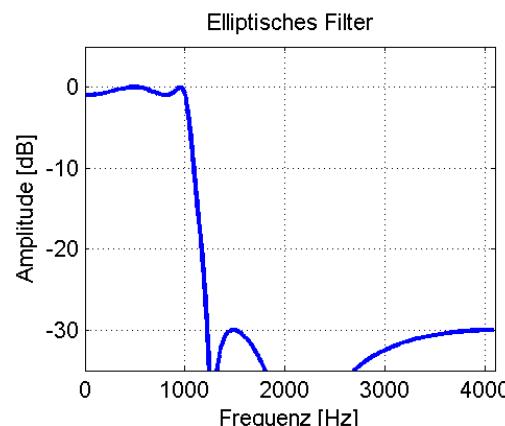
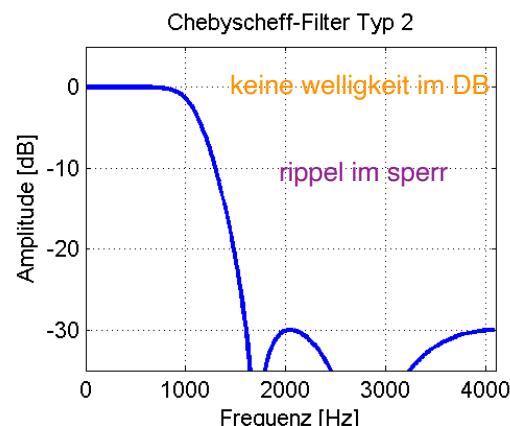
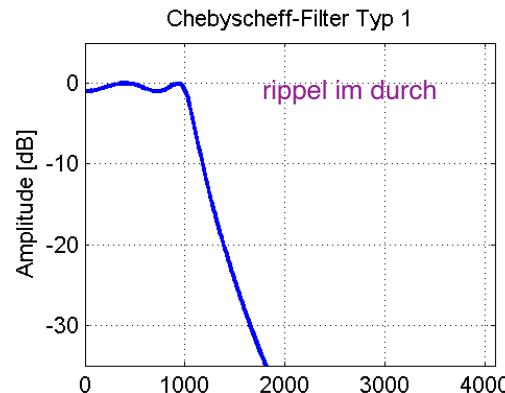
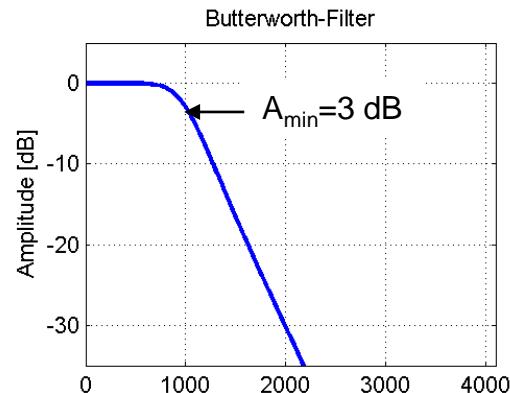
TP-HP/BP/BS-Trafo  
(Achtung: BP und BS  
haben doppelte Ordnung)

# IIR-Filterentwurf mit div. Prototypen

## Vergleich Filter 4. Ordnung ( $f_s = 8192$ Hz)

=> Durchlassbereich:  $A_{\max} = 1$  dB,  $f_{DB} = 1$  kHz

=> Sperrbereich:  $A_{\min} = 30$  dB,  $f_{SB} = 2$  kHz



## Butterworth-Filter

- Steilheit: klein
- $|H(f)|$ : monoton
- $\varphi(f)$ : Nichtlinearität klein

## Chebyscheff-Filter

- Steilheit: mittel
- $|H(f)|$ : Rippel im DB oder SB
- $\varphi(f)$ : Nichtlinearität mittel

durchlassbereich  
sperrbereich

## Elliptisches Filter (Cauer)

- Steilheit: gross
- $|H(f)|$ : Rippel im DB und SB
- $\varphi(f)$ : Nichtlinearität gross

auch equiripplefilter  
genannt

gut für klienste Ordnung

## Besselfilter

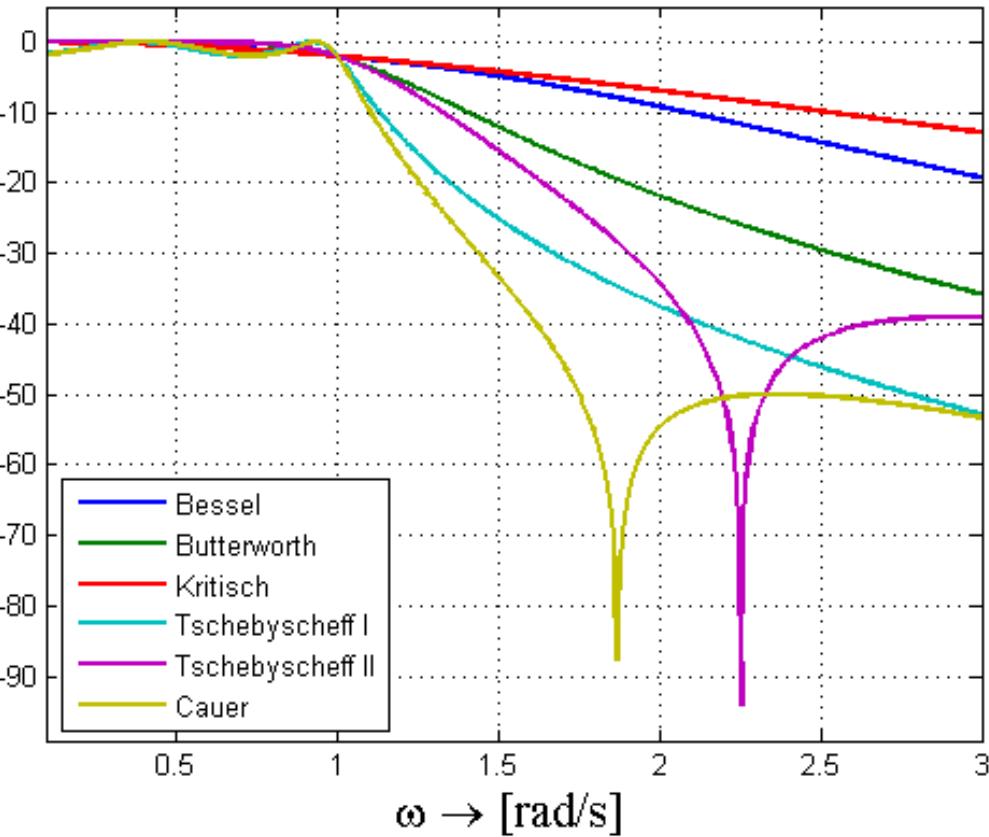
- Steilheit: sehr klein
- $|H(f)|$ : monoton
- $\varphi(f)$ : Nichtlinearität sehr klein

linearität wird vor allem für datenübertragungen gefordert da die bits genau wieder erkannt werden müssen.

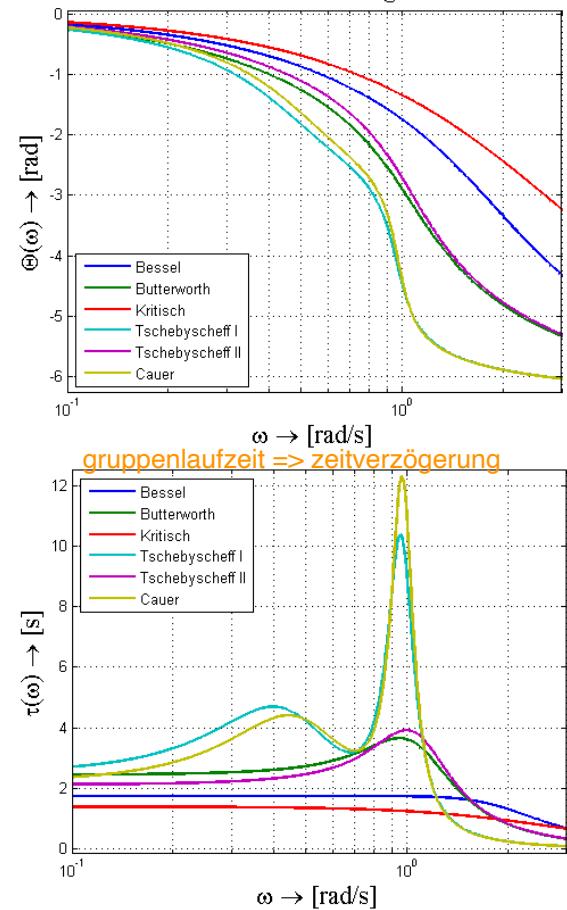
# IIR-Filterentwurf: Analoge Prototypen

Filter 4. Ordnung

$20 \log_{10}(|H(j\omega)|) \rightarrow [\text{dB}]$



Filter 4. Ordnung



	Butterworth-Filter	Chebyscheff Filter, Typ 1	Chebyscheff Filter, Typ 2	Elliptisches Filter	Bessel-Filter
Durchlassbereich	monoton	Rippel	monoton	Rippel	monoton
Sperrbereich	monoton	monoton	Rippel	Rippel	monoton
Steilheit	klein	mittel	mittel	gross	sehr klein
Linearität $\varphi(f)$	gross	mittel	mittel	klein	sehr gross

## Problem der sz-Transformation

- durch Einsetzen von  $z=e^{sT_s}$  in  $H(s)$  entsteht **keine** gebrochen rationale bzw. realisierbare z-UTF  $H(z)$

=> wenn einfach so ersetzt wird erhält man nicht  $H(z) = b\text{-polynom} / a\text{-polynom}$  sondern ein log. => problematisch deshalb folgendes:

**Ziel der bilinearen Transformation** bedeutet nur das Zähler und Nenner linear sind. => das ganze an sich ist jedoch nicht linear

- Approximation der sz-Transformation  $z=e^{sT_s}$  so, dass aus  $H(s)$  eine realisierbare z-UTF  $H(z)$  resultiert, damit die b/a-Filterkoeffizienten bestimmt werden können

## Lösungsansatz

- Approximation mit Taylor-Reihe (1. Ordnung)

dies hat jetzt wieder Eigenschaft b-poly / a-poly

$$z = e^{sT_s} = \frac{e^{sT_s/2}}{e^{-sT_s/2}} \approx \frac{1+sT_s/2}{1-sT_s/2} \quad \text{bzw.}$$

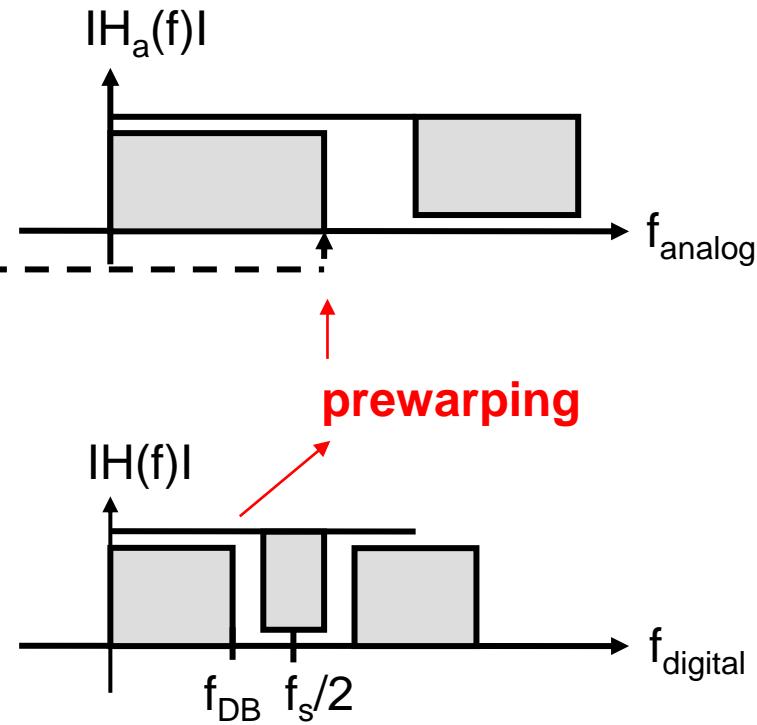
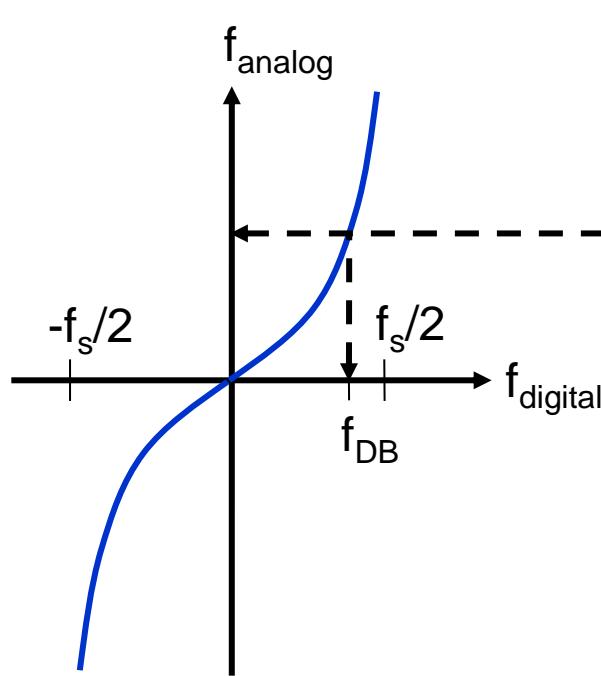
$$s = \frac{2}{T_s} \cdot \frac{z-1}{z+1}$$

b poly  
a poly

# Bilineare Transformation

**sz-Trafo**  $s = \frac{2}{T_s} \cdot \frac{z-1}{z+1}$  einsetzen von  $s=j2\pi f_{\text{analog}}$  und  $z=\exp(j2\pi f_{\text{digital}} T_s)$  gibt:

**f-Trafo:**  $2\pi \cdot f_{\text{analog}} = (2/T_s) \cdot \tan(\pi \cdot f_{\text{digital}} \cdot T_s)$  **kein Aliasing !**  
**aber Frequenzstauchung !**



**RC-Tiefpass 1. Ordnung**  
mit Zeitkonstante  $\tau = 1/(2\pi \cdot f_g)$

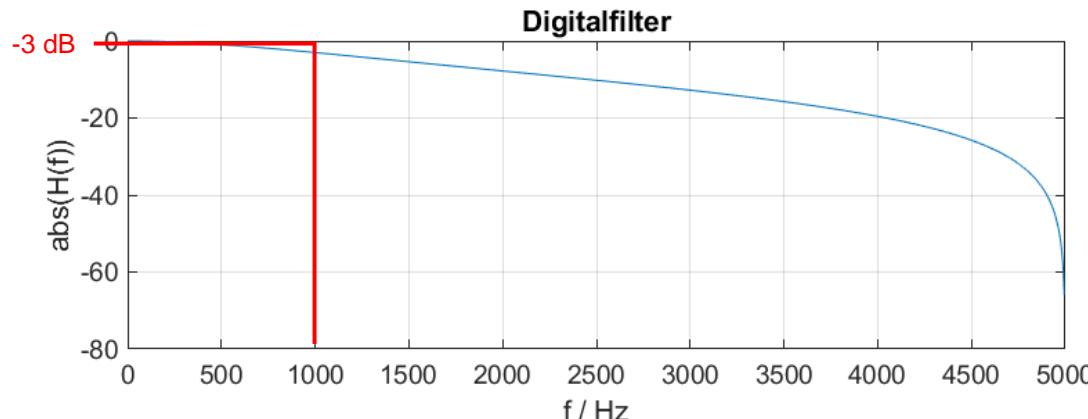
$$H(s) = \frac{1}{1 + s \cdot \tau}$$

↓ bilineare Trafo

**z-UTF IIR-Filter 1. Ordnung**  
mit  $b = [1 \ 1] / (1+c)$   
und  $a = [1 \ (1-c)/(1+c)]$

$$H(z) = \frac{1}{1 + \frac{2\tau}{T_s} \cdot \underbrace{\frac{z-1}{z+1}}_c} = \frac{1}{1 + c} \cdot \frac{1 + z^{-1}}{1 + \frac{1-c}{1+c} \cdot z^{-1}}$$

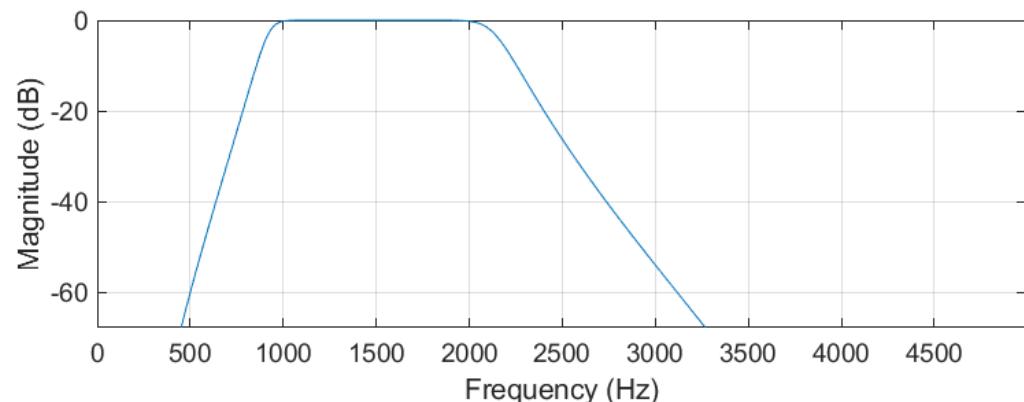
**Prewarping** für  $f_{\text{digital}} = 1 \text{ kHz}$ ,  $f_s = 10 \text{ kHz}$ :  $f_g = (f_s/\pi) \cdot \tan(\pi/10) = 1034.3 \text{ Hz}$   
d.h.  $\tau = 0.154 \text{ ms}$  und  $c = 2\tau/T_s = 3.0777$



# Beispiel Butterworth IIR-BP-Filter

Digitales **Butterworth BP-Filter** soll zwischen 1 kHz und 2 kHz maximal 1 dB durchlassen und bis 500 Hz und ab 4 kHz mit mindestens 60 dB dämpfen. Die Abtastfrequenz  $f_s$  ist 10 kHz. MATLAB-Entwurf:

```
fs=1e4; Amin=60; Amax=1;  
fDB=[1 2]*1e3; fSB=[0.5 4]*1e3; % beides bezogen auf fs/2  
[N,WP]=buttord(fDB/(fs/2),fSB/(fs/2),Amax,Amin);  
N % Resultat ist 7. Ordnung  
[b,a]=butter(N,WP);  
freqz(b,a,1000,fs)
```



Die Entwurfsfunktionen für die Standard-Prototypen lauten:

`ellip()`, `cheby1()`, `cheby2()` und `butter()`

Die Funktionen zur Ermittlung der minimalen Filterordnung lauten:

`ellipord()`, `cheblord()`, `cheb2ord()` und `buttord()`

Beispiel:

- Grenzfrequenzen  $f_{DB}$  und  $f_{SB}$  vorgeben
- Maximale und minimale Dämpfung  $A_{pass}$  und  $A_{stop}$  vorgeben
- Ordnung und normierte Eckfrequenz bestimmen:

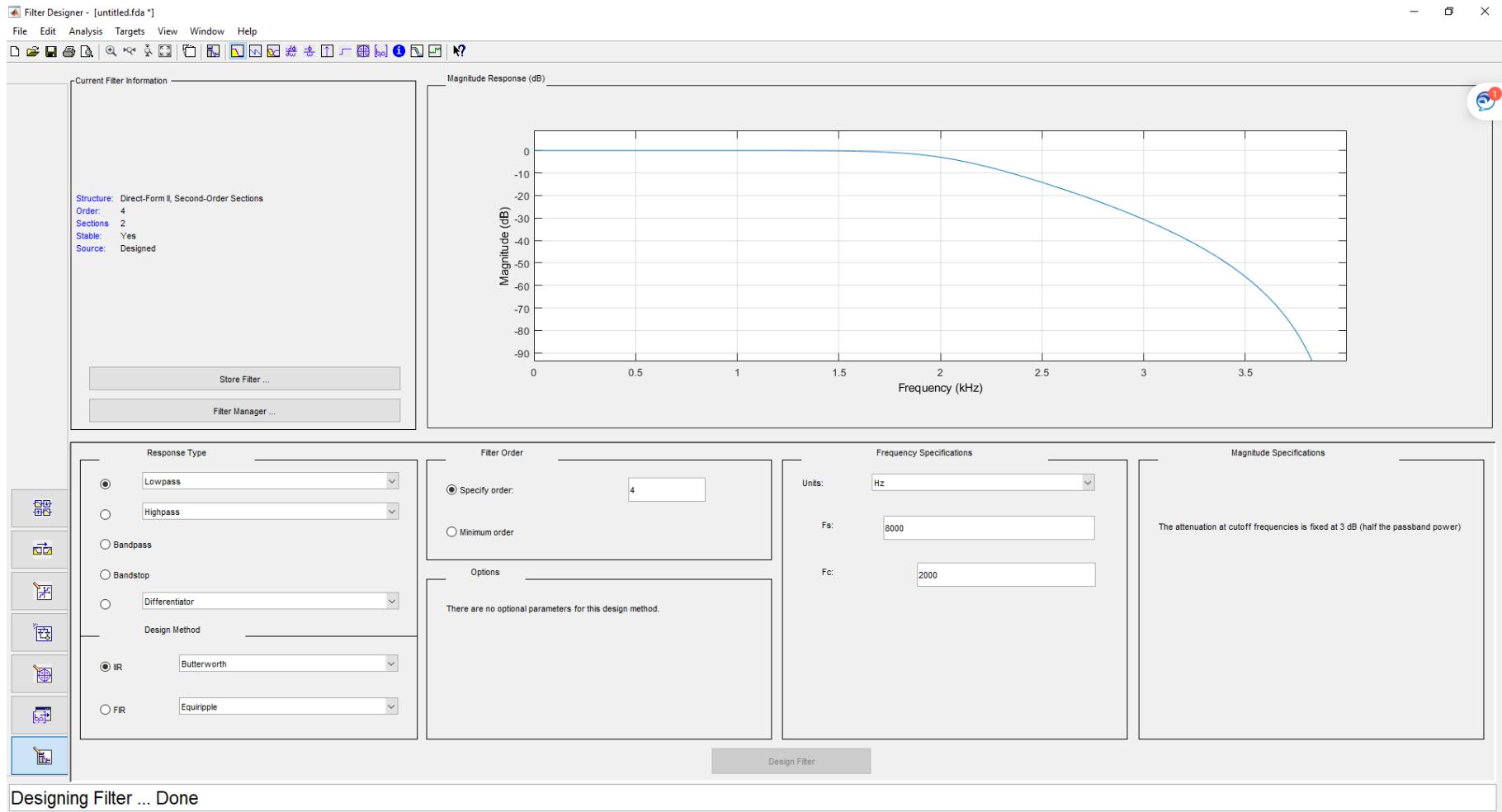
```
[N, Wn] = buttord(fDB/(fs/2), fSB/(fs/2), Apass, Astop);
```

- Butterworth-IIR-Filter entwerfen

```
[b, a] = butter(N, Wn);
```

# Matlab-Entwurf von FIR und IIR-Filtern

>> filterDesigner



# Real-time Signalverarbeitung mit FFT

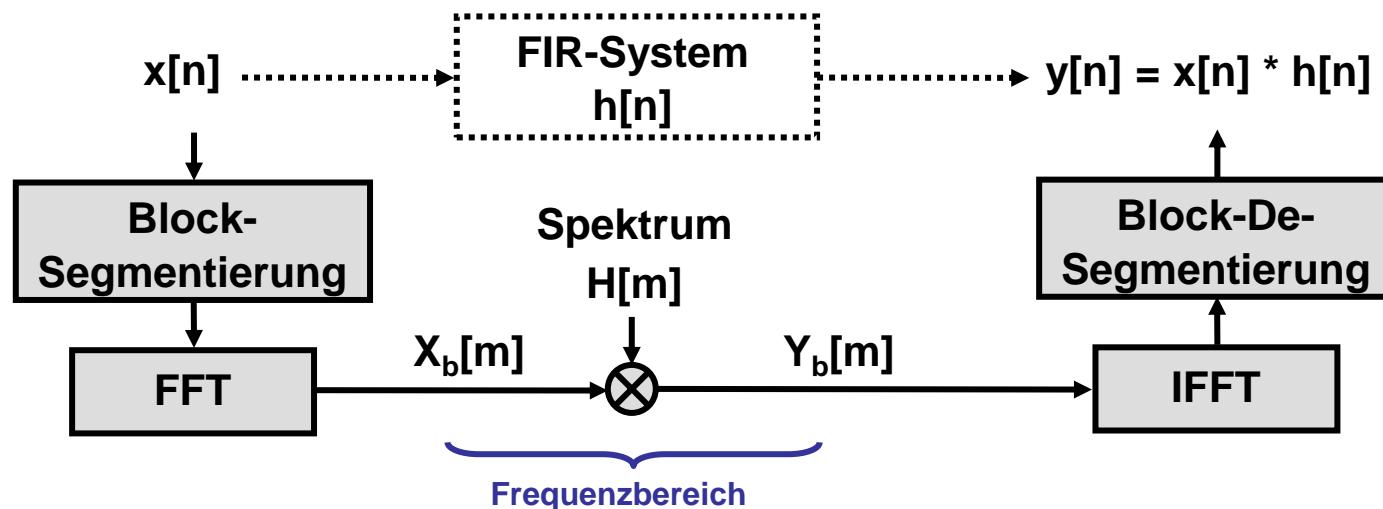
## Filterung mit FIR-Filter der Ordnung N im Zeitbereich

Berechnung der Faltungssumme,  $(N+1) \cdot f_s$  MAC-Operationen pro s

## Fast-convolution Algorithmus

Bestimmung der Ausgangsfolge via FFT/IFFT

Iohnt sich schon ab  $N > 100$



# Rechenbeispiel

## Problem

Echtzeit-Filterung mit FIR-Filter mit langer Impulsantwort  $h[n] = b_n$  von 44'100 Samples bzw. der Dauer von 1s wenn  $f_s = 44.1 \text{ kSps}$

## Option 1: FIR-Filterung im Zeitbereich

$44'100 \text{ MAC-Operationen/Sample} \cdot 44'100 \text{ Samples/s} = 1.94 \text{ GMAC/s}$

## Option 2: fast-convolution Algorithmus mit FFT/IFFT

Segmentierung des Signals in Blöcke von  $M = 20'000 \text{ Samples}$

zero-padding und FFT mit Blocklänge  $L = 2^{16} = 65'536 \text{ Samples}$

Multiplikation statt Faltung mit DFT-Frequenzgang  $H[m]$ ,  $m = 0, \dots, L-1$

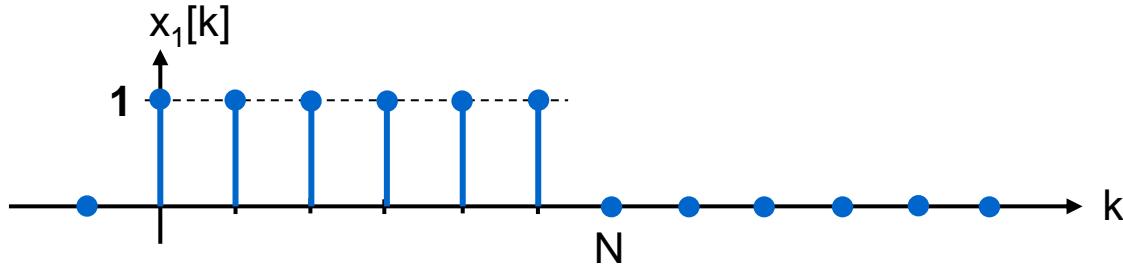
Inverse FFT und Addition der Blöcke der Länge  $L$

$\Rightarrow (2 \cdot L \cdot \log_2(L) + L) / M = 109 \text{ komplexe Multiplikationen pro Sample}$

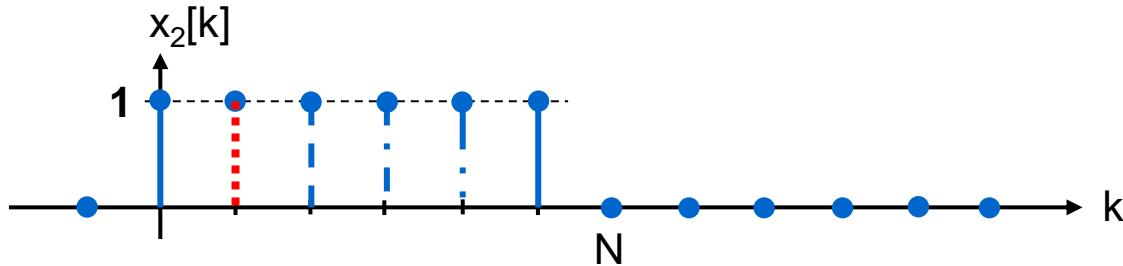
$\Rightarrow 436 \text{ reelle Multiplikationen pro Sample}$

$\Rightarrow 436 \cdot 44'100 \approx 20 \text{ MMultiplikationen/s}$

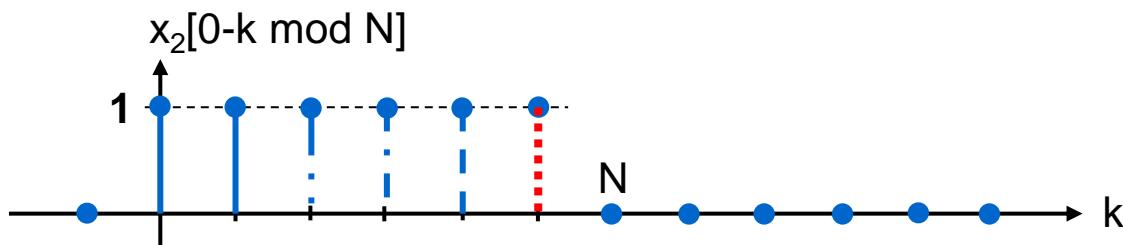
# Zirkuläre Faltung (z.B. von 2 rect-Pulsen)



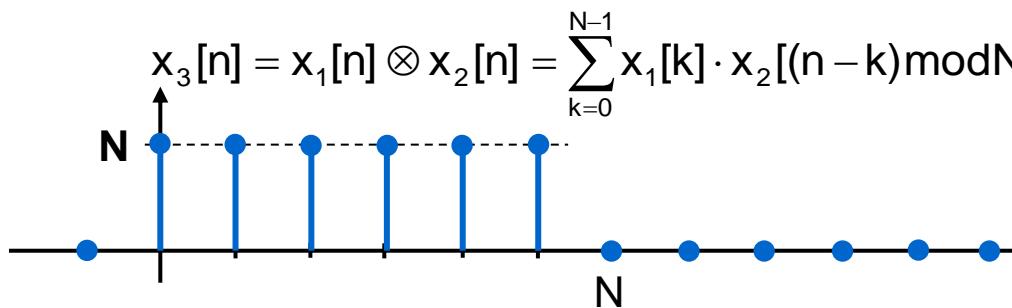
DFT  
○-●  $X_1[m] = \begin{cases} N & m = 0 \\ 0 & m = 1, \dots, N-1 \end{cases}$



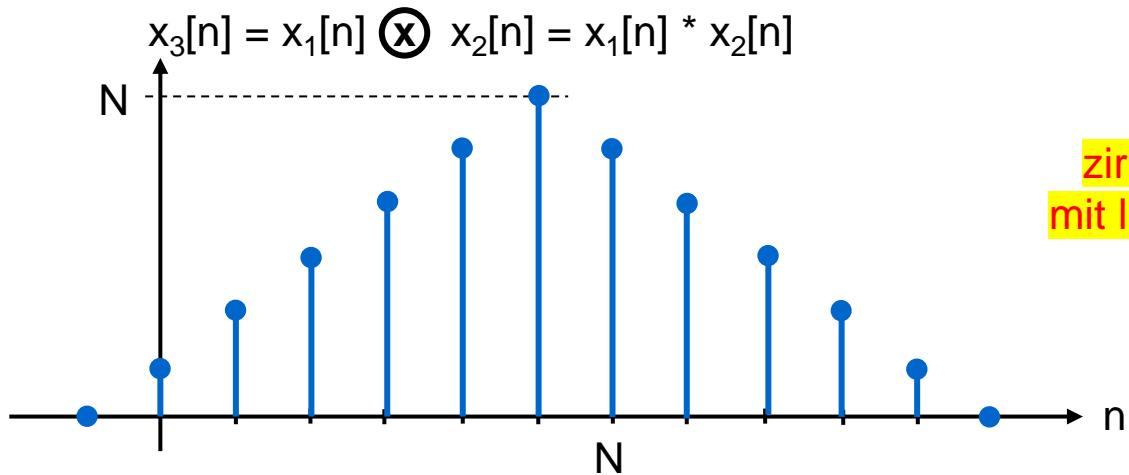
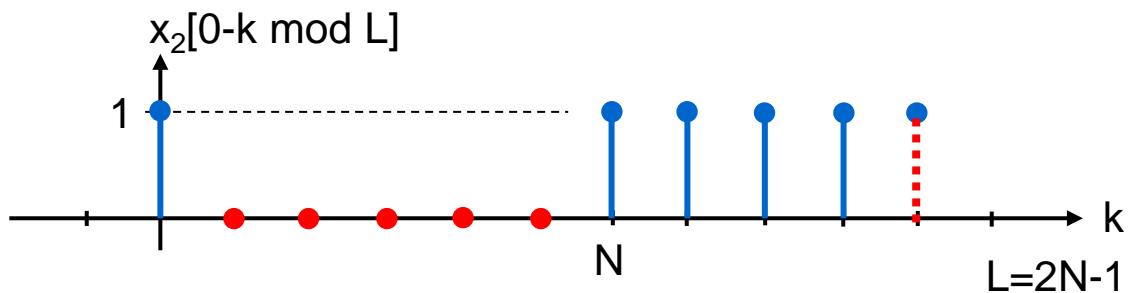
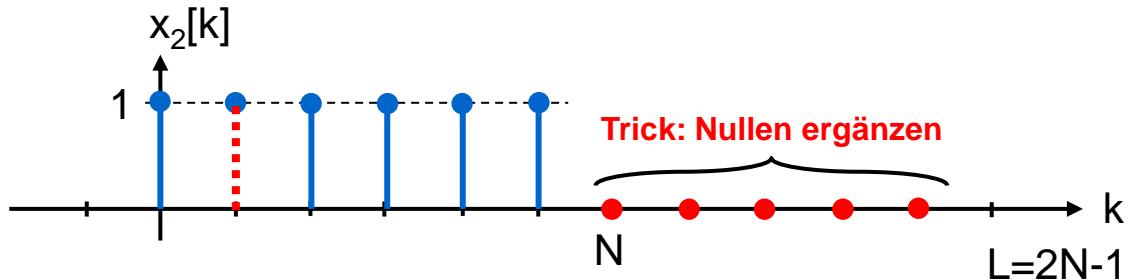
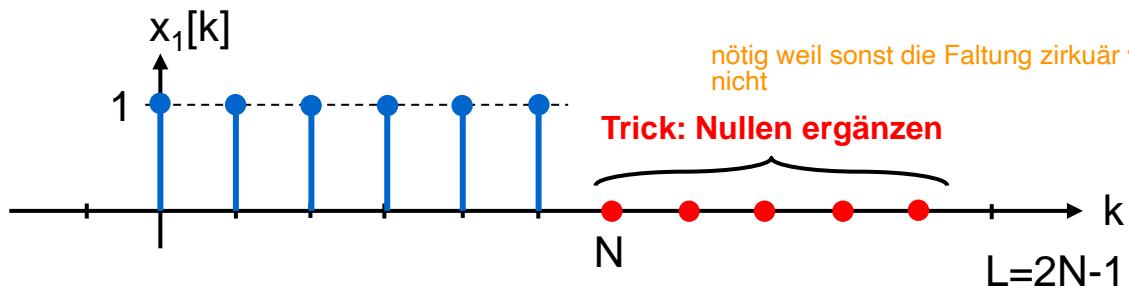
DFT  
○-●  $X_2[m] = \begin{cases} N & m = 0 \\ 0 & m = 1, \dots, N-1 \end{cases}$



nicht Dreieck-förmig !!!

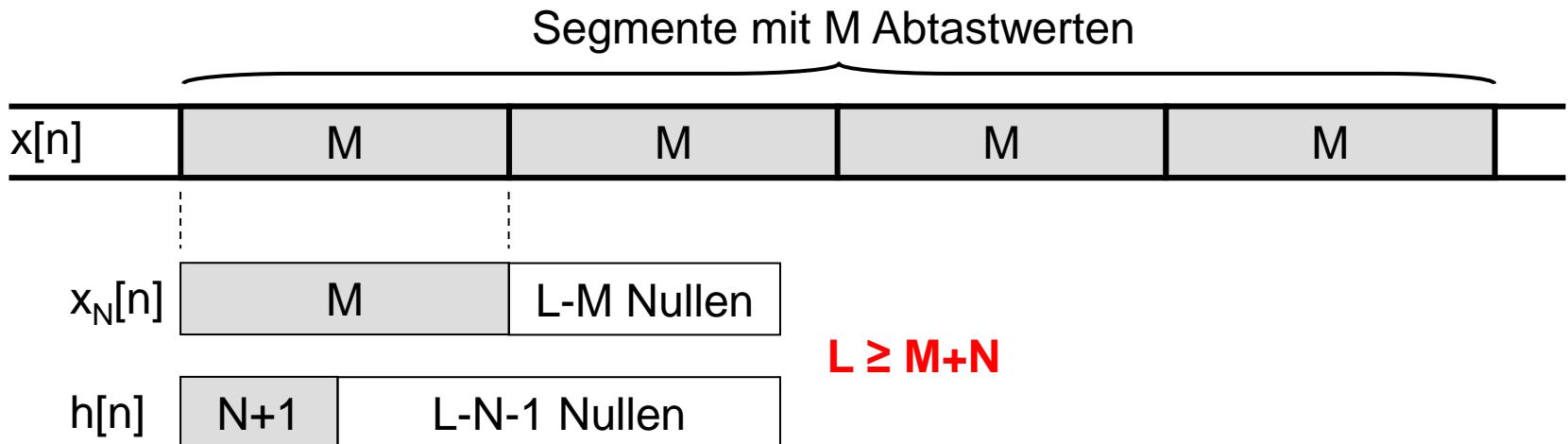


DFT  
○-●  $X_1[m] \cdot X_2[m]$   
 $= \begin{cases} N^2 & m = 0 \\ 0 & m = 1, \dots, N-1 \end{cases}$

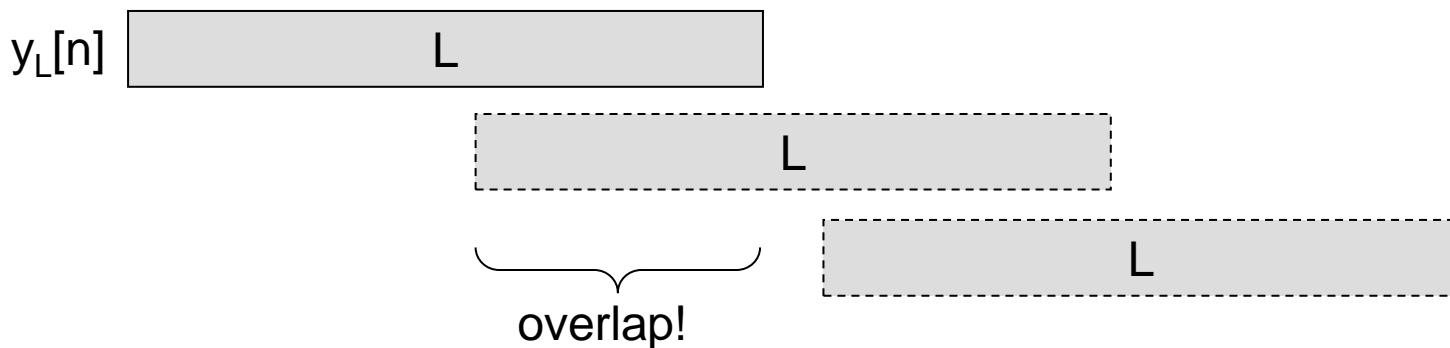


# Overlap-Add-Methode Matlab `fftfilt()`

DSV1, FFT-Filterung, 5



$L$ -Punkt-FFT, Multiplikation  $X_L[m] \cdot H_L[m]$ ,  $L$ -Punkt-IDFT



# Kapitel 5: Spezielle FIR-Filter

---

## Inhalt

- FIR-Differentiator-Filter
- FIR-Hilbert-Filter
- Raised-Cosine Puls-Shaping Filter
- Kamm- bzw. IFIR-Filter
- Echtzeit-Signalverarbeitung mit der FFT bzw. schnelle Faltung  
(overlap-add und overlap-save Algorithmen)

Linearphasig (PHASE) heisst, dass alle Frequenzkomponenten gleich lang verzögert werden. -> das heisst der output ist wieder genau gleich "verschoben" wie der input.

## Literatur

- A.V. Oppenheim, R.W. Schafer, J. R. Buck: „Zeitdiskrete Signalverarbeitung“, 2., überarbeitete Auflage, ISBN 3-8273-7077-9, Pearson Studium, 2004.
- Daniel Ch. Von Grünigen, “Digitale Signalverarbeitung: Bausteine, Systeme, Anwendungen”, Fotorotar Print und Media AG, 2008.
- D.G. Manolakis, V.K. Ingle, "Applied Digital Signal Processing", Theory and Practice, Cambridge University Press, 2011.

# Review FIR-Filter: Entwurf mit Fenstermethode

Theorie: FIR-Koeffizienten  $b_n$  stimmen mit LTI-Impulsantwort  $h[n]$  überein  
Ziel:  $b_n = h[n]$  so bestimmen, dass  $H(f)$  die Spezifikationen erfüllt

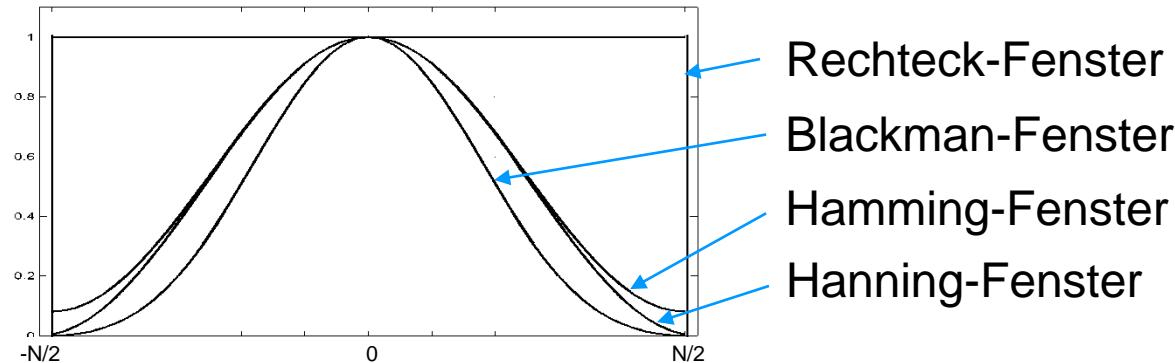
Start im Analogen:  $H_a(f)$  «brickwall»-förmig ○-●  $h_a(t)$  sin(t)/t-förmig

1. Stossantwort  $h_a(t)$  mit  $f_s$  abtasten (und  $T_s$  normieren):  $h_d[n] = T_s \cdot h_a(t=nT_s)$

idealer TP: 
$$h_d[n] = \frac{\sin(n\pi f_{DB}/(f_s/2))}{n\pi} \quad -\infty < n < \infty$$

2. Relevanten Anteil ausschneiden:  $h_c[n] = w[n] \cdot h_d[n]$  für  $-N/2 \leq n \leq N/2$

Fenster  $w[n]$ :



3. FIR-Filter mit Zeitverschiebung kausal machen:  $h[n] = h_c[n-N/2]$

# FIR-Differentiator-Filter

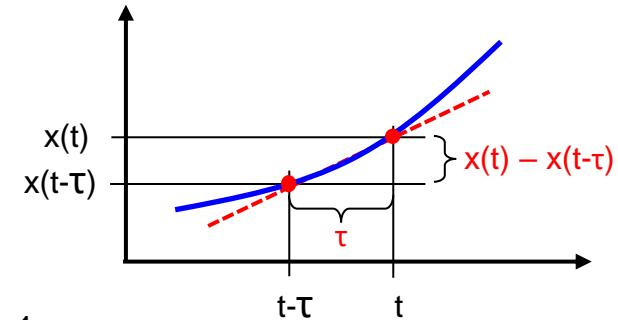
Es gibt noch weitere (nicht-brickwall-förmige) Referenz-Filter.

## Differentiator 1. Ordnung

Analog:  $y(t) = dx(t)/dt \approx (x(t) - x(t-\tau)) / \tau$

Differentiator 1. Ordnung

Digital:  $y[n] = x[n] - x[n-1]$  Matlab: `diff()`



z-UTF linearphasiges FIR-Filter:  $H(z) = 1 - z^{-1}$

Näherung ist aber nur „gut“ für tiefe Frequenzen (z.B.  $f < 0.15 \cdot f_s$ ) bzw. langsam veränderliche Signale (in Bezug zur Abtastfrequenz)

## FIR-Differentiatoren höherer Ordnung

Start im Analogen:  $y(t) = dx(t)/dt$      $\circ - \bullet$      $Y(f) = j2\pi f \cdot X(f)$

ableitung ersetzen durch  $j * \omega$  in frequenzland

Start im Analogen:  $y(t) = dx(t)/dt$      $\circ - \bullet$      $Y(f) = j2\pi f \cdot X(f)$

Impulsantwort linearphasiges FIR-Filter (ohne Fenster, acausal)

$H_a(f) = j2\pi f$      $\circ - \bullet$      $h_a(t)$      $\Rightarrow$  Sampling

inverse FFT

$$h_d[n] = \begin{cases} 0 & n = 0 \\ \frac{\cos(n \cdot \pi)}{n} & |n| > 0 \end{cases}$$

linearphasig!

# Beispiel: FIR-Differentiator 6. Ordnung

b-Koeffizienten mit Rechteckfenster

$$b = [1/3 \ -1/2 \ 1 \ 0 \ -1 \ 1/2 \ -1/3];$$

antisymmetrisch -> linearphasig

Hamming-Fenster  $w = \text{hamming}(7)'$

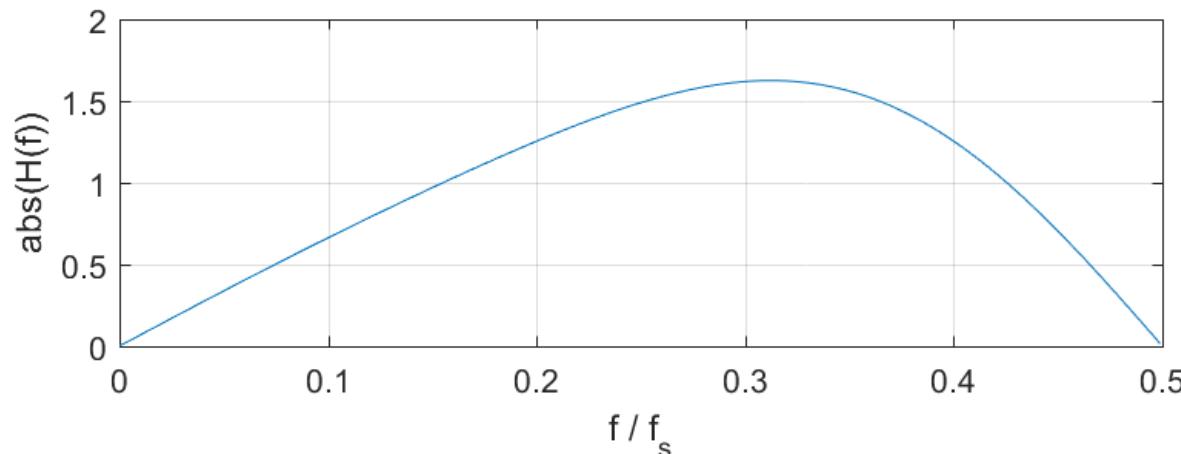
$$h_d[n] = \begin{cases} 0 & n = 0 \\ \frac{\cos(n \cdot \pi)}{n} & |n| > 0 \end{cases}$$

$$w = [0.08 \ 0.31 \ 0.77 \ 1.00 \ 0.77 \ 0.31 \ 0.08]$$

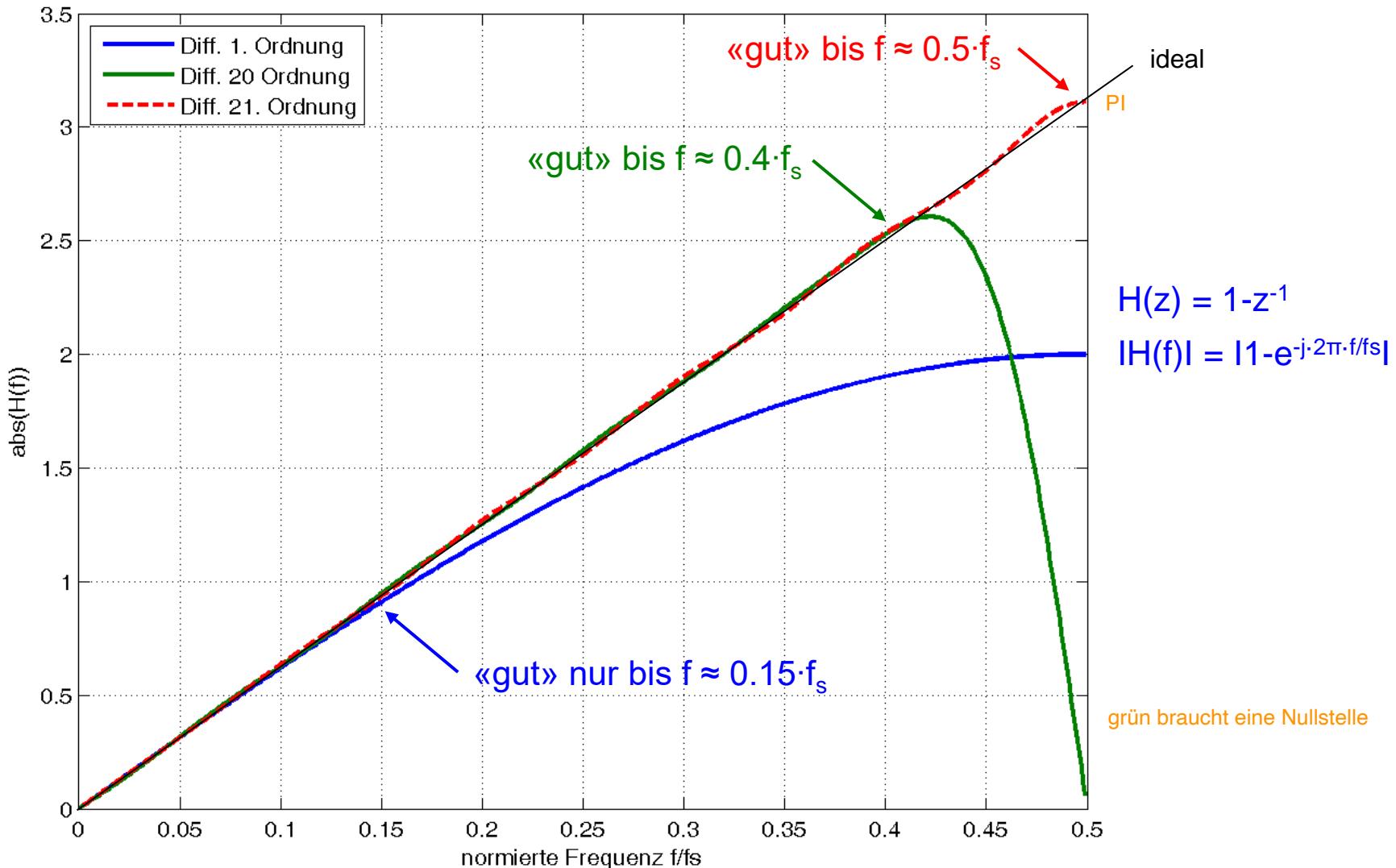
b-Koeffizienten mit Hamming-Fenster

$$b = [0.0267 \ -0.155 \ 0.77 \ 0 \ -0.77 \ 0.155 \ -0.0267]$$

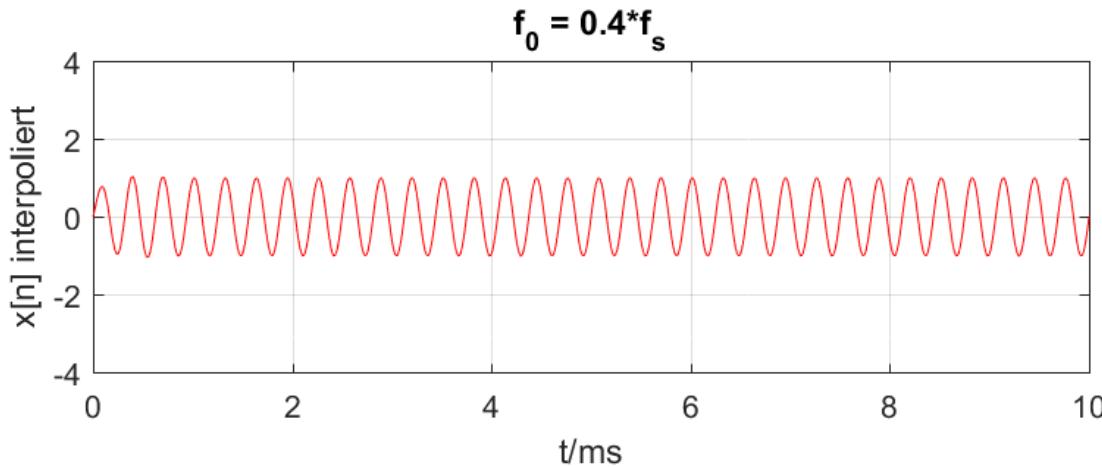
Amplitudengang  $|H(f)|$  (Phasengang ist linear, Zeitverzögerung  $3 \cdot T_s$ )



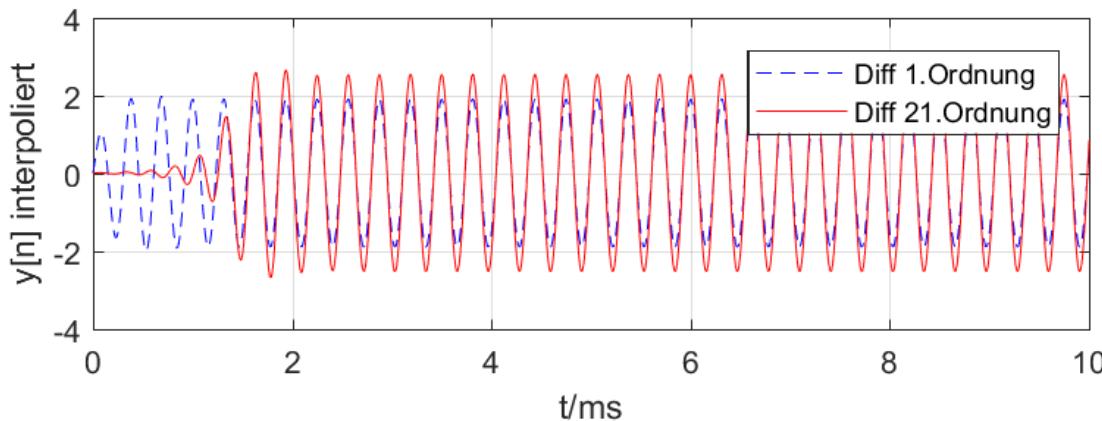
# FIR-Differentiator-Filter: Beispiel (I)



# FIR-Differentiator-Filter: Beispiel (II)



$$f_s = 8000 \text{ Hz}$$
$$f_0 = 3200 \text{ Hz}$$



analog:  $d/dt \sin(2\pi \cdot f_0 \cdot t) = A \cdot \cos(2\pi \cdot f_0 \cdot t)$  wobei  $A = 2\pi \cdot f_0$

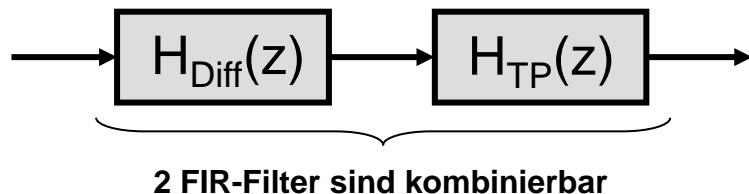
digital: Amplitude idealer Differentiator  $A \cdot T_s = 2\pi \cdot f_0 / f_s = 0.8 \cdot \pi = 2.51$

# FIR-Differentiator-Filter: Beispiel

- Mit der Matlab-Funktion `firpm()` können auch FIR-Differentiatoren vom **Typ 4 (anti-sym., ungerade Ordnung)** entworfen werden, die **keine Nullstelle bei  $f_s/2$**  aufweisen, und dem idealen Amplitudengang bis zu ganz hohen Frequenzen folgen können.
- Mit der Matlab-Funktion können FIR-Filter entworfen werden, deren Amplitudengänge **optimal im Minimax-Sinn** mit einem gewünschten Amplitudengang übereinstimmen.

$f = 0 \dots f_s/2$       Amplitude  
`b=firpm(21, [0 1], [0 pi], 'differentiator');` siehe oben

- Bei der Differentiation hat man oft das Problem, dass man das **höherfrequente Rauschen ungewollt verstärkt**. Abhilfe: TP in Serie

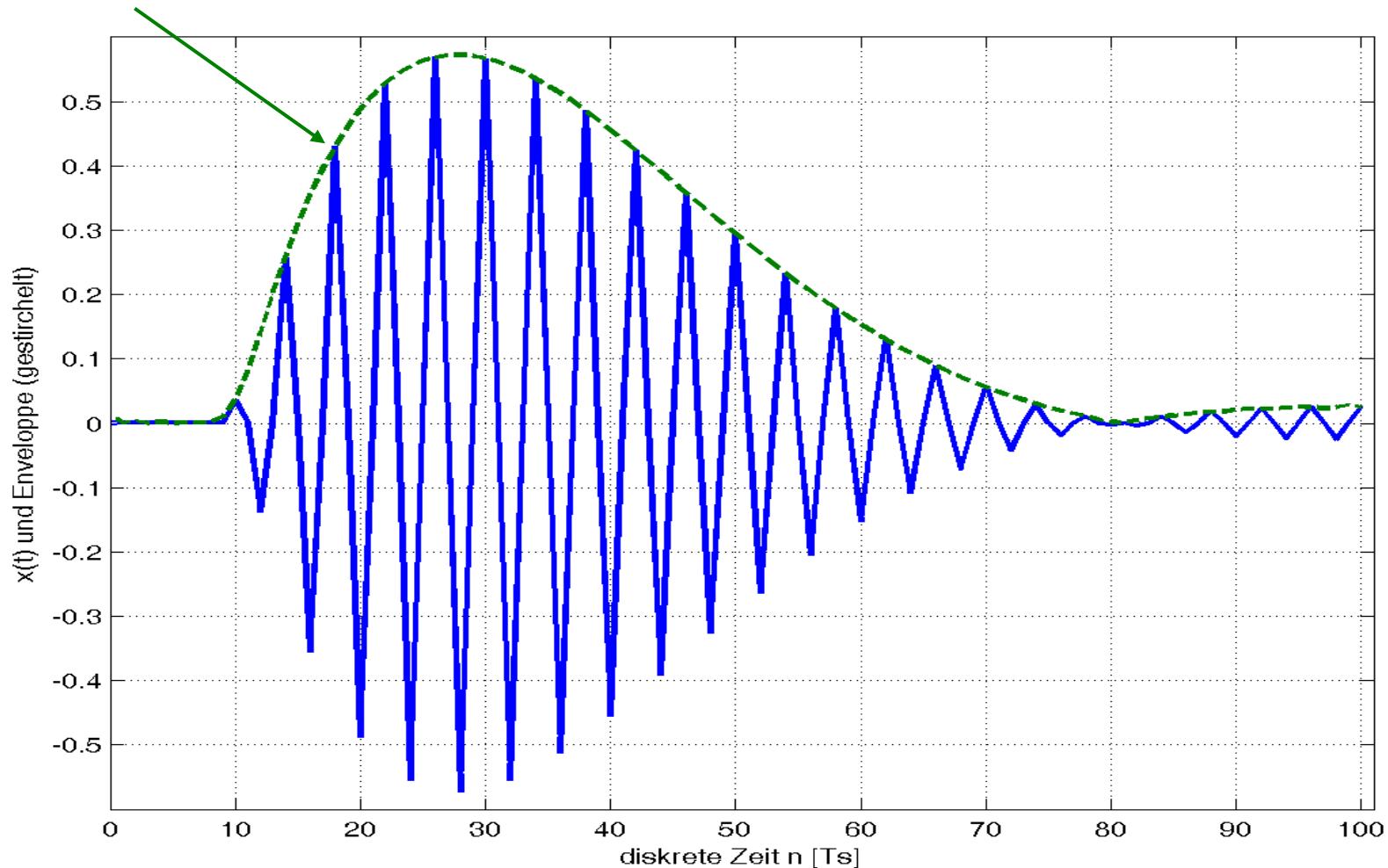


siehe z.B. Matlab-Tool `filterDesigner`

# Motivation Hilbert-Filter, -Transformation

## Bestimmung der Enveloppe / Hüllkurve eines Bandpass-Signals

**envelope** = abs(hilbert(**x**))



# FIR-Hilbert-Filter

**Start im Analogen:** ein Hilbert-Filter ist ein Allpass mit

$$\text{Frequenzgang } H_a(f) = \begin{cases} -j & f > 0 \\ +j & f \leq 0 \end{cases} \quad \text{breitbandig } 90^\circ\text{-Phasenschiebung}$$

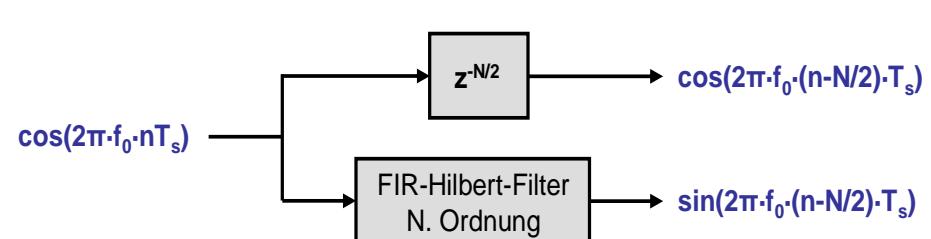
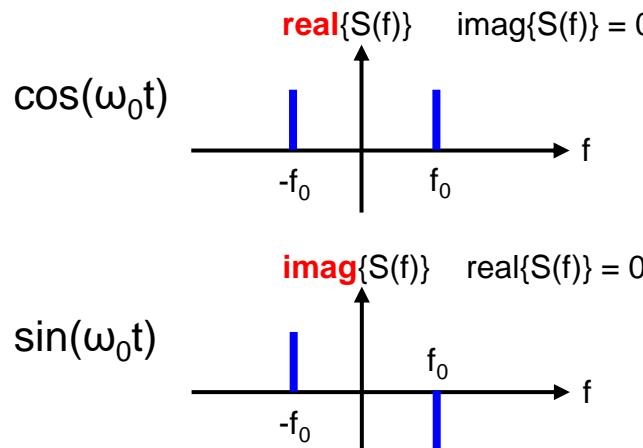
**Impulsantwort** linearphasiges FIR-Filter (akausal, ohne Fenster)

$$H_a(f) \circledast h_a(t) \Rightarrow \text{Sampling} \Rightarrow$$

$$h_d[n] = \begin{cases} 0 & n \text{ gerade} \\ \frac{2}{n \cdot \pi} & n \text{ ungerade} \end{cases}$$

linearphasig

## Mögliche Anwendung

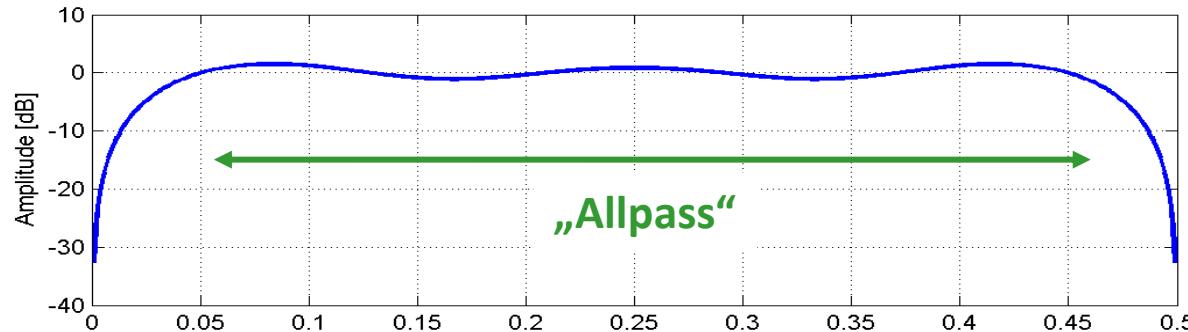


# FIR-Hilbert-Filter: Beispiel

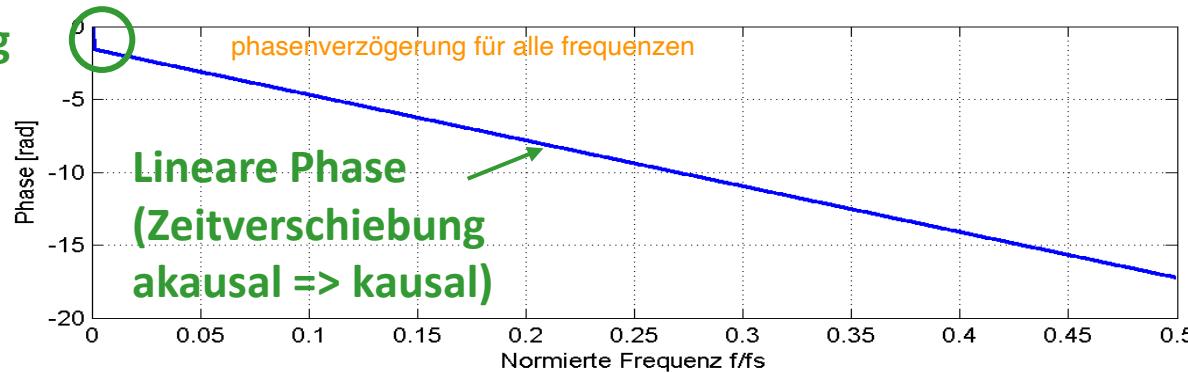
**Referenzfilter:**  $H_a(f) = -j \cdot \text{sign}(f) \rightarrow h_d[n] = 2 / (n\pi)$ , n ungerade sonst 0

**Beispiel:** FIR-Hilbert-Filter 10. Ordnung (Rechteck-Fenster)

$$b = [-0.1273 \ 0 \ -0.2122 \ 0 \ -0.6366 \ \textcolor{red}{0} \ 0.6366 \ 0 \ 0.2122 \ 0 \ 0.1273]$$



Phasensprung  
von  $-90^\circ$



# Hilbert-Transformation

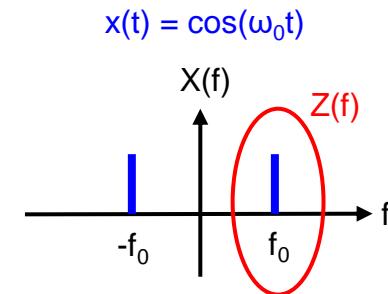
## Analytisches Signal $z(t)$ eines reellen Signals $x(t)$

weist nur positive Frequenzkomponenten auf (komplex-wertig)

Beispiel: gesucht  $z(t)$  von  $x(t) = \cos(\omega_0 \cdot t)$

$$x(t) = 0.5 \cdot \exp(j\omega_0 t) + 0.5 \cdot \exp(-j\omega_0 t)$$

$$z(t) = \exp(j\omega_0 t)$$



**Hilbert-Transformation:**  $z(t) = x(t) + j \cdot H(x(t))$  wobei  $H(\cdot)$  Hilbert-Filter  
addiere imaginären Hilbert anteil

Beweis:  $Z(f) = X(f) + j \cdot X(f) \cdot (-j \cdot \text{sign}(f))$

$f < 0$ :  $Z(f) = X(f) - X(f) = 0$  (keine negativen  $f$ -Komponenten!)

$f > 0$ :  $Z(f) = 2 \cdot X(f)$  ■

enthält nur positiven frequenzen von  $x(t)$

Matlab:  $z = \text{hilbert}(x)$

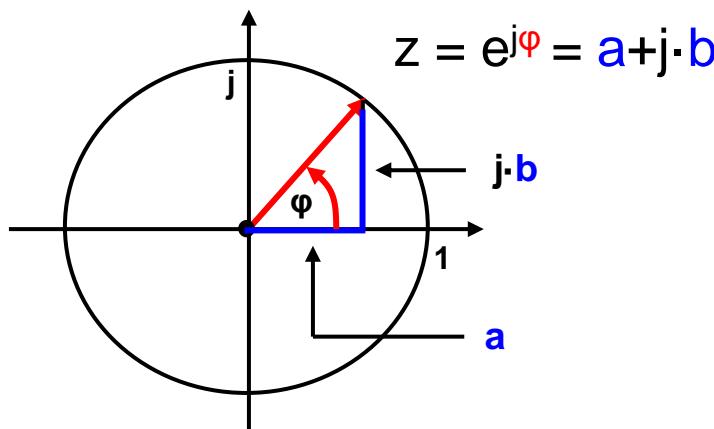
# Hilbert-Transformation

**Hilbert-Transformation:**  $z(t) = x(t) + j \cdot H(x(t))$  wobei  $H(\cdot)$  Hilbert-Filter

Beispiel: Hilbert-Transformation von  $x(t) = \cos(\omega_0 \cdot t)$

$$\begin{aligned} z(t) &= \cos(\omega_0 \cdot t) + j \cdot H(\cos(\omega_0 \cdot t)) \\ &= \cos(\omega_0 \cdot t) + j \cdot \sin(\omega_0 \cdot t) = \exp(j\omega_0 t) \end{aligned}$$

Reminder:



# Hilbert-Transformation

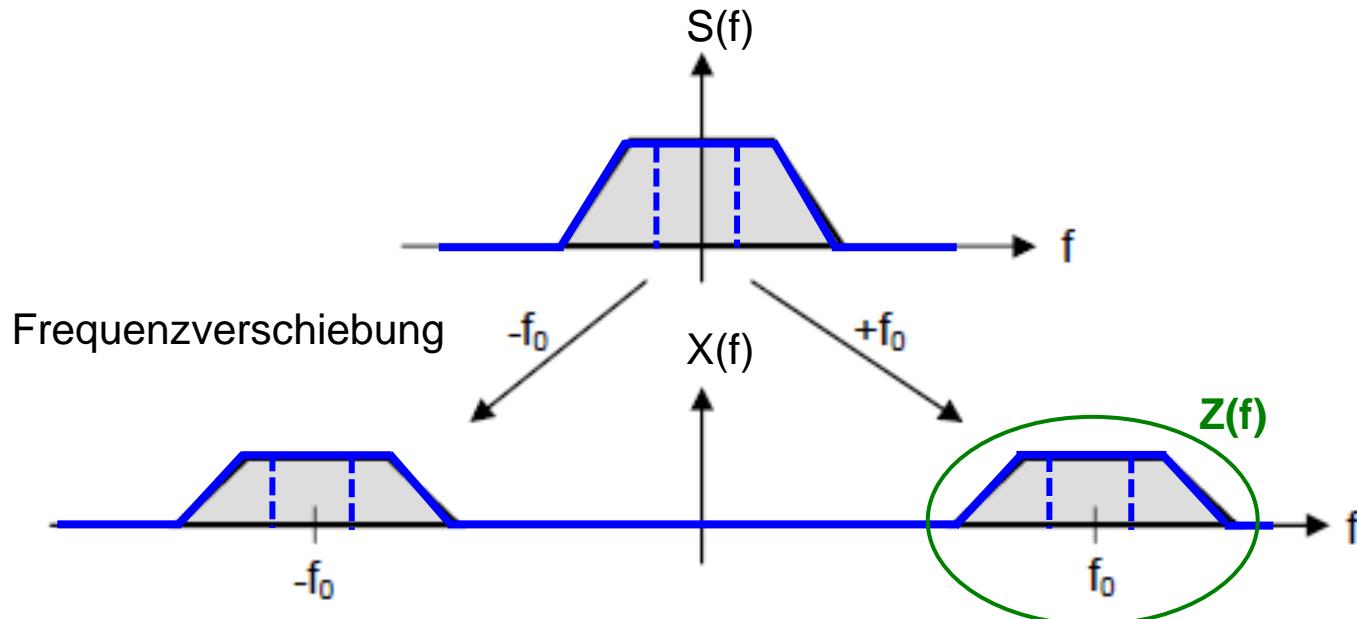
## Enveloppe eines Bandpass- bzw. AM-Signals

$$x(t) = s(t) \cdot \cos(\omega_0 \cdot t) \quad \bullet-\circ \quad X(f) = 0.5 \cdot S(f-f_0) + 0.5 \cdot S(f+f_0)$$

$$y(t) = H(x(t)) = s(t) \cdot \sin(\omega_0 \cdot t) \quad \bullet-\circ \quad Y(f) = -j \cdot 0.5 \cdot S(f-f_0) + j \cdot 0.5 \cdot S(f+f_0)$$

$$z(t) = x(t) + j \cdot y(t) \quad \bullet-\circ \quad Z(f) = S(f-f_0)$$

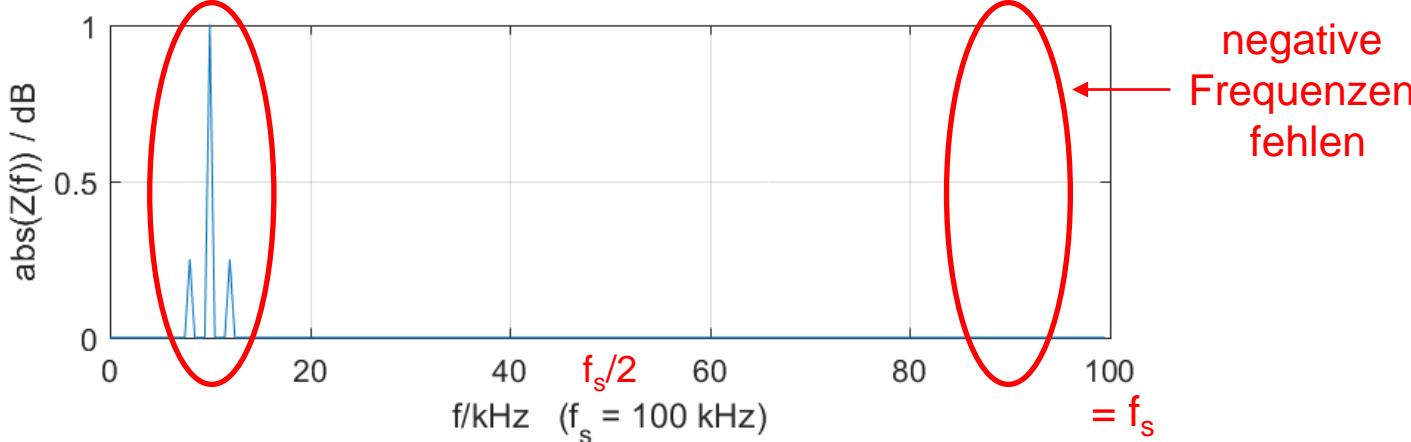
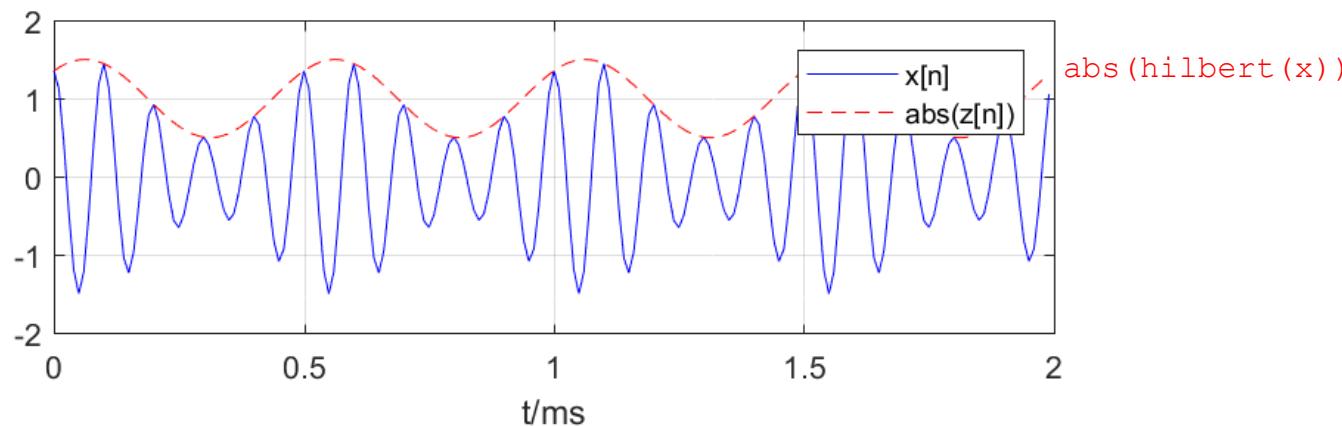
$|z(t)| = |s(t)| \cdot |\exp(j\omega_0 t)| = |s(t)|$  bzw. Umhüllende von  $x(t)$  !



# Hilbert-Transformation: Beispiel

```
s = 1+0.5*cos(2*pi*f0*t-pi/4);
x = s.*cos(2*pi*fc*t); % AM-Signal, f0=2e3, fc=10e3
enveloppe = abs(hilbert(x));
```

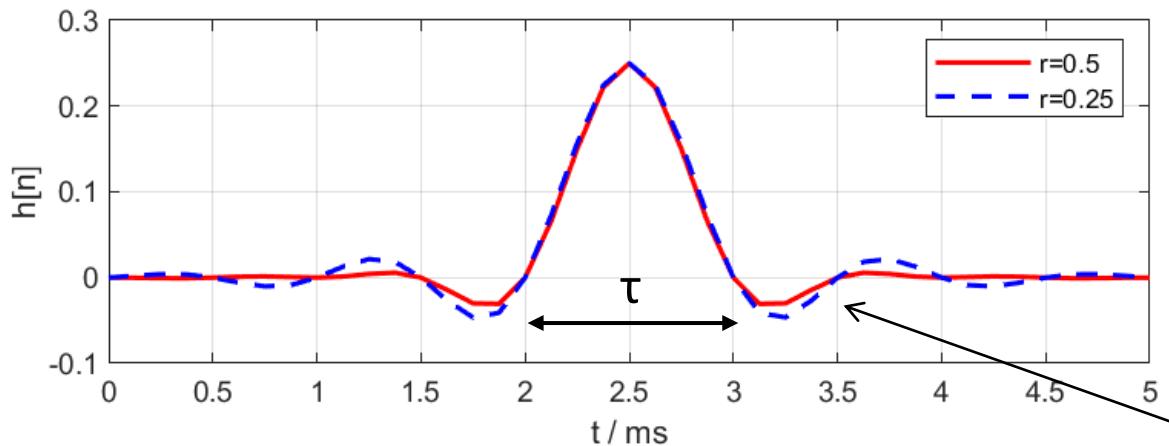
abs weil hilbert komplexwertig ist.



# Raised Cosine Pulse (an einem Beispiel)

nicht prüfungsrelevant?

```
b=firrcos(N,fc,r,fs,'rolloff');
```



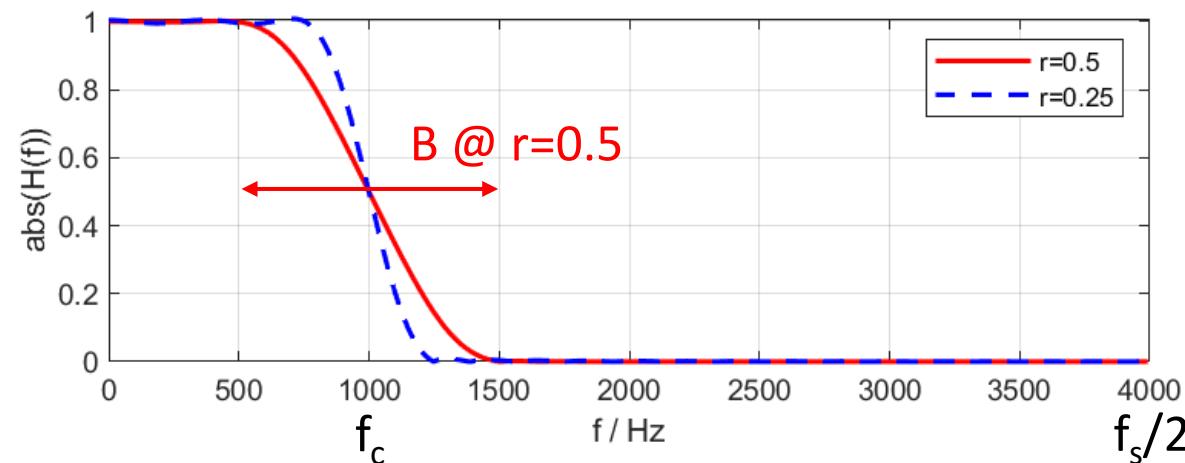
Filterordnung N=40

cutoff-frequency  $f_c = 1/\tau$

roll-off factor r

Abtastfrequenz  $f_s = 8 \text{ kSps}$

Übergangsbereich  $B = f_c \cdot 2r$



gedämpfter sin(t)/t-Verlauf

(je kleiner r, desto sinc-ähnlicher und langandauernder der Puls bzw. rechteckförmiger das Spektrum)

Weiteres PulsfILTER: Gauss-FIR-Filter gaussfir(), gaussdesign()

# Schmalband-I FIR-Filter

## Interpolated FIR-Filter

sind (sehr) schmalbandige FIR-Filter

## Kamm-Filter

jede Verzögerung  $z^{-1}$  des FIR-Filters  $H(z)$  ver-M-fachen

$$H_{\text{comb}}(z) = H(z^M) \quad \text{bzw.} \quad H_{\text{comb}}(f) = H(z = e^{j2\pi M f T_s})$$

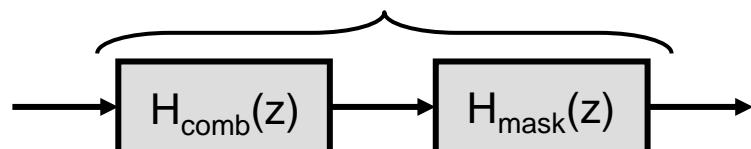
$H_{\text{comb}}(f)$  weist M Perioden des ursprünglichen Digitalfilters  $H(f)$  auf

=> Kamm-Form bzw. Frequenzstauchung

FIR-Filter  $H(z)$  hat Bandbreite B, Kamm-Filter  $H_{\text{comb}}(z)$  Bandbreite B/M

## Kaskade von Kamm-Filter mit Maskierungsfilter ergibt IFIR-Filter

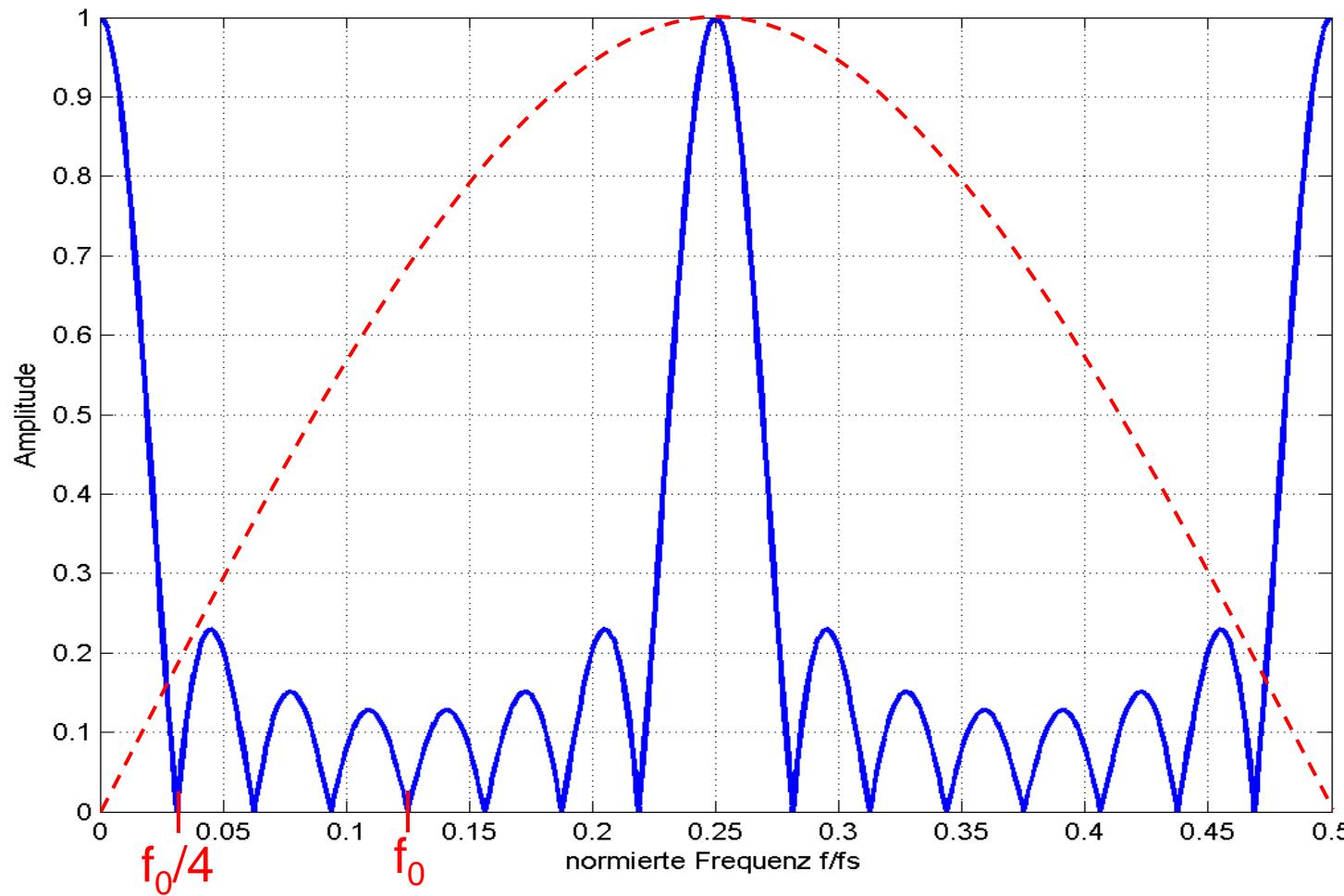
$$H_{\text{IFIR}}(z) = H_{\text{comb}}(z) \cdot H_{\text{mask}}(z)$$



# Schmalband-IIR-Filter: Beispiel (I)

Moving-Average-Filter  $H(z) = (1+z^{-1}+\dots+z^{-7})/8$  hat 1. Nullstelle @  $f_0 = f_s/8$

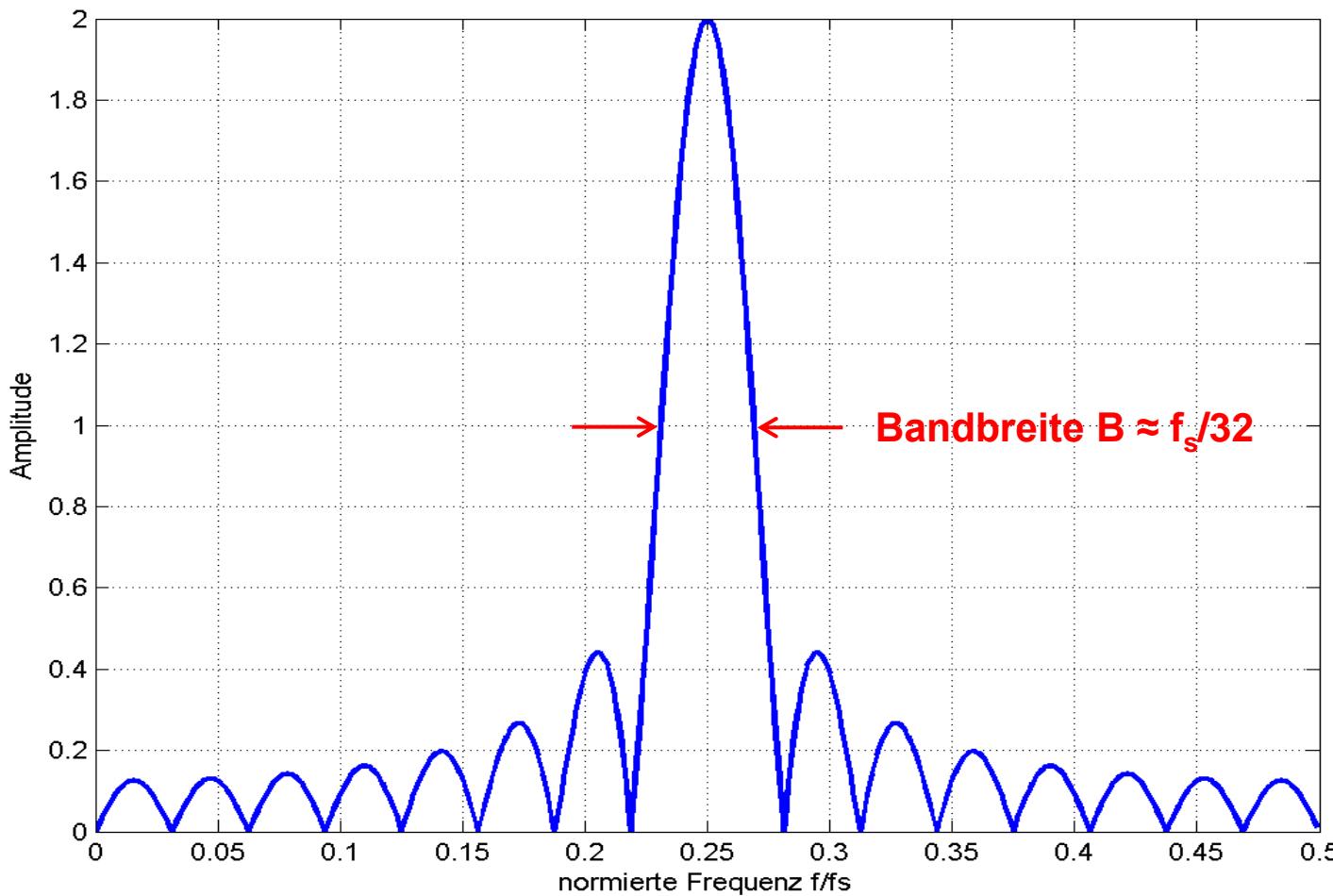
Kamm-Filter  $H_{\text{comb}}(z) = H(z^4)$  hat 1. Nullstelle @  $f_0/4 = f_s/32$  (blaue Linie)

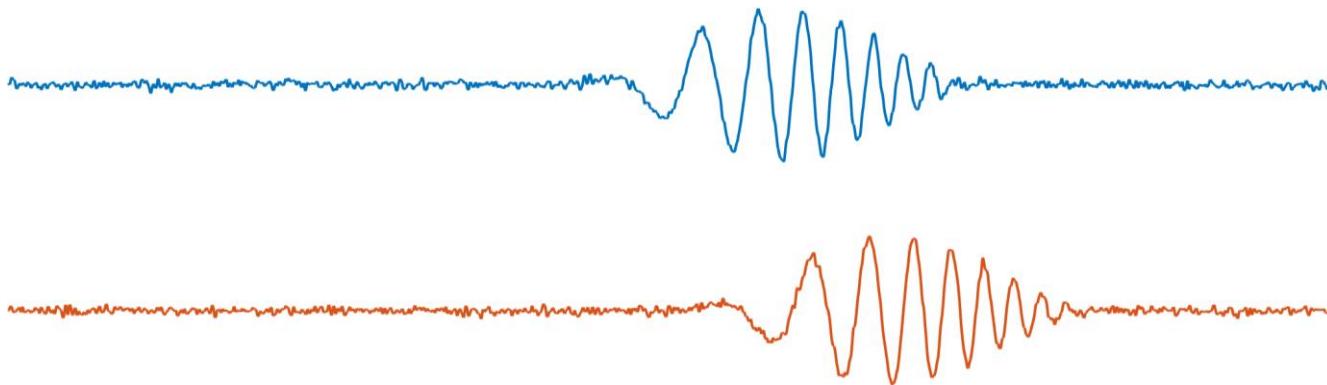


# Schmalband-IFIR-Filter: Beispiel (II)

$H_{\text{mask}}(z) = 1 - z^{-2}$  mit Nullstellen @  $f=0$  und  $f=f_s/2$ , siehe rote Linie oben

schmales FIR-BP-Filter bzw. IFIR-Filter  $H_{\text{IFIR}}(z) = H_{\text{comb}}(z) \cdot H_{\text{mask}}(z)$





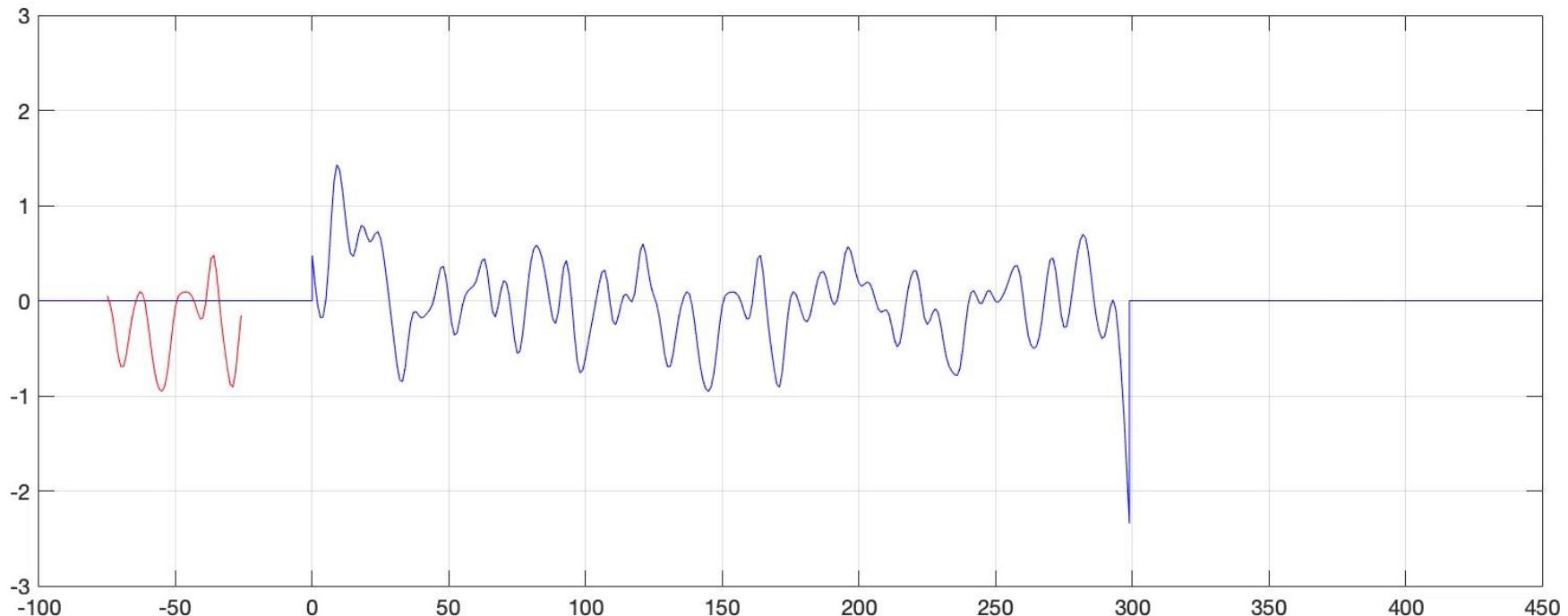
Die folgenden Folien stammen grösstenteils aus der ZHAW-Probelektion von Josquin Rosset vom 28.4.2023 mit dem Titel «Korrelation von digitalen Signalen».

## Weitere Referenzen:

- [1] Von Grünigen, Daniel Ch.: Digitale Signalverarbeitung. Grundlagen und Anwendungen. AT Verlag, Aarau 1993.
- [2] Ohm, Jens-Rainer; Lüke, Hans Dieter: Signalübertragung. Springer-Verlag, Heidelberg 2002.
- [3] Smith, Steven W.: The Scientist and Engineer's Guide to Digital Signal Processing, <https://www.dspguide.com/ch7/3.htm>, abgerufen am 26.04.2023.

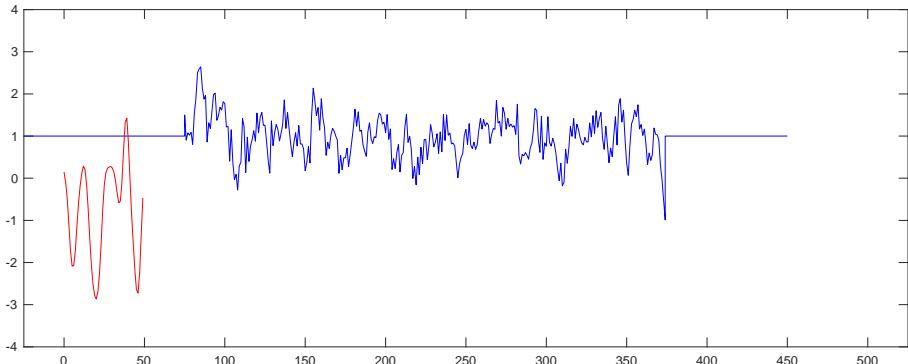
# Einführendes Beispiel

Problem: Finden Sie den **roten Signalausschnitt  $y[n]$**  (auch Pattern oder Muster genannt) im **blauen Signal  $x[n]$** .



...und wenn das blaue Signal  $x[n]$

- verrauscht ist?
- eine andere Verstärkung hat (Gain)?
- nicht DC-frei ist (Offset)?
- invertiert ist?



→ Das **Skalarprodukt** bzw. die **Korrelation** ist immer dann am grössten, wenn die Vektoren bzw. Signalausschnitte am ähnlichsten sind, und Null, wenn die Vektoren orthogonal sind.

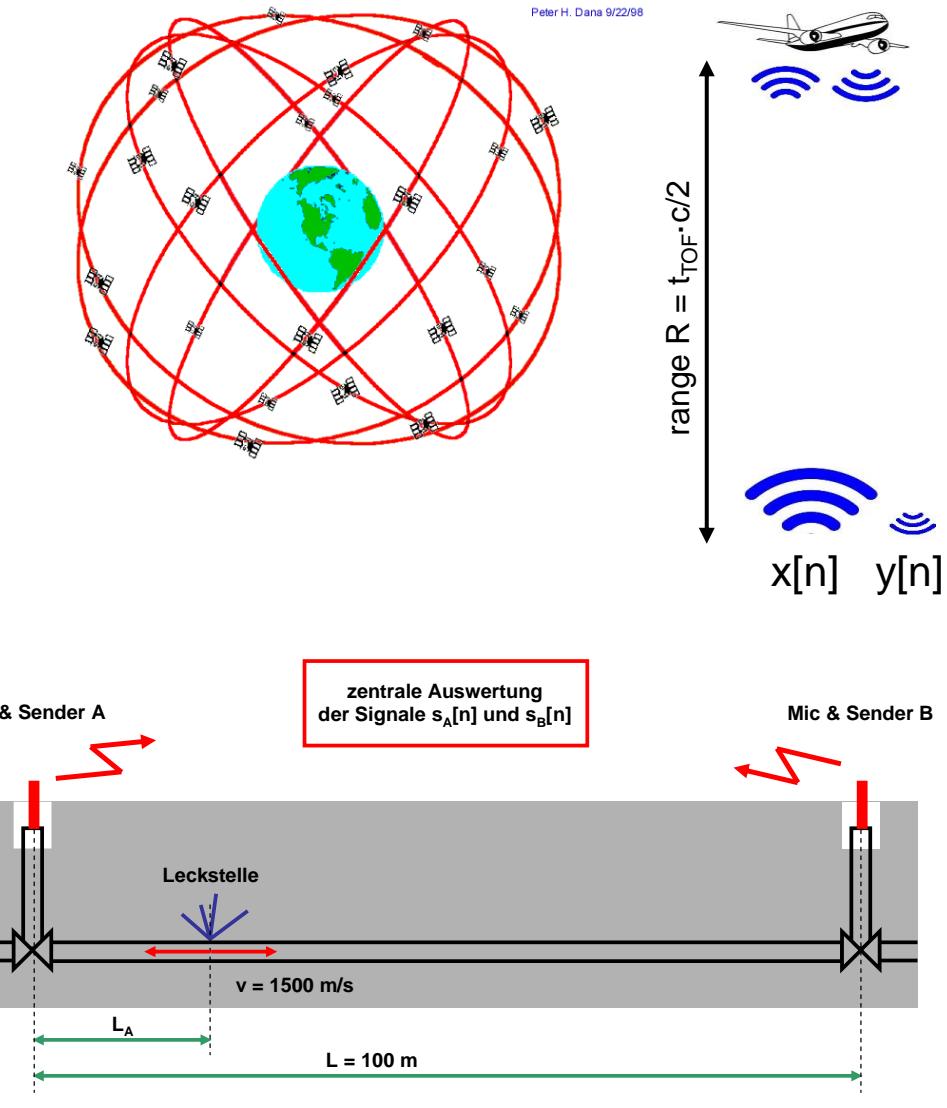
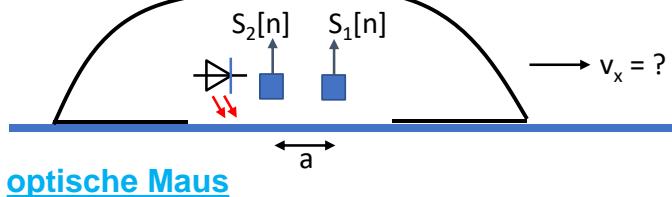
Die Korrelation ist ein **Mass für die Ähnlichkeit** zweier Merkmale, Zustände, Funktionen.

Die Korrelation zweier Signale  $x(t)$  und  $y(t)$  ergibt eine Funktion  $r_{xy}(\tau)$ , welche die **Ähnlichkeit der beiden Signale** in Abhängigkeit der Verschiebungszeit  $\tau$  beschreibt [1].

# Anwendungen

Sehr viele, hier nur einige:

- GPS
- Ortung (TDOA) / TOF (time-of-flight)
- Radar
- Bildverarbeitung
- Stereo-Vision (2D-Korrelation)
- Empfänger-Synchronisierung
- Matched-Filter in Nachrichtentechnik
- Messung der Impulsantwort mit Pseudo-Noise
- ...



# Kreuzkorrelationsfunktion (KKF)

Die (unnormierte) **Kreuzkorrelationsfunktion**  $r_{xy}[m]$  für die reellen Signale  $x[n]$  und  $y[n]$  ist wie folgt definiert, wobei  $m$  den «lag» bezeichnet:

$$r_{xy}[m] = \sum_{n=-\infty}^{\infty} x[n + m] \cdot y[n]$$

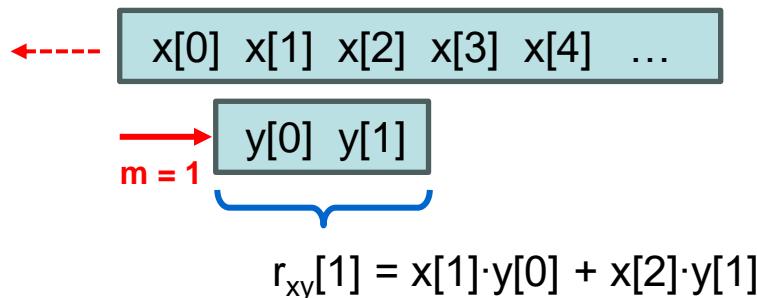
Wenn das Signal  $y[n]$   $N$  Samples lang ist, d.h.  $y[n] = [y[0] \dots y[N-1]]$ , gilt:

$$r_{xy}[m] = \sum_{n=0}^{N-1} x[n + m] \cdot y[n]$$

$$m \geq -(N - 1)$$

Matlab: `rxy = xcorr(x, y)`

Beispiel: Wert  $r_{xy}[1]$  wenn  $m = 1$  und  $y[n]$  nur  $N = 2$  Samples lang ist



Für  $m > 0$ , Verschiebung von

- $y[n]$  nach rechts (→)  
d.h.  $\sum_n x[n] \cdot y[n-m]$
- oder,  $x[n]$  nach links (←)  
d.h.  $\sum_n x[n+m] \cdot y[n]$

Eigenschaft der KKF

$$r_{xy}[m] = r_{yx}[-m]$$

d.h. wenn x und y vertauscht werden, ist  $r_{xy}$  an der Ordinate gespiegelt.

Numerisches Beispiel

Die KKF  $r_{xy}[m] = \sum_{n=0}^{N-1} x[n+m] \cdot y[n]$  der folgenden Signale

$$x[n] = [ 0.5 \ 1.0 \ 1.0 \ -0.5 \ -1.0 ]$$

$$y[n] = [ 1.0 \ -0.5 ] \rightarrow$$

gibt:  $r_{xy}[m] = [ -0.25 \ 0 \ 0.5 \ 1.25 \ 0 \ -1 ]$  für  $m = -1, 0, \dots, 4$

Achtung Definition der KKF in der Literatur oft  $r_{xy}[m] = \sum_{n=0}^{N-1} x[n] \cdot y[n+m]$   
Hier wird aber die gleiche Definition wie in Matlab verwendet.

Falls  $x[n] = y[n]$  wird  $r_{xy}[m] = r_{xx}[m]$  und heisst **Autokorrelationsfunktion**.

- Die Autokorrelation  $r_{xx}[m]$  ist ein Mass für die Ähnlichkeit eines Signals  $x[n]$  mit einer um **m** Abtastintervalle verschobenen Kopie von sich **selbst**.
- Der Wert der Autokorrelationsfunktion ist maximal, wenn  **$m = 0$**  ist, und entspricht dann der **Signalleistung**:

$$r_{xx}[m = 0] = \sum_{n=-\infty}^{\infty} x[n] \cdot x[n] = \sum_{n=-\infty}^{\infty} x[n]^2$$

## Numerisches Beispiel

Die AKF des Signals  $x[n] = [1 \ 1 \ -1]$  lautet

$\Rightarrow r_{xx}[m] = [-1 \ 0 \ 3 \ 0 \ -1]$  für  $m = -2, \dots, 2$

## Faltung

$$(x * y)[m] = \sum_{n=-\infty}^{\infty} x[m-n] \cdot y[n]$$

(Faltung ist kommutativ)

$$= \sum_{n=-\infty}^{\infty} x[n] \cdot y[m-n]$$

## Kreuzkorrelationsfunktion

$$r_{xy}[m] = \sum_{n=-\infty}^{\infty} x[n+m] \cdot y[n]$$

+ bedeutet x nach links schieben  
wenn positiv und nach rechts  
wenn negativ

nicht kommutativ!

Die Kreuzkorrelation kann durch Faltung von  $x[n]$  mit dem umgedrehten bzw. gefalteten Signal  $y'[n] = y[-n]$  berechnet werden, d.h.

$$r_{xy}[m] = x[n] * y'[n] = x[n] * y[-n]$$

$k = -n$

$y[-n] = y'[n]$

Beweis:  $r_{xy}[m] = \sum_k x[k+m] \cdot y[k] = \sum_n x[m-n] \cdot y[-n] = x[n] * y'[n]$

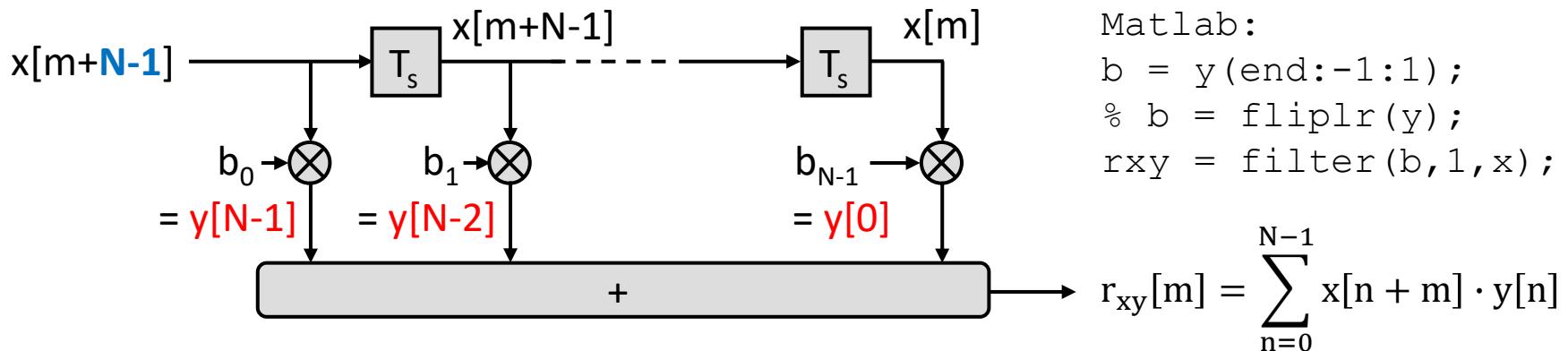


Die **KKF** kann mit der **Faltung**  $r_{xy}[m] = x[n] * y[-n]$  berechnet werden.

Die **Faltung** kann mit dem **FIR-Filter** mit Ordnung  $N-1$  und Koeffizienten

$$b_n = y[N-1-n], \quad n = 0, \dots, N-1$$

berechnet werden, mit einer **Verzögerung** von  $(N-1) \cdot T_s$ .



Mit einem **FIR-Filter** kann also das Ende des Musters  $y[n]$ ,  $n = 0, \dots, N-1$ , im Signal  $x[n]$  detektiert werden, wenn das Muster  $y[n]$  umgedreht und als Koeffizientenvektor  $b$  verwendet wird (**=> FIR-Korrelator**).

## Cross-correlation `xcorr`

- Doku: “By default, `xcorr` computes raw correlations with no normalization.”
- Doku: “If  $x$  and  $y$  have different lengths, the function appends zeros to the end of the shorter vector so it has the same length as the other.”  
=> eher ungeeignet, wenn das Signal  $x[n]$  lange und das Muster  $y[n]$  kurz ist
- Achtung:  $r_{xy}[\text{lag} = 0]$  ist in der Mitte des resultierenden Korrelationsvektors !!!

## Beispiel

```
x = [0.5 1 1 -0.5 -1];
y = [1 -0.5];

[rxy,lags] = xcorr(x,y)
=> rxy = 0    0    0 -0.25 0  0.5 1.25 0   -1
=> lags = -4  -3  -2  -1  0   1   2   3   4

xcorr(x,y,2) % nur lags im Bereich [-2 2]
=> 0 -0.25 0 0.5 1.25
```

## Convolution conv

- Berechnung der Korrelation mit der Faltung:  $r_{xy}[m] = x[n] * y[-n]$  bzw.  
 $r_{xy} = \text{conv}(x, \text{fliplr}(y))$
- kein zero-padding,  $\text{length}(r_{xy}) = \text{length}(x) + \text{length}(y) - 1$
- Achtung:  $r_{xy}[\text{lag} = 0]$  ist an der Stelle  $N = \text{length}(y)$  von  $r_{xy}$

## Beispiel

```
x = [0.5 1 1 -0.5 -1];  
y = [1 -0.5]; % d.h. N = 2
```

```
rxy = conv(x, fliplr(y)) % Wert für lag = 0 ist an 2. Stelle  
=> -0.25 0 0.5 1.25 0 -1.0
```

m = -1

m = 4

## Filter filter

- Berechnung der Korrelation mit dem FIR-Filter

```
b = fliplr(y); a = 1; % FIR-Filter  
rxy = filter(b,a,x)
```

- `length(rxy) = length(x)`, wenn y kürzer ist !!!

- Detektion des **Endes** des Musters  $y[n]$  mit N Samples im Signal  $x[n]$ :

```
[maxval,maxindex] = max(rxy);  
x(maxindex-N+1:maxindex) und y(1:N) haben die grösste Ähnlichkeit  
"optimale" Verschiebung m_opt = maxindex-N;
```

## Beispiel:

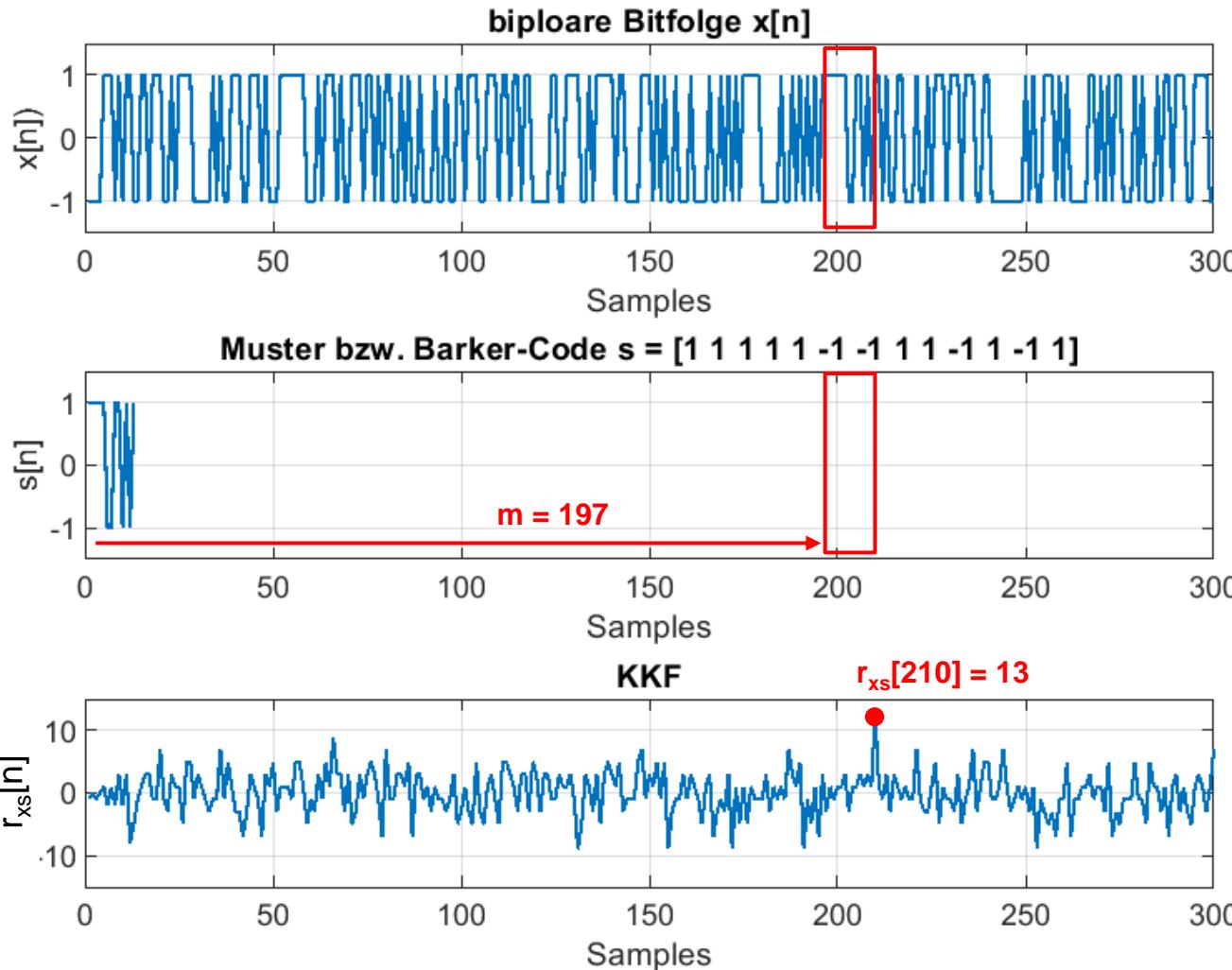
```
x = [0.5 1 1 -0.5 -1];  
y = [1 -0.5]; % d.h. N = 2
```

```
rxy = filter(fliplr(y),1,x)  
=> -0.25 0 0.5 1.25 0  
% max(rxy)=1.25 ist an der Stelle maxindex = 4  
% d.h. [x(3) x(4)] = [1 -0.5] und y haben grösste Ähnlichkeit
```

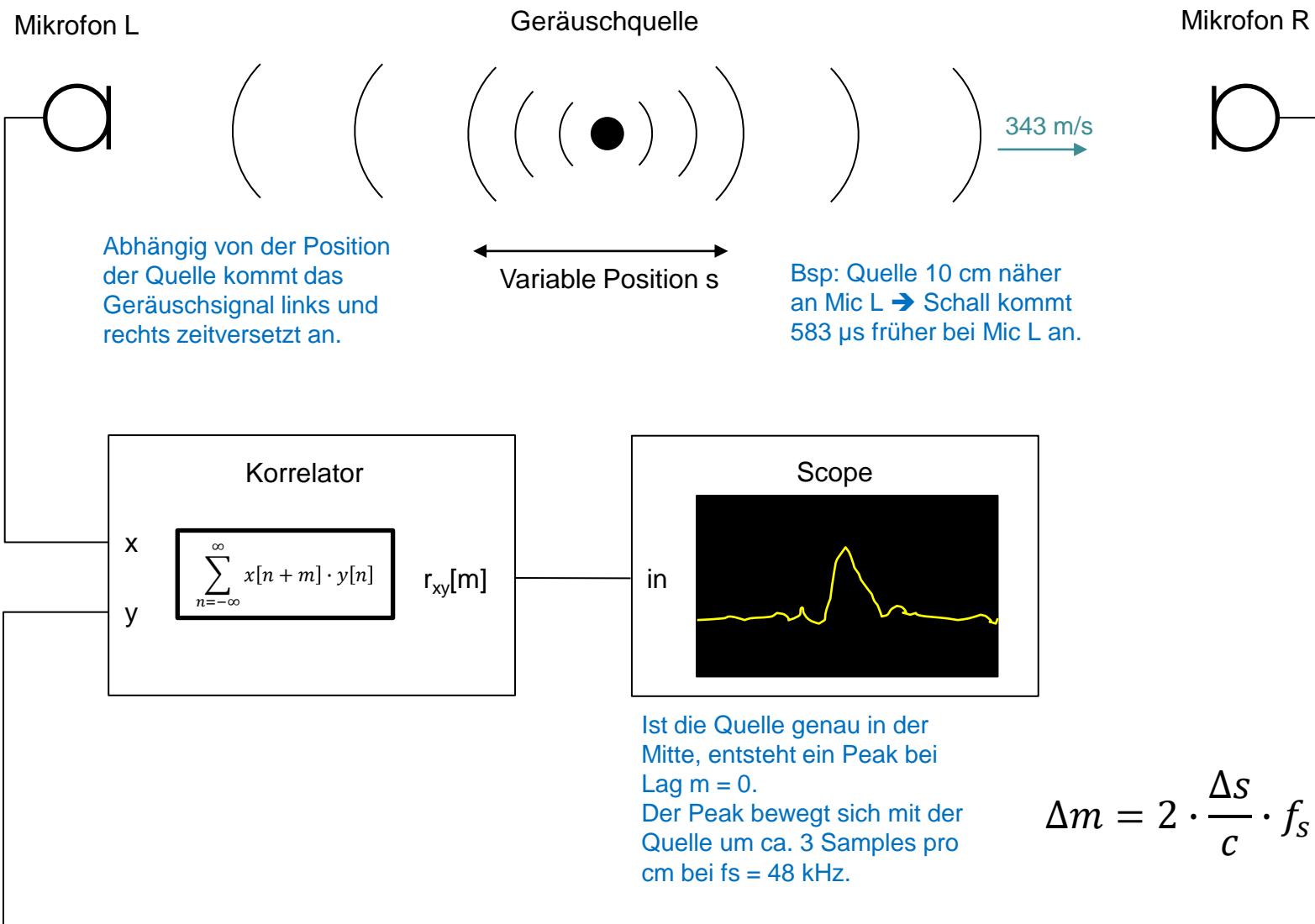
# Beispiel: Bitmuster-Synchronisation

Kommt der Barker-Code  $s[n]$  der Länge 13 in der Bitfolge  $x[n]$  vor und wenn ja, wo?

=> Antwort: ja, Max-Wert 13 tritt in der KKF an der Stelle 210 auf, d.h.  $x(198:210) = s$



# Demo Geräuschlokalisierung



**Kreuzkorrelation** zwischen reellen Signalen  $x[n]$  und  $y[n]$ :

$$r_{xy}[m] = \sum_{n=-\infty}^{\infty} x[n+m] \cdot y[n]$$

**z-Transformation:**  $R_{xy}(z) = X(z) \cdot Y(1/z) = X(z) \cdot Y(z^{-1})$

- Beweis:  $r_{xy}[m] = x[n]^*y[-n]$   
Faltung im Zeitbereich  $\Leftrightarrow$  Multiplikation im z-Bereich  
 $y[-n] \Leftrightarrow Y(1/z)$  ■
- Beispiel:  $x[n] = [1 \ 1]$ ,  $y[n] = [1 \ -1]$ ,  $n \geq 0 \Rightarrow r_{xy}[m] = [-1 \ 0 \ 1]$ ,  $m = -1, 0, 1$   
 $R_{xy}(z) = X(z) \cdot Y(1/z) = (1+z^{-1}) \cdot (1-z) = -z + z^{-1}$

**Spektrum:**  $R_{xy}(f) = X(f) \cdot Y^*(f)$  und  $R_{xx}(f) = |X(f)|^2$

- Beispiel: Wenn  $x[n]$  eine «impulsförmige» AKF  $r_{xx}[m] \approx \delta_0[m]$  hat, dann gilt:  $R_{xx}(f) = |X(f)|^2 \approx 1$ , d.h.  $|X(f)| \approx 1$ , d.h.  $x[n]$  enthält «alle» Frequenzkomponenten bzw. ist weiss.

# 7. Multiraten-Signalverarbeitung

## Inhalt

- Motivation für Multiratensysteme
  - Downsampling und Dezimation
  - Upsampling und Interpolation
  - Rationale Ratenverhältnisse
  - $\Sigma\Delta$ -Wandler
  - Polyphasenfilter, CIC-Filter
- 
- Folien-Satz 1
- Folien-Satz 2

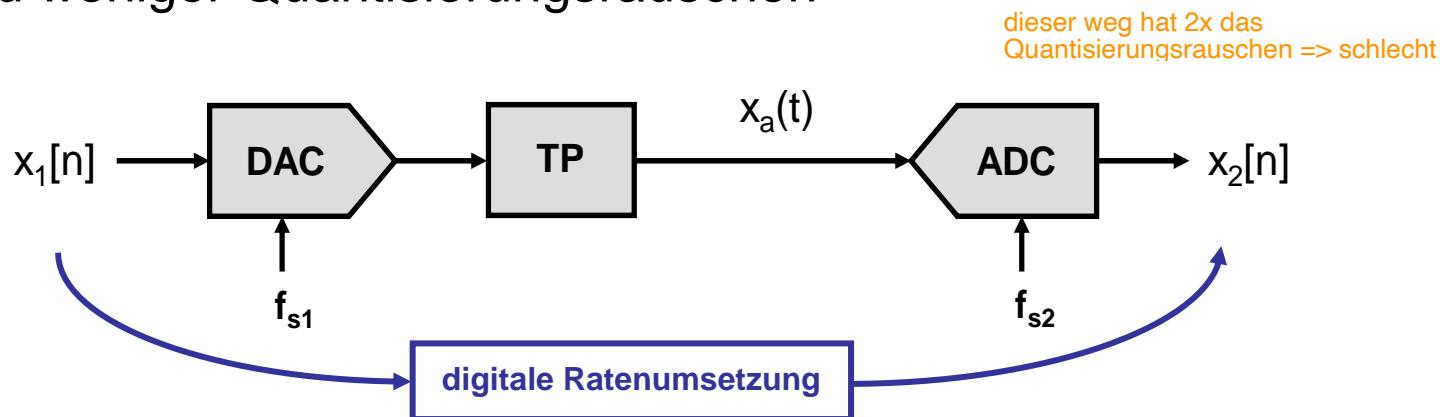
## Referenzen

- [1] Oppenheim, Schafer, „Zeitdiskrete Signalverarbeitung“, Pearson, 2004.
- [2] S.K. Mitra, „Digital Signal Processing“, Mc Graw Hill, 2006.
- [3] Kester, Editor, „Analog-Digital Conversion“, Analog Devices, 2004.
- [4] D.G. Manolakis, V.K. Ingle, "Applied Digital Signal Processing", Theory and Practice, Cambridge University Press, 2011. => 15.0\_pp\_705\_776\_Multirate\_signal\_processing.pdf
- [5] DSV2-Skript, <\\shared.zhaw.ch\\public\\staff\\rumc\\out\\dsv2>

# Motivation

## Taktratenänderungen in der DSV

- sind manchmal unumgänglich, z.B. wegen verschiedener Normen (Audio-CD  $f_s = \underline{44.1}$  kHz, Audio-Technik  $f_s = 32, \underline{48}, \underline{96}, 192$  kHz)
- sind oft gewollt, weil effizienter (z.B. Rechenaufwand bzw. Ordnung von FIR-TP-Filtern mit  $f_g \ll f_s$  ist typisch gross, Abhilfe  $f_s$  verkleinern)
- sind dem Umweg DAC-ADC auf jeden Fall vorzuziehen, weil einfacher und weniger Quantisierungsrauschen



## Multiraten-Systeme

sind zeitdiskrete Systeme, die verschiedene Abtastraten in unterschiedlichen Teilsystemen verwenden

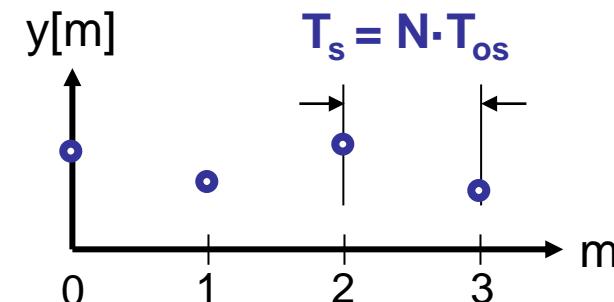
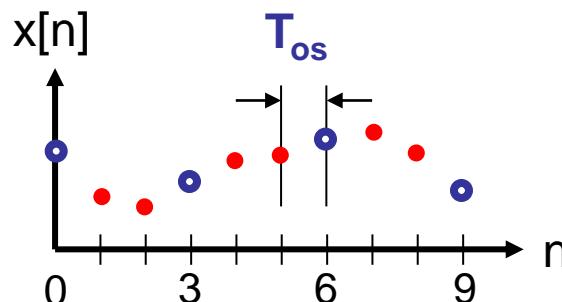
# Downsampling: Analyse im Zeitbereich

## Abwärtstastung

- nur jeder N-te Sample von  $x[n]$  wird in  $y[m]$  kopiert  
 $\Rightarrow$  N-fache Ratenreduktion von  $f_{os}$  auf  $f_s$
- einfachste Form der (verlustbehafteten) Datenreduktion
- Blockdiagramm und Beispiel mit  $N=3$  und  $Phase=0$

$$f_{os} = f_{\text{oversampling}}$$

$$f_s = f_{os}/N$$



- Matlab: `x=[0:20]; N=4; Phase=1; y=downsample(x,N,Phase)`  
 $y = 1 \ 5 \ 9 \ 13 \ 17$

# Downsampling: Analyse im Frequenzbereich

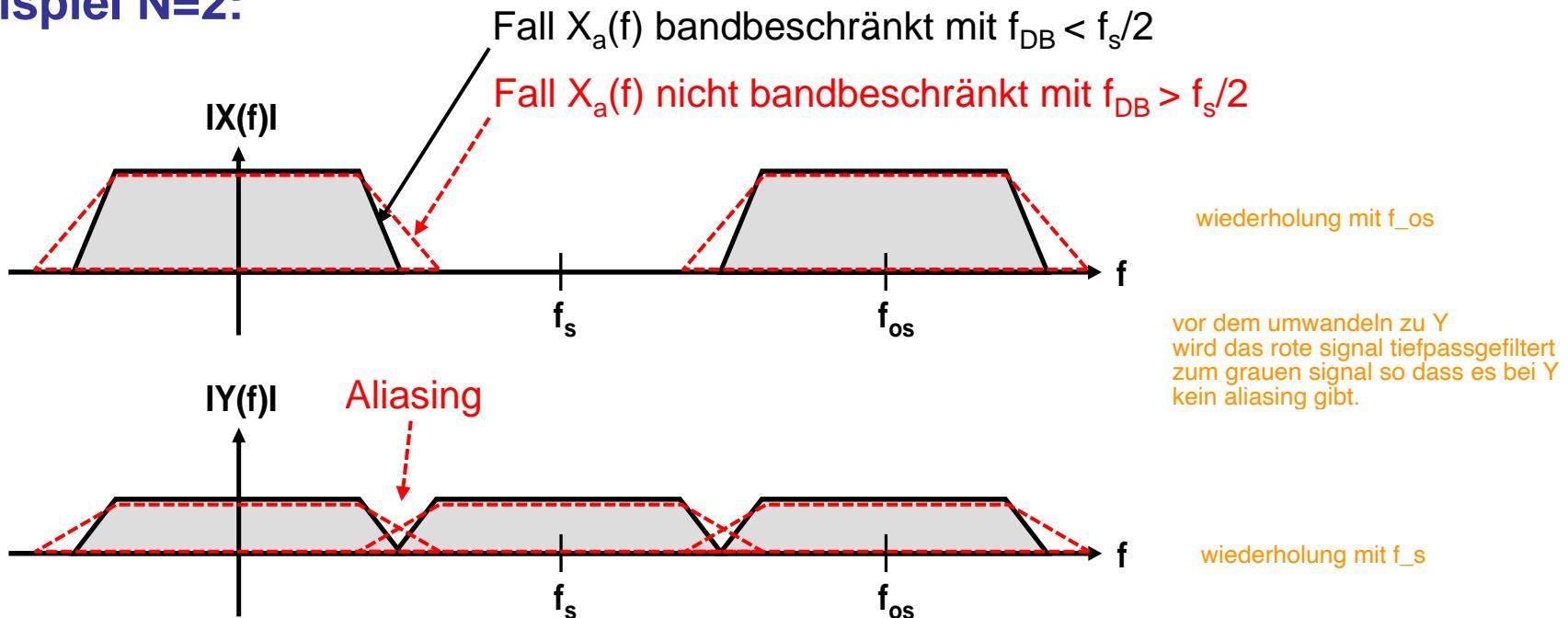
Spektrum wird skaliert und quasi zusammengeschoben

$$X(f) = \frac{1}{T_{os}} \cdot \sum_{n=-\infty}^{\infty} X_a(f - n \cdot f_{os})$$

$$Y(f) = \frac{1}{N \cdot T_{os}} \cdot \sum_{n=-\infty}^{\infty} X_a(f - n \cdot f_{os}/N)$$

Skalierung

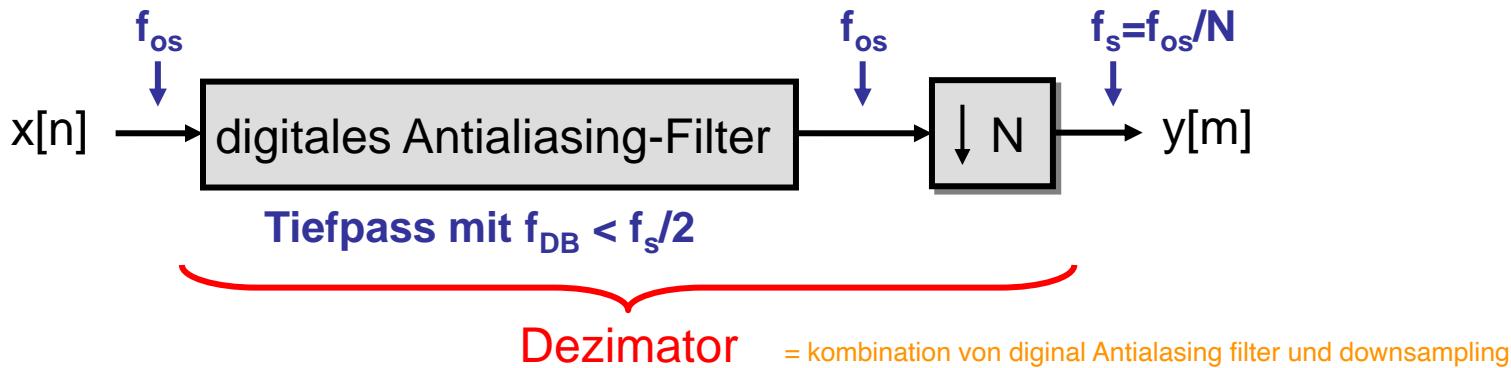
Beispiel N=2:



# Dezimation

## Kaskade digitales Anti-Aliasing-Tiefpass-Filter und Downampler

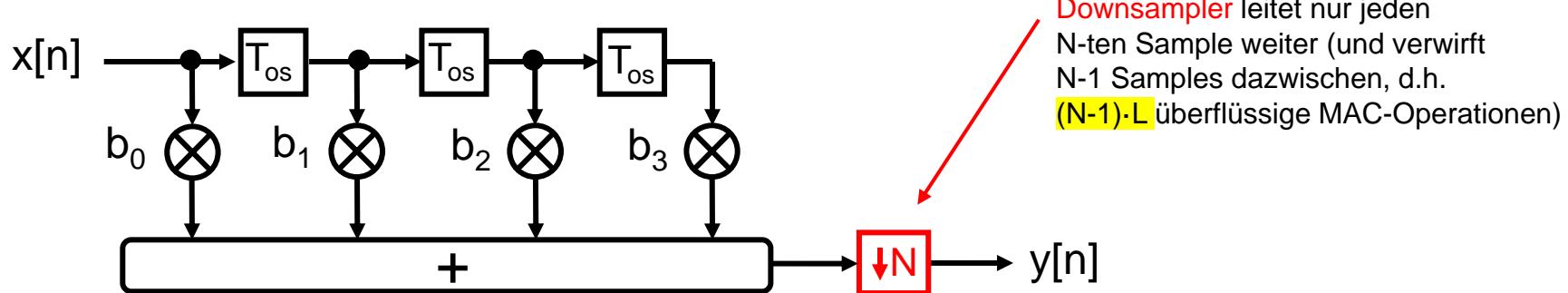
- Blockdiagramm



- Matlab: `y=decimate(x,N)`  
default ist Chebyshev-1-TP 8. Ordnung mit  $f_{DB}=0.8 \cdot (f_s/2)$   
(FIR-Filter-Option hat default TP-Filter 30. Ordnung)  
in 2 Schritten: `yos=filter(b,a,x); y=downsample(yos,N);`
- Problem: Rechenaufwand für Dezimationsfilter ist hoch (Takt ist  $f_{os}!$ )

# Dezimation: Reduktion des Rechenaufwands

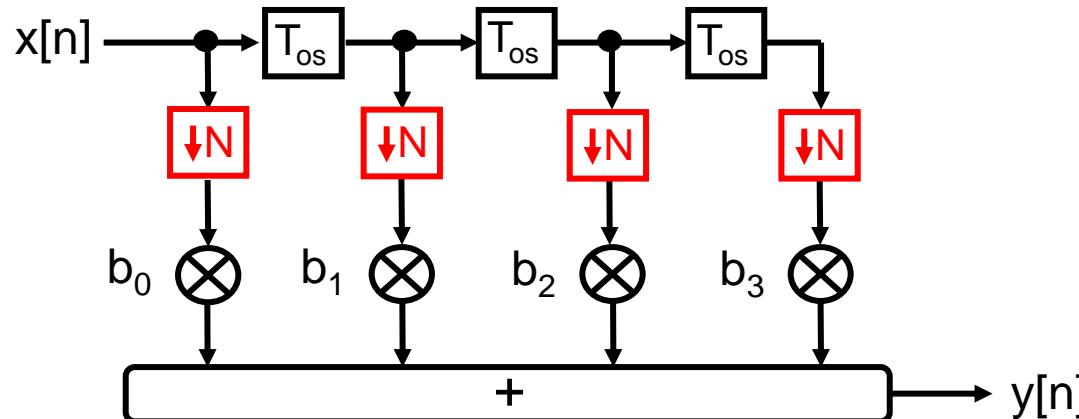
## FIR-Filter-Dezimator mit L Taps



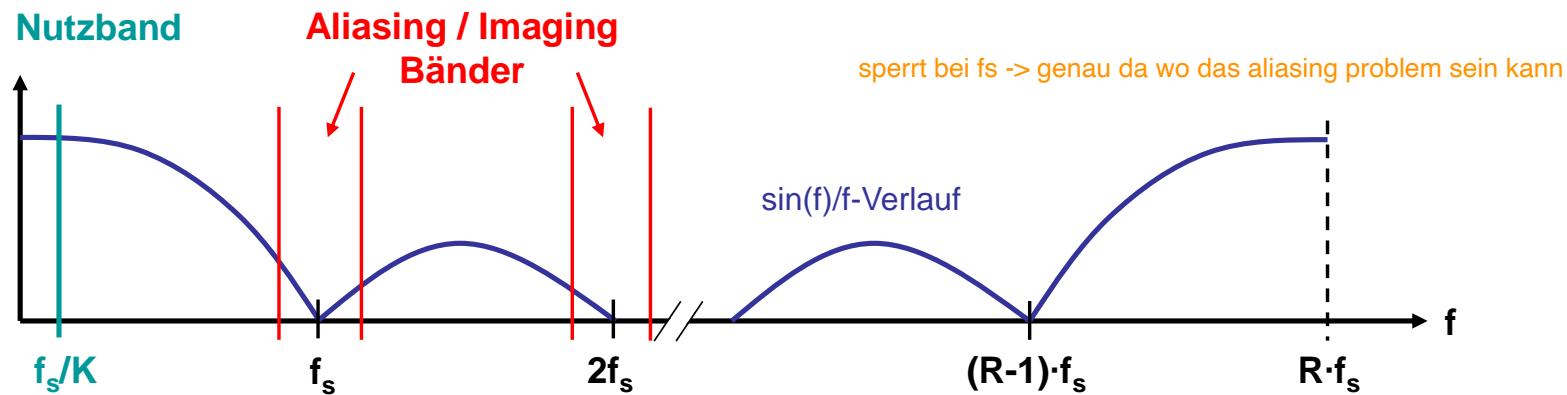
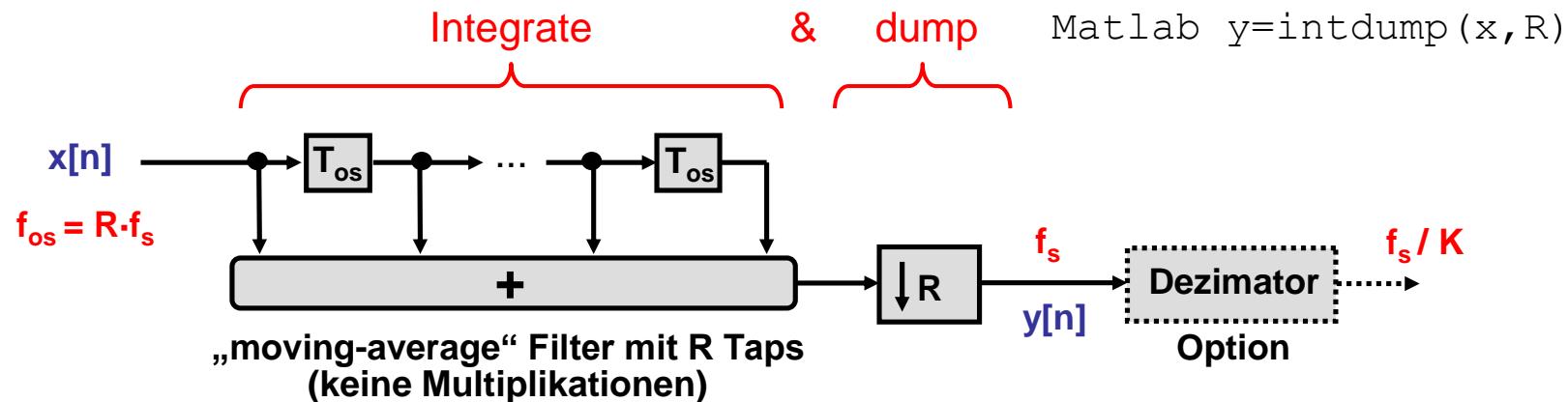
## Verschiebung des Downsamplers

N mal „schieben“, nur 1 mal rechnen

=> nur L (statt NL) MAC-Operationen pro output-Sample  $y[n]$ !



# Beispiel: Integrate & Dump Dezimator



## Berechnung



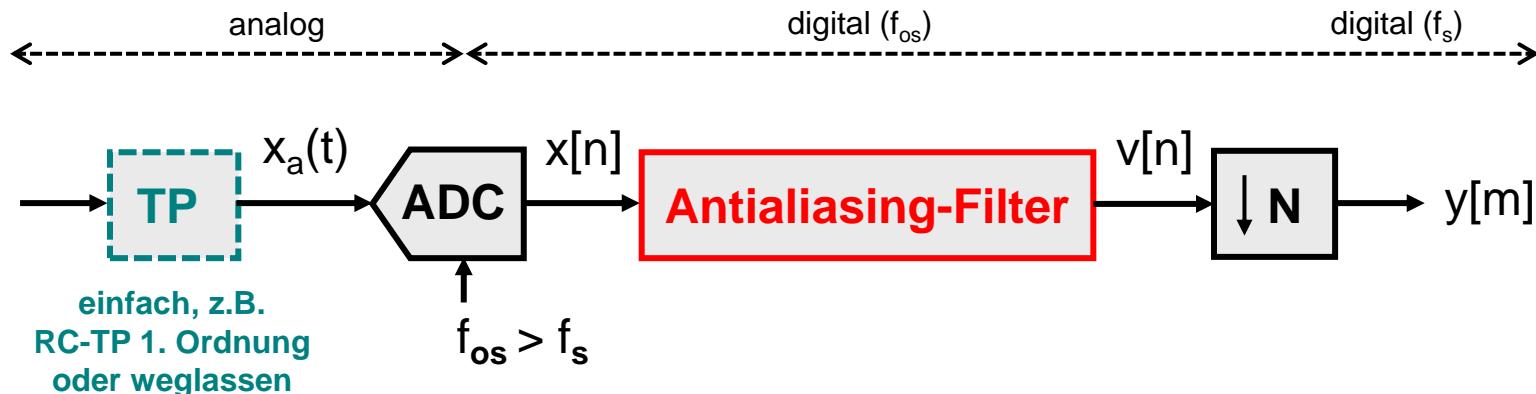
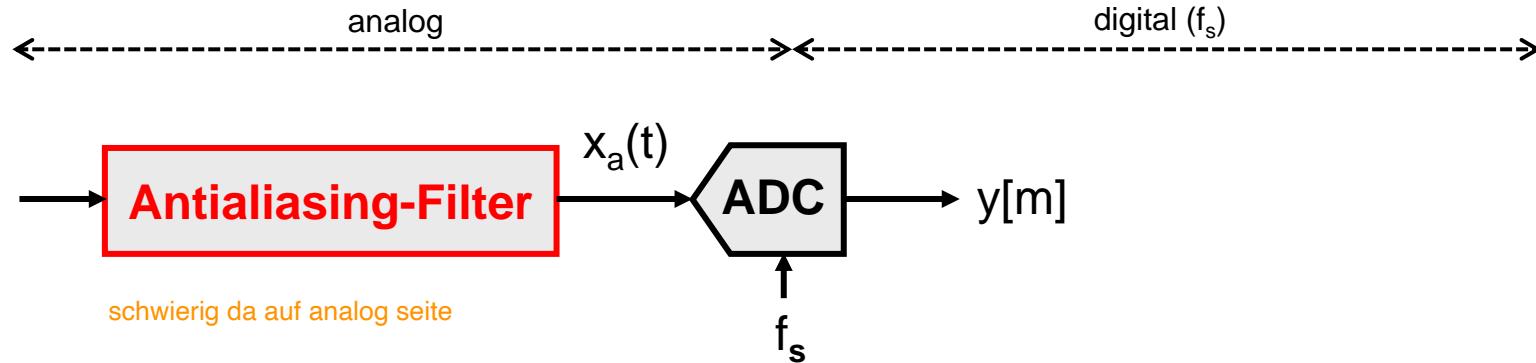
Summe oder «Mittelwert»:  $y[0]$

$y[1]$

$y[2]$

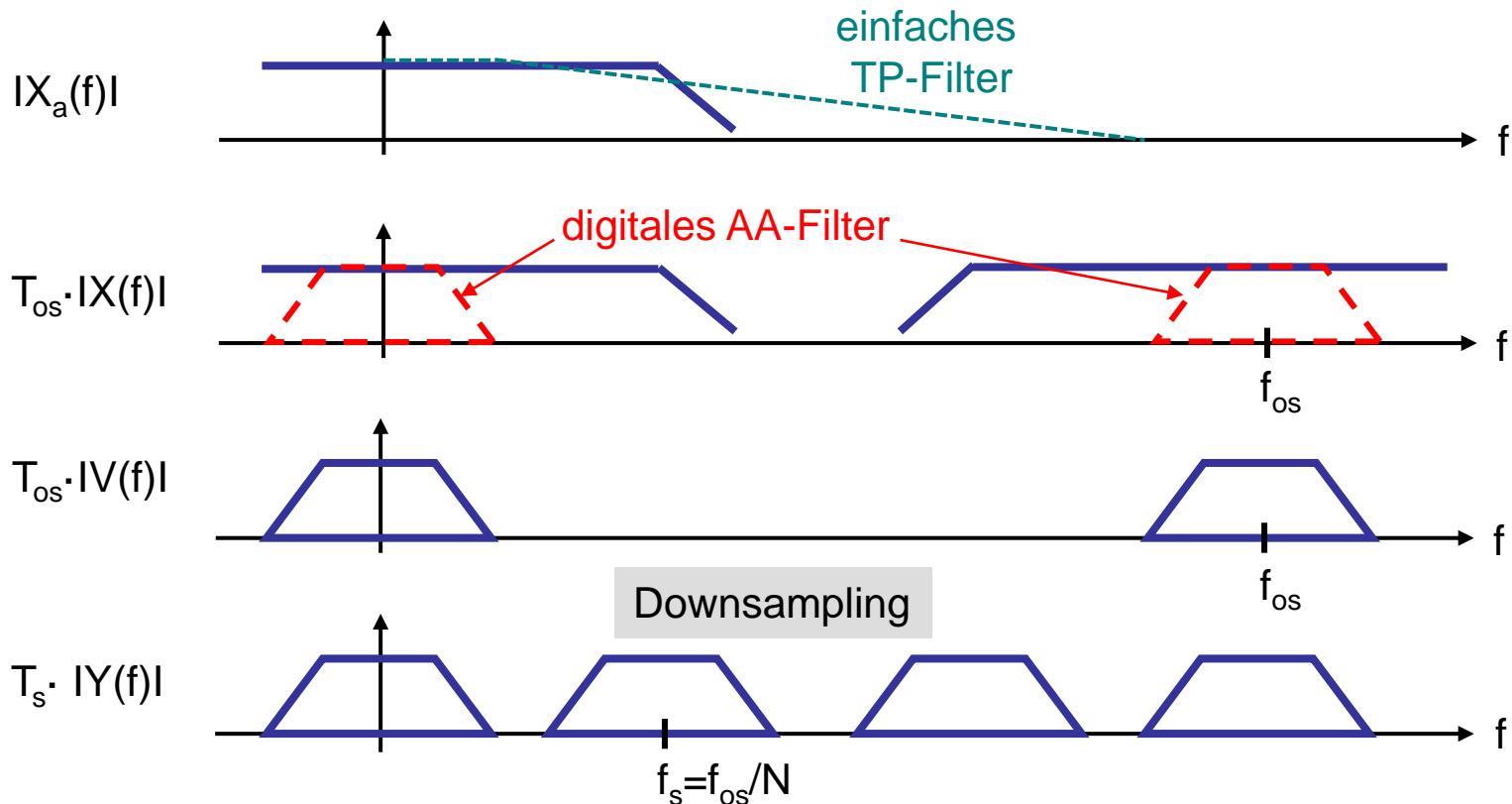
# Dezimation: Beispiel ADC mit Oversampling

Verschiebung des Anti-Aliasing-Filters vom Analogem ins Digitale



# Dezimation: Beispiel ADC mit Oversampling

## Verschiebung des AA-Filters vom Analogen ins Digitale



# Dezimation: Hörbeispiel

Matlab-Demo: dezimator.m

$f_{os} = 8192 \text{ Hz}$

„splat“  
**Ausgabe 1**  
(Original)

TP

$f_{DB} < 1048 \text{ Hz}$

$f_s = 2048 \text{ Hz}$

**Ausgabe 2**

(TP eliminiert  
Töne > 1 kHz)

↓ 4

↓ 4

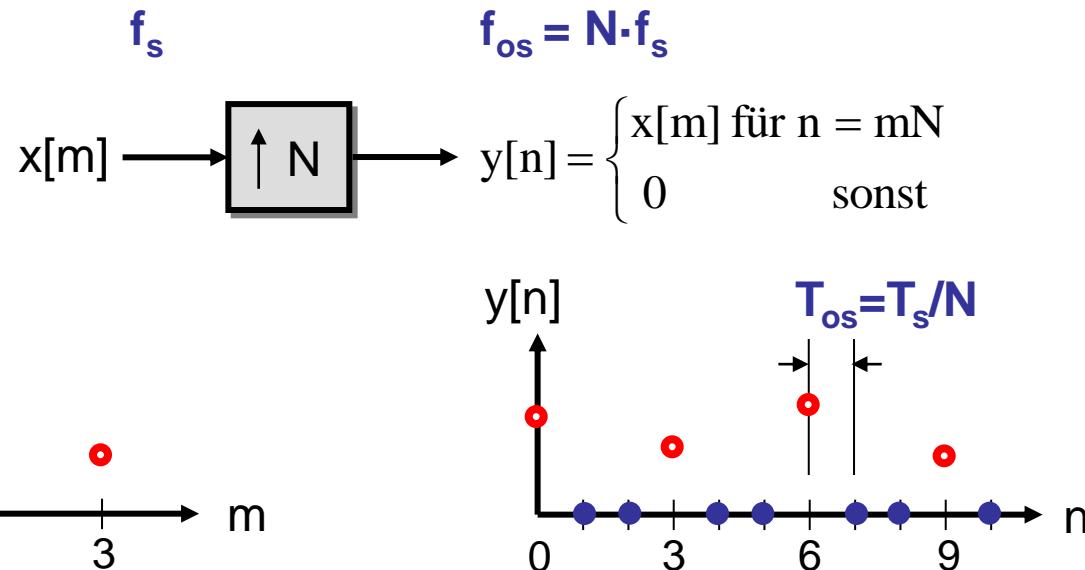
**Ausgabe 3**

(Aliasverzerrungen  
sind hörbar)

# Upsampling: Analyse im Zeitbereich

## Aufwärtstastung

- zwischen je zwei x-Samples werden N-1 Nullen eingefügt  
=> N-fache Ratenerhöhung von  $f_s$  auf  $f_{os}$
- Blockdiagramm und Beispiel mit N=3



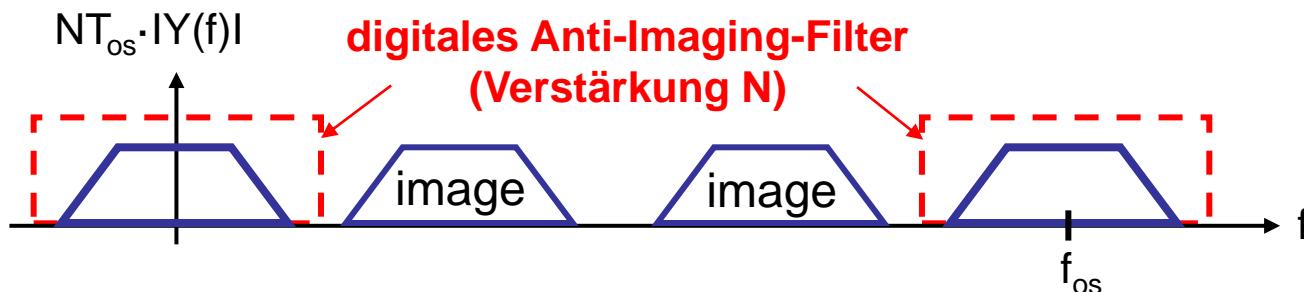
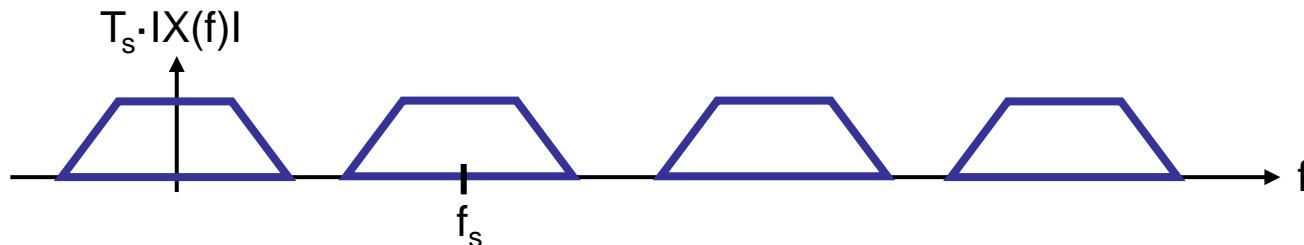
- Matlab:  $x = [1 : 4]; N = 3; y = \text{upsample}(x, N)$
- ```
1 0 0 2 0 0 3 0 0 4 0 0
```

# Upsampling: Analyse im Frequenzbereich

## Upsampling verändert das Spektrum nicht!

$$Y(f) = Y(z = e^{j2\pi f T_{os}}) = \sum_{m=-\infty}^{\infty} x[m] \cdot e^{-j2\pi f m N T_{os}} = \sum_{m=-\infty}^{\infty} x[m] \cdot e^{-j2\pi f m T_s} = X(z = e^{j2\pi f T_s}) = X(f)$$

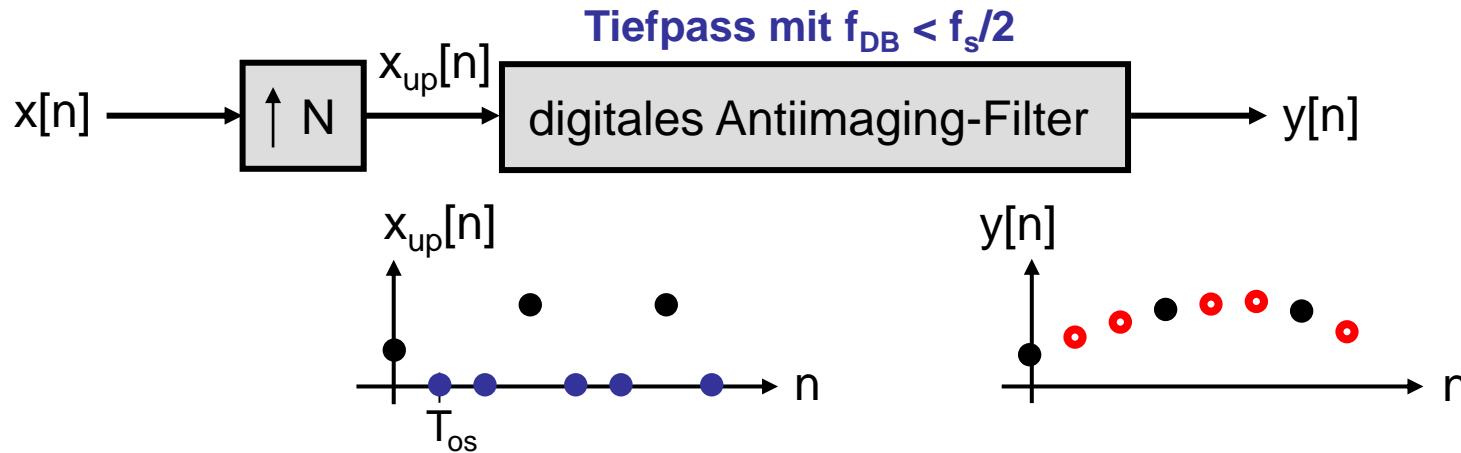
- aber es entstehen  $N-1$  images in  $Y(f)$  zwischen 0 und  $f_{os}$   
=> digitales Anti-Imaging-Filter (Interpolationsfilter) erforderlich
- Beispiel mit  $N=3$ :



# Interpolation

## Kaskade Upsampler und digitales Anti-Imaging-Filter

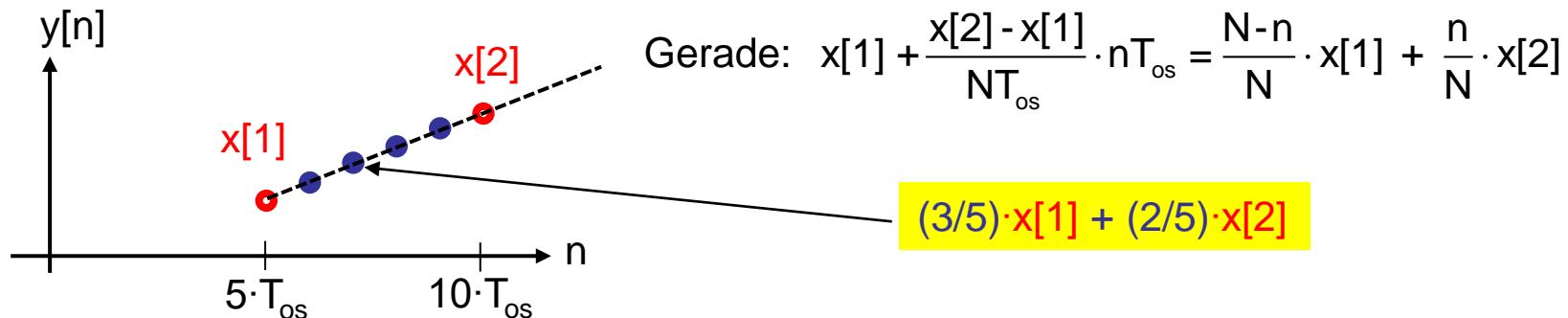
- Blockdiagramm und Beispiel mit  $N=3$



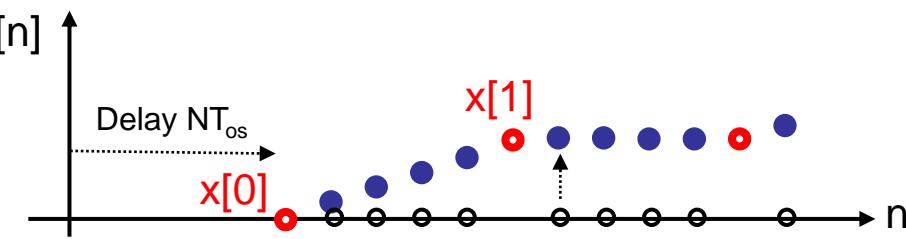
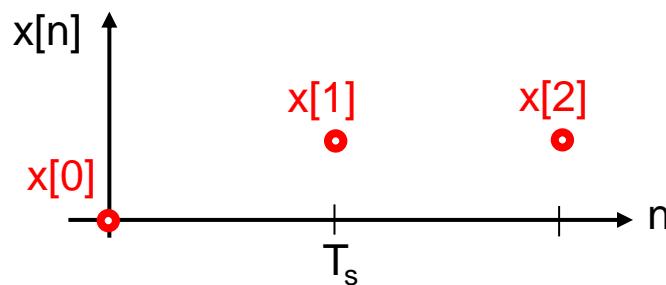
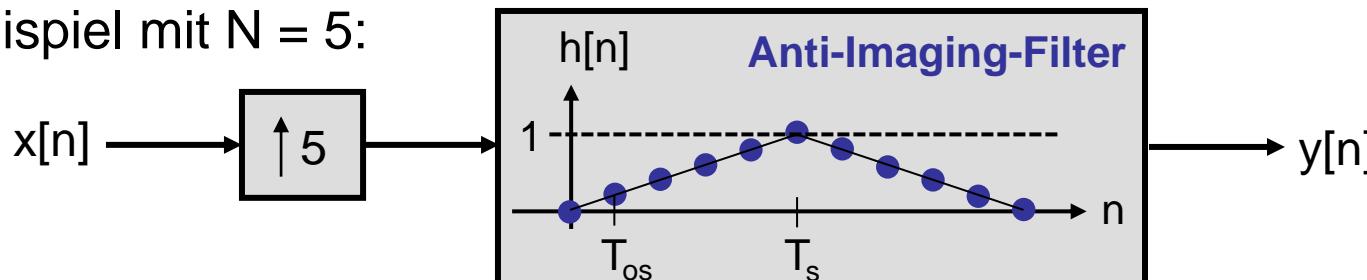
- Anti-Imaging-Filter ist ein Tiefpass-Filter mit  $f_{DB} < f_s/2$   
TP lässt nur langsame Änderungen durch => interpoliert Nullen!
- Matlab: `interp(x, N)` Interpolation im mean-square Sinn  
in 2 Schritten `xup = upsample(x, N); y = N*filter(b, a, xup)`  
amplitude muss mit  $N$  multipliziert werden

# Lineare Interpolation

Manchmal genügt eine einfache lineare Interpolation statt einer Anti-Imaging-Filterung, Beispiel mit N=5:



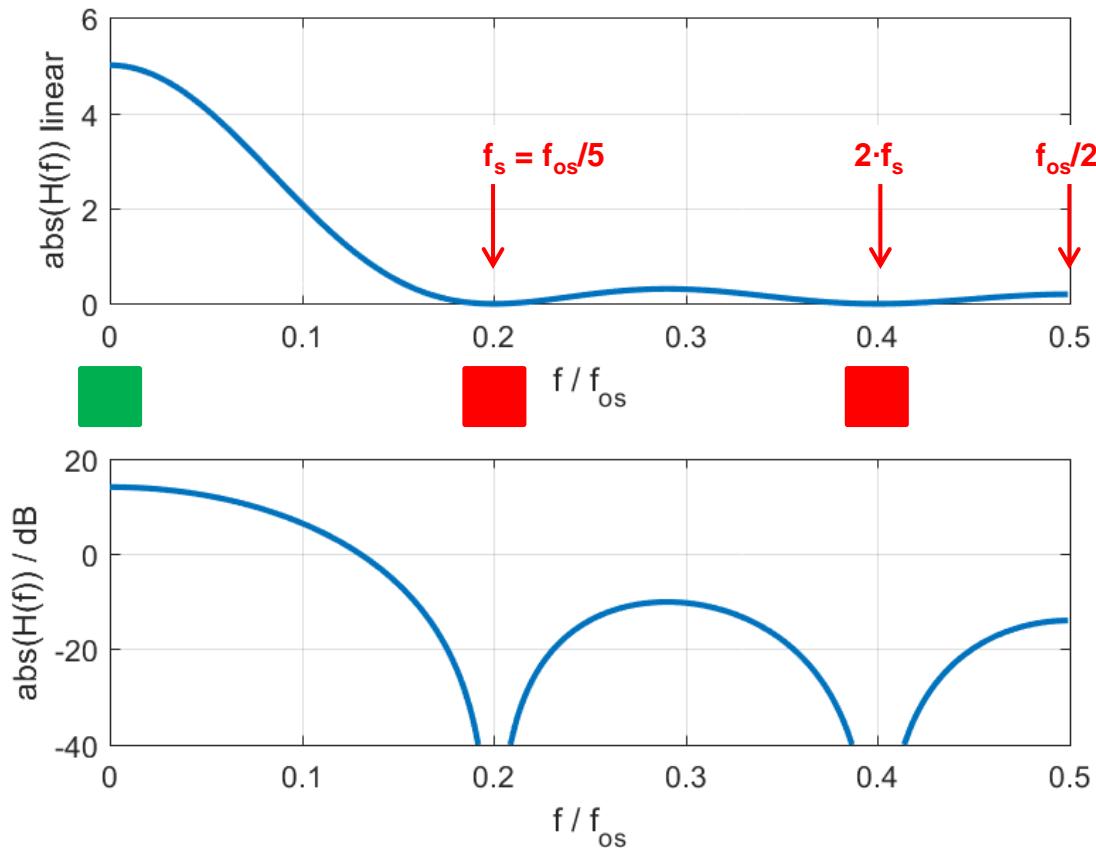
Interpolations-Filter hat  $\Delta$ -förmige Impulsantwort:  $h = [[0:N]/N [N-1:-1:0]/N]$ ;  
Beispiel mit  $N = 5$ :



# Lineare Interpolation: Analyse im f-Bereich

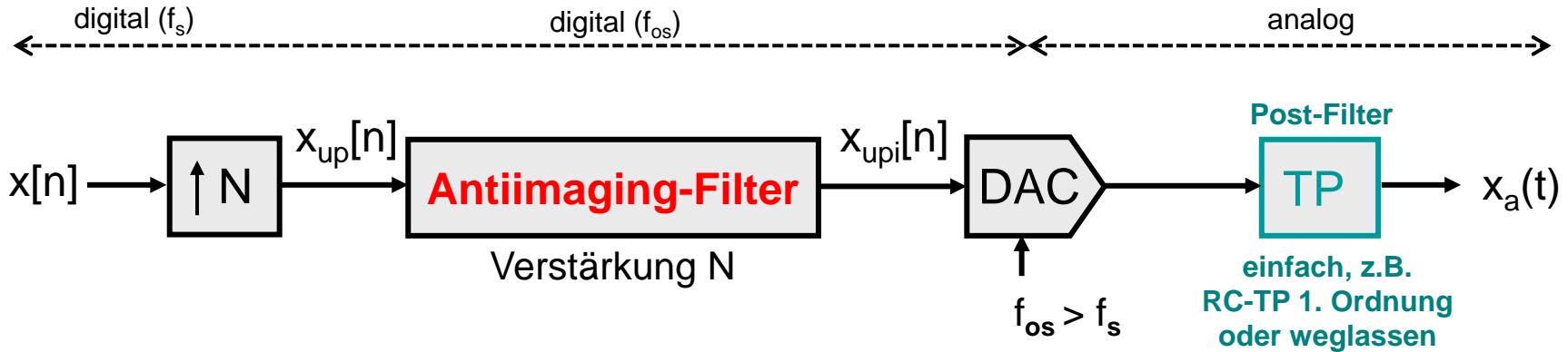
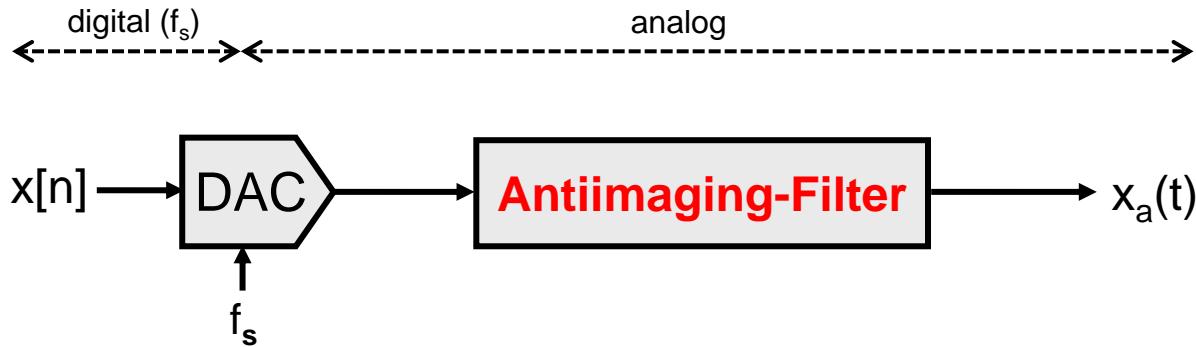
- Interpolationsfilter  $H(f) = [ \sin(\pi f/f_s) / \sin(\pi f/f_{os}) ]^2 / N$   $\stackrel{=}{\text{sinc}^2}$   
hat (fast) sinc<sup>2</sup>-Verlauf und Nullstellen bei  $n \cdot f_s$ ,  $n \neq mN$
- linearer Interpolator ist nur gut, wenn die Bandbreite  $X(f) \ll f_s/2$ ,  
d.h.  $x[n]$  langsam veränderlich ist
- Beispiel mit  $N=5$ :

 Nutzband  
 Images



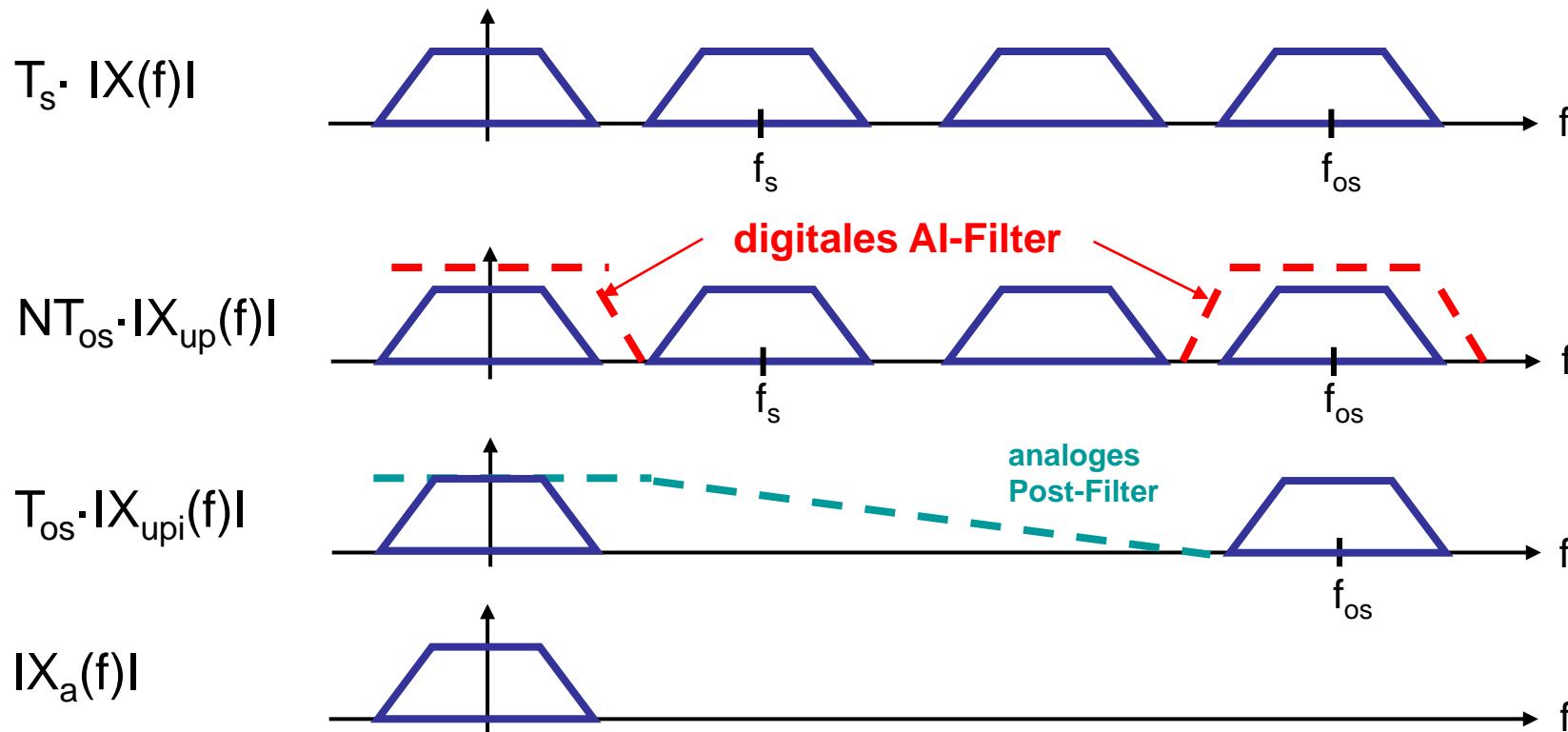
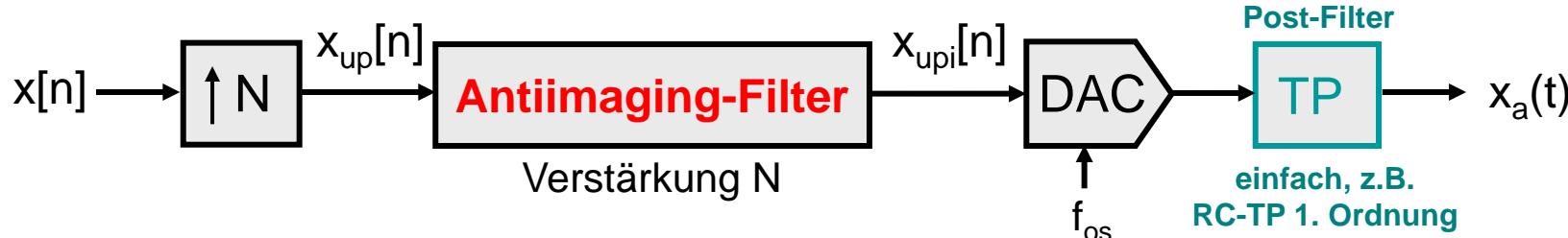
# Interpolation: Beispiel DAC mit Oversampling

Verschiebung des Anti-Imaging-Filters vom Analogen ins Digitale



# Interpolation: Beispiel DAC mit Oversampling

## Verschiebung des Anti-Imaging-Filters vom Analogem ins Digitale



# Interpolation: Hörbeispiel

Matlab-Demo: interpolator.m

$f_s = 8192 \text{ Hz}$

„gong“  
**Ausgabe 1**  
(Original)

$f_{os} = 32768 \text{ Hz}$

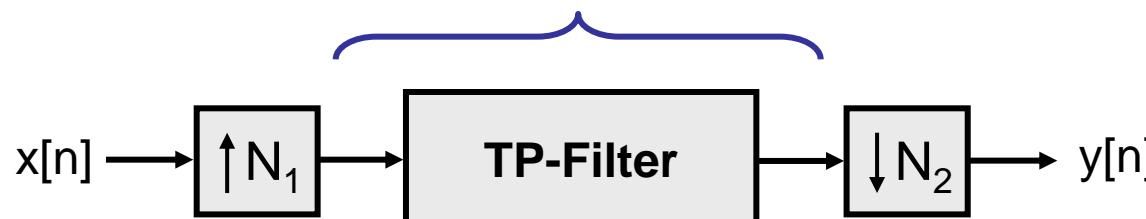
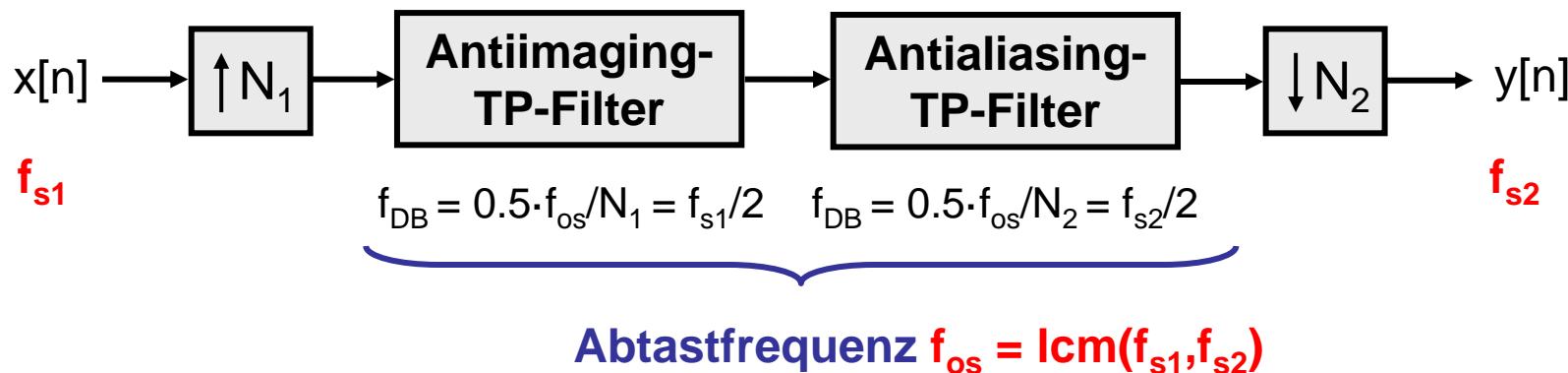
**Ausgabe 2**  
(Images  
sind hörbar)

$\uparrow 4$   
**TP**  
 $f_{DB} < 4096 \text{ Hz}$

**Ausgabe 3**  
(TP eliminiert  
Töne > 4 kHz)

# Rationale Ratenverhältnisse $N_1 / N_2 = f_{s2} / f_{s1}$

- zuerst **Interpolation** mit  $N_1$  und anschliessend **Dezimation** mit  $N_2$
- Anti-Imaging und Anti-Aliasing-Filter können zusammengefasst werden
- Eckfrequenz  $f_{DB} = \min\{ 0.5 \cdot f_{os}/N_1, 0.5 \cdot f_{os}/N_2 \} = \min\{ f_{s1}/2, f_{s2}/2 \}$



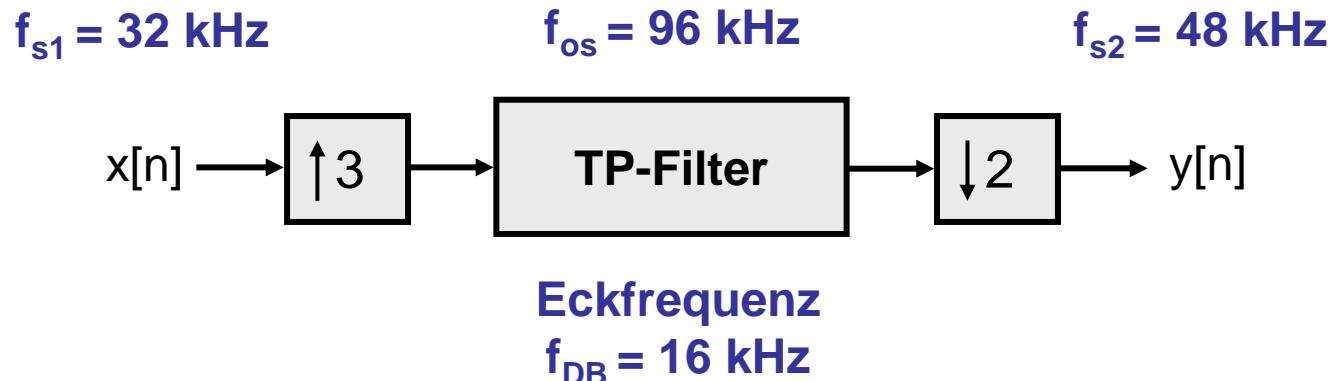
Typische Prüfungsfrage: Zeichnen und beschriften der Boxen

Lösung «zuerst Dezimation und dann Interpolation» macht keinen Sinn weil  
 a) bei der Dezimation i.A. Signalkomponenten weggefertigt werden müssen und  
 b) die beiden Tiefpass-Filter dann nicht zusammengelegt werden können.

# Rationale Ratenverhältnisse: Beispiel

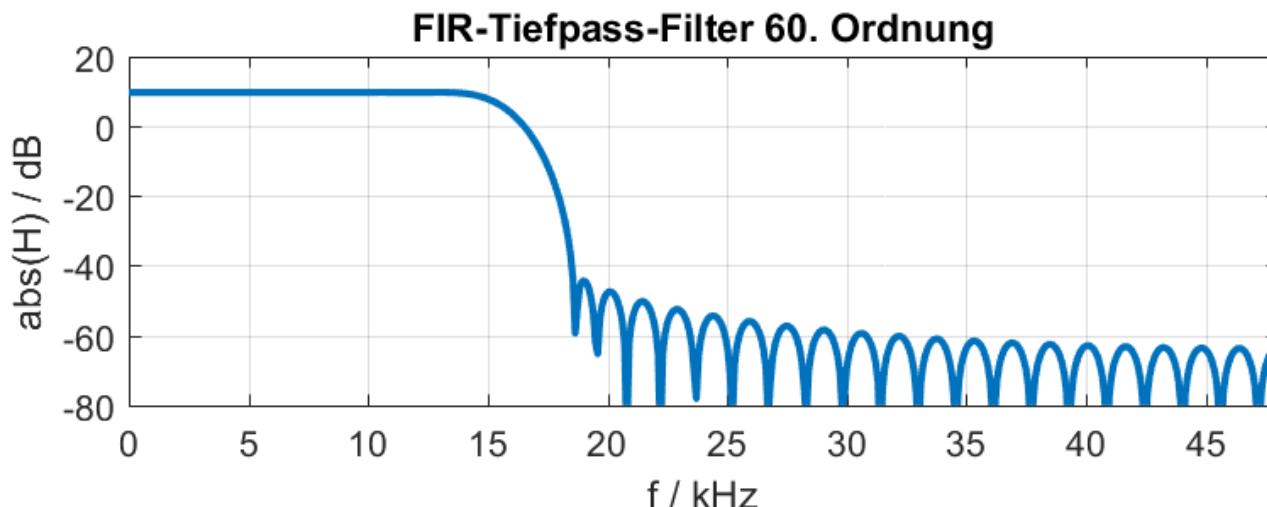
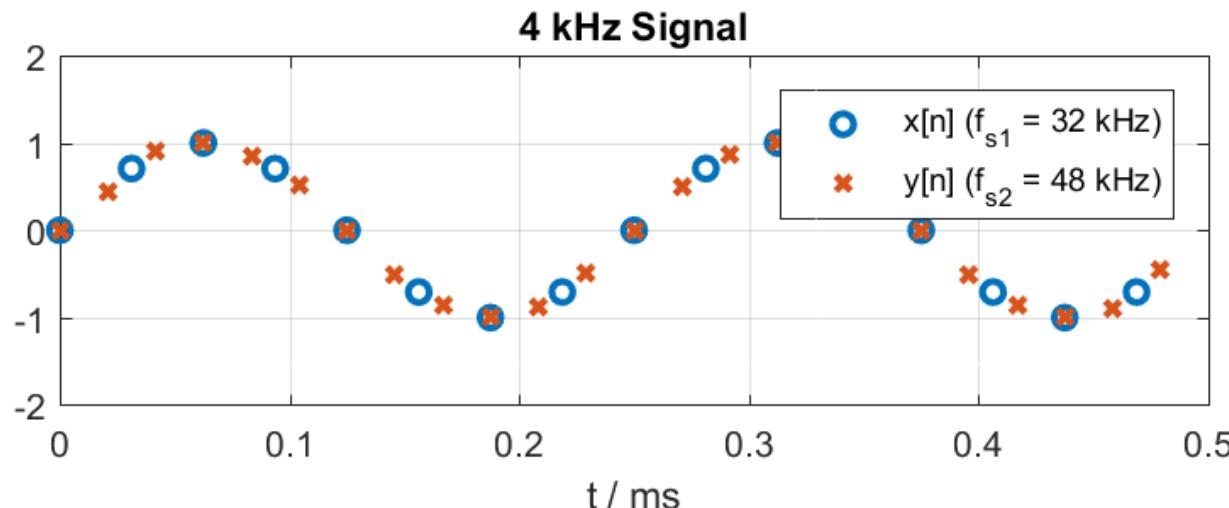
## Beispiel Ratenänderung von $f_{s1} = 32 \text{ kHz}$ auf $f_{s2} = 48 \text{ kHz}$

- Rationales Ratenverhältnis  $f_{s2} / f_{s1} = N_1 / N_2 = 3 / 2$
- Interpolation mit  $N_1 = 3$  auf  $f_{os} = 96 \text{ kHz}$ , Dezimation mit  $N_2 = 2$
- Eckfrequenz Tiefpass-Filter  $f_{DB} = \min\{ f_{s1}/2, f_{s2}/2 \} = 16 \text{ kHz}$
- Skalierung mit  $N_1 = 3$



- Matlab: `y=resample(x,N1,N2)` oder `upfirdn()`

# Rationale Ratenverhältnisse: Beispiel



```
N1=3; N2=2; [y,h]=resample(x,N1,N2);
```

# Sigma-Delta-Wandler ( $\Sigma\Delta$ -Wandler)

Walt Kester, "Data Conversion Handbook", Kapitel 3.3, Seiten 3.114 - 3.121, Analog Devices bzw. <https://www.analog.com/en/education/education-library/data-conversion-handbook.html>.

- ◆ **Low Cost, High Resolution (to 24-bits)**
- ◆ **Excellent DNL (Differential Non-Linearity)**
- ◆ **Low Power, but Limited Bandwidth (Voiceband, Audio)** → *(heute auch schneller, z.B. AD7760: 2.5 MSPS, 24-Bit, 100 dB Dynamic Range)*
- ◆ **Key Concepts are Simple, but Math is Complex**
  - **Oversampling**
  - **Quantization Noise Shaping**
  - **Digital Filtering**
  - **Decimation**
- ◆ **Ideal for Sensor Signal Conditioning**
  - **High Resolution**
  - **Self, System, and Auto Calibration Modes**
- ◆ **Wide Applications in Voiceband and Audio Signal Processing**

# Quantisierung (zur Erinnerung)

---

## Quantisierungsstufe bzw. 1 LSB bei einem W-Bit ADC

- $q = A / 2^W$  wobei A den Aussteuerbereich bezeichnet

## Quantisierungsrauschen

- RMS-Rauschwert:  $N_{rms} = q / \sqrt{12}$
- Rauschleistung:  $P_N = q^2/12$

## Signal-to-Noise-Ratio @ Sinus-Vollaussteuerung

- $SNR = 6.02 \cdot W + 1.76 \text{ dB}$

## Effective Number Of Bits

- **ENOB** =  $(SNR - 1.76 \text{ dB}) / 6.02 \text{ dB}$
- Je kleiner die Rauschleistung, desto grösser die Bitauflösung

# Quantisierung (zur Erinnerung)

## Quantisierungsstufe bzw. 1 LSB bei einem W-Bit ADC

- $q = A / 2^W$

## Quantisierungsrauschen

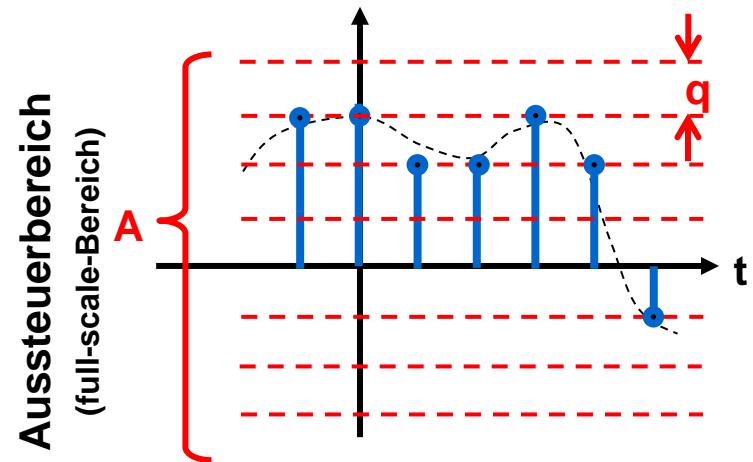
- RMS-Rauschwert:  $N_{\text{rms}} = q / \sqrt{12}$
- Rauschleistung:  $P_N = q^2/12$

## Signal-to-Noise-Ratio für Sinus-Vollaussteuerung

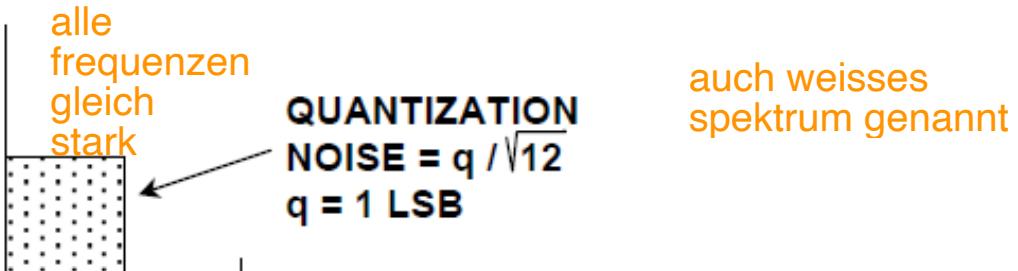
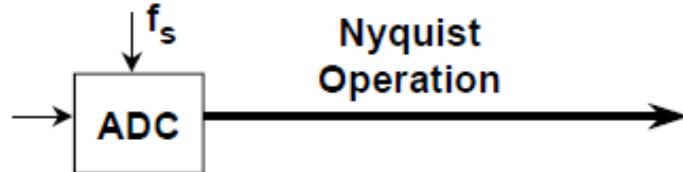
- $\text{SNR} = 6.02 \cdot W + 1.76 \text{ dB}$

## Effective Number Of Bits

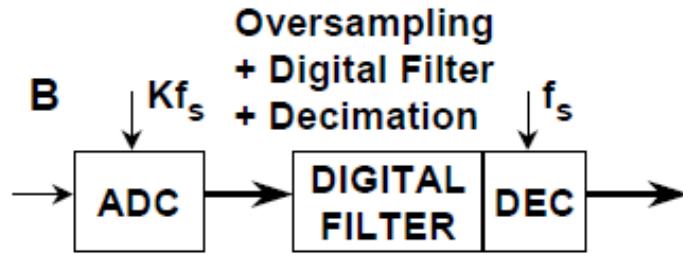
- **ENOB** =  $(\text{SNR} - 1.76 \text{ dB}) / 6.02 \text{ dB}$
- Je grösser das SNR bzw. kleiner die Rauschleistung,  
desto höher ist die Bitauflösung



# Funktionsprinzip Sigma-Delta ADC

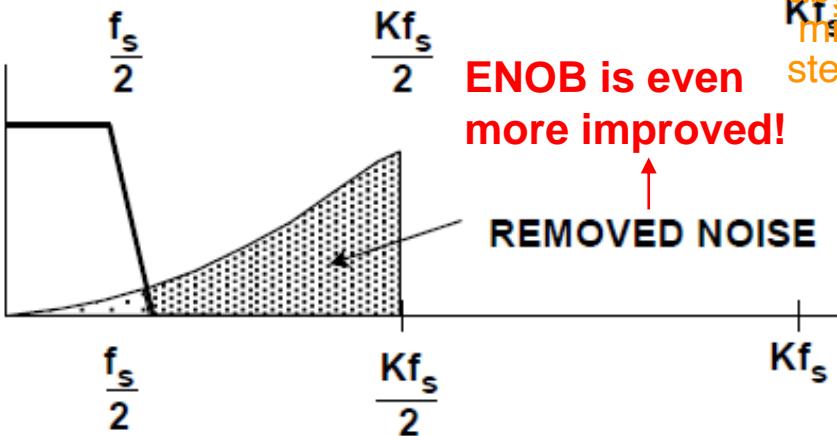
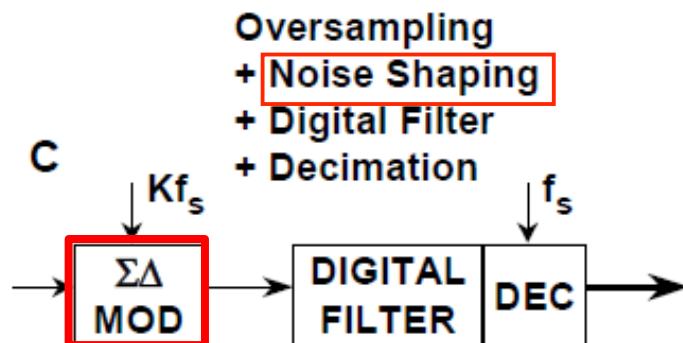
**A**

auch weisses spektrum genannt

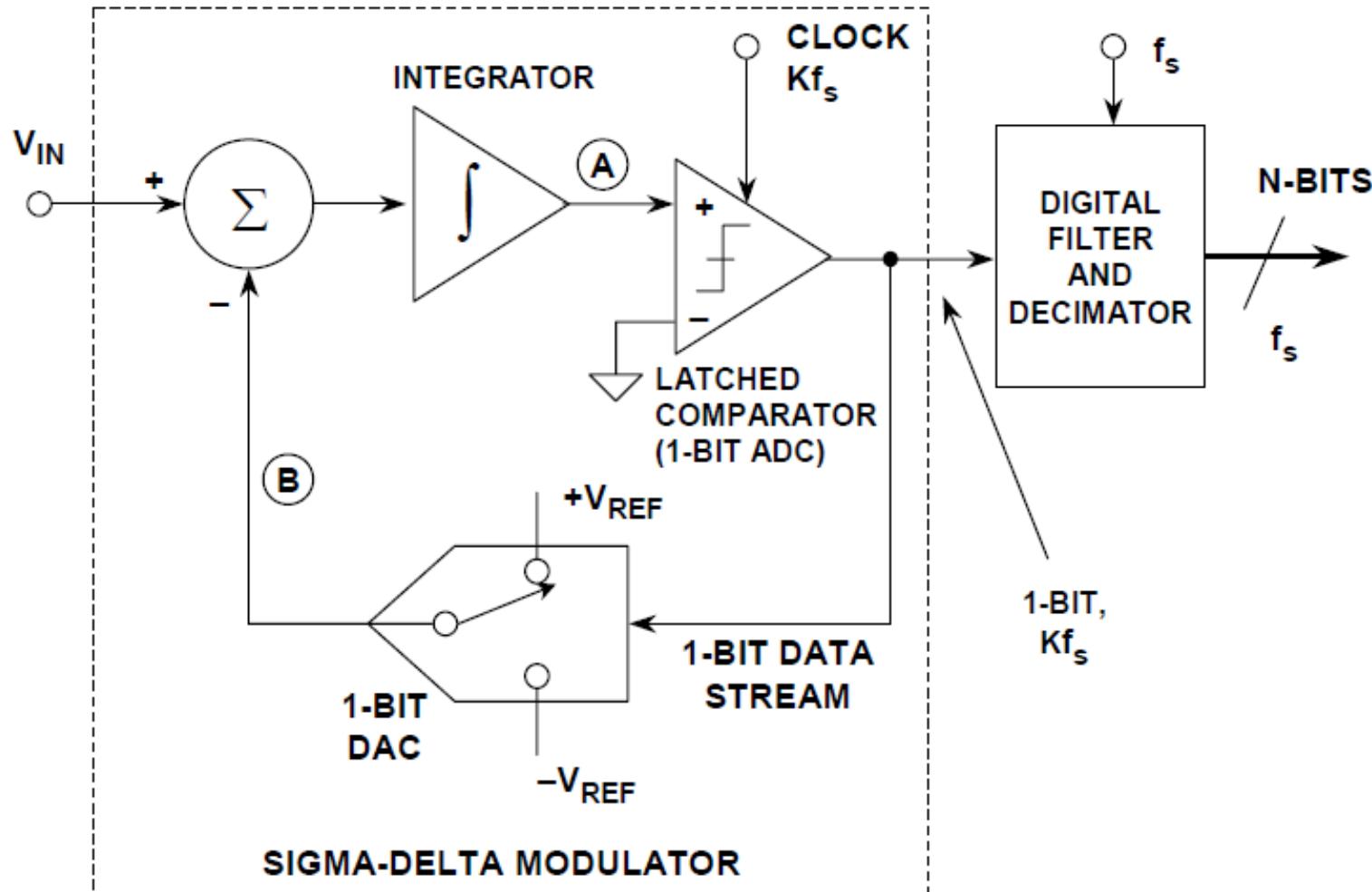
**B**

oversampling with  $K = 2^N$   
=> N Bit more ENOB!

rauschen ist auf mehr frequenzen verteilt somit weniger hoch .. und wird abgeschnitten mit TP => SNR steigt

**C**

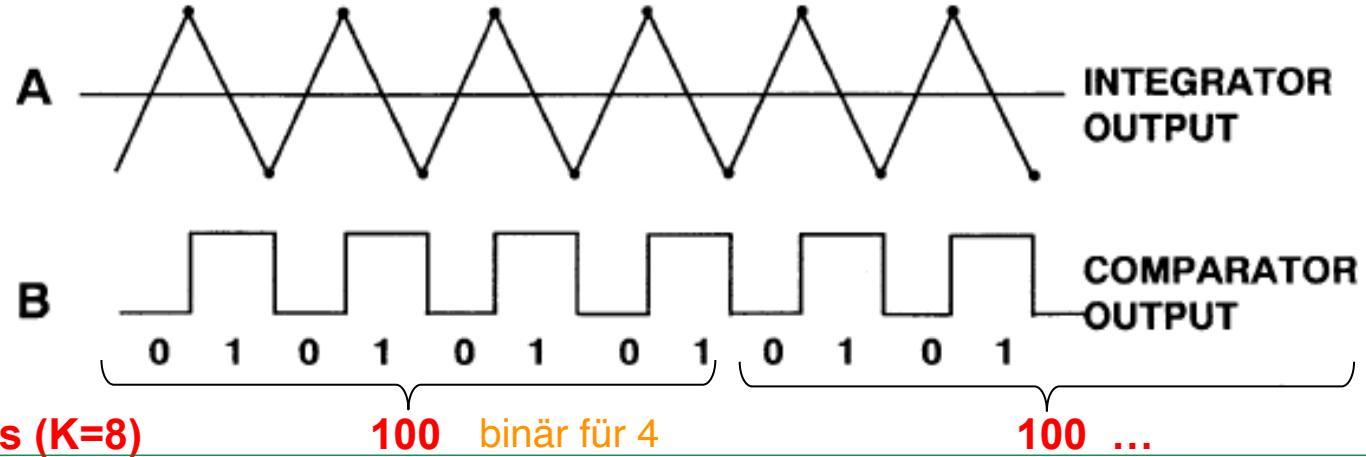
# Blockdiagramm Sigma-Delta ADC 1. Ordnung



# Wellenformen Sigma-Delta-Modulator

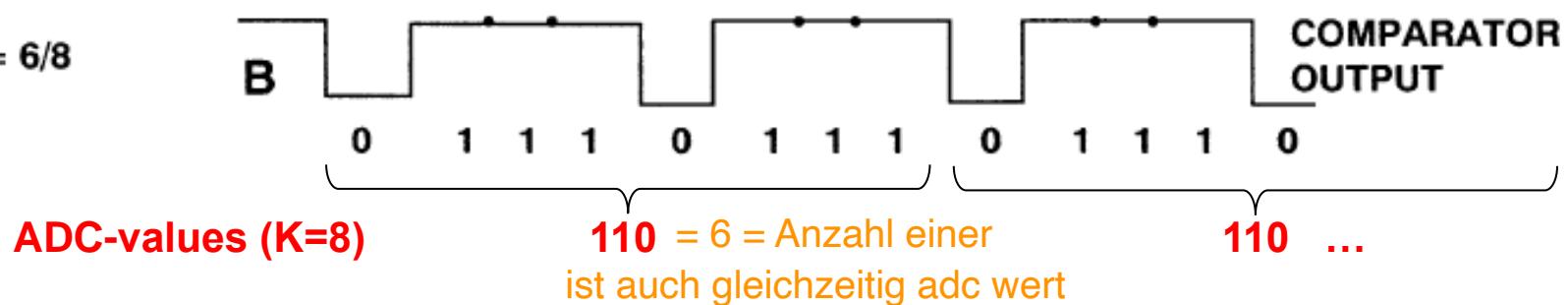
Beispiel für 0

$$\begin{aligned} V_{IN} &= 0V \text{ DC} \\ &= 2/4 \\ &= 4/8 \end{aligned}$$

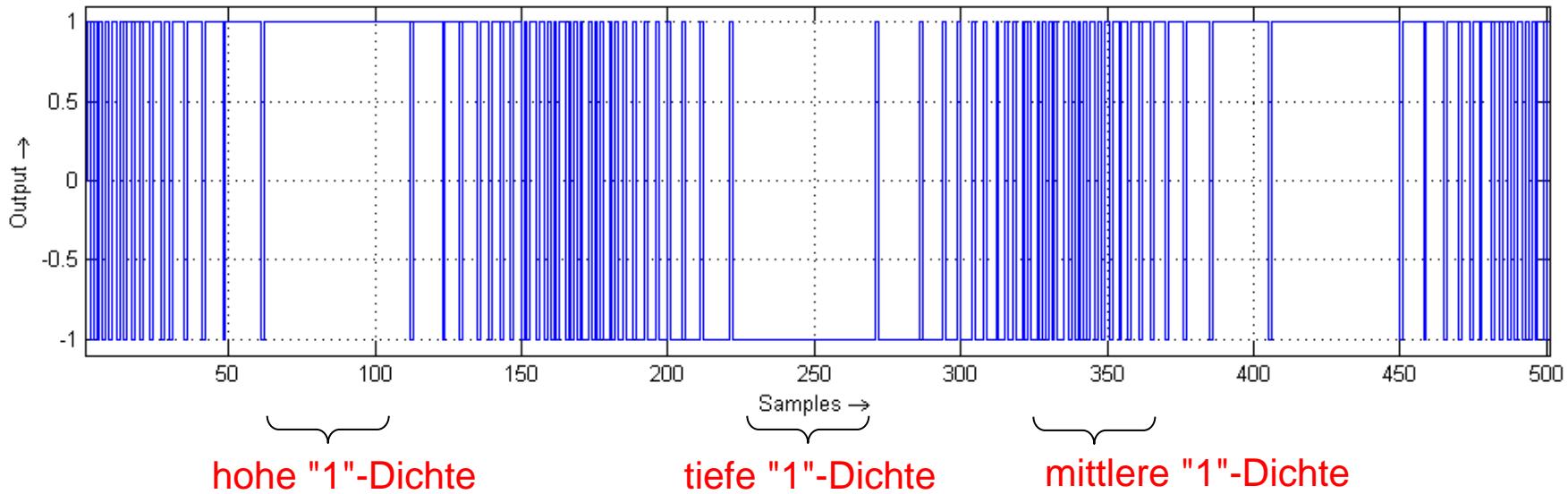
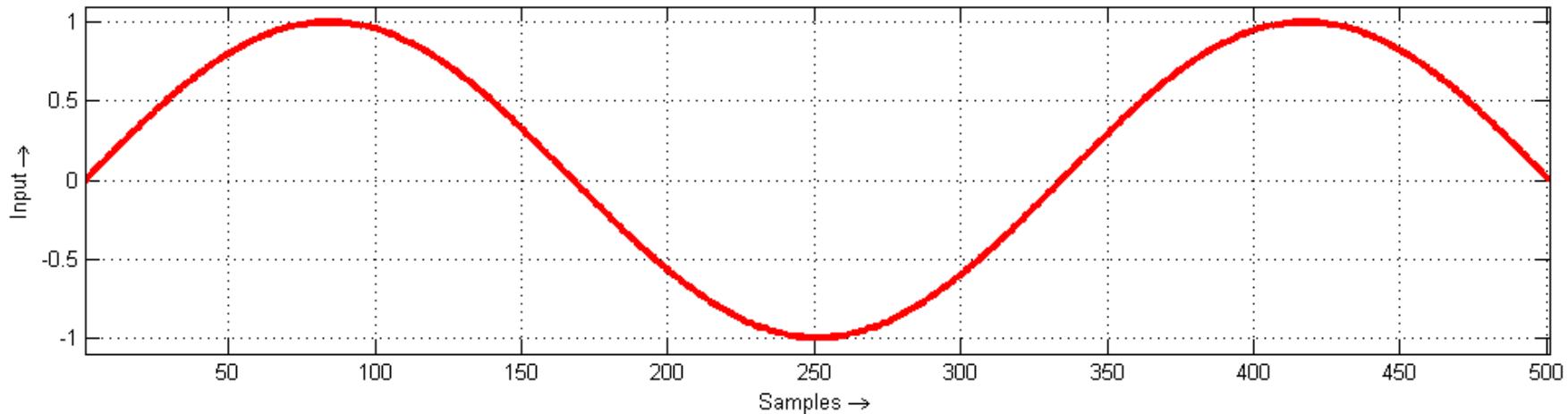


Beispiel für  $V_{ref}/2$

$$\begin{aligned} V_{IN} &= +\frac{V_{ref}}{2} \text{ DC} \\ &= 3/4 \\ &= 6/8 \end{aligned}$$



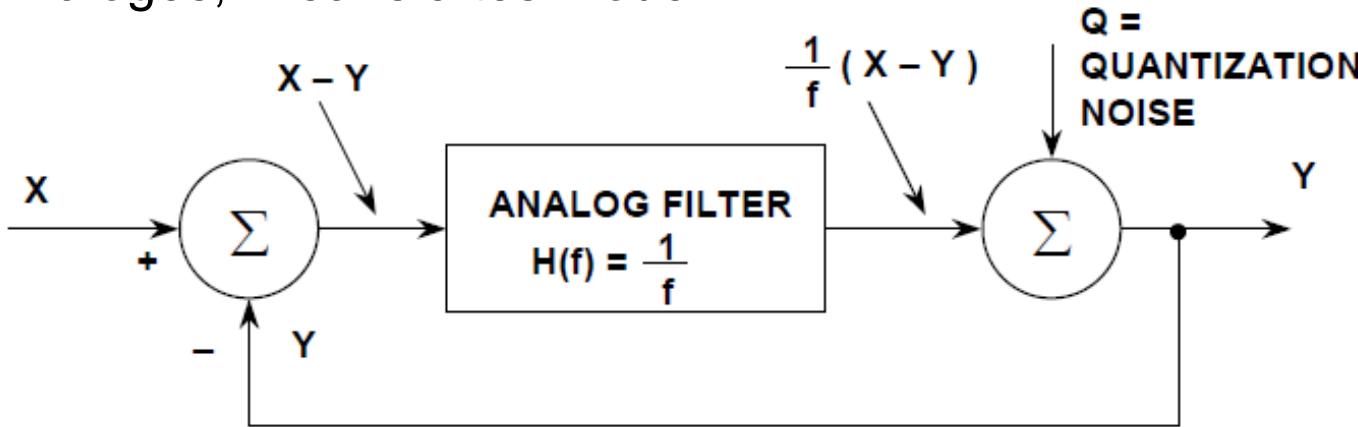
# Sigma-Delta-Modulation eines Sinus-Signals



# Sigma-Delta-ADC: Analyse im Frequenzbereich

Walt Kester, "Data Conversion Handbook", Analog Devices.

Analoges, linearisiertes Modell:



$$Y = \frac{1}{f} (X - Y) + Q$$

REARRANGING, SOLVING FOR Y:

$$Y = \frac{X}{f+1} + \frac{Qf}{f+1}$$

**SIGNAL TERM**

Tiefpassfilterung

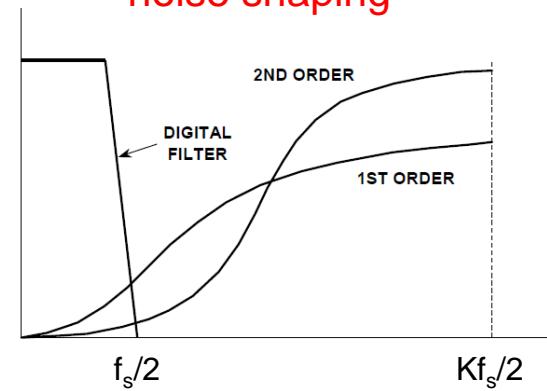
1. Ordnung von  $X(f)$

**NOISE TERM**

Hochpassfilterung

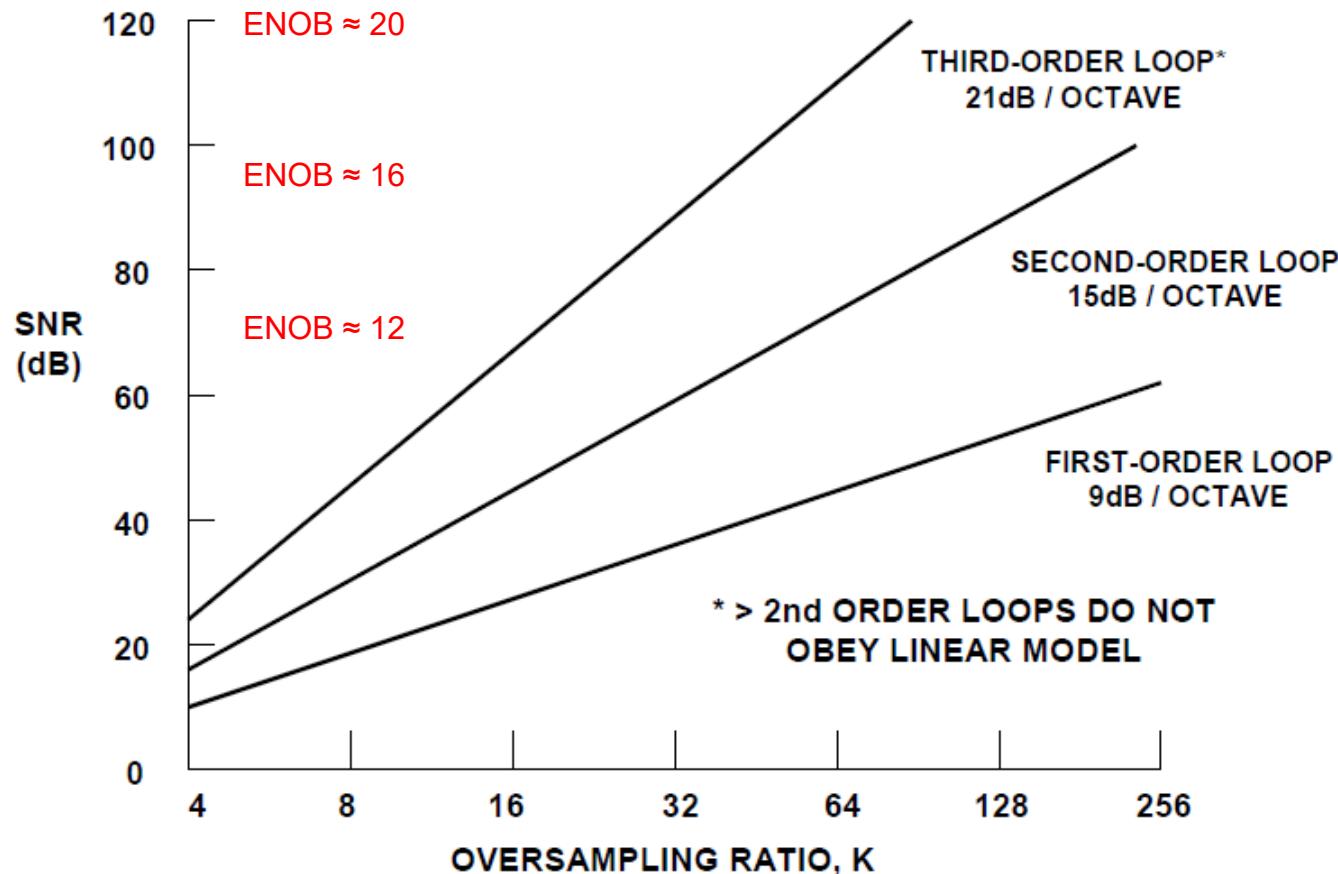
1. Ordnung von  $Q(f)$

noise shaping



# Sigma-Delta-ADC: Performance

Walt Kester, "Data Conversion Handbook", Analog Devices.

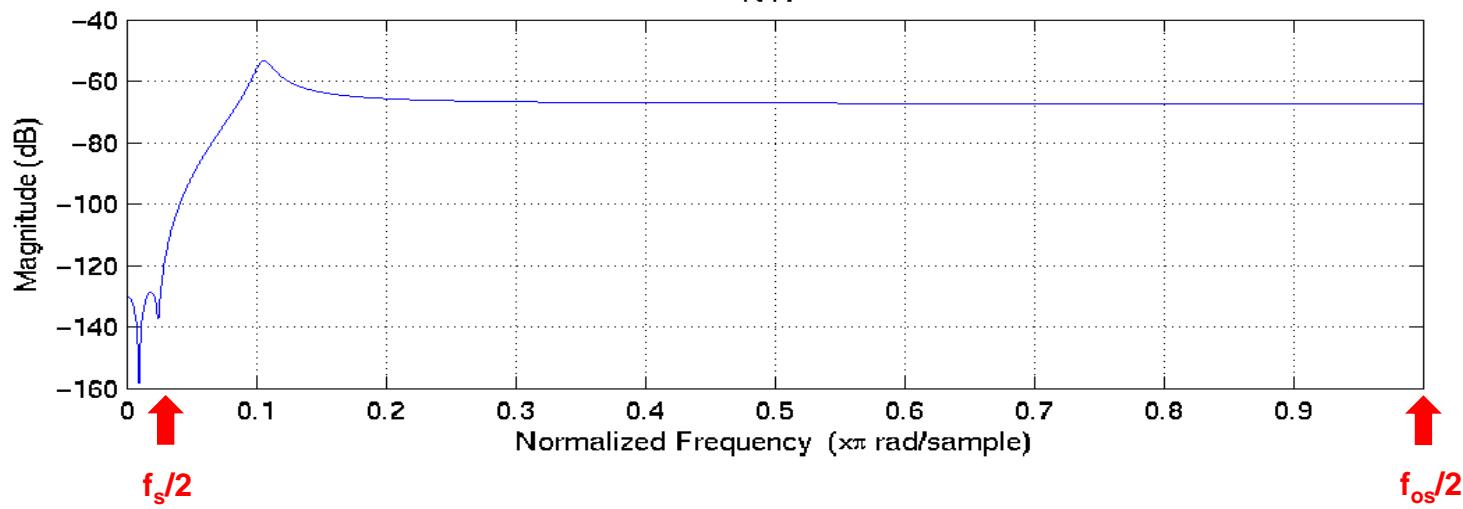
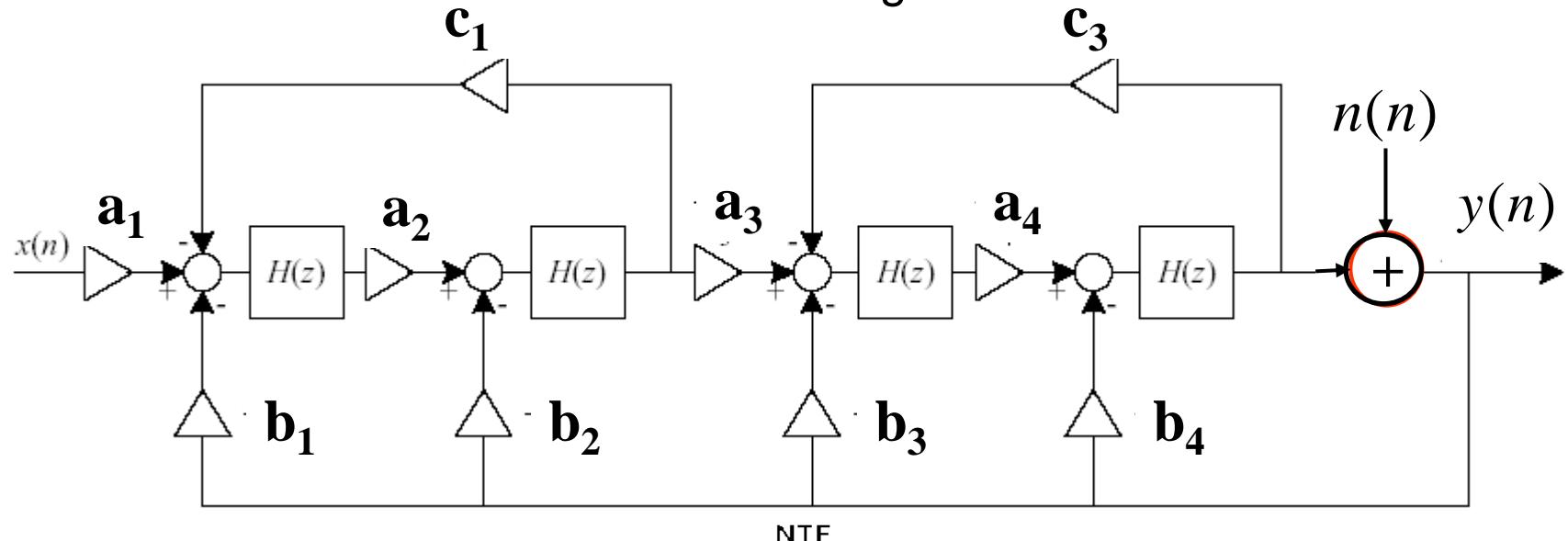


**Beispiel AD1871:** 24-Bit, 96 kSps Sigma-Delta-ADC, 105 dB dynamic range, second Order Multi-Bit Converter, 128-/64-faches Oversampling

# SNR-Performance Hörgeräte $\Sigma\Delta$ -ADC

Quelle: Dr. S. Wyrsch

Struktur eines  $\Sigma\Delta$  -Modulators 4. Ordnung



# Polyphasen-Filter: Einführung

## Beispiel: Einfacher Interpolator (N=2 upsampling, FIR mit L=4 Taps)

Zeitpunkt  $m=2n$  :

|        |   |          |   |     |
|--------|---|----------|---|-----|
| $x[n]$ | 0 | $x[n-1]$ | 0 | ... |
|--------|---|----------|---|-----|

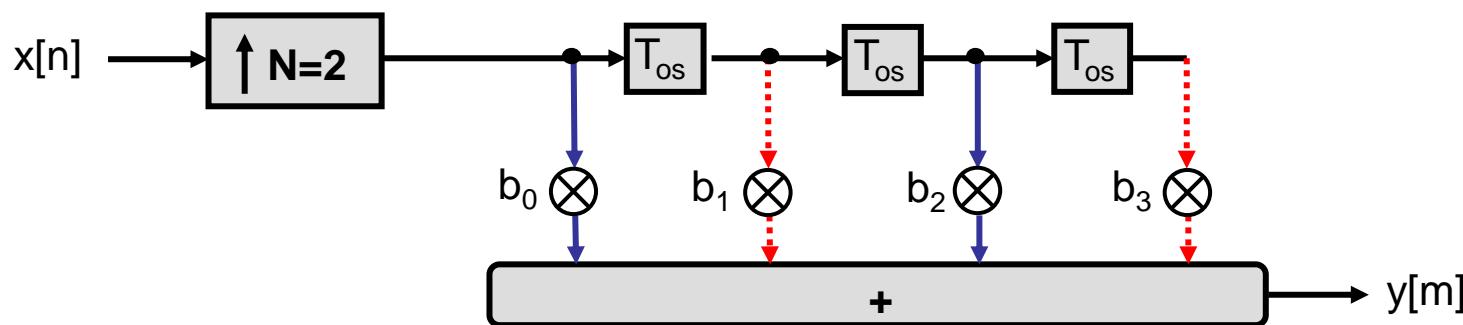
Zeitpunkt  $m=2n+1$  :

|   |        |   |          |     |
|---|--------|---|----------|-----|
| 0 | $x[n]$ | 0 | $x[n-1]$ | ... |
|---|--------|---|----------|-----|

Zeitpunkt  $m=2n+2$  :

|          |   |        |   |     |
|----------|---|--------|---|-----|
| $x[n+1]$ | 0 | $x[n]$ | 0 | ... |
|----------|---|--------|---|-----|

bei geraden zeiten  
müssen nur die blauen  
komponenten  
bei ungeraden zeiten  
nur die roten zeiten  
evaluiert werden da  
sonst alles 0 ist.

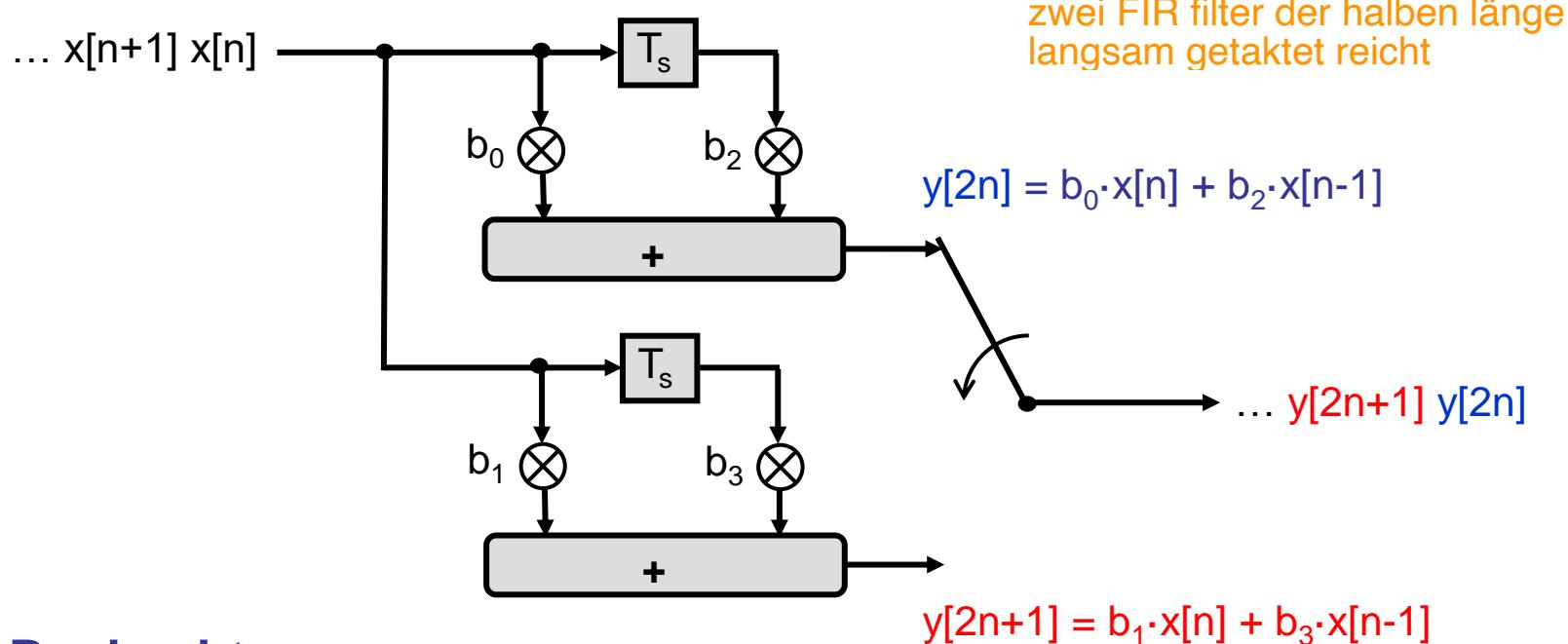


## Beobachtungen

- es befinden sich immer 2 Nullen in der Delay-Line
- 2 Phasen: blaues Teilfilter, dann rotes Teilfilter usw.
- $y[2n] = b_0 \cdot x[n] + b_2 \cdot x[n-1]$ ,  $y[2n+1] = b_1 \cdot x[n] + b_3 \cdot x[n-1]$

# Polyphasen-Filter: Einführung

## Polyphasen-Realisierung einfacher Interpolator

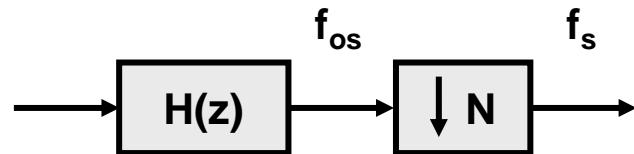


## Beobachtungen

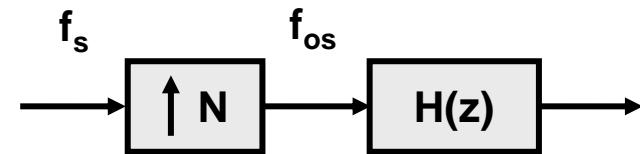
- statt 1 FIR-Filter mit  $L=4$  Taps mit  $f_{os}$  schnell getaktet  
 N=2 FIR-Filter der Länge  $L/2$  Taps mit  $f_s$  langsam getaktet
- statt  $NL = 8$  MAC-Operationen pro Eingangswert  $x[n]$   
 nur  $N \cdot L/N = L = 4$  MAC-Operationen pro Eingangswert  $x[n]$
- Delay-Lines können zusammengefasst werden

# Polyphasen-Dezimator/Interpolator

## Dezimator



## Interpolator



## Polyphasenzerlegung FIR-Filter

$z$ -UTF FIR-Filter

$$H(z) = h[0] + h[1] \cdot z^{-1} + \dots + h[L-1] \cdot z^{-(L-1)}$$

$k$ -te Polyphasenkomponente

$$H_k(z) = h[k] + h[k+N] \cdot z^{-1} + h[k+2N] \cdot z^{-2} + \dots$$

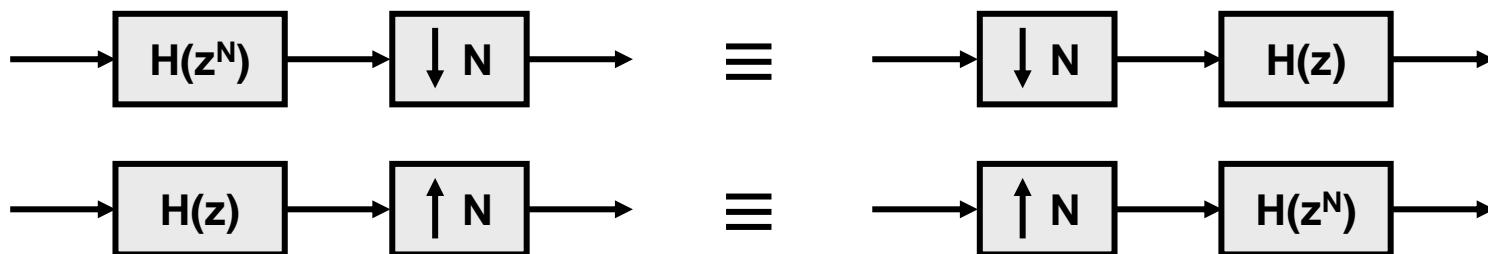
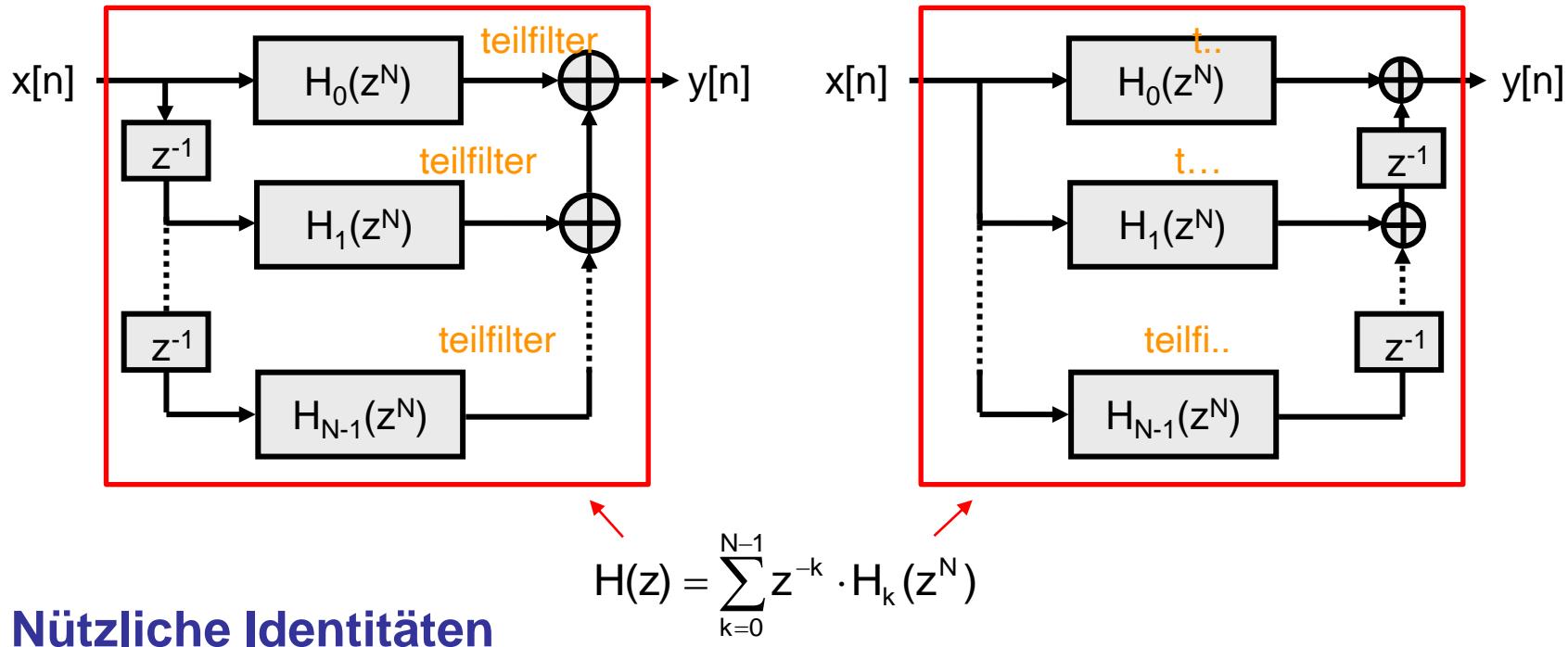
$$H_k(z^N) = h[k] + h[k+N] \cdot z^{-N} + h[k+2N] \cdot z^{-2N} + \dots$$

$N$ -fache Polyphasenzerlegung

$$H(z) = \sum_{k=0}^{N-1} z^{-k} \cdot H_k(z^N)$$

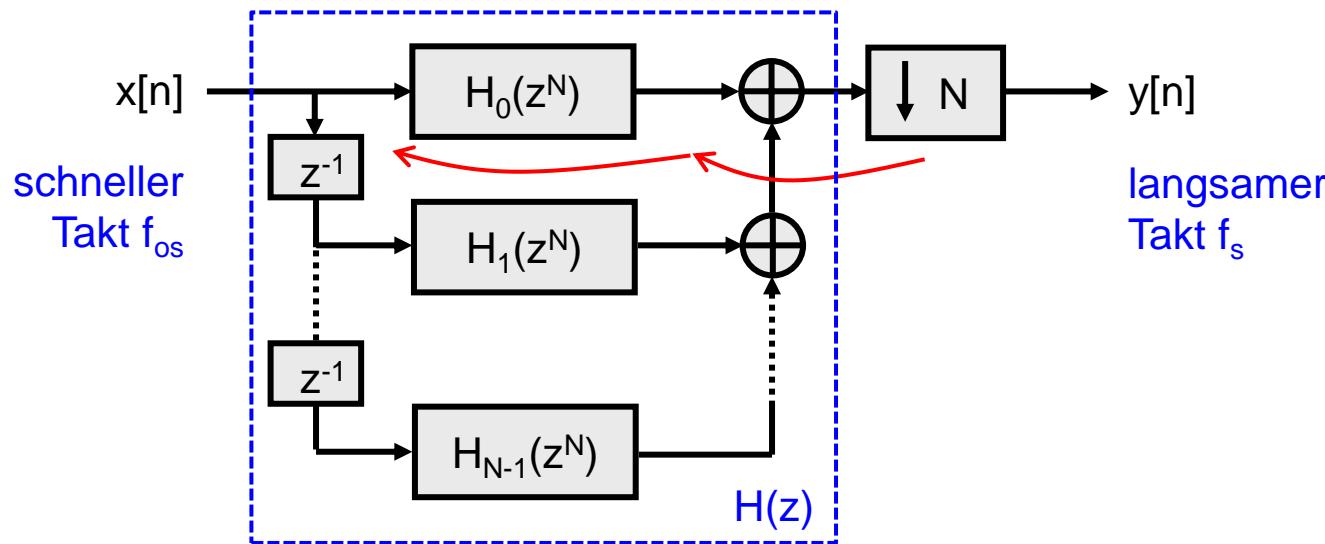
# Polyphasen-FIR-Filter

mit Verzögerungskette am Eingang bzw. am Ausgang

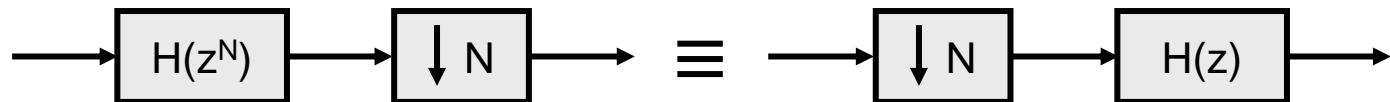


# Polyphasen-Dezimator

## Kaskade von Polyphasen FIR-Filter und Downampler

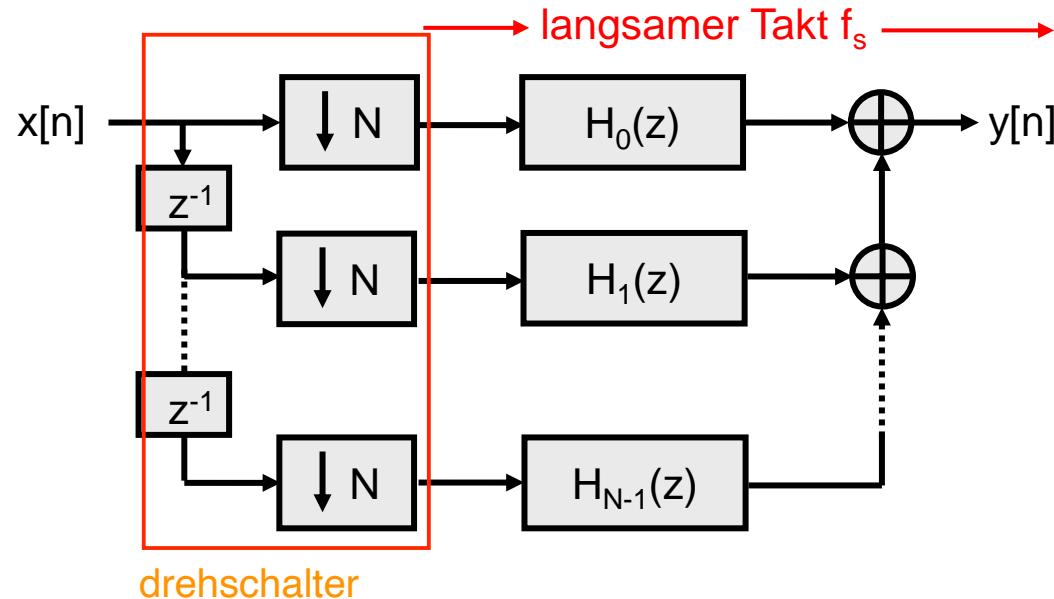


↷ Verschiebung Downampler bzw. Filter vom Gebiet mit schnellem Takt  $f_{os}$  ins Gebiet mit langsamem Takt  $f_s$  dank der Identität

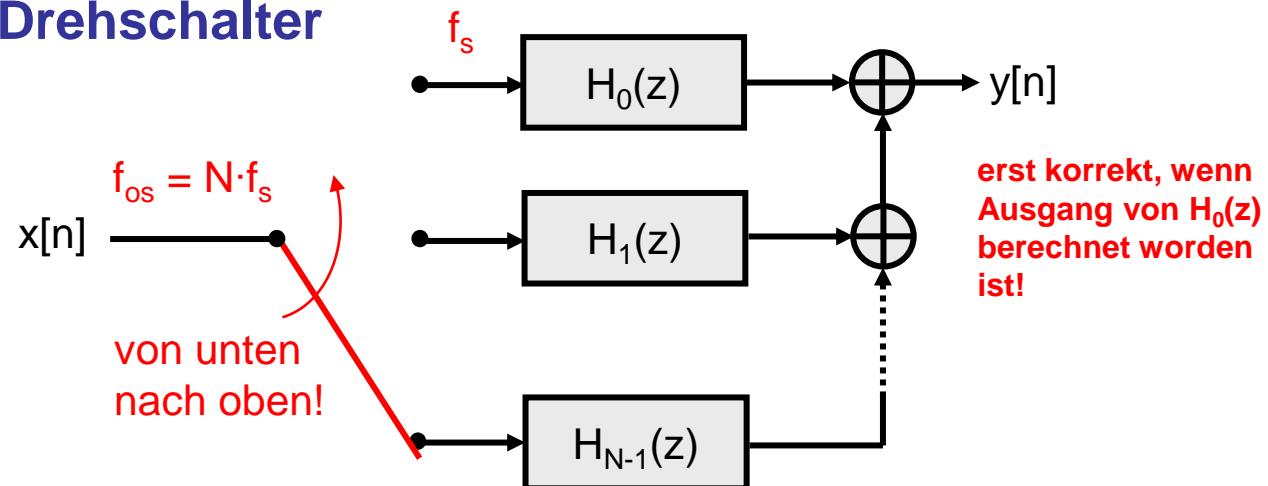


# Polyphasen-Dezimator

## Dezimator



## mit Kommutator / Drehenschalter



# Polyphasen-Dezimator

## Aufwand ohne Polyphasen-Struktur

- $N \cdot L$  MAC-Operationen pro Ausgangswert  $y[n]$

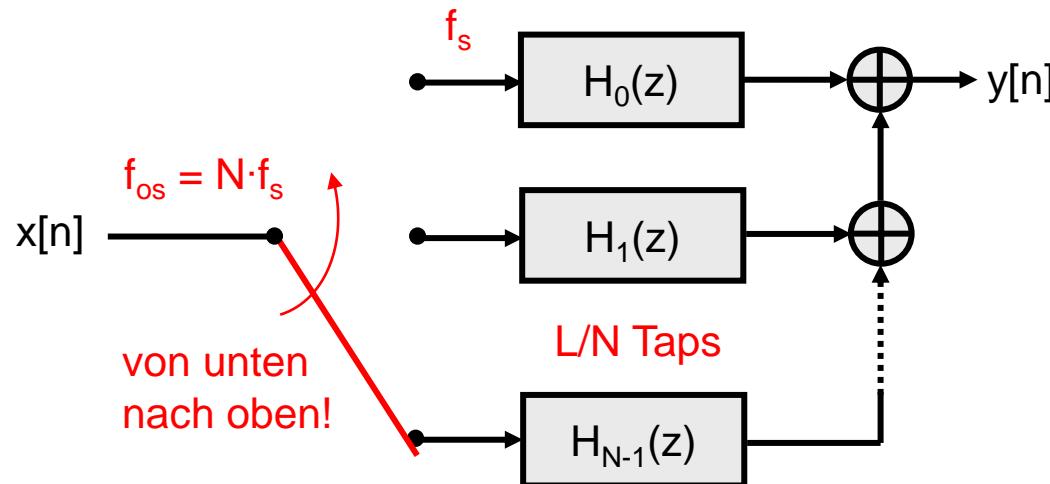
$N =$  wie viel down sample

$L =$  Ordnung



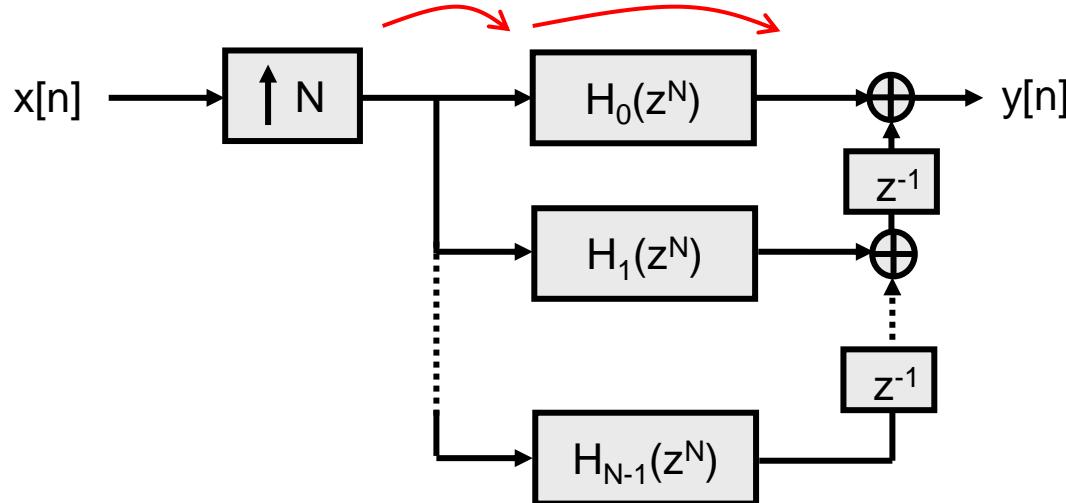
## Aufwand mit Polyphasen-Struktur

- nur  $N \cdot (L/N) = L$  MAC-Operationen pro Ausgangswert  $y[n]$



# Polyphasen-Interpolator

## Kaskade von Upsampler und Polyphasen FIR-Filter



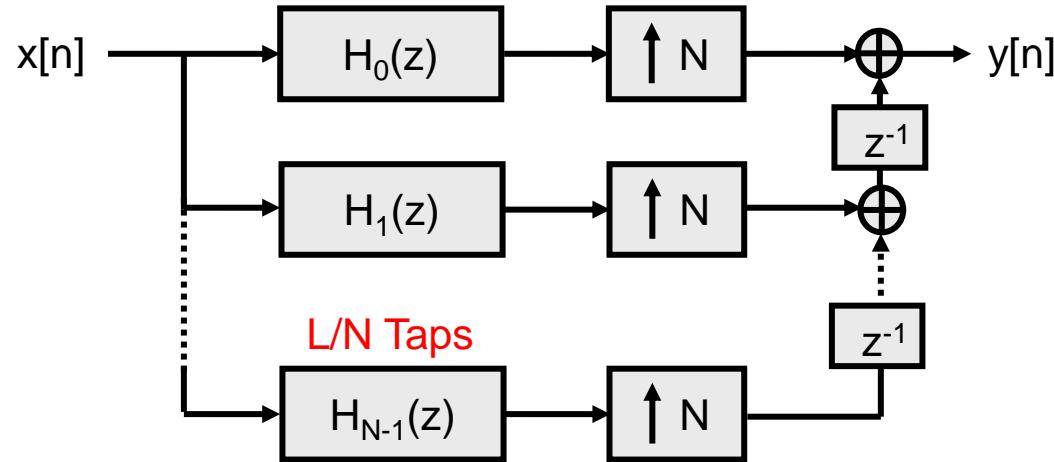
Übertragungsfunktion ist  
in mehreren teil  
übertragungsfunktionen  
geschrieben.

→ Verschiebung Upsampler bzw. Filter vom Gebiet mit schnellem Takt  $f_{os}$  ins Gebiet mit langsamem Takt  $f_s$  dank der Identität

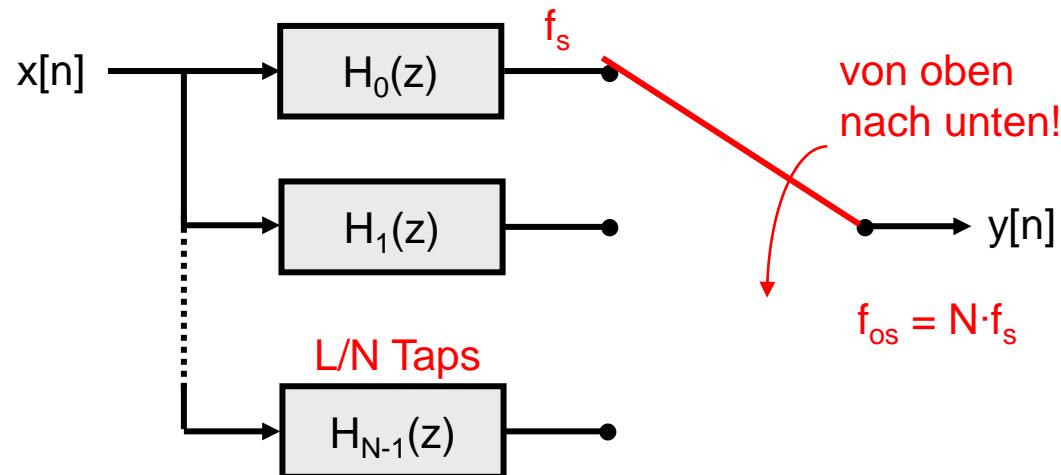


# Polyphasen-Interpolator

## Interpolator



## mit Kommutator/Drehschalter



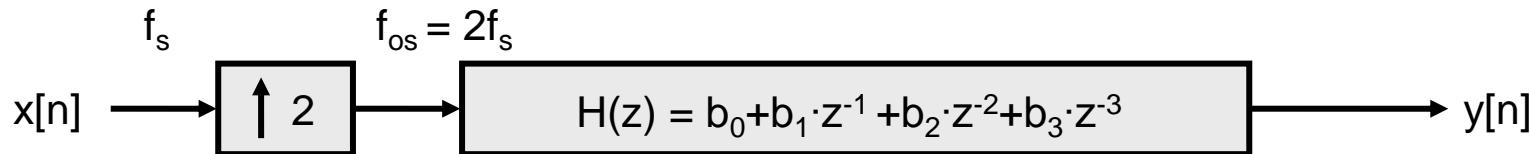
nur  $N \cdot (L/N) = L$  MAC-Operationen pro Eingangswert  $x[n]$  (statt  $N \cdot L$  MACs)

# Polyphasen-Interpolator: Beispiel

vgl. Folien 7.2-1 und 2

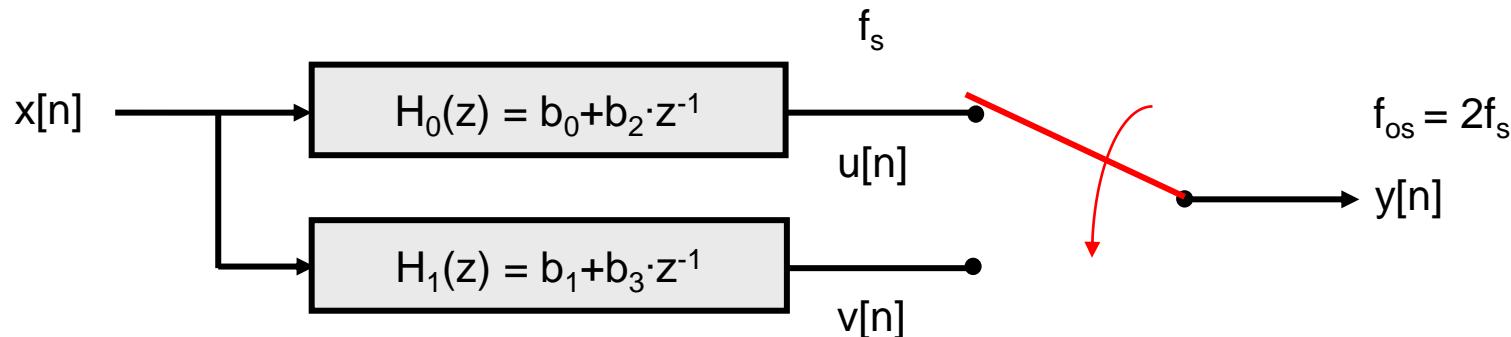
## Interpolator ohne Polyphasen-Struktur

- $N \cdot L = 8$  MAC-Operationen pro Eingangswert  $x[n]$



## Interpolator mit Polyphasen-Struktur

- $L = 4$  MAC-Operationen pro Eingangswert  $x[n]$



- Verifikation:  $y[2n] = u[n] = b_0 \cdot x[n] + b_2 \cdot x[n-1]$

$$y[2n+1] = v[n] = b_1 \cdot x[n] + b_3 \cdot x[n-1]$$

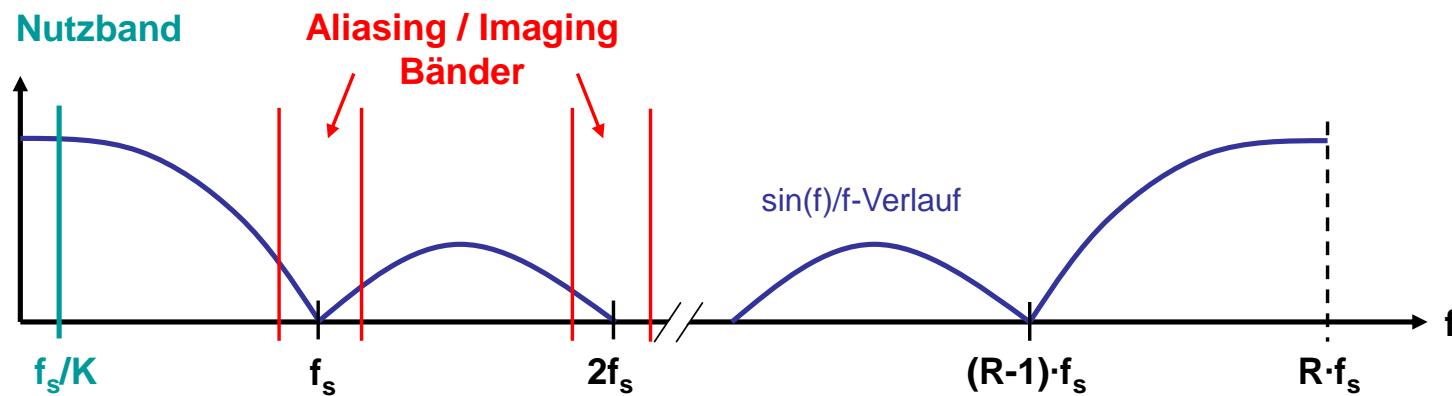
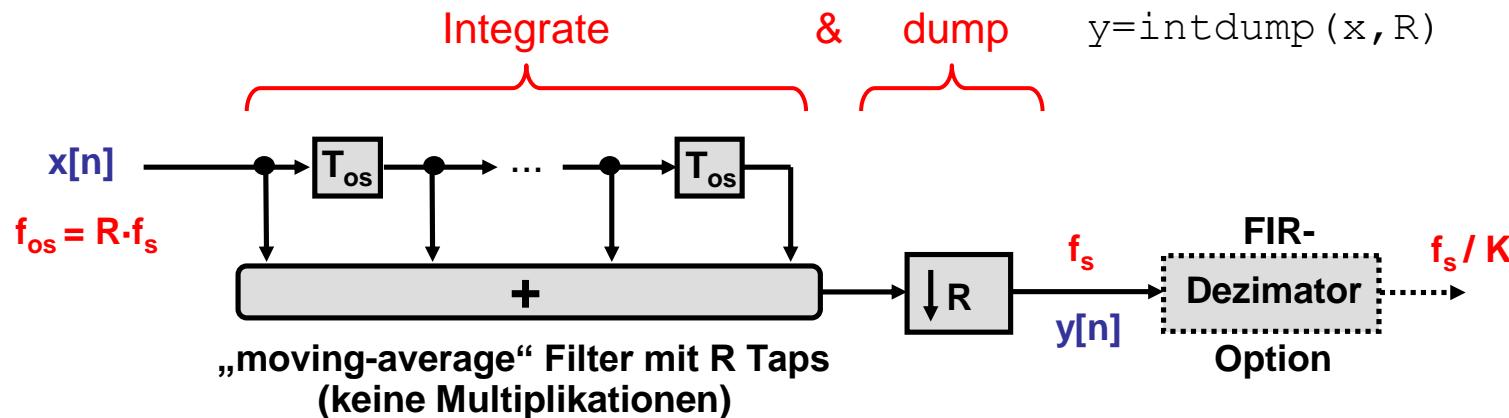
# CIC-Dezimatoren und Interpolatoren

---

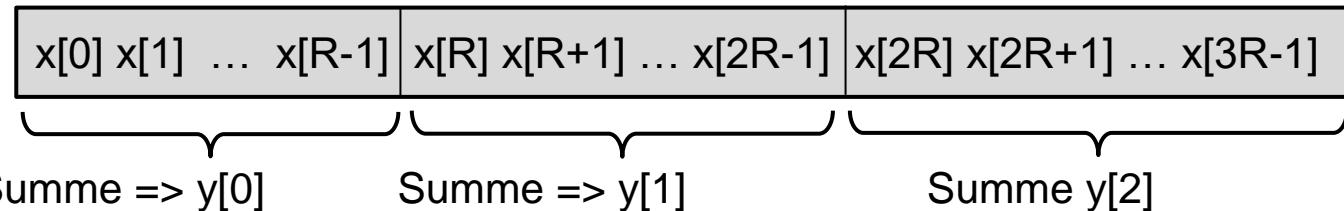
Dank der Verfügbarkeit von immer schnelleren und kostengünstigeren ADCs lohnt es sich, selbst schmalbandige Bandpass-Signale bei hohen Frequenzen abzutasten (z.B. nahe beim Sensor oder nahe an der Antenne bzw. **weit vorne in der Signalverarbeitungskette**), um sie dann im Digitalen weiter zu verarbeiten. Meistens werden diese Signale zuerst in einen tieferen Frequenzbereich verschoben. Dann sollte aber auch die Abtastrate reduziert werden, bevor das Signal weiterverarbeitet wird.

Wenn die Abtastrate sehr hoch ist, muss das Dezimationsfilter ganz besonders ökonomisch realisiert werden. In diesem Unterkapitel lernen wir digitale Dezimations- und Interpolationsfilter kennen, die ohne Multiplikatoren und mit nur sehr wenigen Speicherzellen realisiert werden können und sich deshalb sehr gut für eine Hardware-Realisierung eignen. Sie heißen **Cascaded-Integrator-Comb-** bzw. **CIC-Filter**.

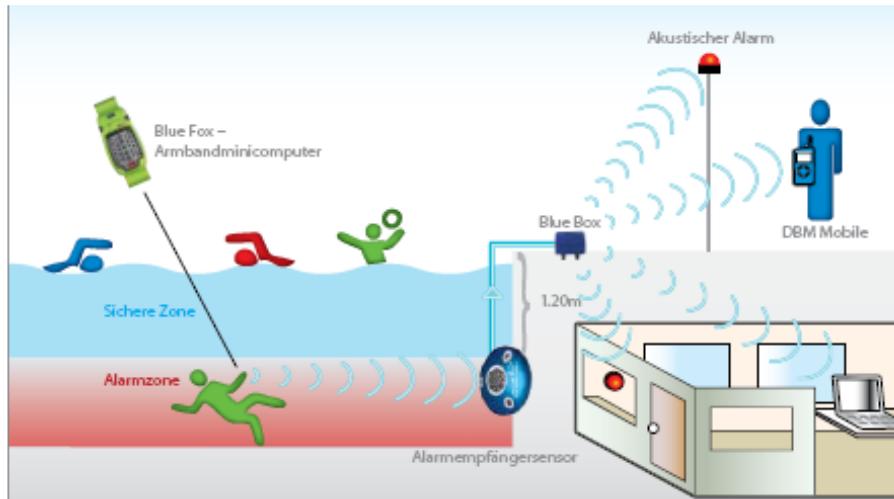
# Integrate & Dump Dezimator



## Berechnung



# Beispiel: Ultraschall-basiertes Pool Safety System



## Ultraschall Pool Safety System

(Quelle: Deep Blue AG, Hallwil)



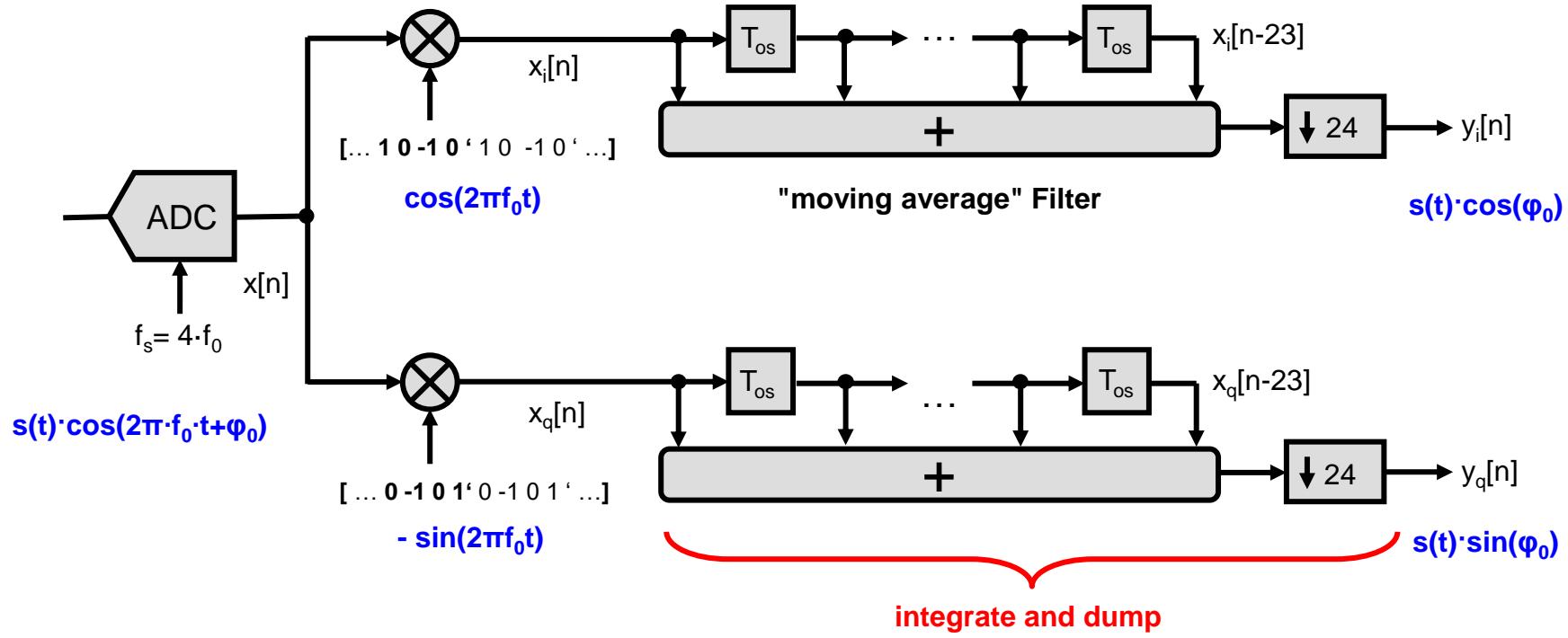
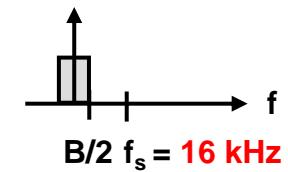
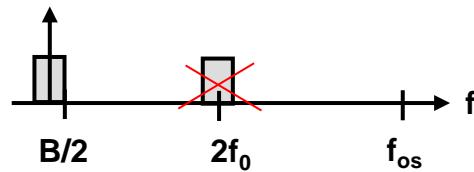
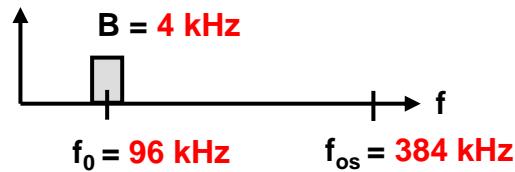
[SRF Beitrag](#) vom 2.7.2022

## Problem

- 4 kHz breites (Ultraschall-) Signal bei  $f_0 = 96$  kHz
  - Abtasten (mit  $4 \cdot f_0 = 384$  kHz) !
  - Frequenzverschiebung ins Basisband im Digitalen
  - Dezimation auf 16 kHz
- $\left. \begin{array}{l} \text{Frequenzverschiebung ins Basisband im Digitalen} \\ \text{Dezimation auf 16 kHz} \end{array} \right\}$  Digital-Down-Converter (DDC)

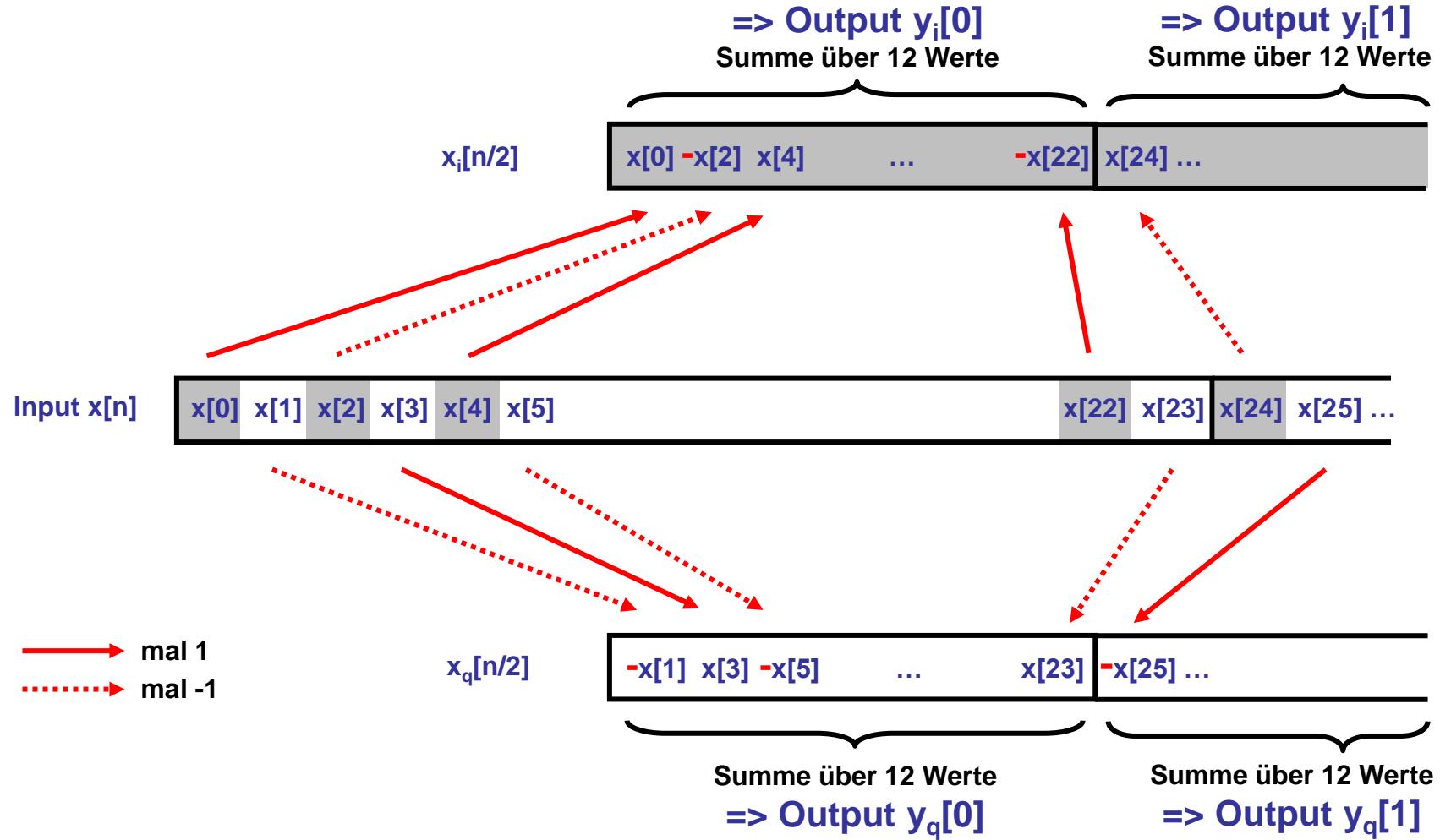
# Beispiel: Ultraschall-basiertes Pool Safety System

## Lösung



# Beispiel: Ultraschall-basiertes Pool Safety System

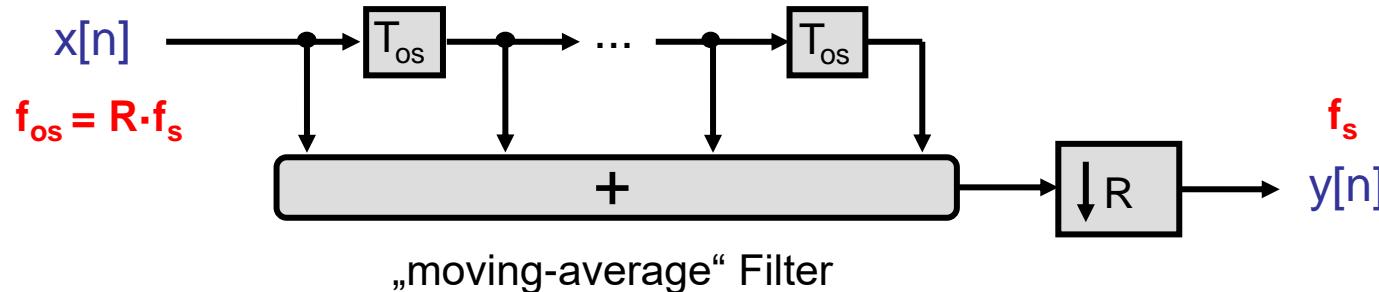
## Lösung



# CIC-Filter

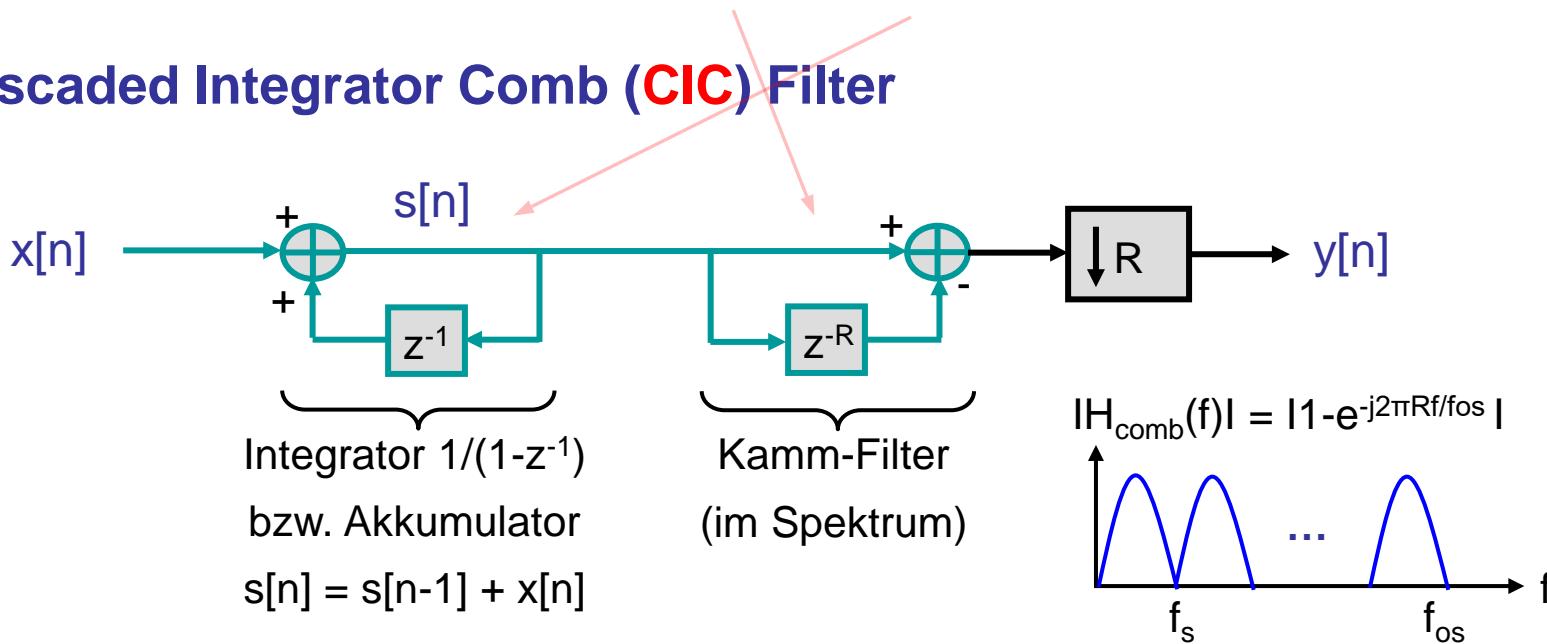
E. B. Hogenauer, "An economical class of digital filters for decimation and interpolation", IEEE Trans. on Acoustics, Speech and Signal Processing, April 1981.

## Einfacher R-facher Dezimator ohne Multiplikationen



$$H(z) = 1 + z^{-1} + \dots + z^{-(R-1)} = (1 - z^{-R}) / (1 - z^{-1})$$

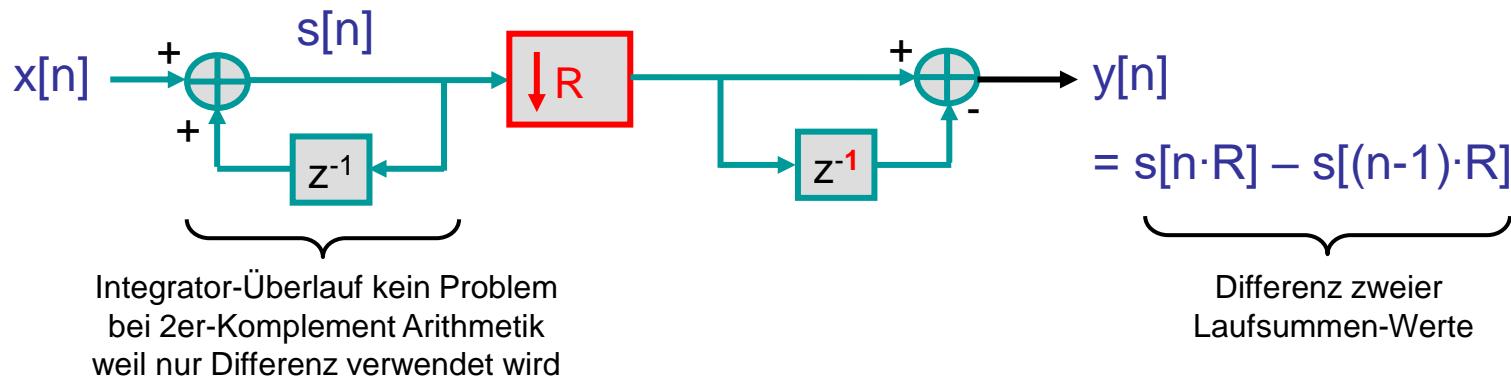
## Cascaded Integrator Comb (CIC) Filter



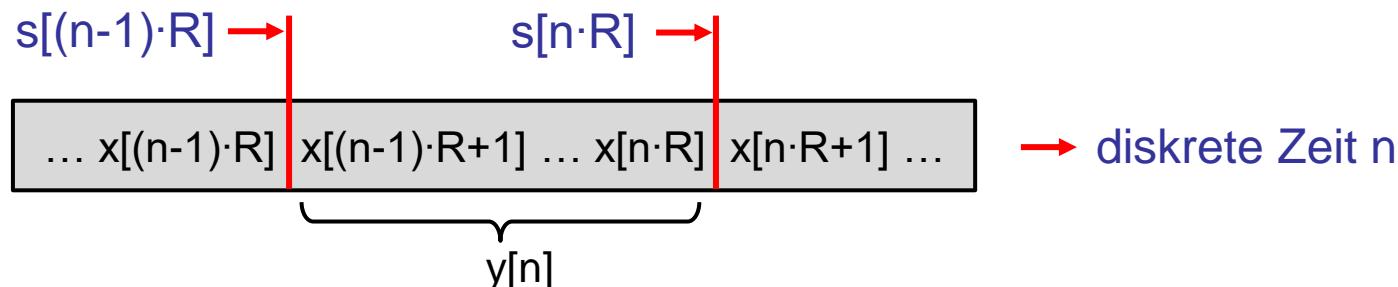
# CIC-Dezimatoren

## CIC-Dezimator (1-stufig)

- Verschiebung Downampler mit Hilfe nützlicher Identität



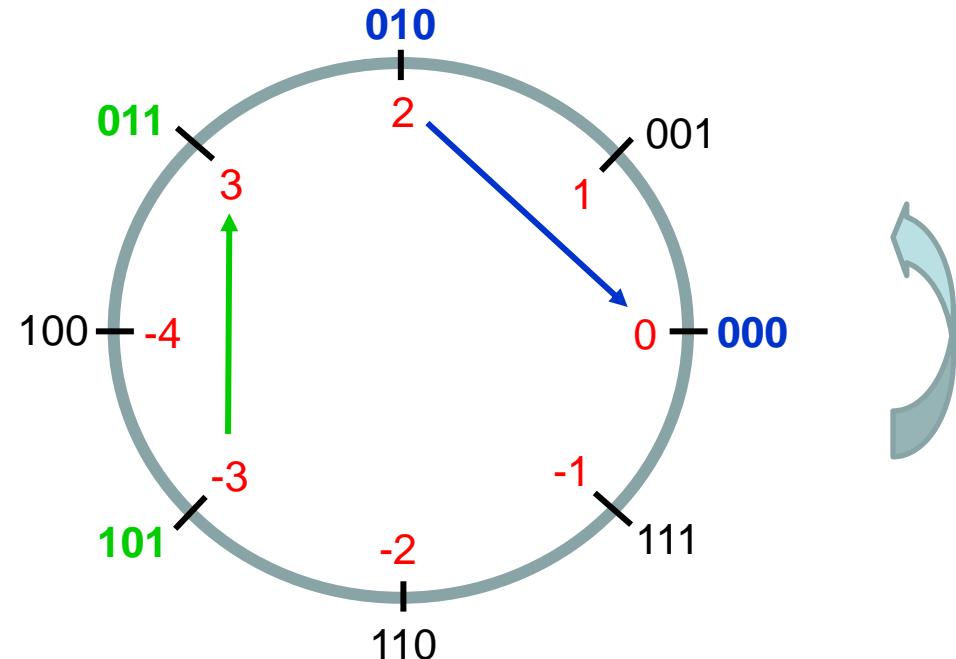
- Grafische Anschauung



# Integrator-Überlauf

Beispiel: Q2 signed integer Format, Differenz von 2 mit/ohne Überlauf

| [ $b_2 \ b_1 \ b_0$ ] | signed integer |
|-----------------------|----------------|
| [ 0 0 0 ]             | 0              |
| [ 0 0 1 ]             | 1              |
| [ 0 1 0 ]             | 2              |
| [ 0 1 1 ]             | 3              |
| [ 1 0 0 ]             | -4             |
| [ 1 0 1 ]             | -3             |
| [ 1 1 0 ]             | -2             |
| [ 1 1 1 ]             | -1             |



$$((-3) - 3) \bmod 4 = 2$$

$$\begin{array}{r} 101 \\ - 011 \\ \hline 010 \end{array} \quad \begin{array}{r} 101 \\ + 101 \\ \hline 010 \end{array}$$

$$2 - 0 = 2$$

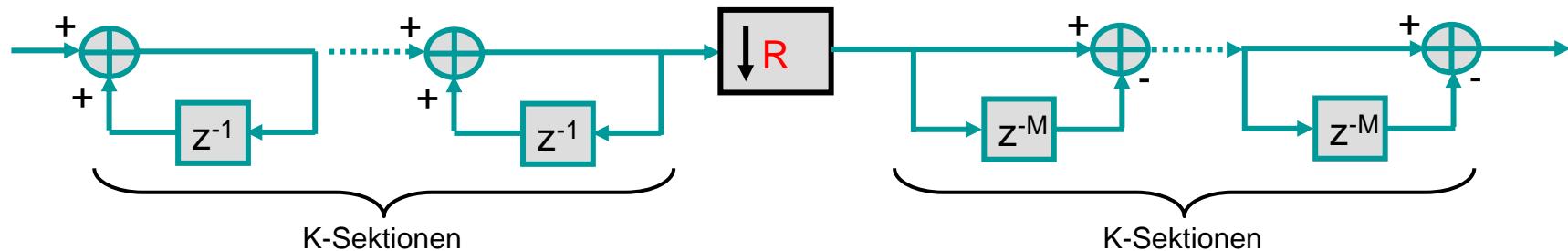
$$\begin{array}{r} 010 \\ - 000 \\ \hline 010 \end{array} \quad \begin{array}{r} 010 \\ + 000 \\ \hline 010 \end{array}$$

Reminder: Eine binäre Zahl  $b$  kann negiert werden, indem alle Bits von  $b$  invertiert und 1 LSB addiert wird.

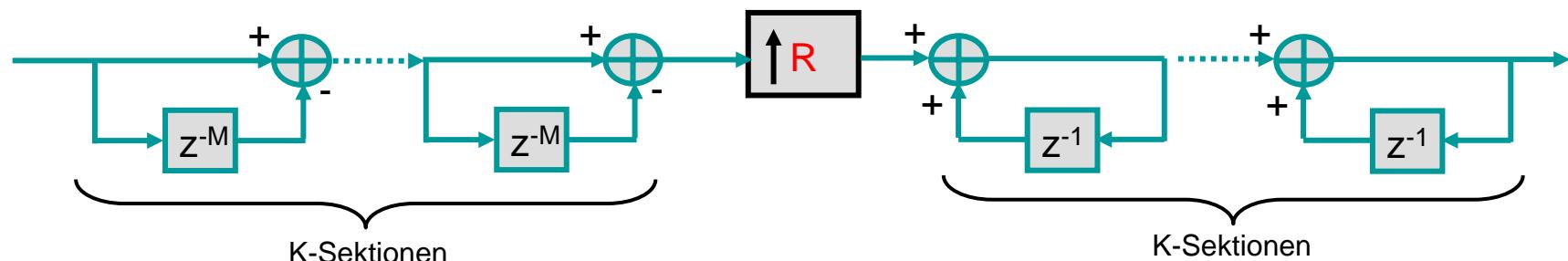
# CIC-Dezimatoren

## CIC-Filter mit K-Sektionen

- $K > 1$  "moving average" Filter mit MR-Koeffizienten, wobei  $M=1$  oder  $2$
- $H(z) = (1+z^{-1}+\dots+z^{-(MR-1)})^K = ((1-z^{-MR}) / (1-z^{-1}))^K$
- selektiverer,  $(\sin(f)/f)^K$ -förmiger Frequenzgang



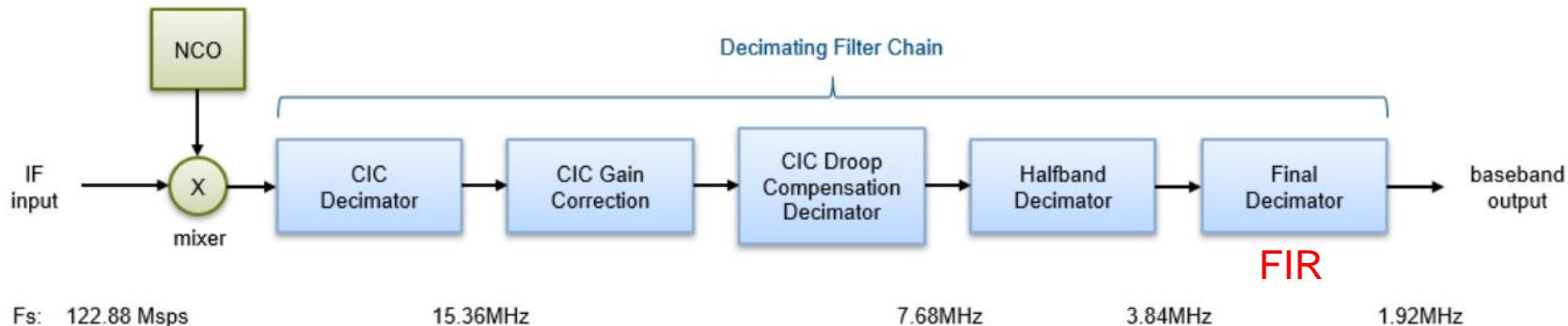
## CIC-Interpolator mit K-Sektionen



# Digital Down Converter (DDC)

Hardware-Implementation in **FPGA** oder direkt im **ADC**!

- meist Kaskade aus NCO => CIC-Dezimator => FIR-Dezimator
  - Numerical-Controlled-Oscillator (NCO): Frequenzverschiebung
  - CIC-Dezimator: hohe Abtastrate am Input, grosser Dezimations-Faktor, einfachste Struktur ohne Multiplikatoren, wenig frequenzselektiv
  - FIR-Dezimator: moderate Abtastrate am Input, kleiner Dezimations-Faktor, FIR-Struktur, frequenzselektiv
- Beispiel: [Implement Digital Downconverter for FPGA](#)



äquivalente Struktur für Digital-Up-Converter (DUC)