Digitale Signalverarbeitung

Zusammenfassung

Andreas Ming / Quelldateien

Table of contents

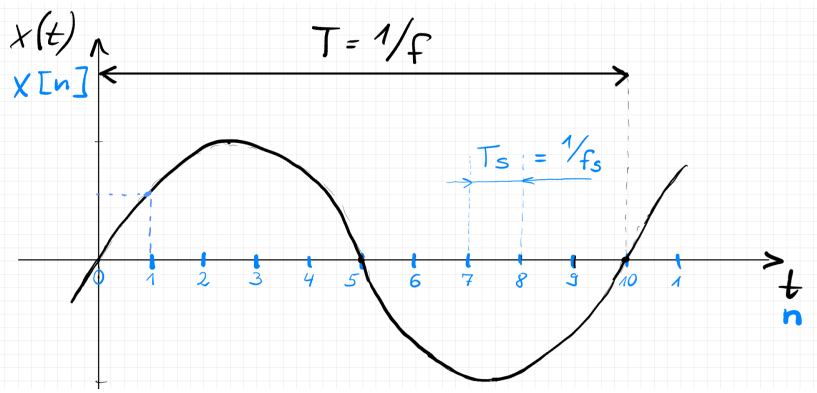
Digital Signals in the Time Domain

Signal Analysis

Sampling of Analog Signals

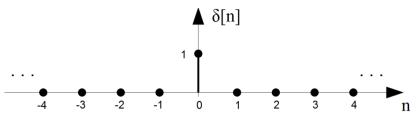
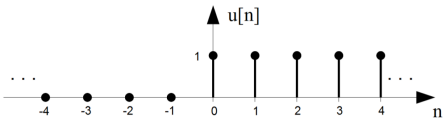
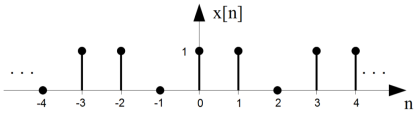
By sampling $x(t)$ in the interval of T_S we get the sequence of signal values $x[n]$ with $-\infty \leq n \leq +\infty$

$$x(n \cdot T_S) = x[n]$$



Signal	Property
causal	$x[n] = 0$ for $n < 0$
real	$x[n]$ Real
complex	Re & Im or Amplitude & Phase

Basic Digital Signals

unit impulse	unit step	periodical signal
$\delta[n] = \begin{cases} 0 : n \neq 0 \\ 1 : n = 0 \end{cases}$	$u[n] = \begin{cases} 0 : n < 0 \\ 1 : n \geq 0 \end{cases}$	$x[n] = x\left[n + \frac{T_0}{T_S}\right]$ with $\frac{T_0}{T_S} = k$
		

There is also a **complex hamonic** sequence with the period duration of $T_0 = \frac{1}{f_0}$

$$x[n] = \hat{X} \cdot e^{j2\pi f_0 n T_S}$$

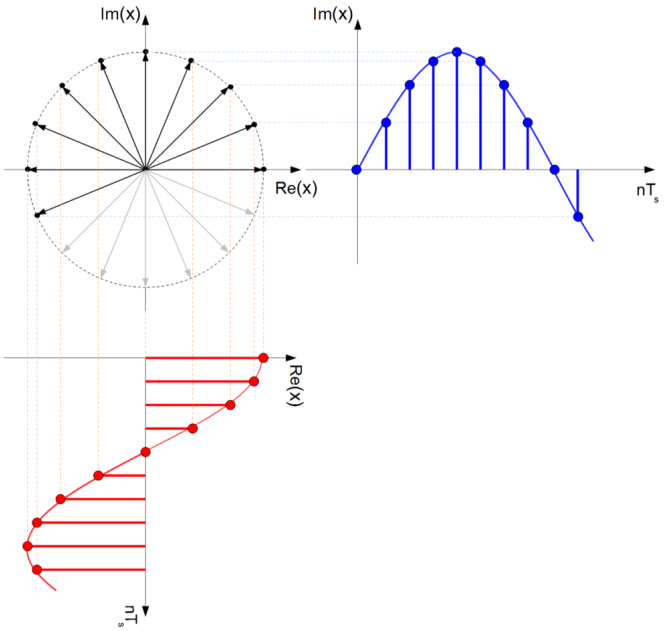


Figure 1: Complex hamronic sequence with period duration $T_0 = 16 \cdot T_S$

Statistical Signal Parameters

Stochastic signals must be qualified by *statistical signal parameters* within the **observation interval** $T = N \cdot T_S$.

expected / mean	quadratic mean	variance
DC-component $\mu_x = \frac{1}{N}\sum_{i=0}^{N-1} x[i]$	average power *(w/ DC)* $\rho_x^2 = \frac{1}{N}\sum_{i=0}^{N-1} x[i]^2 = P_{avg}$	average power *(w/o DC)* $\sigma_x^2 = \frac{1}{N}\sum_{i=0}^{N-1} (x[i] - \mu_x)^2 = P_{AC}$

Signal Operations

Correlation

	cross-correlation	auto-correlation
Static Linear	$R = \frac{1}{N}\sum_{i=0}^{N-1} x[i]y[i]$ $r_{xy}[n] = \sum_{i=-\infty}^{\infty} x[i]y[i+n]$	$R = \frac{1}{N}\sum_{i=0}^{N-1} x[i]x[i]$ $r_{xx}[n] = \sum_{i=-\infty}^{\infty} x[i]x[i+n] = P_{avg}$

For **linear correlation** the resulting length of r_{xy} equals

$$N_{xy} = N_x + N_y - 1$$

and the range of shifts for the computation is given by

$$-N_x + 1 \leq n \leq N_y - 1$$

For signals differing in length, zero-padding can be applied.

Convolution

The *Convolution* involves folding the time-displaces signal around the point $n = 0$

$$z[n] = \sum_{i=-\infty}^{\infty} x[i]y[-i+n] \tag{0.1}$$

A convolution equals a polynomial multiplication.

The range of shifts for the computation is given by

$$0 \leq n \leq N_x + N_y - 2$$

The Convolution described in Equation 0.1 is called a **linear convolution** and can be applied to two signals of different length

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

```
z = conv(x, y)
```

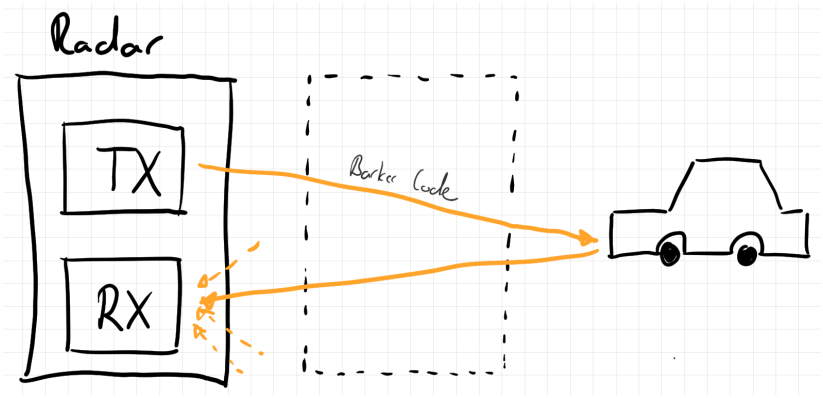
There is also the **circular convolution** which requires both signals to be of equal length N. If necessary, *zero padding* can be applied. The resulting signal then also is of length N.

$$z[n] = x[n] \circledast_N y[n] = y[n] \circledast_N x[n]$$

The circular convolution corresponds to matrix multiplication In order to compute $x[n] \circledast_N y[n]$, the NN-matrix constructed from circular shifting y must be multiplied with vector x.

```
z = convmtx(x,y)
```

Anwendung: Radar



Um bei einem Radar nur auf das gewünschte Signal zu reagieren, also auf das eigene, wird vom Radar ein Barker-Code ausgesendet. Über Korrelation kann so die Laufzeit eindeutig zugeordnet werden.

> **Barker-Code**
>
> Es können auch andere Codes ausgesendet werden, die verwendeten Signale müssen jedoch sehr gute Autokorrelationseigenschaften aufweisen.

Analog-to-Digital & Digital-to-Analog Conversion

Steps of A/D- and D/A-Conversion

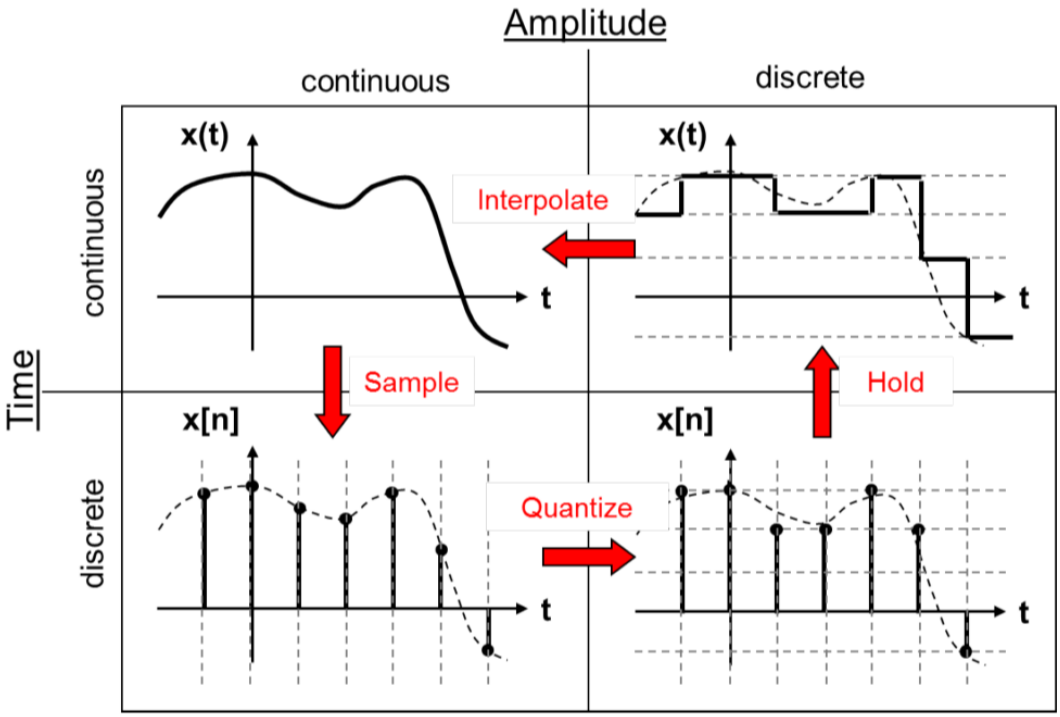


Figure 2: Signal classification in a/d- and d/a-conversion

A/D

Sample: Signal values are recorded at sampling rate f_S. This yields a train of pulses.

Quantize: The discrete signal values are mapped to a given number of quantization levels.

Code: The quantified values can be stored in a coded way. DSPs most often store the quantified values.

D/A

Decode: The coded samples are converted back into a suitable representation for the digital-to-analog conversion method used.

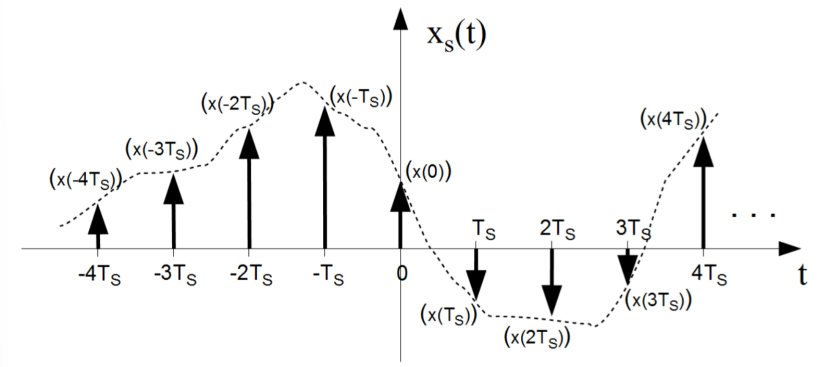
Hold: A momentary discrete signal value is constant over the sample period T_S.

Interpolate: The continuous staircase signal form is smoothed by a low-pass-filter.

Sampling and Aliasing

Sampling a time-continuous signal $x(t)$ corresponds to a multiplication with a Dirac impulse series. The resulting signal $x_S(t)$ can be regarded as a train of weighted Dirac impulses.

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



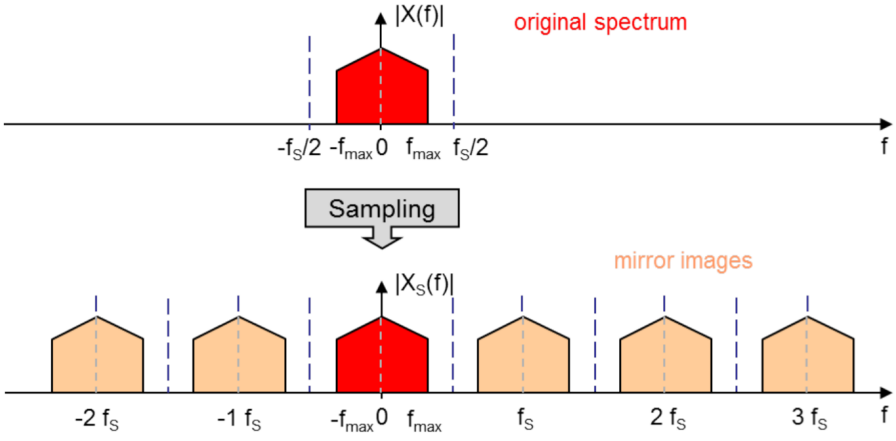
Through the application of the Fourier property $x(t)e^{j2\pi f_0 t} \circ\!\!-\!\!\bullet X(f - f_0)$ we obtain the frequency spectrum of the sampled signal as

$$X_S(f) = \frac{1}{T_S} \sum_{k=-\infty}^{\infty} X(f - kf_S)$$

! Observation

The frequency of the analog signal $x(t)$ consists of the original spectrum $X(f)$ superimposed (*überlagert*) by mirror images of the spectrum

$$f_k = k \cdot \frac{f_S}{N}$$

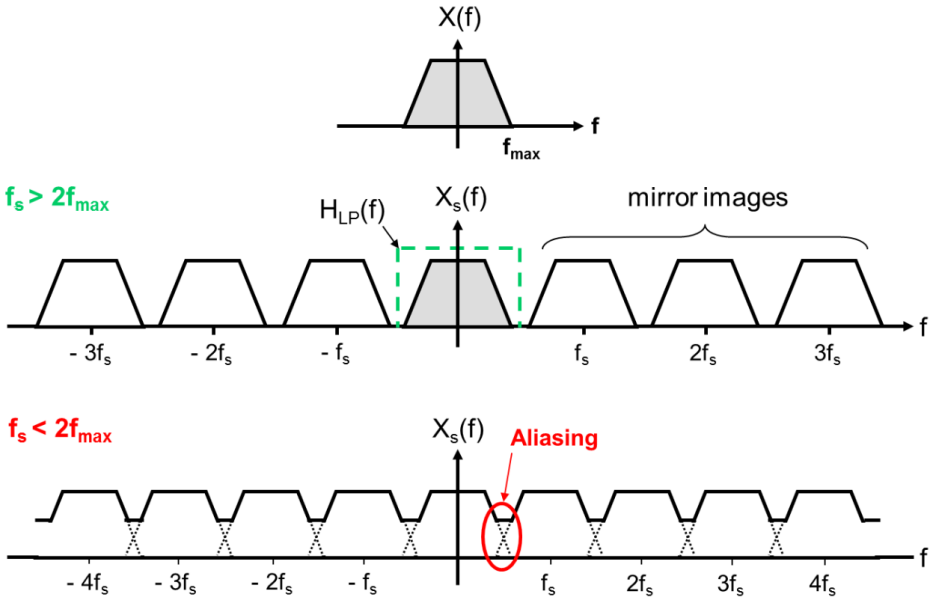


Aliasing

i Sampling Theorem

An analog signal $x(t)$ with $X(f) = 0$ for $|f| > |f_{max}|$ is uniquely defined by its sample values $x[n] = x(nT_S)$, if for the sampling frequency $F_S = \frac{1}{T_S}$ holds:

$$f_s > 2 \cdot f_{max}$$



Band-Pass Sampling

$x(t)$ can be perfectly reconstructed if an integer $N \geq 0$ exists, such that $X(f) = 0$ holds for all frequencies f outside

$$-\frac{N+1}{2} f_S \leq f \leq -\frac{N}{2} f_S \qquad \text{and} \qquad \frac{N}{2} f_S \leq f \leq \frac{N+1}{2} f_S$$

For a given band-pass signal with given limits f_{min} and f_{max} it can be checked if the sampling frequency f_S can be used ($N \geq 1$)

$$\frac{2 \cdot f_{min}}{N} \geq f_S \geq \frac{2 \cdot f_{max}}{N+1}$$

For sampling with $N =$ **even** we get the mirror spectrums

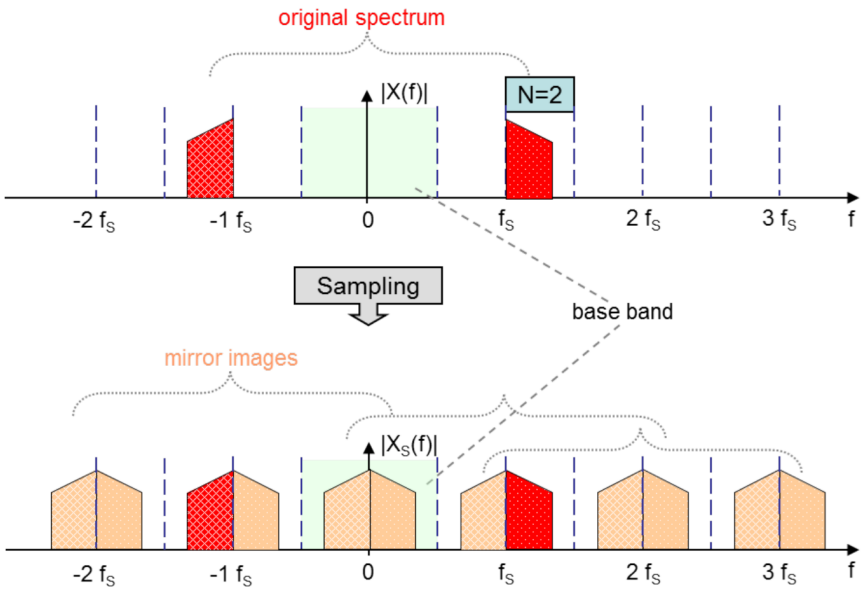


Figure 3: Band-pass sampling for even N

For sampling with $N =$ **odd** we get the mirror spectrums

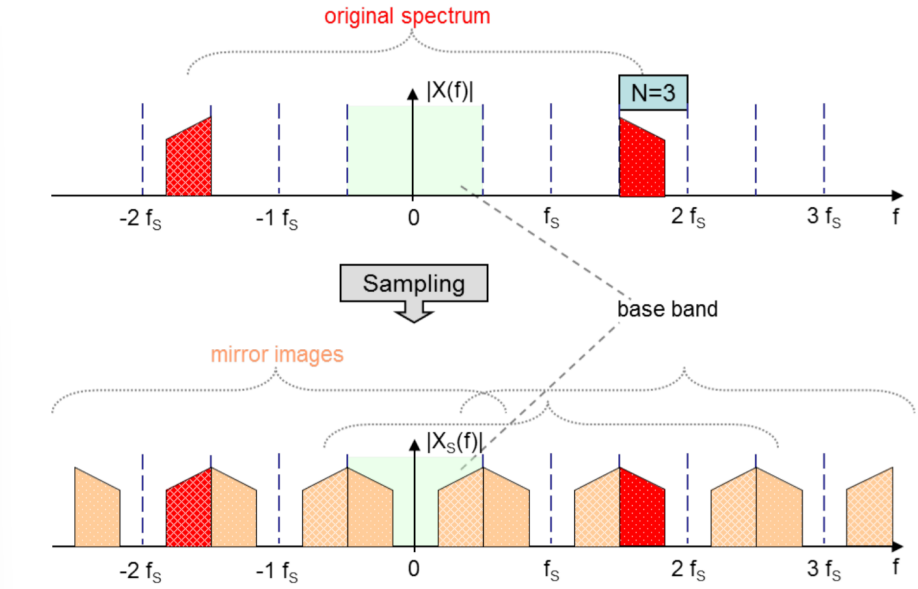


Figure 4: Band-pass sampling for odd N

! Spectrum Correction

Note that for N odd, the original spectrum appears "inverted" in the base band. The original structure of the spectrum can be re-obtained by changing the sign of every second sample of the time-domain sequence, i.e.

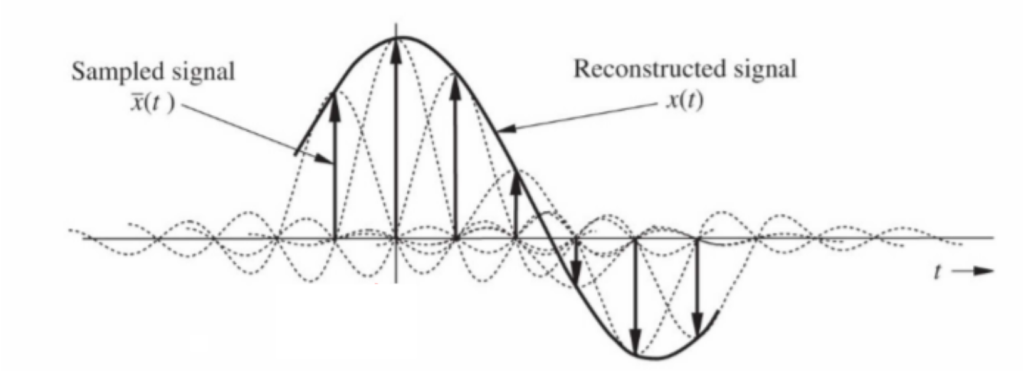
$$\tilde{x} = (-1)^n \cdot x[n]$$

Reconstruction

Ideal Reconstruction

Sampled signals w/o Aliasing can be *theoretically* reconstructed error-free. For this all mirror-spectra must be eliminated by a ideal low-pass filter. Because of the property $\text{rect}\left(\frac{t}{T}\right) \circ\!\!-\!\!\bullet |T| \cdot \text{si}(\pi T f)$ the ideal interpolation equals

$$x(t) = \sum_{n=-\infty}^{\infty} x(nT_S) \cdot \text{sinc}(\pi f_S (t - nT_S))$$



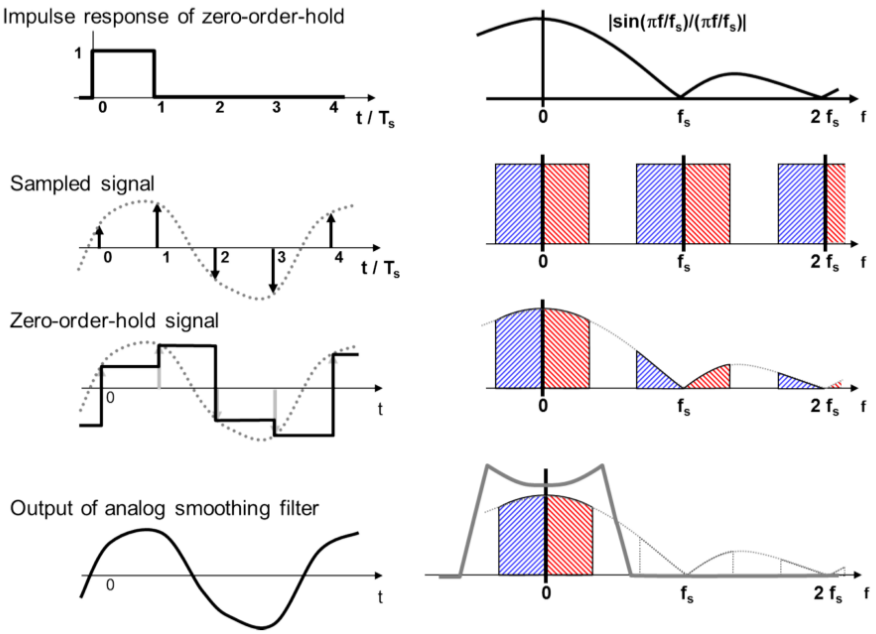
> **Ideal values**
>
> At the points $t = nT_S$ all values of except of $x(nT_S)$ equal 0. Thus at every point of $x(nT_S)$ the signal reaches the right value.
> **Caution!** because of the infinit sum of sinc-pulses, the values between $x(nT_S)$ aren't particularly correct. Also the further to the "edge" of x you get, the more inaccurate it gets.

Practical Reconstruction

In practice Reconstruction is very often done with a simple *zero-order-holder* (ZHO). Such operation holds each sample value constant over a subsequent sample interval T_S. This results in a stair-case waveform, thus making a very poor low-pass filter. For this reason a analog low-pass filter is usually implemented.

Without analog filtering the **SNR** can be approximated as

$$\text{SNR} \approx 6dB \cdot \log_2\left(\frac{f_S}{f_0}\right) - 11dB$$

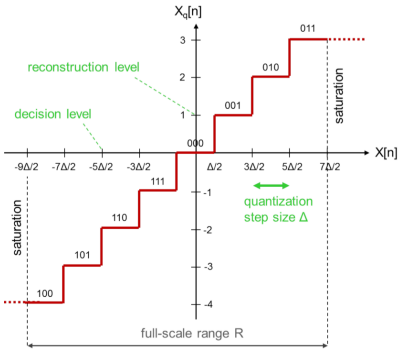
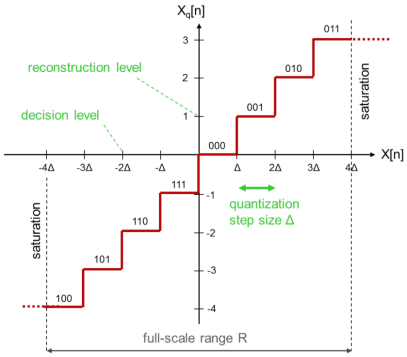


Quantization of Signals

Uniform Quantization

When quantizing sample values with W bits, the dynamic range R of the sampled signal $x[n]$ can be divided into 2^W intervals of equal width. Thus, the width of one **quantization step** is given by

$$\Delta = \frac{R}{2^W}$$

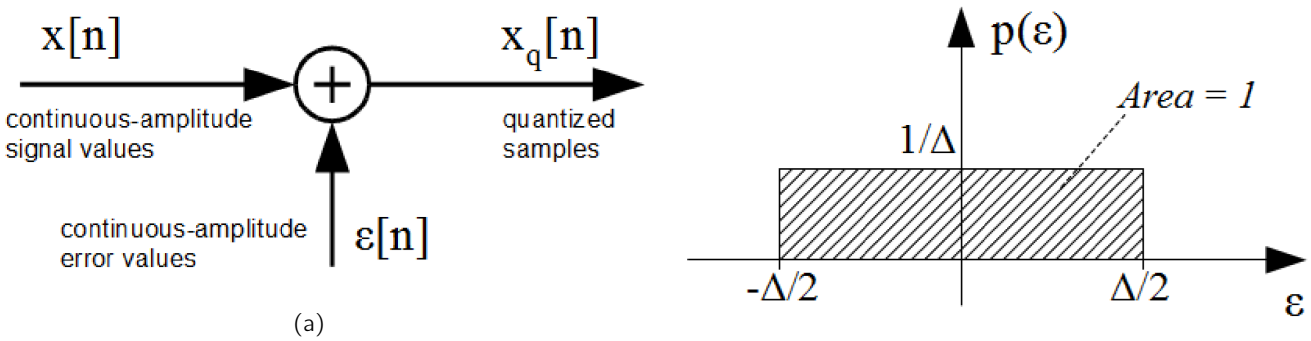
Mid-Tread	Mid-Rise
Decision levels at $\pm 0.5\Delta, \pm 1.5\Delta, \pm 2.5\Delta, \ldots$	Decision levels at $0, \pm\Delta, \pm 2\Delta, \ldots$
	
$-\frac{\Delta}{2} < \epsilon \leq \frac{\Delta}{2}$	$-\Delta < \epsilon \leq 0$

Furthermore **Clipping** occurs if the signal values of $x[n]$ are outside of the full-scale range R.

Quantization noise

The quantization error ϵ manifests itself as noise superimposed to the quantized signal

$$\epsilon[n] = x_q[n] - x[n]$$



(a) (b)

Figure 5: Model of quantization noise (Figure 5a) and probability density function (Figure 5b)

The power of the quantization noise signal is

$$P_\epsilon = \frac{1}{\Delta}\int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} \epsilon^2 d\epsilon = \frac{\Delta^2}{12}$$

The signal-to-noise ratio expressed in dB yields

$$SNR_{dB} = 6 \cdot W + 10 \cdot \log_{10}\left(\frac{12 \cdot P_x}{R^2}\right)$$

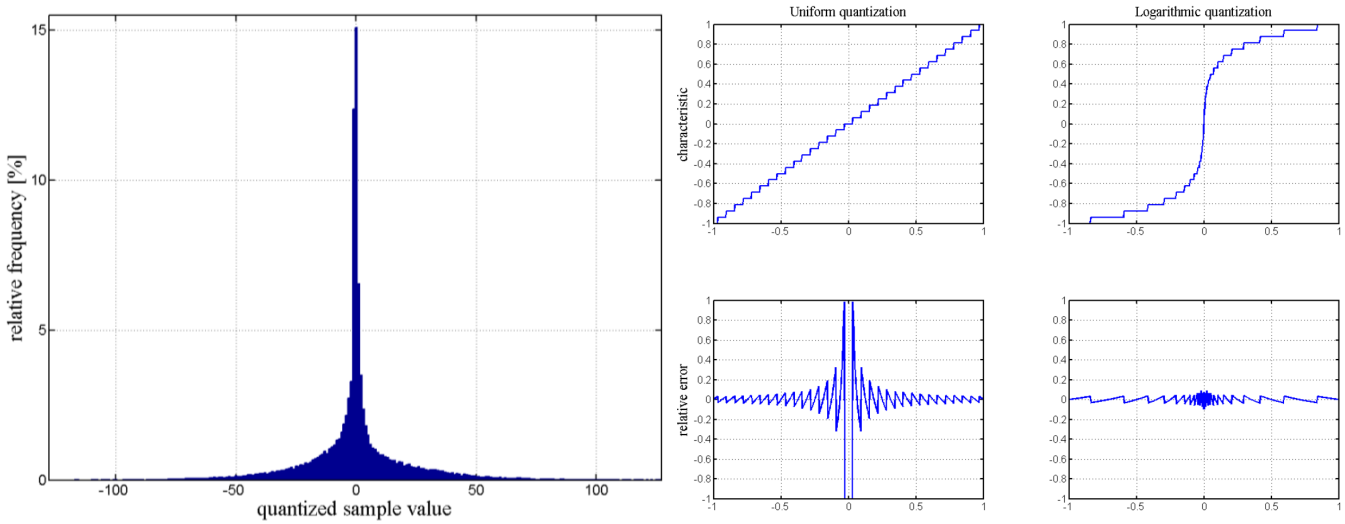
For Harmonic- & Full-Scale-Signals

$$SNR_{dB} = 6 \cdot W + 1.76 \approx 6 \cdot W$$

For every additional Bit, the SNR can be sixfold in $[dB]$.

Logarithmic Quantization

One way to increase the SNR associated with quantization, is to adapt the quantizer characteristics to the statistical properties of the signal being quantized. One kind of signal with these properties are voice signals were very small amplitude values are orders of magnitude more likely than large amplitude values (Figure 6a).



(a) (b)

Figure 6: Comparison of uniform and logarithmic quantization

There are several standards for implementing Logarithmic Quantization. One such standard is the μ-law algorithm

$$f_\mu(x) = \text{sgn}(x) \cdot \frac{\ln(1 + \mu \cdot |x|)}{\ln(1 + \mu)} \qquad -1 \leq x \leq 1$$

With this applied the relative error can be significantly improved (Figure 6b).

Digital Signals in the Frequency Domain

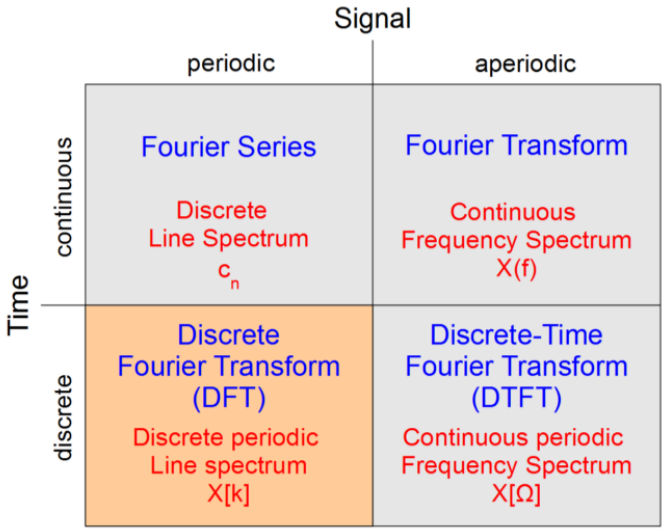


Figure 7: Comparison of Fourier methods

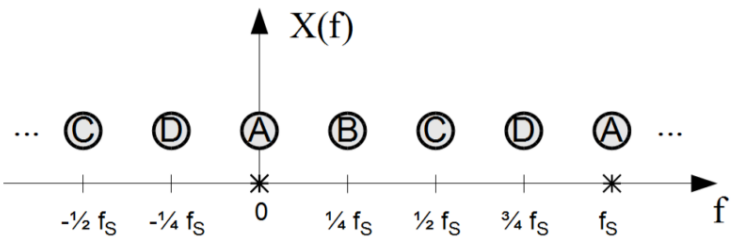
Fourier in Discrete Time

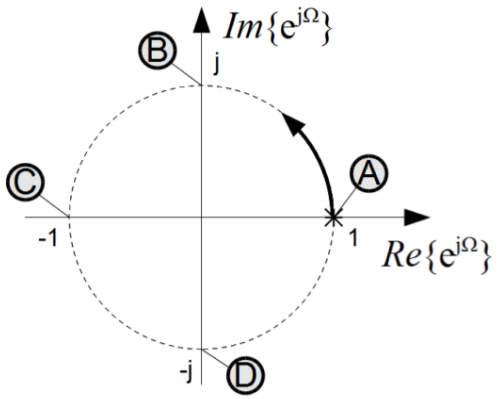
Discrete Time

Substituteing the continuous time t and integration with the discrete sample points nTs and summation, we get the frequency spectrum of the sampled signal $x_S(t)$. Thourgh the *normalized angular frequency*, we obtain the **Discrete-Time Fourier Transform (DTFT)**.

$$\underbrace{X_S(f) = \sum_{n=-\infty}^{\infty} x(nT_S)e^{-j2\pi f nT_S}}_{\text{Continuous Fourier Transform}} \overset{\Omega=2\pi f T_S=2\pi\frac{f}{f_S}}{=} \underbrace{X(\Omega) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\Omega n}}_{\text{Discrete Fourier Transform}}$$

The **DTFT** produces a 2π-periodic, continuous spectrum. This is due to the mapping of the linear frequency axis onto the unit circle at the sampling rate of f_S.





Finite Measurement Interval

The lowest frequency we can capture in the measurement interval T, with a finite amount of sample points N is

$$f_1 = \frac{1}{T} = \frac{1}{N \cdot T_S} = \frac{f_S}{N}$$

! Resolvable frequencies

From the equation we learn that we can resolve lower frequencies if we increase N, on the other hand the highest frequency we can cover is f_S. The following generally applies

$$f_k = k \cdot \frac{f_S}{N}$$

With this in mind we let the discrete-time-index n only run from 0 to $N-1$. We replace the $\frac{f}{f_S}$ in Ω with $\frac{k}{N}$ and get the **Discrete Fourier Transform (DFT)** where k is the discrete frequency index

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi n \frac{k}{N}} \qquad k = 0, 1, 2, \ldots, N-1$$

i Observation

We get exactly as many *Fourier coefficients* $X[k]$ back, as we provide *samples* $x[n]$.
The DFT produces a discrete and periodic line spectrum. The frequency resolution is $\frac{f_S}{N}$, i.e. we get the spectral values at the frequency points

$$0, \frac{f_S}{N}, 2\frac{f_S}{N}, \ldots, (N-1)\frac{f_S}{N}$$

Inverse Discrete Fourier Tranfsorm (IDFT)

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j2\pi n \frac{k}{N}} \qquad n = 0, 1, 2, \ldots, N-1$$

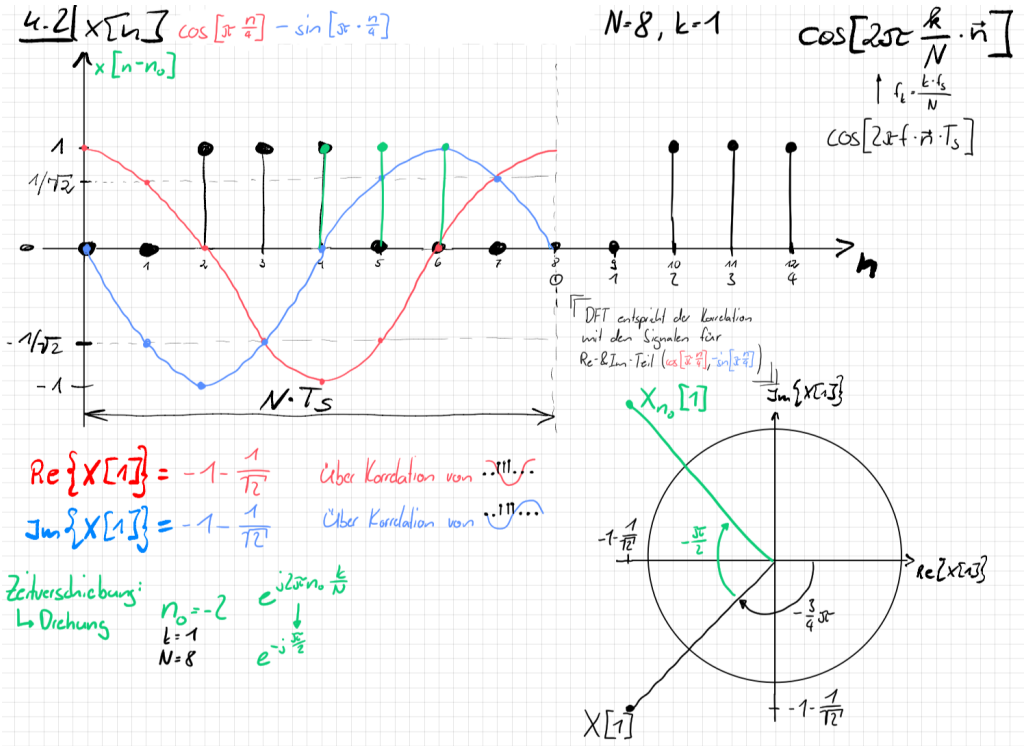


Figure 8: A intuitive approach to the DFT, also check chapter 4.2 of the main script by Wassner

Porperties of the DFT

Table 6: Periodicity

The *DFT* is f_S-periodic	The *IDFT* is periodic with $T = NT_S$
$X[k] = X[k+N]$	$x[n] = x[n+N]$

Table 7: Symmetry

DFT of a *real-valued signal* is symmetric around $k = \frac{N}{2}$
$X\left[\frac{N}{2}+m\right] = X^*\left[\frac{N}{2}+m\right]$

Table 8: Time/Frequency Shifting

Shifting by n_0 corresponds to a linear phase offset to all spectral values of the original time sequence	Multiplying the time sequence with a complex exponential, results in a constant frequency shift of the original spectrum
$x[n+n_0] \quad \circ\!\!-\!\!\bullet \quad e^{j2\pi n_0 \frac{k}{N}} \cdot X[k]$	$e^{j2\pi k_0 \frac{n}{N}} \cdot x[n] \quad \circ\!\!-\!\!\bullet \quad X[k-k_0]$

Table 9: Modulation

A direct consequence of the frequency shifting property
$\cos(2\pi k_0 \frac{n}{N} \cdot x[n] \quad \circ\!\!-\!\!\bullet \quad \frac{1}{2}(X[k+k_0] + X[k-k_0])$

Table 10: Parseval Theorem

Between the signal samples $x[n]$ and the Fourier coefficients $X[k]$ following relationship exists $\frac{1}{N}\sum_{n=0}^{N-1} x[n]^2 = \sum_{k=0}^{N-1} \left\lvert \frac{X[k]}{N} \right\rvert^2$

Table 11: Correspondence of Convolution and Multiplication

$x[n] \circledast_N y[n]$	$\circ\!\!-\!\!\bullet$	$X[k] \cdot Y[k]$	$(k = 0, 1, \ldots, N-1)$

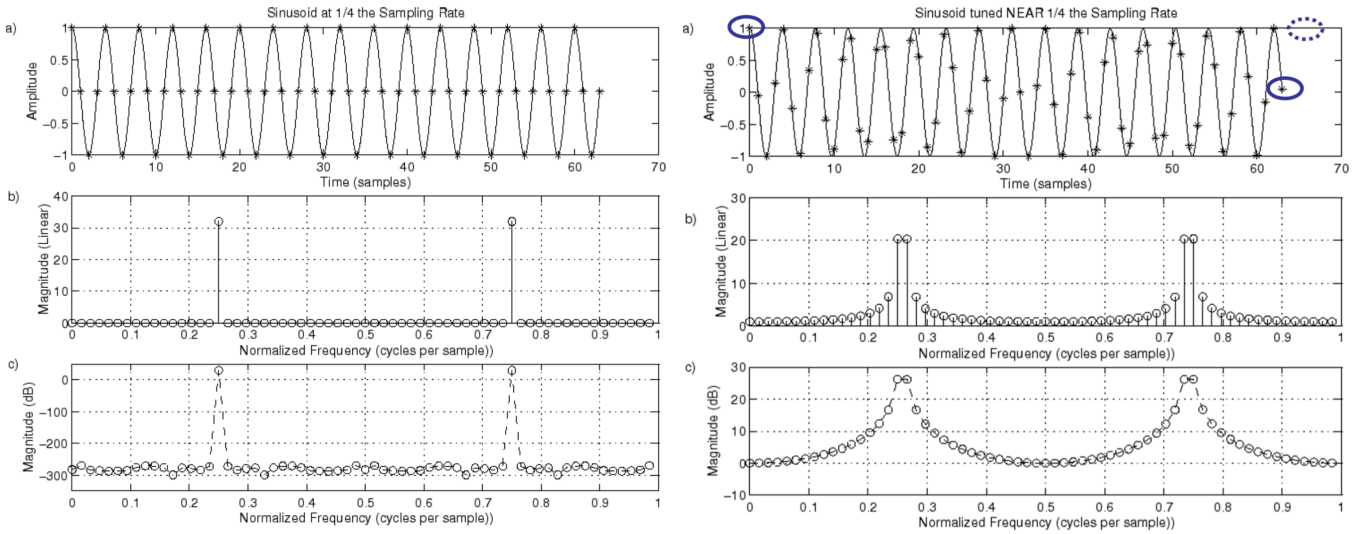
Range of Validity of the DFT

The Discrete Fourier Transform (DFT) accurately represents the spectrum of periodic signals within a finite measurement interval but requires an infinite interval for aperiodic signals to approach the correct result of the Discrete-Time Fourier Transform (DTFT). In the case of aperiodic signals, the DFT provides an approximation of the DTFT spectrum, and windowing functions are commonly employed to minimize the associated approximation errors in practical applications. In this case the DFT samples the DTFT at discrete points of the noramlized angular frequency Ω

$$X[k] = X(\Omega)|_{\Omega=2\pi\frac{k}{N}}$$

Practical Application Aspects of the DFT

Choice of Measurement Interval

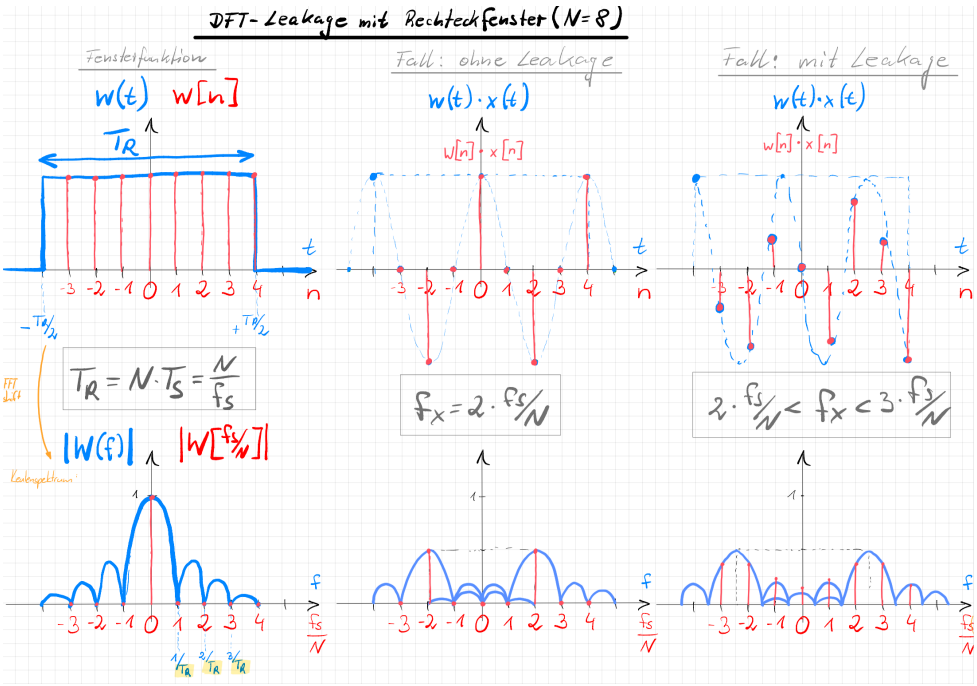


(a) (b)

Figure 9: 9b shows the DFT of a cosine with correct measurement interval ($N = 64$). 9b shows the application of the DFT with incorrect measurement interval which yields jumps in the virtual periodic extension of the signal being analyzed and thus **leakage** in the frequency domain occures.

Leakage

The *leakage-effect* manifests itself in the main frequency "leaking" into several frequency indexes surrounding the two true frequency points (see Figure 9).



DFT and Zero-padding

To "increase" spectral resolution zero padding is added.

> **Zero-padding**
>
> Zero-padding **won't** increase spectral resolution, it just corresponds to a better interpolation between the N frequency points of the DFT spectrum.
> To increase spectral resolution, additional samples of the signal $x[n]$ must be sampled.

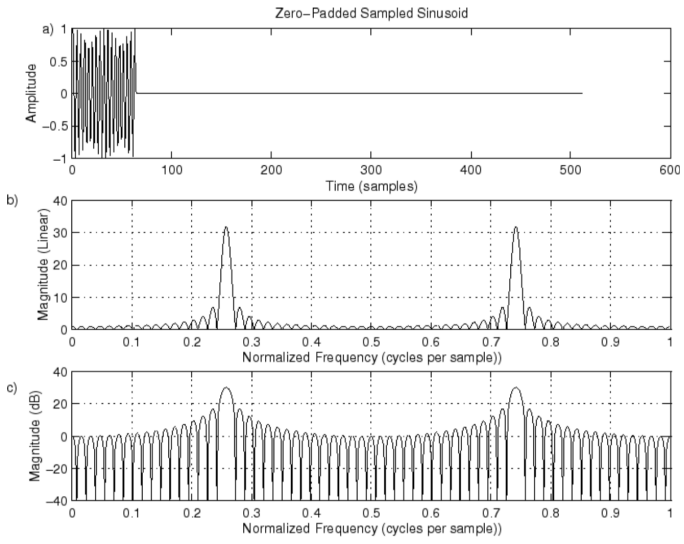
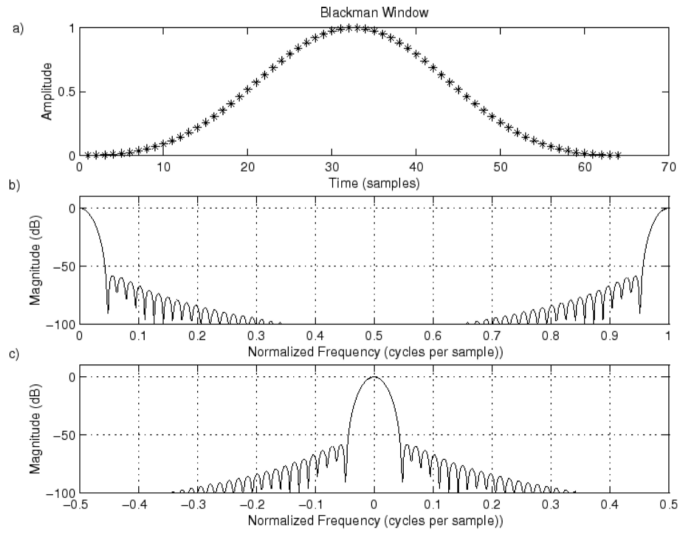


Figure 10: Zeropadded signal in time & frequency domain

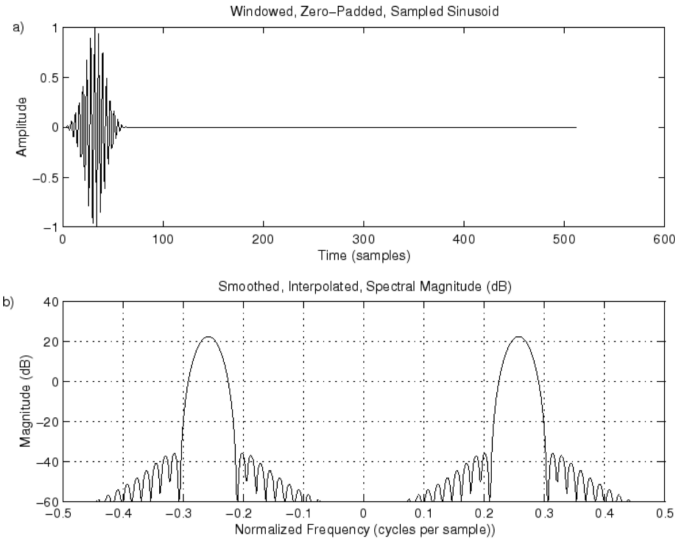
The finite measurement interval of the DFT corresponds to multiplying the signal $x[n]$ with a *rect*-Window, which corresponds to a a convolution of $X(\Omega)$ with the $\frac{\sin(x)}{x}$-function, thus introducing the form of lobes. This lobe-structure is independent of the form of the signal waveform $x[n]$.

DFT and Windowing

To overcome the problem of lobe-introduction, we apply **windowing**. A window can be any function, that is applied to the signal in the measurement interval.



(a)



(b)

Figure 11: 11a shows the Blackman Window for $N = 64$ in time and frequency. The Signal from Figure 10 with a Blackman window applied before zero-padding shows much smaller lobes in frequency domain (Figure 11b)

Effects introduced by Windowing

- Comparing the original signal Figure 10 to the windowed in Figure 11b that the first side lobe is significantly smaller ($12dB \rightarrow -60dB$), so the *leakage is minimized*
- The width of the main lobe on the other hand has become wider ($0.03 \rightarrow 0.1$ cycles per sample), which equals to the *reduction of the frequency resolution*

Windowing functions Different windowing functions will have different influence on the time and frequency domains of signals and should be chosen there for.

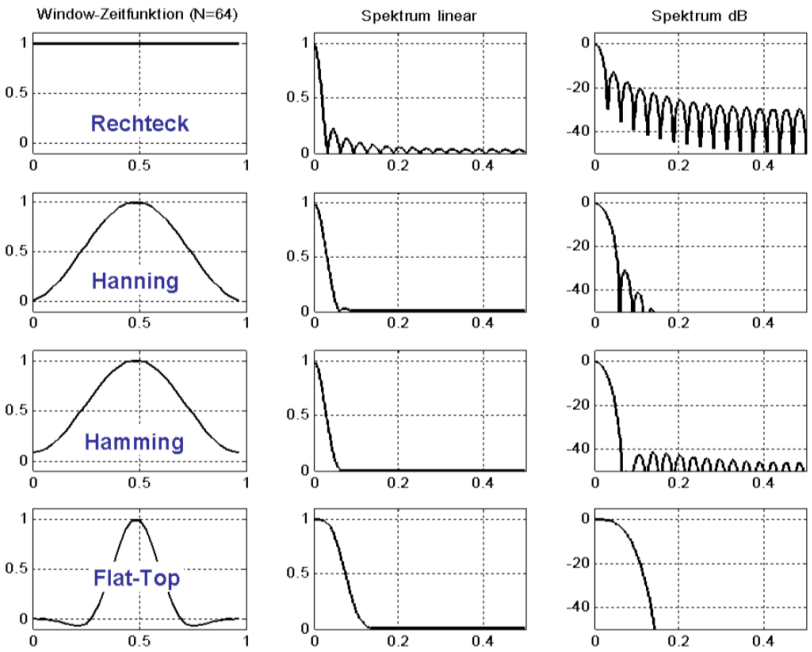


Figure 12: Comparison of windowing functions in the time and frequency domain

Short-Time DFT

When assessing the development of the frequency spectrum over time a **continuous** evaluation of the frequency spectrum of short signal sections is required **(short-time DFT)**.

> ! short-time DFT
>
> Since such signal sections are unlikely to fit a signal period perfectly, windowing is applied, which results in *less leakage* but a *convolution of the signal with the window spectrum*.
> When setting the length N of the window, one compromises between:
> - high spectral resolution *(N large)*
> - high time resolution *(N small)*

*(*For both **good spectral and time resolution** we have to overlap the DFT windows, see Figure 13. In the limit, we could overlap two consecutive DFT windows by $N-1$ sample values *if computationally feasible*.

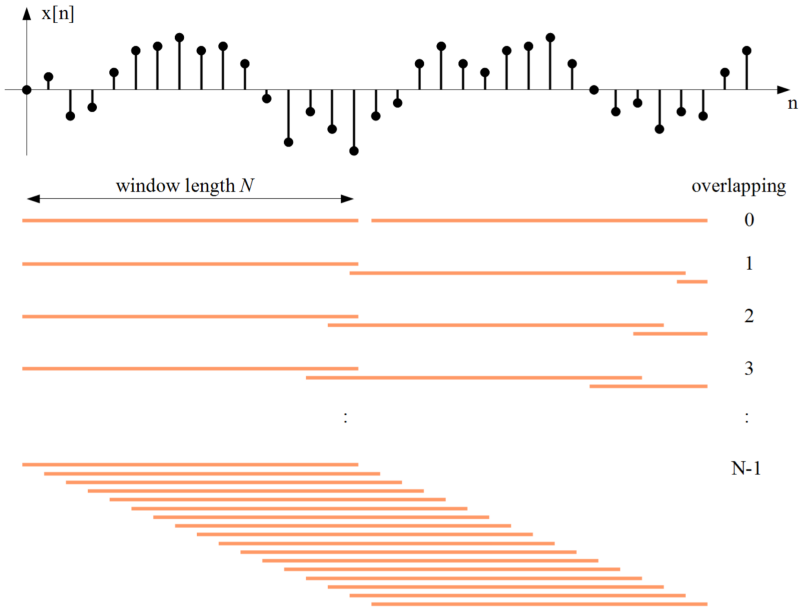


Figure 13: **Overlapping of windows in the short-time DFT**

Fast Fourier Transform FFT

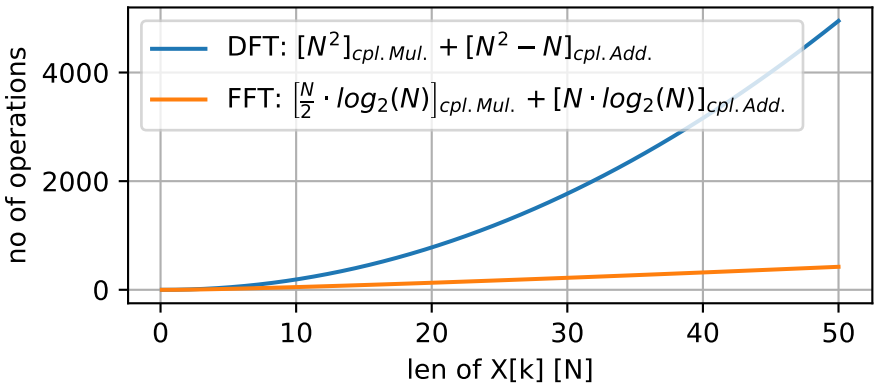
Computational effort DFT vs. FFT

To calculate a spectrum through DFT ($x * y$)

$$[N^2]_{cpl.Mul.} + [N^2 - N]_{cpl.Add.}$$

To calculate a spectrum through FFT ($ifft(fft(x) \cdot fft(y))$)

$$\left[\frac{N}{2} \cdot log_2(N)\right]_{cpl.Mul.} + [N \cdot log_2(N)]_{cpl.Add.}$$



That gives us a speedup factor (assumption: real addition= real multiplication) of

$$\frac{8N - 2}{5 \cdot log_2(N)} \approx 1.5 \frac{N}{log_2(N)}$$

Twiddle Factors

The **twiddle factors** are the building part of every complex harmonic sequence in a N-point DFT and are defined as

$$W_N = e^{-j\frac{2\pi}{N}}$$

Periodicity: W_N^k can evaluate N different numpery only $\rightarrow$ N-periodic:

$$W_N^{k+N} = W_N^k$$

Symmetry: Apart from the sign, W_N^k only holds $\frac{N}{2}$ numbers:

$$W_N^{k+\frac{N}{2}} = -W_N^k$$

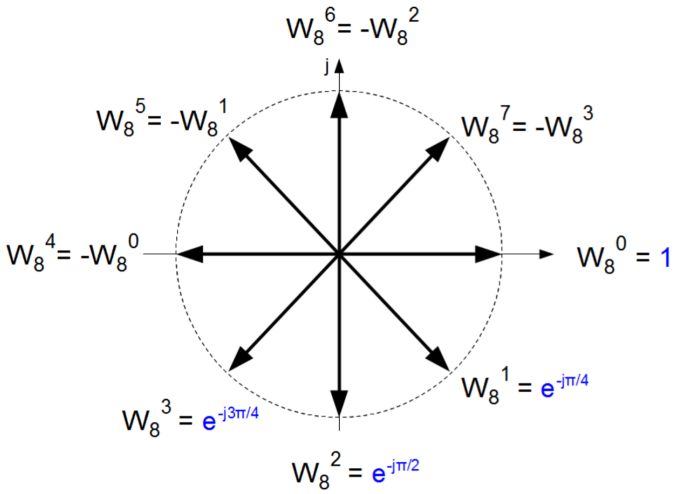


Figure 14: Example for $N = 8$

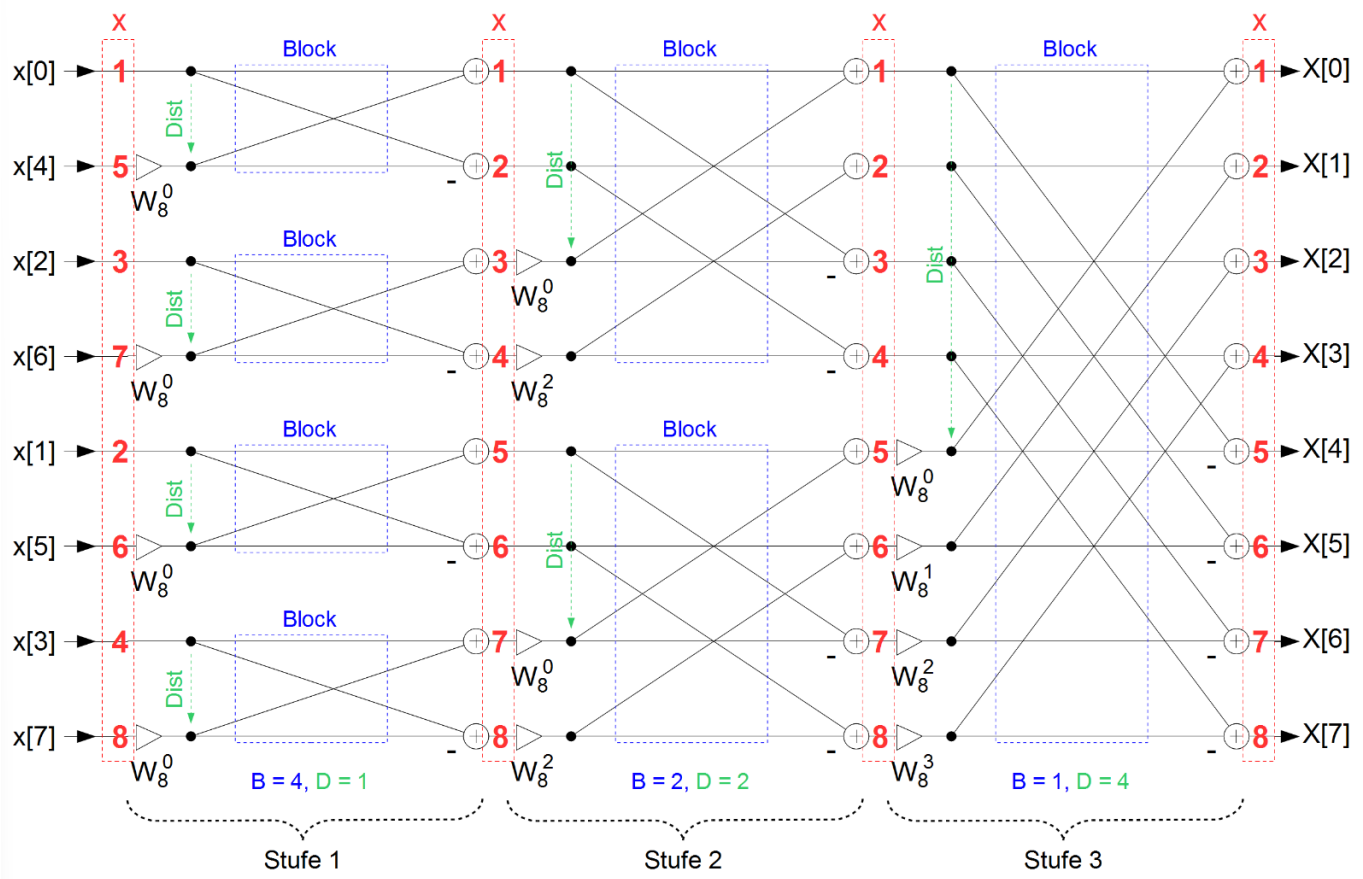
Butterfly operation

Using the twiddle factors we can process a FFT much faster. For this we obtain the **Radix-2 decimation-in-time.**

$$\begin{aligned}
X[k] &= \sum_{n=0}^{N-1} x[n] e^{-j2\pi n\frac{k}{N}}, \qquad k = 0, 1, \ldots, N-1 \\
&= \sum_{n=0}^{N-1} x[n] W_N^{nk} \\
&= \sum_{m=0}^{N/2-1} x[2m] W_N^{2mk} + \sum_{m=0}^{N/2-1} x[2m+1] W_N^{(2m+1)k} \qquad \left| W_N^2 = W_{N/2} \right. \\
&= \underbrace{\sum_{n=0}^{N/2-1} x_1[n] W_{N/2}^{nk}}_{X_1[\tilde{k}]} + W_N^k \cdot \underbrace{\sum_{n=0}^{N/2-1} x_2[n] W_{N/2}^{nk}}_{X_2[\tilde{k}]} \qquad \left| \tilde{k} = k \mod N/2 \right.
\end{aligned}$$

$x_1[n]$ and $x_2[n]$ contain samples of $x[n]$ with even and odd time index, respectively.

Through this we get a calculation architecture that is built by *2-point-FFTs*



Allgemein kann das Vorgehen so beschrieben werden:

- Werte in *bit-reversed*-Reihenfolge in Verarbeitung geben (Stufe 1), anschliessend linear im Speicher halten *(Endergebniss stimmt ohne weitere umsortierung)*
- Aus verarbeitungsstufen $s = 1 \dots \log_2(N)$ die Anzahl Blöcke $B = \frac{N}{2^s}$ und Distanz $D = 2^{s-1}$ berechnen
- Inidizes $i_1 = (b-1) \cdot (2^s) + d$ und $i_2 = i_1 + D$ berechnen
- Index für Drehfaktor $i_W = (d-1) \cdot B + 1$ berechen
- Butterfly auf berechnete indizes anwenden

```
W = exp(-1j*2*pi*[0:N/2-1]/N);  % twiddle factors (only N/2 because of symmetry)

x = bitrevorder(x); % bit-reversed order to allow in-place computation
for s = 1:log2(N)   % for each stage to do ...
    B = N/(2^s);        % get # of blocks in current stage
    D = 2^(s-1);        % get # of butterflies per block in current stage
    for b = 1:B         % for each block in current stage to do ...
        for d = 1:D         % for each buttergly in current block to do ...
            i1 = (b-1)*(2^s)+d;     % get 1st operand idx
            i2 = i1+D;              % get 2nd operand idx
            iW = (d-1)*B+1;         % get twiddle factor idx
            [x(i1) x(i2)] = butfly(x(i1),x(i2),W(iW));      % compute butterfly in-place
        end
    end
end
```

Efficient FFT implementation

The entire FFT can be performed **in-place**, this is because the memory place of the input-pair can be filled after a butterfly execution with its result. Thus only $2N$ memory locations (N complex values) plus $N/2$ for the twiddle factors are needed. Additionally by the application of **bit-reversed addressing** the algorithm can work much faster and the result is correct without reordering.

linear order		bit-reversed order	
decimal	binary	binary	decimal
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

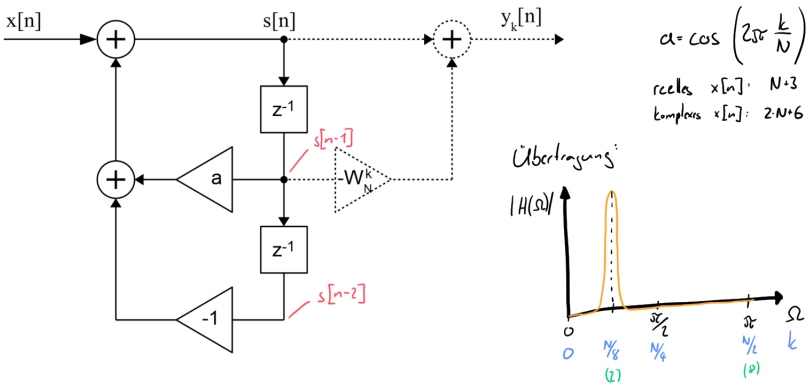
Figure 15: Bit-reversed order for $N = 8$. Indeces with symmetric binary representation remain unchanged.

Goertzel Algorithm

If only individual $X[k]$ out of all N spectral components are needed, the *Goertzel algorithm* can be applied. This is also a linear filtering technique. The IIR-system of difference equations is

$$s[n] = x[n] + a \cdot s[n-1] - s[n-2], \qquad a = 2\cos\left(2\pi\frac{k}{N}\right) \tag{0.2}$$

$$y_k[n] = s[n] - W_N^k \cdot s[n-1] \tag{0.3}$$



Parseval theorem

Goertzel gives us the DFT spectral component at frequency

$$f_k = k\frac{f_S}{N}$$

Through Parsevals theorem we get the power content P_k in a real-valued signal $x[n]$ around f_k

$$P_k = 2\left|\frac{X[k]}{N}\right|^2 = \frac{2}{N^2}\left(\Re\{X[k]\}^2 + \Im\{X[k]\}^2\right)$$

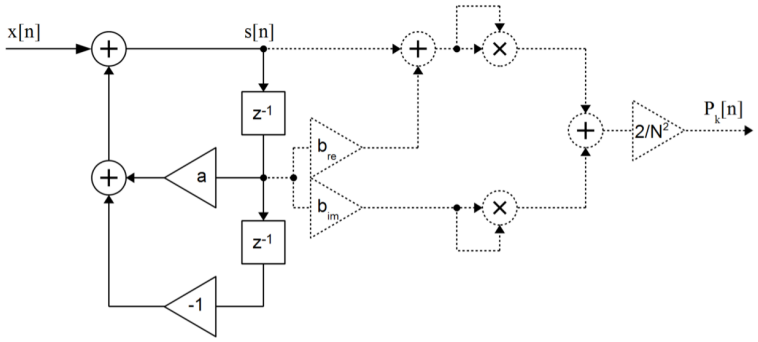


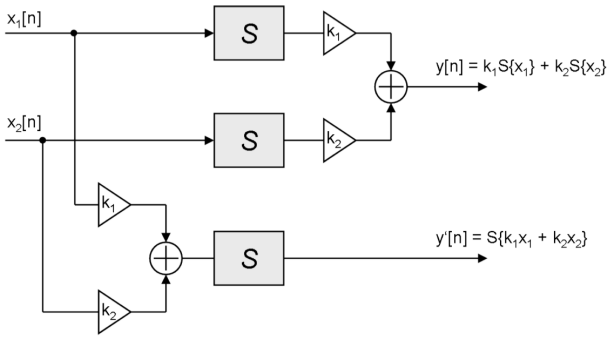
Figure 16: With $b_{re} = \Re\{-W_N^k\}$, $b_{im} = \Im\{-W_N^k\}$

Digital LTI Systems

Definition

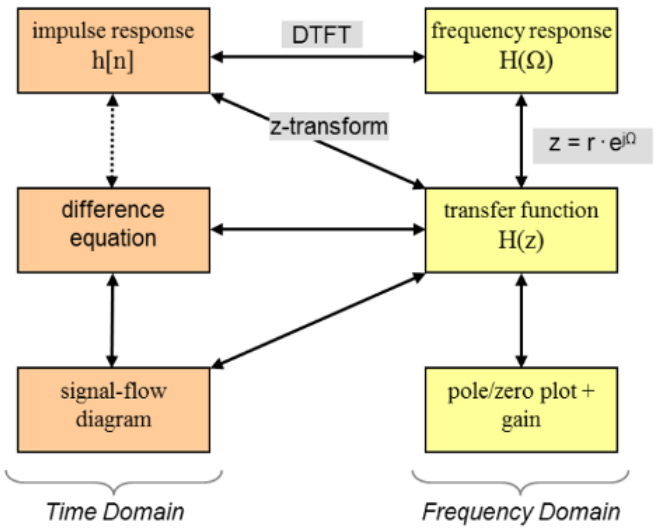
Linearity:

- Multiplication of a signal with a constant $y[n] = a \cdot x[n]$
- Addition of two signals $y[n] = x_1[n] + x_2[n]$



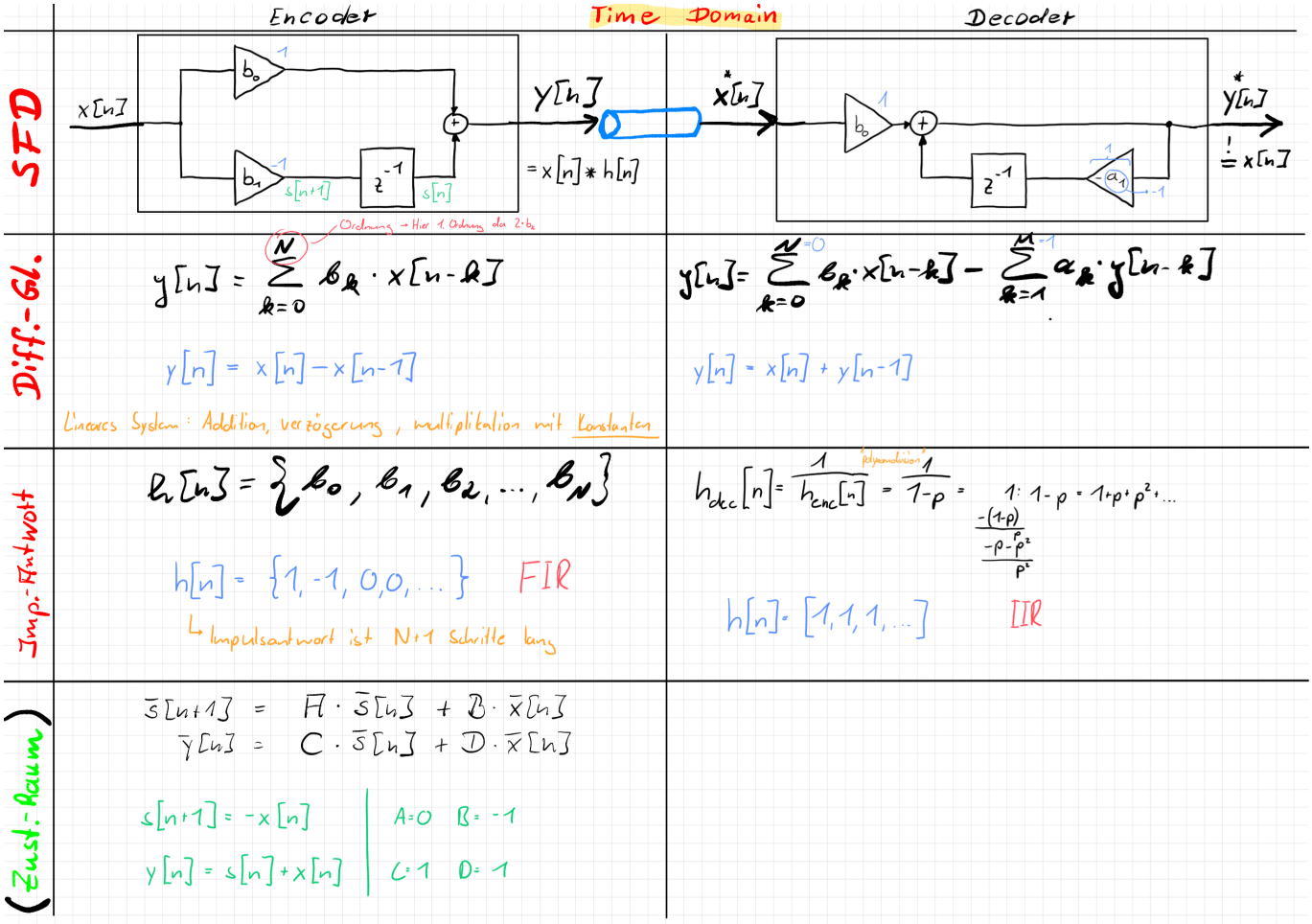
Time-Invariance: Time delay of a signal by $d = k \cdot T_S$, $x[n] \rightarrow y[n] = x[n-d] \rightarrow y[n-d]$

Description of LTI systems



Time Domain	**impulse response**	system identification, measurement
	difference equation	system implementation (algorithm)
	signal-flow diagram	system implementation (architecture)
Frequency Domain	**transfer function**	coupling analysis and implementation
	pole/zero plot	intuitive analysis and design
	frequency response	system identification, analysis and design

System Descriptions in the Time Domain



Impulse response

```
% plot impulse response of a system
impz([b0 b1 ...],[a0 a1 ...], n) % a, b: transfer function coeff, n: no. of samples
```

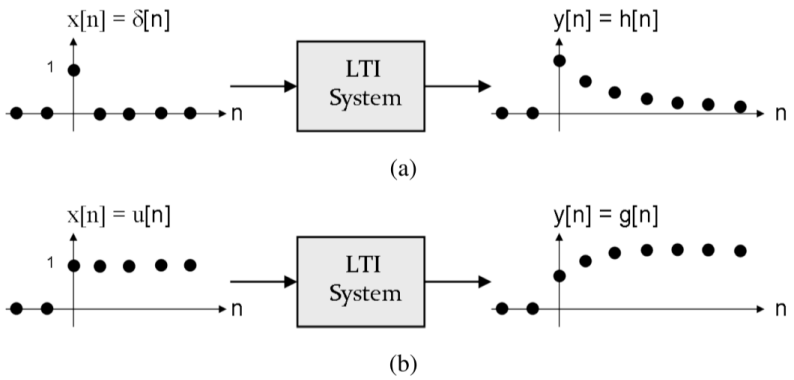


Figure 17: Impulse response (a) and step response (b)

A *LTI*-System is reacts to each weighted unit impulses $x_S(i) \cdot \delta[n-i]$ with a weighted impuls response $x_S(i) \cdot h[n-i]$. Through superposition we obtain the systems reaction

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

Dies entspricht der linearen Faltung.

Difference Equation

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

$max(N.M)$ is the **order** of the system. A system $M \geq 1$ is **recursive**. Directly convertible into *Signal-Flow Diagram*.

Signal-Flow Diagram

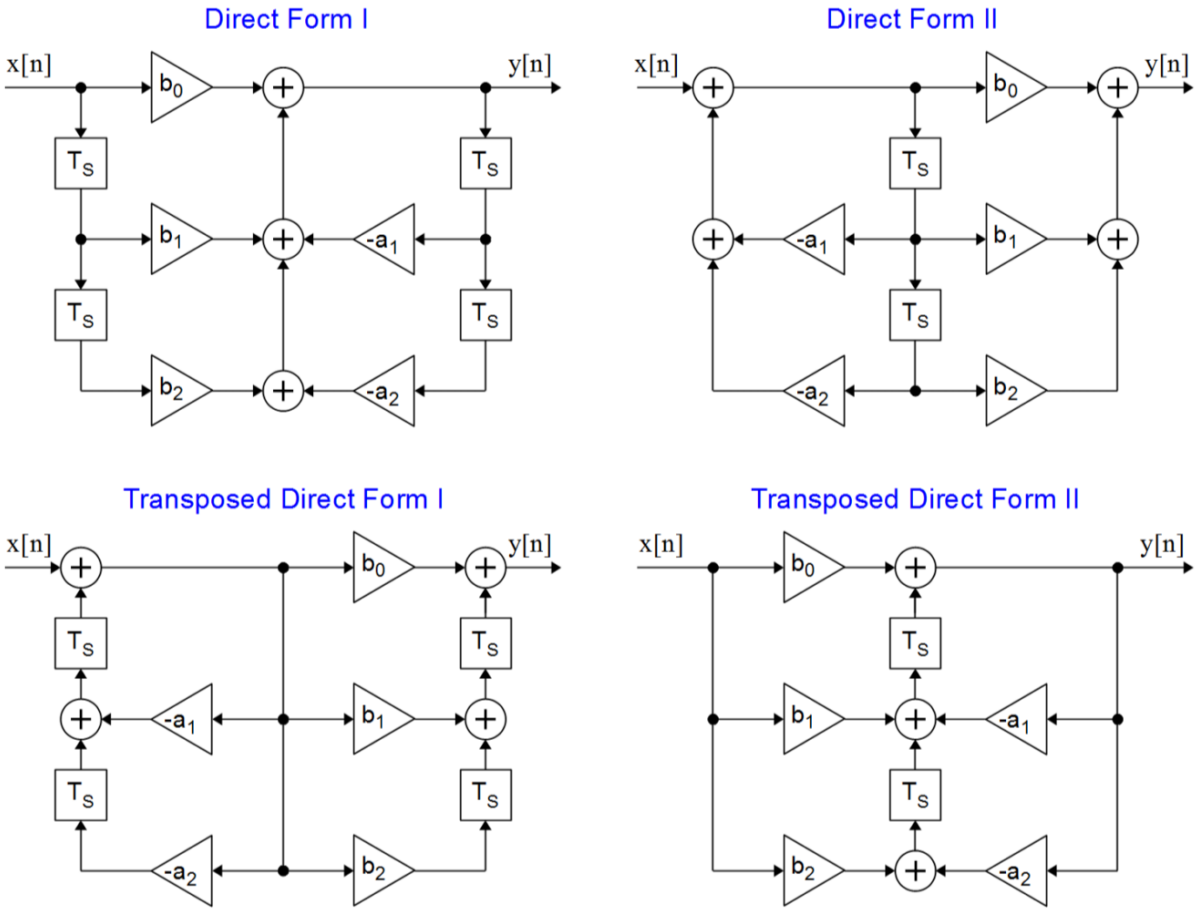
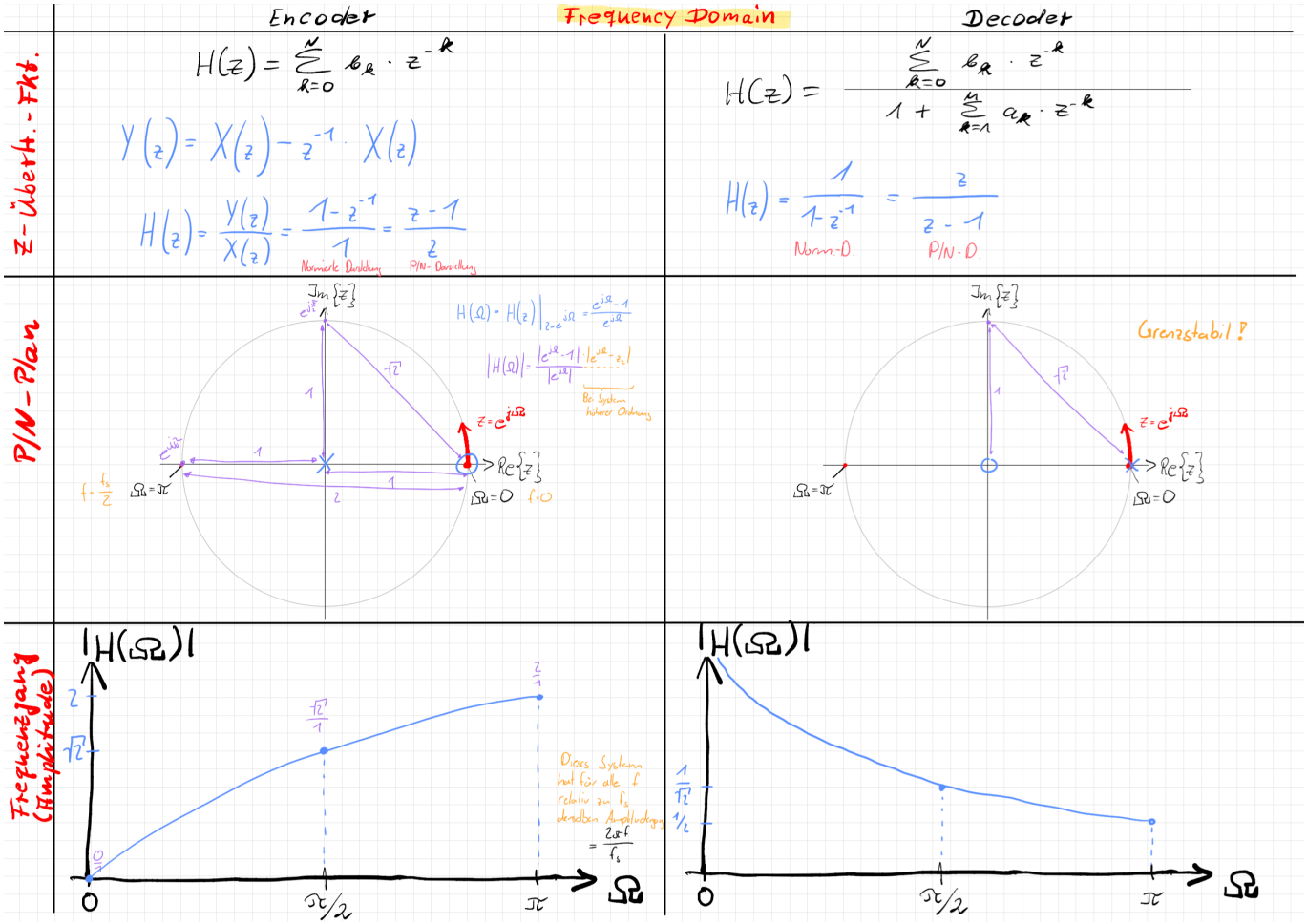


Figure 18: Normalized forms of signal-flow diagrams for digital LTI-systems

System Description in the Frequency Domain



Transfer Function

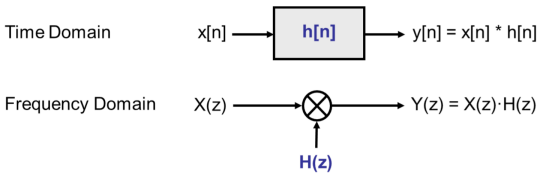
$$y[n] = \sum_{k=0}^{N} b_k x[n-k] - \sum_{k=1}^{M} a_k y[n-k] \circ\!\!-\!\!\bullet Y(z) = \sum_{k=0}^{N} b_k z^{-k} X(z) - \sum_{k=1}^{M} a_k z^{-k} Y(z)$$

by reordering we obtain the **z-transfer-function** *(normalized if $a_0 = 0$)*

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_N z^{-N}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_M z^{-M}}$$

The transfer function $H(z)$ of a digital LTI system is the z-transform of its impulse response $h[n]$, thus we get

$$y[n] = x[n] * h[n] \circ\!\!-\!\!\bullet Y(z) = X(z) \cdot H(z)$$



Pole/Zero-Plot

```
% plot the zero-poles-diagram
zplane(z,p) % with zeros and poles as in z=[z0 z1 ...], p=[p0 p1 ...]
zplane(b,a) % with transferfunction coeffs
```

To analyze the system behavior we use the **pole/zero-form**

$$H(z) = K_0 \cdot \frac{(z-z_1)(z-z_2)\cdots(z-z_N)}{(z-p_1)(z-p_2)\cdots(z-p_M)} \cdot z^{M-N}$$

Tip

- $N > M$: there are $N - M$ additional poles at $z = 0$. If $b_0 \neq 0$, the inverse is true!
- $M > N$: there are $M - N$ additional zeros at $z = 0$. In this case $K_0 = b_0$ is true

Stability

A causal digital LTI-system is stable if all poles p_i are located within the unit circle.

Frequency Response

```
% Plot Bode-Diagram of a system
freqz(b,a) % a,b: transferfunction coeffs
```

Systembehavior as a function of its input signal *(Bode-Plot)*. **Measure**: Frequency sweep over all frequencies, **Indirect Measurement**: obtain impulse response and transform into frequency response $h[n] \circ\!\!-\!\!\bullet H(\Omega)$. The frequency response turns out to be f_S or 2π periodic and complex valued. We represent the frequency response in polar coordinates:

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

Where the **amplitude response** is mostly written in dB

$$|H(\Omega)|_{dB} = 20 \cdot \log_{10}(|H(\Omega)|$$

For a *distortionfree* transmission a **linear phase response** $\varphi(H(\Omega)) = -K \cdot \Omega$ is desirable, thus will make all frequencies undergo the sampe time delay. This delay is called the **group delay** and is defined as

$$\tau_g = -\frac{d\varphi(H(\Omega))}{d\Omega} \cdot T_S = \frac{K}{2\pi}$$

A LTI-System will react to a sinusoidal input with a sinusoidal output of the same frequency

$$x[n] = \cos(2\pi f_0 n T_S) \rightarrow y[n] = |H(\Omega_0)| \cdot \cos(2\pi f_0 n T_S + \varphi(H(\Omega_0)))$$

We can obtain the *amplitude* through (Attention: Add in dB!)

$$|Y(\Omega)| = |X(\Omega)| \cdot |H(\Omega)|$$

and the *phase*

$$\varphi(Y(\Omega)) = \varphi(X(\Omega)) + \varphi(H(\Omega))$$

Relation between frequency response and transfer function

In general: $z = re^{j\Omega}$, which gives us

$$H(\Omega) = H(z)|_{z=e^{j\Omega}}$$

We can obtain the bode plot from the pole/zero-form where the **amplitude** is

$$|H(z)| = |K| \cdot \frac{|(z-z_1)||(z-z_2)|\cdots|(z-z_N)|}{|(z-p_1)||(z-p_2)|\cdots|(z-p_M)|} \cdot |z|^{M-N}$$

and the **phase**

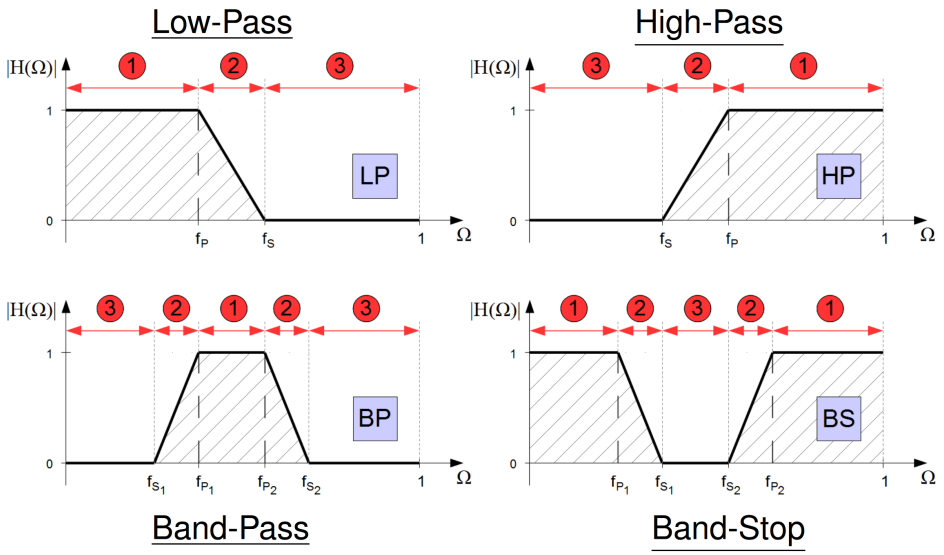
$$\varphi(H(z)) = \sum_{k=1}^{N} \varphi(z-z_k) - \sum_{k=1}^{M} \varphi(z-p_k) + \sum_{k=N+1}^{M} \varphi(z)$$

Design of Digital Filters

For a Filter we need to determine the following **filter parameters**:

- b_k and a_k *filtercoefficients*
- N and M *filter order*

Filters can take the following forms:



1 = Pass Band, 2 = Transition Band, 3 = Stop Band

criteria	FIR	IIR
stability	always stable	not guaranteed, must be verified for actual implementation
filter performance / filter order	higher order for same amplitude response	lower filter order for same amplitude response
phase response	exactly linear possible	linear phase not possible, introduces phase distortion
Arbitrary frequency response	easy to realize	more difficult to realize
design process	optimum design only with CAD, manual design possible	basic filter characteristics can be designed manually
design with analog filter model	no direct conversion possible	standard design method
finite word width effects	relatively low	sensitive to quantization and scaling
implementation	non-critical in HW and SW, direct support in DSPs	more complicated, requires detailed analysis of word widths

FIR-Filter

Definition

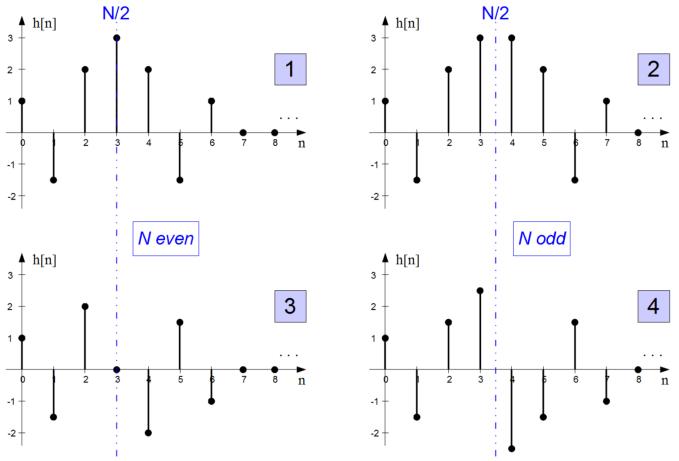
- $M = 0$, finite impulse response *(FIR)*
- $H(z) = b_0 + b_1 z{-}1 + \cdots + b_N z{-}N$
- impulse response is $N + 1$ time steps long
- impulse response corresponds to the coeffs $h[n] = \{b_0, b_1, \ldots, b_N, 0, 0, \ldots\}$
- **Stability**: all poles are $z = 0$, FIR is stable by definition
- **Linear Phase**: Easy → constant group delay
- **Implementation**: Realization in HW or SW straightforward and noncritical
- **Drawback**: High filter order N needed for often desired output → computational effort / signal delay

Symmetric FIR Filters

FIR Filter is called *symmetric* if the coefficients hold

$$\underbrace{b_i = b_{N-i}}_{\text{symmetric}} \quad \text{or} \quad \underbrace{b_i = -b_{N-i}}_{\text{anti-symmetric}} \quad \text{with } i = 0, 1, \ldots, N$$

The length of N defines a filter further as *even* or *odd*.



Type	Symmetry	Order N	$\lvert H(f=0)\rvert$	$\lvert H(f=f_S/2)\rvert$	$H(\Omega)$
1	$h[n] = h[N-n]$ (symmetric)	even	any	any	$e^{-j\Omega\frac{N}{2}} \cdot H_{zp}(\Omega)$
2		odd	any	0	
3	$h[n] = -h[N-n]$ (anti-symmetric)	even	0	0	$e^{-j\left(\Omega\frac{N}{2}-\frac{\pi}{2}\right)} \cdot H_{zp}(\Omega)$
4		odd	0	any	

Type	low-pass (LP)	high-pass (HP)	band-pass (BP)	band-stop (BS)
1	yes	yes	yes	yes
2	yes	–	yes	–
3	–	–	yes	–
4	–	yes	yes	–

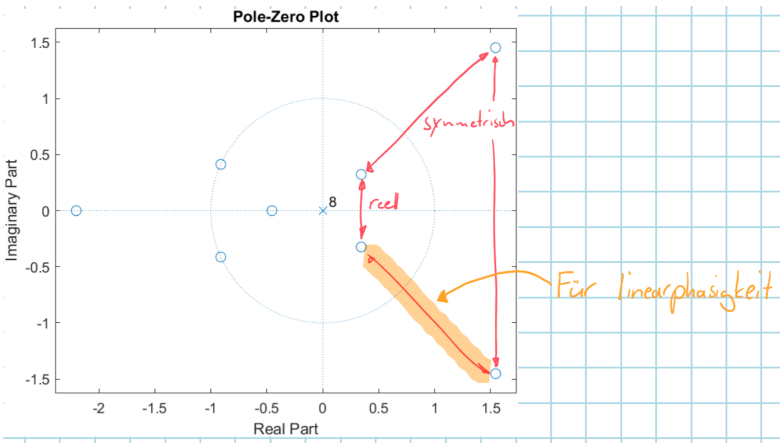
Figure 19: The four types of linear-phase FIR-filters. Type 3 and 4 feature a constant phase offset of 90°.

Symmetric FIR filters have a constant group delay of $N/2$ sample times, all frequencies get delayed by the same amount

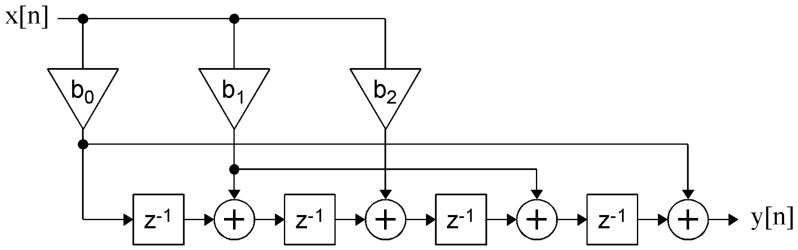
$$\tau_g = \frac{N}{2} \cdot T_S$$

180°-Phase-Jumps

In the stop bans of a symmetrical FIR filter, 180°-phase-jumps can occur. This is because of *complex-conjugate zeros at the unit circle*.
The jumps can be eliminated by *duplicating the complex-nojugate pairs of zeros*, with the drawback of a higher filter order.



Symmetrical FIR filters can be implemented with $\frac{N}{2}+1$ (N even) or $\frac{N+1}{2}$ (N odd) multiplications



Window Design Method

```
% design a FIR-filter with a Hamming Window
b = fir1(n,Wn,ftype) % n: filter order, Wn: frequency between 0 an 1 (pi rad/sample)
                     % ftype: 'low' | 'bandpass' | 'high' | 'stop' | 'DC-0' | 'DC-1'
```

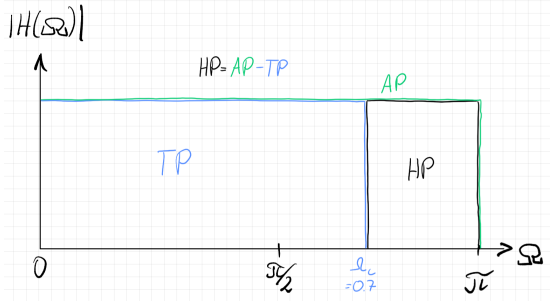
A ideal *lowpass* with a cut-off frequency f_c has a sinc-function as an ideal impulse response (2)

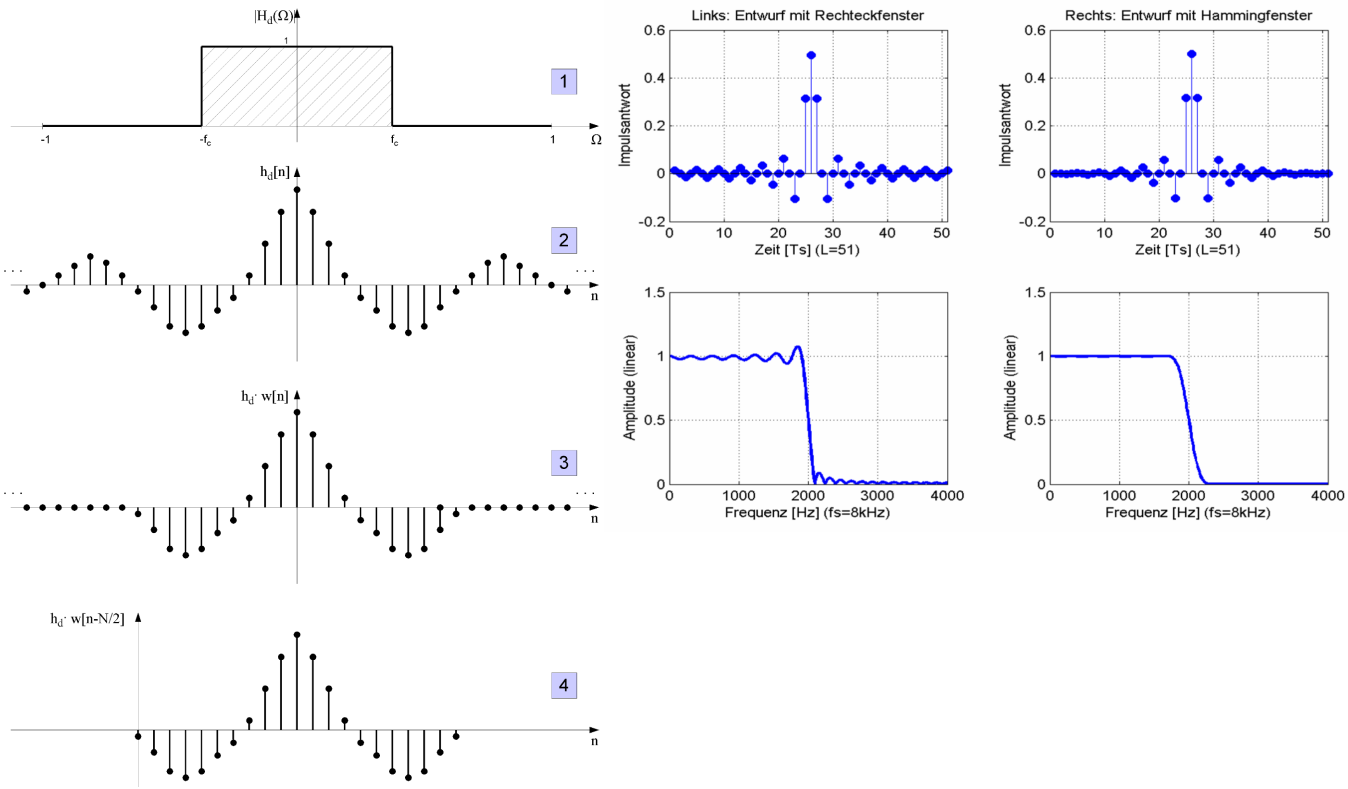
$$h_{d_{TP}}[n] = \frac{\sin(\Omega_c n)}{\pi n}$$

This response is infinitly long, to realize it, a **window is applied** ($1 \rightarrow 3$) and shifted by $\frac{N}{2}$ sample times (4).

Any other filter characteristic can be achieved by sum and difference of low pass filters at different cut-off frequencies. A **high-pass** filter at f_c can be realized with a low-pass at $f_c = \frac{f_s}{2}$ minus another low-pass with f_c.

$$h_{d_{HP}}[n] = \frac{\sin(\pi n)}{\pi n} - \frac{\sin(\Omega_c n)}{\pi n}$$





The choice of the window can influence the resulting frequency-response greatly.

IIR Filter

> Definition

- The order of the denominator polynomial in normalized form $M > 0$
- At least one pole at $z \neq 0$, infinite impulse response
- **Stability**: Potentially unstable
- **Linear Phase**: Not possible
- **Drawback**: high sensitivity on coefficient quantization

Design with Analog Prototype Filter

A systematic approach of designing IIR-Filters is by deriving the needed filter from a prototype filter. So you have to choose first a filter from one of the four basic types and work on from there.

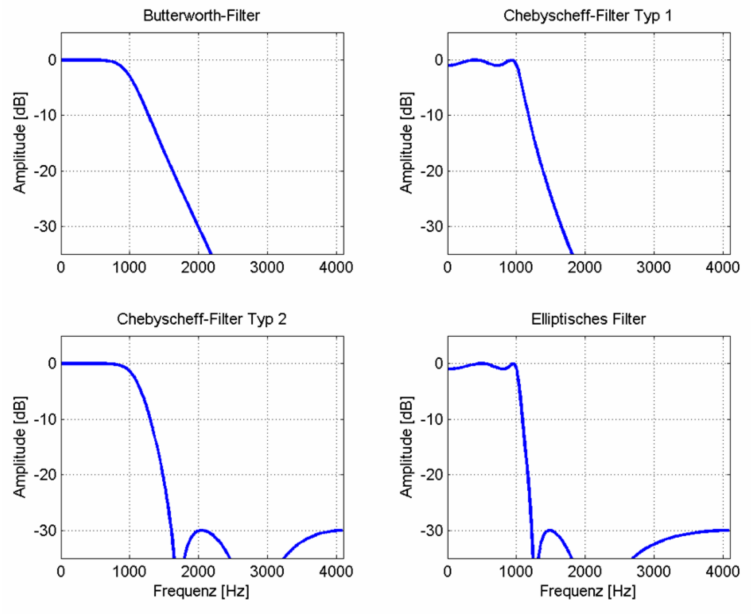
IIR Design by Approximation of the Differential Equation (rare)

Approximating the differential equation through a difference equation

$$\left.\frac{dy(t)}{dt}\right|_{t=nT_S} = \frac{y[n]-y[n-1]}{T_S} \quad \circ\!\!-\!\!\bullet \quad \underbrace{p}_{\text{Laplace}} = \underbrace{\frac{1-z^{-1}}{T_S}}_{\text{z-Transform}}$$

Thus, the analog differentiator corresponds to a 1st-order FIR system. For higher order derivatives the equation extends to

$$p^k = \left(\frac{1-z^{-1}}{T_S}\right)^k$$

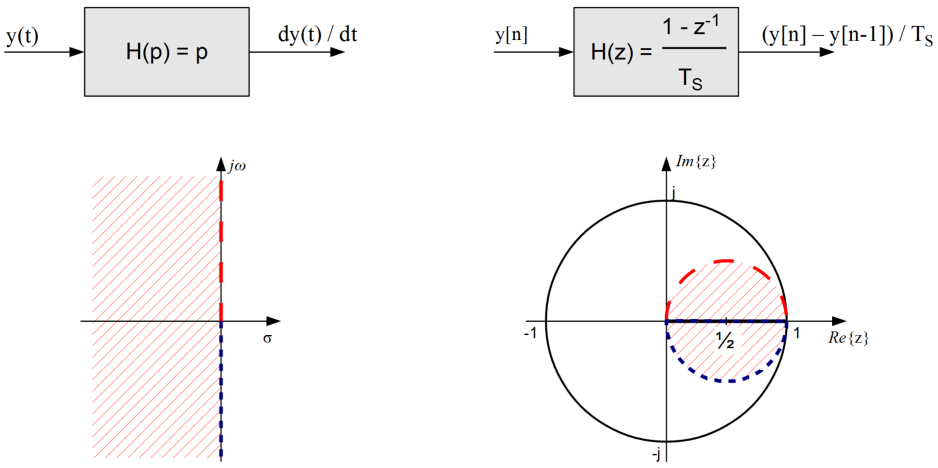


	Passband	Stopband	Steilheit	Phasen-Linearität	Ordnung
Butterworth	monoton	monoton	klein	gut	hoch
Chebyshev I	wellig	monoton	mittel	mittel	mittel
Chebyshev II	monoton	wellig	mittel	mittel	mittel
Elliptic (Cauer)	wellig	wellig	gross	schlecht	klein

We get the digital IIR filter from the analog transfer function through

$H(z) = H_a(p)|_{p=\frac{1-z^{-1}}{T_S}}$

Through $z = \frac{1}{1-pT_S}$ we get the always stable z-System



IIR Design by Impulse-Invariant Transformation (rare)

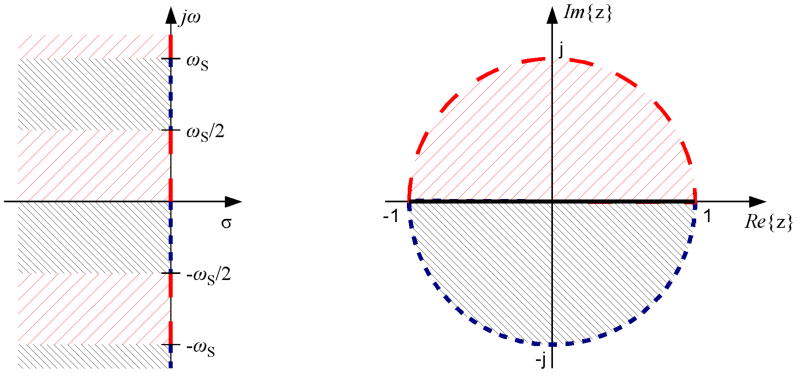
Transforming through sampling the impulse response of the continuous analog transfer function into the discrete impuls response of the IIR-Filter

$$h[n] = h_a(nT_S) \qquad n = 0, 1, 2, \ldots$$

Aliasing

To stay clear of aliasing, the sampling frequency f_S needs to be at least twice the highest pass-frequency of the analog prototype. Thus this method **can not transform high-pass and band-stop filter characteristics**.

Mapping through $z = e^{pT_S}$ we obtain

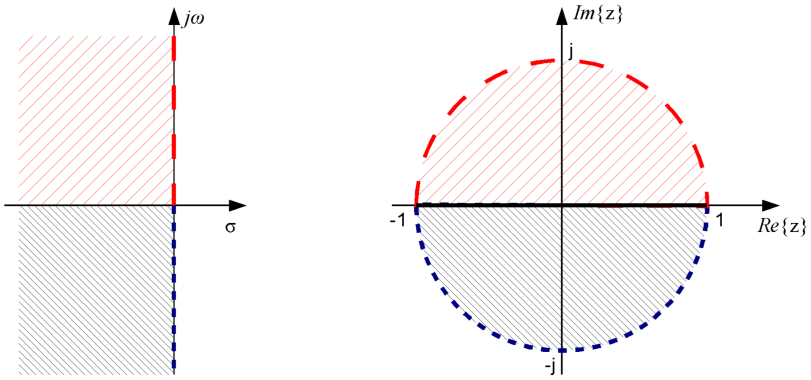


IIR Design by Bilinear Transformation (standard)

A unique mapping of the left half of the p-plane into the unit circle of the z-plane through bilinear transformation

$$z = \frac{2 + pT_S}{2 - pT_S}$$

So ist eine eindeutige, aber nicht lineare Abbildung des gesamten $j\omega$ Bereiches auf den Einheitskreis möglich



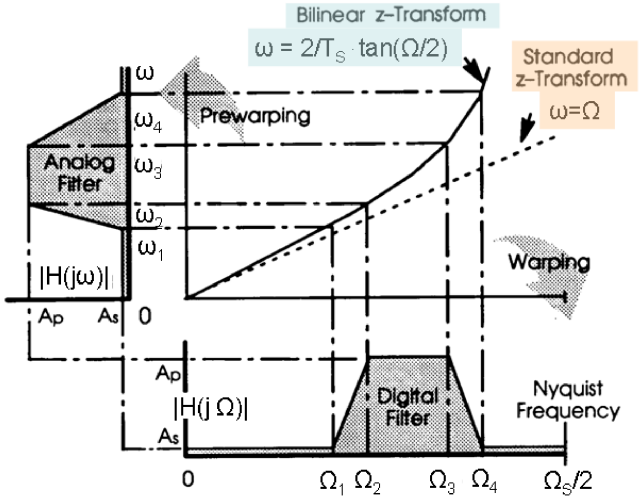
Prewraping

We see that the bilinear transformation maps the complete analog frequency range into $-\pi \cdots + \pi$

$$\Omega = 2 \arctan\left(\frac{\omega T_S}{2}\right)$$

This can be prevented by prewraping the frequency response through

$$\omega = \frac{2}{T_S} \cdot \tan\left(\frac{\Omega}{2}\right)$$



(note the longer transition band)

Filter Implementation Aspects

Choice of Sampling Frequency

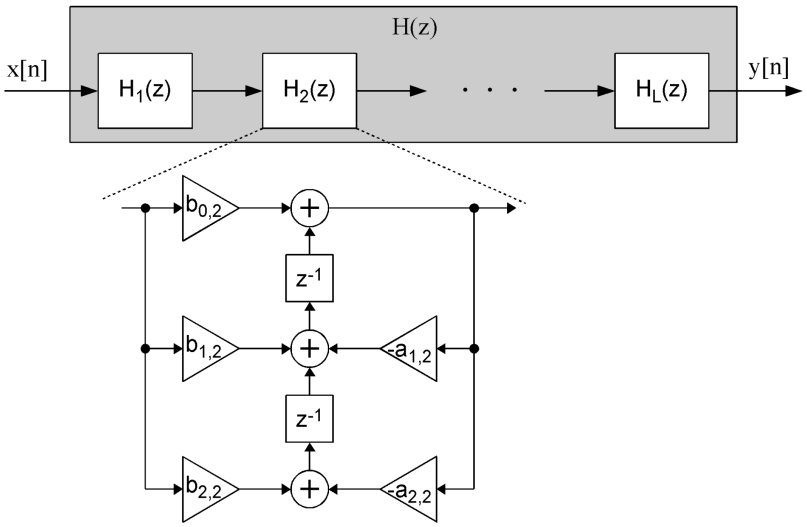
For a good system characteristic a sampling frequency f_S double the highest frequency component of the signal is recommended. On the other hand, the higher the sampling frequency...

- ... the bigger the computational effort
- ... the higher the slope of the filter, which implies higher order filters
- ... the closer the poles and zeros in the z-plane, which gives higher sensitivity to coefficient quantization effects

IIR Filter Implementation

To control the effects of coefficient quantization effects, higher-order IIR filters are usually implemented as **biquads** (secon-order sections or *SOS*)

$$H(z) = K \cdot H_1(z) \cdot H_2(z) \cdots H_L(z) \qquad = K \cdot \frac{(z-z_1)(z-z_1^*)}{(z-p_1)(z-p_1^*)} \cdot \frac{(z-z_2)(z-z_2^*)}{(z-p_2)(z-p_2^*)} \cdots \frac{(z-z_L)(z-z_L^*)}{(z-p_L)(z-p_L^*)} \tag{0.4}$$

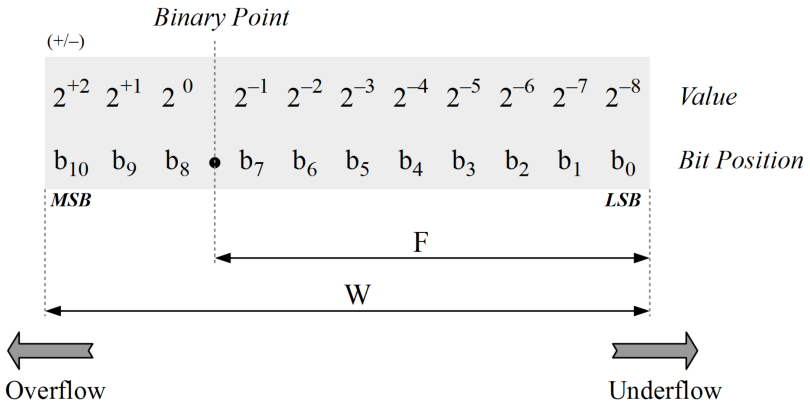


Given the coefficients of $H(z)$, obtain the SOS coefficients through **Matlab**

```
% get second order sections
[sos,g] = tf2sos(b,a) % a,b: filtercoeffs, g: overall gain
```

Fix-Point Implementation

Implementation by a fix-point number representation with the number of bits W and the fractional bits F

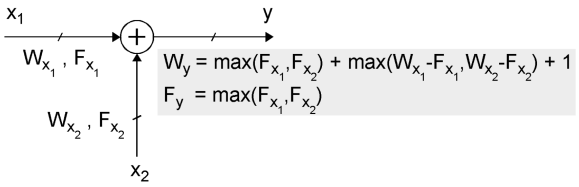


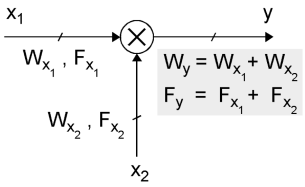
Results in the *unsigned* and *signed* values

$$D_{unsigned} = \sum_{k=0}^{W-1} b_k \cdot 2^{k-F} \qquad D_{signed} = -b_{W-1} \cdot 2^{W-F-1} + \sum_{k=0}^{W-2} b_k \cdot 2^{k-F}$$

Overflow: Result gets larger than largest representable number → saturation (limit to highest value), wrap-around (re-count from lowest value)

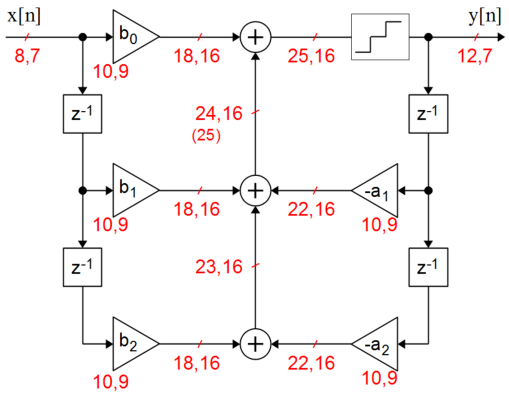
Underflow: lack of resolution, value to small to represent → trunctuation (neglect LSBs), rounding (round to nearest number representable)



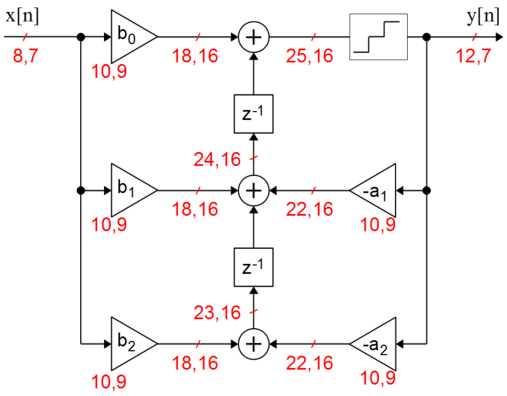


The needed memory by a implementation largely depends on basic architectures. This can be seen if you compare the implementation of the direct form I *(32 bits to be stored)* and transposed direct form II *(47 bits to be stored)*

Direct Form I



Transposed Direct Form II



Multirate Signal Processing and Filter Banks

Decimation

For reducing the sample rate of a signal downsample by selecting every Dth sample. But the **Abtasttheorem** must not be harmed!

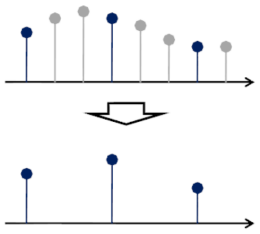
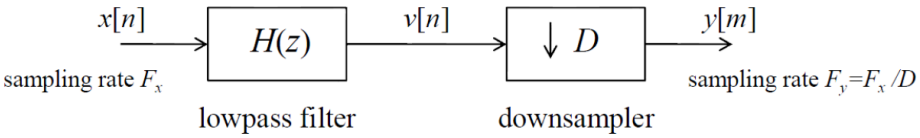


Figure 20: Downsampling by a factor of 3

To be safe withing the sampling theorems boundaries we apply a low-pass filter first.

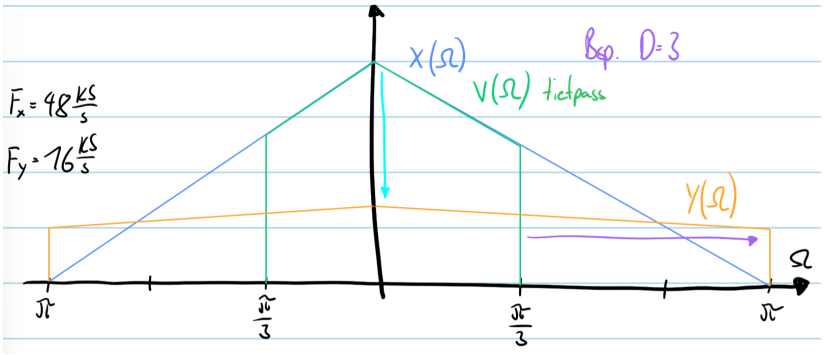


Where the low-pass filter ideally holds

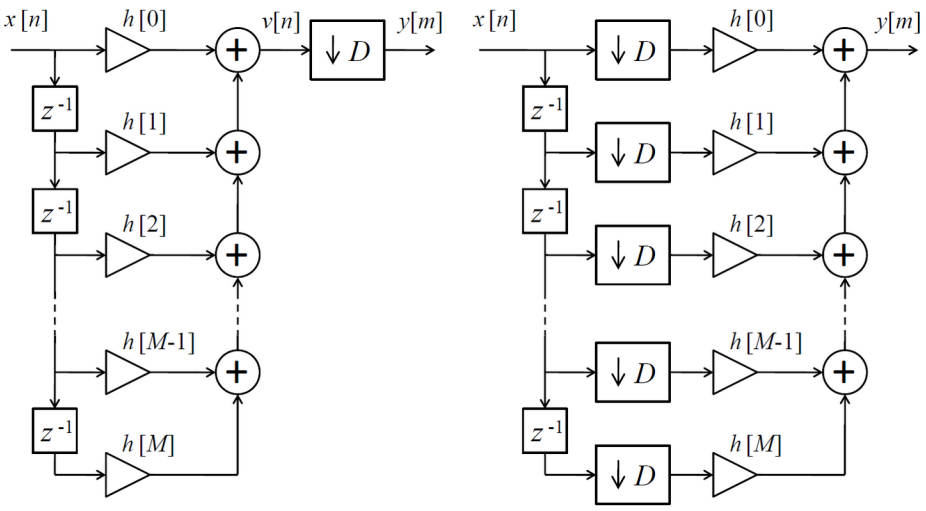
$$H(\Omega) = \begin{cases} 1 & \text{if } \Omega \in \left[-\frac{\pi}{D}, \frac{\pi}{D}\right] \\ 0 & \text{otherwise} \end{cases}$$

In the frequency domain the downsampling has the effect of spreading the spectrum by a factor D. In the case of an ideally lowpass filtered signal $v[n]$ the spectral input-output relation of the downsampler is simply

$$Y(\Omega) = \frac{1}{D} V\left(\frac{\Omega}{D}\right) \qquad \text{or for non-ideal lowpass filtering} \qquad Y(\Omega) = \frac{1}{D} \sum_{d=0}^{D-1} V\left(\frac{\Omega}{D} - \frac{2\pi d}{D}\right)$$



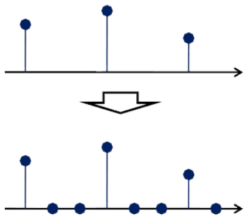
The implementation can be done through a direct implementation of a FIR filter of order M producing $v[n]$ where only every Dth sample of $v[n]$ is used. More efficiently the downsamplers are placed before the multiplication with the filter coefficients, thus they only have to work at the speed of the lower sample rate.



Interpolation

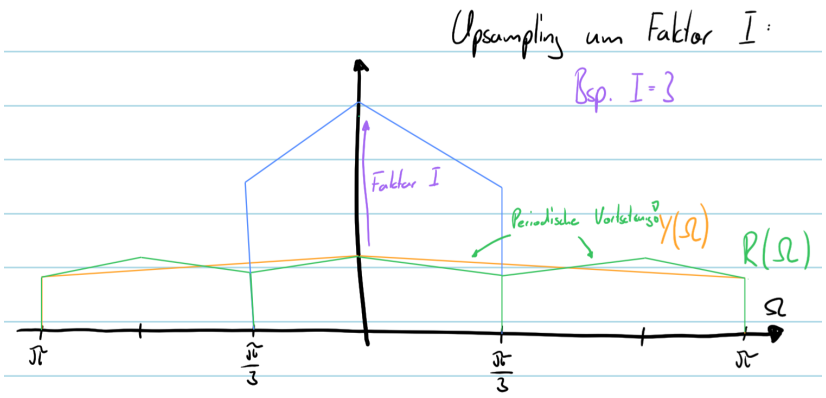
For increasing the sampling rate by the factor of I we simply insert $I-1$ zeros between any two samples of the discrete signal

$$r[n] = \begin{cases} y[n/I] & \text{if } n \in \{0, \pm I, \pm 2I, \dots\} \\ 0 & \text{otherwise} \end{cases}$$

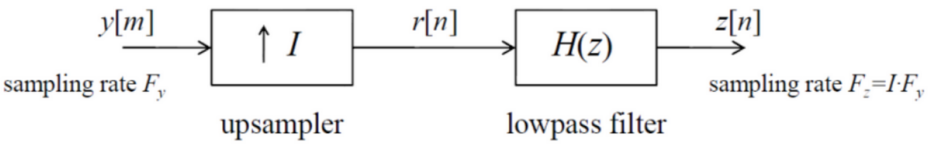


The spectrum of the upsampled signal becomes

$$R(\Omega) = \sum_{n=-\infty}^{\infty} r[n] e^{-j\Omega n} = \sum_{m=-\infty}^{\infty} y[m] e^{-j\Omega I m} = Y(I\Omega)$$



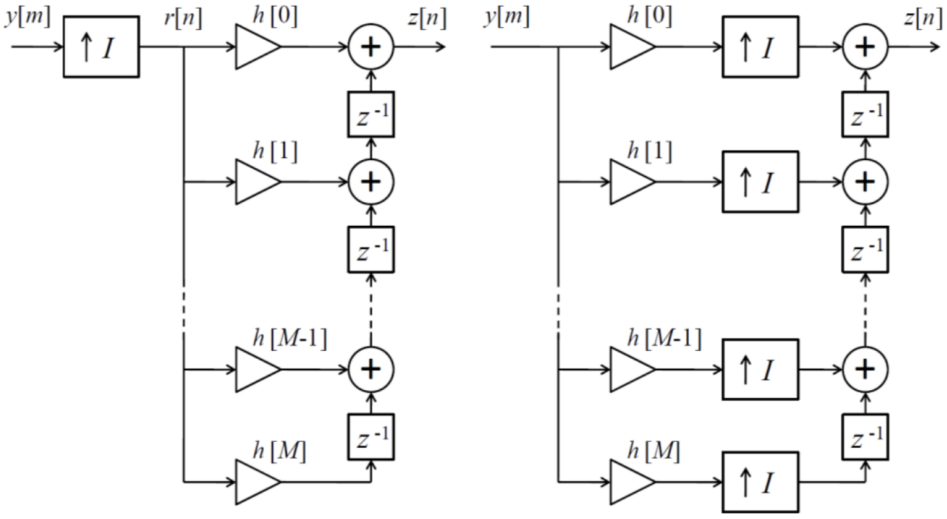
We can see, that the spectrum of $r[n]$ is $\frac{2\pi}{I}$ periodic. This can be prevented by a low-pass filter



Where the low-pass ideally holds

$$H(\Omega) = \begin{cases} I & \text{if } \Omega \in \left[-\frac{\pi}{I}, \frac{\pi}{I}\right] \\ 0 & \text{otherwise} \end{cases}$$

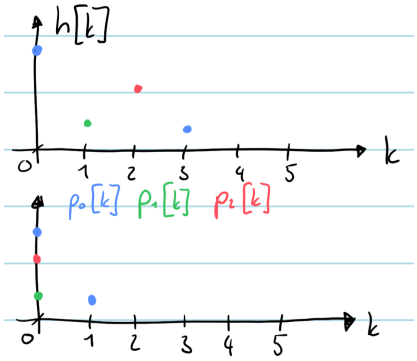
An FIR or IIR filter can be employed as lowpass filter. However, a direct implementation is again rather inefficient since $I-1$ out of every I sample values at the filter input are zeros. THere is a more efficient implementation in addition to the direct implementation involving an FIR filter. Note that in the more efficient implementation the multipliers operate at the lower sampling rate F_y rather than at F_z.



Polyphase Filter Structures

Polyphase structures are another concept towards more efficient filter implementations. First we define the impulse response of $h[k]$ by means of M downsampled polyphase variants $p_i[k]$

$$p_i[k] = h[kM + i], \quad i = 0, 1, \ldots, M-1 \tag{0.5}$$



The corresponding z-Transform is

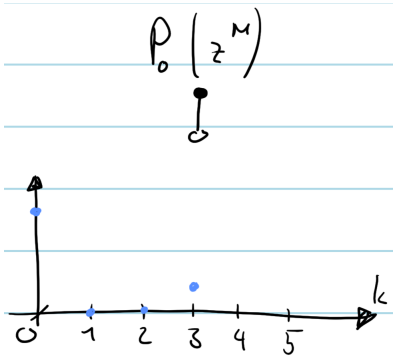
$$P_i(z) = \sum_{k=-\infty}^{\infty} p_i[k] z^{-k} = \sum_{k=-\infty}^{\infty} h[kM+i] z^{-k}, \quad i = 0, 1, \ldots, M-1$$

We note further that the z-Transform $H(z)$ of $h[k]$ can be written in terms of $P_0(z), \ldots, P_{M-1}(z)$ as

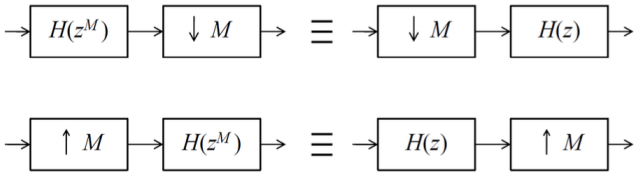
$$H(z) = \sum_{i=0}^{M-1} z^{-i} P_i(z^M)$$

Noble identities

Note that for every term $P_i(z^M)$ holds

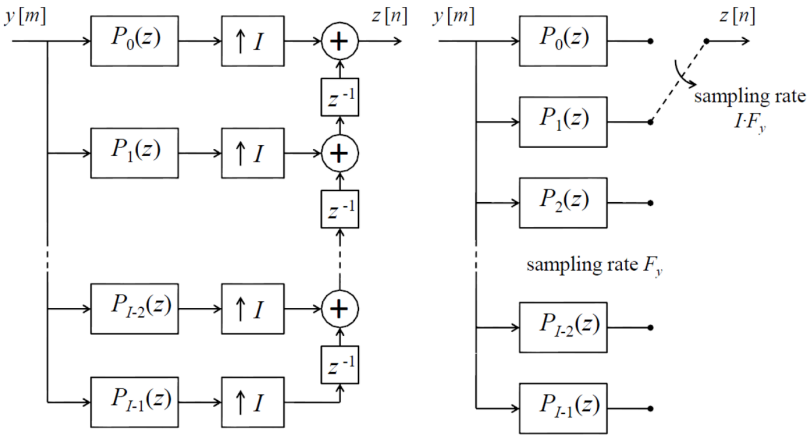


And that the *Noble identities* hold



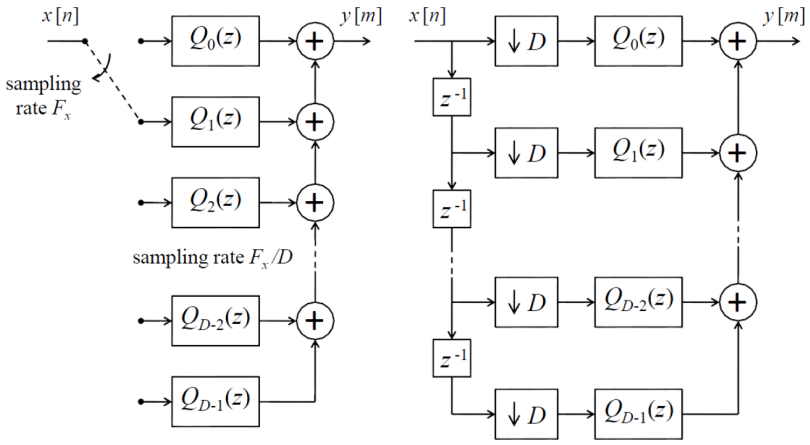
Implementation Interpolation

Durch das Auftreten der Samplewerte nut noch an den Positionen $M = I$ kann über den Schalter eine gute Tiefpass-Charackteristik erzielt werden



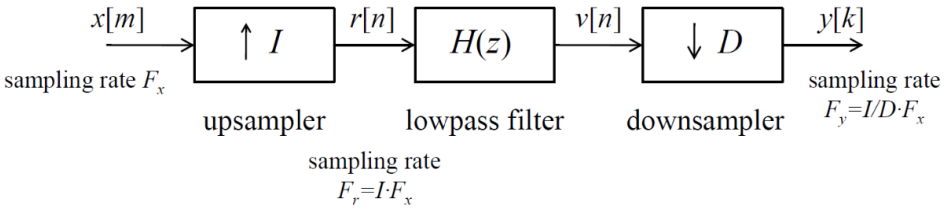
Implementation Decimation

Die beiden Vorteile sind, (1) Filter müssen nur mit *geringer Abtastrate* arbeiten, (2) Prozess ist so gut *paralleisierbar*.



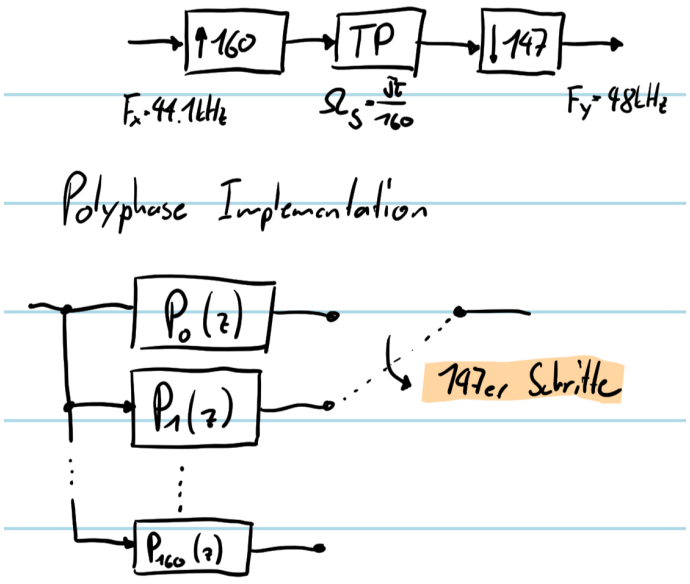
Sampling Rate Conversion

To convert the sampling rate by a non-integer factor use a **Up-** and **Downsampling** combination to convert the samplingrate by a factor of $\frac{I}{D}$. It is preferable to first upsample and downsample afterwards, the other way around information loss can occur, but the low-pass filter has to work at the higher samplingrate. Furthermore the low-pass filter of both conversions can be combined



Through the use of I Polyphase filters a efficient implementation can be achieved. The trick is to let the "switch" always make D-steps.

Example: $I = 160$, $D = 147$

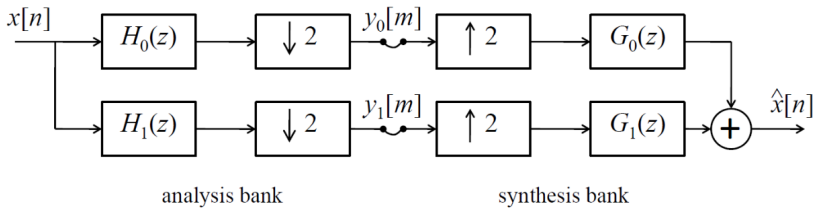


Implementation in Python:

```
from scipy import signal
import matplotlib.pyplot as plt

# upsample by factor I=160 and downsample by a factor D=147
I = 160
D = 147

# FIR lowpass filter (impulse response h) with order: 3840; wpass: 1/200; wstop: 1/160
h = signal.firwin(3840, 1/200)

# create Polyphase filter from FIR lowpass
P = np.zeros((I, len(h)//I))
for i in range(I):      # get each coeff P_i
    P[i,:] = h[i::I]    # take every 160th sample

# apply Polyphase filter to x
y = np.zeros(((len(x)*160)//147))
i = 0; m = 0
for n in range(len(x)): # apply polyphase
    i = (D*n)%I  # idx of filter P_i
    m = (D*n)//I # idx of signal
    y[n] = I*np.convolve(P[i,:],x[m:m+24],mode='valid')
```

Quadrature Mirror Filters

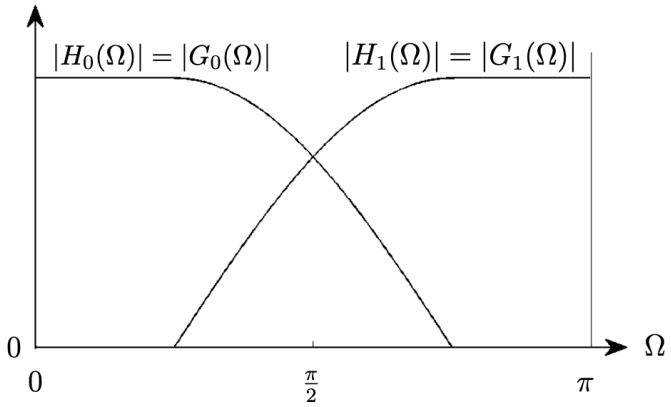
As one would lose information by downsampling because of the applied low-pass filter, we can low- and high-pass filter a signal and downsample both resulting signals



If the following is true

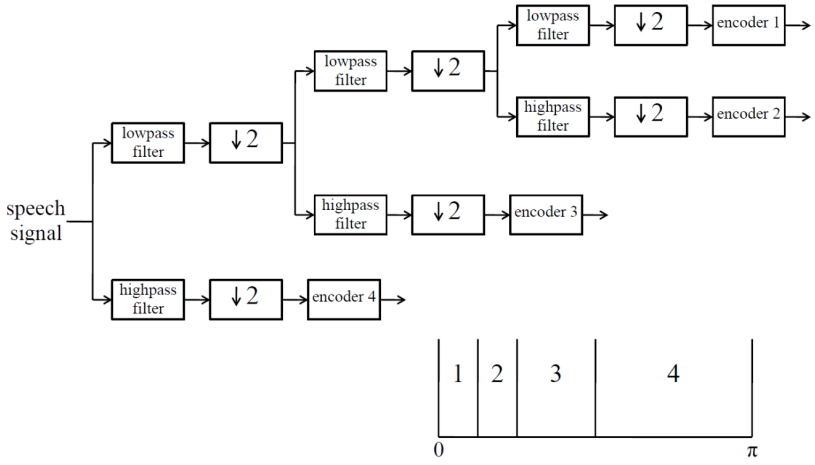
$$H_0(\Omega) = H(\Omega) \tag{0.6}$$
$$H_1(\Omega) = H(\Omega - \pi) \tag{0.7}$$
$$G_0(\Omega) = H(\Omega) \tag{0.8}$$
$$G_1(\Omega) = -H(\Omega - \pi) \tag{0.9}$$

We gain theoretical **perfect reconstruction**



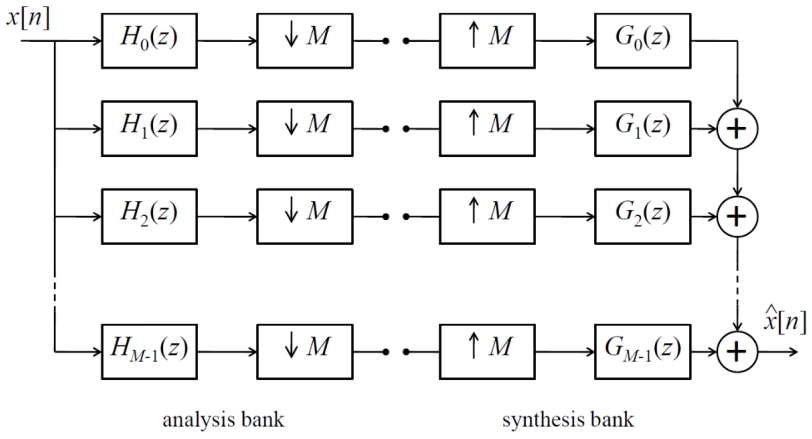
Because $H_1 = H_0(\Omega - \pi)$, we speak of *power symmetric filters*.

Application: **Subband Coding** can be used for audio and image compression, because different frequency bands can be assigned with different code rates and thus emphasize the regions with a higher power density. Speech for example holds the most power in the lower frequency bands, thus quadrature mirror filter banks can be applied



DFT Filter Banks

By applying more than just two filter banks, we get a *M-channel filter bank*. There are M filters parallel in both the analysis and synthesis banks. As a result the subband signals have a far lower sampling rate than the original signal



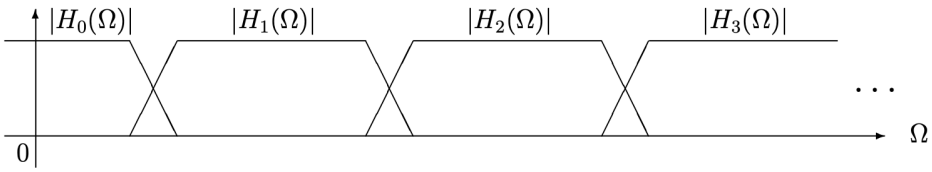
Over-/Undersampling

The analysis bank employs the M filters $H_0(z), H_1(z), \ldots, H_{M-1}(z)$ and downsamples with the same factor M
If we apply N Filters $H_0(z), H_1(z), \ldots, H_{N-1}(z)$ but downsample by a different factor M we can achieve...

- ... undersampling when $N > M$
- ... oversampling when $M > N$

The filters H_ℓ are generaly defined as

$$H_\ell(z) = H(z \cdot e^{-j2\pi\ell/M}), \quad \ell = 0, 1, \ldots, M-1$$

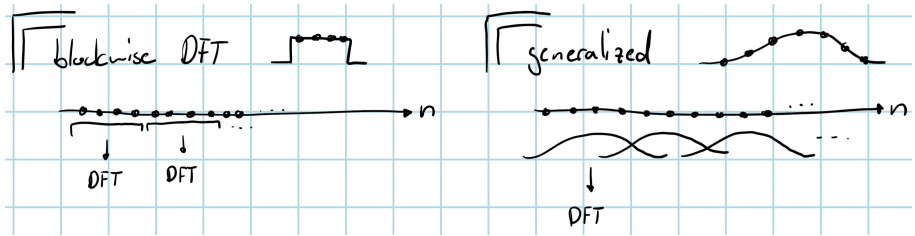


With the corresponding impulse response

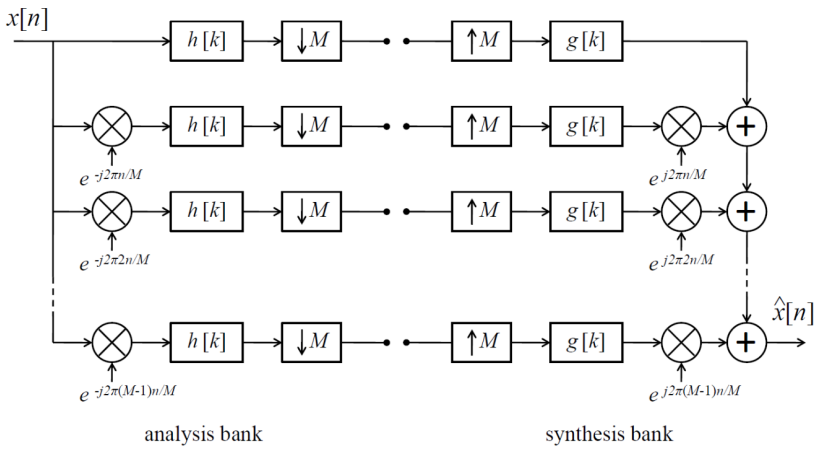
$$h_\ell[k] = h[k] \cdot e^{j2\pi\ell k/M}, \quad \ell = 0, 1, \ldots, M-1$$

Blockwise DFT

Blockwise / Generalized



We can implement the same behaviour or exclude the rotation factor from the filters and apply them seperatly. This gives us a blockwise *DFT* on the analysis side and a *IDFT* on the synthesis side



The filters are now the same with every subband and can be definded as

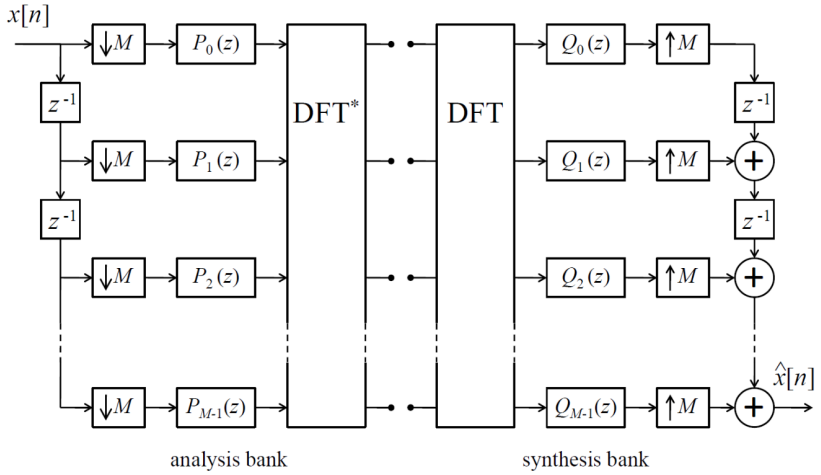
$$h[k] = \begin{cases} 1 & \text{if } k \in \{0, 1, \ldots, M-1\} \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad g[k] = \begin{cases} \frac{1}{M} & \text{if } k \in \{0, 1, \ldots, M-1\} \\ 0 & \text{otherwise} \end{cases}$$

where they act as the **window function** and can be changed accordingly.

Blockwise Polyphase DFT

A particularly efficient implementation of a non-rectangular window on a DFT can be achieved through a polyphase filter. Through this we get the M-component polyphase decomposition

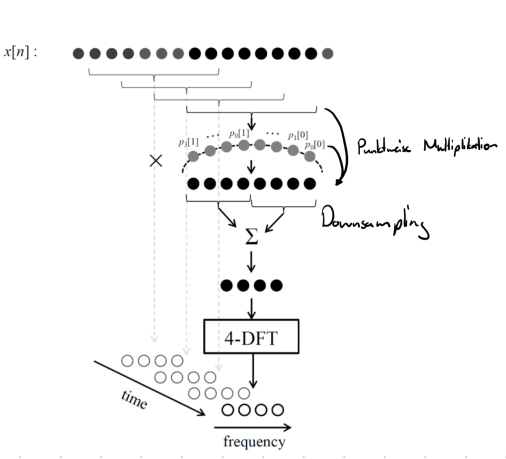
$$H_\ell(z) = \sum_{i=0}^{M-1} z^{-i} \cdot e^{j2\pi\ell i/M} \cdot P_i(z^M) = \sum_{i=0}^{M-1} e^{j2\pi\ell i/M} \cdot \left(z^{-i} \cdot P_i(z^M)\right) \qquad \text{with } \ell = 0, 1, \ldots, M-1$$

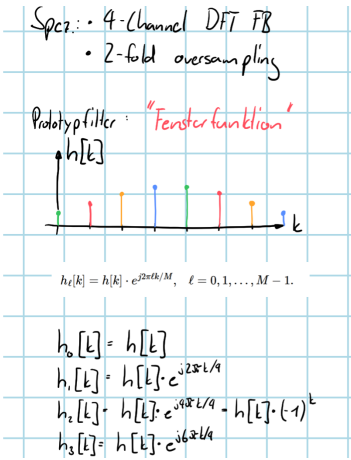


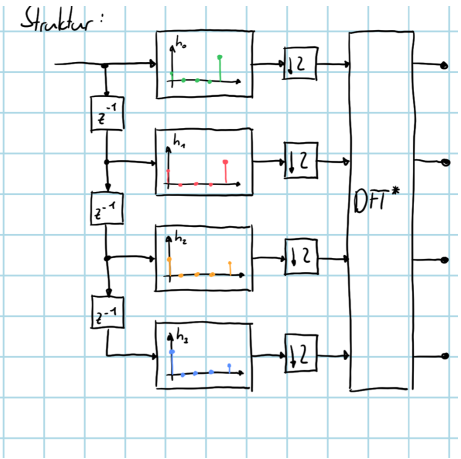
There are two aspects that make the filter bank implementation in Fig. 4 very attractive:

1. The polyphase components are FIR filters, the impulse responses of which are downsampled versions of $h[k]$ or $g[k]$. These parallel filters normally have a very low complexity.
2. For the DFT computations efficient Fast Fourier Transform algorithms are available.

Working example

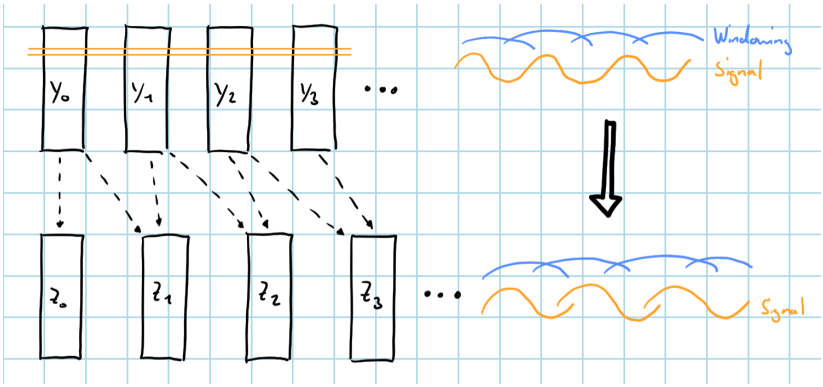






Application Time Scale Modification

There are in fact a number of methods for achieving the goal without affecting the signal frequencies, none of them trivial or perfect however. One approach is to make use of DFT filter banks and accomplish the time scaling in the time-frequency domain as sketched in the following illustration.

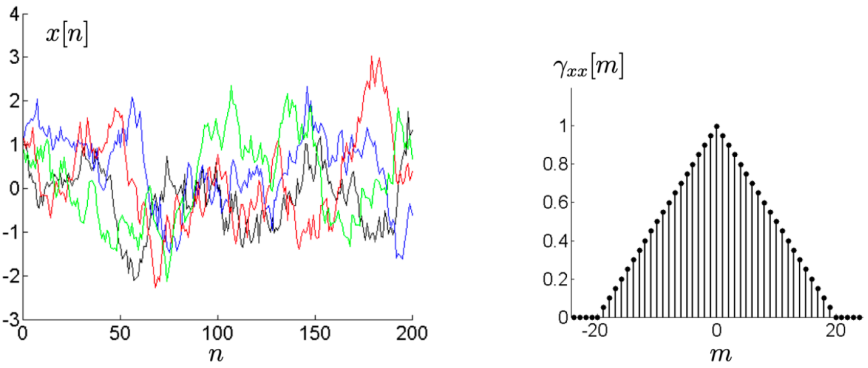


Random Signals

Definition

Vector of a random variable where the *mean* is $m_x = E\{x[n]\}$. Wobei die Autokorrelation eine Funktion über den Abstand zweier samples darstellt

$$\gamma_{xx}[m] = E\{x^*[n] \cdot x[n+m]\}$$



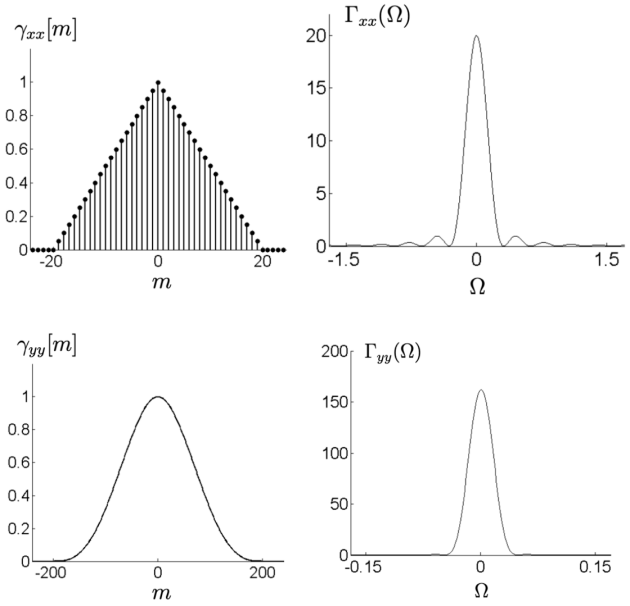
Spectrum

So the mean and autocorrelation characterizes a random signal in the time domain. In the frequency domain a random signal is defined by its **power density spectra**

$$\Gamma_{xx}(\Omega) = \sum_{m=-\infty}^{\infty} \gamma_{xx}[m] \cdot e^{-j\Omega m}$$

reverse correlation

One can see, that a wide Autocorrelation corresponds to a small power density spectra



There are several properties derived from *Wiener-Khinchin theorem*

- The Power P_x of a signal $x[n]$ is

$$P_x = \frac{1}{2\pi}\int_{-\pi}^{\pi} \Gamma_{xx}(\Omega)\mathrm{d}\Omega \qquad \text{or} \qquad P_x = \gamma_{xx}[0]$$

- The value $P_x = \gamma_{xx}[0]$ is the maximum of the autocorrelation sequence magnitude and

$$|\gamma_{xx}[m]| \leq P_x \quad \text{for all } m$$

- For stationary random signals with nonzero mean m_x, $\gamma_{xx}[m] = c_{xx}[m] \cdot e{-}j\Omega m + m_x^2 \cdot 2\pi\delta(\Omega)$ holds

$$\Gamma_{xx}(\Omega) = \sum_{m=-\infty}^{\infty} c_{xx}[m] \cdot e^{-j\Omega m} + m_x^2 \cdot 2\pi\delta(\Omega)$$

White noise

A special class of stationary stochastic processes are *white noise* signals. These are characterized by an autocorrelation of the form

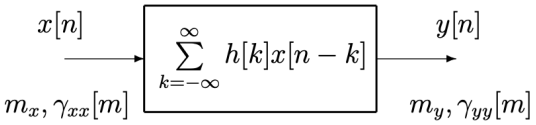
$$\gamma_{ww}[m] = \begin{cases} \sigma_w^2 & \text{if } m = 0 \\ 0 & \text{if } m \neq 0 \end{cases}$$

implying that consecutive samples are uncorrelated. The corresponding power density spectrum is constant over Ω, given as

$$\Gamma_{ww}(\Omega) = \sigma_w^2$$

Spectral Shaping in LTI Systems

If we pass a random signal through a LTI-System



we get the following output signal characteristics

mean: $m_y = H(0) \cdot m_x$

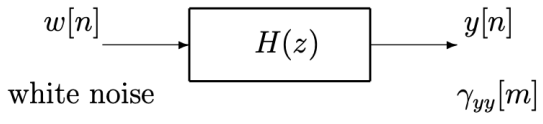
autocorrelation: $\gamma_{yy}[m] = h^*[-i] * \gamma_{xx}[m] * h[k]$

spectra: $\Gamma_{yy}(\Omega) = |H(\Omega)|^2 \cdot \Gamma_{xx}(\Omega)$

z-transformation: $\Gamma_{yy}(z) = H(z^{-1}) \cdot \Gamma_{xx}(z) \cdot H(z)$

Linear Models for Stochastic Processes

First we apply **filtering of white noise** through the system $H(z)$

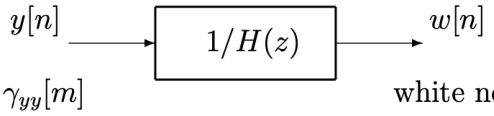


$$\Gamma_{yy}(z) = \sigma_w^2 \cdot H(z) \cdot H(z^{-1})$$

Making up a filter by picking zeros and poles lying within the unit circle for guaranteed stability of the system we get the filter $H(z)$ and the corresponding power density spectra $\Gamma_{yy}(z)$

$$H(z) = \frac{B(z)}{A(z)} = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}} \qquad \text{thus} \qquad \Gamma_{yy}(z) = \sigma_w^2 \cdot \frac{B(z) \cdot B(z^{-1})}{A(z) \cdot A(z^{-1})}$$

By taking the inverse $\frac{1}{H(z)}$ of the system we get a **noise whitening filter** with the properties



$$\frac{1}{H(z)} = \frac{A(z)}{B(z)} = \frac{1 + \sum_{k=1}^{N} a_k z^{-k}}{\sum_{k=0}^{M} b_k z^{-k}}$$

In the following we distinguish three different models for white noise filters

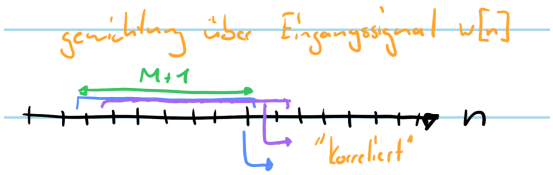
Model	$H(z)$	$\frac{1}{H(z)}$
Moving Average *AV*	FIR	IIR
Autoregressive *AR*	IIR	FIR
ARMA	IIR	IIR

Moving average (MA) model

If $H(z)$ represents an FIR filter of order M, the white noise $w[n]$ is transformed into the random signal

$$y[n] = \sum_{k=0}^{M} b_k w[n-k]$$

In the special case of $b_0 = b_1 = \cdots = b_M = (M+1)^{-1}$, every sample at the filter output represents the average of the input signal within a sliding window of length $M+1$

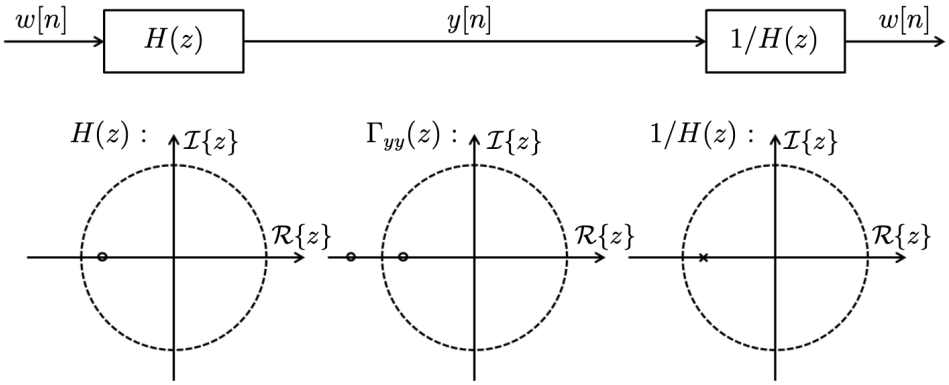


For finite M the pole-zero plot of $H(z)$ contains only zeros. By inversion $\frac{1}{H(z)}$ every zero gets substituted by a pole, thus making it a all-pole filter.

The following holds for the autocorrelation

$$\gamma_{yy}[m] = \begin{cases} \sigma_w^2 \sum_{k=m}^{M} b_k \cdot b_{k-m}^* & \text{if } 0 \le m \le M \\ 0 & \text{if } m > M \\ \gamma_{yy}^*[-m] & \text{if } m < 0 \end{cases}$$

Example: $H(z) = 0.6 + 0.8z^{-1}$



Autoregressive (AR) model

In the so-called *autoregressive* (AR) model, a random sequence is generated by an all-pole filter according to

$$y[n] = w[n] - \sum_{k=1}^{N} a_k y[n-k] \qquad \circ\!\!-\!\!\bullet \qquad H(z) = \frac{z^N}{z^N + a_1 z^{N-1} + \cdots + a_N}$$

The corresponding noise whitening filter is a causal FIR filter with no poles in the pole-zero plot. To determine the filter coefficients $a_1, a_2, \ldots, a_N$ we need the noise variance σ_w^2 and the following equations

$$\gamma_{yy}[m] = \begin{cases} -\sum_{k=1}^{N} a_k \cdot \gamma_{yy}[m-k] & \text{if } m > 0 \\ \sigma_w^2 - \sum_{k=1}^{N} a_k \cdot \gamma_{yy}[-k] & \text{if } m = 0 \\ \gamma_{yy}^*[-m] & \text{if } m < 0 \end{cases}$$

$$\begin{pmatrix} \gamma_{yy}[0] & \gamma_{yy}[-1] & \cdots & \gamma_{yy}[-N] \\ \gamma_{yy}[1] & \gamma_{yy}[0] & \cdots & \gamma_{yy}[-N+1] \\ \vdots & \vdots & & \vdots \\ \gamma_{yy}[N] & \gamma_{yy}[N-1] & \cdots & \gamma_{yy}[0] \end{pmatrix} \cdot \begin{pmatrix} 1 \\ a_1 \\ \vdots \\ a_N \end{pmatrix} = \begin{pmatrix} \sigma_w^2 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

The desired coefficients can be found by solving the *Yule-Walker equations*.

ARMA model

The MA and AR models are special cases of the general so-called ARMA model. Here the orders of the polynomials $B(z)$ and $A(z)$ (i.e., M and N, respectively) are both one or higher. Hence the pole-zero plots of both $H(z)$ and $\frac{1}{H(z)}$ contain poles *and* zeros. For some random processes the ARMA model is more suitable than the above discussed special versions because of a smaller number of required coefficients $b_1, b_1, \ldots, b_M, a_1, a_2, \ldots, a_N$ for an accurate modeling.

Spectral Density Estimation

Nonparametric methods

Directly computing the DTFT squares leads to the **periodogram**

$$\widehat{\Gamma}_{xx}(\Omega) = \frac{1}{N} \left| \sum_{n=0}^{N-1} x[n] \cdot e^{-j\Omega n} \right|^2$$

Because for any Ω_0 the estimate $\widehat{\Gamma}xx(\Omega_0)$ has a large variance, which doesn't decrease with increasing sample basis we introduce the following

- Biased autocorrelation estimator:

$$\widehat{\gamma}_{xx}[m] = \frac{1}{N} \sum_{n=0}^{N-m-1} x^*[n] \cdot x[n+m], \quad \text{for } m \text{ between 0 and } N-1$$

- Unbiased autocorrelation estimator:

$$\widehat{\gamma}_{xx}[m] = \frac{1}{N-m} \sum_{n=0}^{N-m-1} x^*[n] \cdot x[n+m], \quad \text{for } m \text{ between 0 and } N-1$$

Parametric methods

Parametric methods build on a specific model tuned by a fixed number of parameters. Usually a ARMA model is used to get a good system property. But also AR and MA models can be of good use. In a AR model the parameters can be deduced from an estimate of the autocorrelation sequence by means of the Yule-Walker equations. Substituting $\gamma_{xx}[m]$ with $\widehat{\gamma}_{xx}[m]$.

Another way to compute the parameters $a_1, \ldots, a_k$ is through *least squares* model fitting. According to the AR model the nth sample can be written as $y[n] = \hat{h}[n] + w[n]$ with

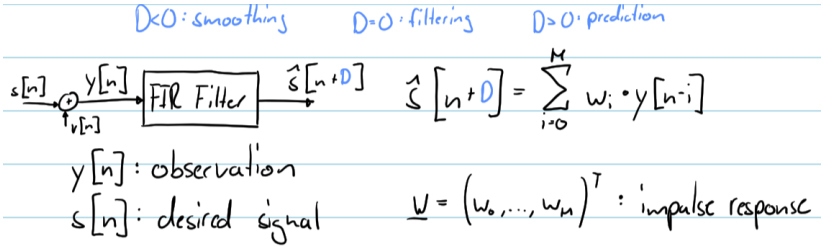
$$\hat{y}[n] = -\sum_{k=1}^{N} a_k y[n-k]$$

Optimum Linear Filters

When we want to recover a distorted signal as good as possible we want to minimize the error (often mean squared error) as much as possible. One rather easy approach is the ussage of linear filters.

Wiener Filters (stationary systems)

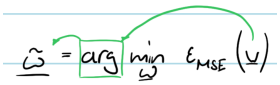
Filters that are optimal in the mean-squared error sense for stationray signals are known as **Wiener filters**. The system with a inputsignal $s[n] + v[n]$ must be tuned to optimal wiener coefficients $w_0, w_1, \dots w_M$ such that the filter output serves as a good estimate $\hat{s}[n + D]$



where the criterion is the *mean-squared error*

$$\varepsilon_{MSE}(\underline{w}) = E\left\{ |\underbrace{\hat{s}[n+D] - s[n+D]}_{\text{error signal}}|^2 \right\}$$

The optimal wiener-coefficients can be found at the point with the lowest mean-squared error



This can be achieved through the **Wiener-Hopf equations**

$$\mathbf{R}_{yy} = \begin{pmatrix} \gamma_{yy}[0] & \gamma_{yy}[-1] & \cdots & \gamma_{yy}[-M] \\ \gamma_{yy}[1] & \gamma_{yy}[0] & \cdots & \gamma_{yy}[1-M] \\ \vdots & \vdots & & \vdots \\ \gamma_{yy}[M] & \gamma_{yy}[M-1] & \cdots & \gamma_{yy}[0] \end{pmatrix} \quad \text{with } \gamma_{yy}[m] = E\{y[n] \cdot y^*[n-m]\}\text{: autocorrelation of filter input}$$

$$\mathbf{r}_{sy} = \begin{pmatrix} \gamma_{sy}[D] \\ \gamma_{sy}[D+1] \\ \vdots \\ \gamma_{sy}[D+M] \end{pmatrix} \text{ and the vector } \widetilde{\mathbf{w}} = \begin{pmatrix} \widetilde{w}_0 \\ \widetilde{w}_1 \\ \vdots \\ \widetilde{w}_M \end{pmatrix} \quad \text{with } \gamma_{sy}[m] = E\{s[n] \cdot y^*[n-m]\}\text{: crosscorr. of } s[n] \text{ and } y[n]$$

The equation system can be written as $\mathbf{R}_{yy}\widetilde{\mathbf{w}} = \mathbf{r}_{sy}$ which leads to the optimal filter vector

$$\widetilde{\mathbf{w}} = \mathbf{R}_{yy}^{-1}\mathbf{r}_{sy}$$

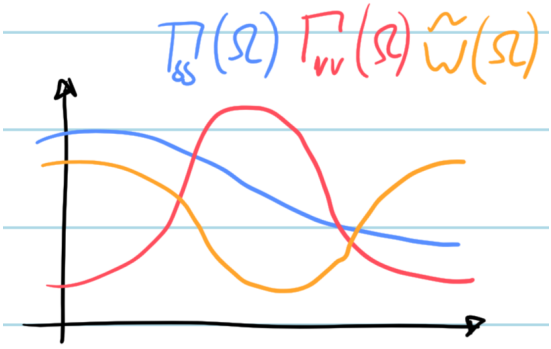
Unconstrained Wiener Filters

Through the unconstrained filter one gets a different filter response, but is not suitable for practical applications. We, for this example, implement a non-causal IIR filter with the impuls response $\ldots, \tilde{w}-1, \tilde{w}_0, \tilde{w}_1, \ldots$, setting $D = 0$

$$\sum_{i=-\infty}^{\infty} \tilde{w}_i \cdot \gamma_{yy}[m-i] = \gamma_{sy}[m+D], \quad m = 0, 1, \ldots, M$$

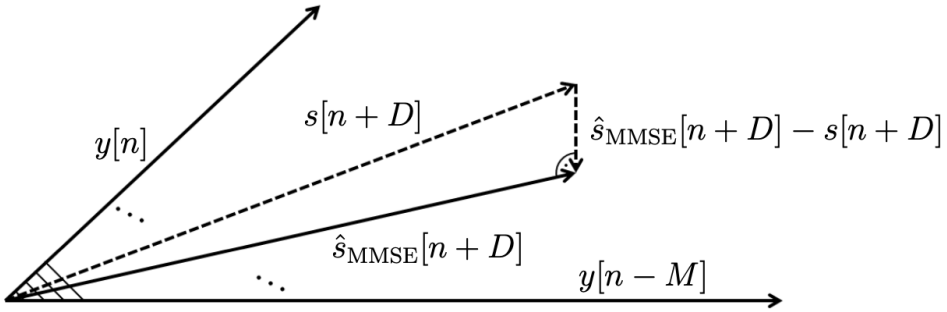
In the frequency domain we get

$$\widetilde{W}(\Omega) = \frac{\Gamma_{ss}(\Omega)}{\Gamma_{ss}(\Omega) + \Gamma_{vv}(\Omega)}$$



Principle of Orthogonality

Da das gewünschte Signal nicht im Unterraum der Beobachtungen $y[n]$ und $y[n-M]$ liegt, kann das gewünschte Signal $s[n+D]$ nicht direkt erreicht werden, man kann nur annähern



Since uncorrelated random variables with zero mean are said to be *orthogonal*, this interesting result is referred to as the *principle of orthogonality*. The estimate $\hat{s}_{MMSE}[n+D]$ is a linear combination of the observations and thus an element of the linear subspace spanned by the random vectors $y[n], y[n-1], \ldots, y[n-M]$. Forming the linear combination in such a way that the distance to $s[n+D]$ becomes minimal results in the optimal estimate. Obviously, the vector $(\hat{s}_{MMSE}[n+D] - s[n+D])$, representing the estimation error, has minimal length when being orthogonal to the subspace spanned by the observations.

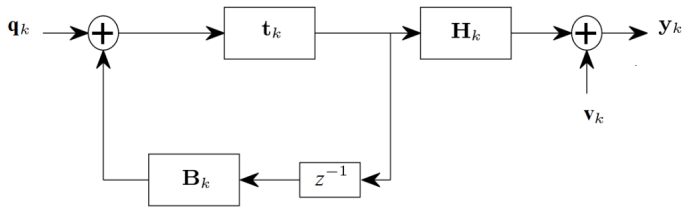
Implementation

```
def wiener(g, b, P, M, D):
    # Wiener filter design
    # g : impulse response of distorting system
    # b : one-sided autocorrelation sequence of desired signal
    # P : additive noise variance
    # M : Wiener filter order
    # D : filter delay
```

```
    # return: impulse response of optimal filter
    # compute autocorrelation of Wiener filter input signal
    acy = np.convolve(np.convolve(g,np.concatenate((np.flip(b)[:-1],b))),np.flip(g))
    # truncate/zero-pad acy such that center is at element M
    if (len(acy)>2*M+1):
        acy = acy[int((len(acy)-(2*M))/2):]
    elif (len(acy)<2*M+1):
        acy = np.concatenate((np.zeros(int(((2*M+2)-len(acy))/2)),acy,np.zeros(int(((2*M+2)-len(acy))/2))))
    # add noise variance
    acy[M] = acy[M]+P

    # compute correlation matrix of random input signal
    Ry = np.zeros((M+1,M+1))
    for m in range(M+1):
        Ry[m,:] = acy[M-m:2*M+1-m]

    # input/output signal correlation vector
    q = np.concatenate((np.flip(b)[:-1],b,np.zeros(len(g)+M+np.abs(D))))
    # truncate/zero-pad q such that first element is autocorrelation of desired signal at m=D
    if (len(b)+D<1):
        q = np.concatenate((np.zeros(1-len(b)-D),q))
    elif (len(b)+D>1):
        q = q[len(b)+D-1:]

    rsy = np.zeros(M+1)
    for m in range(M+1):
        rsy[m] = np.sum(g*q[m:m+len(g)])

    # Wiener filter
    h = np.matmul(np.linalg.inv(Ry),rsy)

    return h
```

Kalman Filter (dynamic systems)

To increment the order of a Wiener filter, a **Kalman filter** can be applied. The Kalman filter takes every additional observation to enhance the filter without a limit on the filter order. Furthermore the Kalman filter can be applied into a *dynamic system*. It is built on a *state space representation* of a physical system, where the inner system state may be a coordinate of a vehicle



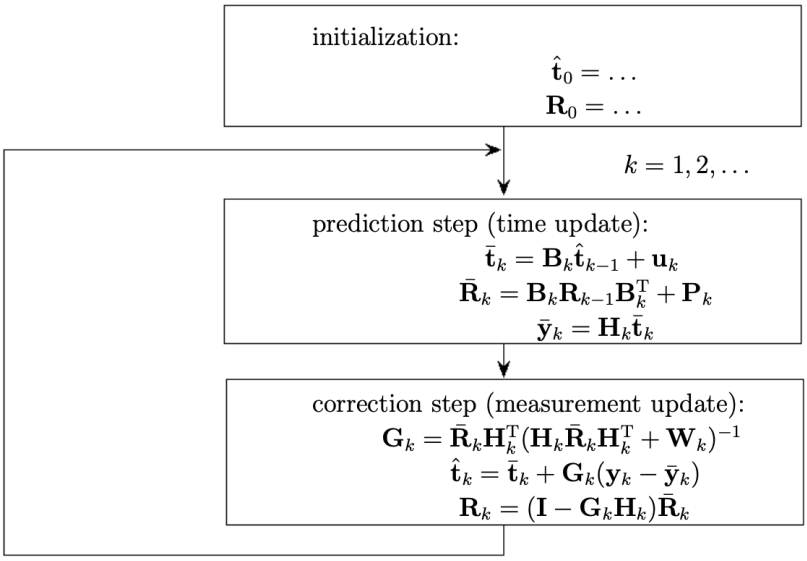
What is observable from the outside are the measurements $y_1, y_2, \ldots$ which are linear maps of the internal state subject to measurement errors, Specifivally, the kth observation is given by

$$y_k = H_k t_k + v_k$$

Everey entity of the algorithm:

entity	symbol	deterministic/random
state vector	$\mathbf{t}_k$	random
covariance matrix of $\mathbf{t}_k$	$\mathbf{R}_k$	deterministic
observation	$\mathbf{y}_k$	random
measurement matrix	$\mathbf{H}_k$	deterministic
measurement error	$\mathbf{v}_k$	random with mean $\mathbf{0}$, covariance $\mathbf{W}_k$
state transition matrix	$\mathbf{B}_k$	deterministic
input	$\mathbf{q}_k$	random with mean $\mathbf{u}_k$, covariance $\mathbf{P}_k$

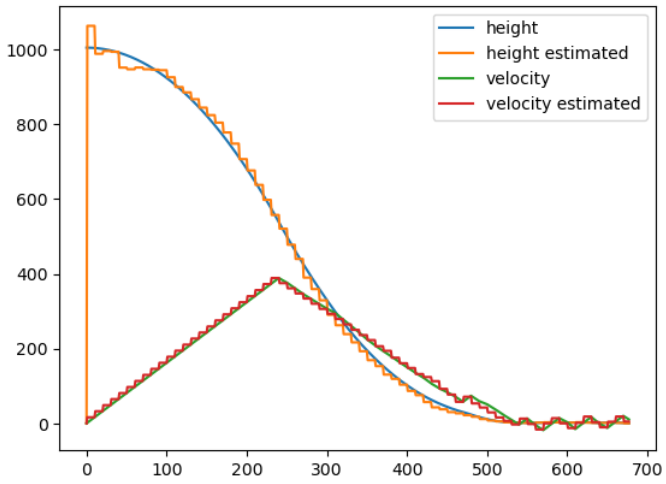
And the Kamlan Algorithm holds:



Example: Lunar lander

```
# with Kalman filter
h_undock = np.random.normal(loc=1000.0, scale=10.0)    # undock height
dt = 0.1
thrust = FperKg

# initial altitude/speed
h = h_undock
v = 0.0

t_est = np.array([0.0,0.0])    # state vector estimate
R_t = np.array([[1.0e9,0.0],[0.0,0.0]])    # state vector covariance matrix
B = np.array([[1,-1],[0,1]])    # state transition matrix
P = np.array([[0,0],[0,(thrust/10)**2]])    # input error variance
H = np.array([1,0])

h_rec = np.array([h])
v_rec = np.array([v])
h_est_rec = np.array([t_est[0]])
v_est_rec = np.array([t_est[1]])
for i in range(1000):
    if ((i%10)==0):
        # noisy measurement
        h_meas = np.random.normal(loc=h, scale=(h/10.0))
```

```
        # correction step
        W = (h_meas/10)**2
        G = R_t[:,0]/(R_t[0,0]+W)
        t_est = t_est + G*(h_meas-t_est[0])
        R_t = np.matmul(np.array([[1-G[0],0],[-G[1],1]]),R_t)

        # prediction of state one second ahead without thrust
        t_pred = np.matmul(B,t_est) + np.array([-g_moon/2,g_moon])

        # action: thrust on/off
        h_1 = t_pred[0]
        v_1 = t_pred[1]
        if ((v_1>0) and (v_1*v_1>2.0*h_1*(thrust-g_moon))):
            # full thrust
            a_set = -thrust+g_moon
            a_true = np.random.normal(loc=-thrust, scale=(thrust/10))+g_moon
            Pi = P
        else:
            # no thrust
            a_set = g_moon
            a_true = g_moon
            Pi = np.zeros((2,2))

        # prediction step
        t_est = np.matmul(B,t_est)+np.array([-a_set/2,a_set]).T
        R_t = np.matmul(B,np.matmul(R_t,B.T))+Pi

    # altitude/speed update
    h -= ((v+a_true*dt/2)*dt)
    v += (a_true*dt)
    h_rec = np.append(h_rec, h)
    v_rec = np.append(v_rec, v)
    h_est_rec = np.append(h_est_rec, t_est[0])
    v_est_rec = np.append(v_est_rec, t_est[1])
    if (h<0.0):
        break
```



Adaptive Filters

Wiener and Kalman filters are optimal filters under the assumption, that the statistics of the involved random processes are perfectly known. This is rarely the case, that's why we introduce **adaptive filters** which can adapt to unknown and possibly varying conditions.

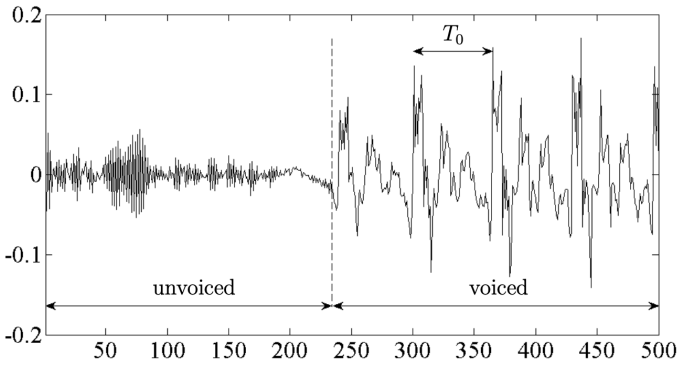
Linear Predictive Coding (LMS)

It has been found that rather than using *waveform coder* which aim at preserving signal waveform, *vocoders* build on a certain speech synthesis model and extract the parameters of the model. Transmitting the model parameters requires usually less bandwidth than transmitting the waveform.

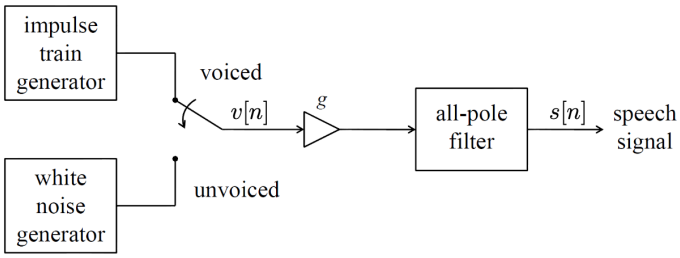
The human vocal tract can be modeled as an AR all-pole model with the system function

$$H(z) = \frac{g}{1 - \sum_{k=1}^{P} a_k z^{-k}}$$

with P the order. The **LPC-10e** standard implements this system with the order of $P = 10$. Furthermore voice is split into voiced (vowels e.g. "a") and unvoiced (consonants e.g. "f").



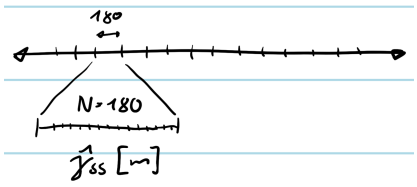
On the synthesis side we are looking at a model which derives the voiced and unvoiced parts from white noise resp. an impuls train



To extract the coefficients $a_1, a_2, \ldots, a_P$, they are chosen to reproduce the signal segment $s_{in}[n]$ as closely as possible through

$$\hat{s}[n] = \sum_{k=1}^{P} a_k s_{\text{in}}[n-k]$$

where the segment is a part of the sampled length



The Parameters can be defined through the Yule-Walker equation with the autocorrelation $\widehat{\gamma}_{ss}[m] = \frac{1}{N} \cdot \sum_{n=1}^{N-m} s_{in}[n] \cdot s_{in}[n+m]$ with

$$\mathbf{R}_{ss} = \begin{pmatrix} \gamma_{ss}[0] & \gamma_{ss}[1] & \cdots & \gamma_{ss}[P-1] \\ \gamma_{ss}[1] & \gamma_{ss}[0] & \cdots & \gamma_{ss}[P-2] \\ \vdots & \vdots & & \vdots \\ \gamma_{ss}[P-1] & \gamma_{ss}[P-2] & \cdots & \gamma_{ss}[0] \end{pmatrix} \quad \text{and} \quad \mathbf{r}_{ss} = \begin{pmatrix} \gamma_{ss}[1] \\ \gamma_{ss}[2] \\ \vdots \\ \gamma_{ss}[P] \end{pmatrix}$$

and solving for $\mathbf{a}$

$$\mathbf{R}_{ss}\mathbf{a} = \mathbf{r}_{ss}$$

Note that a Levinson-Durbin recursion can be applied, as $\mathbf{R}_{ss}$ is a Toeplitz matrix.

To determine the remaining parameters, consider the error signal

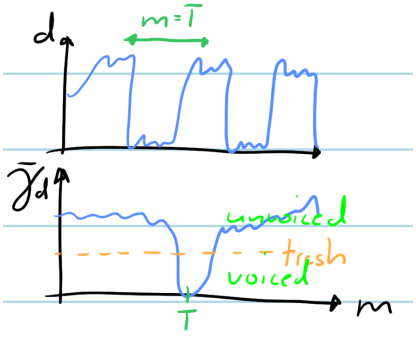
$$d[n] = s_{\text{in}}[n] - \hat{s}[n] = s_{\text{in}}[n] - \sum_{k=1}^{P} a_k s_{\text{in}}[n-k]$$

When defining the excitation $g \cdot v[n]$ through $v[n]$ as a signal with unit power, we get for g

$$g = \sqrt{\frac{1}{N} \sum_{n=1}^{N} d^2[n]}$$

To distinguish between voiced and unvoiced sound as well as for the pitch estimation, the *LPC-10e* algorithm relies on the *average magnitude difference function*

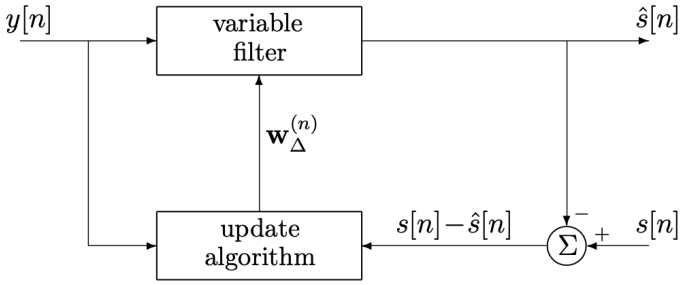
$$\bar{\gamma}_d[m] = \frac{1}{N-m} \sum_{n=1}^{N-m} \left| \frac{d[n]}{g} - \frac{d[n+m]}{g} \right|$$



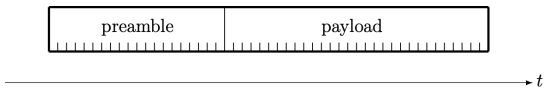
If the value of $\bar{\gamma}_d$ falls under a certain threshold, the segment is declared **voiced** and the value $m = T$ sets the pitch.

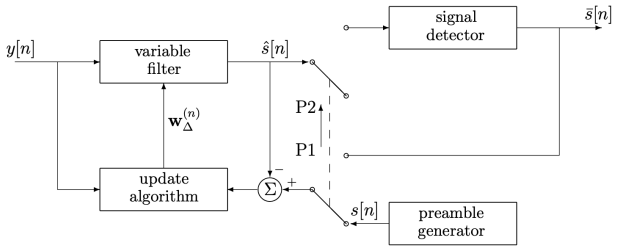
The LMS Algorithm

The **LMS Algorithm** works adaptively and can adjust its parameters through a update algorithm



When the desired signal $s[n]$ isn't available a so called *preamble* with a defined signal sequence is used to determine the channels behaviour

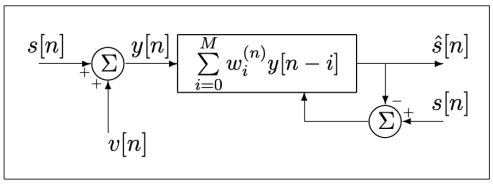


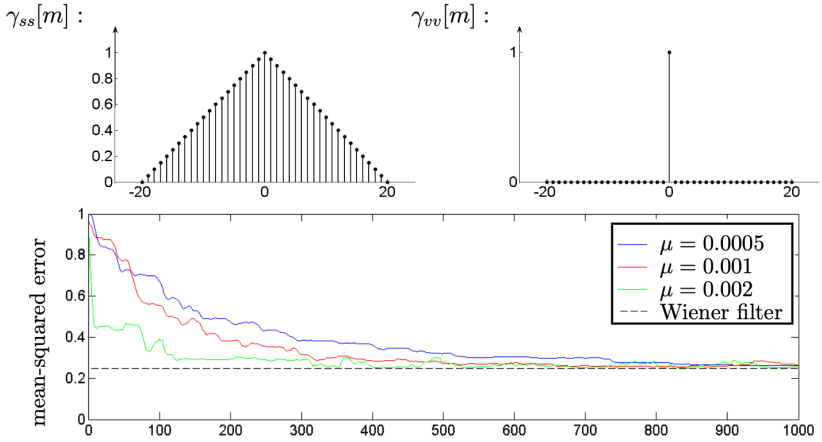


The LMS algorithm works through the continuous update of the parameters $\mathbf{w}$. First, the coefficient vector is initialized as $\mathbf{w}^{(0)} = \mathbf{0}_{M+1}$ (i.e., a vector with $M+1$ zeros). Any prior knowledge may of course be used to find a better starting vector $\mathbf{w}^{(0)}$. Subsequently, as $n = 0, 1, 2, \ldots$

$$\mathbf{w}^{(n+1)} = \mathbf{w}^{(n)} + \underbrace{\mu \left(s[n] - \left(\mathbf{w}^{(n)}\right)^\mathsf{T} \mathbf{y}_n \right) \cdot \mathbf{y}_n}_{=\mathbf{w}_\Delta^{(n)}}$$

The convergence speed can by changed by the stepsize μ





RLS Algorithm

A Algorithm which converges much faster is the **recursive least squares** (RLS) Algorithm. It is much more complex but can be used for echo cancellation with really fast changing echos