

# Digital signal processing mathematics

*M. Hoffmann*

DESY, Hamburg, Germany

## Abstract

Modern digital signal processing makes use of a variety of mathematical techniques. These techniques are used to design and understand efficient filters for data processing and control. In an accelerator environment, these techniques often include statistics, one-dimensional and multidimensional transformations, and complex function theory. The basic mathematical concepts are presented in four sessions including a treatment of the harmonic oscillator, a topic that is necessary for the afternoon exercise sessions.

## 1 Introduction

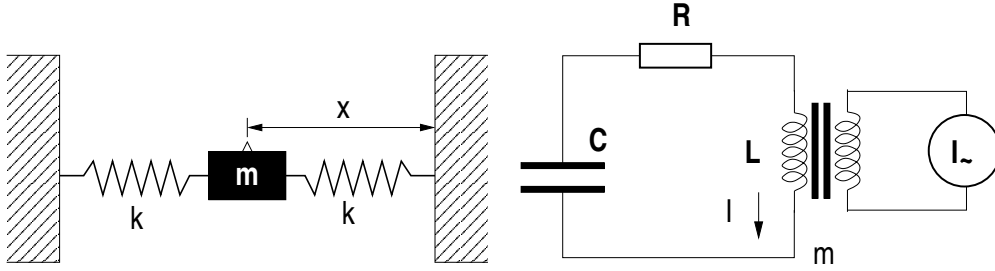
Digital signal processing requires the study of *signals* in a digital representation and the methods to interpret and utilize these signals. Together with analog signal processing, it composes the more general modern methodology of signal processing. Although the mathematics that are needed to understand most of the digital signal processing concepts were developed a long time ago, digital signal processing is still a relatively new methodology. Many digital signal processing concepts were derived from the analog signal processing field, so you will find a lot of similarities between the digital and analog signal processing. Nevertheless, some new techniques have been necessitated by digital signal processing, hence, the mathematical concepts treated here have been developed in that direction. The strength of digital signal processing currently lies in the frequency regimes of audio signal processing, control engineering, digital image processing, and speech processing. Radar signal processing and communications signal processing are two other subfields. Last but not least, the digital world has entered the field of accelerator technology. Because of its flexibility, digital signal processing and control is superior to analog processing or control in many growing areas.

Around 1990, diagnostic devices in accelerators began to utilize digital signal processing, e.g., for spectral analysis. Since then, the processing speed of the hardware [mostly standard computers and digital signal processors (DSPs)] has increased very quickly, such that now *fast* RF control is now possible. In the future, direct sampling and processing of all RF signals (up to a few GHz) will be possible, and many analog control circuits will be replaced by digital ones.

The design of digital signal processing systems without a basic mathematical understanding of the signals and its properties is hardly possible. Mathematics and physics of the underlying processes need to be understood, modelled, and finally controlled. To be able to perform these tasks, some knowledge of trigonometric functions, complex numbers, complex analysis, linear algebra, and statistical methods is required. The reader may look them up in his undergraduate textbooks if necessary.

The first session covers the following topics: the dynamics of the harmonic oscillator and signal theory. Here we try to describe what a signal is, how a digital signal is obtained, and what its quality parameters, accuracy, noise, and precision are. We introduce *causal time invariant linear systems* and discuss certain fundamental special functions or signals.

In the second session we are going to go into more detail and introduce the very fundamental concept of convolution, which is the basis of all digital filter implementations. We are going to treat the Fourier transformation and finally the Laplace transformation, which are also useful for treating analog signals.



**Fig. 1:** Principle of a physical pendulum (left) and of an electrical oscillator

The third session will make use of the concepts developed for analog signals as they are applied to digital signals. It will cover digital filters and the very fundamental concept and tool of the  $z$ -transformation, which is *the* basis of filter design.

The fourth and last session will cover more specialized techniques, like the Kalman filter and the concept of wavelets. Since each of these topics opens its own field of mathematics, we can just peek at the surface to get an idea of its power and what it is about.

## 2 Oscillators

One very fundamental system (out of not so many others) in physics and engineering is the harmonic oscillator. It is still simple and linear and shows various behaviours like damped oscillations, resonance, bandpass or band-reject characteristics. The harmonic oscillator is, therefore, discussed in many examples, and also in this lecture, the harmonic oscillator is used as a work system for the afternoon lab-course.

### 2.1 What you need to know about...

We are going to write down the fundamental differential equation of all harmonic oscillators, then solve the equation for the steady-state condition. The dynamic behaviour of an oscillator is also interesting by itself, but the mathematical treatment is out of the scope of this lecture. Common oscillators appear in mechanics and electronics, or both. A good example, where both oscillators play a big role, is the accelerating cavity of a (superconducting) linac. Here we are going to look at the electrical oscillator and the mechanical pendulum (see Fig. 1).

#### 2.1.1 The electrical oscillator

An R-L-C circuit is an electrical circuit consisting of a resistor ( $R$ ), an inductor ( $L$ ), and a capacitor ( $C$ ), connected in series or in parallel (see Fig. 1, right).

Any voltage or current in the circuit can be described by a second-order linear differential equation like this one (here a voltage balance is evaluated):

$$RI + L\dot{I} + \frac{Q}{C} = mI_{\sim}$$

$$\Leftrightarrow \ddot{I} + \frac{R}{L}\dot{I} + \frac{1}{LC}I = KI_{\sim} . \quad (1)$$

### 2.1.2 Mechanical oscillator

A mechanical oscillator is a pendulum like the one shown in Fig. 1 (left). If you look at the forces which apply to the mass  $m$  you get the following differential equation:

$$m\ddot{x} + \kappa\dot{x} + kx = F(t)$$

$$\Leftrightarrow \ddot{x} + \frac{k}{m}\dot{x} + \frac{\kappa}{m}x = \frac{1}{m}F(t) . \quad (2)$$

This is also a second-order linear differential equation.

### 2.1.3 The universal differential equation

If you now look at the two differential equations (1) and (2) you can make them look similar if you bring them into the following form (assuming periodic excitations in both cases):

$$\boxed{\ddot{x} + 2\beta\dot{x} + \omega_0^2 x = T e^{i(\omega_{\sim} t + \xi)}} , \quad (3)$$

where  $T$  is the excitation amplitude,  $\omega_{\sim}$  the frequency of the excitation,  $\xi$  the relative phase of the excitation compared to the phase of the oscillation of the system (whose absolute phase is set to zero),

$$\beta = \frac{R}{2L} \quad \text{or} \quad \frac{k}{2m}$$

is the term which describes the dissipation which will lead to a damping of the oscillator and

$$\omega_0 = \frac{1}{\sqrt{LC}} \quad \text{or} \quad \sqrt{\frac{\kappa}{m}}$$

gives you the eigenfrequency of the resonance of the system.

Also one very often uses the so-called *Q-value*

$$Q = \frac{\omega_0}{2\beta} \quad (4)$$

which is a measure for the energy dissipation. The higher the *Q-value*, the less the dissipation, the narrower the resonance, and the higher the amplitude in the case of resonance.

## 2.2 Solving the DGL

For solving the second-order differential equation (3), we first do the following ansatz:

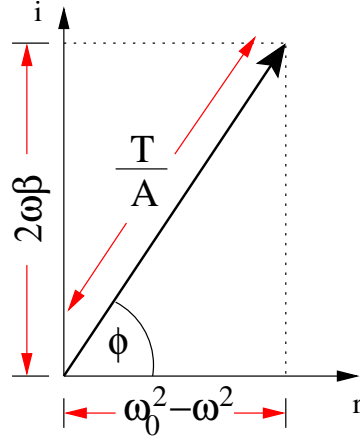
$$\begin{aligned} x(t) &= A e^{i(\omega t + \phi)} \\ \dot{x}(t) &= i\omega A e^{i(\omega t + \phi)} \\ \ddot{x}(t) &= -\omega^2 A e^{i(\omega t + \phi)} . \end{aligned}$$

By inserting this into (3) we get the so-called *characteristic equation*:

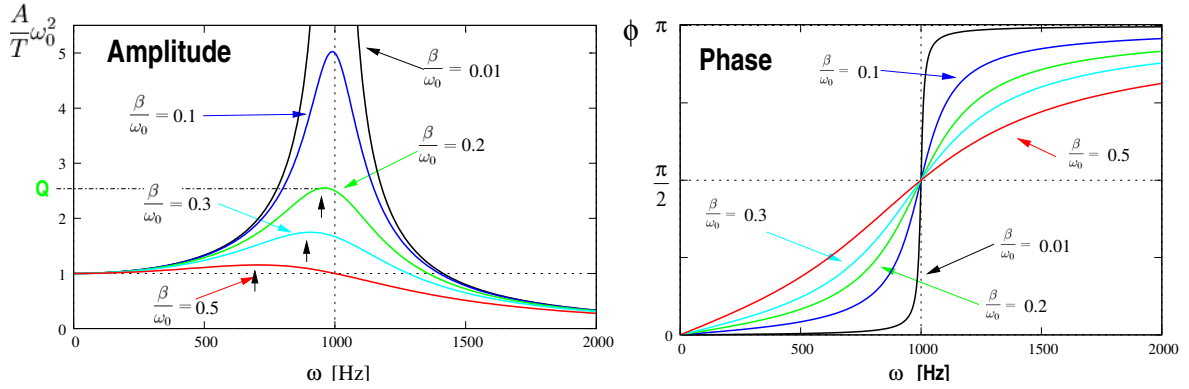
$$-\omega^2 A e^{i(\omega t + \phi)} + 2i\omega\beta A e^{i(\omega t + \phi)} + \omega_0^2 A e^{i(\omega t + \phi)} = T e^{i(\omega_{\sim} t + \xi)}$$

$$\Leftrightarrow -\omega^2 + 2i\omega\beta + \omega_0^2 = \frac{T}{A} e^{i((\omega_{\sim} - \omega)t + (\xi - \phi))} .$$

In the following, we want to look only at the special solution  $\omega \stackrel{!}{=} \omega_{\sim}$  (o.B.d.A  $\xi = 0$ ), because we are only interested in the steady state, for which we already know that the pendulum will take over the



**Fig. 2:** Graphical explanation of the characteristic equation in the complex plane



**Fig. 3:** Amplitude and phase of the excited harmonic oscillator in steady state

excitation frequency. Since we are only interested in the phase difference of the oscillator with respect to the excitation force, we can set  $\xi = 0$ .

In this (steady) state, we can look up the solution from a graphic (see Fig. 2). We get one equation for the amplitude

$$\left(\frac{T}{A}\right)^2 = (\omega_0^2 - \omega^2)^2 + (2\omega\beta)^2$$

$$\Leftrightarrow A = T \frac{1}{\sqrt{(\omega_0^2 - \omega^2)^2 + 4\omega^2\beta^2}}$$

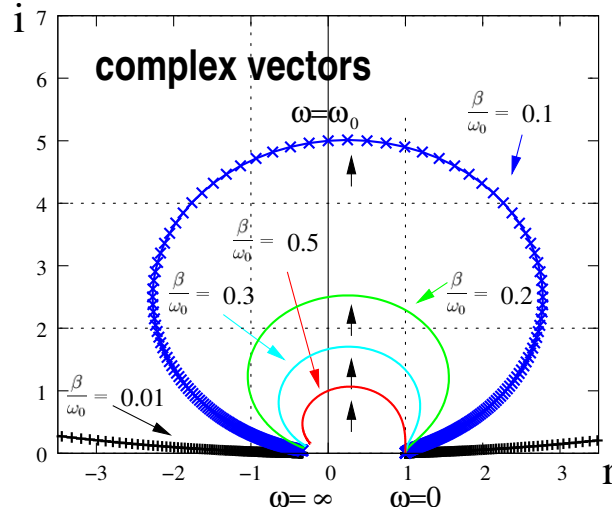
and another for the phase

$$\tan(\phi) = \frac{2\omega\beta}{\omega_0^2 - \omega^2}$$

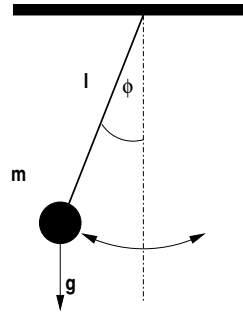
of the solution  $x(t)$ .

Both formulas are visualized in Fig. 3 as a function of the excitation frequency  $\omega$ . Amplitude and phase can also be viewed as a complex vector moving in the complex plane with changing frequency. This plot is shown in Fig. 4. You should notice that the  $Q$ -value gets a graphical explanation here. It is linked to the bandwidth  $\omega_{1/2}$  of the resonance by

$$\omega_{1/2} = \beta = \frac{\omega_0}{2Q},$$



**Fig. 4:** Complex vector of the harmonic oscillator moving with frequency for different  $Q$  values



**Fig. 5:** The gravity pendulum. A mass  $m$  oscillates in the gravity field.

and this also gives

$$Q = \frac{\omega_0}{2\beta} = \left[ \frac{A}{T} |\omega^2| \right]_{\omega=\omega_0},$$

a relation to the height of the resonance peak.

## 2.3 Non-linear oscillators

Besides the still simple harmonic oscillator described above, which is a linear oscillator, many real oscillators are non-linear or at least linear only in approximation. We are going to discuss two examples of simple looking non-linear oscillators. First the *mathematical pendulum*, which is linear in good approximation for small amplitudes, and a yo-yo-like oscillator which is non-linear even for small oscillations.

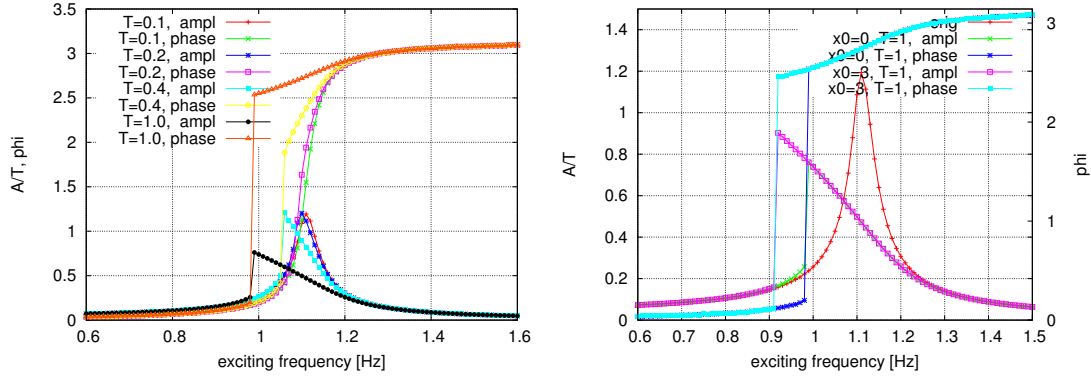
### 2.3.1 The mathematical pendulum

The differential equation which represents the approximate motion of the simple gravity pendulum shown in Fig. 5 is

$$ml\ddot{\phi} + \kappa\dot{\phi} - mg\sin(\phi) = F(t),$$

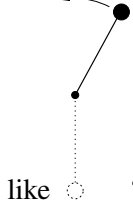
where  $\kappa$  is the dissipation term (coming from friction from the air).

The problem with this equation is that it is unintegrable. But for small oscillation amplitudes, one can approximate:  $\sin(\phi) = \phi$  and treat it as the harmonic, linear mechanical pendulum described in the

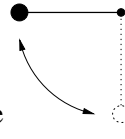


**Fig. 6:** Simulated behaviour of the mathematical pendulum

previous section. But what if we have large amplitudes like



like ?



or even a rotation of the pendulum

Well, this system is unbounded (rotation can occur instead of oscillation) and so the behaviour is obviously amplitude dependent. We especially expect the resonance frequency to be a function of the oscillation amplitude,  $\omega = F(A)$ . At least, we can still assume  $\omega = \omega_0$  for the steady state solution; this means that the system will follow the excitation frequency after some time.

Figure 6 shows the simulated behaviour of the mathematical pendulum in the steady state. You can see the single resonance peak, which for small amplitudes looks very similar to the one seen in Fig. 3. For larger amplitudes, however, this peak is more and more bent to the left. When the peak hangs over<sup>1</sup>, a jump occurs at an amplitude-dependent excitation frequency, where the system can oscillate with a small amplitude and then suddenly with a large amplitude. To make things even worse, the decision about which amplitude is taken by the system depends on the amplitude the system already has. Figure 6 (right) shows that the jump occurs at different frequencies, dependent on the amplitude  $x_0$  at the beginning of the simulation.

Last but not least, *coupled* systems of that type may have a very complicated dynamic behaviour and may easily become chaotic.

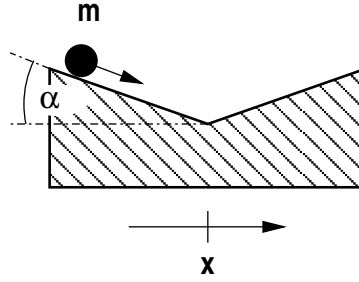
### 2.3.2 The yo-yo

Another strongly non-linear oscillator is the one known as the yo-yo and which is in principle identical to the system shown in Fig. 7.

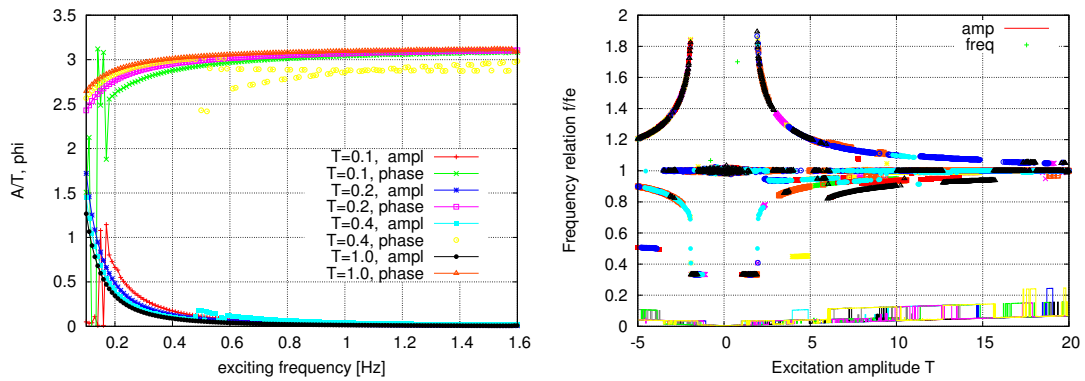
The differential equation of this system expresses like:

$$\frac{m}{\cos(\alpha)} \ddot{x} + \kappa \dot{x} - \text{sgn}(x) \cdot mg \sin(\alpha) = F(t),$$

<sup>1</sup>A similar emergence can be observed for superconducting cavities: Lorentz force detuning.



**Fig. 7:** The yo-yo. A mass  $m$  on the inclined plane. For simplicity, the rotation of the ball is not considered here.



**Fig. 8:** Simulated frequency response of the yo-yo for different excitation frequencies and amplitudes (left). On the right you can see different oscillation modes of this system depending on the excitation amplitude for different excitation frequencies. The system responds with different oscillation frequencies in an unpredictable manner.

where

$$\text{sgn}(x) := \begin{cases} \frac{x}{|x|} & x \neq 0 \\ 0 & x = 0 \end{cases}.$$

Now let us answer the questions: Is there a resonance? And if so, what is the resonance frequency?

Obviously, the resonance frequency here would also be highly amplitude-dependent ( $\omega_0 \stackrel{!}{=} f(A)$ ) because it takes longer for the ball to roll down the inclined plane if it starts with a bigger amplitude. But if we look at the simulated frequency response with different excitation amplitudes (see Fig. 8) it looks like there is a resonance at 0 Hz!?

Looking closer at the situation one finds that the oscillation frequency can differ from the excitation frequency:  $\omega \neq \omega_0$ . Figure 8 (right) shows all possible oscillation frequencies (in relation to the excitation frequency) with different starting amplitudes  $x_0$  (colours) under excitation with different amplitudes. The system responds with oscillations in an unpredictable manner.

Now you know why linear systems are so nice and relatively easy to deal with.

### 3 Signal theory

The fundamental concepts we want to deal with for digital signal processing are *signals* and *systems*. In this section we want to develop the mathematical understanding of a signal in general, and more specifically look at the *digital* signals.

### 3.1 Signals

The signal  $s(t)$  which is produced by a measurement device can be seen as a real, time-varying property (a function of time). The property represents physical observables like voltage, current, temperature, etc. Its instant power is defined as  $s^2(t)$  (all proportional constants are set to one<sup>2</sup>).

The signal under investigation should be an *energy signal*, which is

$$\int_{-\infty}^{\infty} s^2(t) dt < \infty. \quad (5)$$

This requires that the total energy content of that signal be finite. Most of the elementary functions (e.g.,  $\sin()$ ,  $\cos()$ ,  $\text{rect}()$ , ...) are not energy signals, because they ideally are infinitely long, and the integral (5) does not converge. In this case one can treat them as *power signals*, which requires

$$\lim_{T \rightarrow \infty} \int_{-T/2}^{T/2} s^2(t) dt < \infty. \quad (6)$$

(The energy of the signal is finite for any given time interval.) Obviously  $\sin()$  and  $\cos()$  are signals which fulfill the relation (6).

Now, what is a physical signal that we are likely to see? Well, wherever the signal comes from, whatever sensor is used to measure whatever quantity, in the end — if it is measured electrically — we usually get a voltage as a function of time  $U(t)$  as (input) signal. This signal can be *discrete* or *continuous*, *analog* or *digital*, *causal* or *non-causal*. We shall discuss these terms later.

From the mathematical point of view we have the following understanding/definitions:

- Time:  $t \in \mathbb{R}$  (sometimes  $\in \mathbb{R}_0^+$ )
- Amplitude:  $s(t) \in \mathbb{R}$  (usually a voltage  $U(t)$ )
- Power:  $s^2(t) \in \mathbb{R}_0^+$  (constants are renormed to 1)

Since the goal of *digital signal processing* is usually to measure or filter continuous, real-world analog signals, the first step is usually to convert the signal from an analog to a digital form by using an *analog-to-digital converter*. Often the required output is another analog signal, so a *digital-to-analog converter* is also required.

The algorithms for signal processing are usually performed using specialized electronics, which either make use of specialized microprocessors called *digital signal processors* (DSPs) or they process signals in real time with purpose-designed *application-specific integrated circuits* (ASICs). When flexibility and rapid development are more important than unit costs at high volume, digital signal processing algorithms may also be implemented using *field-programmable gate arrays* (FPGAs).

#### Signal domains

Signals are usually studied in one of the following domains:

1. time domain (one-dimensional signals),
2. spatial domain (multidimensional signals),
3. frequency domain,
4. autocorrelation domain, and

---

<sup>2</sup>For example, the power considering a voltage measurement would be  $P = U^2/R$ , considering a current measurement  $P = I^2 R$ , so we can set  $R := 1$  and get the relations  $P = U^2$  or  $P = I^2$ .



## 5. wavelet domains.

We choose the domain in which to process a signal by making an informed guess (or by trying different possibilities) as to which domain best represents the essential characteristics of the signal. A sequence of samples from a measuring device produces a time or spatial domain representation, whereas a discrete Fourier transform produces the frequency domain information, the frequency spectrum. Autocorrelation is defined as the cross-correlation of the signal with itself over varying intervals of time or space. Wavelets open various possibilities to create localized bases for decompositions of the signal. All these topics will be covered in the following sections. First we are going to look at how one can obtain a (digital) signal and what quantities define its quality. Then we are going to look at special fundamental signals and linear systems which transform these signals.

**Discrete-time signals**

Discrete-time signals may be *inherently discrete-time* (e.g., turn-by-turn beam position at one monitor) or may have originated from the sampling of a continuous-time signal (**digitization**). Sampled-data signals are assumed to have been sampled at periodic intervals  $T$ . The sampling rate must be sufficiently high to extract all the information in the continuous-time signal, otherwise **aliasing** occurs. We shall discuss issues relating to amplitude **quantization**, but, in general, we assume that discrete-time signals are continuously valued.

**3.2 Digitization**

The digitization process makes out of an analog signal  $s(t)$  a series of *samples*

$$s(t) \longrightarrow s_n := s[n] := s(nT) \quad n \in \mathbb{Z} \text{ (sometimes } \in \mathbb{N}_0)$$

by choosing discrete sampling intervals  $t \longrightarrow nT$  where  $T$  is the period.

The sampling process has two effects:

1. time discretization (sampling frequency)  $T = 1/f_s$  and
2. quantization (AD conversion, integer/float).

The second effect must not be neglected, although in some cases there is no special problem with this if you can use a high enough number of bits for the digitization. Modern fast ADCs have 8, 14 or 16 bits resolution. High-precision ADCs exist with 20 or even more effective bits, but they are usually much slower. Figure 9 illustrates the digitization process.

**Dithering**

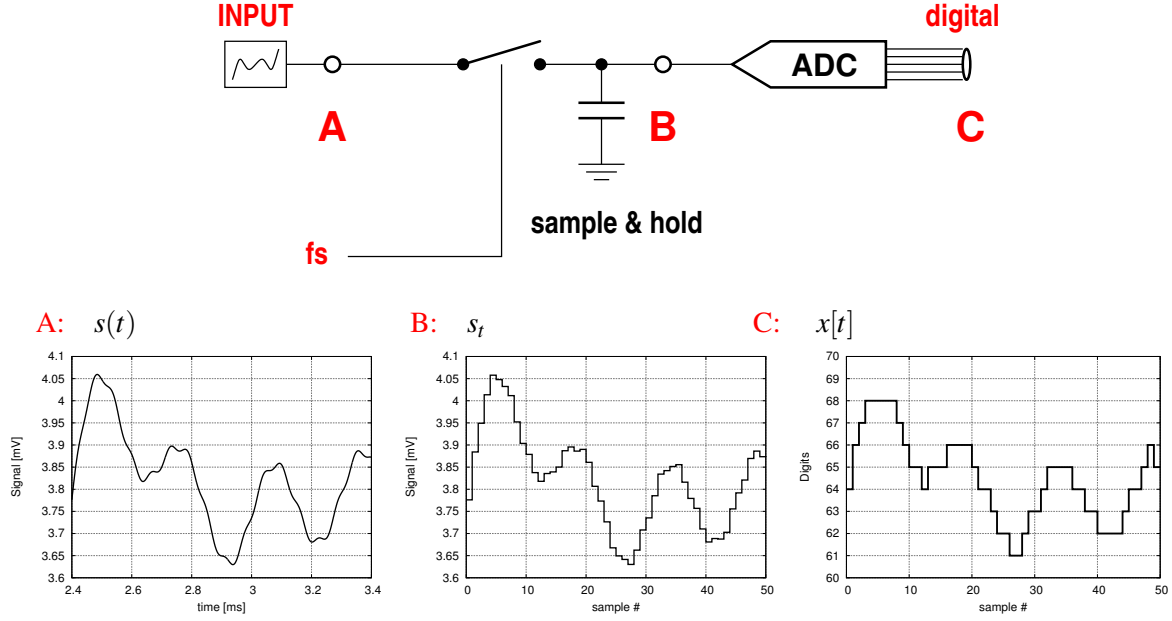
Because the number of bits of ADCs is a cost issue, there is a technique called *dithering* which is frequently used to improve the (amplitude) resolution of the digitization process. Surprisingly, it makes use of noise which is added to the (analog) input signal. The trick is that you can subtract the noise later from the digital values, assuming you know the exact characteristics of the noise, or even better, you produce it digitally using a DAC, and therefore know the value of each noise sample. This technique is illustrated in Fig. 10.

**3.3 Causal and non-causal signals**

A signal is *causal* if (at any time) only the *present* and *past* values of that signal are known.

$$\text{given } x[t_n] \quad \text{where } t_0 := \text{present}, \quad n < 0 : \text{future}, \quad n > 0 : \text{past}$$

So if  $x[t_n] = 0 \quad \forall n < 0$  the *system* under investigation is causal.



**Fig. 9:** The digitization process is done in two steps: First, samples are taken from the analog input signal (A). The time discretization is done with the sampling frequency  $f_s$ . The voltage is stored in a sample-and-hold device (B) (a simple capacitor can do). Finally the voltage across the capacitor is converted into a digital number (C), usually represented by  $n$  bits of digital logic signals. The digital representation of the input signal is not perfect (as can be seen on the bottom plots) as it has a limited resolution in both time and amplitude.

The only situation where you may encounter non-causal signals or non-causal algorithms is under the following circumstances: Say, a whole chunk of data has been recorded (this can be the whole pulse train in a repetitive process or the trace of a pulse of an RF system). Now you want to calculate a prediction for the next measurement period from the last period's data. From some viewpoint, this data is seen as a non-causal signal: If you process the data sample by sample, you always have access to the whole dataset, which means you can also calculate with samples before the sample actually processes. You can thereby make use of non-causal algorithms, because from this algorithm's perspective your data also contains the future. But from the outside view, it is clear that it does not really contain the future, because the whole chunk of data has been taken in the past and is now processed (with a big delay). A measurement can not take information from the future! Classically, nature or physical reality has been considered to be a causal system.

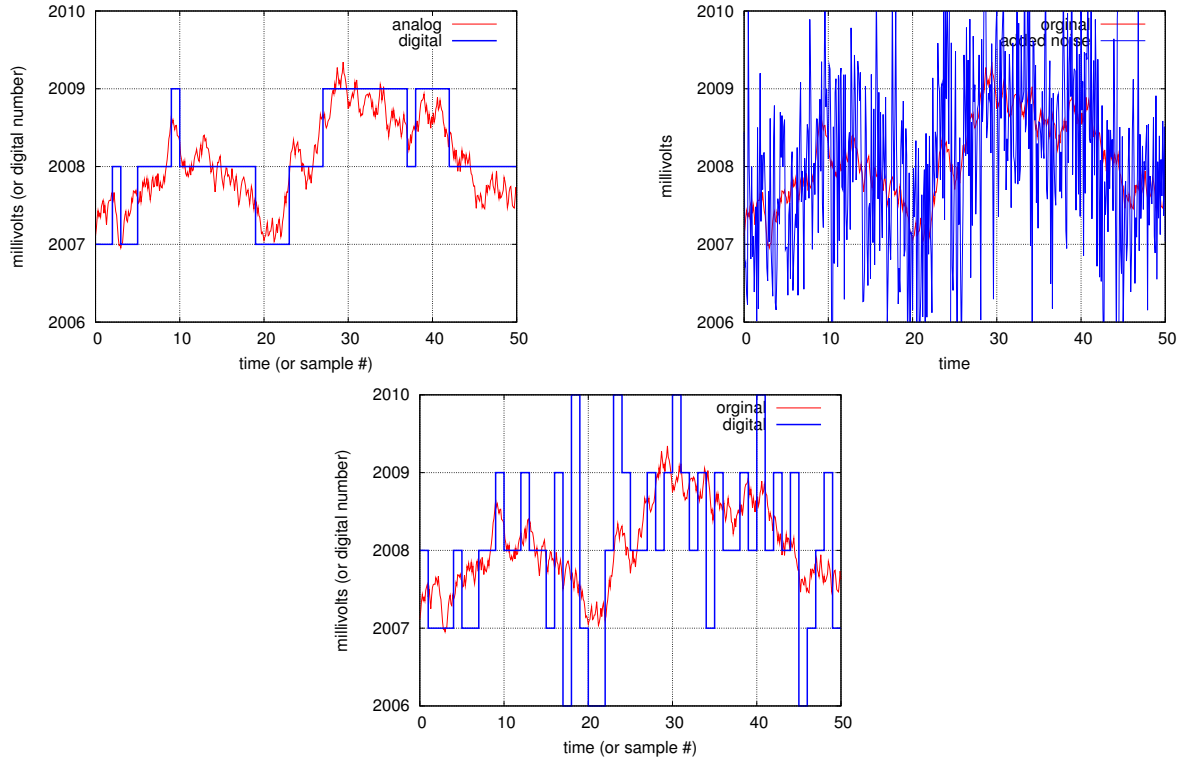
### 3.3.1 Discrete-time frequency units

In the discrete world, you deal with numbers or digits instead of voltage, with sample number instead of time, and so we ask what is the discrete unit of frequency? Let us go straightforward starting with an analog signal:

$$x(t) = A \cdot \cos(\omega t) =: A \cdot \cos(2\pi f_c t) ,$$

sampling at intervals  $T = \frac{1}{f_s} = \frac{2\pi}{\omega_s}$  leads to

$$\begin{aligned} \Rightarrow x[n] &= A \cdot \cos(\omega n T) \\ &= A \cdot \cos\left(n \frac{\omega}{f_s}\right) = A \cdot \cos\left(n \frac{2\pi\omega}{\omega_s}\right) \\ &=: A \cdot \cos(\omega_d n) , \end{aligned}$$



**Fig. 10:** The *dithering* technique makes use of (random) noise which is added to the analog signal. If this noise is later removed from the digital signal (e.g. using a digital low pass filter or statistics) the accuracy of the digital values can be improved. The best method would be the **subtractive dither**: produce the ‘random’ noise by a DAC and subtract the known numbers later.

where

$$\omega_d = \frac{2\pi\omega}{\omega_s} = \omega T \quad (7)$$

is the *discrete time frequency*. The units of the discrete-time frequency  $\omega_d$  are **radians per sample** with a range of

$$-\pi < \omega_d \leq \pi \quad \text{or} \quad 0 \leq \omega_d < 2\pi .$$

### 3.4 The sampling theorem

Proper sampling means that you can exactly reconstruct the analog signal from the samples. *Exactly* here means that you can extract the ‘key information’ of the signal out of the samples. One basic key information is the frequency of a signal. Figure 11 shows different examples of proper and not proper sampling. If the sampling frequency is too low compared with the frequency of the signal, a signal reconstruction is not possible anymore. The artefacts which occur here are called *aliasing*.

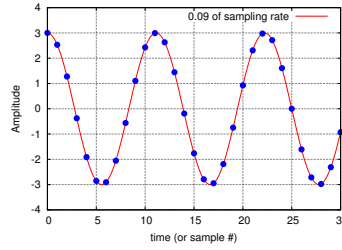
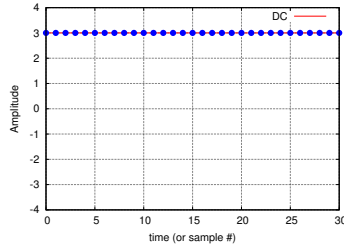
To express a condition, when a signal is properly sampled, a sampling theorem can be formulated. This theorem is also known as *the Nyquist/Shannon theorem*. It was published in 1940 and points out one of the most basic limitations of the sampling in digital signal processing.

Given  $f_s \triangleq$  sampling rate:

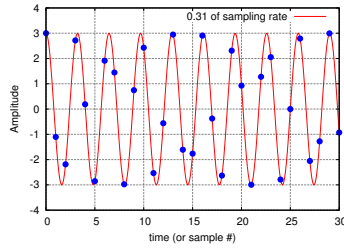
“A continuous signal can be properly sampled if it does not contain frequency components above

$$f_{\text{crit}} = \frac{f_s}{2} , \quad \text{the so-called Nyquist frequency” .$$

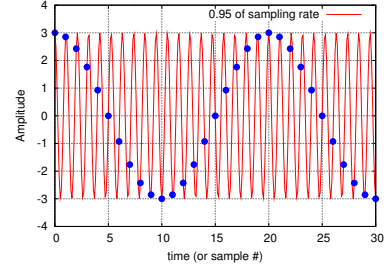
**Proper:**



**Still proper:**



**Not proper:**



**"aliasing"**

**Fig. 11:** Different examples of proper and not proper sampling. If the sampling frequency is too low compared with the frequency of the signal, a signal reconstruction is not possible anymore.

Frequency components which are larger than this critical frequency ( $f > f_{\text{crit}}$ ) are **aliased** to a mirror frequency  $f^* = f_{\text{crit}} - f$ .

The sampling theorem has consequences on the choice of the sampling frequency you should use to sample your signal of interest. The digital signal cannot contain frequencies  $f > f_{\text{crit}}$ . Frequencies greater than  $f_{\text{crit}}$  will add up to the signal components which are still properly sampled. This results in information loss at the lower frequency components because their signal amplitudes and phases are affected. So except for special cases (see undersampling and down-conversion) you need

1. a proper choice of sampling rate and
2. an *anti-aliasing filter* to limit the input signal spectrum.

Otherwise your signal will be affected by aliasing (see Fig. 12).

### 3.4.1 Mathematical explanation of aliasing

Consider a continuous-time sinusoid  $x(t) = \sin(2\pi ft + \phi)$ . Sampling at intervals  $T$  results in a discrete-time sequence

$$x[n] = \sin(2\pi fTn + \phi) = \sin(\omega_d n + \phi) .$$

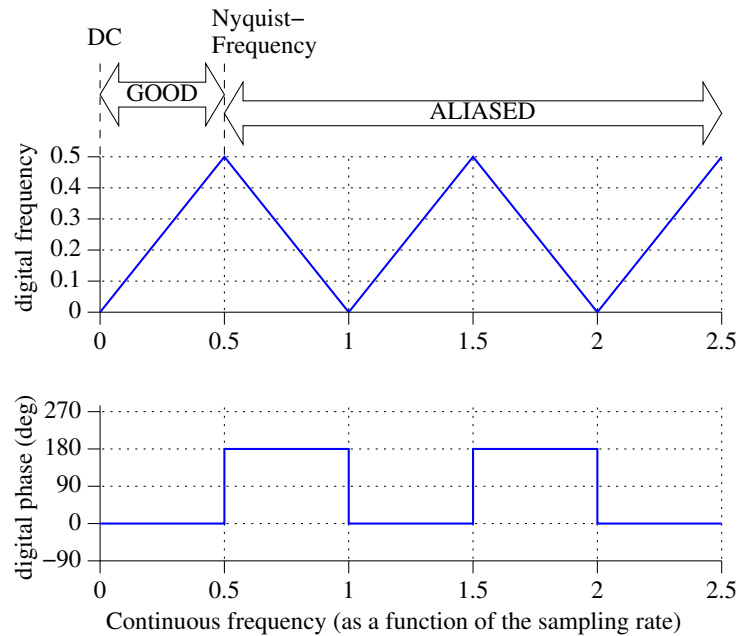
Since the sequence is unaffected by the addition of any integer multiple of  $2\pi$ , we can write

$$x[n] = \sin(2\pi fTn \pm 2\pi m + \phi) = \sin(2\pi T(f \pm \frac{m}{Tn})n + \phi) .$$

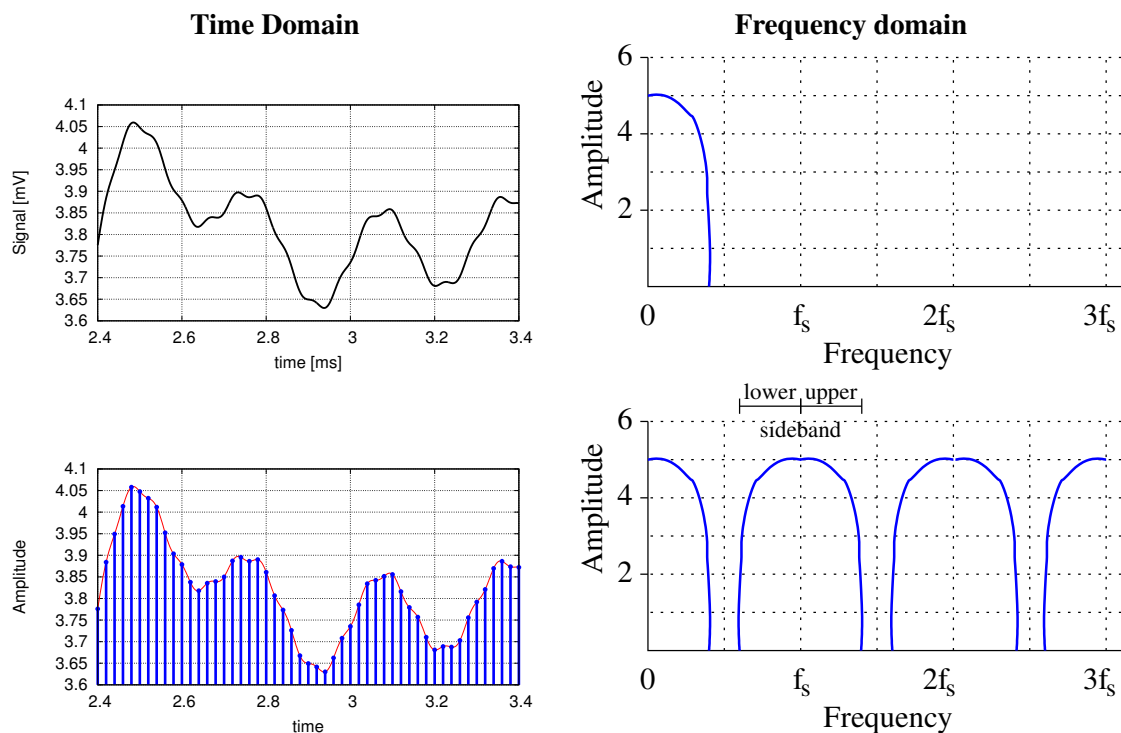
Replacing  $\frac{1}{T}$  by  $f_s$  and picking only integers  $m = kn$  we get

$$x[n] = \sin(2\pi T(f \pm kf_s)n + \phi) .$$

This means: when sampling at  $f_s$ , we cannot distinguish between  $f$  and  $f \pm kf_s$  by the sampled data, where  $k$  is an integer.



**Fig. 12:** Mapping of the analog frequency components of a continuous signal to the digital frequencies. There is a good area where the frequencies can be properly reconstructed and several so-called Nyquist bands where the digital frequency is different. Also the phase jumps from one Nyquist band to the other.



**Fig. 13:** Aliasing example. In frequency domain the continuous signal has a limited spectrum. The sampled signal can be seen as a pulse train of sharp ( $\delta$ -)pulses which are modulated with the input signal. So the resulting spectrum gets side-bands which correspond to the Nyquist bands seen from inside the digital system. By the way: the same applies if you want to convert a digital signal back to analog.

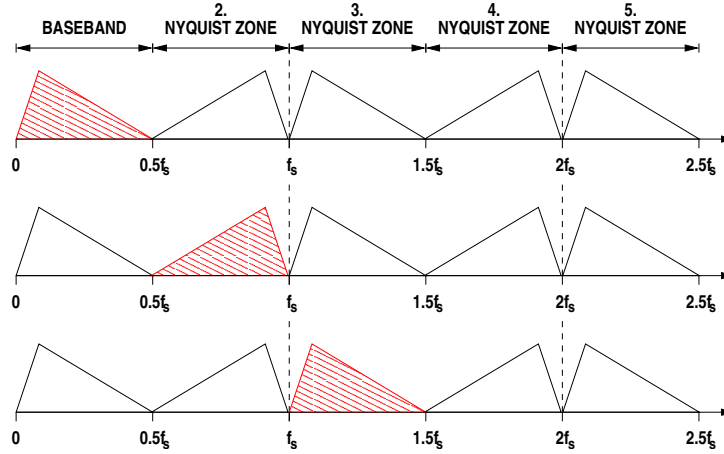


Fig. 14: Principle of undersampling

The aliasing can also be seen the other way round: Given a continuous signal with a limited spectrum (see Fig. 13). After sampling we cannot distinguish if we originally had a continuous and smooth signal or a signal consisting of a pulse train of sharp ( $\delta$ )-pulses which are modulated corresponding to the input signal. Such a signal has side-bands which correspond to the Nyquist bands seen from inside the digital system. The same principle applies if you want to convert a digital signal back to analog.

This concept can be further generalized: Consider the sampling process as a **time-domain multiplication** of the continuous-time signal  $x_c(t)$  with a sampling function  $p(t)$ , which is a periodic impulse function (Dirac comb). The frequency-domain representation of the sampled data signal is the **convolution** of the frequency domain representation of the two signals, resulting in the situation seen in Fig. 13. If you do not understand this by now, never mind. We shall discuss the concept of convolution in more detail later.

### 3.4.2 Undersampling

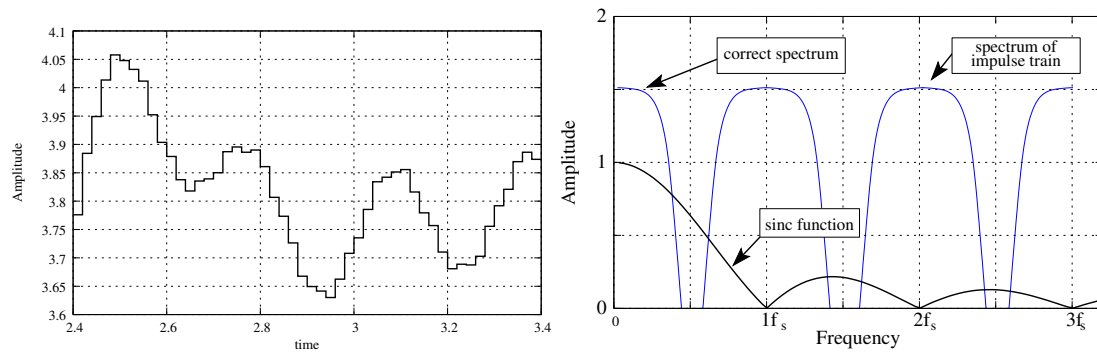
Last but not least, I want to mention a technique called *undersampling*, *harmonic sampling* or sometimes also called *digital demodulation* or *downconversion*. If your signal is modulated onto a carrier frequency and the spectral band of the signal is limited around this carrier, then you may take advantage of the ‘aliasing’. By choosing a sampling frequency which is lower than the carrier but synchronized with it (this means it is exactly a fraction of the carrier), you are able to demodulate the signal. This can be done with the spectrum of the signal lying in any Nyquist zone given by the sampling frequency (see Fig. 14). Just keep in mind that the spectral components may be reversed and also the phase of the signal can be shifted by  $180^\circ$  depending on the choice of the zone. And also — of course — any other spectral components which leak into the neighboring zones need to be filtered out.

## 3.5 Analog signal reconstruction

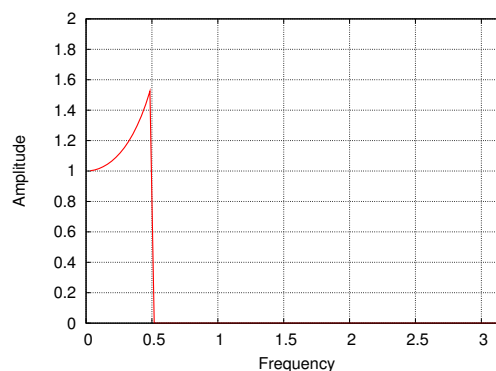
As mentioned before, similar problems, like aliasing for analog-to-digital conversion (ADC), also apply to **Digital-to-Analog Conversion (DAC)**! Usually, no impulse train is generated by a DAC, but a **zero-order hold** is applied. This modifies the output amplitude spectrum by multiplication of the spectrum of the impulse train with

$$H(f) = \left| \text{sinc}\left(\frac{f}{f_s}\right) \right| := \left| \frac{\sin(\pi f / f_s)}{\pi f / f_s} \right|,$$

which can be seen as a convolution of an impulse train with a rectangular pulse. The functions are illustrated in Fig. 15.



**Fig. 15:** Frequency response of the zero-order hold (right) which is applied at the DAC and generates the step function (left)

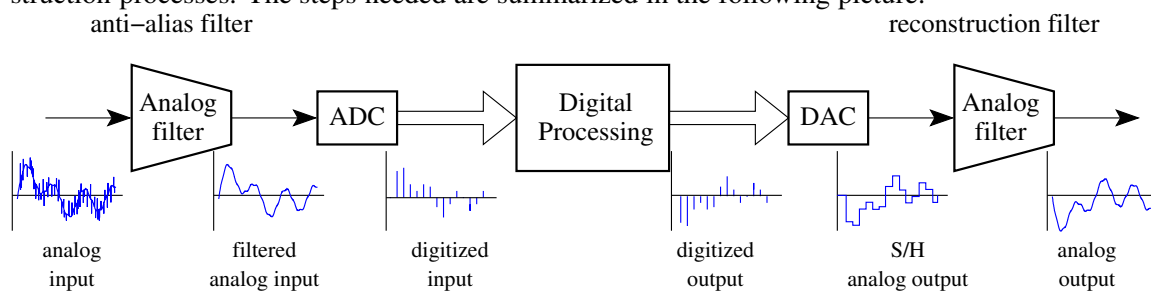


**Fig. 16:** Transfer function of the (ideal) reconstruction filter for a DAC with zero-order hold

As you can imagine, this behaviour appears to be unpleasant because now, not only components of the higher order sidebands of the impulse train spectrum are produced on the output (though attenuated by  $H(f)$ ), but also the original spectrum (the baseband) is shaped by it. To overcome this ‘feature’, a *reconstruction filter* is used. The reconstruction filter should remove all frequencies above one half of  $f_s$  (an analog filter will be necessary, which is sometimes already built into commercial DSPs), and boost the frequencies by the reciprocal of the zero-order-hold’s effect ( $\frac{1}{\text{sinc}()})$ . This can be done within the digital process itself! The transfer function of the (ideal) reconstruction filter is shown in Fig. 16.

### 3.6 Anti-aliasing techniques

Putting it all together, digital signal processing needs additional care concerning the sampling and reconstruction processes. The steps needed are summarized in the following picture:



To design your digital signal processing system, you need to know about (analog) *filter design*, the

*characteristics* of anti-aliasing and reconstruction filters, and about *limitations* of signal processing like bandwidth and noise of the analog parts and, for the digital parts, sampling frequency and quantization.

## 4 Noise

The terms *error* and *noise* are closely related. Noise is some fluctuation on the input signal which can come from different sources, can have different spectral components and in many cases (except for the dithering methods) is unwanted. It can cover the information you want to extract from the signal and needs to be suppressed with more or less advanced techniques. Usually, some of the noise components can hardly be avoided and, therefore, we shall have to deal with it. Noise on the signal can cause an *error*. But there are also errors which do not come from noise. We therefore distinguish between *systematic* (deterministic) errors on the one hand and unsystematic (statistical) errors (or *noise*) on the other hand. We are going to take a closer look at this distinction.

**Systematic error**  $\longleftrightarrow$  **accuracy** comes from characteristics of the measurement device (ADC/DAC: *offset, gain, linearity-errors*). It can be improved by improvements of the apparatus, like calibration. The only limits here come from the practical usefulness and from quantum mechanics, which keeps you from measuring certain quantities with absolute accuracy.

**Statistical error** comes from unforeseen random fluctuations, stochastics, and noise. It is impossible to avoid them completely, but it is possible to estimate the extent and it can be reduced through statistical methods (averaging), multiple repetitive measurements etc. This determines the **precision** of the measurement.

Note that the definition is context dependent: The accuracy of 100 devices can be a matter of precision! Imagine that you measure the same property with 100 different devices where each device has a slightly different systematic (calibration) error. The results can now be distributed in much the same way as they are with a statistical measurement error — and so they can be treated as statistical errors, in this case, and you might want to use the statistical methods described in the following section.

The distinction above leads to the terms *accuracy* and *precision*, which we shall define in the following sections. Besides this, we want to deal with the basic concepts of statistics which include

- random variables and noise (e.g., white noise, which has an equal distribution, Gaussian noise, which has a Gaussian distribution, and  $1/f$  or pink noise, which is  $1/f$  distributed),
- the mean and the standard deviation, variance, and
- the normal or Gaussian distribution.

### 4.1 Basic statistics

#### 4.1.1 Mean and standard deviation

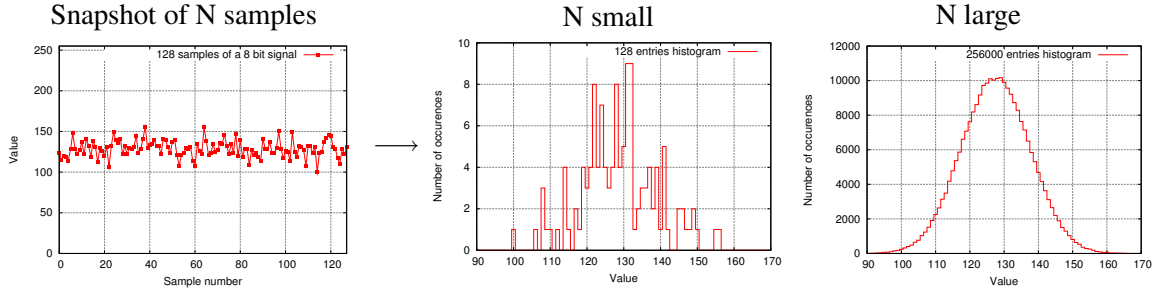
Assuming that we do  $N$  measurements of a quantity which result in a series of measurement values  $x_i$ , the *mean* (or average) over  $N$  samples can be calculated as:

$$\hat{x} := \frac{1}{N} \sum_{i=0}^{N-1} x_i .$$

The *variance*  $\sigma^2$  ( $\sigma$  itself is called *standard deviation*) is a measure of the ‘power of fluctuations’ of the set of  $N$  samples. It is a direct measure of the precision of the signal.

$$\sigma^2 := \frac{1}{N-1} \sum_{i=0}^{N-1} (x_i - \hat{x})^2 . \quad (8)$$





**Fig. 17:** Creating a histogram from a snapshot of samples

Equation (8) can also be written in the following form:

$$\sigma_N^2 = \frac{1}{N-1} \left[ \underbrace{\sum_{i=0}^{N-1} x_i^2}_{\text{sum of squares}} - \frac{1}{N} \left( \underbrace{\sum_{i=0}^{N-1} x_i}_{\text{sum}} \right)^2 \right],$$

which is useful, if you want to calculate running statistics ‘on the fly’.

There are also quantities which are derived from the mean and the variance like

$$\text{The Signal to Noise Ratio (SNR): } \text{SNR} = \frac{\hat{x}^2}{\sigma^2}, \quad (9)$$

$$\text{the Coefficient of Variation (CV): } \text{CV} = \frac{\sigma}{\hat{x}} \cdot 100\% \quad \text{and} \quad (10)$$

$$\text{the Root Mean Square (RMS): } x_{\text{rms}} := \sqrt{\frac{1}{N} \sum_{i=0}^{N-1} x_i^2}. \quad (11)$$

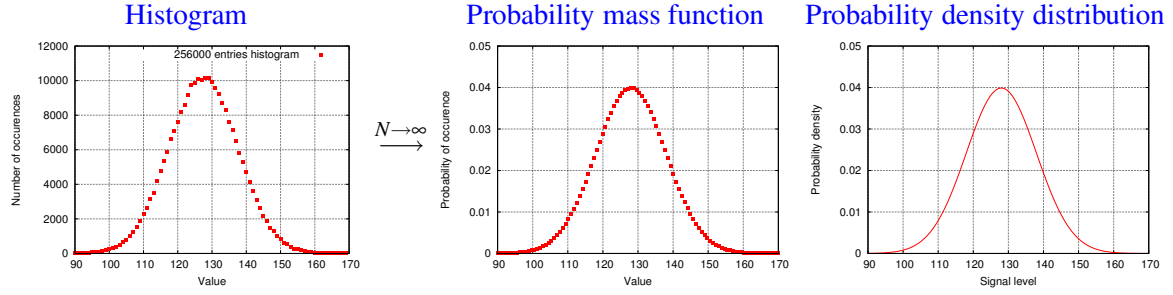
The latter is a measure of the ‘Power of fluctuations **plus** power of DC component’.

#### 4.1.2 Histograms and the probability density distribution

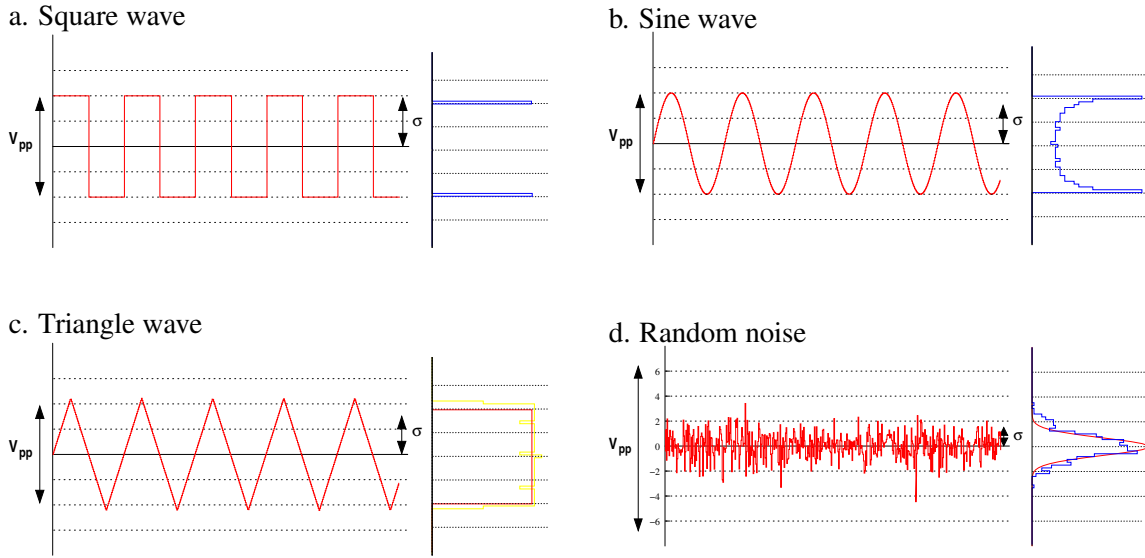
A common way to reduce the amount that must be processed is to use *histograms*. A snapshot of  $N$  samples is summed up in  $(M)$  bins (see Fig. 17). Each bin now contains the number of occurrences of a certain value (or range of values)  $H_i$  and the mean and variance can now be calculated using this histogram:

$$\begin{aligned} N &= \sum_{i=0}^{M-1} H_i, \\ \hat{x} &:= \frac{1}{N} \sum_{i=0}^{M-1} i \cdot H_i, \\ \sigma^2 &:= \frac{1}{N-1} \sum_{i=0}^{M-1} (i - \hat{x})^2 H_i. \end{aligned}$$

As you already saw in Fig. 17, with a large number of samples the histogram becomes smooth and it will converge in the limit  $N \rightarrow \infty$  to a distribution which is called the *probability mass function*. This is an approximation of the (continuous) *probability density distribution*. This is illustrated in Fig. 18. In this case, the fluctuations of the samples have a Gaussian distribution. Examples of probability mass functions and probability density distributions of common waveforms are shown in Fig. 19.



**Fig. 18:** Histogram, probability mass function, and probability density distribution



**Fig. 19:** Probability mass functions and probability density distributions of common waveforms

#### 4.1.3 The normal distribution

The best known and most common distribution is the *normal distribution* that has the form of a Gauss function:

$$P(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x - \hat{x})^2}{2\sigma^2}} .$$

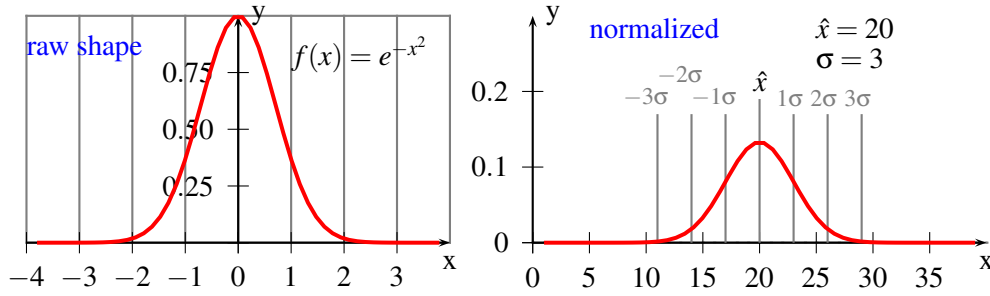
The Gauss formula is illustrated in Fig. 20. Note that the probability density is normalized, so that the integrated density is the overall probability. This should, of course, be equal to one:

$$\int_{-\infty}^{+\infty} P(x) dx = 1 .$$

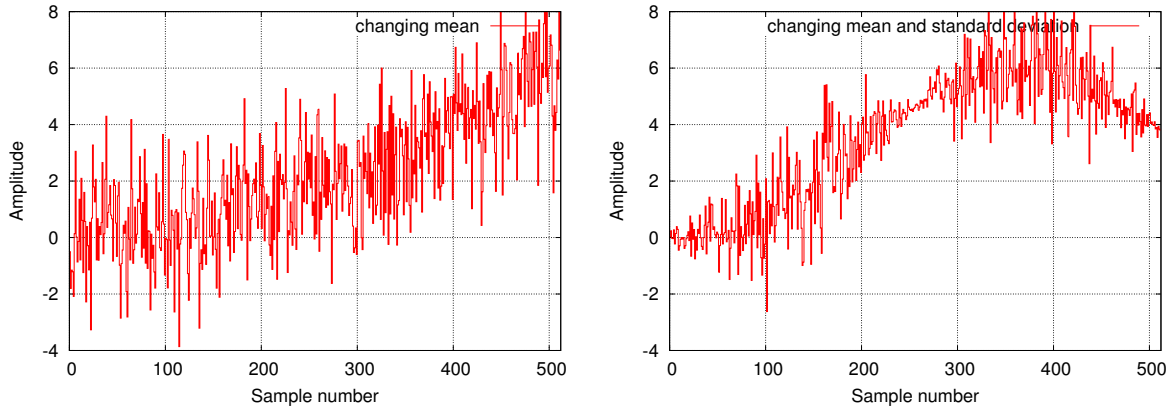
Now what is this good for? Imagine that we have  $N$  samples of a measured quantity. Then we can define the

$$\text{typical error: } \Delta A = \frac{\sigma_N}{\sqrt{N}} .$$

Here  $\sigma_N$  is an estimate of the standard deviation of the *underlying process* over  $N$  samples (e.g., extracted from the histogram). This is the best information about the underlying process you can extract out of the



**Fig. 20:** The raw shape and the normalized shape of the Gauss function. The area of one standard deviation  $\pm\sigma$  integrates to 68.3%, the area of  $\pm 2\sigma$  to 95.4%.



**Fig. 21:** A signal with changing mean and standard deviation

sampled signal. In practice, that means that the more samples you take, the smaller the typical error  $\Delta A$  is. But this can only be done if the underlying quantity does not change during the time the samples were taken. In reality, the quantity and also its fluctuations may change, as in Fig. 21, and it is a real issue to select the proper and useful number of samples to calculate the mean and standard deviation  $\sigma_N$  to get a good approximation of what the real process may look like. There is no such thing as an *instant error*; the probability density function cannot be measured, it can only be approximated by collecting a large number of samples.

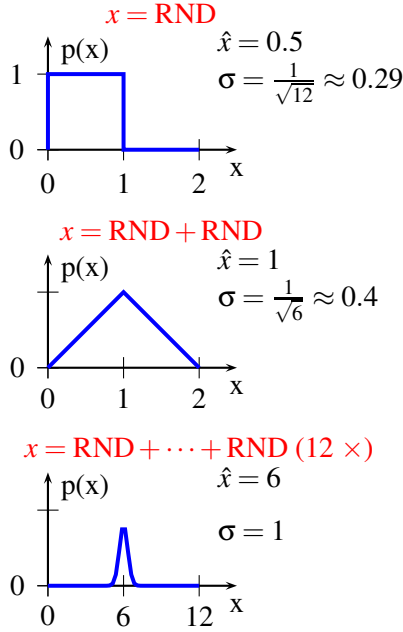
## 4.2 The central limit theorem

Why does a normal distribution occur so frequently? Why are most processes and most signals normally distributed? Why is it always a good assumption that the probability density distribution of an arbitrary measurement is Gaussian, and we know everything we can get about the underlying process if we know the measurement value  $A$  and its typical error  $\Delta A$ ?

This is the consequence of the *central limit theorem* which says:

*The sum of independent random numbers (of any distribution) becomes Gaussian distributed.*

The practical importance of the central limit theorem is that the normal distribution can be used as an approximation of some other distributions. Whether these approximations are sufficiently accurate depends on the application for which they are needed and the rate of convergence to the normal distribution. It is typically the case that such approximations are less accurate in the tails of the distribution.



**Fig. 22:** Consequence of the central limit theorem: Summing up more and more equally distributed random numbers will result to good approximation in a Gaussian distributed random variable

It should now be clear why most of your measurements may be Gaussian distributed. This is simply because the measurement process is a very complicated one with many different and independent error sources which all together contribute to the final measurement value. They do so without caring about the details of their mechanisms — as long as there are enough contributors, the result will be approximately Gaussian.

There is also a practical application of the theorem in computing. Suppose you need to generate numbers which have a Gaussian distribution. The task is quite easy; you just have to have a function which generates any kind of (pseudo-) random numbers and then sum up enough of them.

Here is an example: first generate *white noise* using a function which produces equally distributed random numbers between zero and one  $\text{RND} := [0; 1[$ . This is often implemented in the form of a pseudo random generator which calculates

$$\text{RND} = (as + b) \bmod c ,$$

where  $s$  is the *seed* and  $a, b$  and  $c$  are appropriately chosen constants. The new random number is used as a seed for the next calculation and so on.

The distribution of this function is shown in Fig 22, top. If you now add two such random numbers, the result will have a distribution as shown in the figure in the centre. After adding 12 random numbers you already get a very good approximation of a Gaussian distribution with a standard deviation of  $\sigma = 1$  and a mean value of  $\hat{x} = 6$ . If you subtract 6 from this sum, you are done. But do not really implement it like this, because there is a simpler formula which only uses 2 random variables and will also do a good job ( $\hat{x} = 0, \sigma = 1$ ):

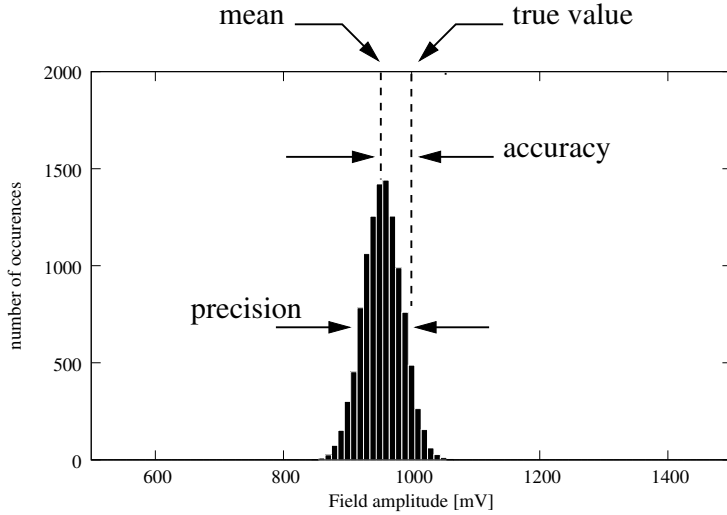
$$x = \sqrt{-2 \log_{10}(\text{RND}_1)} \cdot \cos(2\pi \text{RND}_2) .$$

### 4.3 Accuracy and precision

Having understood the probability density distribution of a series of measurement samples, it is now straightforward to define precision and accuracy. Figure 23 illustrates the difference.

To summarize:

- *accuracy* is a measure of *calibration*,
- *precision* is a measure of *statistics*.



**Fig. 23:** The difference between accuracy and precision: Accuracy is the difference between the true value and the mean of the underlying process that generated the data. Precision is the spread of the values coming from fluctuations, noise and any other statistical error. It is specified by the standard deviation or the signal noise ratio.

#### 4.3.1 Signal-to-noise ratio

Because it is a very common term in engineering, let us define the *signal-to-noise ratio* which is a measure of the relative error of a signal. From the statistical mathematics point of view we already defined it in Eq. (9). But maybe you are more familiar with the following definitions which deal with the power  $P$  and the amplitude  $A$  of a signal. In these terms, the signal-to-noise ratio is the power ratio, averaged over a certain bandwidth of the power spectrum  $p(v)$ :

$$\text{SNR} := \frac{\bar{P}_{\text{signal}}}{\bar{P}_{\text{noise}}} = \left( \frac{\hat{A}_{\text{signal,rms}}}{\hat{A}_{\text{noise,rms}}} \right)^2,$$

$$\bar{P} := \int_{\text{BW}} p(v) dv.$$

Quantities which come from ratios are very often — for practical reasons (you avoid multiplication and division) — expressed in decibels, a logarithmic pseudo-unit:

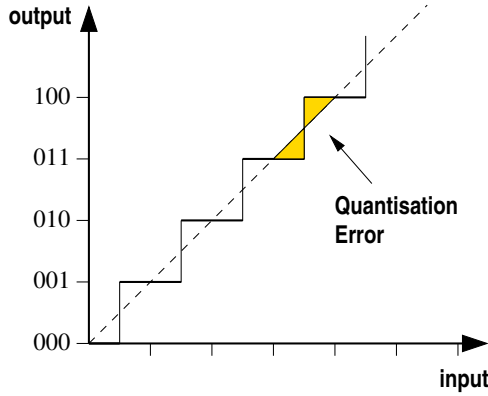
$$\begin{aligned} \text{SNR(dB)} &:= 10 \log_{10} \left( \frac{\bar{P}_{\text{signal}}}{\bar{P}_{\text{noise}}} \right) = 20 \log_{10} \left( \frac{\hat{A}_{\text{signal,rms}}}{\hat{A}_{\text{noise,rms}}} \right)^2 \\ &= P_{\text{signal}}[\text{dBm}] - P_{\text{noise}}[\text{dBm}]. \end{aligned}$$

A similar ‘unit’ is used if you talk about the carrier as reference:  $[\text{SNR(dB)}] = \text{dBc}$  (=‘dB below carrier’), and so you can also define a  $\text{CNR} = \text{carrier-to-noise ratio}$ .

#### 4.4 Error sources in digital systems

From the digital processing, the digitization, and the analog reconstruction of the signals, there are various sources of errors:

1. **Systematic errors:** Most importantly, ADC and DAC distortions: e.g. offset, gain and linearity errors. These types of errors can be corrected for through calibration.
2. **Stochastic errors:** quantization noise, quantization distortions, as well as aperture and sampling errors (clock jitter effects).
3. **Intrinsic errors:** DAC-transition errors and glitches. They are random, unpredictable, and sometimes systematic, but it is hard to correct the source of these errors, and so they need to be filtered.



**Fig. 24:** Transfer function of an ADC. The quantization noise comes from the difference between the continuous (analog) input signal level and the signal level represented by the digital number produced by the ADC. Because the ADC has a finite resolution, this error can be no more than  $\pm \frac{1}{2}$  of the step height.

The systematic errors can in principle be corrected for through calibration, and this is also the recommended way to treat them wherever possible. The intrinsic errors are hard to detect, may cause spurious effects and therefore make life really bad. If they bother you, a complete system analysis and probably a rework of some components may be required to cure them. There is (nearly) no way to overcome them with some sort of data processing. Therefore we focus here on the stochastic errors, because the way we treat them with data processing determines the quality of the results. At least, we can improve the situation by use of sophisticated algorithms which, in fact, can be implemented in the digital processing system more easily than in an analog system.

#### 4.4.1 Quantization noise

The transfer function of an analog-to-digital converter (ADC) is shown in Fig. 24. The quantization noise comes from the difference between the continuous (analog) input signal level and the signal level represented by the digital number produced by the ADC. Because the ADC has a finite resolution, this error can be no more than  $\pm \frac{1}{2}$  of the step height (least significant bit resolution  $|A| < 0.5\text{LSB}$ ). The RMS error of the quantization noise is

$$\text{RMS}(\Delta A) \approx \sqrt{12}\text{LSB}.$$

Although this error is not really independent of the input value, from the digital side it actually is, because there is no control when the least significant bit flips. It is, therefore, best to treat this error as a (quantization) noise source.

For a full-scale  $\sin()$  signal, the signal-to-noise ratio coming from the quantization noise is

$$\text{SNR} = 6.02n + 1.76\text{dB} + 10 \log \left( \frac{f_s}{2\text{BW}} \right). \quad (12)$$

As you can see, it increases with lower BW. This means that doubling the sampling frequency increases the SNR by 3dB (at the same signal bandwidth). This is effectively used with so-called ‘oversampling’ schemes. Oversampling is just a term describing the fact that with a sampling frequency that is much higher than would be required by the Nyquist criterium, you can compensate for the quantization noise caused by a low ADC bit resolution. Especially for 1-bit ADCs, this is a major issue.

In Eq. (12), it is assumed that the noise is equally distributed over the full bandwidth. This is often not the case! Instead, the noise is often *correlated* with the input signal! The lower the signal, the more correlation. In the case of strong correlation, the noise is concentrated at the various harmonics of the input signal; this is exactly where you do not want them. Dithering and a broad input signal spectrum randomizes the quantization noise.

Nevertheless, this simple quantization noise is not the only cause of errors in the analog-to-digital conversion process. There are two common, related effects: *missing codes* and *code transition noise*. These effects are intrinsic to the particular ADC chip in use. Some binary codes will simply not be produced because of ADC malfunction as a consequence of the hardware architecture and internal algorithm responsible for the conversion process. Especially for ADCs with many bits, this is an issue. Last but not least, the ADC may show code transition noise; this means that the output oscillates between two steps if the input voltage is within a critical range even if the input voltage is constant.

## 5 Linear systems

You now know some of the main consequences, advantages, and limitations of using digitized signals. You know how to deal with aliasing, downsampling, and analog signal reconstruction. You know the concepts of noise and the basic mathematical tools to deal with it.

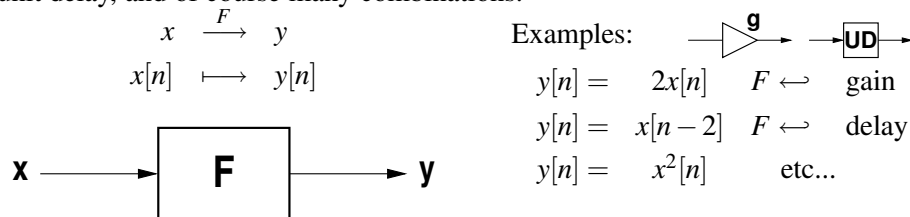
Next, we are going to look more closely at the **systems** which transform the (digital) signals. Of course, there are *analog* systems as well as *digital* ones. But, since there are not many conceptual differences, we can focus mainly on the digital ones. The analogy to analog system concepts will be drawn from whenever useful.

We are also going to use different notations in parallel: besides the mathematical notation, we show the rather symbolic expressions commonly used in engineering fields. In contrast to the mathematical notation, which is slightly different for analog systems (e.g.  $y(t) = 2x(t)$ ) and digital systems (e.g.  $y[n] = 2x[n]$ ), the latter does not make a formal difference here. Both concepts and notations are in use in different books on the field. They are, however, easy to understand, so you will quickly become familiar with both notations.

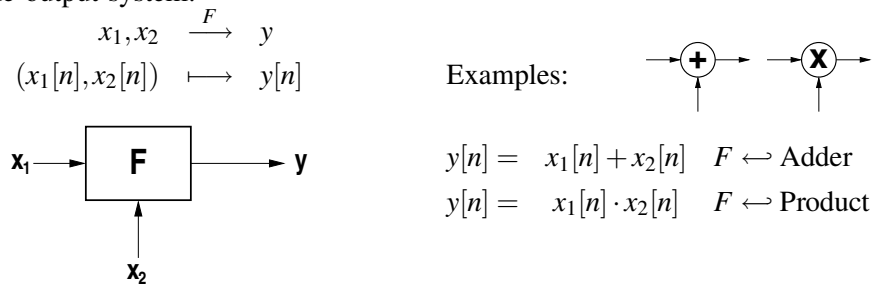
### 5.1 Discrete-time systems

A **system** receives one or more **inputs** and generates one or more **outputs** dependent on the inputs. We distinguish between three kinds of systems:

1. **MIMO** (Multiple-Input-Multiple-Output) systems; these are the most general.
2. **SISO** (Single-Input-Single-Output) systems; such are many of the elementary systems, e.g. gain and the unit delay, and of course many combinations:



3. and **MISO** (Multiple-Input-Single-Output) systems; here the adder is the most popular double-input-single-output system:



Besides this, there is also a way to split signals. This produces a generic Single-Input-Double-Output system.

Starting from elementary systems, the concept of *superposition* allows us to combine systems to create more complex systems of nearly any kind.

## 5.2 Superposition

Systems may be of any complexity. It is, therefore, convenient to look at them as a composition of simpler components. If we restrict ourselves to the class of *linear systems*, it is possible to first decompose the input signals and then process them with simple systems. In the end, the result will be synthesized by superposition for prediction of the output. In this way, we can split up the problems into many pieces of simpler complexity, and even use only a few fundamental systems. Without the concept of decomposition and linear systems, we would be forced to examine the individual characteristics of many unrelated systems, but with this approach, we can focus on the traits of the linear system category as a whole.

Although most real systems found in nature are not linear, most of them can be well approximated with a linear system, at least for some limited range of ‘small’ input signal amplitudes.

## 5.3 Causal, linear, time-invariant systems

Systems under investigation in this lecture should therefore be *linear*, *causal*, and *time invariant*. We shall see what this means in detail.

### 5.3.1 Linearity

Given system  $F$  with  $F(x_1[n]) = y_1[n]$  and  $F(x_2[n]) = y_2[n]$ , then  $F$  is said to be *linear* if

$$F(x_1[n] + x_2[n]) = F(x_1[n]) + F(x_2[n]) ,$$

(it follows that  $F(x[n] + x[n]) = F(2x[n]) = 2F(x[n])$ ), and for two linear systems  $F_1$  and  $F_2$

$$F_1(F_2(x[n])) = F_2(F_1(x[n])) .$$

### 5.3.2 Time-invariance

(also ‘shift-invariance’) Given  $F$  with  $F(x[n]) = y[n]$  is considered *time-invariant* if

$$F(x[n-k]) = y[n-k] \quad \forall k \in \mathbb{N} .$$

### 5.3.3 Causality

The system is *causal* if the output(s) (and internal states) depend only on the *present and past* input and output values.

$$\text{Causal: } y[n] = x[n] + 3x[n-1] - 2x[n-2]$$

$$\text{Non-causal: } y[n] = x[n+1] + 3x[n] + 2x[n-1] .$$

In the latter case the system  $Y$  produces its output  $y$  by using an input value of the input signal  $x$  which is ahead of time (or the currently processed time step  $n$ ).

### 5.3.4 Examples

Which of the following systems are **linear** and/or **time-invariant** and/or **causal**?

$$\textcircled{1} \quad y[n] = Ax[n] + Bx[n-2] \quad l, ti, c$$

$$\textcircled{4} \quad y[n] = -2x[-n] \quad l, c$$

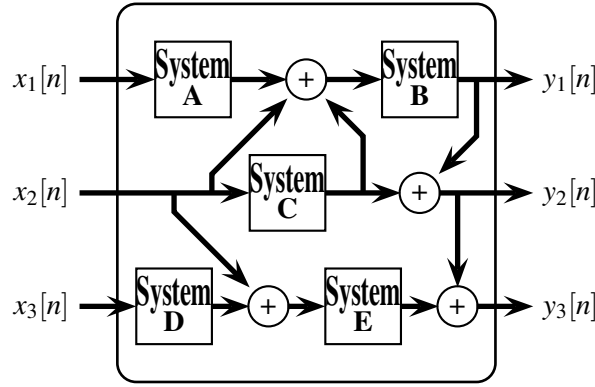
$$\textcircled{2} \quad y[n] = x[2n] \quad l$$

$$\textcircled{5} \quad y[n] = Ax[n-3] + C \quad ti, c$$

$$\textcircled{3} \quad y[n] = x^2[n] \quad ti, c$$

$$\textcircled{6} \quad y[n] = x[2n+1] \quad l$$



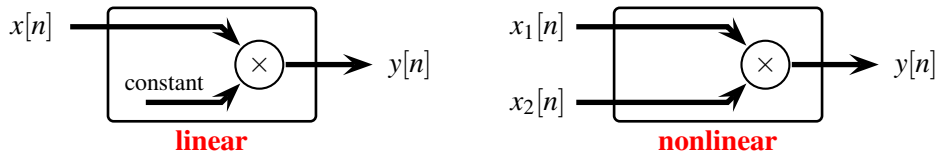


**Fig. 25:** A linear MIMO system composed of linear SISO systems and adders

#### 5.4 Linearity of MIMO and MISO systems

Any MIMO system will be linear if it is composed of linear systems and signal additions, like in the example in Fig. 25.

However, multiplication is not always linear ...



#### 5.5 Decompositions

An important consequence of the linearity of systems is that there exist algorithms for different ways of decomposing the input signal. The spectral analysis is based on this, so one can say the concept of decomposition is really fundamental. The simplest decompositions are

- Pulse decomposition

$$x[n] = x_0[n] + x_1[n] + x_2[n] + x_3[n] + \dots + x_{11}[n] + \dots$$

- Step decompositions

$$x[n] = x_0[n] + x_1[n] + x_2[n] + x_3[n] + \dots + x_{11}[n] + \dots$$

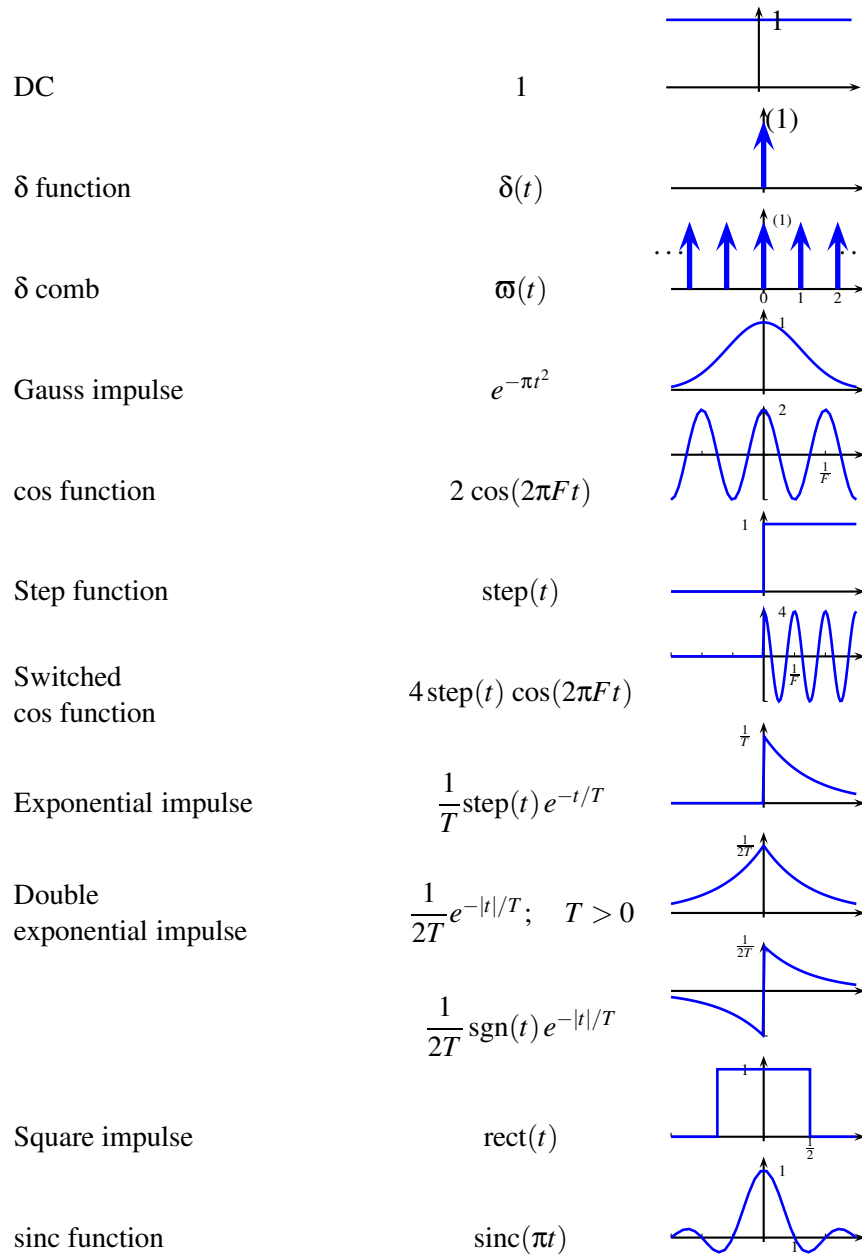
- Fourier decomposition

$$x[n] = x_{c0}[n] + x_{c1}[n] + x_{c2}[n] + x_{c3}[n] + \dots + x_{c6}[n] + x_{c7}[n] + \dots$$

$$N = 16 \quad x_{s0}[n] + x_{s1}[n] + x_{s2}[n] + x_{s3}[n] + \dots + x_{s6}[n] + x_{s7}[n] + \dots$$

- and many others.

Later, we shall make extensive use of special decompositions and also convolutions (which is the opposite process). Their applications are in the Fourier transformation, the Laplace and  $z$ -transformation, wavelets and filters.

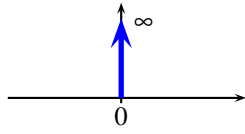
**Fig. 26:** Common waveforms

## 6 Special functions

In this very short section I wish to introduce you to some very common functions and signal forms shown in Fig. 26. Special focus will be put on the  $\delta$ -function (or better: the  $\delta$ -distribution; but in practice the difference does not play a big role). Other common waveforms are shown in the figure.

### 6.1 The $\delta$ -function

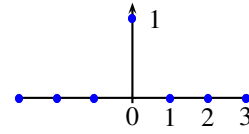
The  $\delta$ -function can be defined for continuous and for discrete signals:

**continuous:**

$$\delta(x) := \begin{cases} 0 & x \neq 0 \\ \infty & x = 0 \end{cases}$$

$$\int_{-\infty}^{\infty} \delta(x) dx = 1$$

naive definition...

**discrete:**

$$\delta[k] := \begin{cases} 0 & k \neq 0 \\ 1 & k = 0 \end{cases}$$

$$\sum_{-\infty}^{\infty} \delta[i] = 1$$

well defined!

The continuous  $\delta$ -function is not well defined that way. This is because its nature is that of a **distribution**. One important and required property of the  $\delta$ -function cannot be seen this way: it is normalized (like the Gaussian distribution) so that

$$\int_{-\infty}^{\infty} \delta(x) dx = 1 .$$

The definition above can be improved if you look at the  $\delta$ -function as the limit of a series of functions. Some popular definitions include

sinc functions:

$$\delta(x) = \lim_{\kappa \rightarrow \infty} \frac{\sin(\kappa x)}{\pi x}$$

Gauss functions:

$$\delta(x) = \lim_{\varepsilon \rightarrow 0} \frac{1}{\sqrt{\pi \varepsilon}} e^{-\frac{x^2}{\varepsilon}}$$

Lorentz functions:

$$\delta(x) = \frac{1}{\pi} \lim_{\varepsilon \rightarrow 0} \frac{\varepsilon}{x^2 + \varepsilon^2}$$

rectangles:

$$\delta(x) = \lim_{\varepsilon \rightarrow 0} \frac{1}{2\varepsilon} r_{\varepsilon}(x) \quad ; \quad r_{\varepsilon}(x) := \begin{cases} 0 & |x| \geq \varepsilon \\ 1 & |x| < \varepsilon \end{cases}$$

Also a complex (Fresnel) definition is possible:

$$\delta(z) = \lim_{\alpha \rightarrow \infty} \sqrt{\frac{\alpha}{i\pi}} e^{i\alpha z^2} .$$

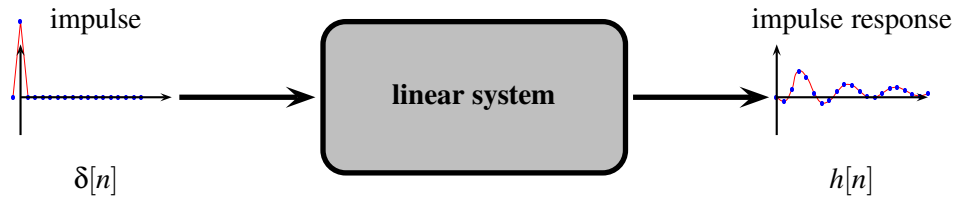
More important than the correct definition are the calculation rules of the  $\delta$ -function, which can be applied independently of its definition, whether you use  $\infty$  or the limits of series. The most important ones are given here:

Continuous convolution rule:

$$\int_{-\infty}^{\infty} f(x) \delta(x - x_0) dx = f(x_0)$$

Discrete convolution rule:

$$\sum_{i=-\infty}^{\infty} f[i] \delta[i - n] = f[n]$$



**Fig. 27:** The concept of impulse response

Fourier transform:

$$\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \delta(t) e^{-i\omega t} dt = \frac{1}{\sqrt{2\pi}}$$

Laplace transform:

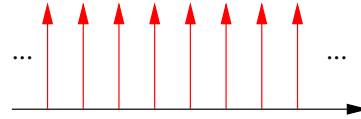
$$\int_0^{\infty} \delta(t-a) e^{-st} dt = e^{-as}$$

Scaling rule:

$$\delta(\alpha x) = \frac{\delta(x)}{|\alpha|}$$

Another popular pseudo function is the so-called *Dirac comb*, which is a combination of an infinite number of equally shifted  $\delta$ -functions:

$$C(x) = \sum_{k \in \mathbb{Z}} \delta(x-k).$$



## 7 Convolution

As already mentioned before, decomposition and convolution are the fundamental operations of linear systems. Here we are going to look more closely at the concept of convolution because the technique is the basis of all digital filters. The specialized digital signal processors always have special and ready-made instructions built in to support this operation.

### 7.1 The impulse response

The *impulse response* of a linear system is its response to a  $\delta$ -pulse on its input. For digital systems, the discrete  $\delta$  pulse, which is a unit pulse here, is applied and a sufficient number of samples  $h[n]$  of the output of the system are recorded (see Fig. 27).

This is like ringing a bell with a hammer. The hammer produces a  $\delta$  like excitation and after that the bell rings for a while; this is its impulse response. The way in which it rings is very characteristic for that bell; it contains, for example, all its eigenfrequencies, each of which decays with some characteristic time constant. What you cannot hear are the phases of the spectrum. The impulse response  $h[n]$  is the fingerprint of the system. If two linear systems have the same impulse response, then they are identical. This means that all possible information about the system can be found in its impulse response. One can say the impulse response  $h[n]$  is the system. Now, let us look at it from a mathematical point of view:

For an arbitrary input signal written in the form

$$x[n] := \sum_{i=0}^{N-1} x_n \delta[n-i]$$

we can now immediately write down the output of the system if we know its impulse response:

$$y[n] = \sum_{i=0}^{N-1} x_n h[n-i] .$$

This arises because the system is linear and so the sum stays a sum and the product with a scalar ( $x_n$ ) transforms to a product with a scalar. Only the response to the  $\delta$ -function needs to be known, but this is just the impulse response! Try to really understand this fundamental fact, recapitulate the linearity criteria if necessary and make it clear to yourself what  $x_n \delta[n-i]$  means. The features you should remember are

- $h[n]$  has all information to process the output of the system for any input signal !
- $h[n]$  is called **filter kernel** of the system (and can be measured by impulse response).
- The system is ‘**causal**’ if  $h[i] = 0 \quad \forall i < 0$ .
- The output for any input signal  $x[n]$  is

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

where  $*$  is the **convolution operator**. The mathematical definition follows.

## 7.2 Convolution

Given two functions  $f, g : D \rightarrow \mathbb{C}$ , where  $D \subseteq \mathbb{R}$ , the **convolution** of  $f$  with  $g$ , written  $f * g$  and defines as the integral of the product of  $f$  with a mirrored and shifted version of  $g$ :

$$(f * g)(t) := \int_D f(\tau) g(t - \tau) d\tau .$$

The domain  $D$  can be extended either by periodic assumption or by zero, so that  $g(t - \tau)$  is always defined.

Given  $f, g : D \rightarrow \mathbb{C}$ , where  $D \subseteq \mathbb{Z}$ , the **discrete convolution** can be defined in a similar way by the sum:

$$(f * g)[n] := \sum_{k \in D} f[k] g[n - k]$$

Two examples of discrete convolutions are shown in Fig. 28 and Fig. 29. As you can see, it is very simple to realize digital filters with this technique by choosing the appropriate filter kernels. You may ask where the filter kernels come from. Well, this is the topic of filter design where a practical formalism can be used which we briefly discuss in the section about the  $z$ -transform.

## 7.3 Calculating with convolution

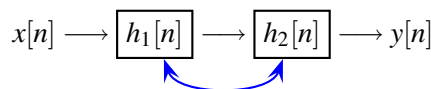
### 7.3.1 Commutative property

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

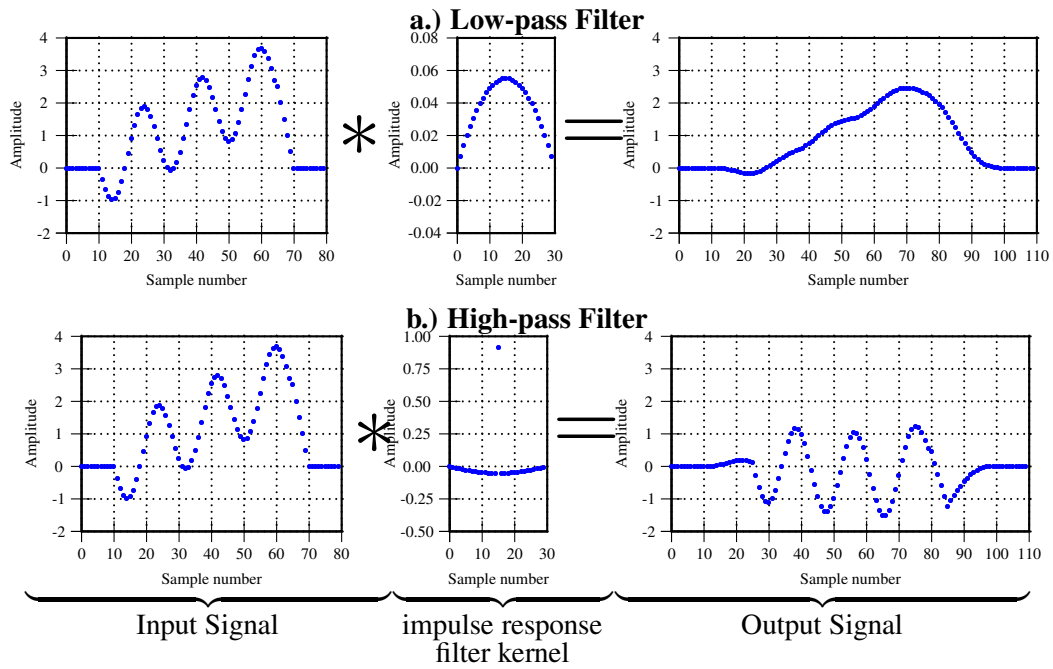
The commutative property of convolution tells you that the result will be the same if you exchange the input signal with the filter kernel (whatever sense this makes). It makes more sense if you look at the

### 7.3.2 Associative property

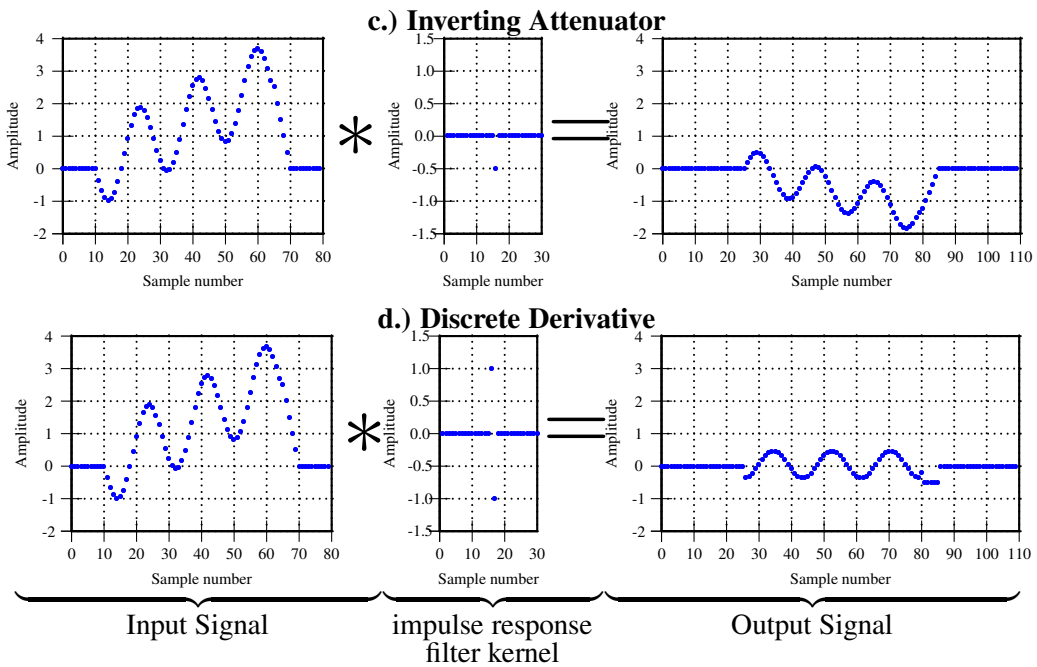
$$(a * b) * c = a * (b * c)$$



This feature allows you to rearrange systems which are in series in different and arbitrary orders. It does not matter if you first pass a differentiator and then a low-pass or vice versa. The result will be the same.



**Fig. 28:** Realization of a low-pass and a high-pass filter with convolution. The input signal is convoluted with an appropriate filter kernel and the result is the output signal.



**Fig. 29:** Realization of a digital attenuator and calculating the derivative of an input signal

### 7.3.3 Basic kernels

Identity:

$$x[n] * \delta[n] = x[n]$$

Scaling:

$$x[n] * k \cdot \delta[n] = kx[n]$$

Shift:

$$x[n] * \delta[n - a] = x[n - a]$$

Integrator:

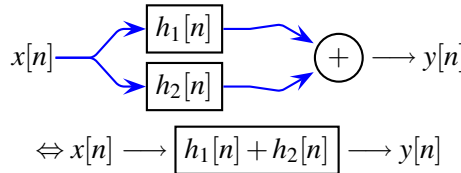
$$h[n] = \begin{cases} 1 & n \geq 0 \\ 0 & n < 0 \end{cases}$$

Differentiator:

$$h[n] = \delta[n] - \delta[n - 1]$$

### 7.3.4 Distributive property

$$a * b + a * c = a * (b + c)$$



From the distributive property, it follows that parallel systems whose output is added can be treated in the way that you add the systems (add its impulse response and then treat it as one system).

### 7.3.5 Exercise

Given  $x[n]$  a ‘pulse-like’ signal ( $x[n] = 0$  for small and large  $n$ ), what is the result of

$$x[n] * x[n] * x[n] * \cdots * x[n] = ?$$

Well, remember the **central limit theorem**. The result will be approximately Gaussian with  $\sigma = \sigma_x \cdot \sqrt{m}$  and shifted in time, due to the latency which comes from the fact that the pulse  $x[n]$  lies in the positive range of  $n$ .

## 7.4 Correlation functions

One useful application of the convolution, which is essentially the convolution of one signal with itself, is the *correlation function*. The *cross-correlation* is a measure of similarity of two signals, commonly used to find features in an unknown signal by comparing it to a known one. It is a function of the relative time between the signals and has applications in pattern recognition. A high value of the cross-correlation function for a given lag of time indicates a high similarity of the two signals at this time lag. In an *auto-correlation*, which is the cross-correlation of a signal with itself, there will always be at least one peak at a lag of zero.

### 7.4.1 Cross-correlation

Given two functions  $f, g : D \rightarrow \mathbb{C}$ , where  $D \subseteq \mathbb{R}$ , the **cross correlation** of  $f$  with  $g$ :

$$(f \circ g)(t) := K \int_D f(\tau) g(t + \tau) d\tau.$$

The cross-correlation is similar in nature to the convolution of two functions. Whereas convolution involves reversing a signal, then shifting it and multiplying it by another signal, correlation only involves shifting it and multiplying (no reversing).

### 7.4.2 Auto-correlation

$$A_g(t) := g \circ g = K \int_D g(\tau)g(t+\tau)d\tau$$

. The auto-correlation can be used to detect a known waveform in a noisy background, e.g., echoes of a signal. This can also be used to detect periodicities in a very noisy signal. The auto-correlation function of a periodic signal is also a periodic signal with the same period (but the phase information is lost). Because white noise at one time is completely independent of white noise at a different time, the auto-correlation function of white noise is a  $\delta$  pulse at zero. So, for the analysis of periodicities, you just look at the auto-correlation function for bigger time lags and ignore the values around zero, because this area contains only the information about the strength of the noise contribution.

### 7.4.3 Discrete correlation

For discrete systems and signals we use the discrete version of the correlation integral: Given  $f, g : D \rightarrow \mathbb{C}$ , where  $D \subseteq \mathbb{Z}$ , the **discrete correlation**:

$$(f \circ g)[n] := \alpha \sum_{k \in D} f[k]g[n+k],$$

which is identical to

$$f[n] \circ g[n] = f[n] * g[-n].$$

## 8 Fourier transform

The *Fourier transform* is a linear operator that maps complex functions to other complex functions. It decomposes a function into a continuous spectrum of its frequency components, and the inverse transform synthesizes a function from its spectrum of frequency components. The Fourier transform of a signal  $x(t)$  can be thought of as that signal in the *frequency domain*  $X(\omega)$ .

<b>time domain</b>	$\longrightarrow$	<b>frequency domain</b>
$x(t)$		$X(\omega)$

Information is often hidden in the **spectrum** of a signal. Figure 30 shows common waveforms and its Fourier transforms. Also looking at the *transfer function* of a system shows its frequency response. The Fourier transform is, therefore, a commonly used tool. As you will see later, a discretized version of the Fourier transform exists which is the *Discrete Fourier Transform*.

Given  $f : D \rightarrow \mathbb{C}$ , where  $D \subseteq \mathbb{R}$ , the **Fourier transformation** of  $f$  is:

$$F(\omega) := \int_D f(t) e^{-i\omega t} dt$$

and the **inverse Fourier transformation**

$$f(t) := \int_{-\infty}^{\infty} F(\omega) e^{+i\omega t} d\omega.$$

The Fourier transform can also be expressed using the convolution

$$F(\omega) = [f(t) * e^{i\omega t}]_{t=0}.$$



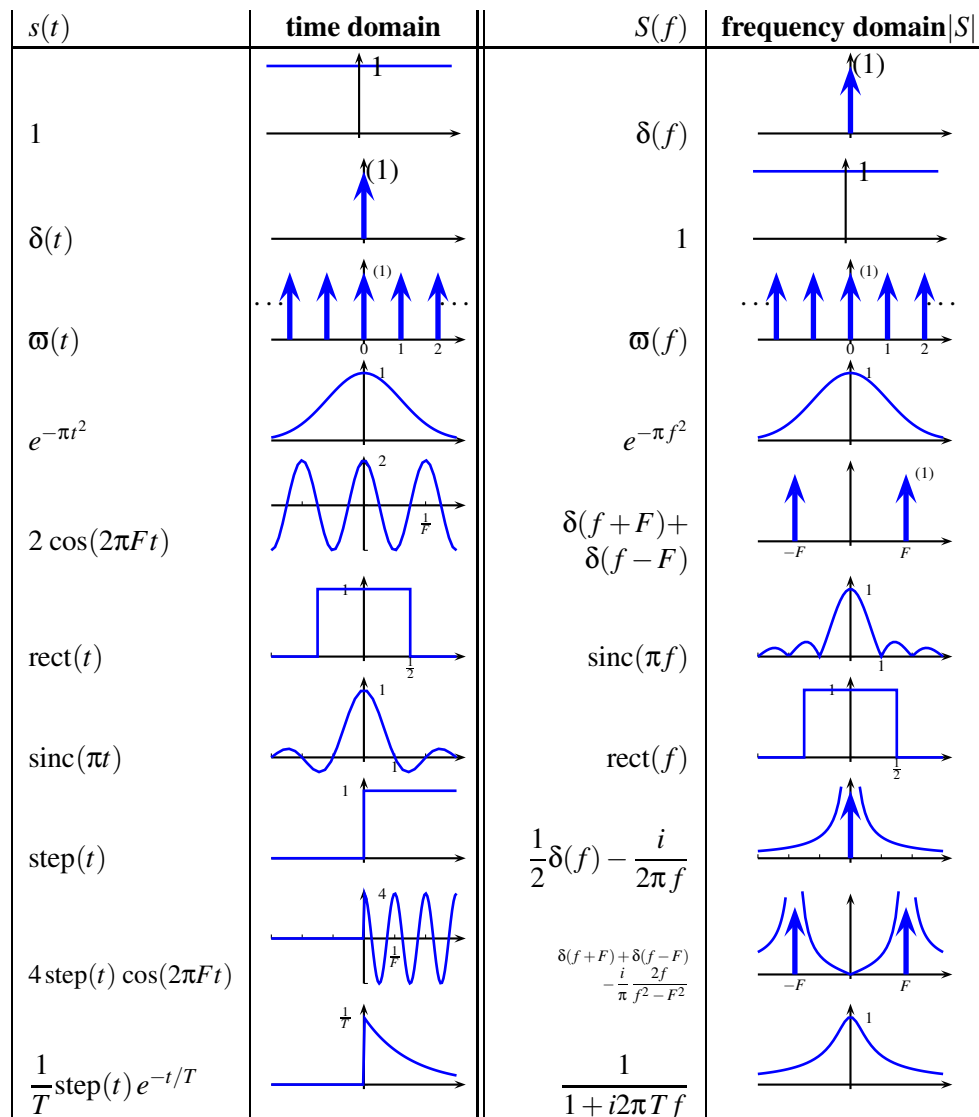


Fig. 30: Fourier transformation examples of common waveforms

### 8.1 Calculation with Fourier transforms

For a *real* input, the transformation produces a *complex* spectrum which is symmetrical:

$$X(\omega) = X^*(-\omega)$$

complex conjugate

The Fourier transform of a cos-like signal will be purely real, and the Fourier transform of a sin-like signal will be purely imaginary. If you apply the Fourier transform twice, you get the time-reversed input signal  $x(t) \xrightarrow{\text{FT}} X(\omega) \xrightarrow{\text{FT}} x(-t)$ . In the following, most important calculation rules are summarized:

**Symmetry:**

$$\text{FT}^2\{x(t)\} = x(-t)$$

**Linearity:**

$$\text{FT}\{c_1 x_1(t) + c_2 x_2(t)\} = c_1 X_1(\omega) + c_2 X_2(\omega)$$

**Scaling:**

$$\mathbf{FT}\{x(\lambda t)\} = \frac{1}{|\lambda|} X\left(\frac{\omega}{\lambda}\right)$$

**Convolution:**

$$\mathbf{FT}\{x_1(t) * x_2(t)\} = X_1(\omega) \cdot X_2(\omega) \quad ; \quad \mathbf{FT}\{x_1(t) \cdot x_2(t)\} = X_1(\omega) * X_2(\omega) \quad (13)$$

**Integration:**

$$\mathbf{FT}\left\{\int_{-\infty}^t h(\tau) d\tau\right\} = \frac{1}{i\omega} X(\omega) + \frac{1}{4\pi} \overbrace{\left[\int_{-\infty}^{\infty} h(\tau) d\tau\right]}^{\text{DC offset}} \delta(\omega) \quad (14)$$

**Time-shift:**

$$\mathbf{FT}\{x(t + t_0)\} = e^{i\omega t_0} X(\omega)$$

## 8.2 The transfer function

Consider the following signal path consisting of two linear systems with impulse responses  $h_1$  and  $h_2$ :

$$x(t) \xrightarrow{\quad} \boxed{h_1} \xrightarrow{c(t)} \boxed{h_2} \xrightarrow{y(t)} .$$

The output signal will be the convolution of the input signal with each of the impulse response vectors

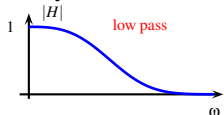
$$y(t) = x(t) * h_1 * h_2 . \quad (15)$$

If we now look at the spectrum of the output signal  $Y(\omega)$  by Fourier transforming Eq. (15), we get

$$\Rightarrow Y(\omega) = X(\omega) \cdot H_1(\omega) \cdot H_2(\omega) .$$

Transfer Functions

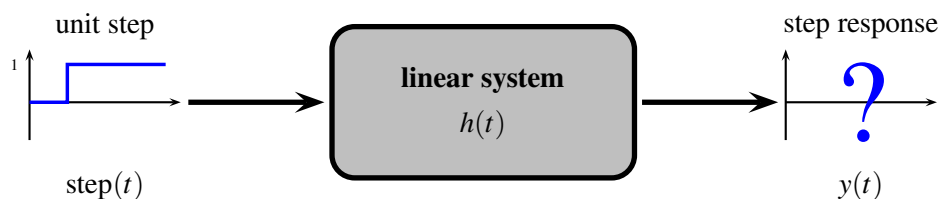
Here we made use of the calculation rule (13), that the Fourier transform of a convolution of two signals is the product of the Fourier transforms of each signal. In this way, we are going to call the Fourier transforms of the impulse responses *transfer functions*. The transfer function also completely describes a (linear) system; it contains as much information as the impulse response (or kernel) of the system. It is a very handy concept because it describes how the spectrum of a signal is modified by a system. The transfer function is a complex function, so it not only gives the amplitude relation  $|H(\omega)|$  of a system's output relative to its input, but also the phase relations. The absolute value of the transfer function can tell you immediately what kind of filter characteristic the system has. For example, a function like  $|H| =$



behaves like a low-pass.

It is now also very easy to tell what the output spectrum of a multiplier will be:

$$\begin{array}{c}
 x_2 \\
 \downarrow \\
 x_1 \longrightarrow \bigcirc \times \longrightarrow y \\
 y(t) = x_1(t) \cdot x_2(t) \\
 \Rightarrow Y(\omega) = X_1(\omega) * X_2(\omega) .
 \end{array}$$



**Fig. 31:** What is the step response of a dynamic system?

It is the convolution of the two input spectra. In the special case, where one input signal consists only of a single frequency peak, the spectrum of the second input will be moved to this frequency. So a multiplier (sometimes also called a *mixer*) can be used to shift spectra. Exercise: what does the resulting spectrum look like if you have a single frequency on each of the two inputs? Which frequency components will be present? Do not forget the negative frequencies!

### 8.3 Step response

Earlier in this lecture we defined the impulse response of a system. This was a way to extract the essential information of that system. But this is not the only way to do it. An equivalent method uses the *step response* instead. The step response is the response of a system to a unity step. Unity step means, the input changes instantly from 0 to unity value (1). The system will react on this excitation showing its step response (see Fig. 31). It also contains all the information of the system, and can also be used as a fingerprint, exactly the same as the impulse response. There are rather practical reasons why one might prefer to look at the step response: Knowing the step response of a system gives information on the dynamic stability of such a system and on its ability to reach a stationary state, starting from another state.

Showing the equivalence to the impulse response is now an easy task with the convolution rule (13) and the integration rule (14) of the Fourier calculus:

$$\begin{aligned}
 y(t) &= \text{step}(t) * h(t) = ? \\
 Y(\omega) &= \underbrace{\left( \frac{1}{i\omega} + \frac{1}{4\pi} \delta(\omega) \right)}_{\text{from table}} \cdot \underbrace{H(\omega)}_{\text{low-pass}} = \underbrace{\frac{H(\omega)}{i\omega}}_{\text{low-pass}} + \underbrace{\frac{1}{4\pi} \delta(\omega) H(\omega)}_{\text{DC offset}} \\
 y(t) &= \int_{-\infty}^t h(\tau) d\tau
 \end{aligned}$$

The step response is the integral over time of the impulse response.

#### 8.3.1 Correlation revisited

Coming back to correlation, what does the spectrum of the correlation function tell us?

**auto-correlation:**

$$s(t) * s(-t) \xrightarrow{\text{FT}} S(\omega) \cdot S^*(\omega) = |S(\omega)|^2$$

**Energy spectrum**

The spectrum of the auto-correlation function of a signal  $s(t)$  is identical to its *energy spectrum*. The information about phase (or time/shift/location) is lost, so one can say that the auto-correlation function is *time invariant*.

**cross-correlation:**

$$s(t) * g(-t) \xleftrightarrow{\text{FT}} S(\omega) \cdot G^*(\omega)$$

Here, the *real* part of the spectrum of the cross-correlation of two signals tells us about parts which are *similar*, and the *imaginary* part of the spectrum tells us about parts which are *not correlated*.

**9 Laplace transform**

You have seen how handy the Fourier transformation can be in describing (linear) systems with  $h(t)$ . But a Fourier transform is not always defined:

- for example,  $x(t) = e^{-t}$  has an infinite frequency spectrum  $X(\omega) > 0$  everywhere;
- for example,  $x(t) = e^t$  is unbounded and can not even be represented;
- for example,  $\text{step}(t) \longrightarrow$  infinite frequency spectrum;
- a lot of  $\delta$ -functions appear, etc.

To handle this, we decompose these functions, not only into a set of ordinary cosine and sine functions, but we also use exponential functions and exponentially damped or growing sine and cosine functions. It is not so complicated to do. We just substitute the frequency term  $i\omega$  by a general complex number  $p$ . You can look at this as introducing a complex frequency

$$p = \sigma + i\omega,$$

where  $\omega$  is the known real frequency and  $\sigma$  is a (also real) damping term. The functions to deal with now become

$$f(t) = e^{-pt} = e^{-\sigma t} \cdot e^{-i\omega t}.$$

Instead of the Fourier transform we now introduce a more general transform, called the *Laplace transform*: Given  $s : \mathbb{R}^+ \rightarrow \mathbb{R}$ , the **Laplace transformation** of  $s$  is:

$$s(t) \xrightarrow{\text{L}} S(p) := \int_0^{\infty} s(t) e^{-pt} dt$$

and the **inverse transformation**

$$S(p) \xrightarrow{\text{L}^{-1}} s(t) := \begin{cases} \frac{1}{2\pi i} \int_{\sigma-i\infty}^{\sigma+i\infty} S(p) e^{pt} dp & \text{for } t \geq 0 \\ 0 & \text{for } t < 0. \end{cases}$$

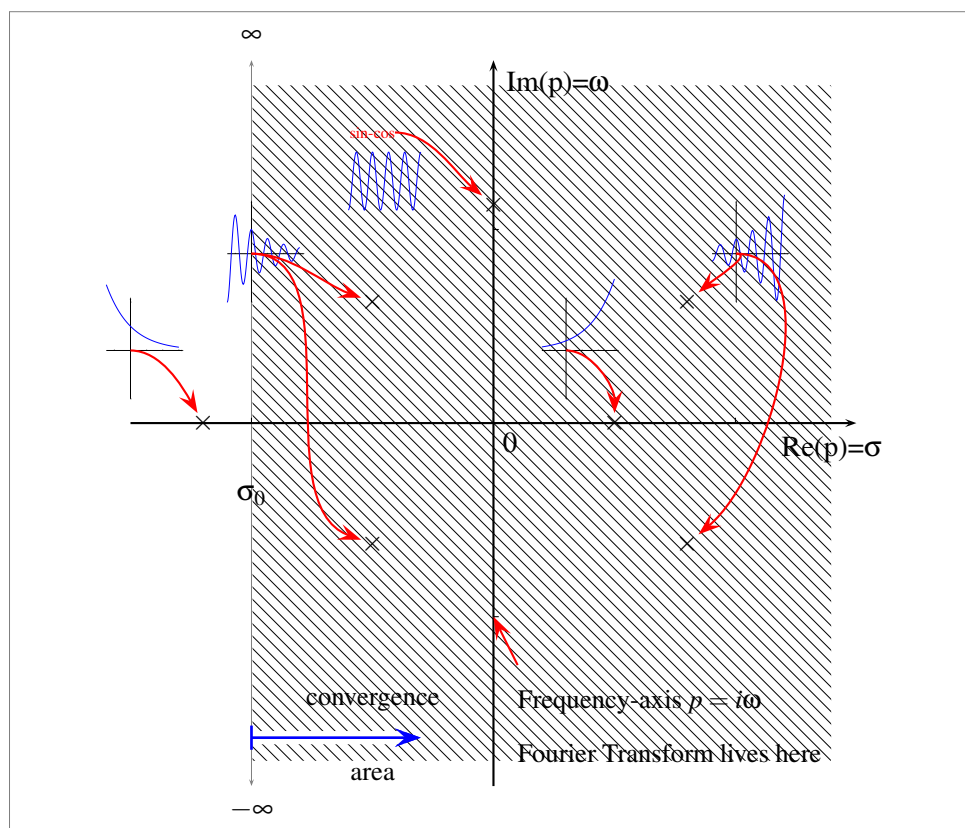
Remember:

- $s(t)$  is real and  $s(t) = 0$  for  $t < 0$ .
- $S(p)$  is a complex function:  $\mathbb{C} \xrightarrow{S} \mathbb{C}$  (in contrast to the Fourier transformation, where  $\mathbb{R} \xrightarrow{F} \mathbb{C}$ ).

We shall come back to the inverse Laplace transform later in Section 9.5.

**9.1 Region of convergence**

As we mentioned before, the Fourier transformation was not always defined, even a simple sine wave produced a  $\delta$ -peak in its spectrum which causes a problem. So does the Laplace transform always exist? The Answer is: no, but there is an area  $\text{Re}(p) > \sigma_0$ ;  $\sigma_0 \in \mathbb{R}$  where it exists (*region of convergence*). This means  $\exists M < \infty : |s(t)| \leq M e^{\sigma_0 t}$ . If  $\sigma_0 < 0$ , the Fourier transform also exists.

Fig. 32: The  $p$  plane

To see what that means in practice, it is useful to visualize the functions and numbers we deal with in a diagram. This is possible if we look at the complex plane shown in Fig. 32, called the  $p$  plane. Different points on the plane correspond to different types of base functions as shown. The ordinary sine and cosine functions are also present and live on the imaginary axis. If the imaginary axis lies inside of the convergence area, then also the Fourier transform exists, and you will get it if you go along the imaginary axis. In addition, you also get spectral values for other regions of the plane.

What is this good for? Well, it is good for solving differential equations, especially those for analog filters. Let us see how this works. Formally, we do exactly the same as we did with the Fourier transform:

$$\begin{array}{c}
 x \longrightarrow \boxed{\text{System}} \longrightarrow y \\
 \downarrow \mathcal{L} \quad \quad \quad \downarrow \mathcal{L} \quad \quad \quad \uparrow \mathcal{L}^{-1} \\
 X \quad \cdot \quad H \quad = \quad Y
 \end{array}$$

We will have the concept of transfer functions slightly extended onto the whole  $p$  plane, but the concept stays the same. So we may get answers to questions like: What filter do I need for getting a specific output  $y(t)$ ? Or we can compose the system out of subsystems by multiplying the transfer functions of each of them, etc. This implies that we will have nearly the same or similar calculation rules as for the Fourier transformation, and indeed that is exactly true.

**Calculating with the Laplace transform**

As before, we have similar calculation rules to the ones for the Fourier transform. In addition, the integration and differentiation operations also can be transformed.

$$x(t) \xrightarrow{\mathbf{L}} X(p) \xrightarrow{\mathbf{L}^{-1}} x(t)$$

**Linearity:**

$$\mathbf{L}\{c_1 x_1(t) + c_2 x_2(t)\} = c_1 X_1(p) + c_2 X_2(p)$$

**Scaling:**

$$\mathbf{L}\{x(\lambda t)\} = \frac{1}{|\lambda|} X\left(\frac{p}{\lambda}\right); \quad \lambda > 0$$

**Time-shift:**

$$\mathbf{L}\{x(t - t_0)\} = e^{-pt_0} S(p) \quad ; \quad \mathbf{L}^{-1}\{X(p + p_0)\} = e^{-p_0 t} s(t); \quad t_0 > 0$$

**Convolution:**

$$\mathbf{L}\{x_1(t) * x_2(t)\} = X_1(p) \cdot X_2(p) \quad ; \quad \mathbf{L}^{-1}\{X_1(p) * X_2(p)\} = x_1(t) \cdot x_2(t)$$

**Integration:**

$$\begin{aligned} \mathbf{L}\left\{\int_0^t s(\tau) d\tau\right\} &= \frac{S(p)}{p} \\ \mathbf{L}^{-1}\left\{\int_p^\infty S(p') dp'\right\} &= \frac{s(t)}{t} \end{aligned}$$

**Differentiation:**

$$\mathbf{L}\left\{\frac{d^n}{dt^n} s(t)\right\} = p^n S(p) \quad \text{if } \frac{d^k s}{dt^k} \Big|_{t=0} = 0 \quad \forall k < n$$

**Laplace transformation examples**

$$\mathbf{L}\{\text{step}(t)\} = \frac{1}{p} \quad ; \quad \sigma > 0$$

$$\mathbf{L}\{\delta(t)\} = \mathbf{L}\left\{\frac{d}{dt} \text{step}(t)\right\} = \frac{p}{p} = 1$$

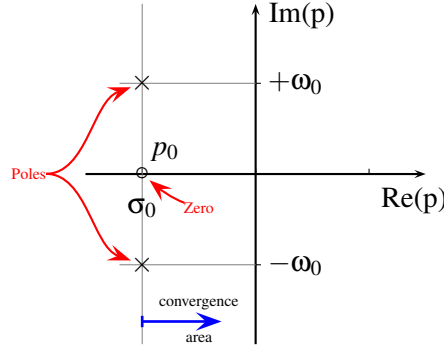
$$\mathbf{L}\{\text{step}(t - t_0)\} = \frac{e^{-t_0 p}}{p} \quad ; \quad t_0, \sigma > 0$$

$$\mathbf{L}\{e^{p_0 t} \text{step}(t - t_0)\} = \frac{1}{p - p_0} \quad ; \quad \sigma > \text{Re}(p_0)$$

$$\mathbf{L}\{\cos(\omega_0 t) \text{step}(t)\} = \frac{1}{2} \left( \frac{1}{p - i\omega_0} + \frac{1}{p + i\omega_0} \right) = \frac{p}{p^2 + \omega_0^2}$$

$$\mathbf{L}\{\sin(\omega_0 t) \text{step}(t)\} = \frac{1}{2i} \left( \frac{1}{p - i\omega_0} - \frac{1}{p + i\omega_0} \right) = \frac{\omega_0}{p^2 + \omega_0^2} \quad ; \quad \sigma > 0$$

$$\mathbf{L}\{t^n \text{step}(t)\} = \frac{n!}{p^{n+1}} \quad ; \quad \sigma > 0, n \in \mathbb{N}_0$$



**Fig. 33:** Two poles and one zero in the  $p$  plane for the complex spectrum of a damped oscillation

## 9.2 Poles and zeros

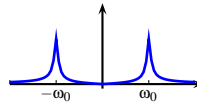
Now, we are interested in regions where the Laplace transformation does not exist. Such functions were also of interest when looking at the Fourier transformation. Remember: the spectrum of a sine function has two poles. These were expressed by  $\delta$ -functions, but in practice this means that there are unphysical infinities and spectral peaks with zero bandwidth. Not nice. Because the Fourier spectrum is included in the Laplace spectrum, we also expect such poles here. Clearly they cannot be inside the convergence region, but anyway they are of interest. We shall soon see why. Other areas of interest are where the spectral components are exactly zero. That means, signals of these frequencies are completely rejected by the system. Zeros are of interest because they have the potential to cancel out poles. This is of major interest because system instabilities (which usually means that they have poles and the output signal will grow ad infinitum, mostly also oscillating) can be cured if another system with an appropriate pole is put in series (or in parallel) to the unstable one. But first, let us have a closer look at poles in the  $p$  plane. Remember that the real spectrum of the signals live only on the imaginary axis. We shall see how a pole (or a zero) near this axis will influence the spectrum.

First we are going to analyse the Laplace transform of a causal (all values for  $t < 0$  are zero) damped oscillation:

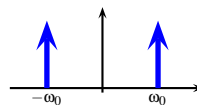
$$\mathbf{L}\{\cos(\omega_0 t) \exp(p_0 t) \text{step}(t)\} = \frac{p - p_0}{(p - p_0)^2 + \omega_0^2}, \quad (16)$$

where  $\omega_0$  is a real number and  $p_0$  should be real but  $p_0 < 0$ . After having calculated the Laplace transform of this function (using the calculation rules given above), one can read from Eq. (16) that there is one zero at  $p_0$  and two poles  $p_0 \pm j\omega_0$  (see Fig. 33). Since no pole lies to the right of  $p_0$ , the region of convergence is  $\sigma - \text{Re}(p_0) > 0$  ( $\sigma_0 = \text{Re}(p_0)$ ).

Because the imaginary axis is inside the region of convergence, a Fourier transform also exists. It looks like



and you can see the resonance. If the poles were on the  $j$ -axis, a  $\delta$  function would be necessary for expressing the spectrum:



### 9.3 Laplace transform of linear systems

To be more general, we now have a look at arbitrary linear systems. The general differential equation for such analog filters is

Coefficients from the filter components (resistors, capacitors, inductivities, ...)

$$\begin{aligned}
 y(t) &= \sum_{k=0}^M a_k \frac{d^k}{dt^k} x(t) + \sum_{k=1}^N b_k \frac{d^k}{dt^k} y(t) \\
 Y(p) &= \sum_{k=0}^M a_k p^k \cdot X(p) + \sum_{k=1}^N b_k p^k \cdot Y(p) \\
 &= \frac{\sum_{k=0}^M a_k p^k}{1 - \sum_{k=1}^N b_k p^k} \cdot X(p) =: H(p) \cdot X(p) \quad .
 \end{aligned} \tag{17}$$

Here, the transfer function  $H(p)$  is defined for the whole complex plane using the coefficients from the differential equation. Its general form is

$$H(p) = \frac{\sum_{k=0}^M a_k p^k}{1 - \sum_{k=1}^N b_k p^k} = \frac{a_M \prod_{k=1}^M (p - p_{0k})}{-b_N \prod_{k=1}^N (p - p_{pk})} \quad .$$

Factorizing is always possible.  $p_{0k}$  are the **zeros** and  $p_{pk}$  the **poles** of the transfer function. The transfer function is fully determined by its poles and zeros (except for a complex factor  $\frac{a_M}{b_N}$ )!

### 9.4 The transfer function

Consider you know all poles and zeros of a system, you can immediately estimate without too much calculating what the frequency response of the system will be. Therefore we look at the absolute value of the transfer function (it is also possible and also very easy to calculate the phases<sup>3</sup>):

$$|H(p)| = \left| \frac{a_M}{b_N} \right| \cdot \frac{\prod_{k=1}^M |p - p_{0k}|}{\prod_{i=1}^N |p - p_{pi}|} \quad .$$

As a matter of interpretation you can think of this as

$$|H(p)| = \frac{\prod \text{distances between } z \text{ and zeros}}{\prod \text{distances between } z \text{ and poles}} \quad .$$

Figure 34 illustrates how you can read the frequency response from a small diagram. You scan from zero along the imaginary axis (which gives the real frequency  $\omega$ ) and from each point  $z$  you measure the distances between  $z$  and zeros and the distances between  $z$  and poles, multiply and divide them together and plot the result in the diagram in dependency of  $\omega$  as shown in Fig. 34. This is the way your filter design tools do it (no magic).

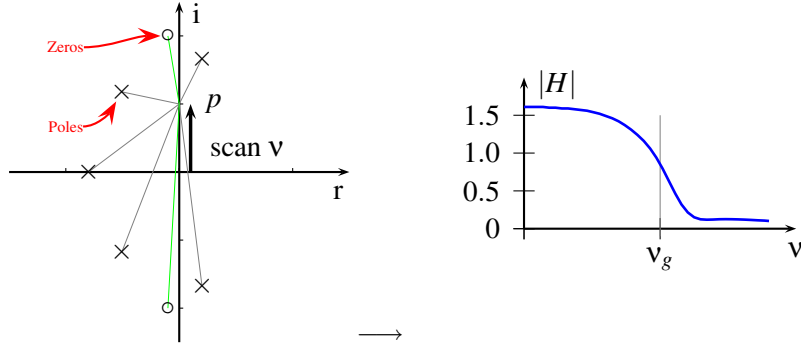
### 9.5 The inverse Laplace transform

We have already defined the *inverse Laplace transformation* as

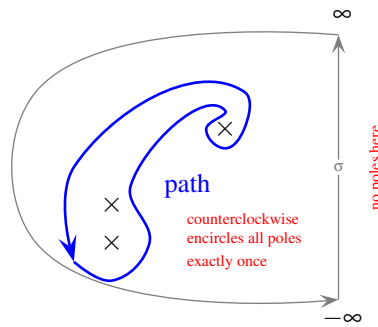
$$S(p) \xrightarrow{\mathbf{L}^{-1}} s(t) := \frac{1}{2\pi i} \int_{\sigma - i\infty}^{\sigma + i\infty} S(p) e^{pt} dp \quad \text{for } t \geq 0 \quad . \tag{18}$$

<sup>3</sup>Have a look at Section 11.1. It works the same here.





**Fig. 34:** Calculating the frequency response of a system from the poles and zeros of its transfer function



**Fig. 35:** Different integration paths around poles for the inverse Laplace transform

Now the question might be, why does the integration go from minus infinity to infinity exactly on the boundary of the convergence area? Indeed it is not necessary to do it this way. But, the integration path needs to encircle all poles (exactly once) anticlockwise. From residual theory we know that the contribution of a holomorph function on an area where there are no poles is zero for any closed integration loop. So we define

$$s(t) = \frac{1}{2\pi i} \oint_{\text{path}} S(p) e^{pt} dp$$

$$s(t) = \sum_{p_{pk}} \text{Res}_{p_{pk}} (S(p) e^{pt}) .$$

Recapitulate the definition of the *residuum* of a pole of function  $f$  at  $p_0$  of order  $k$ :

$$\text{Res}_{p_0}(f) := \frac{1}{(k-1)!} \frac{d^{k-1}}{dp^{k-1}} \left[ f(p) \cdot (p - p_0)^k \right] . \quad (19)$$

Looking at Fig. 35, we see how the definition (18) fits into that picture. The integration path  $\int_{\sigma-i\infty}^{\sigma+i\infty}$  which lies completely inside or at least at the left border of the region of convergence (no poles are on the right side) already contains the whole residue information. As you can see in Fig. 35, the integration path can be extended by an anticlockwise encirculation of all poles in the far region of the  $p$ -plane. Now, the *initial value theorem* (also a calculation rule we have not yet mentioned)

$$s(0) = \lim_{p \rightarrow \infty} pS(p) < \text{const}$$

tells us that the behaviour of  $S(p)$  for large  $|p|$  should be at least a decay of the order of  $|S(p)| < \frac{1}{|p|}$  so that for  $\lim_{p \rightarrow \infty}$  the contribution to this path of integration is zero.

### Examples

Finally, let us do two examples of inverse Laplace transforms to see how it works out:

1.  $p_0$  single pole of

$$S(p) := \frac{1}{p+a}, \quad k=1, \quad p_0 = -a.$$

$$\begin{aligned} s(t) &= \text{Res}_{-a} [S(p) e^{pt}] \\ &= \frac{1}{(1-1)!} \cdot \frac{d^0}{dp^0} \left( \frac{1}{p+a} e^{pt} (p+a)^1 \right) \Big|_{p=-a} = e^{-at}. \end{aligned}$$

2.  $p_0$  pole of third order of

$$S(p) := \frac{1}{p^3}, \quad k=3, \quad p_0 = 0.$$

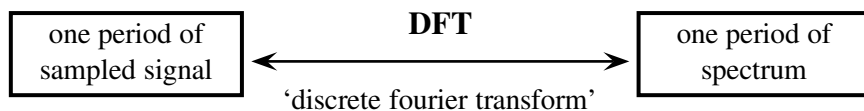
$$\begin{aligned} s(t) &= \text{Res}_0 [S(p) e^{pt}] \\ &= \frac{1}{2!} \cdot \frac{d^2}{dp^2} \left( \frac{1}{p^3} e^{pt} p^3 \right) \Big|_{p=0} = \frac{t^2}{2}. \end{aligned}$$

## 10 Discrete transforms and digital filters

In the previous section, we have mainly developed the mathematical formalism for analog signal processing and for continuous linear systems. Since the aim of this lecture is to treat digitized data and realize systems with digital signal processing, our job is now to transform the concepts in such a way that we can make use of them for the design of digital systems. Many concepts are the same or at least similar. The quantization of time and value has only a small effect, and in the limit for an infinitely high sampling rate and real numbers, it should be the same. Nevertheless, as mentioned before, we have to deal with these quantization effects, and there are fundamental differences, some of which we have already discussed. Remember: The spectrum of sampled signals is always **periodic**, because of aliasing, and the spectrum of a (continuous) periodic signal is also always periodic — this is also true for sampled signals.

### 10.1 The discrete Fourier transform

Because of this, we can define a transformation which maps between the periods in time domain and the periods in frequency domain. This can be done with the *discrete Fourier transform*. In contrast to the continuous Fourier transform, the discrete Fourier transform is always defined and maps uniquely between these domains, without ambiguities.



Given a period (of length  $N$ ) of samples  $s[n]$  ( $\in \mathbb{R}$  or  $\mathbb{C}$ ) with  $n \in [0, \dots, N[ \subset \mathbb{N}_0$ , the *discrete Fourier transform* is defined as

$$S[k] = \sum_{n=0}^{N-1} s[n] e^{-2\pi i \frac{nk}{N}},$$

where  $S[k] \in \mathbb{C}$ ,  $k \in [0, \dots, N[ \subset \mathbb{N}_0$ .

The **inverse discrete Fourier transform** is

$$s[n] = \frac{1}{N} \sum_{k=0}^{N-1} S[k] e^{2\pi i \frac{nk}{N}}.$$

Calculation rules for the DFT are exactly the same as for the continuous Fourier transforms (**linearity, symmetry, etc.**), just replace  $\omega$  with the *discrete frequency*

$$\omega_d : 2\pi \frac{k}{N} \Rightarrow S[\omega_d] = \sum_{n=0}^{N-1} s[n] e^{-i\omega_d n}$$

and then substitute  $k = \frac{\omega_d}{2\pi} \cdot N$ ,  $k \in \mathbb{N}_0$ .

But there are also two important differences, one is the

**Scaling:** ( $\lambda \in \mathbb{Z}$ )

$$\mathbf{DFT}\{x[\lambda n]\} = \frac{1}{|\lambda|} X\left(\frac{\omega_d}{\lambda}\right) = X[??]$$

which will not work, because the length of the period itself is modified. A little modification needs to be applied to the

**Time-shift:**

$$\mathbf{DFT}\{x[n + n_0]\} = e^{i\omega_d n_0} X[k].$$

And finally, with the **convolution**, one needs to pay attention, because if the result has more samples than the period, it needs to be folded back into the period.

## 10.2 The Fast Fourier Transform (FFT)

If the number of samples of the data snapshot is  $N = 2^m$ ,  $m \in \mathbb{Z}_0$ , there is a fast and efficient algorithm to compute the discrete Fourier transform (DFT) and its inverse. The details of these (there are many) algorithms are beyond the scope of this lecture and so you may refer to the literature.

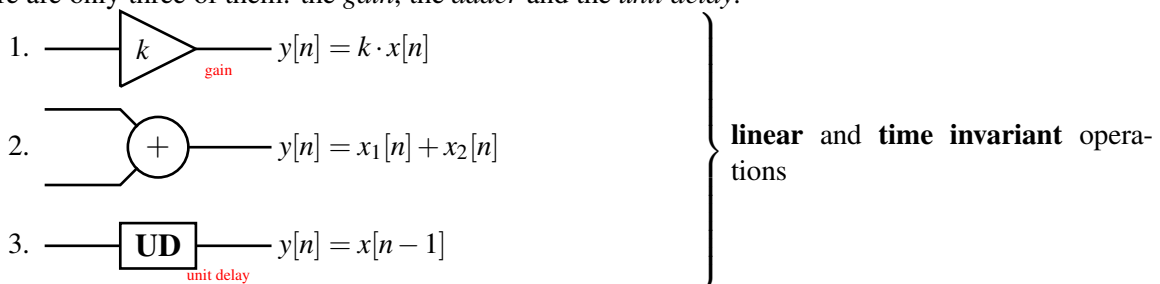
Since this algorithm can only be applied to snapshots with a number of samples that is a power of 2, there are several techniques, which match the number of samples  $N$  to  $2^m$ , mainly

1. **zero-padding**
2. **windowing**, also for estimation of FT of an aperiodic signal and time-frequency analysis.

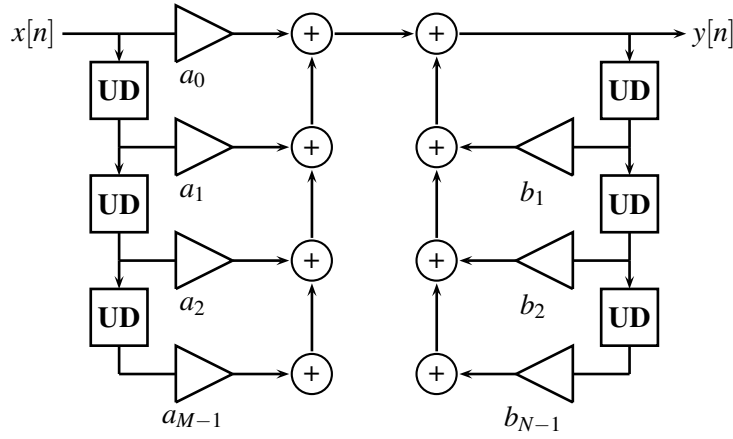
Here, you may also refer to the literature.

## 10.3 Digital filters

Let us consider the very basic systems and elementary linear operations we can think of. We find that there are only three of them: the *gain*, the *adder* and the *unit delay*.



Any combination of these operations is called a ‘**digital filter**’.



**Fig. 36:** Alternative presentation of Eq. (20)

In analogy to the differential equation for analog systems (see Eq. (17))

$$y(t) = \sum_{k=0}^N \alpha_k \frac{d^k}{dt^k} x(t) + \sum_{k=1}^M \beta_k \frac{d^k}{dt^k} y(t) ,$$

we can define a similar *equation of differences* for the digital systems which only consists of the above mentioned three operations (compare with the equivalent notation shown in Fig. 36):

$$\Rightarrow y[n] = \underbrace{\sum_{k=0}^{N-1} a_k x[n-k]}_{\text{direct}} + \underbrace{\sum_{k=1}^{M-1} b_k y[n-k]}_{\text{recursive}} . \quad (20)$$

Using the convolution operator we can also write

$$y[n] = a[M] * x[n] + b[N] * y[n] ,$$

where we have two filter kernels, one direct kernel  $a[M]$  and one recursive kernel  $b[N]$ .

### 10.3.1 Impulse response of a digital filter

Now, what does the impulse response of the general digital filter denoted in Eq. (20) look like? Remember, the digital unit impulse is defined as

$$x[n] = \delta[n] := \begin{cases} 1 & n = 0 \\ 0 & n \neq 0 \end{cases} .$$

Let us write down the response to that input:

$$\Rightarrow h[n] = y[n] = \begin{cases} 0 & n < 0 \\ a_0 & n = 0 \\ a_n + \sum_{k=1}^{\min(n,N)} b_k h[n-k] & n > 0 \end{cases} .$$

Now we can distinguish between two different cases:

1. If  $b_k \equiv 0 \longrightarrow h[n] = a_n, \quad n \geq 0$  we talk about a **Finite Impulse Response filter (FIR)** and
2. if at least one  $b_{k_0} \neq 0$ , then we call it **Infinite Impulse Response filter (IIR)**.

It is clear why it is named in this way: for the FIR filter the impulse response has only a finite number of non-zero values, which means that there is a  $n_f$  where  $h[i] = 0 \quad \forall i > n_f$ . In contrast to this, the impulse response will (in general) be of infinite length, although only a finite set of the coefficients ( $a_k, b_k$ ) generate it.

### 10.3.2 Order of a filter

Besides the class of the digital filter (FIR or IIR), another important characteristic parameter is the *order* of the filter. It is defined as follows:

$$\exists(N, M) : (a_n = 0 \quad \forall n > N) \wedge (b_n = 0 \quad \forall n > M)$$

$$\text{Order} := \max(N, M) .$$

So the order is the minimum number of coefficients needed to implement it. The order is also a measure for the maximum **latency** (or delay) of a filter, because it counts the maximum number of unit delays needed to complete the output (refer to Fig. 36).

For an FIR filter, the order of the filter is equal to the length of the impulse response. For an IIR filter this is not the case.

### 10.3.3 Transfer function of a digital filter

With the help of the discrete Fourier transform, it is straightforward to find an expression for the general form of the *transfer function* of such a digital filter, starting with Eq. (20):

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

DFT DFT time shift time shift rule

$$Y(\omega_d) = \sum_{k=0}^N a_k X(\omega_d) e^{-i\omega_d k} + \sum_{k=1}^M b_k Y(\omega_d) e^{-i\omega_d k}$$

$$\xrightarrow{X(\omega_d) \neq 0 \quad \forall \omega_d} H(\omega_d) := \frac{Y(\omega_d)}{X(\omega_d)} = \frac{\sum_{k=0}^N a_k (e^{-i\omega_d})^k}{1 - \sum_{k=1}^M b_k (e^{-i\omega_d})^k} . \quad (21)$$

Remember that the digital frequency  $\omega_d$  is periodic with  $\omega_s$  ( $-\pi < \omega_d < \pi$ ).

Further, remember that we developed a similar formula in Section 9.3. In that case, we used the Laplace transform (as a more general expression which was extended to complex frequencies) instead of the Fourier transform. It is also possible to do (more or less) the same thing here for the digital systems. We can substitute  $z := e^{i\omega_d}$  and extend it to the complex by including a damping term  $\sigma$

$$z := e^{i\omega_d - \sigma} .$$

The resulting transform (which is a modified DFT) will be called *z transform*.

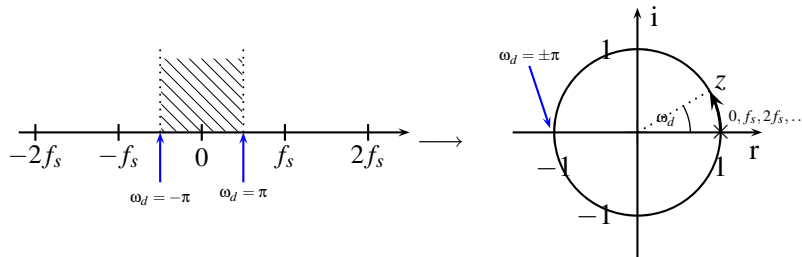
## 11 The $z$ transform

Introducing the  $z$ -transform, we develop a tool which is as powerful as the Laplace transform mentioned in Section 9, but also applicable for digital systems and digital signals. The concept is based on the periodicity of the spectra of digital signals. With a suitable transformation, all tools and methods developed for analog systems and analog signals using the Laplace transform can be adapted for use with digital ones.

Starting with the discrete transfer function, we simply do the substitution  $z := e^{i\omega_d} (= e^{2\pi i \frac{k}{N}})$  in Eq. (21):

$$H(\omega_d) \xrightarrow{\text{substitution}} H(z) = \frac{\sum_{k=0}^N a_k z^{-k}}{1 - \sum_{k=1}^M b_k z^{-k}}.$$

This substitution maps the frequency axis to the unit circle in the complex  $z$ -plane:



This concept is useful because it automatically accounts for the periodicity of  $\omega_d$ . The  $z$ -plane (or the unit circle) is a representation of one period of the digital frequency. Frequencies above the Nyquist frequency are automatically mapped to the place where its aliasing frequency would be. So there will be no aliasing from now on.

Now, we can extend this concept to the whole complex plane  $z \in \mathbb{C}$ . We therefore add a damping term to the digital frequency  $\omega_d$ :

$$\begin{aligned} \omega_d \in \mathbb{R}_{[-\pi, \pi]} &\longrightarrow \omega_{dc} \in \mathbb{C} \\ \omega_{dc} &= \omega_d + i\sigma, \\ \Rightarrow z &= e^{i\omega_{dc}} = e^{i\omega_d - \sigma}. \end{aligned}$$

As shown in Fig. 37, different points and regions for  $z$  correspond to different classes of (sampled) functions. As with the  $p$ -plane for the Laplace transform, besides discrete sine and cosine functions there are also discrete exponential functions, as well as exponentially damped and growing functions. Together, this set of functions forms a basis for the decomposition of any discrete signal. In particular, for the expression of the transfer functions of discrete systems, we can find a very handy way; it is similar to what we did with the transfer functions of analog systems, factorized in poles and zeros in the  $p$ -plane.

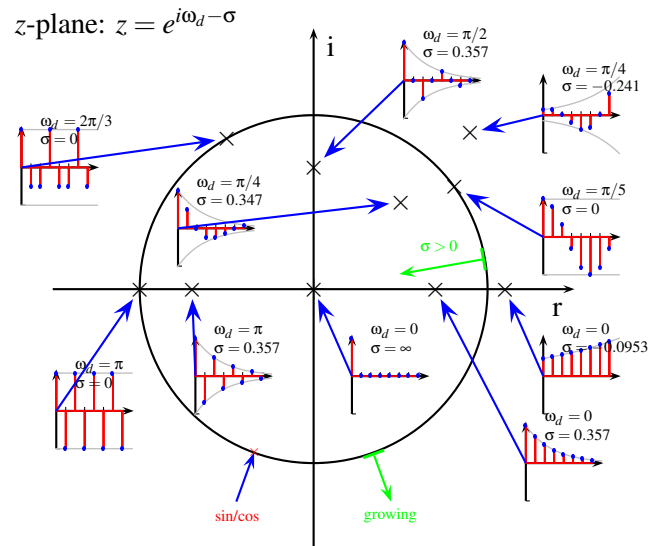
### 11.1 Poles and zeros in the $z$ -plane

Also, in the  $z$ -plane, a factorization of Eq. (21) is always possible:

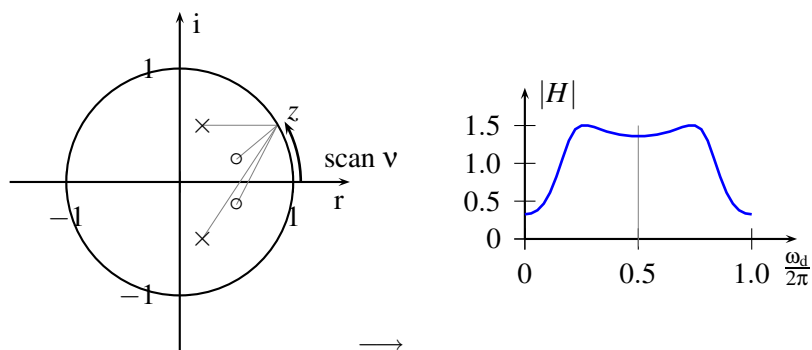
$$H(z) = \frac{\alpha_0 \prod_{k=1}^M (1 - z_{0k} z^{-1})}{\prod_{i=1}^N (1 - z_{pi} z^{-1})},$$

where  $z_{0k}$  are the *zeros* and  $z_{pi}$  are the *poles* of the function. The absolute value of the frequency response can be calculated in a similar way, by scanning  $z$  along the unit circle as shown in Fig. 38

$$|H(z)| = \frac{\prod \text{distances between } z \text{ and zeros}}{\prod \text{distances between } z \text{ and poles}}.$$



**Fig. 37:** The  $z$ -plane



**Fig. 38:** Calculation of the frequency response of a digital system from its poles and zeros of its transfer function in the z-plane

A handy formula for the phases of  $H(z)$  is also available

$$\angle H(z) = \sum \angle(z - \text{zeros}) - \sum \angle(z - \text{poles}) .$$

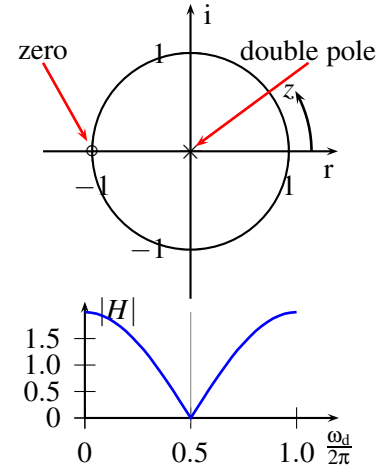
All this can be done by a very easy calculation, or even graphically, if you like.

## Examples

### 1. 2nd order non-recursive filter (FIR)

$$a_0 = \frac{1}{2} \quad ; \quad a_1 = 1 \quad ; \quad a_2 = \frac{1}{2} \quad ; \quad b_1 = b_2 = 0$$

$$\begin{aligned}
 h[n] &= \left\{ \frac{1}{2}, 1, \frac{1}{2} \right\} \\
 \longrightarrow y[n] &= \frac{1}{2}x[n] + x[n-1] + \frac{1}{2}x[n-2] \\
 \longrightarrow H(e^{i\omega_d}) &= \frac{\frac{1}{2} + e^{-i\omega_d} + \frac{1}{2}e^{-2i\omega_d}}{1} \\
 \longrightarrow H(z) &= \frac{1}{2} + z^{-1} + \frac{1}{2}z^{-2} = \frac{z^{-2}}{2}(z+1) \\
 \text{Poles: } z_{p1} = z_{p2} &= 0, \quad \text{Zeros: } z_{01} = -1
 \end{aligned}$$

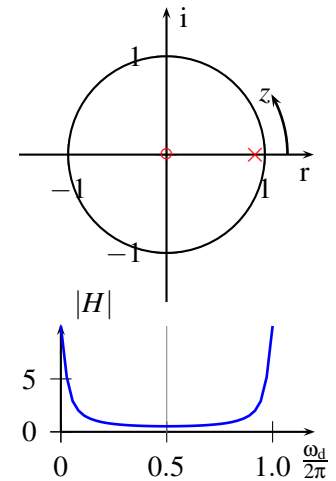


## 2. 1st order recursive filter (IIR)

$$a_0 = 1 \quad ; \quad a_1 = a_2 = \dots = a_n = 0 \quad ; \quad b_1 = 0.9 \quad ; \quad b_2 = \dots = b_m = 0$$

$$\begin{aligned}
 h[n] &= (0.9)^n \quad ; \quad n \geq 0 \\
 \longrightarrow H(z) &= \frac{1}{1 - 0.9z^{-1}} = \frac{z}{z - 0.9}
 \end{aligned}$$

$$\text{Poles: } z_{p1} = 0.9, \quad \text{Zeros: } z_{01} = 0$$



## 11.2 The z-transformation

Given  $h : \mathbb{Z}_0^+ \rightarrow \mathbb{R}$ , the **z-transformation** of  $h$  is

$$h[n] \xrightarrow{\mathbf{Z}} H(z) := \sum_{n=-\infty}^{\infty} h[n]z^{-n}$$

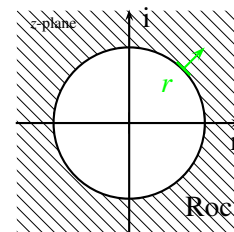
causal:  $h[n] = 0 \quad \forall n < 0$

This is the same as DFT plus substitution  $z := e^{i\omega_d}$ . [2ex]

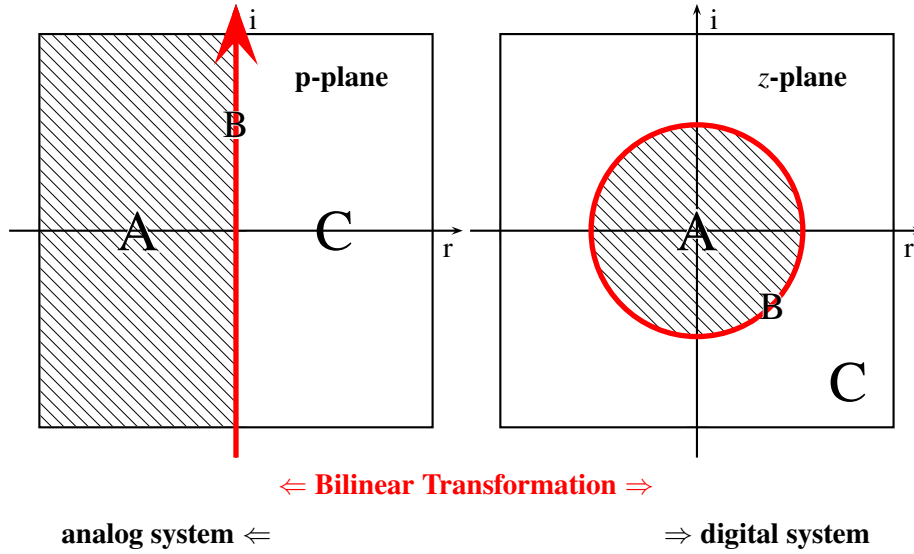
### Region of convergence

The Region of convergence (Roc) can be defined as follows:

$$\begin{aligned}
 \text{Roc} &:= \left\{ z : \left| \sum_{n=-\infty}^{\infty} h[n]z^{-n} \right| < \infty \right\} \\
 \Rightarrow \text{if } |h[n]| &< Mr^n \quad \forall n, \quad r \in \mathbb{R}^+ \quad \Rightarrow h(z) \text{ exists } \forall z \in \mathbb{C} : |z| > r \\
 &(\text{all the poles of } h(z) \text{ lie inside a circle of } |z| < r.)
 \end{aligned}$$






 Fig. 39:  $p$ -plane and  $z$ -plane

Signal	$z$ -Transform	Pole	Zero	Roc
$\delta[n]$	$1 \cdot z^0 = 1$	-	-	$z \in \mathbb{C}$
$\delta[n - n_0]; \quad n_0 \in \mathbb{N}$	$z^{-n_0}$	0	-	$z \neq 0$
step[n]	$\sum_{k=0}^{\infty} z^{-k} = \frac{1}{1 - z^{-1}} = \frac{z}{z - 1}$	1	0	$ z  > 1$
$h[n] = \alpha n$	$\frac{\alpha z}{(z - 1)^2}$	1	0	$ z  > 1$
$h[n] = b^n; \quad b \in \mathbb{R}$	$\sum_{k=0}^{\infty} \left(\frac{b}{z}\right)^k = \frac{z}{z - b}$	b	0	$ z  > b$
$h[n] = \cos(\omega n)$	$\frac{z(z - \cos(\omega))}{z^2 - 2z \cos(\omega) + 1}$	?	$0, \cos(\omega)$	$ z  > 1 - ?$
$h[n] = \sin(\omega n)$	$\frac{z \sin(\omega)}{z^2 - 2z \cos(\omega) + 1}$	?	0	$ z  > 1 - ?$

 Fig. 40:  $z$ -transform of common signals

If the impulse response of the system is decaying faster than or approximately exponentially, then all poles lie inside a circle of finite size, and the  $z$ -transform exists outside of that circle.

### 11.3 $z$ -transforms of common signals

Some  $z$ -transforms of common signals are given in Fig. 40 and the most common calculation rules are summarized in Fig. 41. As you can see, the behaviour is quite similar to what you already know from the Fourier and the Laplace transforms. The convolution rules, in particular, are the same. Also, the regions of the planes can be comparable if you want to compare analog systems and corresponding discrete systems by looking at their poles and zeros in the  $p$ -plane or  $z$ -plane, respectively. If you map both planes with the so-called *bilinear transformation* on each other, as shown in Fig. 39, you can directly compare frequency responses, stability issues, and a lot more. This mapping from analog to digital is also a common technique in digital filter design.

	Signal	z-Transform	$r_2 <  z  < r_1$
Linearity:	$a_1x_1[n] + a_2x_2[n]$	$a_1X_1(z) + a_2X_2(z)$	$\mathbb{D}_1 \cap \mathbb{D}_2$
Time shifting:	$x[n-k]$	$z^{-k}X(z)$	$\begin{cases} z \neq 0 \text{ if } k > 0 \\ z \neq \infty \text{ if } k < 0 \end{cases}$
Scaling:	$a^n x[n]$	$X(a^{-1}z)$	$ a r_2 <  z  <  a r_1$
Time reversal:	$x[-n]$	$X(z^{-1})$	$\frac{1}{r_1} <  z  < \frac{1}{r_2}$
Convolution:	$x_1[n] * x_2[n]$	$X_1(z)X_2(z)$	$\mathbb{D}_1 \cap \mathbb{D}_2$
Multiplication:	$x_1[n] \cdot x_2[n]$	$\frac{1}{2\pi i} \oint_C X_1(v)X_2\left(\frac{z}{v}\right)v^{-1}dv$	$r_{1l}r_{2l} <  z  < r_{1u}r_{2u}$
Differentiation:	$x[n] - x[n-1]$	$\frac{z-1}{z}X(z)$	
initial value:	$x[0] =$	$\lim_{z \rightarrow \infty} X(z)$	
final value:	$x[\infty] =$	$\lim_{z \searrow 1} (z-1)X(z)$	

**Fig. 41:** Calculating with the z-transform

### 11.4 The inverse z-transformation

Similar to the inverse Laplace transform, we now define the *inverse z-transform* as follows:

$$X(z) \xrightarrow{\text{Z}^{-1}} x[n] = \frac{1}{2\pi i} \oint_C X(z)z^{n-1}dz, \quad (22)$$

where  $C$  is an anticlockwise, closed path encircling the origin and entirely in the **region of convergence**.  $C$  must encircle all of the poles of  $X(z)$ . In this case Eq. (22) can be expressed using the calculus of residuals

$$x[n] = \sum_{z_{pk}} \text{Res}_{z_{pk}} (X(z)z^{n-1}).$$

#### Example

$z_0$  single pole of

$$X(z) := \frac{1}{z}, \quad k=1, \quad z_0=0.$$

$$\begin{aligned} x[n] &= \text{Res}_0 [X(z)z^{n-1}] \\ &= \frac{1}{(1-1)!} \cdot \frac{d^0}{dz^0} \left( \frac{1}{z} z^{n-1} \cdot (z-0)^1 \right) \Big|_{z=0} = \begin{cases} 1 & n=1 \\ 0 & n \neq 1 \end{cases} = \delta[n-1]. \end{aligned}$$

Remember the definition of the residuum, Eq. (19).

## 12 Digital filter design

In this section, I would like to give you some hints about how you can design a filter for your applications. We cannot go into details here, since filter design is a profession it itself and there are many books about it and also advanced toolboxes for computer-aided design. The sessions about modelling tools and control theory cover parts of it.

Having the mathematical concepts in mind, we can now use them. A common problem is to find the filter coefficients  $\alpha_i$  and  $\beta_i$  (for analog filters) or  $a_i$  and  $b_i$  (for digital filters), or, if you want, to have a simple implementation, the filter kernel  $h[n]$  of a FIR filter. You should have at least some idea about the frequency response; should it have low-pass, high-pass or band-pass characteristics, what is the centre or

edge frequency and what is the phase response of the system to be created? This is especially necessary if you design feedback loops, and stability is your concern.

Well, this is how you could start:

- Do not specify  $a_i, b_i$  but **zeros**  $z_{0k}$  and **poles**  $z_{pi}$  by the transfer function  $H(z)$ ,  $H(\omega_d)$ , the impulse response  $h[n]$ , or the step-response  $s[n]$ . Usually, you do this by trial and error: you place some poles and zeros on the  $z$ -plane and calculate the frequency response (if you are interested in the frequency domain), or the step response (if you are interested in time domain) or both. Then you can move these poles around to see how that changes the responses. You could add more zeros or poles and try to cancel out resonances if they bother you, etc.
- Then calculate  $a_i$  and  $b_i$  or  $h[n]$  for implementation. The implementation is straightforward and not very difficult; if you keep the order of your filter small, there will not be so many surprises later.

To make this trial-and-error job a little more sophisticated, you should know that

1. Because  $a_i, b_i$  usually should be real (for implementation),  $\rightarrow z_{0k}$  and  $z_{pi}$  need to be real or they appear in complex conjugate **pairs**.
2. The filter kernel should be finite or at least

$$\lim_{n \rightarrow \infty} h[n] \stackrel{!}{=} 0$$

otherwise the filter might be unstable. A consequence of this boundary is that  $|z_{pk}| < 1$ , which means the poles need to be located inside the unit circle.

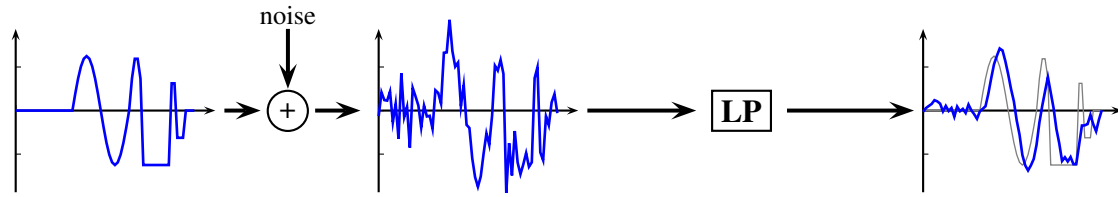
### Filter design check-list

Finally I shall give you a check-list for filter design:

1. Specify the *transfer function*  $H(f)$ .
2. Specify the type of the filter (*FIR or IIR*) *numerical stability, dispersion*.
3. Specify the *order* of the filter (number of poles and zeros) *compromise between implentational effort and approximation of ideal transfer function*.
4. Select the *method* for filter design:
  - numerical (computer-based) optimization of coefficients;
  - convert analog filter to digital filter *impulse response invariant (windowing) design, transfer function invariant design, bilinear transformation*;
  - use filter transformation prototypes.

## 13 The Kalman filter

In this section, I present the idea of a special, highly useful filter, the *Kalman filter*. Though this filter can be implemented as an ordinary digital IIR filter, the concept behind it may be more difficult to understand. The Kalman filter is useful to filter out the noise of a signal whose signal-to-noise ratio is very poor, but about which you know something of the underlying process producing the input stream of measurements. From this extra knowlege, one can take advantage and improve the signal quality, effectively removing noise from the signal. The Kalman filter does this best, it is the optimal filter with respect to virtually *any criterion* that makes sense.



**Fig. 42:** Filtering a noisy signal with a low-pass filter. The result is time-shifted and the high-frequency components (the sharp edges) of the signal are not well reconstructed.

### 13.1 Fighting the noise

We discussed common sources of noise earlier in this lecture. The noise may come from the nature of the process that gives you the measurements or it may come from the detector or sensor (including the noise sources which belong to the digitization process). In any event, you very often end up with a stream of measurements  $x[n]$  which has a bad signal-to-noise-ratio. For signal processing, you need to improve the signal quality, remove the noise, and since you cannot improve the measurement hardware, you will have to do your best to do it within the digital signal process itself.

The first idea which comes to mind is to use a low-pass filter (do some averaging of the input signal). This idea is not too bad and can work well if your sampling frequency is high enough and two side effects do not bother you: mainly the latency, time shift and phase response introduced with the filter and the fact that you remove only the high-frequency noise components. As a consequence, the higher harmonics of your signal may be smeared out and you keep the low-frequency noise components on your signal. If you can live with this, fine, but in many situations, like the one shown in Fig. 42, you might not be very satisfied with the result.

The Kalman filter can improve the situation. This means that it introduces nearly no latency while doing a good job of noise filtering and conserving the high-frequency components of your signal. Last but not least, the Kalman filter is still a causal system, but it can only work if you have some extra knowledge about the underlying process and if you are able to create a model (at least a very simple one) of it.

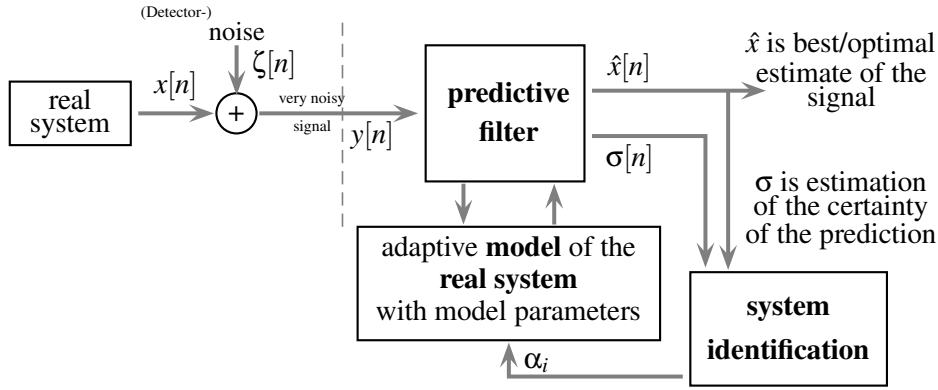
If you also have the chance to use non-causal filtering (maybe because the data is produced in chunks and can be processed as a whole), then techniques which are described in the section about wavelets may also be applicable.

### 13.2 Predictive/adaptive filters

First, the Kalman filter is an *adaptive filter*; this means that its filter coefficients are not constant, but instead change in adaptation to the filtered data itself. Figure 43 explains the principle. A special system identification block within the filter analyses the data and calculates new optimal filter coefficients from it using sophisticated algorithms.

The second attribute of the Kalman filter is that it is *predictive*. This simply means that it has some algorithm which allows it to calculate a prediction (expectation) of the current measurement or input value, based on the latest measurements and a *model* of the underlying process. Both the measured value and the predicted value are then combined to produce the output of the filter.

The trick is how to do this combination in such a way that, depending on the certainty of either the predicted or the measured value, the output represents the best certainty of both together. This trick is based on the rule of ‘propagation of error’ (a well-known concept which will be discussed soon). This way, it is guaranteed that the output variance is always smaller than (or equal to) the variance of the input signal.



**Fig. 43:** Principle of an adaptive and predictive filter (like the Kalman filter). The filter consists of a model of the underlying process which can calculate predicted values from the model parameters (and the latest measured values). The model parameters are adapted from a system identification block. The algorithm is essential for the optimality of the Kalman filter; it always follows the variance of the measured data and the predicted data, based on the rule of ‘propagation of error’. In this way, it is guaranteed that the output variance is always smaller than (or equal to) the variance of the input signal.

### 13.3 Example: navigation

To understand how that works, we are going to develop a simple, one-dimensional example: the estimation of the position of a boat on the (one-dimensional) ocean. Suppose we are only interested in one parameter (e.g., the latitude or longitude). For position measurements, the crew can use different methods, let us say a sextant for navigation with the sun or the stars and a modern GPS receiver. Depending on the person who is doing the position determination, the position values  $x[n]$  may have different precision (expressed by the variances  $\sigma[n]$ ), depending on the method used or the person who does it.

First think of the simple situation where the boat is more or less at rest and a first position measurement is done (at time  $t_1$ ) with an uncertainty known to be  $\sigma_1$ , which might be very large, because let us say a beginner does this navigation:

$$\text{first measurement: } x_1 \pm \sigma_1 := (x(t_1) \pm \Delta x(t_1)) \quad .$$

Now we analyse how a second measurement (nearly at the same time  $t_2 \approx t_1$ )<sup>4</sup>

$$\text{second measurement: } x_2 \pm \sigma_2 := (x(t_2) \pm \Delta x(t_2))$$

can improve the knowledge of the position. Assume the uncertainty  $\sigma_2$  of this second measurement is smaller than the first one (because now the captain himself did the measurement). You could throw away the first measurement and only use the second one. But this would be not the best solution, because the first measurement also contains information we could benefit from. So the clever way is to combine both measurements

$$\begin{aligned} \Rightarrow \text{best estimate: } \hat{x} &= \frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2} \cdot x_1 + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} \cdot x_2 \\ \text{uncertainty: } \hat{\sigma} &= \frac{1}{\sqrt{\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}}} \leq \min(\sigma_1, \sigma_2) \end{aligned}$$

<sup>4</sup>For the moment time does not play a role because we assumed the boat to be at rest.

so that the variance  $\hat{\sigma}$  of the resulting position measurement  $\hat{x}$  is even better than the best of each single measurement.

But what if some noticeable/relevant time has passed between the measurements?

To be more general we can say:

$$\begin{aligned}\hat{x}(t_2) &= \frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2} \cdot x(t_1) + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} \cdot x(t_2) \\ &= x(t_1) + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} \cdot (x(t_2) - x(t_1)) .\end{aligned}$$

Now we consider a stream of (new) input data  $x_{n+1} := x(t_{n+1})$  which should be combined with the latest best value  $\hat{x}_n$  to produce a new best value  $\hat{x}_{n+1}$ . (Remember: the variances  $\sigma_{n+1}$  of each measurement are assumed to be known.) This is trivial if the measurement device is the same for all measurements, since  $\sigma_{n+1}$  can be assumed to be constant. But even if  $\sigma_{n+1}$  is not known in advance, one can estimate it by calculating the variance (e.g., with the method of running statistics) of the input signal stream.

$$\hat{x}_{n+1} = \hat{x}_n + K_{n+1} (x(t_{n+1}) - \hat{x}_n) \quad (23)$$

$$\hat{\sigma}_{n+1} = \frac{1}{\sqrt{1/\hat{\sigma}_n^2 + 1/\sigma_{n+1}^2}} ; \quad (24) \quad \text{‘prediction equation’}$$

$$\text{where } K_{n+1} := \frac{\hat{\sigma}_n^2}{\hat{\sigma}_n^2 + \sigma_{n+1}^2} .$$

‘Kalman gain’

The new prediction (say, best estimate of the position) is based on the old prediction and the new measurement value. There is one curiosity:  $\hat{\sigma}$  is shrinking all the time! Becoming smaller and smaller and approaching zero. This means that with time you can get a really precise value of your position with any bad and ugly measurement device. But remember that this holds only if the boat is really at rest and the position does not change with time.

Finally, we want to discuss the more realistic case that the boat is moving (with a constant velocity  $v$ ). We can now extend the example to be even more general:

Since we know the physical influence of a velocity of the boat’s position

$$\frac{dx}{dt} = v + \Delta v$$

(this is the underlying process) we can introduce a **model**:

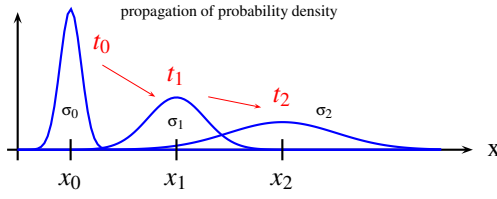
$$x(t) = v(t) \cdot (t - t_0) + x(t_0) ,$$

where  $v(t) \pm \Delta v(t)$  is assumed to be known by a different measurement (called system identification). Besides this, we also assume that  $v$  is constant or changing only adiabatically (slowly compared to the sampling rate of the position measurements). The model also tells us the expected uncertainty of the calculated position value:

$$\sigma(t) = \sqrt{(\Delta v \cdot (t - t_0))^2 + (\sigma(t_0))^2} .$$

If, at this moment, you do not understand this formula, read the paragraph about ‘propagation of error’. Figure 44 shows you what this means: Because the velocity also has a non-zero variance, the variances of the position derived from it become larger with time ( $\sigma$  is growing!), so the uncertainty is increasing!

Now let us see what this means for our example. Since the boat is moving, we cannot simply combine the latest best value with the new measurement, because some time has passed since the last



**Fig. 44:** Evolution of the probability density of the position derived from the model. Because the velocity also has a non-zero variance, the variances of the positions derived from it become larger with time, so the uncertainty is increasing.

measurement and we know (by our model) that the position must have changed in the meantime and we cannot simply combine it (to produce an average). Instead, we have to consider this position change since the last measurement. This can be done by a prediction of the actual position by our model:  $x(t_{n+1}) =: \bar{x}_{n+1}$  based on the model parameter  $v$ , the last ‘known’ position  $\hat{x}(t_n)$ :

$$\bar{x}_{n+1} := v_n \cdot (t_{n+1} - t_n) + \hat{x}(t_n) , \quad (25)$$

and a measure of the certainty of this prediction:

$$\bar{\sigma}_{n+1} := \sqrt{(\Delta v_n \cdot (t_{n+1} - t_n))^2 + \hat{\sigma}_n^2} . \quad (26)$$

### Propagation of error

Consider a function

$$f = f(\alpha_1, \alpha_2, \dots, \alpha_n)$$

which is a function of one or more (model) parameters  $\alpha_i$ , each with corresponding errors  $\Delta\alpha_i$ . Now you want to know the consequence of these errors on the overall error or uncertainty of  $f$ .

$$\Rightarrow \Delta f = \sqrt{\sum_i \left( \frac{\partial f}{\partial \alpha_i} \Delta \alpha_i \right)^2} .$$

Maybe you have seen this before, because this is a very common formula in physics and applies everywhere where measurements are done. In our example this means:

$$x(t) = v \cdot t + x_0$$

$$\begin{aligned} \Rightarrow \Delta x &= \sqrt{\left( \frac{\partial x}{\partial v} \Delta v \right)^2 + \left( \frac{\partial x}{\partial x_0} \Delta x_0 \right)^2 + \left( \frac{\partial x}{\partial t} \Delta t \right)^2} \\ &= \sqrt{(\Delta v \cdot t)^2 + (\Delta x_0)^2} , \end{aligned}$$

(assuming  $\Delta t = 0$ ).

This assumes that the individual errors are not correlated and are Gaussian distributed. This is likely because of the central limit theorem, but not guaranteed!

### The Kalman gain

Now we use the Kalman prediction Eqs. (23) and (24) to combine the new measurement  $x_{n+1}$  with the value from the model, which is a propagation of the latest predicted value  $\hat{x}_n$  for the time  $(t_{n+1} - t_n)$  using the model parameters ( $v$ ) [see Eqs. (25) and (26)]. For all values, the uncertainty or measurement error is taken into account.

The output ( $\hat{x}_n$  from input  $x_n := x(t_n)$ ) of the Kalman filter becomes:

$$\hat{x}_n = \bar{x}_n + \bar{K}_n (x_n - \bar{x}_n) \quad ; \quad \hat{\sigma}_n = \frac{1}{\sqrt{1/\bar{\sigma}_n^2 + 1/\sigma_n^2}}$$

where

$$\bar{K}_n := \frac{\bar{\sigma}_n^2}{\bar{\sigma}_n^2 + \sigma_n^2}$$

is the redefined *Kalman gain*.

With some additional substitutions,  $T := t_{n+1} - t_n$ ,

$$\begin{array}{lll} \hat{x}_n & \longrightarrow & y[n] \quad \text{Kalman filter output} \\ \hat{x}_{n-1} & \longrightarrow & y[n-1] \quad \text{last output value} \\ x_n & \longrightarrow & x[n] \quad \text{input value ,} \end{array}$$

one can see the general structure (difference equation) of the digital filter:

$$\begin{aligned} y[n] &= v_n \cdot T + y[n-1] + \bar{K}_n (x[n] - v_n \cdot T - y[n-1]) \\ &= \bar{K}_n \cdot \underbrace{x[n]}_{\text{measurement}} + (1 - \bar{K}_n) \underbrace{(v_n \cdot T + y[n-1])}_{\text{model}} \end{aligned}$$

weights which represent the accuracy  
of the data and the model

And this is also the way the Kalman filter could be implemented. Notice that the second term is a recursive part of the filter. The Kalman gain is the weight which decides how much model and how much input data goes to the output. If the prediction from the model is bad (the corresponding estimated variance  $\bar{\sigma}$  is large), the Kalman gain tends to  $\bar{K} = 1$  and so the input will be directly passed to the output without using the model at all, but also without making the output data more noisy than the input. On the contrary, if the input data occasionally has a lot of noise and the model and its model parameters are still fine,  $\bar{K}$  will be closer to zero and the output data of the Kalman filter will be dominated by the model predictions and its statistics.

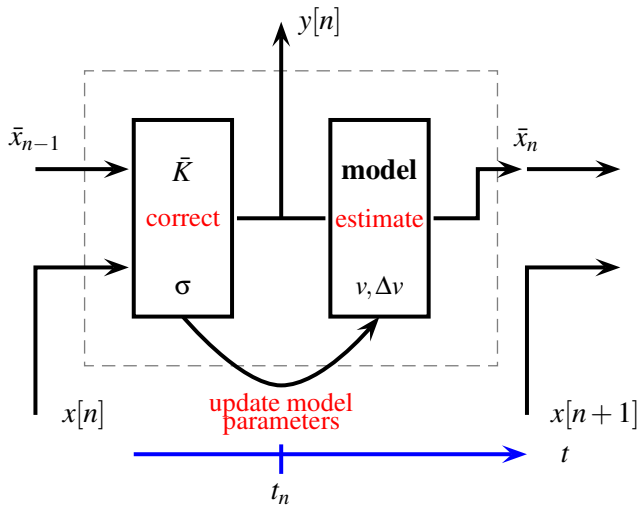
### 13.4 The internal structure

Now let us summarize, what we learned about the Kalman filter:

- The Kalman filter makes use of an internal model and model parameters.
- The ‘internal’ system/model parameters ( $\sigma, v, \Delta v$ ) are calculated from the input data itself.
- Also the variances of the input data stream and the variances of the derived predicted values belong to the internal parameters.
- The Kalman filter makes use of the ‘propagation of error’ principle.
- The Kalman filter has three fundamental functional blocks:
  1. The combination of model predicted with input data stream.
  2. The prediction block for the next model value.
  3. The system identification block for update of the model parameters.

The internal structure of the Kalman Filter is shown in Fig. 45.





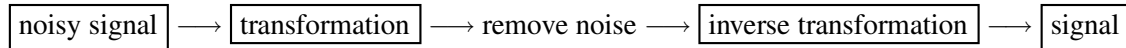
**Fig. 45:** Internal structure of the Kalman filter. In one block, the model prediction  $\bar{x}_{n-1}$  from the last step is combined with the actual input value  $x[n]$  and passed to the output  $y[n]$ . The second block calculates the prediction for the next time-step based on the model parameters and their variances. In parallel, the model parameters need to be updated by a system identification algorithm.

We discussed a fairly simple example. In more realistic applications, the model can be very complex. But with more and more model parameters, more error contributions are added; this means that the optimal model complexity needs to be evaluated. The model should be as complex as necessary to reduce the noise, but also as simple as possible.

The trick is that the  $\sigma$  of the output will always be smaller than (or in worst case equal to) the  $\sigma$  of the input<sup>5</sup>. So the output will be best noise-filtered (depending on the model). A bad model generates a  $\bar{K}$  near to one, so the input is not corrected much (no effective filtering).

## 14 Wavelets

As mentioned in the previous section, wavelets can be helpful (among other things) at removing noise with a special spectral characteristics from a (non-causal) signal.



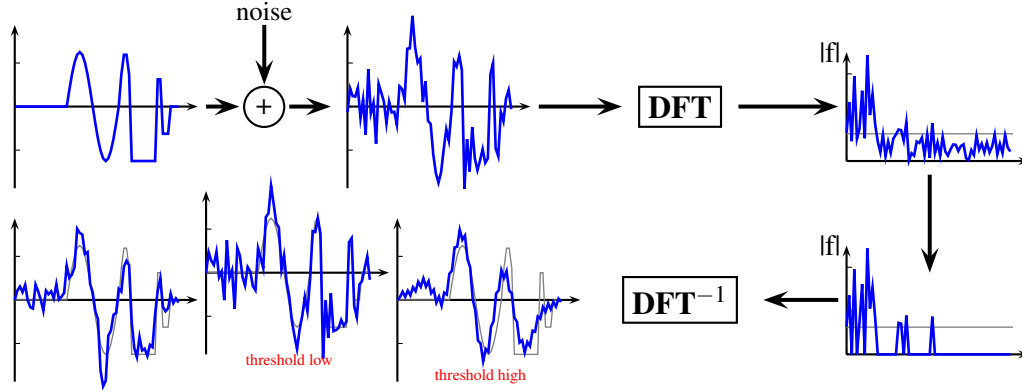
A similar method can also be used to select only especially desired spectral components with special filters; this is also often used for (lossy) data compression as for audio signals or images. Last but not least, the wavelet transformation also has applications in solving special classes of differential equations. We shall not go into these very popular fields, but instead restrict ourselves to the question of how we can make use of the wavelets for noise removal.

### 14.1 Fighting noise with the DFT

A quick solution will be to use the digital Fourier transformation, defined in Section 10.1, to do this job. Let us see how that works out (Fig. 46).

As you can see in this example, the reconstruction of the original signal is not too bad (in contrast with the method of the low-pass filter discussed in the previous section, there is no problem with time shift or amplitude degradation of the higher frequency components of the signal), but one major problem is immediately visible: High frequencies are there, but the phase information of the higher harmonics is distorted by cutting away some of the (noisy) components. Modifying only single spectral components has effects everywhere in time! This is the nature of the Fourier decomposition. To avoid this, one wants to use a different transformation like the wavelet transformation, because wavelets are not only localized in frequency (like  $\sin()$  and  $\cos()$ ), but also localized in time. This means that if you remove some of the wavelet components from the wavelet spectrum, the time domain signal is affected only locally. In this

<sup>5</sup>The principle only works if the errors of the signal measurements are independent of each other and the distribution of the noise and of all internal parameters is Gaussian. If this is not the case, the 'propagation of error' formula underestimates the resulting variance.



**Fig. 46:** Example for using the digital Fourier transformation for noise filtering. The noisy signal is transformed to frequency domain. Then all spectral components which are below a certain threshold (for amplitude) are removed (which means they are set to zero) and finally the data is transformed back into time domain. Depending on the threshold used, the result is fairly good, or still too noisy if the threshold was too low, or the reconstruction of the signal is bad, if the threshold was set too high.

way, it is possible to remove high-frequency noise where the original signal is smooth and still conserve the sharp edges of the waveform.

## 14.2 Localized functions

Localized functions have a body that is located around some time  $t_0$ . This means that, for all times far away from  $t_0$ , the function tends to zero. It especially goes asymptotically to zero for  $t \rightarrow \pm\infty$ . Localized in frequency simply means that for  $\omega \rightarrow \infty$ , the spectrum goes to zero. Sometimes it is also required that the spectrum goes to zero for  $\omega \rightarrow 0$  (which is the case for wavelets). There is also a rigorous definition of localized functions, which are required to be exactly zero outside a region around the body.

There is nothing special about localized functions (see Fig. 47). In contrast to the  $\sin()$  function which is well localized in frequency (only one frequency component is present) but not at all localized in time<sup>6</sup>, the product of a Gauss function with a  $\sin()$  function is localized both in time and in frequency<sup>7</sup>.

Wavelets are special functions  $\Psi(t) : \mathbb{R} \xrightarrow{\Psi} \mathbb{R}$  with special requirements. One which is already mentioned is that  $\Psi$  should be well localized in time and frequency. Second, it is required that

$$\int \Psi dt = 0 .$$

And finally, more technical requirements are needed, in particular, applications to make the calculation easy.

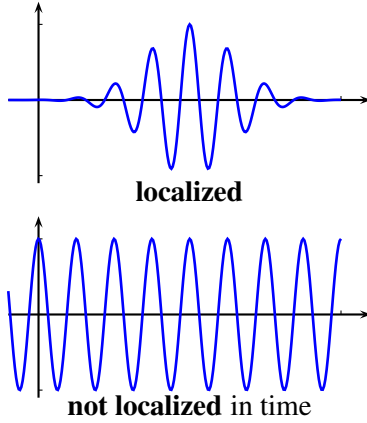
## 14.3 Wavelet families

There are many kinds of wavelets and **wavelet families** coming from practical applications:

- smooth wavelets,
- compactly supported wavelets (**Daubechies, 1988**),
- wavelets with simple mathematical expressions (**Haar, 1900, Meyer, Morlet**),

<sup>6</sup>On the contrary, any function consisting of a  $\delta$ -function will probably be localized in time but definitely not in frequency.

<sup>7</sup>The Gauss function itself would also be localized in frequency, but it does not fulfil the more restrict requirement that the spectrum go to zero for  $\omega \rightarrow 0$ .



**Fig. 47:** Example of localized and not localized functions

- wavelets with simple associated filters,
- discrete wavelets,
- etc.

Each wavelet family is generated from a ‘**mother wavelet**’  $\Psi_{1,0}$  (which fulfils the requirements mentioned above) by a **transformation** which is a combination of translation and dilatation

$$\text{family: } \Psi_{1,0}(x) \mapsto \Psi_{a,b} := \frac{1}{\sqrt{a}} \Psi_{1,0}\left(\frac{x-b}{a}\right) \quad ; a \in \mathbb{R}^+, b \in \mathbb{R} .$$

If you do a proper selection of  $a$ ’s and  $b$ ’s, you can get a wavelet family which forms a **basis** (like  $\{\sin(n\omega t), \cos(n\omega t)\}$  do).

With the following set of parameters  $a$  and  $b$ :

$$a := 2^{-j} \quad ; \quad b := k \cdot 2^{-j} \quad ; \quad j, k \in \mathbb{Z} \quad ,$$

which is called ‘critical sampling’, one gets

$$\Rightarrow \Psi_{j,k}(x) := 2^{j/2} \Psi(2^j x - k) \quad , \quad (27)$$

an orthonormal Hilbert basis.

#### 14.4 Discrete wavelet transformation

As with the discrete Fourier transformation, one can decompose an arbitrary function  $f$  to this basis formed by the wavelets (27). The ‘spectral’ components are expressed by the coefficients  $c_{j,k}$ .

$$f = \sum_{j,k \in \mathbb{Z}} c_{j,k} \cdot \Psi_{j,k} .$$

The difference, compared with the ordinary Fourier transformation, is that the coefficients form a two-dimensional array; and you may ask what the benefit of having even more coefficients than with a Fourier transformation will be. The answer is that the wavelet decomposition can be done in a tricky way: that only (the first) very few coefficients hold most of the information of your function  $f$  and almost all other components can be neglected. Of course, this depends on the class of functions and on the selected (mother) wavelets.

The second big task—after having selected the wavelet family you want to use—is how to get the coefficients, or, more generally, how to perform the *Discrete Wavelet Transformation*

$$f(x) \xrightarrow{\text{DWT}} \{c_{j,k}\} \quad ; \quad c_{j,k} \in \mathbb{R}; j, k \in \mathbb{Z} .$$

The algorithm for the DWT is a bit tricky, but this problem has been solved and a very efficient algorithm exists. Unfortunately it is out of the scope of this lecture to explain how it works. So please consult the literature. But still one word: it makes use of iterative (digital) filter banks and the problem can best be understood in frequency domain. Also, the concept of *the scaling function* plays an important role here, limiting the number of coefficients to a *finite* number.

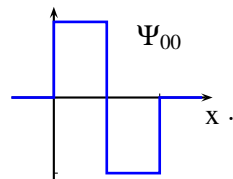
### 14.5 Wavelet applications

What are wavelets good for? Functions can be better approximated with wavelets if they have discontinuities, sharp spikes, or a non-periodic structure (they are localized). Why not use the Fourier basis?

Fourier basis	Wavelets
<ul style="list-style-type: none"> <li>– Basis functions are localized in frequency, but not localized in time domain.</li> <li>– Small changes in the spectrum will produce changes of the signal everywhere in time.</li> <li>– Functions with discontinuities and or sharp spikes need a big number of spectral coefficients, sometimes even an infinite number, to be properly approximated.</li> </ul>	<ul style="list-style-type: none"> <li>– Basis functions are localized in frequency (scale/dilatation) and time (translation).</li> <li>– This can be an advantage for signal processing: Many signals can better be represented in wavelet-basis than in ‘spectral lines’, fewer coefficients → data compression.</li> </ul>

### 14.6 Example: the Haar wavelet

The simplest and oldest of all (mother) wavelets is the **Haar wavelet**:

$$\Psi_{00}(x) := \begin{cases} 1 & 0 \leq x < 0.5 \\ -1 & 0.5 \leq x < 1 \\ 0 & \text{else} \end{cases}$$


With critical sampling (27), the wavelet family can be expressed as

$$\Psi_{j,k}(x) := 2^{j/2} \cdot \Psi_{00}(2^j x - k) \quad ; j, k \in \mathbb{Z}.$$

Let us see if they fulfil the wavelet criteria: Obviously, they are localized in  $x$ ,  $\int \Psi dx = 0$ , and

$$\int \Psi_{j,k} \cdot \Psi_{j',k'} = \begin{cases} 1 & j = j' \text{ and } k = k' \\ 0 & \text{else} \end{cases},$$

so they really form an orthonormal basis of the Hilbert space. Of course, there is also a disadvantage: The Haar wavelets are not smooth, so they may not fit best for smooth functions, but they will do their job fairly well for discrete (sampled) data.

The trick now is that—let us say you have  $n = 2^m$  samples of a digitized function—you first map the data to the interval  $[0; 1[$ . Then write down all wavelets which have a body inside this interval, stop if the wavelets become small enough to fit a single sample, and then do the decomposition. You can do this straightforwardly with a big set of equations (one for each sample point) and solve it. As already mentioned, this is not the most efficient way to do this, but you will get the idea.

In case you really do this homework, I expect the following problem: How can a function with  $\int_0^1 f(x) dx \neq 0$  be transformed? The answer is: either you try to extend the number of coefficients to

infinity (especially all the coefficients with  $j < 0$ ) or—and this is, of course, recommended—you add (at least) one additional function to the set of wavelets which replaces the infinite number of smaller and smaller scale wavelets; namely,  $\Phi_{j_0,k}; k \in \mathbb{Z}$  ( $j_0$  is fixed here, so these functions form only a one dimensional array), the *scaling function*. The scaling function is not a wavelet, since  $\int \Phi(x) dx = 1$  is required, but you can prove that the set

$$\{\Phi_{j_0,k}, \Psi_{j,k} \quad ; j \geq j_0, k \in \mathbb{Z}\}$$

spans the same space as the full basis  $\{\Psi_{j,k} \quad ; j, k \in \mathbb{Z}\}$ .

Now you might still be worried about the  $k$  within the definition, but consider our example: let us choose  $j_0 = 0$ . The restriction of the domain to  $[0; 1[$  means that we need only consider wavelets with  $0 \leq k < j$  and there is a maximal  $j$  because of our sampling resolution ( $j < m$ ). All in all, the number of non-zero wavelet components is limited to a finite number. Finally, the missing scaling function is simply

$$\Phi_{0,k} := \begin{cases} 1 & \text{for } 0 \leq x < 1, \\ 0 & \text{else} \end{cases}$$

independent of  $k$ , so we need only one. Now all functions (with a limited number of samples) can be transformed to a finite set of wavelet coefficients. If the number of non-zero wavelet coefficients is smaller than the number of samples, you might be happy.

Unfortunately, the application of the wavelets is limited: Although the discrete wavelet transformation is well defined, and efficient algorithms have been worked out, the success of using the wavelets depends on the *choice* of the wavelet family. If you cannot find a clever wavelet family which fits well with your particular problem, you will be lost, and there is no generic way to help you out there.

## Acknowledgements

I would like to thank Kirsten Hacker for proofreading the manuscript. Thanks to all those who sent me their comments and also pointed out some bugs and confusions after the presentation at the CERN school. If this started fruitful discussions, I am happy.

## Bibliography

Many ideas for instructive pictures are taken from Smith's book, which is pretty much a beginner's guide to digital signal processing. Figures 12, 14, 23, 25, 28, and 29 have their origin there. There are many other books on digital signal processing, wavelets, and the Kalman filter. Here, I just list a short collection of textbooks and similar papers which inspired and taught me the latter. You have to find out by yourself if they will also be useful to you.

- S.W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing* (California Technical Pub., San Diego, CA, 1997).
- W. Kester, *Mixed-Signal and DSP Design Techniques* (Newnes, Amsterdam, 2003).
- W.H. Press, B.P. Flannery, S.A. Teukolsky and W.T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed. (Cambridge University Press, 1992).
- B.D.O. Anderson and J.B. Moore, *Optimal Filtering* (Prentice-Hall, Englewood Cliffs, NJ, 1979).
- G.F. Franklin, J.D. Powell and M.L. Workman, *Digital Control of Dynamic Systems*, 3rd ed. (Addison-Wesley, Menlo Park, CA, 1998).
- E. Kreyszig, *Advanced Engineering Mathematics*, 8th ed. (Wiley, New York, 1999).
- D. Lancaster, *Don Lancaster's Active Filter Cookbook*, 2nd ed. (Newnes, Oxford, 1996).
- P.M. Clarkson, *Optimal and Adaptive Signal Processing* (CRC Press, Boca Raton, 1993).

- G. Strang and T. Nguyen, *Wavelets and Filter Banks* (Cambridge Univ. Press, Wellesley, MA, 1997).
- B. Widrow and S.D. Stearns, *Adaptive Signal Processing* (Prentice-Hall, Englewood Cliffs, NJ, 1985).
- G. Welch and G. Bishop *An Introduction to the Kalman Filter*, University of North Carolina at Chapel Hill, Department of Computer Science, <http://www.cs.unc.edu/~{welch,gb}>.
- Wikipedia, The Free Encyclopedia. May 1, 2007, <http://en.wikipedia.org/>.