

PH 506

Image Processing

Part II: Image Transforms

Dr. Peter Blümler (pb@ukc.ac.uk)

Recommended Books / Background Reading:

- R. C. Gonzalez and R. E. Woods: „Digital Image Processing“, Addison-Wesley Publ., New York 1993.
- J. C. Russ: „*The Image Processing Handbook*“ CRC Press, Boca Raton, FL, 1992.

Advanced:

- R. N. Bracewell: „*The Fourier Transform and Its Application*“, McGraw-Hill, New York, 1986.

1. Mathematical Background

1.1. Some Definitions

(recap: math II lectures and webpages)

Periodic functions: [1.1.1]

A function is periodic (repetitive) when $f(t + T) = f(t)$ is valid for all t . Then T is called the period of the function.

Even functions:

A function is even when $f(-t) = f(t)$ [1.1.2]

Example: $\cos(t)$

$$\text{Integral: } \int_0^b f(t) dt = \int_{-b}^0 f(t) dt \Rightarrow \int_{-b}^b f(t) dt = 2 \int_0^b f(t) dt \quad [1.1.3]$$

Odd functions:

A function is odd when $f(-t) = -f(t)$ [1.1.4]

Example: $\sin(t)$

$$\text{Integral: } \int_0^b f(t) dt = - \int_{-b}^0 f(t) dt \Rightarrow \int_{-b}^b f(t) dt = 0 \quad [1.1.5]$$

Products of even/odd functions:

$$\text{Even} \bullet \text{Even} = \text{Even} \quad [1.1.6]$$

$$\text{Even} \bullet \text{Odd} = \text{Odd} \quad [1.1.7]$$

$$\text{Odd} \bullet \text{Even} = \text{Odd} \quad [1.1.8]$$

$$\text{Odd} \bullet \text{Odd} = \text{Even} \quad [1.1.9]$$

1.2. Fourier Series

Fourier-Theorem: (Synthesis equation)

Any continuous, periodic (repetitive) function $f(t)$ with a repeat period T can be expressed by a sum of sine- and cosine-functions (*Fourier-series*):

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos\left(\frac{2\pi n}{T}t\right) + b_n \sin\left(\frac{2\pi n}{T}t\right) \quad [1.2.1]$$

or with linear frequency $\nu = \frac{1}{T}$

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos(2\pi\nu nt) + b_n \sin(2\pi\nu nt) \quad [1.2.2]$$

or with angular frequency $\omega = 2\pi\nu$

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos(\omega nt) + b_n \sin(\omega nt) \quad [1.2.3]$$

where $a_0/2$ represents the average value (often called DC-value, because it corresponds to zero frequency). a_n and b_n are the amplitudes of the sine/cosine terms and are called **Fourier-coefficients**.

The typical example is the representation of a square wave (see Fig. 1) as a superposition of sine/cosine-waves. To have a simple start, we define it with period T and even (hence sine-components do not distribute in the superposition and all b_n are zero).

The analysis (see later) gives:

$$f(t) = \frac{1}{2} + \frac{2}{\pi} \cos\left(\mathbf{1} \frac{2\pi}{T}t\right) - \frac{2}{3\pi} \cos\left(\mathbf{3} \frac{2\pi}{T}t\right) + \frac{2}{5\pi} \cos\left(\mathbf{5} \frac{2\pi}{T}t\right) - \frac{2}{7\pi} \cos\left(\mathbf{7} \frac{2\pi}{T}t\right) + \dots$$

$$f(t) = \frac{1}{2} + \frac{2}{\pi} \sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} \cos\left(\mathbf{(2n+1)} \frac{2\pi}{T}t\right)$$

The ‘spectrum’ that is the amplitude of the a_n versus frequency (rather than time!) is then

n	0	1	2	3	4	5	6	7
a_n	$\frac{1}{2}$	$\frac{2}{\pi}$	0	$-\frac{2}{3\pi}$	0	$\frac{2}{5\pi}$	0	$\frac{2}{7\pi}$
$\nu = \frac{1}{T}$	0	1	2	3	4	5	6	7

This distribution (spectrum) is shown in Fig. 2.

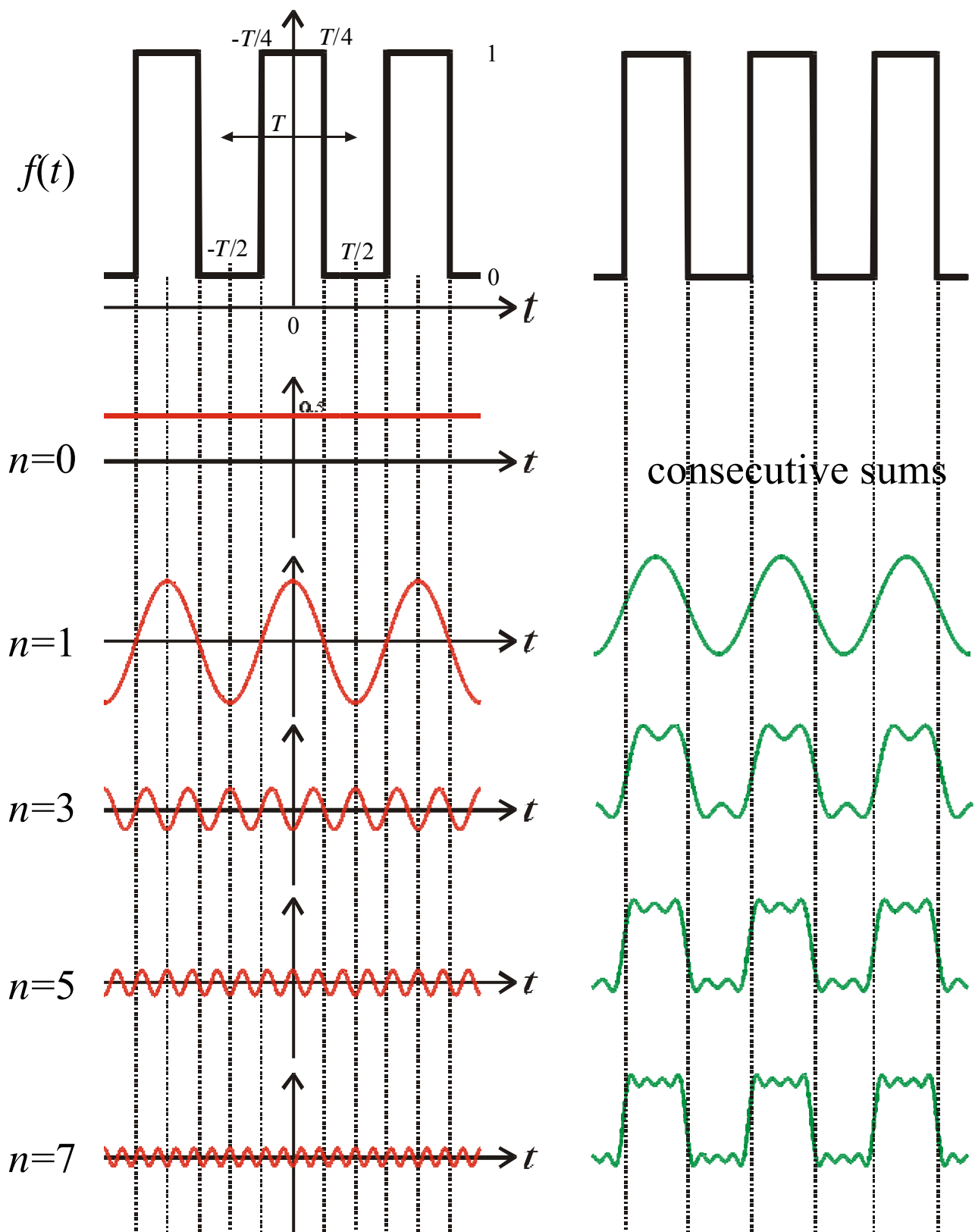


Fig. 1: Superposition of cosine-functions to form a square wave, $f(t)$. On the left individual terms, on the right their sum (e.g. for $n=3$ this sum corresponds to summing $n=0, 1$ and 3).

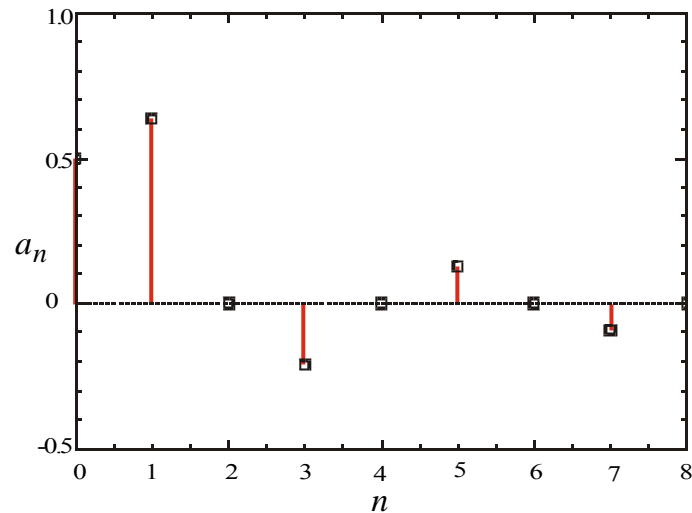


Fig. 2: Spectrum of the *Fourier*-coefficients versus their frequencies (abscissa in multiples of $1/T$).

Fourier Theorem: (Analysis equation)

The *Fourier*-coefficients can be calculated (see Math II handout) by using

$$a_0 = \frac{1}{T} \int_{-T/2}^{T/2} f(t) dt \quad [1.2.4]$$

$$a_n = \frac{2}{T} \int_{-T/2}^{T/2} f(t) \cos\left(\frac{2\pi n}{T} t\right) dt = \frac{2}{T} \int_{-T/2}^{T/2} f(t) \cos(2\pi n t) dt = \frac{2}{T} \int_{-T/2}^{T/2} f(t) \cos(\omega n t) dt \quad [1.2.5]$$

$$b_n = \frac{2}{T} \int_{-T/2}^{T/2} f(t) \sin\left(\frac{2\pi n}{T} t\right) dt = \frac{2}{T} \int_{-T/2}^{T/2} f(t) \sin(2\pi n t) dt = \frac{2}{T} \int_{-T/2}^{T/2} f(t) \sin(\omega n t) dt \quad [1.2.6]$$

Hence for the square-wave in Fig. 1:

$$\text{for one period } T : f(t) = \begin{cases} 1 & \text{for } -\frac{T}{4} < t \leq \frac{T}{4} \\ 0 & \text{for } -\frac{T}{2} \leq t < -\frac{T}{4} \wedge 4 < t \leq \frac{T}{2} \end{cases}$$

$$a_0 = \frac{1}{T} \int_{-T/4}^{T/4} 1 dt = \frac{1}{T} \left[\frac{T}{4} + \frac{T}{4} \right] = \frac{1}{2}$$

$$\begin{aligned}
 a_n &= \frac{2}{T} \int_{-T/4}^{T/4} \cos\left(\frac{2\pi n}{T}t\right) dt = \frac{2}{T} \left[\frac{T}{2\pi n} \sin\left(\frac{2\pi n}{T}t\right) \right]_{-T/4}^{T/4} = \frac{1}{\pi n} \left[2 \sin\left(\frac{\pi n}{2}\right) \right] \\
 &= \frac{2}{\pi}, 0, -\frac{2}{3\pi}, 0, \frac{2}{5\pi}, 0, \dots \quad \text{for } n = 1, 2, 3, 4, 5, 6, \dots \\
 b_n &= \frac{2}{T} \int_{-T/4}^{T/4} \sin\left(\frac{2\pi n}{T}t\right) dt = \frac{2}{T} \left[\frac{T}{2\pi n} - \cos\left(\frac{2\pi n}{T}t\right) \right]_{-T/4}^{T/4} = -\frac{1}{\pi n} \left[\cos\left(\frac{\pi n}{2}\right) - \cos\left(\frac{\pi n}{2}\right) \right] = 0
 \end{aligned}$$

Complex Notation:

It is more compact to express *Fourier* series in complex notation. Then

$$f(t) = \sum_{n=-\infty}^{\infty} c_n \exp\left(i \frac{2\pi n}{T} t\right) \quad [1.2.4]$$

$$f(t) = \sum_{n=-\infty}^{\infty} c_n \exp(i 2\pi n t) \quad [1.2.5]$$

$$f(t) = \sum_{n=-\infty}^{\infty} c_n \exp(i \omega n t) \quad [1.2.6]$$

$$\text{with } \sin(nt) = \frac{1}{2i} [e^{int} - e^{-int}] \text{ and } \cos(nt) = \frac{1}{2} [e^{int} + e^{-int}]$$

$$\text{hence: } c_n = \frac{1}{2} (a_n - ib_n) = \frac{1}{T} \int_{-T/2}^{T/2} f(t) \exp(-in\omega t) dt \quad [1.2.7]$$

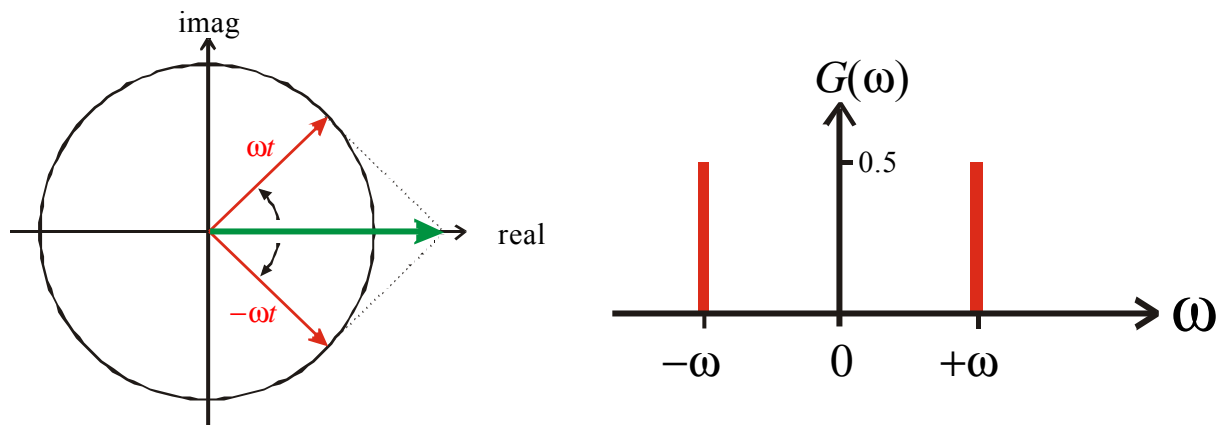
where n can be positive, negative and zero!

This expansion to negative n is a further generalisation. It can be understood in terms, that the complex exponential function is sensitive to the direction in which the function is moving when displayed on a complex circle (*Argand* diagram).

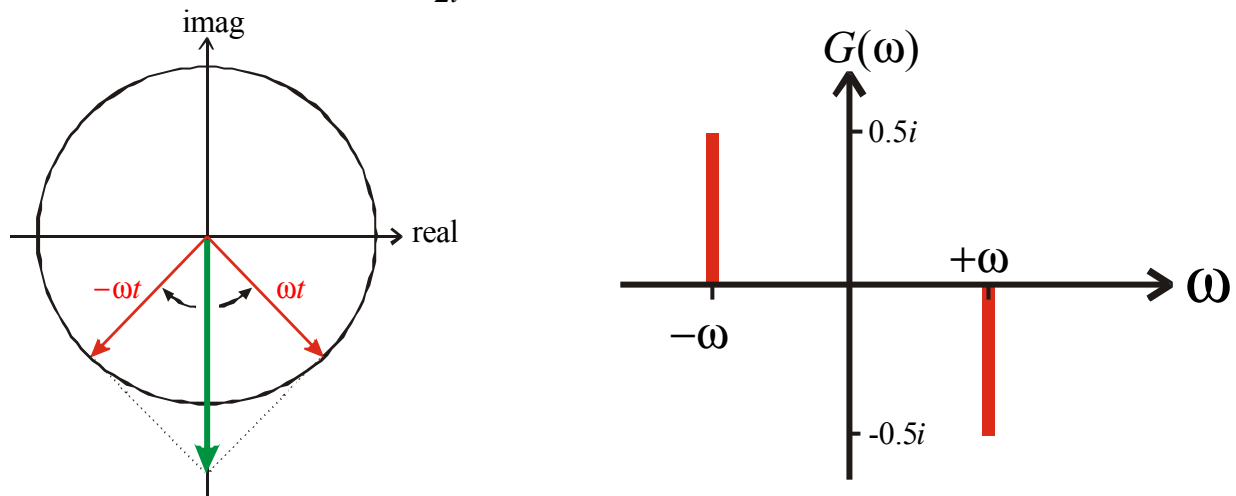
Because of $\cos(\omega t) = \frac{1}{2} [e^{i\omega t} + e^{-i\omega t}]$ we can think of two vectors in the *Argand* circle,

one rotating with positive $\omega = \frac{2\pi}{T}$ anticlockwise

and one rotating with negative $-\omega = -\frac{2\pi}{T}$ clockwise. Their sum however is real, as shown is the following sketch. On the right the spectrum is shown.



Analogously, we get for $\sin(\omega t) = \frac{1}{2i} [e^{i\omega t} - e^{-i\omega t}]$ an imaginary result.



We want to repeat the analysis for the square-wave in Fig. 1 now in complex notation:

$$\text{for one period } T : f(t) = \begin{cases} 1 & \text{for } -\frac{T}{4} < t \leq \frac{T}{4} \\ 0 & \text{for } -\frac{T}{2} \leq t < -\frac{T}{4} \wedge \frac{T}{4} < t \leq \frac{T}{2} \end{cases}$$

$$c_n = \frac{1}{T} \int_{-T/2}^{T/2} f(t) \exp(-in\omega t) dt = \frac{1}{T} \int_{-T/4}^{T/4} \exp(-in\omega t) dt = \frac{1}{T} \left[\frac{\exp(-in\omega t)}{-in\omega} \right]_{-T/4}^{T/4}$$

$$\text{with } \omega = \frac{2\pi}{T}$$

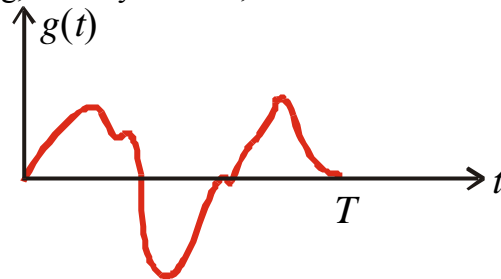
$$\begin{aligned} c_n &= \frac{i}{2\pi n} \left[\exp\left(-in\frac{\pi}{2}\right) - \exp\left(in\frac{\pi}{2}\right) \right] = \frac{i}{2\pi n} \left[\cos\left(n\frac{\pi}{2}\right) - i\sin\left(n\frac{\pi}{2}\right) - \cos\left(n\frac{\pi}{2}\right) - i\sin\left(n\frac{\pi}{2}\right) \right] \\ &= \frac{i}{2\pi n} - 2i\sin\left(n\frac{\pi}{2}\right) = \frac{1}{\pi n} \sin\left(n\frac{\pi}{2}\right) \quad \text{with } n = -\infty, \dots, -2, -1, 0, 1, 2, \dots, +\infty \end{aligned}$$

for $n=0$ we get zero/zero, so we try *L'Hospital's* rule: $c_0 = \lim_{n \rightarrow 0} c_n = \frac{\frac{\pi}{2} \cos\left(n\frac{\pi}{2}\right)}{\pi} = \frac{1}{2}$

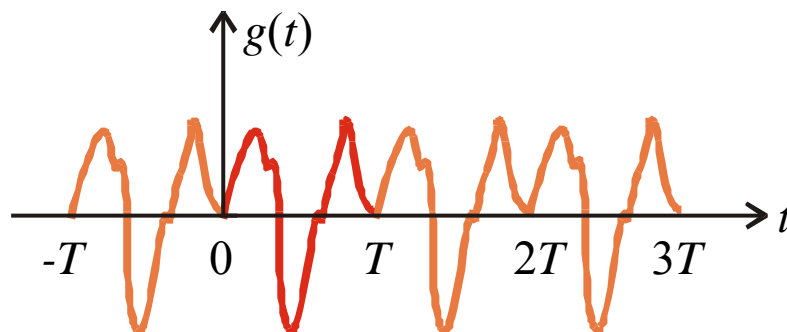
the rest of the c_n is half the value of the a_n as defined in eq. [1.2.7]

1.3. Fourier Transform

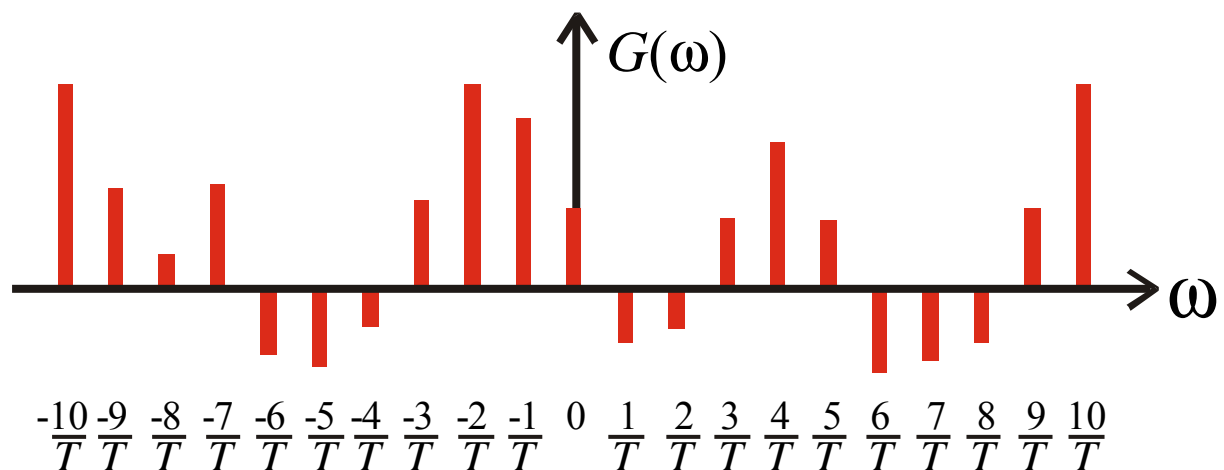
Consider now a non-repeating, arbitrary function, which has a duration equal or less to T .



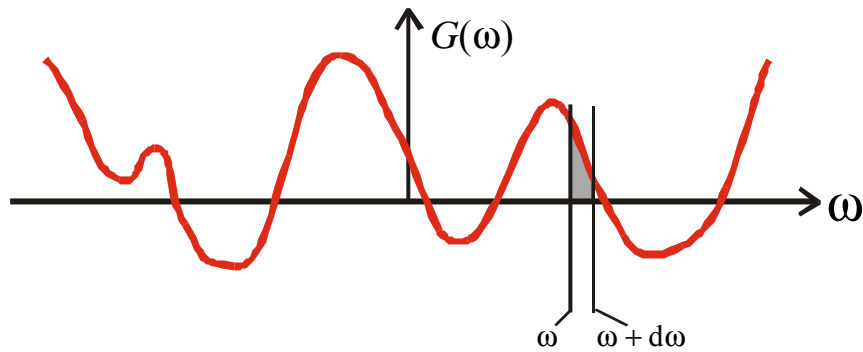
In order to calculate the *Fourier-coefficients* of this function $g(t)$, we *assume* that it repeats after a time T . Hence we can treat it as a *Fourier series*.



The distribution of *Fourier-coefficients*, the spectrum, might then look like the following sketch, note the intervals on the frequency axis. (Important: This graph is showing the real part of the spectrum only, we realise that we might get a somewhat different distribution for the imaginary part of the spectrum - however with the same abscissa).



In the limit $T \rightarrow \infty$ the spectrum becomes continuous. On the other hand that means that the function's abscissa is scaled to infinity, hence repeating doesn't matter anymore - just as a thought. Hence, any function -repetitive or not- can be represented as a superposition of sine/cosine functions. Then the summation is over a continuum of frequencies and the spectrum a continuous function $G(\omega)$ (see following sketch).



The original function $g(t)$ can now be reconstructed from the spectrum $G(\omega)$. Therefore, consider an interval between ω and $\omega + d\omega$ with an amplitude of $G(\omega)$. Thus $g(t)$ can be written as:

$$g(t) = \int_{-\infty}^{\infty} G(\omega) \exp(i\omega t) d\omega \quad [1.3.1]$$

Compare this to eq. [1.2.6]. The continuous sampling converts the sum into an integral and since n becomes continuous we can integrate over it.

Equation [1.3.1] is the definition of the INVERSE FOURIER TRANSFORM.

The forward action, which generates $G(\omega)$ from $g(t)$ is therefore:

$$\text{The FOURIER TRANSFORM: } G(\omega) = \int_{-\infty}^{\infty} g(t) \exp(-i\omega t) dt \quad [1.3.2]$$

However, these functions are not normalised yet. To generate $G(\omega)$ from $g(t)$ and from that again $g(t)$, we have to make sure that consecutive application of the transformations doesn't re-scale the function. We find a factor of $1/2\pi$ missing and distribute it symmetrically over both transforms. (However, normalisation doesn't really matter for this course).

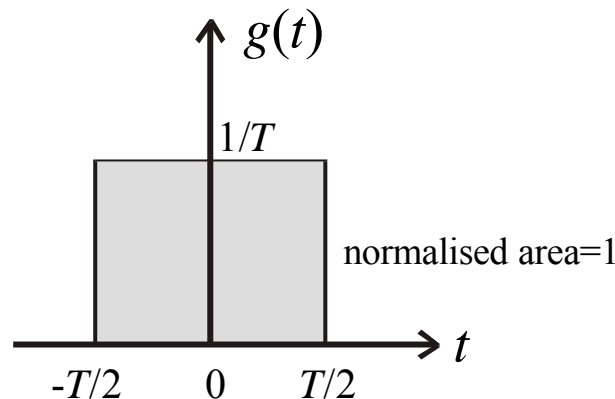
Fourier-Transform:

$$G(\omega) \equiv \mathcal{F} g(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} g(t) \exp(-i\omega t) dt \quad [1.3.3]$$

Inverse Fourier-Transform:

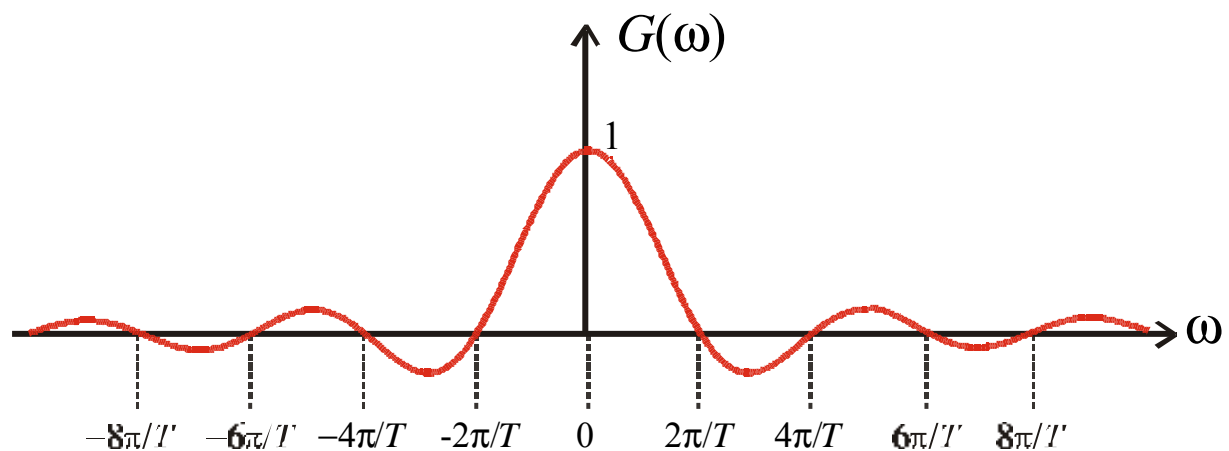
$$g(t) \equiv \overline{\mathcal{F}} G(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} G(\omega) \exp(i\omega t) d\omega \quad [1.3.4]$$

Example: Again the square wave from Fig. 1, now non-repetitive, but normalised... the so-called „Top-Hat“ function. It could be a short DC-(electrical) puls.



$$\begin{aligned}
 G(\omega) &= \int_{-\infty}^{\infty} g(t) \exp(-i\omega t) dt = \int_{-T/2}^{T/2} \frac{1}{T} \exp(-i\omega t) dt \\
 &= \frac{1}{T} \left[-\frac{\exp(-i\omega t)}{i\omega} \right]_{-T/2}^{T/2} = \frac{i}{T} \left[\frac{\exp(-i\omega \frac{T}{2}) - \exp(i\omega \frac{T}{2})}{\omega} \right] \\
 &= \frac{i}{T} \left[\frac{\cos(\omega \frac{T}{2}) - i \sin(\omega \frac{T}{2}) - \cos(\omega \frac{T}{2}) - i \sin(\omega \frac{T}{2})}{\omega} \right] = \frac{i}{T} \frac{-2i \sin(\omega \frac{T}{2})}{\omega} = \frac{2 \sin(\omega \frac{T}{2})}{\omega T} \\
 &= \frac{\sin(\omega \frac{T}{2})}{\omega \frac{T}{2}} \equiv \text{sinc}\left(\omega \frac{T}{2}\right)
 \end{aligned}$$

which is the so-called SINC-function ($\text{sinc}(x) = \sin(x)/x$) (Note that $\text{sinc}(0) = 1$, see example of *L'Hospital's* rule above).



The first roots are at $\omega_{\pm 1} = \pm \frac{2\pi}{T}$ with $G(\omega_{\pm 1}) = 0$. The sinc-function continues to oscillate towards infinity, however with a hyperbolic damped amplitude.

1.4. Theorems of the *Fourier Transform*

1.4.1 The Similarity Theorem (axis-scaling/sampling theorem for discrete data)

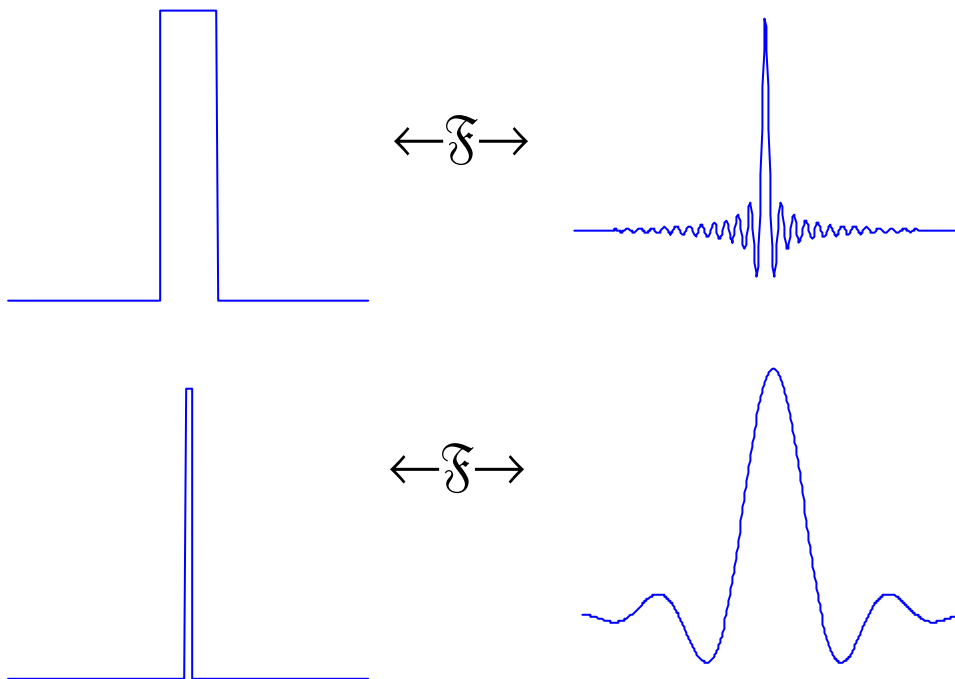
If $f(t)$ has the *Fourier transform* $F(\omega)$, then $f(at)$ has the *Fourier transform* $\frac{1}{a}F(\omega/a)$. [1.4.1]

The wider $f(t)$ the narrower $F(\omega)$ and vice versa.

Proof:
$$\mathcal{F}(f(at)) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(at) \exp(-i\omega t) dt \quad \text{substitute } \tau = at$$

$$\mathcal{F}(f(\tau)) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(\tau) \exp(-i\frac{\omega}{a}\tau) \frac{d\tau}{a} = \frac{1}{a} \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(\tau) \exp(-i\frac{\omega}{a}\tau) d\tau = \frac{1}{a} F(\omega/a)$$

Example:



A pulse of duration Δt contains a frequency spread $\Delta\omega$ such that

$$\Delta t \Delta\omega \approx 2\pi = \text{const.}$$

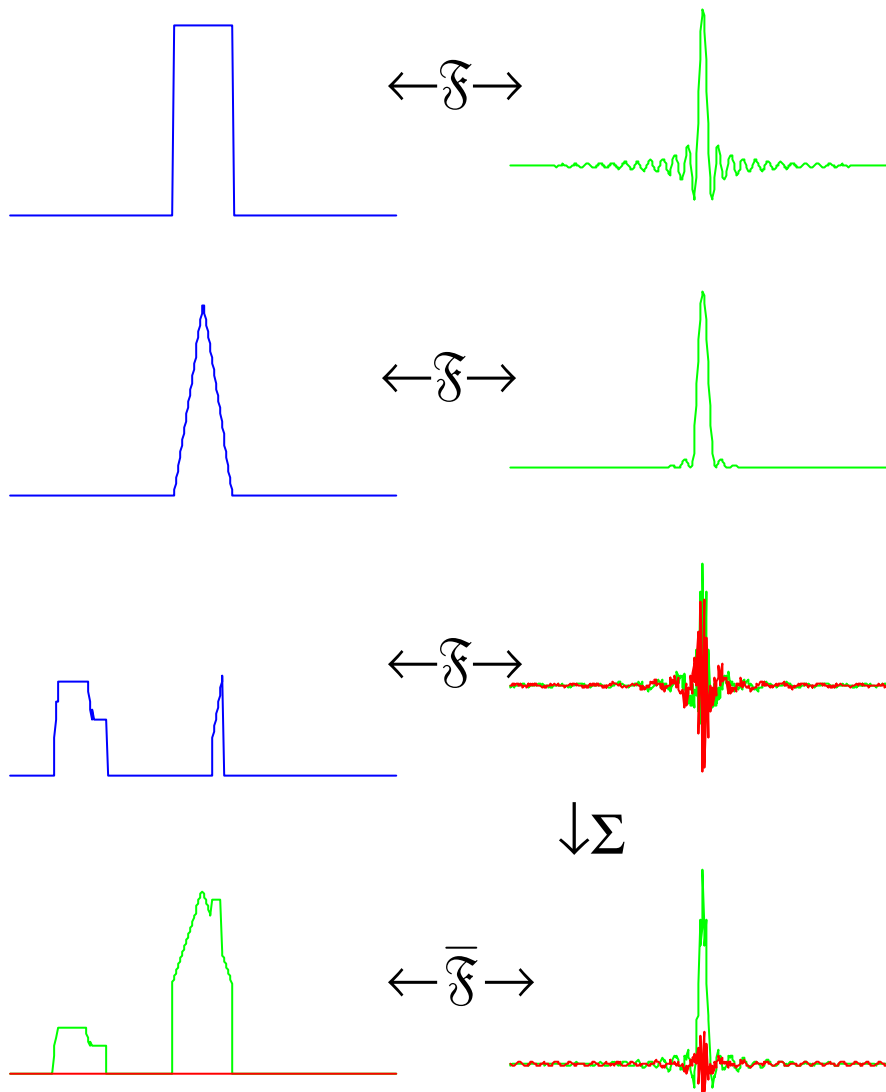
Hence a delta function transforms into a constant!

1.4.2 Addition Theorem

[1.4.2]

If $f(t)$ and $g(t)$ have the *Fourier transforms* $F(\omega)$ and $G(\omega)$ then $f(t) + g(t)$ has the *Fourier transform* $F(\omega) + G(\omega)$.

Proof: trivial

Example:

Object on the left. *Fourier* transform on the right (green: real part, red: imaginary part). Last row right sum of all the FTs. Inverse FT gives the image on the left (clearly the sum of the objects).

1.4.3 Shift Theorem

[1.4.3]

If $f(t)$ has the *Fourier* transforms $F(\omega)$, then $f(t-\tau)$ has the *Fourier* transform $\exp(-i\omega\tau)F(\omega)$.

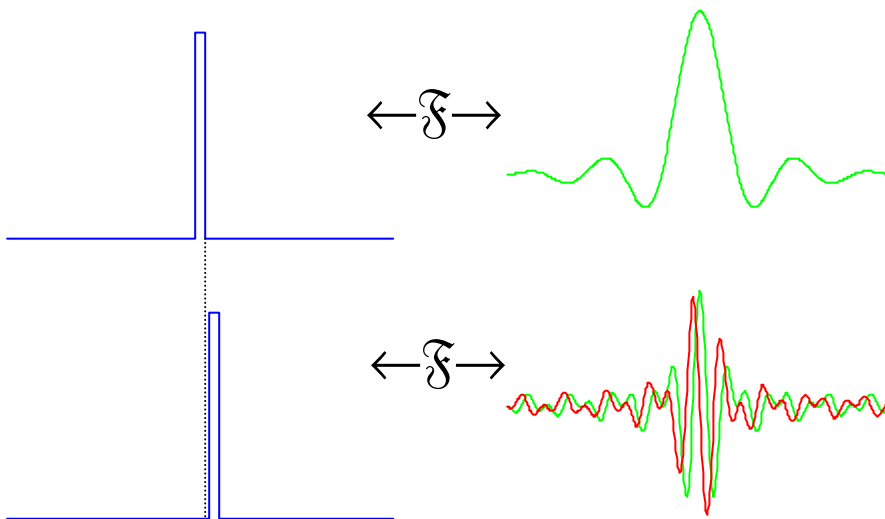
Shifting a function causes an additional phase in the *Fourier* transform.

Proof: $\mathfrak{F}(f(t-\tau)) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t-\tau) \exp(-i\omega t) dt$ substitute $\phi = t - \tau$

$$\mathfrak{F}(f(\phi)) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(\phi) \exp(-i\omega(\phi + \tau)) d\phi = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(\phi) \exp(-i\omega\tau) \exp(-i\omega\phi) d\phi$$

$$= \exp(-i\omega\tau) \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(\phi) \exp(-i\omega\phi) d\phi = \exp(-i\omega\tau) F(\omega)$$

Example:

**1.4.4 Parseval's Theorem (Rayleigh's Theorem)**

[1.4.4]

The integral of the square modulus of a function is equal to the square modulus of its spectrum:

$$\int_{-\infty}^{\infty} |f(t)|^2 dt = \int_{-\infty}^{\infty} f(t) f^*(t) dt = \int_{-\infty}^{\infty} |F(\omega)|^2 d\omega = \int_{-\infty}^{\infty} F(\omega) F^*(\omega) d\omega$$

Proof: needs convolution theorem (later)

Example: (as shown on page 10, where we had neglected the factor $\frac{1}{\sqrt{2\pi}}$ from the definition in eq. [1.3.3])

$$f(t) = \begin{cases} 1 & \text{for } -\frac{T}{2} < t \leq \frac{T}{2} \\ 0 & \text{everywhere else} \end{cases} \quad \text{and} \quad F(\omega) = \frac{1}{\sqrt{2\pi}} \frac{\sin\left(\omega \frac{T}{2}\right)}{\omega \frac{T}{2}} \equiv \frac{\text{sinc}\left(\omega \frac{T}{2}\right)}{\sqrt{2\pi}}$$

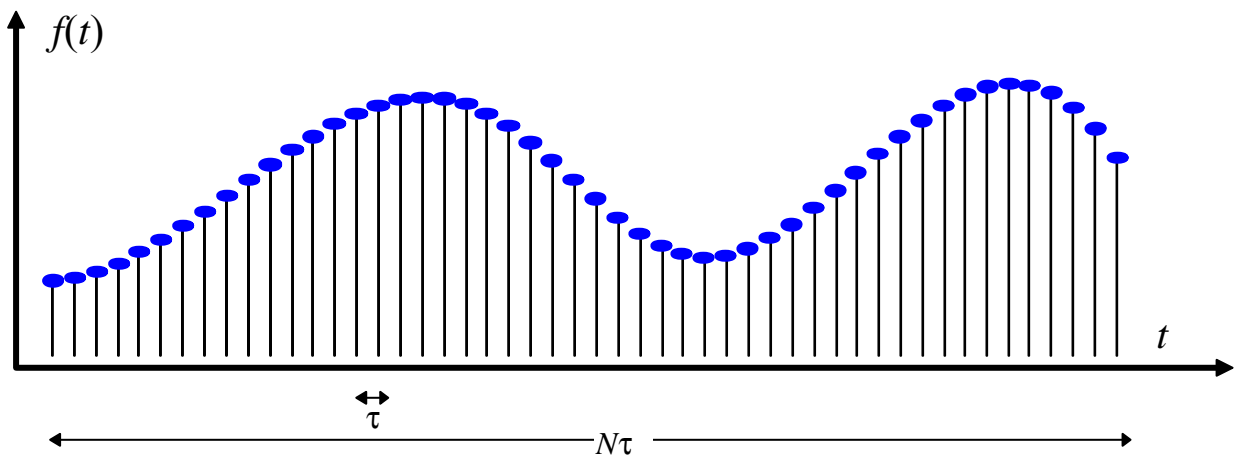
$$\int_{-\infty}^{\infty} |f(t)|^2 dt = \int_{-\infty}^{\infty} f(t) f^*(t) dt = \int_{-T/2}^{T/2} \frac{1}{T^2} dt = \frac{1}{T^2} [t]_{-T/2}^{T/2} = \frac{1}{T}$$

$$\int_{-\infty}^{\infty} |F(\omega)|^2 d\omega = \int_{-\infty}^{\infty} F(\omega) F^*(\omega) d\omega = \frac{1}{2\pi} \int_{-\infty}^{\infty} \frac{\sin^2\left(\omega \frac{T}{2}\right)}{\omega^2 \frac{T^2}{4}} d\omega$$

with $\int_{-\infty}^{\infty} \frac{\sin^2(a\omega)}{(a\omega)^2} d\omega = \frac{\pi}{a}$ hence $\int_{-\infty}^{\infty} |F(\omega)|^2 d\omega = \frac{1}{2\pi} \frac{\pi}{\frac{T}{2}} = \frac{1}{T}$

1.5. Discrete Fourier Transform

1.5.1 Definition



The function above consists of N points sampled at regular intervals τ . It can be assumed to be periodic as done in the definition of the *Fourier* transform but the time axis is now identical to $N\tau$ for which we can substitute using the similarity theorem in [1.4.1]. Hence,

$$f(N\tau) \text{ has the } \textit{Fourier} \text{ transform } \frac{1}{N} F(\omega/N)$$

where we have to replace the integral in the definition of the *Fourier* transform by a discrete sum, giving the following definition of the **discrete Fourier transform**:

$$\mathfrak{F}(f(\tau)) = F(\omega) = \frac{1}{N} \sum_{\tau=1}^N f(\tau) \exp\left(-i \frac{\omega}{N} \tau\right) \quad [1.5.1]$$

The quantity ω/N is analogous to frequency measured in cycles per sampling interval.

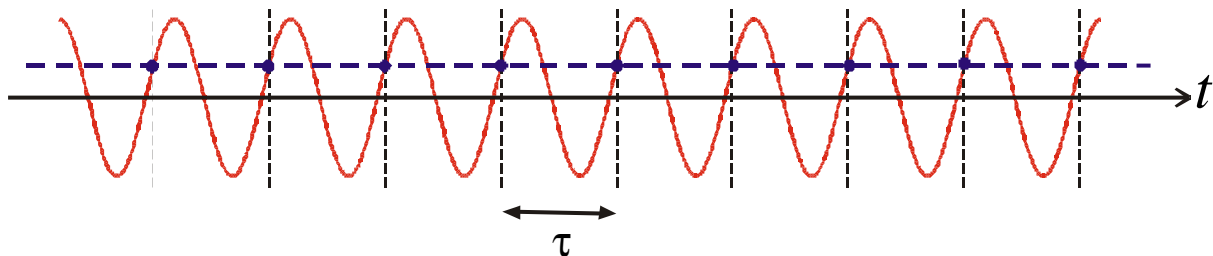
	time	frequency
continuous FT	t	ω
discrete FT	τ	$\frac{\omega}{N}$

Examples: see MATLAB workshop.

1.5.2 Nyquist's Theorem and Aliasing

[1.5.2]

Sampling can cause problems - it can lead to ambiguity in the apparent frequency of the sampled waveform. That is for instance the case when a sine wave has the same period as the sampling rate, τ or sampling frequency $\nu_S = 1/\tau$.



Although the sampled function is a sine, we obtain a **wrong** straight line. However, there is nothing we can do against it at a given sampling frequency.

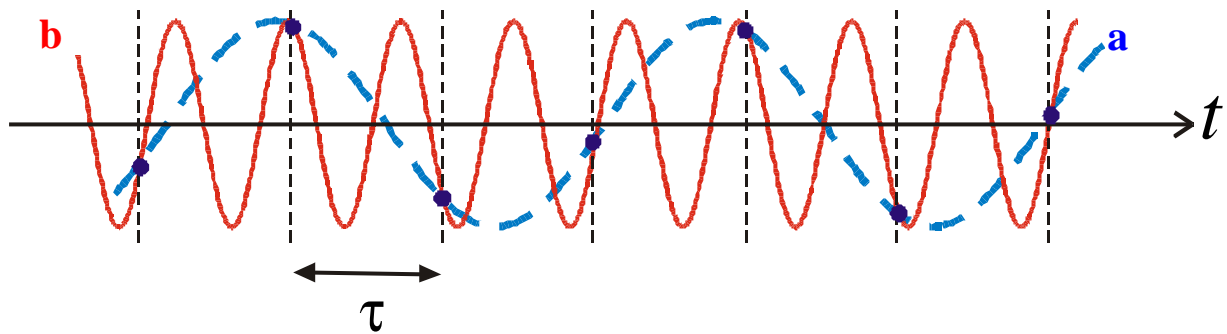
Nyquist's Theorem:

For a waveform to be sampled unambiguously, the waveform must not contain any frequency components greater than $\pm \frac{\nu_S}{2} = \pm \frac{1}{2\tau} = \pm \frac{\omega_S}{4\pi}$.

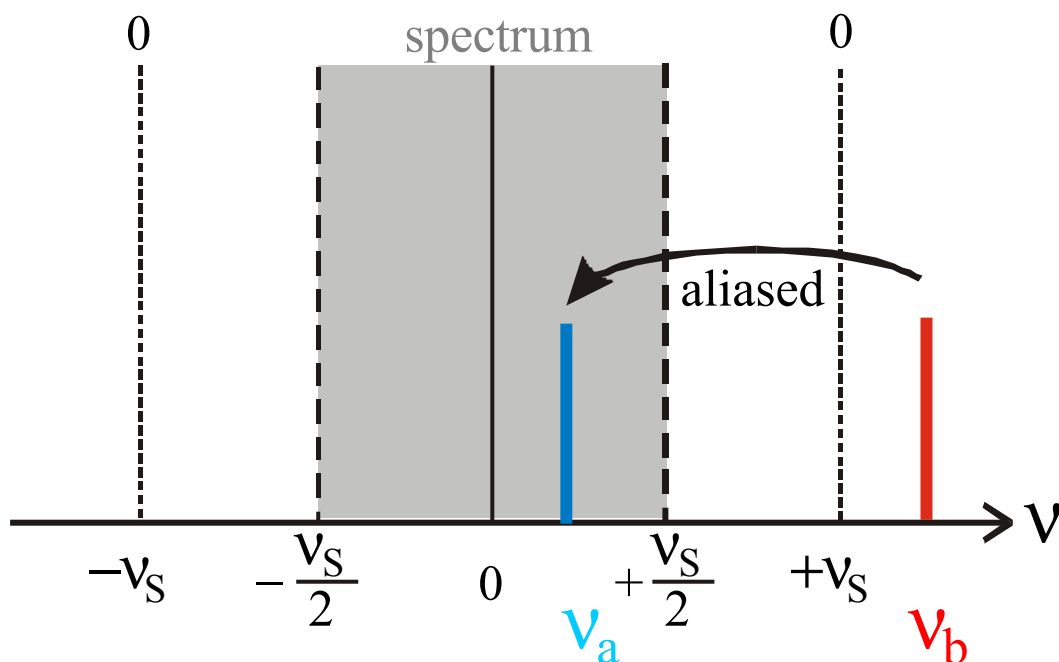
Or alternatively:

For a waveform to be sampled unambiguously, the sampling frequency must be at least twice that of the highest frequency component in the waveform.

Ambiguities cause effects which are called aliasing. This describes a misinterpretation of the measured frequencies having a change in sign.



Although the blue dashed waveform (a) is much lower than the red (b), they both result in the same sampled values, because waveform (b) is higher than half the sampling frequency. What happens to the spectrum is shown below.



From the figure above it becomes clear that:

$$\nu_b = n\nu_S + \nu_a$$

An identical effect is that in movies (24 frames per second) often wheels seem to turn backwards, because they rotate faster than 12 Hz / number of spokes.

Another example is the **recording of music CDs**. The analogue-to-digital conversion (ADC) is done at a frequency of 44kHz. Care must be taken that no frequencies (from microphones, instruments) which exceed half the sampling frequency -here $\pm 22\text{kHz}$ - are recorded. They might be inaudible in the studio, but the digitalisation process converts/aliases them to lower frequencies. E.g. a frequency of 46 kHz would replay as 2kHz - which will be distinctly audible.

To prevent this the audio signal is pre-processed by running through an analogue low-pass filter with a sharp cut-off at 22kHz.

Further examples: See MATLAB workshop.

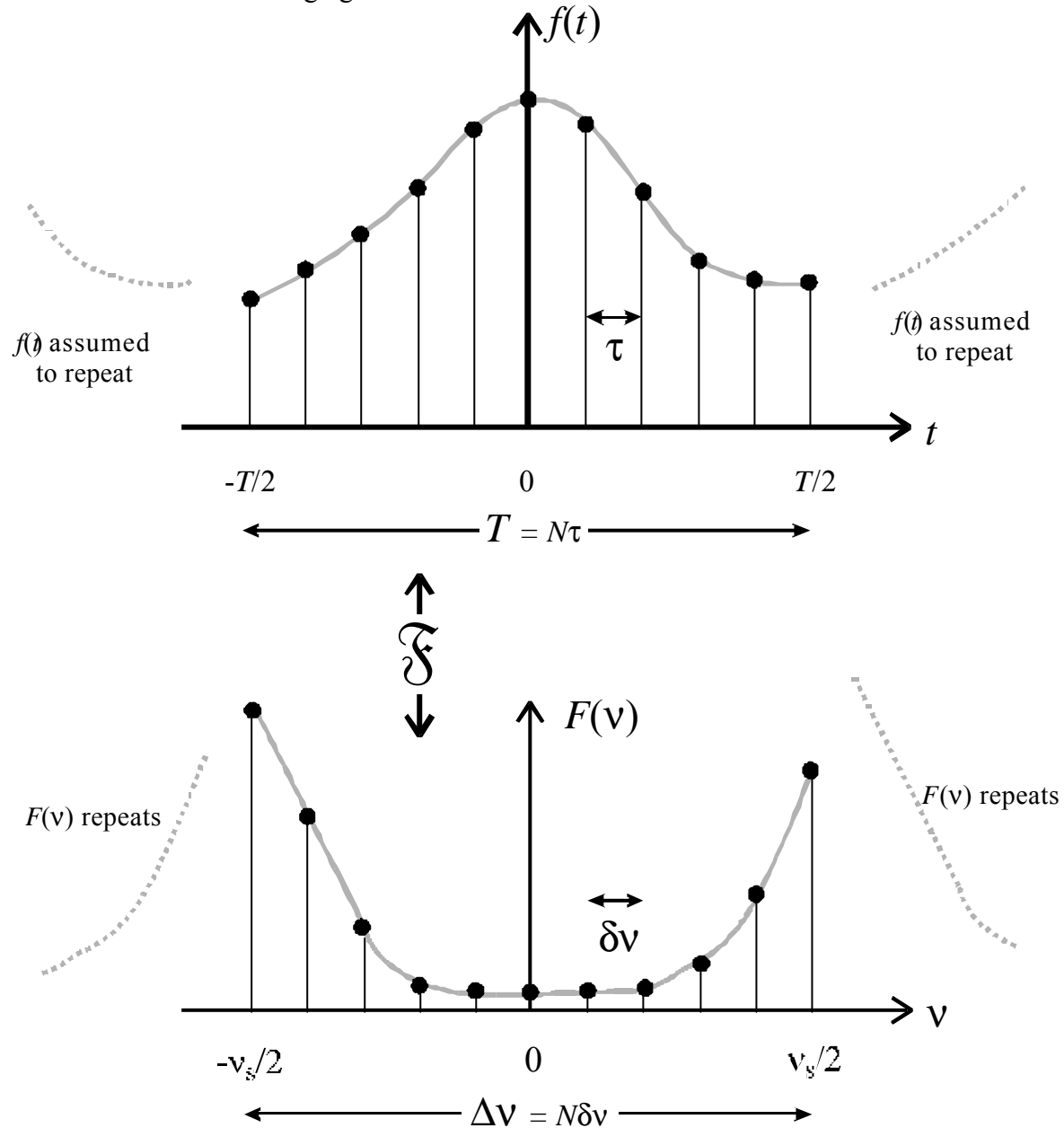
1.5.3 Properties of the Discrete Fourier Transform

Theorem:

[1.5.3]

If $f(t)$ is sampled N times in the time interval $-\frac{T}{2} \leq t \leq \frac{T}{2}$ then $F(v)$ will contain N discrete frequencies in the interval $\pm \frac{v_s}{2}$, with v_s as the sampling frequency.

As illustrated in the following figure:



where $v_s =$ sampling frequency in time domain $= \frac{1}{\tau}$
 $\delta v =$ sampling step in frequency domain $= \frac{1}{T}$
 $\Delta v =$ sampling interval in the frequency domain $= v_s$

Example:

A signal is sampled 1024 times in a millisecond ($T = 10^{-3}$ s), hence has a sampling interval of

$$\tau = \frac{10^{-3} \text{ s}}{1024} = 0.977 \cdot 10^{-6} = 977 \text{ ns} .$$

The sampling frequency is $\nu_S = \frac{1}{\tau} = \frac{1024}{10^{-3} \text{ s}} = 1.024 \text{ MHz}$

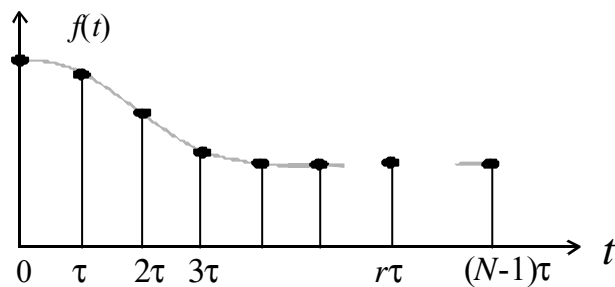
The spectrum $F(\nu)$ (after discrete FT) then consists of N points which are spaced in multiples of

$$\frac{1}{T} = \frac{1}{N\tau} = \frac{\nu_S}{N} = 10^3 \text{ Hz} = 1 \text{ kHz}$$

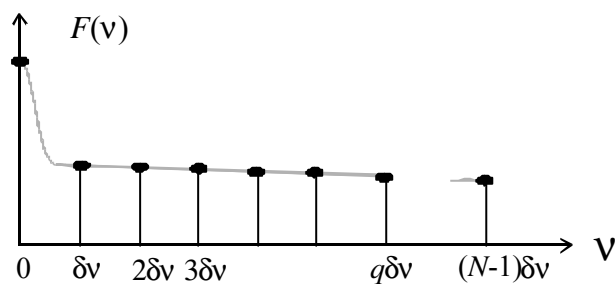
1.5.4 Calculation of the DFT

For simplicity we only use positive times/frequencies:

For N sampled points:



samples are at
 $t = r\tau$ with $0 \leq r \leq (N-1)$
 where $\tau = \frac{T}{N}$



after DFT samples are at
 $\nu = 0, \delta\nu, 2\delta\nu, \dots, (N-1)\delta\nu$
 with $\delta\nu = \frac{1}{T}$
 $\nu = q\delta\nu$ with $0 \leq q \leq (N-1)$
 and $N\delta\nu = \nu_S = \frac{1}{\tau}$

From eq. [1.5.1] the DFT is defined as:

$$\mathfrak{F}(f(\tau)) = F(\omega) = \frac{1}{N} \sum_{\tau=1}^N f(\tau) \exp\left(-i \frac{\omega}{N} \tau\right)$$

or with the new definitions

$$F(q\delta\nu) = \frac{1}{N} \sum_{r=0}^{N-1} f(r\tau) \exp\left(-\frac{2\pi i}{N} q\delta\nu r\tau\right) \quad [1.5.4]$$

$$\text{or } F_q = \frac{1}{N} \sum_{r=0}^{N-1} f_r W^{rq} \quad [1.5.5]$$

$$\text{with } W \equiv \exp\left(-\frac{2\pi i}{N}\right), \quad F_q \equiv F(q\delta v) \quad \text{and } f_r \equiv f(r\tau)$$

then the inverse DFT is

$$f_r = \sum_{q=0}^{N-1} F_q W^{-rq} \quad [1.5.6]$$

Example: 4-point DFT

some function complex g_r $g_0 = 1, g_1 = 1+i, g_2 = 0, g_3 = 1-i$

hence $W = \exp\left(-\frac{2\pi i}{4}\right) = \exp\left(-\frac{\pi i}{2}\right) = -i$

and $W^0 = 1, W^1 = -i, W^2 = -1, W^3 = i, W^4 = 1, \dots$

calculation of $G_q = \frac{1}{4} \sum_{r=0}^3 g_r W^{rq}$ for $q = 0, 1, 2, 3$:

$$\text{for } q=0: G_0 = \frac{1}{4} \sum_{r=0}^3 g_r W^0 = \frac{1}{4} [g_0 + g_1 + g_2 + g_3] \cdot 1 = \frac{1}{4} [1 + 1 + i + 1 - i] = \frac{3}{4}$$

$$\text{for } q=1: G_1 = \frac{1}{4} \sum_{r=0}^3 g_r W^r = \frac{1}{4} [g_0 W^0 + g_1 W^1 + g_2 W^2 + g_3 W^3] = \frac{1}{4} [1 - i + 1 + 0 + i + 1] = \frac{3}{4}$$

$$\text{for } q=2: G_2 = \frac{1}{4} \sum_{r=0}^3 g_r W^{2r} = \frac{1}{4} [g_0 W^0 + g_1 W^2 + g_2 W^4 + g_3 W^6] = -\frac{1}{4}$$

$$\text{for } q=3: G_3 = \frac{1}{4} \sum_{r=0}^3 g_r W^{3r} = \frac{1}{4} [g_0 W^0 + g_1 W^3 + g_2 W^6 + g_3 W^9] = -\frac{1}{4}$$

$$\text{hence } G = \frac{1}{4} [3, 3, -1, -1]$$

We realise that we had to do 4×4 multiplications for the DFT, generally it would be $N \times N$ multiplications.

1.5.5 The Fast *Fourier* Transform Algorithm (FFT) - optional

Since $N \times N$ multiplications are needed for a direct DFT the algorithm needs to evaluate an $N \times N$ matrix. However the calculations required for the DFT can be organised in such a way that only $N \log(N)$ multiplications are needed. This speeds up the calculation dramatically. E. g. for $N=128$ the

FFT speed advantage over normal calculation is $\frac{N^2}{N \log N} = \frac{128}{7} \approx 18$

However the algorithm is only applicable if N is a power of 2.

You can find more details about the FFT algorithm at:

- <http://www.intersrv.com/~dcross/fft.html#section3>
- http://www.ulib.org/webRoot/Books/Numerical_Recipes/bookcpdf/c12-2.pdf

1.6 Symmetry and Fourier-Transforms

Let the function $f(t)$ be a sum of an even function $E(t)$ and an odd function $O(t)$ (both complex):

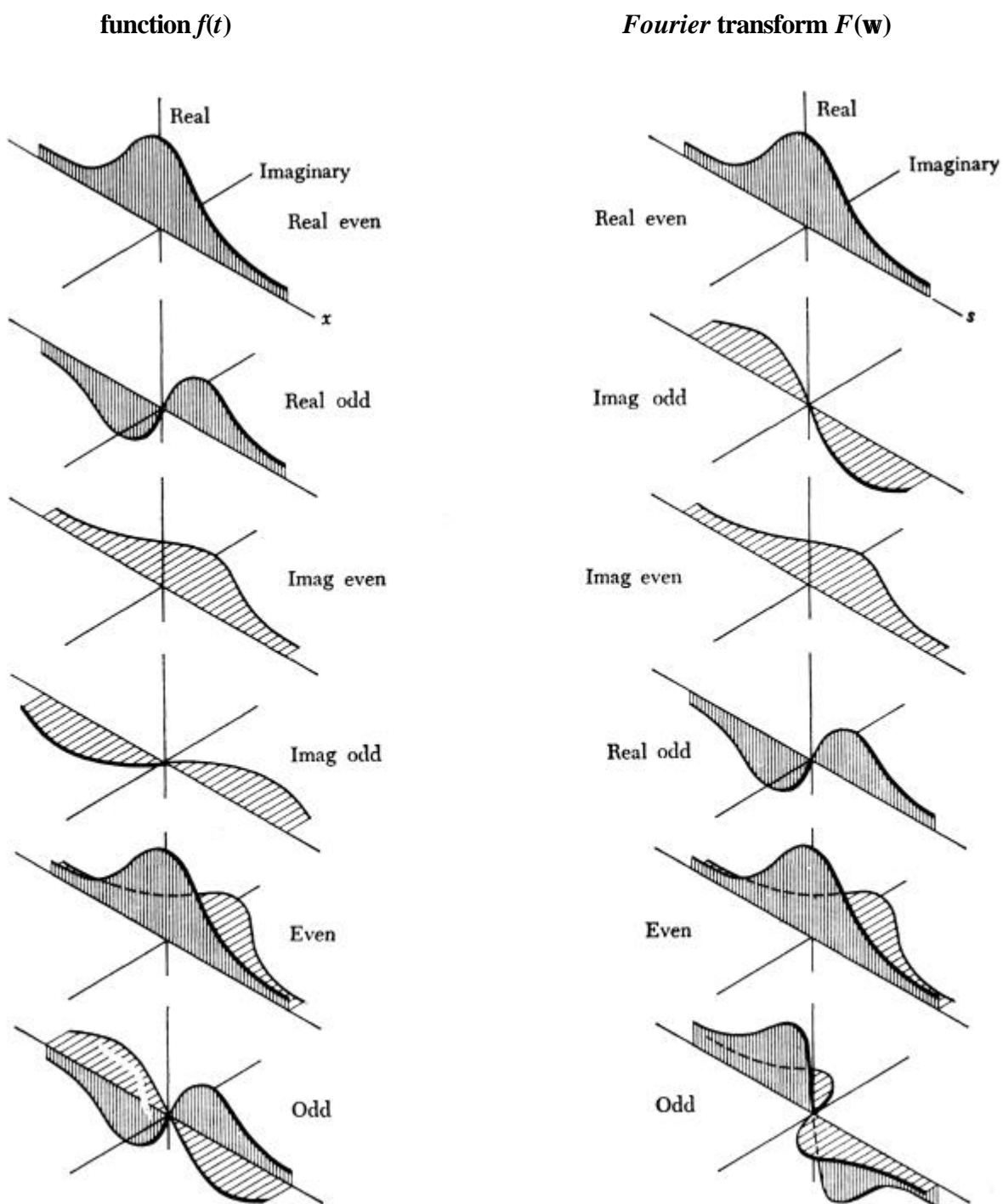
$$f(t) = E(t) + O(t)$$

Then the *Fourier* transformation simplifies to:

$$F(\omega) = 2 \int_0^{\infty} E(t) \cos(\omega t) dt - 2i \int_0^{\infty} O(t) \sin(\omega t) dt$$

Hence the following symmetry relations hold (see also following figure)

function $f(t)$	<i>Fourier</i> transform $F(\omega)$
real and even	real and even
real and odd	imaginary and odd
imaginary and even	imaginary and even
imaginary and odd	real and odd
(complex and) even	(complex and) even
(complex and) odd	(complex and) odd
real even and imaginary odd	real
real odd and imaginary even	imaginary



further examples: see MATLAB workshop

2. Convolution/Correlation

2.1. Convolution

2.1.1 Definition:

The convolution (symbol: \otimes) of two functions $f(t)$ and $g(t)$ is defined as:

$$h(t) = f(t) \otimes g(t) \equiv \int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau \quad [2.1.1]$$

Assuming that the convolution integral exists:

$$\text{Convolution is commutative:} \quad f \otimes g = g \otimes f \quad [2.1.2]$$

$$\text{Convolution is associative:} \quad f \otimes (g \otimes h) = (f \otimes g) \otimes h \quad [2.1.3]$$

$$\text{Convolution is distributive:} \quad f \otimes (g + h) = f \otimes g + f \otimes h \quad [2.1.4]$$

2.1.2 The Fourier-Theorem of convolution (convolution theorem): [2.1.5]

If $f(t)$ has the Fourier-transform $F(\omega)$ and $g(t)$ has the Fourier-transform $G(\omega)$, then $h(t) = f(t) \otimes g(t)$ has the Fourier-transform $H(\omega) = F(\omega)G(\omega)$.

In other words: The transform of a convolution is the product of the transforms.

or: The convolution of two functions is the transform of the product of their (the functions) transforms.

$$h(t) = f(t) \otimes g(t) \equiv \int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau = \mathcal{F}^{-1} \left[\mathcal{F}(f(t)) \cdot \mathcal{F}(g(t)) \right] \quad [2.1.6]$$

Proof (only for interest here): Neglecting scaling/normalisation!

$$H(\omega) \stackrel{\substack{\text{FT definition} \\ [1.3.3]}}{=} \int_{-\infty}^{\infty} h(t) \exp(-i\omega t) dt \stackrel{\substack{\text{definition} \\ [2.1.1]}}{=} \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau \right] \exp(-i\omega t) dt$$

Change the order of integration:

$$H(\omega) = \int_{-\infty}^{\infty} f(\tau) \left[\int_{-\infty}^{\infty} g(t - \tau) \exp(-i\omega t) dt \right] d\tau$$

Substitute $\vartheta = t - \tau$, where τ is a constant in the inner integral (hence the infinite integration limits are not changed):

$$H(\omega) = \int_{-\infty}^{\infty} f(\tau) \left[\int_{-\infty}^{\infty} g(\vartheta) \exp(-i\omega(\vartheta + \tau)) d\vartheta \right] d\tau = \int_{-\infty}^{\infty} f(\tau) \exp(-i\omega\tau) \left[\int_{-\infty}^{\infty} g(\vartheta) \exp(-i\omega\vartheta) d\vartheta \right] d\tau$$

The latter is the definition of the FT in [1.3.3], hence

$$H(\omega) = \int_{-\infty}^{\infty} f(\tau) \exp(-i\omega\tau) G(\omega) d\tau = G(\omega) \int_{-\infty}^{\infty} f(\tau) \exp(-i\omega\tau) d\tau = G(\omega) \cdot F(\omega) \text{ qed!}$$

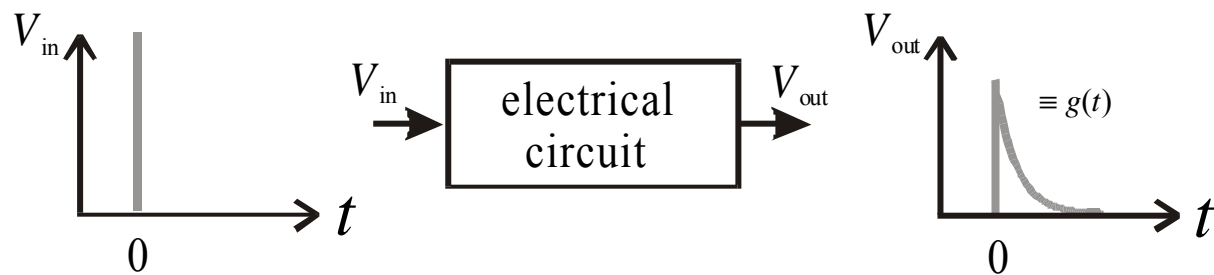
Hence the **recipe** to calculate the convolution $h(t) = f(t) \otimes g(t)$ with FT is:

- calculate the FT of $f(t) \Rightarrow F(\omega) = \mathfrak{F}(f(t))$
- calculate the FT of $g(t) \Rightarrow G(\omega) = \mathfrak{F}(g(t))$
- multiply $F(\omega)$ and $G(\omega) \Rightarrow H(\omega) = F(\omega)G(\omega)$
- take the inverse FT to get $h(t) \Rightarrow h(t) = \mathfrak{F}^{-1}(H(\omega))$

2.1.3 Examples:

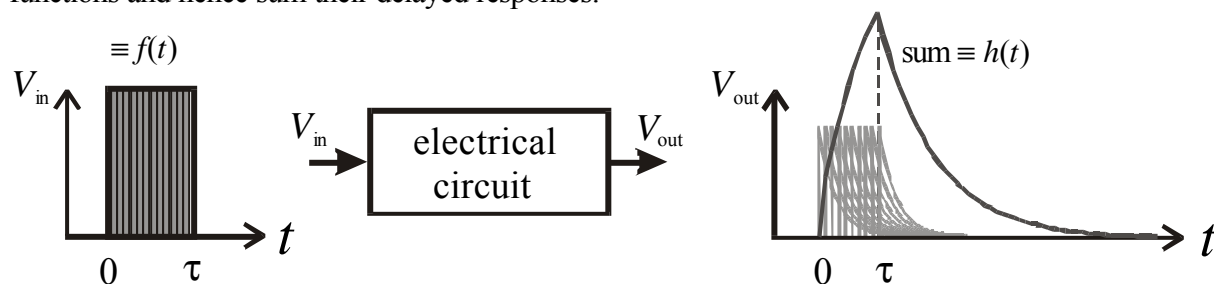
Example 1: Electrical circuit - Plausibility of convolution

Consider an electrical circuit which responds on an input-signal of a sharp impulse (idealised a delta-function) by an exponential decay. The electrical circuit could be an LC-circuit, but we can treat it here like a black-box.



Now what happens to the response if the input signal is of finite duration, τ ?

This can be answered by dividing the input signal into an (infinite) sequence of (delayed) delta-functions and hence sum their delayed responses:



The output is the convolution of $V_{in}(t)$ with the response to the delta-function.

$$h(t) = f(t) \otimes g(t) \equiv \int_{-\infty}^{\infty} f(t') g(t-t') dt'$$

‘sum’

input

delayed response

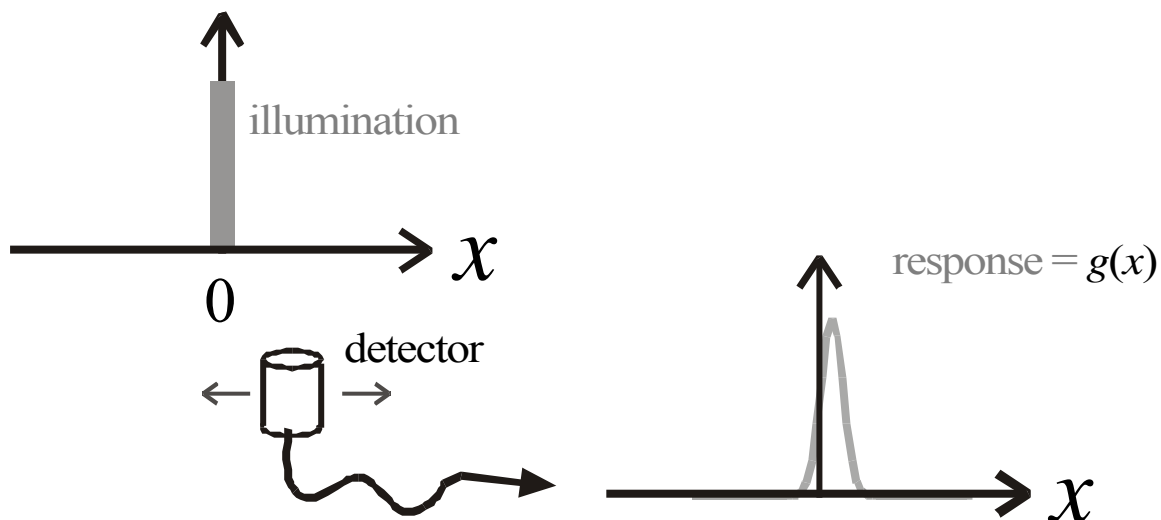
Arrows point from the labels 'sum', 'input', and 'delayed response' to the corresponding parts of the integral equation above.

A more general example would then be, to consider an arbitrary (continuous) function, $f(t)$, as the input. This function still can be subdivided into (an infinite number of) 'delta'-functions of heights given by the function. The output will be then the convolution of the function with the response-function, $g(t)$.

The input is 'spread' or 'smeared' out by the **impulse-response function**, $g(t)$. Therefore, $g(t)$ is also called the **point-spread function**!

Example 2: spatial example (detector) - plausibility of 'point-spread function'

Consider an experiment in which a (photo-) detector of **finite size** is moved past a very narrow slit giving a 'delta-function' of illumination at $x = 0$. The response of the detector, $g(x)$, would be at a maximum near to, but not necessary equal to $x = 0$ (because of imperfections and misalignments). There will be also significant response from the detector at some distance either side of $x = 0$, because it has a finite size.



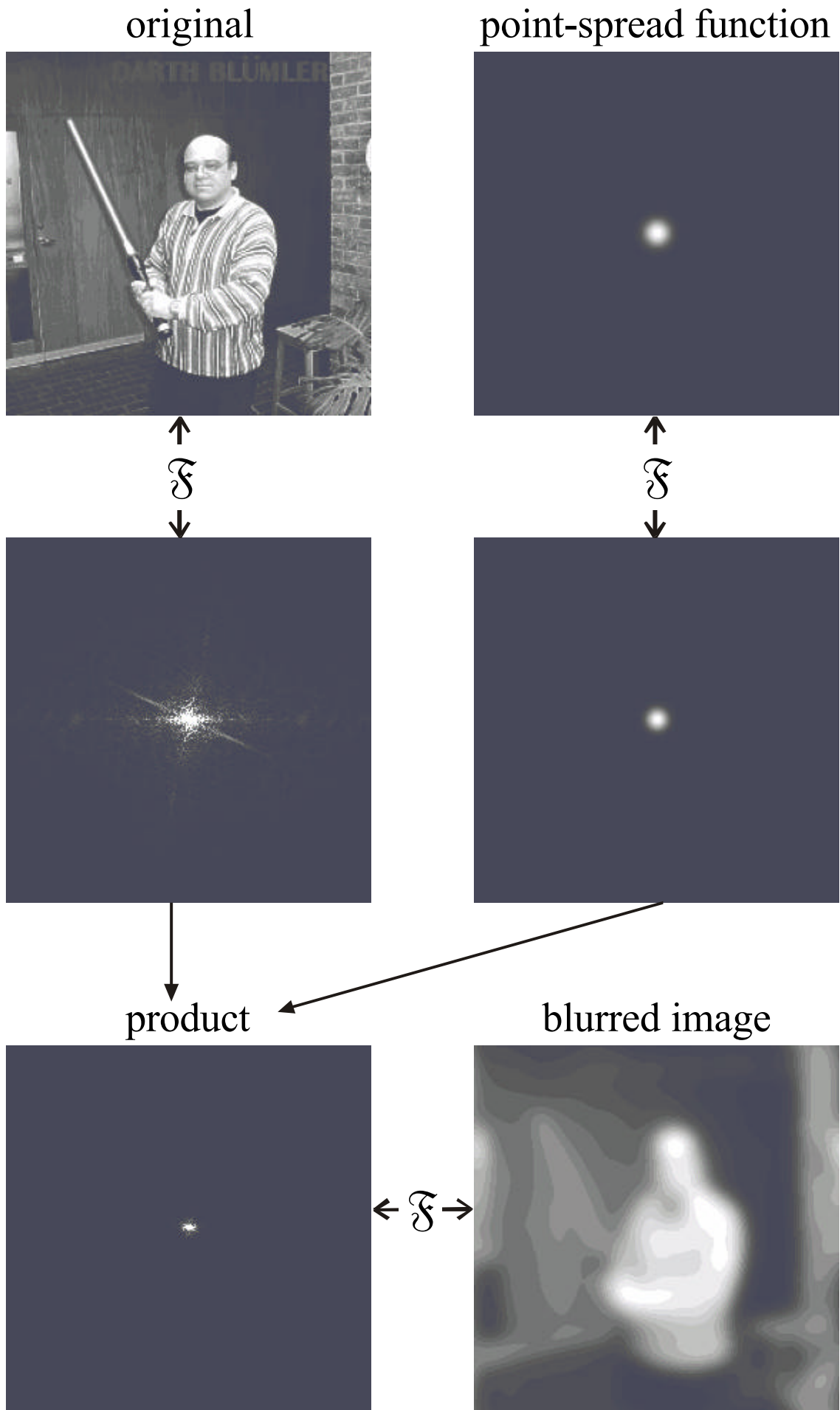
$g(x)$ is the impulse-response or point (of light)-spread function.

An extended illumination $f(x)$ will therefore cause an output of the detector which is additionally smeared out or blurred by the 'point-spread function', $g(x)$ it will be the convolution of the two, because each scanned point of $f(x)$ will be burred into neighbouring points by $g(x)$.

Example 3: out-of focus camera (we will later learn how to calculate a two-dimensional FT)

Each pixel of the original (sharp) photograph is then blurred by the out-of-focus response of the camera (point-spread function) of a single pixel. The resulting out-of-focus picture is a result of the convolution of the sharp scene with the point-spread function.... which is the overlap (sum/integral) of the point-spread function shifted to each position of the original sharp picture and scaled by its gray-scale value.

This process is used to soften images (cf. identical procedure by softening kernels in first part of lecture).



If we could invert this ‘blurring’ process by convolution, we could reconstruct the sharp (hidden) image from the blurred result. This process is called **deconvolution** and is an active field of research. (example: sharpening the first pictures from Hubble-space telescope... before it got its correction lenses).

We will later learn how this can be done and what problems are involved in the process.

Example 4: Maths (problem week 12)

The convolution of two ‘top-hat’ functions gives a triangle (video).

The FT of a ‘top-hat’ function is a sinc-function. Hence the FT of a triangle is a $(\text{sinc})^2$.

2.1.4 Discrete Convolution:

Definition:

The discrete version of eq. [2.1.1] for a discrete vector \mathbf{x} and a vector \mathbf{y} (with n components) is:

$$\mathbf{z} = \mathbf{x} \otimes \mathbf{y}, \quad z_i = \sum_{j=0}^{n-1} x_{i \leftrightarrow j} y_j \quad [2.1.7]$$

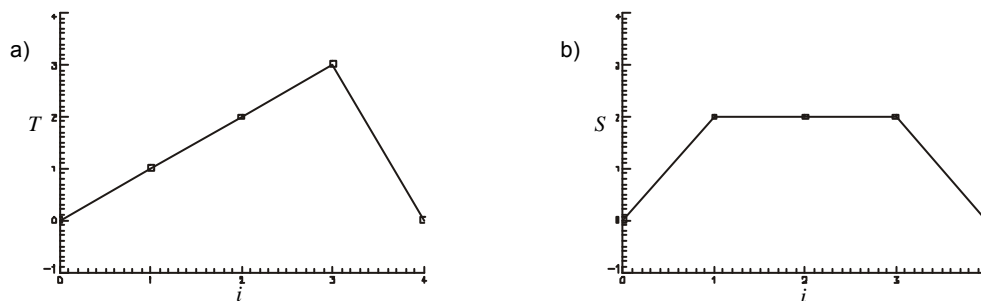
The symbol over the indices means, that they have to be treated cyclically, because negative indices are not defined. This corresponds to (see also table below):

$$i \leftrightarrow j \equiv (i - j + n - 1) \bmod n \quad (12.3)$$

In German convolution is also known as *Faltung* (folding) which is -maybe- more illustrative, because the procedure is like shifting a mirrored (or ‘folded’) function over the other and determine the overlapping areas.

Example: Convolution of a square with a triangle. Represented by two 5 point vectors \mathbf{T} and \mathbf{S} .

$$\mathbf{T} = (0,1,2,3,0) \quad \text{and} \quad \mathbf{S} = (0,2,2,2,0)$$



a) vector \mathbf{T} of an asymmetric triangle, b) vector \mathbf{S} of a „square“.

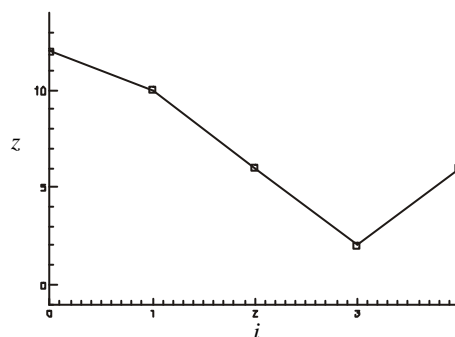
The following table is to illustrate the cyclic indices for $i = j = 0 \dots 4$:

$$\begin{array}{c|ccccc} \begin{array}{c} \leftrightarrow \\ i-j \\ i \end{array} & j=0 & j=1 & j=2 & j=3 & j=4 \\ \hline 0 & 4 & 3 & 2 & 1 & 0 \\ 1 & 0 & 4 & 3 & 2 & 1 \\ 2 & 1 & 0 & 4 & 3 & 2 \\ 3 & 2 & 1 & 0 & 4 & 3 \\ 4 & 3 & 2 & 1 & 0 & 4 \end{array}$$

Manual 'point by point' calculation:

i	T_{ij}					$T_{ij} S_j$					$z_i = \sum T_{ij} S_j$
	$j=0$	1	2	3	4	0	1	2	3	4	
						$S_j =$					
0	0	3	2	1	0	0	6	4	2	0	12
1	0	0	3	2	1	0	0	6	4	0	10
2	1	0	0	3	2	0	0	0	6	0	6
3	2	1	0	0	3	0	2	0	0	0	2
4	3	2	1	0	0	0	4	2	0	0	6

or



The result of $T \otimes S = z$

This example shows again how necessary -not to say paramount- computers are for such problems. Just envision performing this task for 2 vectors of 100 components!!!

Further examples: MATLAB workshop!

2.2 Correlation

2.2.1 Definition:

The correlation (sign \star) of two continuous functions $f(t)$ and $g(t)$ is defined by:

$$f(t) \star g(t) = \int_{-\infty}^{\infty} f^*(\tau)g(t + \tau) d\tau \quad [2.2.1]$$

for $f(t) = g(t)$ this is called '**autocorrelation**'

for $f(t) \neq g(t)$ this is also called '*cross correlation*'

In image processing one usually doesn't work with complex functions, however care has to be taken when Fourier-transforms are applied prior to correlation, because the result of a Fourier-transform is generally complex. So for most cases we can ignore the complex conjugate in [2.2.1].

Then the difference to convolution (see definition in eq. [2.1.1]) is that correlation doesn't fold or mirror one of the functions but simply slides it over the other function. The integration in both cases can be understood as seeking for the individual overlap for each argument of t . (see figure next page)

2.2.2 Correlation theorem:

Since $\mathcal{F}(f^*(t)) = F^*(-\omega)$ we can simply restate the convolution theorem of [2.1.5]

If $f(t)$ has the *Fourier*-transform $F(\omega)$ and $g(t)$ has the *Fourier*-transform $G(\omega)$, then $h(t) = f(t) \star g(t)$ has the *Fourier*-transform $H(-\omega) = F^*(\omega)G(\omega)$. [2.2.2]

However this is of minor importance for the course. Cross-correlations are used in image processing for the analysis of templates or prototype matching to find the closest match between sets of images (refer to Gonzales chapter 9).

We are more interested in the features of the properties and usage of the **autocorrelation function**, which is very important rather in *signal-* than *image-processing*.

2.2.3 The autocorrelation theorem:

If $f(t)$ has the *Fourier*-transform $F(\omega)$, then its autocorrelation function

$$f(t) \star f(t) = \int_{-\infty}^{\infty} f^*(\tau)f(t + \tau) d\tau \quad \text{has the } \textit{Fourier}\text{-transform } |F(\omega)|^2. \quad [2.2.3]$$

In other words: In order to calculate the autocorrelation function of $f(t)$,

- a) we take its FT $\Rightarrow F(\omega)$
- b) take its magnitude $\Rightarrow |F(\omega)|$
- c) and square it $\Rightarrow |F(\omega)|^2$
- d) and swap domain by doing the inverse FT $\Rightarrow f(t) \star f(t)$

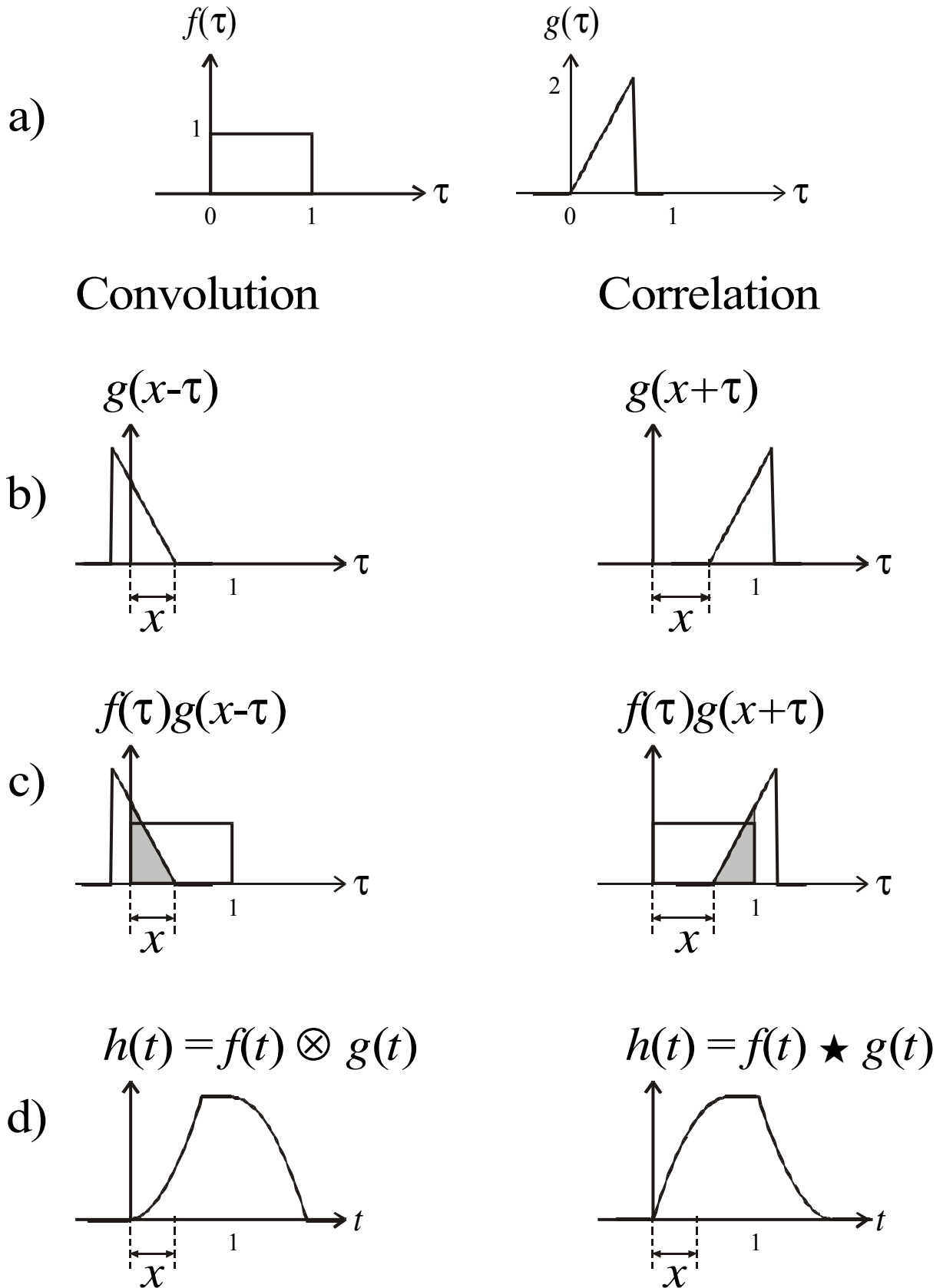


Illustration of convolution and correlation of two real 1D-functions:

- a) the two functions $f(t)$ and $g(t)$, b) the manipulations on $g(t)$ by the operation,
- c) the integration/sliding step, d) the result.

Proof of the autocorrelation theorem (only for interest here): Neglecting scaling/normalisation

For that we want to show that the iFT of $|F(\omega)|^2$ is identical to the definition of $f(t) \star f(t)$.

$$\mathfrak{F}^{-1}\left(|F(\omega)|^2\right) = \int_{-\infty}^{\infty} |F(\omega)|^2 \exp(i\omega t) d\omega = \int_{-\infty}^{\infty} F^*(\omega) F(\omega) \exp(i\omega t) d\omega$$

now we apply the convolution theorem in the form of [2.1.6]:

$$\mathfrak{F}^{-1}\left(|F(\omega)|^2\right) = \mathfrak{F}^{-1}\left[\mathfrak{F}\left(F^*(\omega)\right)\mathfrak{F}\left(F(\omega)\right)\right] = f^*(-t) \otimes f(t)$$

which is by definition [2.1.1]

$$\mathfrak{F}^{-1}\left(|F(\omega)|^2\right) = \int_{-\infty}^{\infty} f^*(-(t-\tau)) f(\tau) d\tau = \int_{-\infty}^{\infty} f^*(\tau-t) f(\tau) d\tau$$

Substitute $\vartheta = \tau - t$

$$\mathfrak{F}^{-1}\left(|F(\omega)|^2\right) = \int_{-\infty}^{\infty} f^*(\vartheta) f(t+\vartheta) d\vartheta = f(t) \star f(t) \quad \text{qed!}$$

As we see the autocorrelation theorem is a special case of the convolution theorem!

What is it good for?

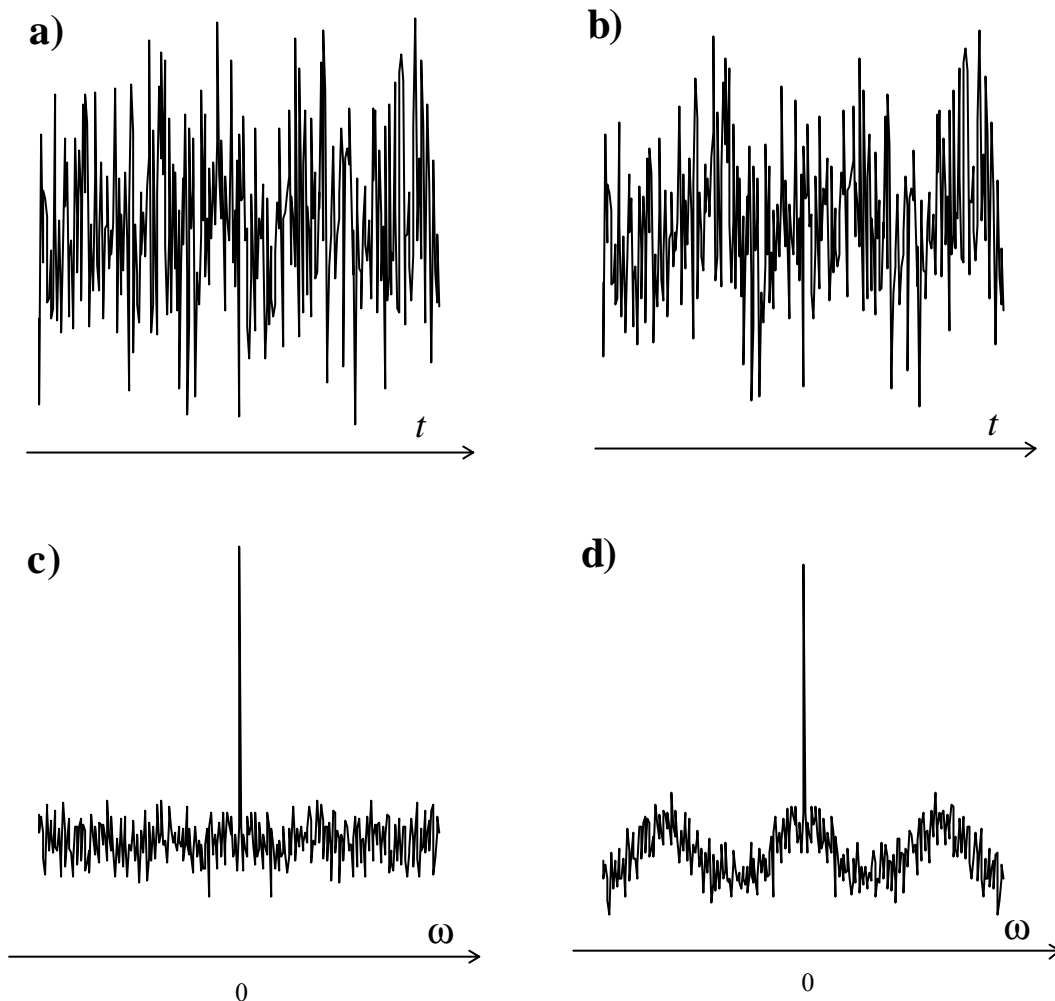
2.2.4 Applications of the autocorrelation theorem:

As the name suggests the autocorrelation of a function tells you something about ‘how the function relates to itself’. Therefore, it can be used to check for the information content in a function.

Consider a function which contains only noise. If we shift it over itself and integrate the overlap, the result will be completely uncorrelated for each shift other than zero, because here the function matches itself perfectly. It is easy to see that the result of the autocorrelation of perfect noise (hard to generate!) will be zero everywhere but at zero, where the result is integral of the square of the function.

Hence, the autocorrelation function of perfect noise is a delta-function. The autocorrelation function of any function will peak at zero, because any function correlates perfectly with itself (at zero shift...trivial). However, deviations at other frequencies than zero give inside into the information content, which might hard to recognise otherwise.

Example: see next figure.



The figure above shows a discrete application of the autocorrelation theorem (MATLABs FFT, which is a discrete FT).

a) a noise vector is generated (using `randn`). c) shows its autocorrelation function. We realise that the expected delta-function is well resembled. However, the fact that it is not perfectly flat shows the limitations in random-number generation.

b) is the same noise vector as in a) but with a sine-signal of low amplitude added.

d) is the autocorrelation function of b). We recognise significant deviations from zero (flatness) at values $\omega \neq 0$, which tell us that b) isn't completely noisy!

Such or similar operations are used to check noise generators for their quality and to seek information in noisy signals. The SETI program uses similar procedures to search for 'intelligent' patterns in radio signals (for a screen-saver see www.setiathome.com).

further examples:

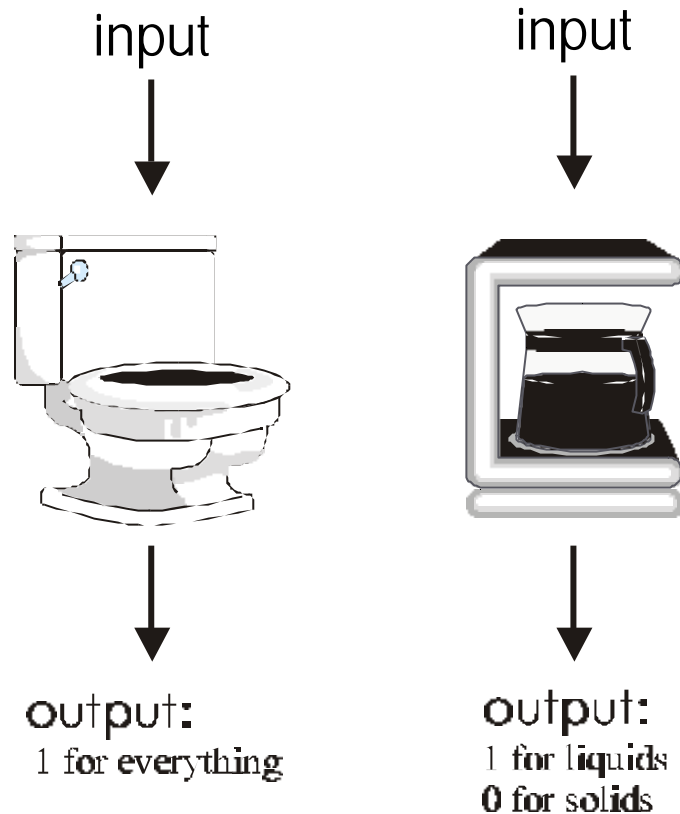
see MATLAB workshop

(for the really hot physicists: you might ask me about the relation of the autocorrelation function to *Patterson* synthesis of the reciprocal lattice in X-ray diffraction)

3. Filtering

3.1. Filters

What is a filter?



Definition:

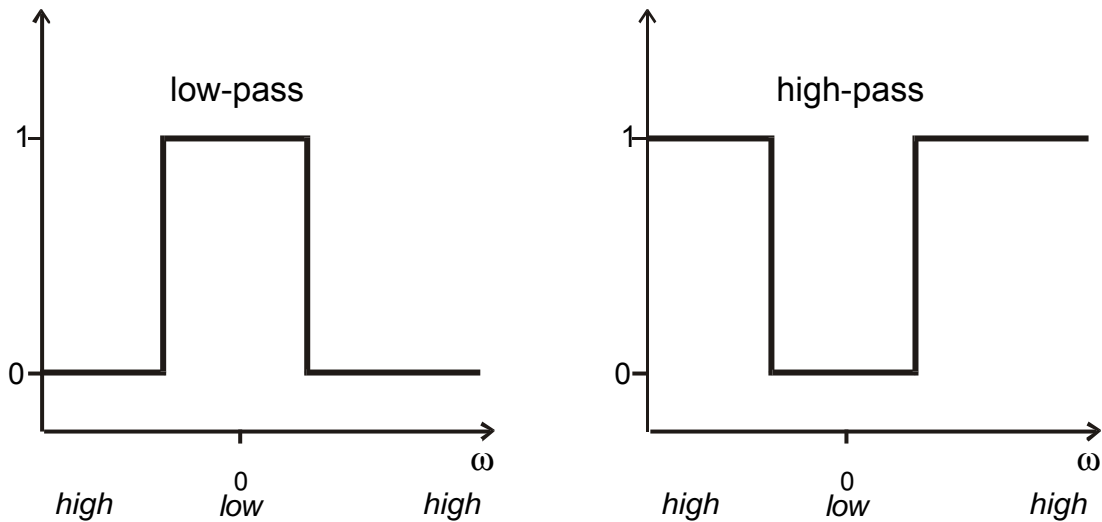
A filter is a function that selectively suppresses (filter-function = 0) or passes (filter-function = 1) or scales ($0 < \text{filter-function} < 1$) an input.

We will define filter-functions predominantly in the frequency domain, which means in the reciprocal domain of the function to be filtered.

This is because we want to select certain frequency components of a function.

When we stick to our definition of the origin of discrete FTs being in the centre of the co-ordinate frame, we will find the low frequencies at the centre and the higher frequencies at the edge. Hence, filter functions are classified by the region of frequencies they let pass or suppress.

Here are two one-dimensional examples: A **low-pass** and a **high-pass** filter:

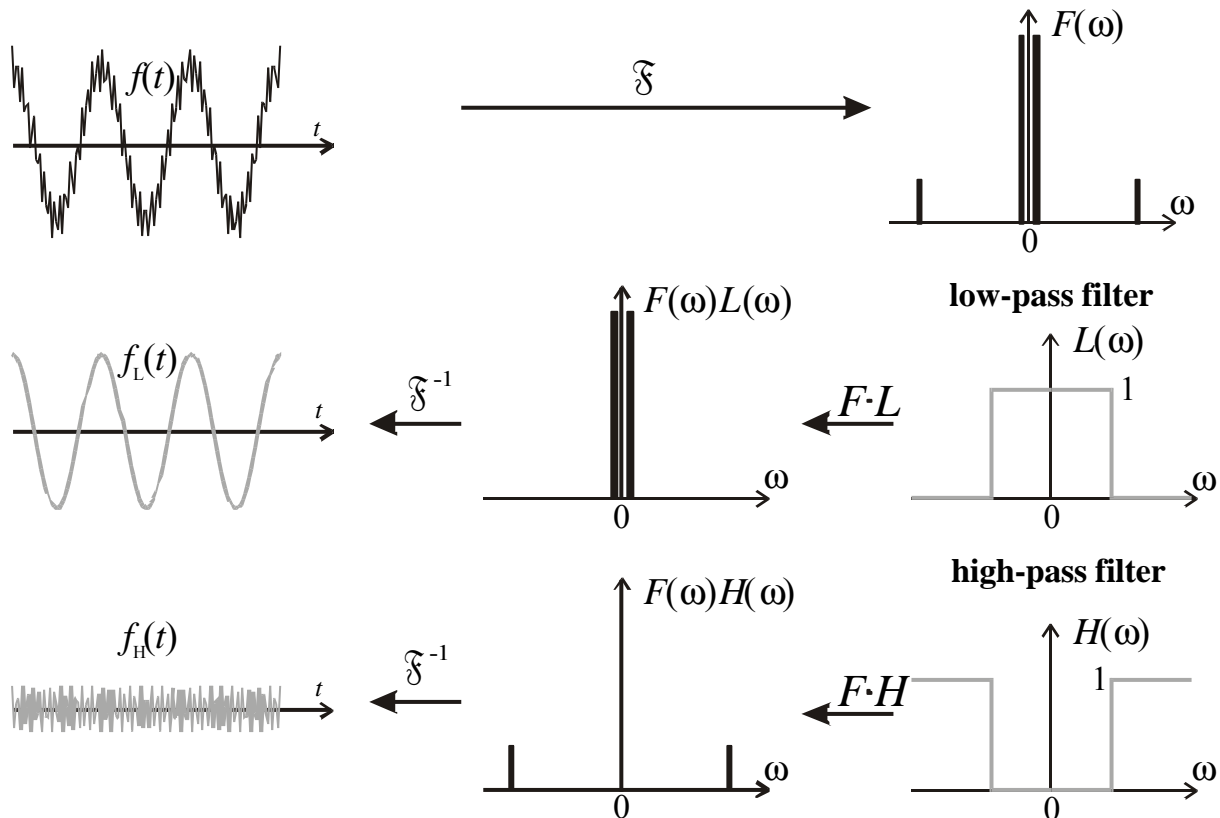


How is this done?

If the function $f(t)$ should be filtered, we convert it into the frequency-domain by performing its FT giving $F(\omega)$. Subsequently we multiply each point of $F(\omega)$ with the filter-function. The iFT then gives the filtered function.

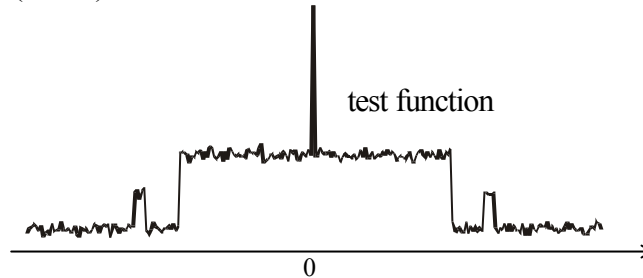
Hence: low-pass filtering (with the ‘top-hat’ function above) corresponds of a convolution with a sinc-function.

high-pass filtering (with the inverted ‘top-hat’ function above) corresponds of a convolution with a delta-function minus the sinc-function....why?).



This is illustrated in the figure above, where function $f(t)$ -which consists of a sum of two cosine-functions of different frequencies and amplitudes- is low-pass filtered giving the component with the low frequency. While a high-pass filter reveals the component with the high frequency (and lower amplitude in this case).

There are many filter functions- the following table gives an overview. In the last column they are applied to a test function (below)



name	formula	graph	result
low-pass	$F(\omega) = \begin{cases} 1 & \text{for } \omega < \omega_C \\ 0 & \text{else} \end{cases}$		
high-pass	$F(\omega) = \begin{cases} 0 & \text{for } \omega < \omega_C \\ 1 & \text{else} \end{cases}$		
band-pass	$F(\omega) = \begin{cases} 0 & \text{for } \omega_1 < \omega < \omega_2 \\ 1 & \text{else} \end{cases}$		
Hanning	$F(\omega) = \frac{1}{2} [1 - \cos(\omega)]$ with $0 < \omega < 2\pi$		
Butterworth of degree n	$F(\omega) = \frac{1}{1 + \left(\frac{\omega}{\omega_C}\right)^{2n}}$		

in this table ω_C (as well as ω_1 and ω_2) denotes a cut-off frequency.

Of course *Hanning*- and *Butterworth*-filters can also be used for high-pass filtering (simply by applying $H(\omega) = 1 - L(\omega)$)

Examples: MATLAB workshop.

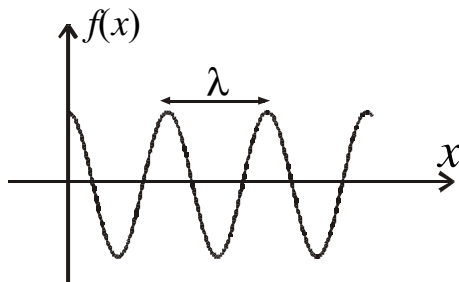
3.2. Reciprocal Domains, Spatial Frequencies and k -space

Fourier transforms are usually introduced using time and frequency as reciprocal domains. However, this isn't very useful for image processing, because images will usually have spatial co-ordinates.

On the other hand, the *Fourier*-transform is just another mathematical operation, which transforms one function into another, and it doesn't really matter on what co-ordinate system it is based.

The above statement is sufficient, and we could leave it as that. However, due to their importance we will have a more detailed look at spatial frequencies.

Consider a quantity that oscillates as a function of position (x) as shown in the next figure. This could be a short-time exposure of a swinging string, water wave or the general feature of a surface (e.g. a grating).



In this case we can write: $f(x) = \cos(k_x x)$

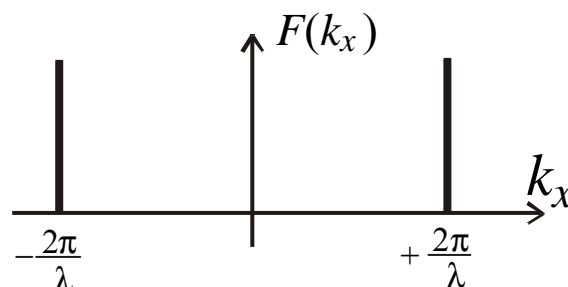
k_x is called spatial frequency (in x -direction)

$$k_x = \frac{2\pi}{\lambda} = 2\pi \frac{\text{number of oscillations}}{\text{metre}}$$

we can also apply complex notation: $\cos \vartheta = \frac{e^{i\vartheta} + e^{-i\vartheta}}{2}$ or $f(x) = \frac{e^{ik_x x} + e^{-ik_x x}}{2}$

Like in the discussion of *Fourier*-series (see page 7) the spectrum of $f(x) = \mathcal{F}(f(x)) = F(k_x)$ consists of two spatial frequencies associated with $\pm k_x$

Likewise we could say that the (even) cosine-function cannot distinguish between positive and negative arguments.



A spatial *Fourier*-transform is then defined completely analogous to the temporal FT:

$$\text{FT:} \quad G(k_x) \equiv \mathfrak{F} g(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} g(x) \exp(-ik_x x) dx \quad [3.1.1]$$

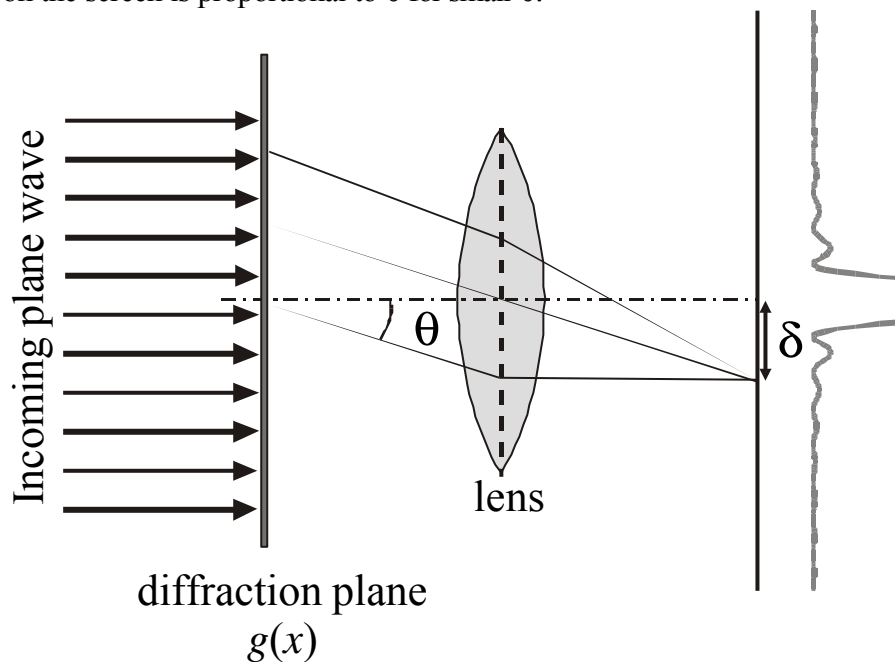
$$\text{iFT:} \quad g(x) \equiv \overline{\mathfrak{F}} G(k_x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} G(k_x) \exp(ik_x x) dk_x \quad [3.1.2]$$

We note that a spatial frequency is likely to vary not only with x , but most likely with y and z as well. This gives rise to the question of two-, three- and multi-dimensional FT. We will cover that in the next chapter.

Physics example: *Fraunhofer* diffraction:

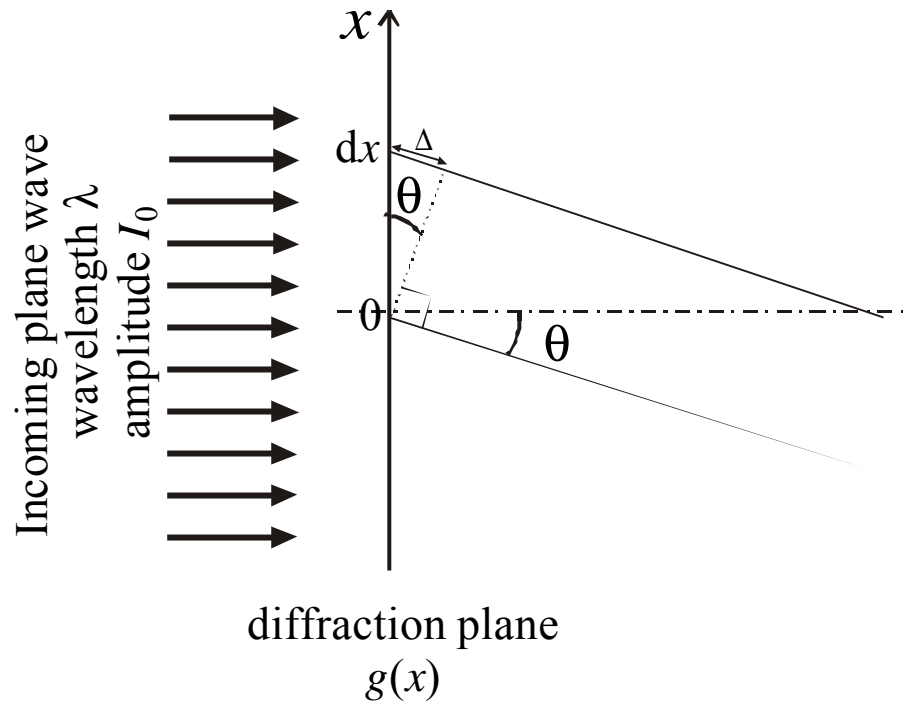
Reminder: Fraunhofer diffraction experiment is arranged such, that the diffracted rays travel a very long distance to the screen. Alternatively a lens is used to focus all rays diffracted through an angle θ to a point onto the observing plane (at finite distance).

The following figure illustrates this set-up. The diffraction plane (e.g. two slits) is described by a general spatial transmission coefficient $g(x)$. Due to the theoretically long distance the diffraction distance, δ , on the screen is proportional to θ for small θ .



Theorem:

The *Fraunhofer* diffraction pattern produced by a spatial transmission coefficient $g(x)$ is the spatial FT of $g(x)$.



The extra path length, Δ , of the upper ray relative to the lower at $x=0$ is given by $\Delta = x \sin\theta$.

Hence, the upper ray has a phase lag of $d\phi = \frac{2\pi x \sin \theta}{\lambda}$

The total amplitude of the transmitted wave - diffracted into direction θ - is then given by integral over all these rays,

$$I(\theta) = \int_{-\infty}^{\infty} I_0 g(x) \exp\left(-i2\pi \frac{x \sin \theta}{\lambda}\right) dx$$

Now we use the fact that θ is small, hence $\sin\theta \approx \theta$, and we substitute $k_x = \frac{2\pi\theta}{\lambda}$

$$I(\theta) \approx \int_{-\infty}^{\infty} I_0 g(x) \exp\left(-i2\pi \frac{x\theta}{\lambda}\right) dx$$

$$I(k_x) = I_0 \int_{-\infty}^{\infty} g(x) \exp(-ixk_x) dx \quad \text{qed!}$$

however, the eye and all (time-integrating) detectors only detect the intensity of light $= |I(k_x)|^2$

Knowing the autocorrelation theorem, we can even extend the statement for detectors:

The *Fraunhofer* diffraction pattern produced by a spatial transmission coefficient $g(x)$ is the spatial FT of $g(x)$. Detectors will show the FT of the autocorrelation function of $g(x)$.

Hence, for a single slit ('top hat') the screen will show an intensity distribution proportional to sinc^2 .

4. 2D, 3D and nD Fourier Transforms

4.1. Definitions

Without proof: Definition of the 2D-spatial FT of a 2D-function $g(x,y)$:

$$\text{2D-FT: } G(k_x, k_y) \equiv \mathfrak{F} g(x, y) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(x, y) \exp[-i(k_x x + k_y y)] dx dy \quad [4.1.1]$$

$$\text{2D-iFT: } g(x, y) \equiv \overline{\mathfrak{F}} G(k_x, k_y) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} G(k_x, k_y) \exp[+i(k_x x + k_y y)] dk_x dk_y \quad [4.1.2]$$

Straightforward analogy then gives the n D-FT of an n -dimensional function $g(x_1, x_2, \dots, x_n)$

$$\begin{aligned} G(k_{x_1}, k_{x_2}, \dots, k_{x_n}) &\equiv \mathfrak{F} g(x_1, x_2, \dots, x_n) = \\ &= \frac{1}{\sqrt[n]{2\pi}} \int_{-\infty}^{\infty} \dots \int_{-\infty}^{\infty} g(x_1, x_2, \dots, x_n) \exp\left[-i \sum_{\alpha=1}^n k_{x_\alpha} x_\alpha\right] dx_1 dx_2 \dots dx_n \end{aligned} \quad [4.1.3]$$

how does the n D-iFT look like?

Example: 2D-rectangle $g(x,y)$ with side-lengths a_x and a_y (see figure next page)

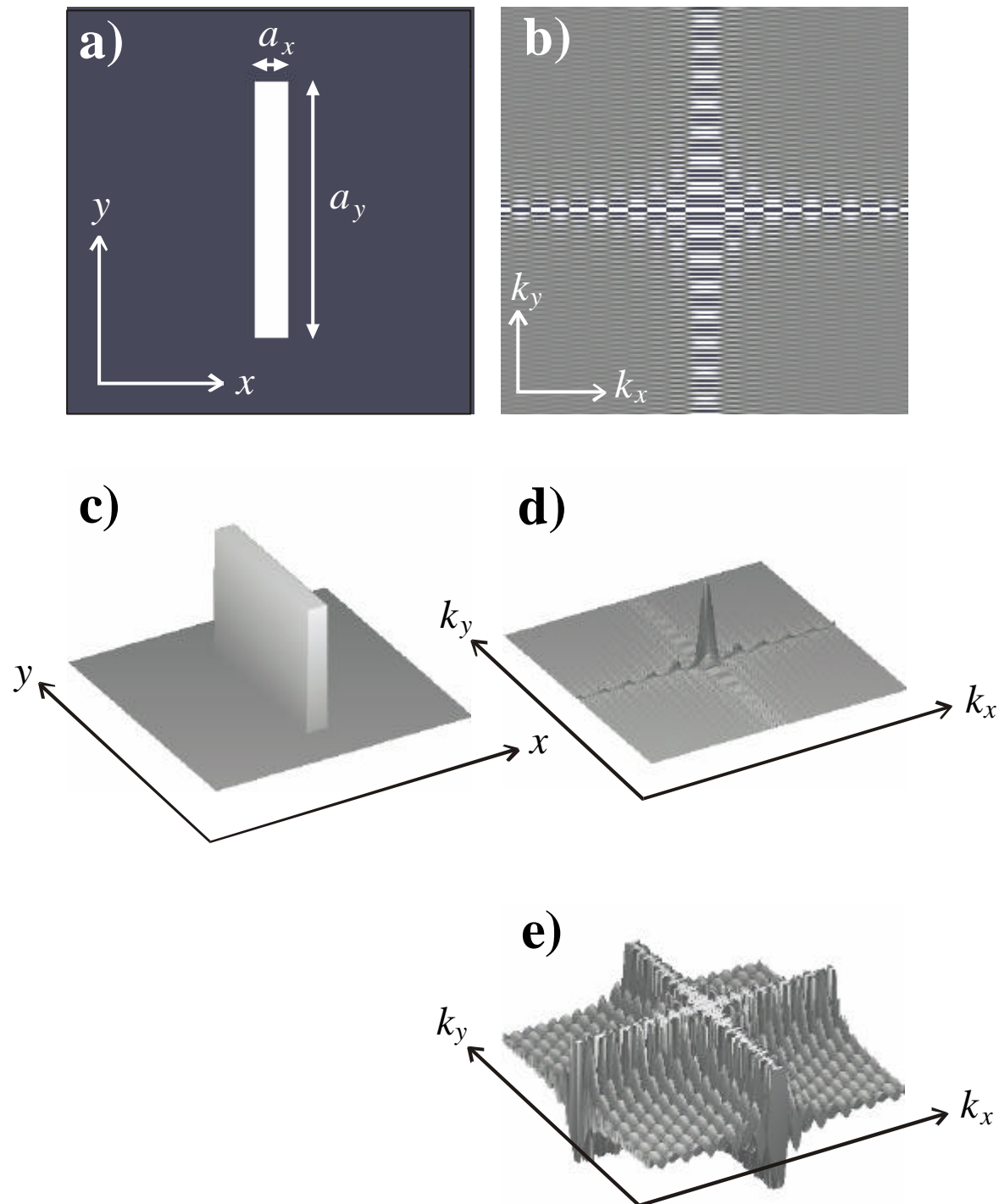
$$\text{function } g(x, y) = \begin{cases} 1 & \text{for } -\frac{a_x}{2} < x < +\frac{a_x}{2} \wedge -\frac{a_y}{2} < y < +\frac{a_y}{2} \\ 0 & \text{else} \end{cases}$$

$$\begin{aligned} G(k_x, k_y) &\equiv \mathfrak{F} g(x, y) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(x, y) \exp[-i(k_x x + k_y y)] dx dy \\ &= \frac{1}{2\pi} \int_{-\frac{a_y}{2}}^{+\frac{a_y}{2}} \exp(-ik_y y) \underbrace{\int_{-\frac{a_x}{2}}^{+\frac{a_x}{2}} \exp(-ik_x x) dx}_{\text{depends not on } y} dy = \frac{1}{2\pi} \int_{-\frac{a_y}{2}}^{+\frac{a_y}{2}} \exp(-ik_y y) dy \int_{-\frac{a_x}{2}}^{+\frac{a_x}{2}} \exp(-ik_x x) dx \end{aligned}$$

hence the FT can be separated into the product of 2 integrals we have already solved in part 1:

$$G(k_x, k_y) = \frac{1}{2\pi} a_x \operatorname{sinc}\left(\frac{k_x a_x}{2}\right) a_y \operatorname{sinc}\left(\frac{k_y a_y}{2}\right) = \frac{a_x a_y}{2\pi} \operatorname{sinc}\left(\frac{k_x a_x}{2}\right) \operatorname{sinc}\left(\frac{k_y a_y}{2}\right)$$

just the product of two sinc-functions (frequency depends on the widths)



2D-FT of a spatial rectangular function. a) spatial function $g(x,y)$, b) 2D-FT as grey-scale plot (limited amplitudes), c) same as a) but as surface plot, d) 2D-FT as surface plot with unlimited amplitude - negative parts are not visible, e) same as b) in surface representation.

Qualitatively it can be recognised how the wider side a_y transforms into a narrower sinc (higher frequency) and vice versa. On the $x=0$ and $y=0$ lines (centre) the FT corresponds to the 1D-FT of 1D-profiles along $g(x,y)$. At higher k -values these profiles are scaled by the orthogonal function (might be zero at a root).

Important:

ALL THE THEOREMS WE HAVE LEARNED IN PREVIOUS CHAPTERS ABOUT ONE-DIMENSIONAL *FOURIER*-TRANSFORMS CAN BE CONVERTED DIRECTLY INTO MORE-DIMENSIONAL *FOURIER*-TRANSFORMS.

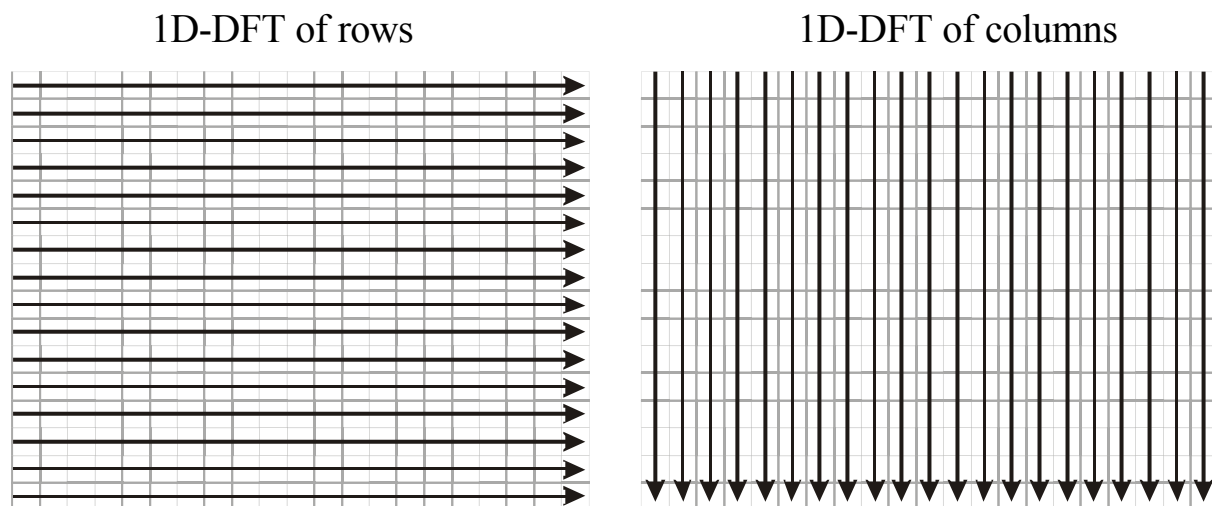
(e.g. addition, similarity, shift, convolution, correlation, symmetry, etc.)

CAREFUL: If the function is for example only shifted along one dimension (e.g. x but not y) the shift-theorem applies only to the corresponding k -dimension (k_x in this case, function will have additional phase along that direction).

Discrete 2D-FT:

In the example above we have seen how the analytical 2D-FT can be separated into two 1D integrals/FTs. Analogously we can apply the 1D-DFT to discrete 2D data.

We do the 1D-DFT for all rows first and then do a 1D-DFT of all columns (or the other way around: first columns then rows) as shown below:

**Example:**

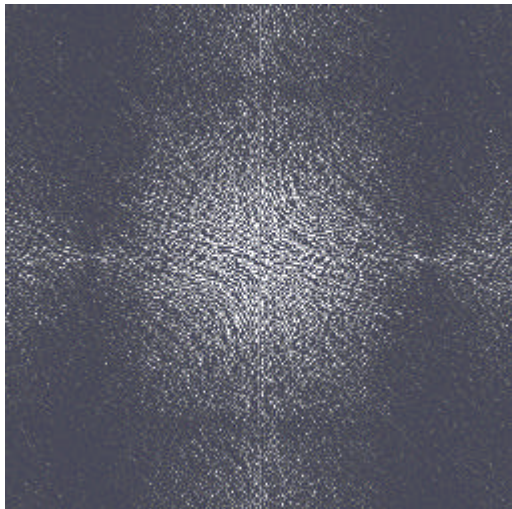
This process is of paramount importance for Magnetic Resonance Imaging (MRI). Where the data are acquired in k -space and 2D- or 3D-FT is used to generate the image.

(see figures next page... details about MRI on request)

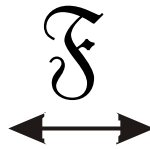
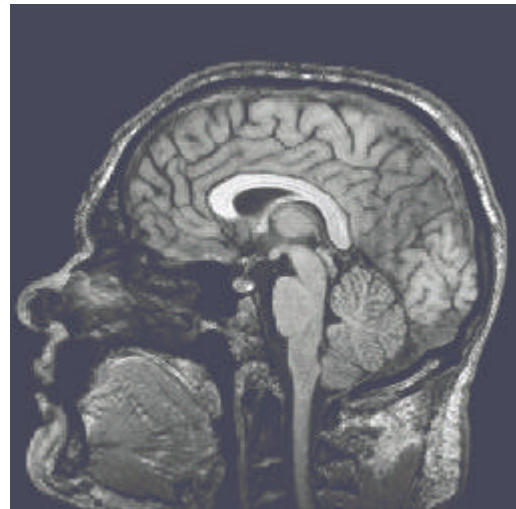
Similar processes are observed in diffraction (the previous figure of the 2D-‘top hat’ function) can be understood as a slit for *Fraunhofer*-diffraction. The diffraction pattern would then be the squared modulus of the 2D-sinc function.

It is important to notice that the observation of an intensity (autocorrelation function) doesn’t allow the reconstruction of the original function. This is causing the so-called ‘phase-problem’ in diffraction (e.g. X-ray crystallography... cf. lab experiment). See second graph on next page.

raw MRI data



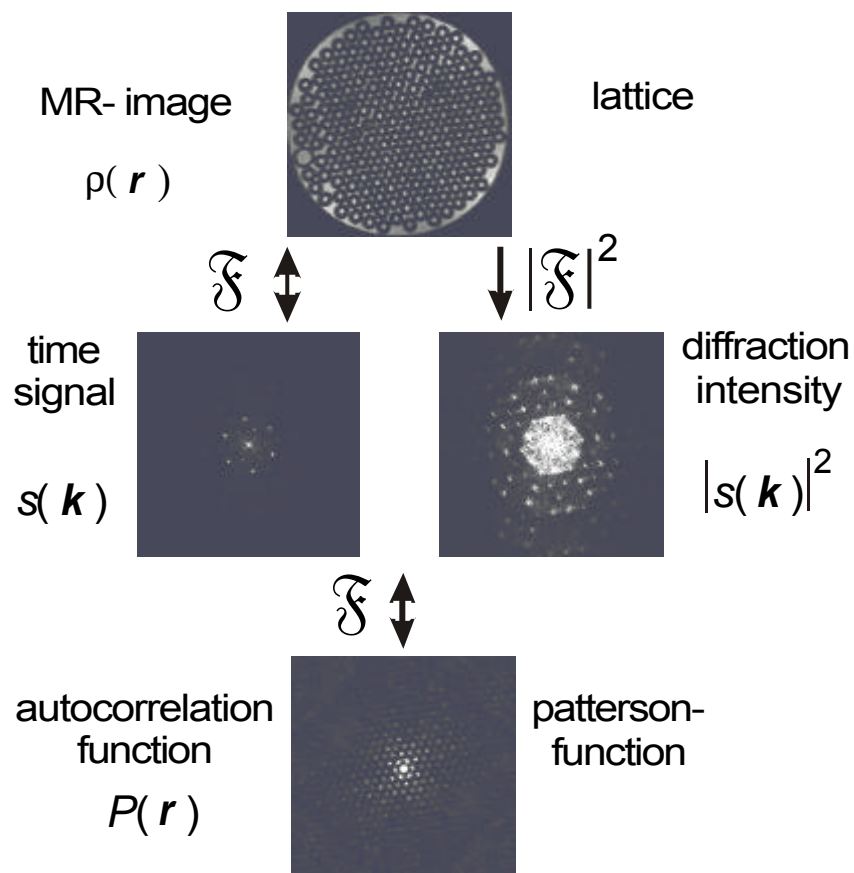
saggital image

**Comment:**

CT- and PET-scans (some older MRI) are produced differently. They apply a method called 'reconstruction from projections' (or 'backprojection' (BP)) using another transform... the *Radon*-transform. (\Rightarrow PH 609 *Medical Physics* or ask me about it)

MRI

Diffraction



4.2. Some special features of 2D - FTs

4.2.1 The standard theorems

As stated before there is nothing new here to learn, and the theorems from part 1.4 can directly be translated into a two-dimensional form. We repeat them here without further proof in their mathematical form.

Similarity theorem (see 1.4.1 and example page 38f.)

$$\text{If } f(x, y) \xrightarrow{\mathfrak{F}} F(k_x, k_y) \text{ then } f(ax, by) \xrightarrow{\mathfrak{F}} \frac{1}{ab} F\left(\frac{k_x}{a}, \frac{k_y}{b}\right)$$

Addition theorem (see 1.4.2)

$$\text{If } f(x, y) \xrightarrow{\mathfrak{F}} F(k_x, k_y) \text{ and } g(x, y) \xrightarrow{\mathfrak{F}} G(k_x, k_y) \text{ then}$$

$$f(x, y) + g(x, y) \xrightarrow{\mathfrak{F}} F(k_x, k_y) + G(k_x, k_y)$$

Shift theorem (see 1.4.3)

$$\text{If } f(x, y) \xrightarrow{\mathfrak{F}} F(k_x, k_y) \text{ then } f(x-a, y-b) \xrightarrow{\mathfrak{F}} \exp[-i(ak_x + bk_y)]F(k_x, k_y)$$

Parseval's theorem (see 1.4.4)

$$\text{If } f(x, y) \xrightarrow{\mathfrak{F}} F(k_x, k_y) \text{ then}$$

$$\int_{-\infty-\infty}^{\infty} \int_{-\infty-\infty}^{\infty} |f(x, y)|^2 dx dy = \int_{-\infty-\infty}^{\infty} \int_{-\infty-\infty}^{\infty} f(x, y) f^*(x, y) dx dy$$

$$= \int_{-\infty-\infty}^{\infty} \int_{-\infty-\infty}^{\infty} |F(k_x, k_y)|^2 dk_x dk_y = \int_{-\infty-\infty}^{\infty} \int_{-\infty-\infty}^{\infty} F(k_x, k_y) F^*(k_x, k_y) dk_x dk_y$$

However, there are some cases which do not have an equivalent in 1D. One important example is rotation (another is shearing and mirroring.... we will not cover them in this course, but if you are interested ask me for additional material).

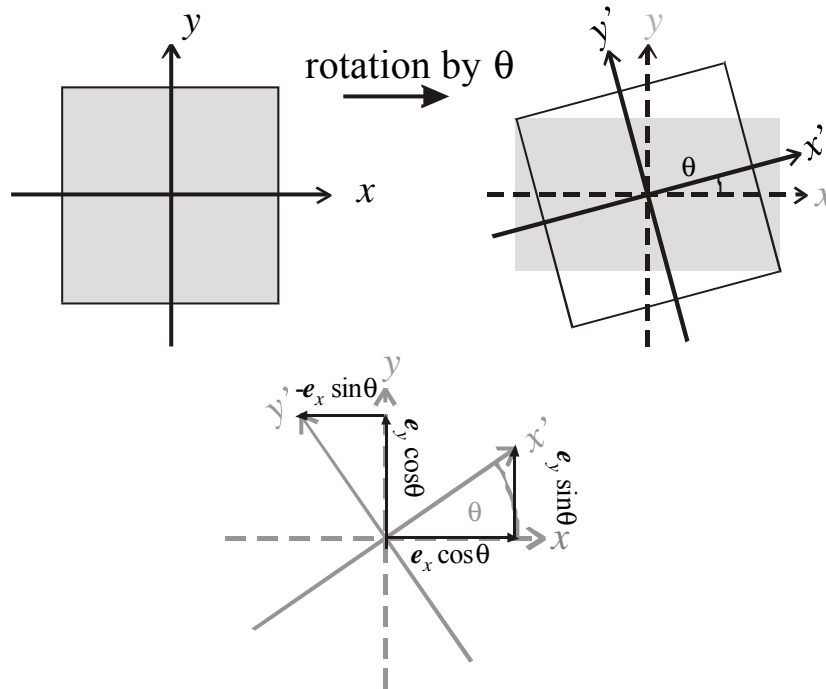
4.2.2 Rotation in two dimensions

Rotation of an image by an angle θ in two dimensions, means that each pixel at a position (x, y) will be transformed into a pixel at a new position (x', y') . This transformation is given by the following transformation matrix:

$$\underline{r}' = \begin{pmatrix} x' \\ y' \end{pmatrix} = \underline{\mathbf{R}}_{\theta} \underline{r} = \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \quad [4.2.1]$$

$$\text{hence:} \quad \begin{aligned} x' &= x \cos \theta + y \sin \theta \\ y' &= -x \sin \theta + y \cos \theta \end{aligned} \quad [4.2.2]$$

this principle is illustrated in the following figure:



The same transformation also applies in k -space (without proof*). Hence,

Rotation theorem:

[4.2.3]

If $f(x, y) \xrightarrow{\mathcal{F}} F(k_x, k_y)$ then

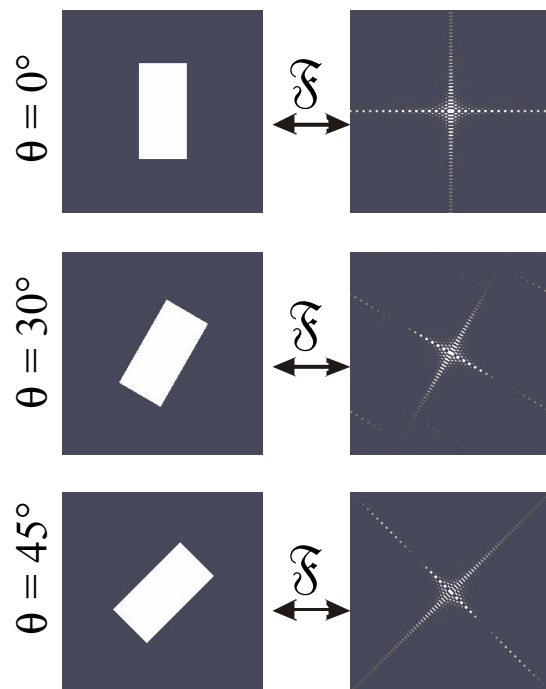
$$\underline{\underline{\mathbf{R}}}_\theta f(x, y) \xrightarrow{\mathcal{F}} \underline{\underline{\mathbf{R}}}_\theta F(k_x, k_y)$$

$$f(x', y') \xrightarrow{\mathcal{F}} F(k'_x, k'_y)$$

$$f(x \cos \theta + y \sin \theta, -x \sin \theta + y \cos \theta) \xrightarrow{\mathcal{F}} F(k_x \cos \theta + k_y \sin \theta, -k_x \sin \theta + k_y \cos \theta)$$

This means that the co-ordinates in k -space are rotated by the same degree. As demonstrated in the Figure on the right.

If you look very closely, you also recognise aliasing in the k -space image for the rotation at 30° . This is because the sinc-function theoretically extends to infinity. Hence it can re-enter the image as if extended on a cyclically wrapped infinite space.




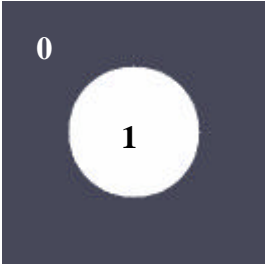
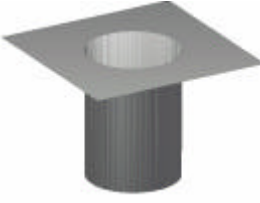
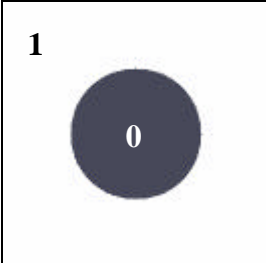






*the proof is relatively easy. Because the FT is a linear transformation affine transformations can be calculated straight-forwardly. Try it!

5. Image Processing in the Frequency Domain

5.1. Filtering

Now we are going to apply the same filters -which were introduced in part 3- in two dimensions.

The following table gives an overview:

name/formula	graphs	
<p>low pass</p> $F(k_x, k_y) = \begin{cases} 1 & \text{for } \sqrt{k_x^2 + k_y^2} < \omega_C \\ 0 & \text{else} \end{cases}$		
<p>high-pass</p> $F(k_x, k_y) = \begin{cases} 0 & \text{for } \sqrt{k_x^2 + k_y^2} < \omega_C \\ 1 & \text{else} \end{cases}$		
<p><i>Gaussian</i></p> $F(k_x, k_y) = \exp\left(-\frac{k_x^2 + k_y^2}{\omega_C^2}\right)$		
<p><i>Hanning</i></p> $F(k_x, k_y) = \frac{1}{2} \left[1 - \cos\left(\sqrt{k_x^2 + k_y^2}\right) \right]$ <p>with $0 < k_x < 2\pi$ and $0 < k_y < 2\pi$</p>		
<p><i>Butterworth of degree n</i></p> $F(k_x, k_y) = \frac{1}{1 + \frac{(k_x^2 + k_y^2)^n}{\omega_C^{2n}}}$		

in this table ω_C denotes a cut-off frequency.

We see that polar co-ordinates are used rather than Cartesian. This is useful, because then we only have to define a single cut-off frequency (the same then for all dimension). Therefore, all we have to do is to replace the 1D co-ordinate from part 3 (ω , see page 34) by a radial definition:

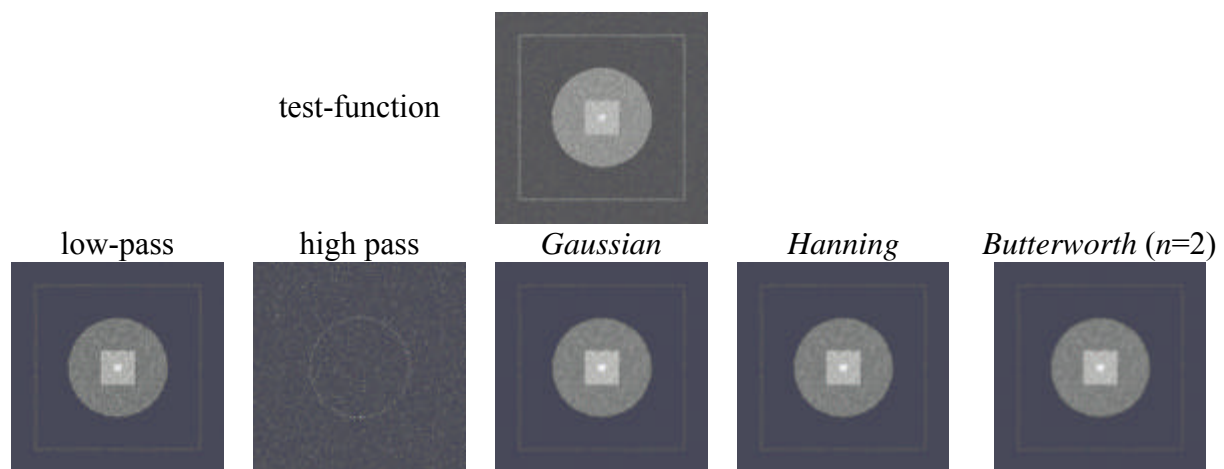
$$k_r = \sqrt{k_x^2 + k_y^2}$$

Of course, other definitions -e.g. for low-pass filters- are also useful (see below). Think about possible applications and how to define them in MATLAB!



We now want to study their effects in image processing. (they are principally the same as in 1D, but we are more used to look at 2D-images).

Therefore, a test-function is defined (as shown below) and the 2D-filters from the previous table are applied by multiplying the filter-function with the 2D-FT of the test-function. The inverse 2D-FT gives then the following images.



We see how the **low-pass filters** (*Gaussian*, *Hanning* and *Butterworth* were also defined with a low-pass characteristic) **reduce noise** and also cause **smoothing** effects (similar to neighbourhood averaging in the spatial domain). This is because they only let the coarse features (at low frequencies) pass.

Noise tends to be random from one to the next pixel and therefore to occur at the highest spatial frequencies. Therefore, low-pass filters reduce the noise, by suppressing these high frequencies.

On the other hand, **high-pass filters** let only these high frequencies pass. Therefore, there is an enhancement of the noise. Remembering the FT of the ‘top-hat’ function, we recall that the edges actually transform to all frequencies. By letting only the high frequencies pass we suppress the coarse features (also amplitudes or grey levels) and selectively filter strong

contrast changes. Hence high-pass filters can be used for **edge-detection** (cf. gradient kernels in the spatial domain).

Example: Low-pass

The influence of the cut-off frequency (radius of the low-pass filter) is further illustrated in the following figure.

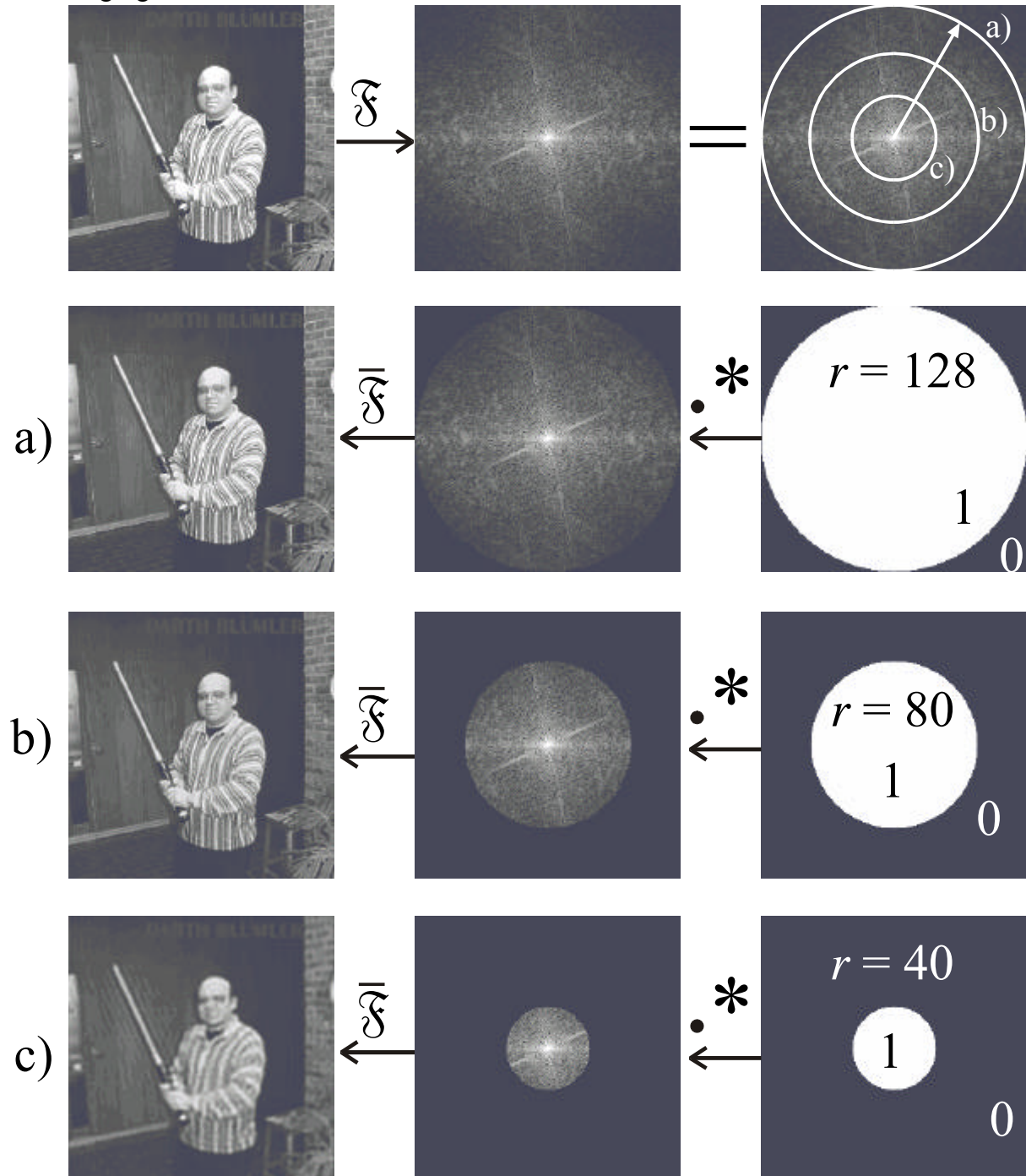


Fig. 5.1.: Application of low-pass-filters of different cut-off frequency (radius r) to an image.

Top row: left: original image (256×256 pixel²), centre: 2D-FT (logarithmic grey-scale); right: as centre but with overlaid radii of the following low-pass filters.

The next rows show the filter function on the right. The centre is the multiplication of the 2D-FT of the original image with the filter-function (symbolised by the MATLAB command `*`). The left column is the reconstruction of the image after inverse 2D-FT of the filtered k -space function in the centre.

a) cut-off frequency = 128 pixel, b) 80 pixel, c) 40 pixel.

We clearly see that a lower cut-off frequency smooths the image significantly (also produces some artefacts).

Example: High-pass

The influence of the cut-off frequency (radius of the high-pass filter) is further illustrated in the following figure. Remember that $HP=1-LP$!

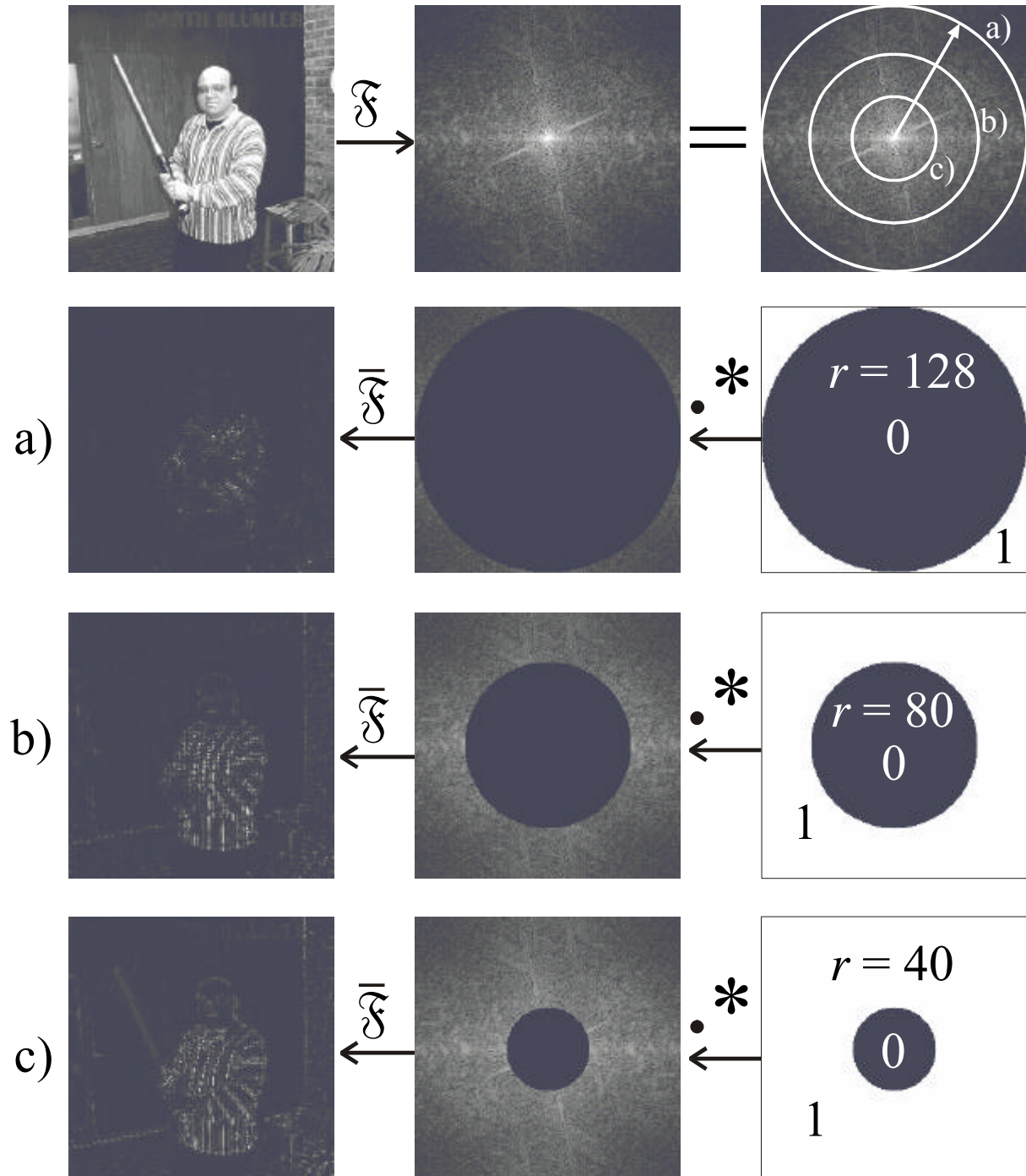


Fig. 5.2: Application of high-pass-filters of different cut-off frequency (radius r) to an image.

Top row: left: original image (256×256 pixel²), centre: 2D-FT (logarithmic grey-scale); right: as centre but with overlaid radii of the following low-pass filters.

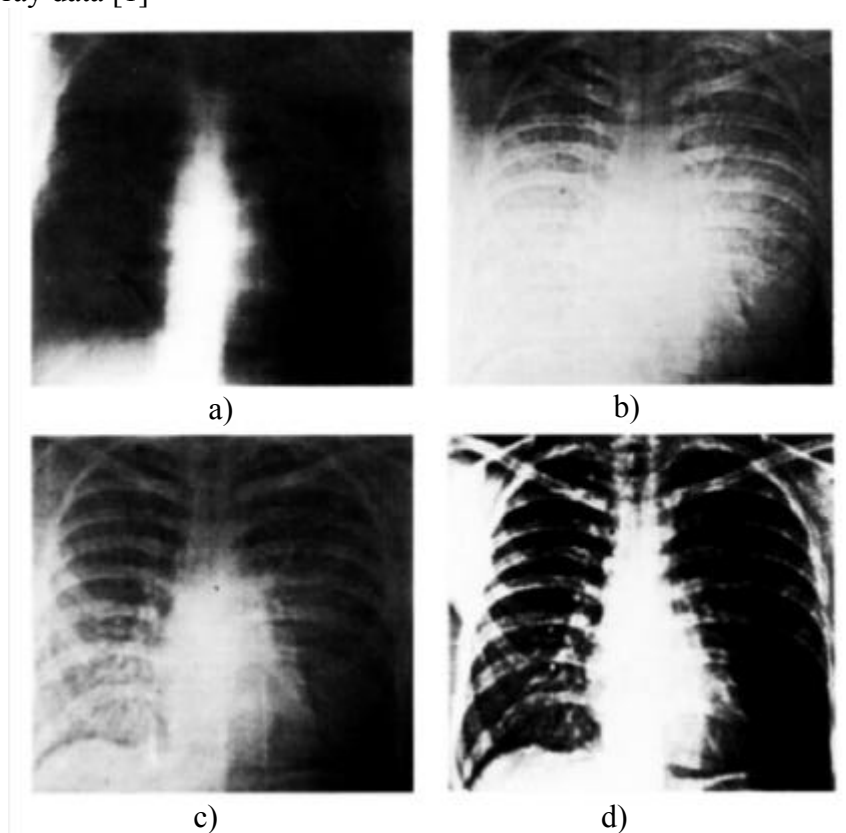
The next rows show the filter function on the right. The centre is the multiplication of the 2D-FT of the original image with the filter-function (symbolised by the MATLAB command `*`). The left column is the reconstruction of the image after inverse 2D-FT of the filtered k -space function in the centre.

a) cut-off frequency = 128 pixel, b) 80 pixel, c) 40 pixel.

This is somewhat the opposite. The higher the cut-off frequency, the stronger the effect. We clearly see the difference in the edges, which are detected. The highest cut-off (a) gives only the black/white stripes on the jumper (highest contrast). Weaker contrasts appear for lower cut-offs.

The sum of the corresponding images in Fig. 5.1 and Fig 5.2 then must reveal (addition theorem!) the original. Hence, we can inspect Fig. 5.2 for what image information was removed by the filter in Fig. 5.1 and vice versa.

Example: X-ray data [1]



Example of high-pass filtering: a) original x-ray image. b) image processed with a high-pass *Butterworth* filter (define!). c) result of **high frequency emphasis** (= *high-pass* + *constant* in order to preserve the low frequency components or coarse image features... however they are damped compared to the edges). d) histogram equalisation (see part I) of c).

For the interested: Advanced filtering also includes homomorphic filtering (see [1] page 213ff.) and adaptive filtering (see [1] page 218ff.) or ask me....

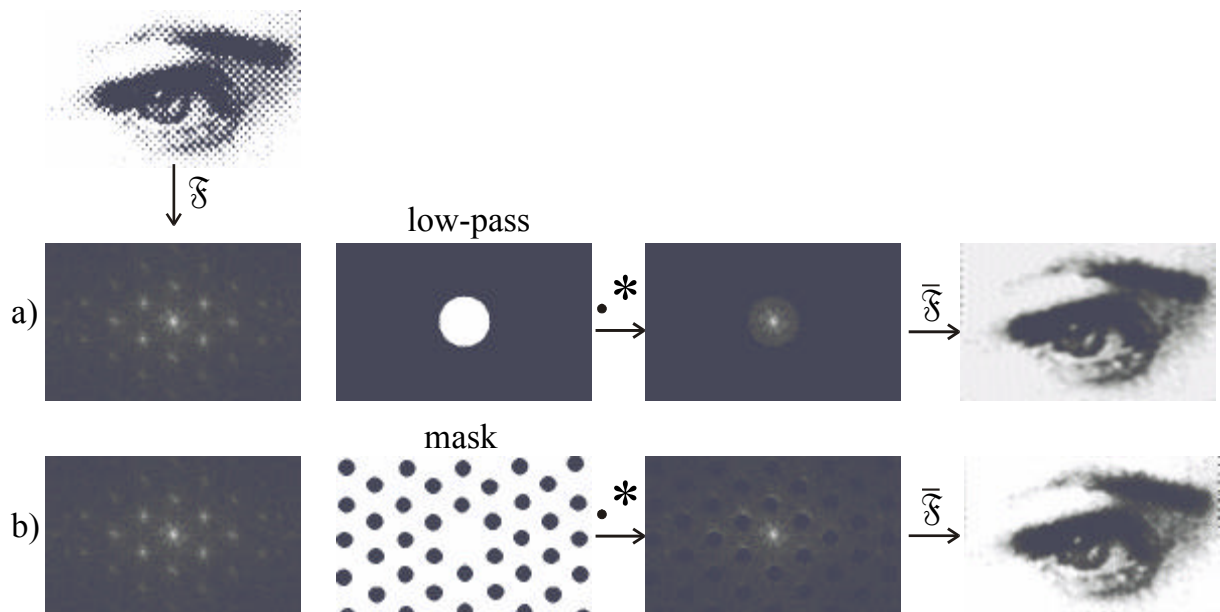
[1] adapted from R. C. Gonzalez and R. E. Woods: „*Digital Image Processing*“, page 214.

5.2. Removal of Moiré patterns and other interference

Moiré patterns arise when halftone images (raster graphics) is scanned in. If the graphic is scanned in with a different resolution (dots-per-inch) or re-scaled (see example below), disturbing interference of the two different rasters can occur.



So how does the FT of the image above look like? The raster is much smaller than the image-feature (eye), hence it appears at high spatial frequencies in k -space. It also has a regular pattern, which must dominate the FT (cf. repetitive structures = single frequency). So we expect isolated peaks at high frequency to be responsible for the raster.



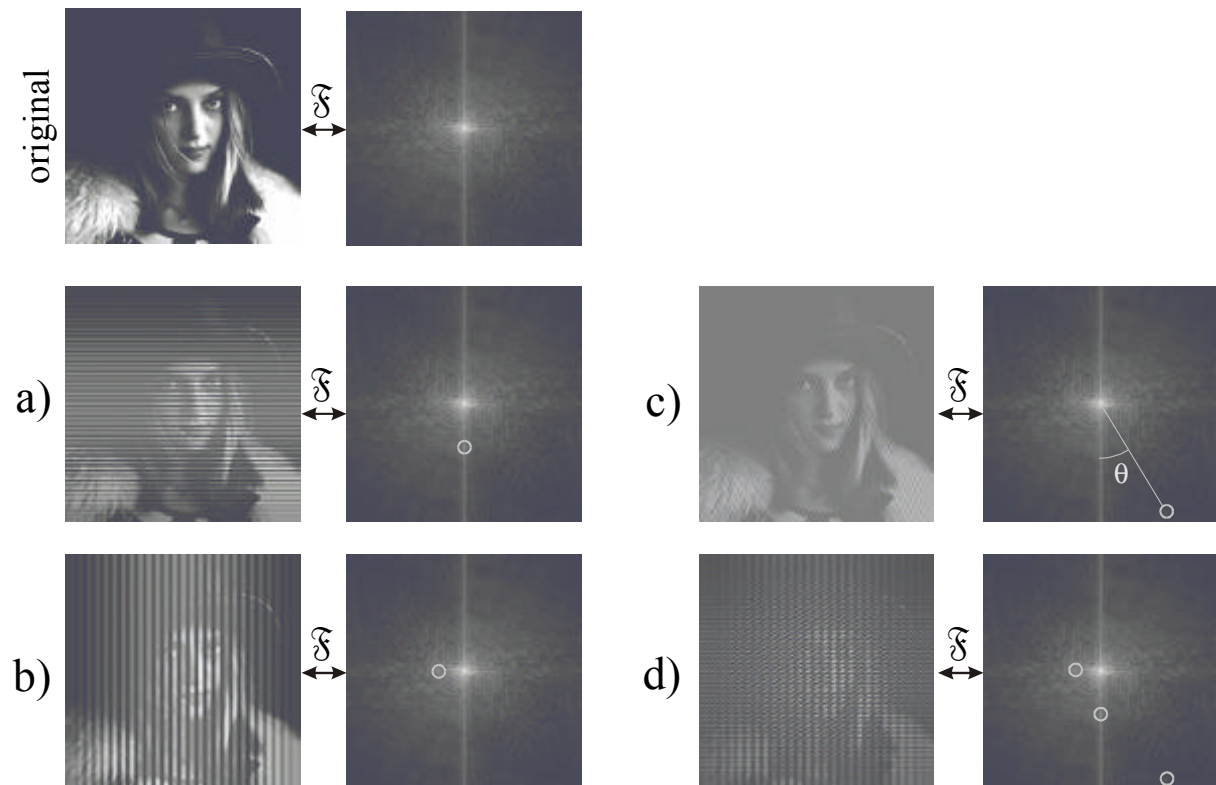
We see that the 2D-FT looks exactly like predicted. So We can remove the Moiré effect by applying a low-pass filter with a cut-off frequency smaller than the first of the reciprocal raster peaks (shown in the figure (a) above). However, this causes smoothing by neighbourhood averaging.

Likewise, we can apply a suitable mask (see (b)) which just blinds the reciprocal raster peaks. The reconstructed image shows more detail.

Similar effects are often observable in physical scanning techniques (which do you know?). The scanning raster is often visible as an artefact and can be removed by masking or low-pass filtering as described above.

Similar effects can also occur, if the data are acquired in spatial frequencies (what physical methods do you know which do that?) and a single or more points obstructed by a ‘spike’ (single miss-read value, typically of high amplitude).

Then the reconstruction of the data will reveal some very intense interference pattern (see figure below).



Explanation figure:

The images are shown together with their 2D-FT. Single ‘spikes’ (one intense pixel) in k -space are marked by circles.

- A spike on the k_y -axis causes horizontal stripes in the image.
- A spike on the k_x -axis causes vertical stripes in the image. (Note the increased distance and width of the interference pattern / stripes. This is due to the fact that the spike is at lower k -values than in a).
- A spike at an angle θ from the centre causes stripes normal to the connecting axis (spike- centre ok k -space). (Note the very high frequency of the stripes, because the spike is close to the edge of k -space = high spatial frequencies).
- Due to the addition theorem, multiple spikes cause a sum-effect.

further examples: MATLAB workshop, question week 17.

6. Deconvolution

6.1. Blurring and Deblurring (the Theory)

Convolutions occur very often in physical data. For instance in blurred images, aperture functions, damping of oscillations, and other experimental or principal blurring conditions. A well-known example was the first solution of the misadjustment of the optics in the *Hubble* space-telescope. The images were all blurred by some instrumental function, which could be simulated and corrected by data analysis (not very successfully though, later correction lenses/mirrors were installed).

Such corrections are known as **deconvolution**, which is a typical example of an '*inverse problem*'.

Inverse problems are a class of problems, where there are many input variants which result in a relatively simple output and some of the input variables should be determined. Such problems are mathematically '*ill-defined*', which means that we have to simulate and/or guess some of the input values to model the output.

We remember the definition of the convolution theorem (see session 12):

If $f(x)$ has the FT $F(k_x)$ and $g(x)$ has the FT $G(k_x)$, then $f(x) \otimes g(x)$ has the FT $F(k_x) \cdot G(k_x)$; that is, convolution of two functions means multiplication of their *Fourier* transforms.

So a blurred function or image is given by $h(x) = f(x) \otimes psf(x)$ and its *Fourier*-transform $H(k_x) = F(k_x) \cdot PSF(k_x)$, where psf is the blurring or point-spread function and PSF its FT.

Why can we not deblur the function $h(x)$ (that is recover the unblurred function $f(x)$) via

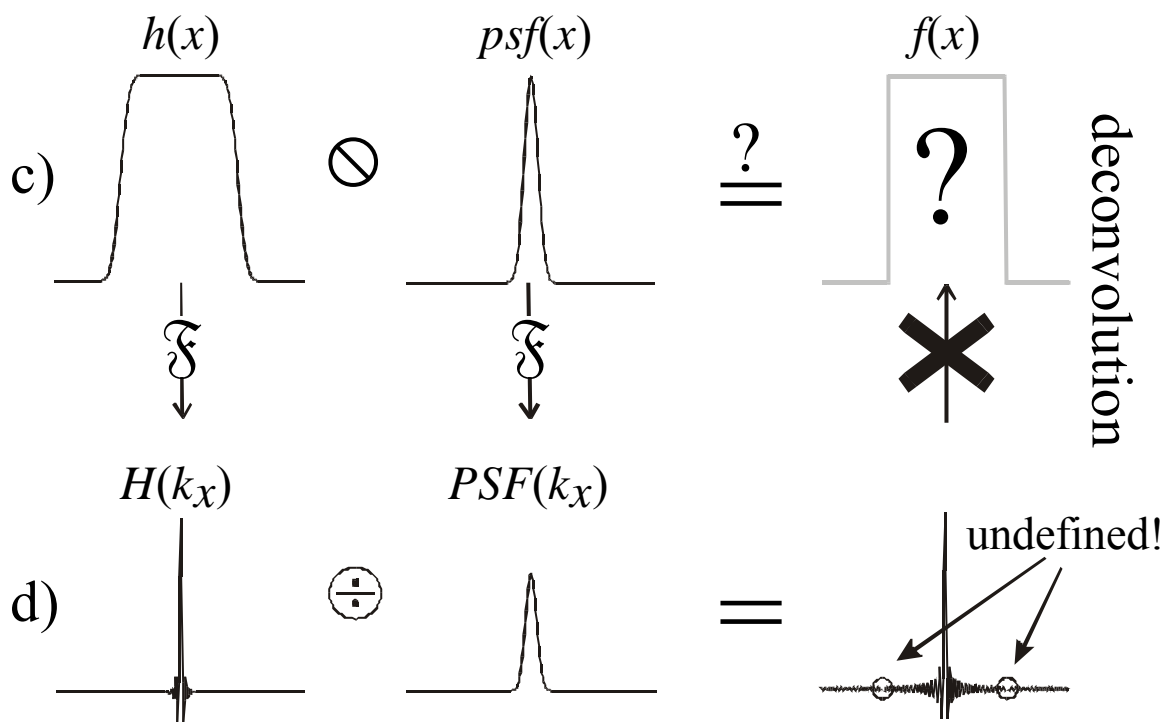
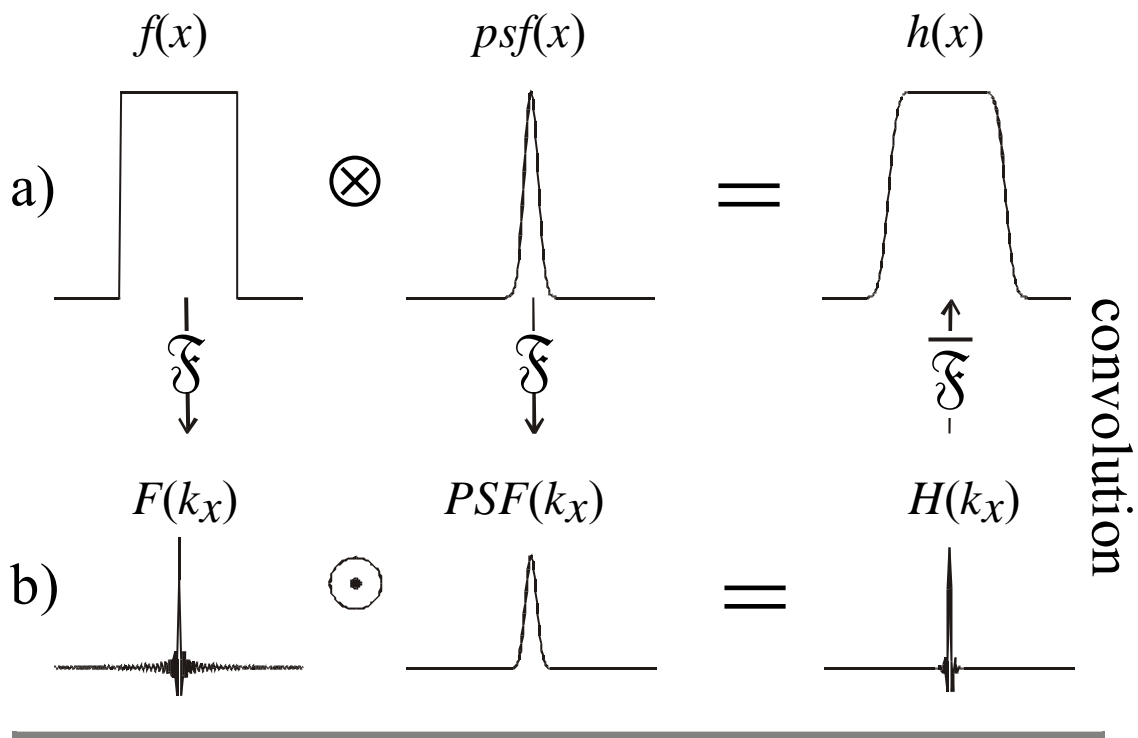
$$f(x) = \overline{\mathcal{F}} \left[\frac{H(k_x)}{PSF(k_x)} \right] \equiv h(x) \oslash psf(x) \quad [6.1]$$

this is the straight-forward definition of a deconvolution (symbol \oslash). However, we do not know if the fraction is defined, because $PSF(k_x)$ could be partially zero, and it usually is!

Therefore, from rigorous maths this problem is usually not solvable (not 'invertable') or generally 'ill-defined'!

The problem is illustrated in the following graphs:

- shows the blurring of function $f(x)$ by the point-spread function $psf(x)$ via convolution $h(x) = f(x) \otimes psf(x)$
- illustrates the same using the convolution theorem in the FT-domain.
- now deblurring of $h(x)$ is tried via deconvolution (however we are not sure if it works)
- this shows inverting the convolution theorem by dividing $H(k_x)$ by $PSF(k_x)$. However the result of this division is not defined everywhere!



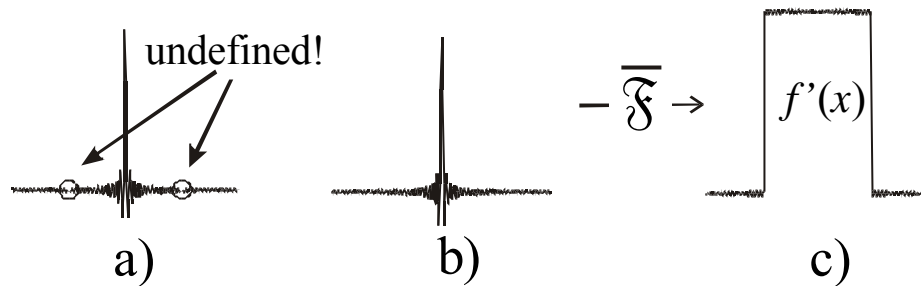
This first inspection of deconvolution might lead to the assumption, that in most cases it will not be possible to deblur data.

However, it would be of great importance for many scientific and technical applications to improve resolution of data. Therefore an entire scientific (essentially engineering and mathematical) research field exists, which has concentrated to solve or optimise this in rigour mathematical terms ‘impossible’ process.) To describe the various theories is definitely beyond the scope of this introduction.

However, a few simple tricks will be shown, how this can be done. Therefore, we focus on the ‘ideal’ (=noise-less) data of the previous example first.

a) Removing undefined regions by replacing them in the result of division:

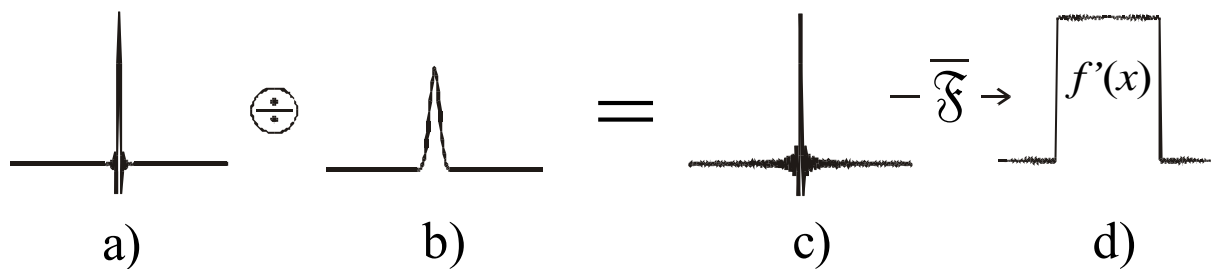
One way of avoiding undefined regions via division in the FT-domain is to identify such regions and replace them with zeros (or more advanced replace them with interpolated or predicted data). This process is illustrated in the following graph.



Here the result of the division a) (see most right figure of part d) in the previous figure) was inspected, and the undefined regions (division by 0) replaced by zeros (creating figure b). Inverse FT then reveals the ‘deblurred’ function $f'(x)$ (in c), which is (of course) not identical to the unblurred function $f(x)$ but very close! The blurring has been removed!

b) Adding a small value to the denominator before division:

Another (and generally easier) way is to avoid undefined regions. They occur in discrete deconvolution by the fact that we divide a quantity by a very small number (possibly 0) which generates a result that doesn’t compute (exceeding the definition range of numbers)! By adding a small constant to the denominator we avoid this problem, because it will influence the general operation only slightly but avoid excessive number generation. This process is illustrated in the next figure.



The addition of a small constant (10^{-22} in this case) doesn’t alter the visual features of graph b) (cf. figure d) on page 52), but tremendously alters the result of the division c) and its reconstruction via inverse FT (d). Hence, this is another way of deblurring!

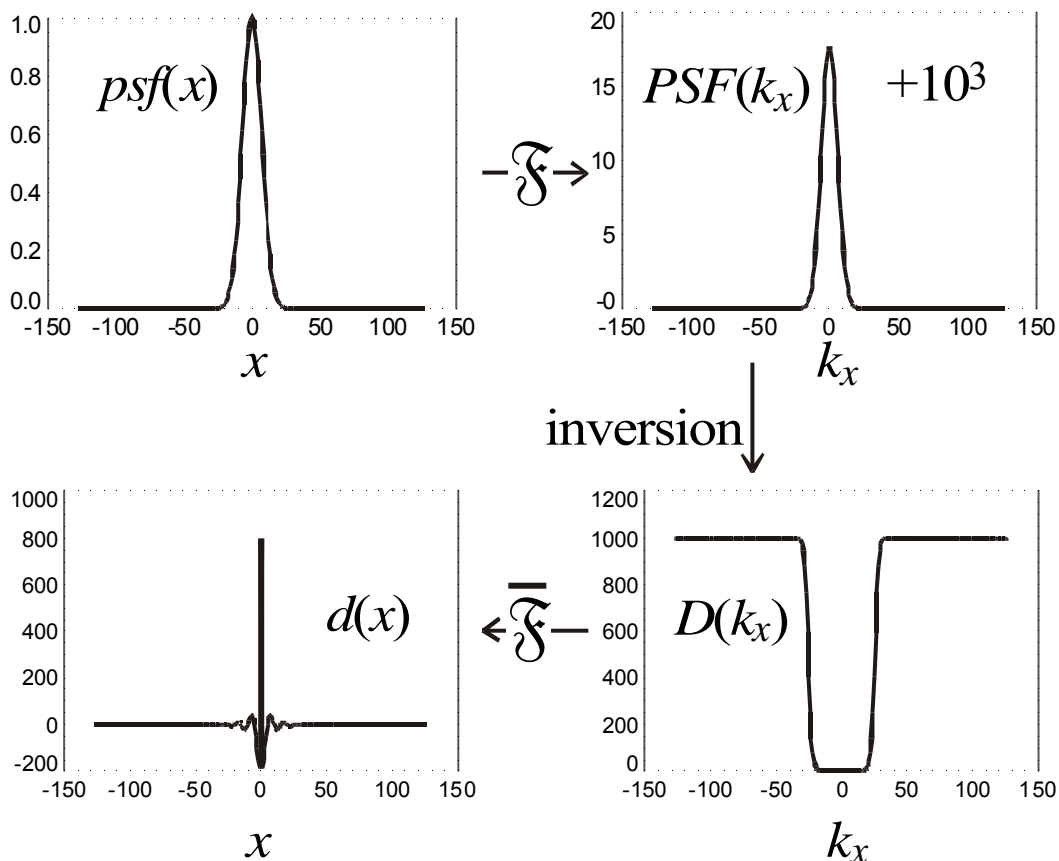
Generally a different strategy is recommended:

Rather performing the deconvolution via the definition in eq. [6.1], it is often easier to create an ‘inverse convolution’ function $d(x)$!

$$f(x) = \overline{\mathcal{F}} \left[\frac{H(k_x)}{PSF(k_x)} \right] = \overline{\mathcal{F}} \left[H(k_x) \cdot \frac{1}{PSF(k_x)} \right] = \overline{\mathcal{F}} [H(k_x) \cdot D(k_x)] \equiv h(x) \otimes d(x) \quad [6.2]$$

This ‘inverse convolution’ function then can be inspected for critical regions (excessive small or large values), noise, etc. Once it has been identified from measured data, it can often be modelled as an ideal function for deconvolution.

For the Gaussian blurring function ($psf(x)$) from above $d(x)$ and $D(k_x)$ look as follows:



(the interested reader is referred to R. H. T. Bates and M. J. McDonnell: „Image Restoration and Reconstruction“ Oxford Science Publ. Series 16, Clarendon Press, Oxford 1986)

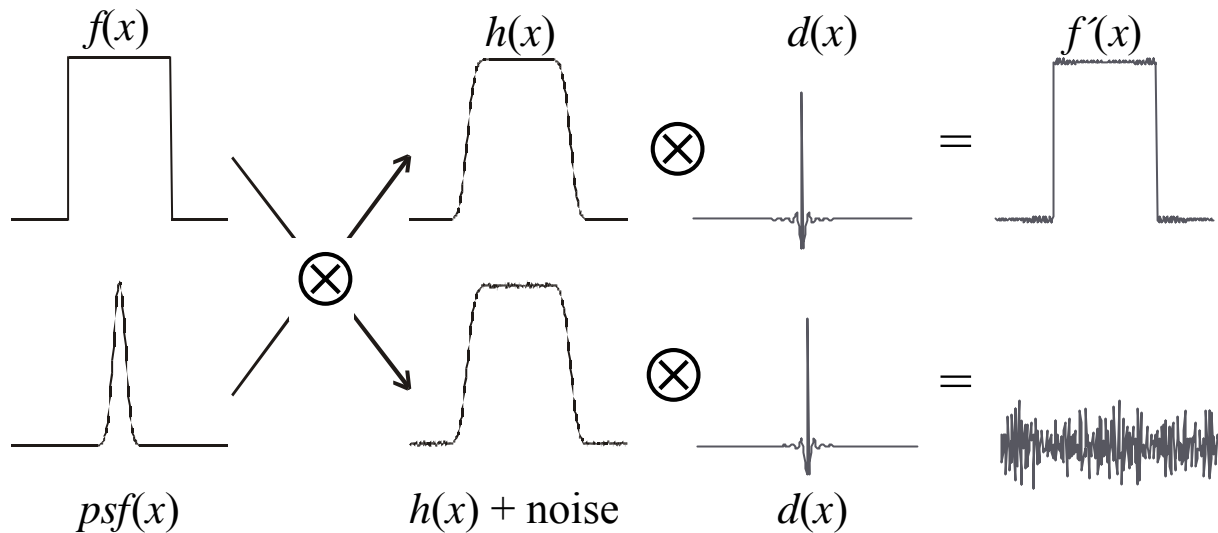
6.2. The Deconvolution-Killer - Noise!

From the previous figure we identify the inverted convolution function of having the characteristics of a high-pass filter (see page 34). Hence, we expect it to emphasise noise.

In fact, deconvolution is extremely susceptible to noise. This is because noise is not part of the blurring or convolution process, as

$$\text{blurred image} = (\text{unblurred image}) \otimes (\text{point-spread function}) + \text{noise}$$

which makes the inversion of the equation as in [6.1] additional complicated. Or in other words, the small values in the FT of the psf will transform in large values in the ‘inverse convolution’ function and hence scale up the noise:



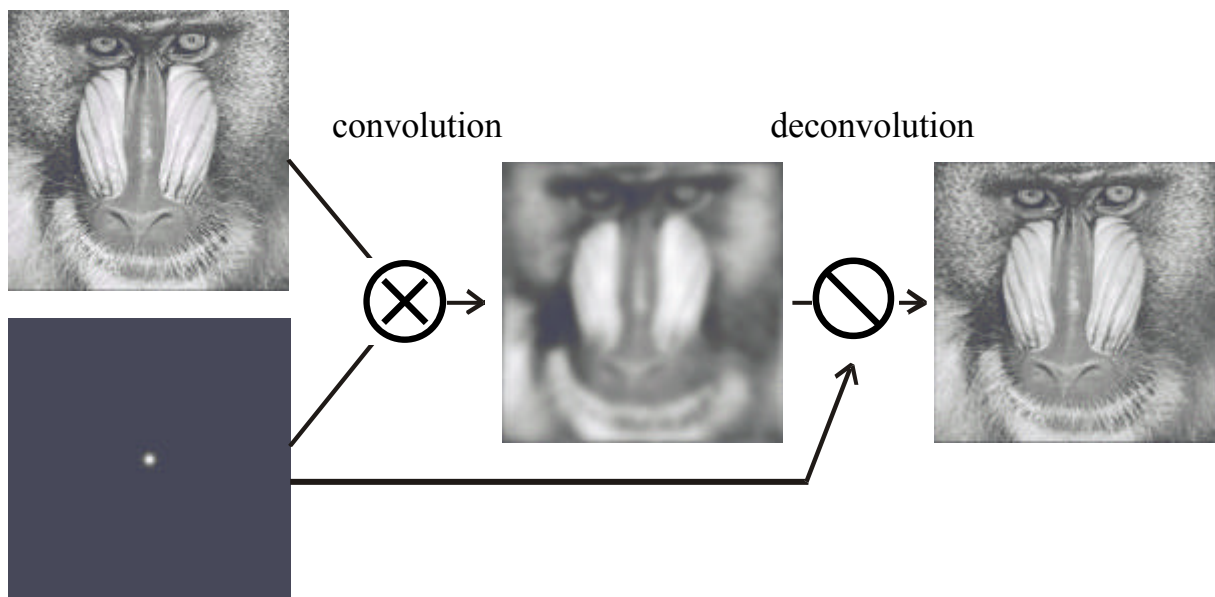
Therefore, deconvolution always needs good (low noise) data. Usually an noise-reduction step is included in deconvolution algorithms in a pre-processing step.

6.3. Application of Deconvolution

Deblurring:

Like in the 1D case deconvolution can be used to deblur or sharpen images. If the image was for instance obtained by a camera, which was out of focus. If we can reconstruct the blurring (convolution) function, we can also deconvolve the image (if it is not too noisy).

The 2D-result looks much better than the 1D-case. This is because our brain is used to look at 2D-images and interpolates and connects neighbouring information (we simply have more information to reconstruct the image in our mind).

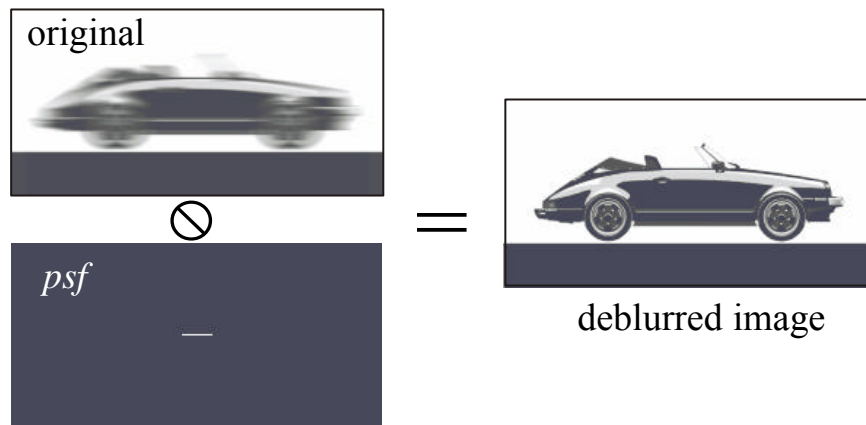


Further examples: MATLAB workshop.

Motion artefacts:

Images which are blurred due to motion can also be described by a convolution, where the static image is convoluted with a 'line' representing the motion of the camera or object during acquisition or exposure. If we know the relative distance during exposure and angle of motion it is straightforward to reconstruct the image.

Alternatively, it can be reconstructed from the blurred image itself....**HOW?**

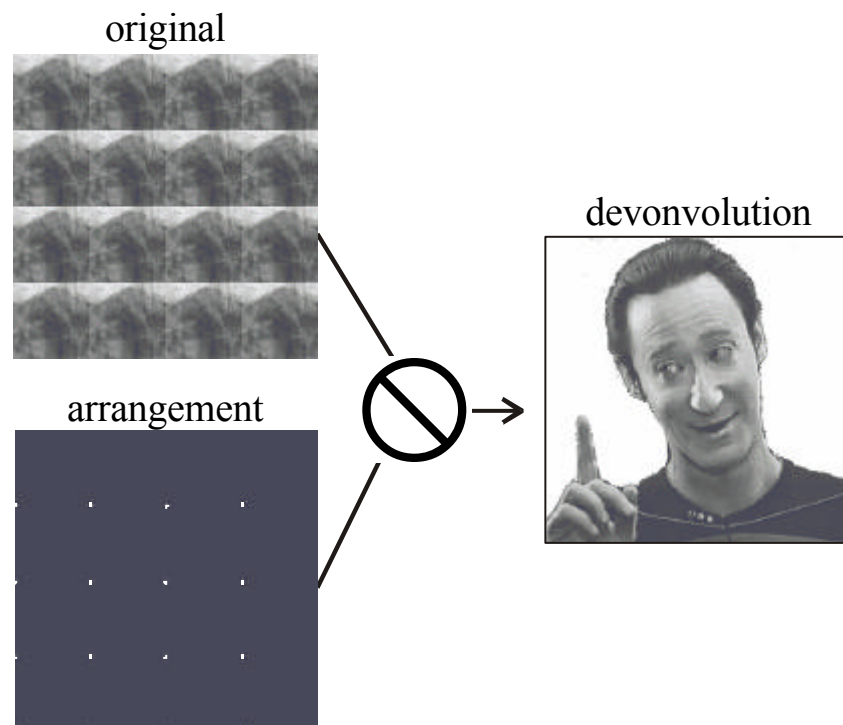


Further examples: MATLAB workshop.

Overlapping Data:

Sometimes images are also obstructed by copies of themselves (often referred to as 'ghosts'). This can be caused by reflections or misadjusted sampling intervals or physical principle (think for instance of an x-ray of many identical objects at different positions in the image plane).

If the arrangement of the objects is known (and they are identical) an arrangement matrix (image) can be constructed which has a 1 (or different weighting) at the centre of mass of the object or another point of reference. Deconvolution of the original and this arrangement matrix then yields the 'single image'.



7. Other Transforms

7.1 Some other Important Transforms:

transform	reference	usage in image processing
Cosine-transform	[1] p. 143ff.	data compression
<i>Abel/Hankel</i> -transform	[2] p.244ff.	Reconstruction of 1D-projections into Cartesian 1D- 'cross-sections'.
<i>Radon</i> -transformation	[3] 517ff.	Reconstruction of cylindrical projections of a 2D- object into 2D-Cartesian co-ordinates

[1] R. C. Gonzalez and R. E. Woods: „*Digital Image Processing*“, Addison-Wesley Publ., New York 1993.

[2] R. N. Bracewell: „*The Fourier Transform and Its Application*“, McGraw-Hill, New York, 1986.

[3] R. N. Bracewell: „*Two-Dimensional Imaging*“, Prentice Hall, New Jersey, 1995.

7.2. The Hadamard Transform

This is one of the less-known transformations. However, it will be discussed in more detail, because it has some very important applications for image transmission in space-science!

7.2.1 Definition of the Hadamard transformation:

The *Hadamard* transform (symbol \mathfrak{H}) $g_{n,n}(x, y)$ of a discrete image (matrix*) $f_{n,n}(x, y)$ (consisting of n rows and n columns, where n is a power of 2) is given by the multiplication with the *Hadamard* matrix $\mathbf{H}_{n,n}$.

The inverse *Hadamard* transform (symbol $\overline{\mathfrak{H}}$) is given by the multiplication with the transpose *Hadamard* matrix $\mathbf{H}'_{n,n}$.

$$g_{n,n}(x, y) = \mathfrak{H} f_{n,n}(x, y) = \mathbf{H}_{n,n} f_{n,n}(x, y) \quad [7.1]$$

$$f_{n,n}(x, y) = \overline{\mathfrak{H}} g_{n,n}(x, y) = \frac{1}{n} \mathbf{H}'_{n,n} g_{n,n}(x, y) \quad [7.2]$$

$$f_{n,n}(x, y) = \overline{\mathfrak{H}} (\mathfrak{H} f_{n,n}(x, y)) = \frac{1}{n} \mathbf{H}'_{n,n} (\mathbf{H}_{n,n} f_{n,n}(x, y)) \quad [7.3]$$

* to be completely rigorous we should have introduced images as matrices as well (so $\mathbf{f}_{n,n}(x, y)$ rather than $f_{n,n}(x, y)$ and $\mathbf{g}_{n,n}(x, y)$ rather than $g_{n,n}(x, y)$ would be more appropriate!)

Note: That this time we have to perform a matrix multiplication (MATLAB `*`) rather than a dyadic product (MATLAB `.*`)!

So what is then a *Hadamard-matrix*?

The most simple *Hadamard-matrix* is for $n=2$: $\mathbf{H}_{2,2} = \begin{pmatrix} +1 & +1 \\ +1 & -1 \end{pmatrix} \equiv \begin{pmatrix} + & + \\ + & - \end{pmatrix}$ [7.4]

where „+“ represents 1 and „-“ = -1.

All other *Hadamard-matrices* can be constructed from $\mathbf{H}_{2,2}$ via the following recursion formula:

$$\mathbf{H}_{2n,2n} = \begin{pmatrix} +\mathbf{H}_{n,n} & +\mathbf{H}_{n,n} \\ +\mathbf{H}_{n,n} & -\mathbf{H}_{n,n} \end{pmatrix} \quad [7.5]$$

Note the same pattern as in $\mathbf{H}_{2,2}$ (all you must remember is $\mathbf{H}_{2,2}$).

$$\text{E.g. } \mathbf{H}_{4,4} = \begin{pmatrix} +\mathbf{H}_{2,2} & +\mathbf{H}_{2,2} \\ +\mathbf{H}_{2,2} & -\mathbf{H}_{2,2} \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} + & + \\ + & - \end{pmatrix} & \begin{pmatrix} + & + \\ + & - \end{pmatrix} \\ \begin{pmatrix} + & + \\ + & - \end{pmatrix} & -\begin{pmatrix} + & + \\ + & - \end{pmatrix} \end{pmatrix} = \begin{pmatrix} + & + & + & + \\ + & - & + & - \\ + & + & - & - \\ + & - & - & + \end{pmatrix}$$

and so on...

$$\mathbf{H}_{8,8} = \begin{pmatrix} +\mathbf{H}_{4,4} & +\mathbf{H}_{4,4} \\ +\mathbf{H}_{4,4} & -\mathbf{H}_{4,4} \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} + & + & + & + \\ + & - & + & - \\ + & + & - & - \\ + & - & - & + \end{pmatrix} & \begin{pmatrix} + & + & + & + \\ + & - & + & - \\ + & + & - & - \\ + & - & - & + \end{pmatrix} \\ \begin{pmatrix} + & + & + & + \\ + & - & + & - \\ + & + & - & - \\ + & - & - & + \end{pmatrix} & -\begin{pmatrix} + & + & + & + \\ + & - & + & - \\ + & + & - & - \\ + & - & - & + \end{pmatrix} \end{pmatrix} = \begin{pmatrix} + & + & + & + & + & + & + & + \\ + & - & + & - & + & - & + & - \\ + & + & - & - & + & + & - & - \\ + & - & - & + & + & - & - & + \\ + & + & + & + & - & - & - & - \\ + & - & + & - & - & + & - & + \\ + & + & - & - & - & + & + & - \\ + & - & - & + & - & + & + & - \end{pmatrix}$$

The result can be checked as follows:

- 1) The sum of all rows and columns must be zero, except for the first row and column, whose sum must give be equal to the rank.

$$\text{e.g. } \mathbf{H}_{8,8} = \begin{pmatrix} + & + & + & + & + & + & + & + \\ + & - & + & - & + & - & + & - \\ + & + & - & - & + & + & - & - \\ + & - & - & + & + & - & - & + \\ + & + & + & + & - & - & - & - \\ + & - & + & - & - & + & - & + \\ + & + & - & - & - & + & + & - \\ + & - & - & + & - & + & + & - \end{pmatrix} \begin{matrix} \Sigma = 8 \\ \Sigma = 0 \\ \Sigma = 0 \\ \Sigma = 0 \\ \Sigma = 0 \\ \Sigma = 0 \\ \Sigma = 0 \\ \Sigma = 0 \end{matrix}$$

$$\Sigma = \begin{matrix} 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

- 2) *Hadamard* matrices are orthogonal, hence

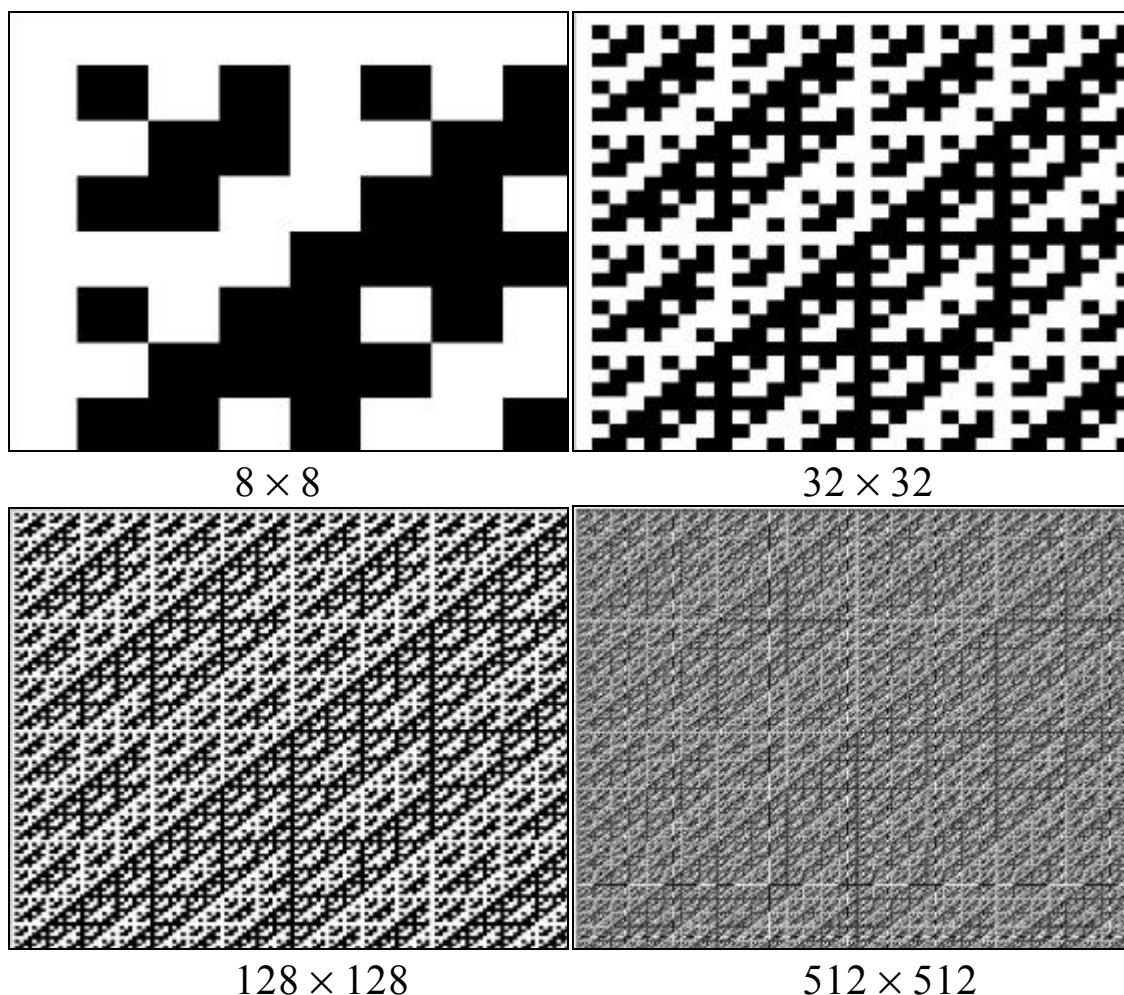
$$\mathbf{H}_{n,n} \mathbf{H}'_{n,n} = n \mathbf{I} \quad [7.6]$$

$$\text{e.g. } \mathbf{H}_{n,n} \mathbf{H}'_{n,n} = \begin{pmatrix} +1 & +1 \\ +1 & -1 \end{pmatrix} \begin{pmatrix} +1 & +1 \\ +1 & -1 \end{pmatrix} = \begin{pmatrix} 1+1 & 1-1 \\ 1-1 & 1+1 \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} = 2 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = 2\mathbf{I}$$

We also see that via definition in eq. [7.4] and recursion in eq. [7.5]:

$$\mathbf{H}_{n,n} = \mathbf{H}'_{n,n} \quad [7.7]$$

Note the symmetry! As also shown in the following graphs of *Hadamard* matrices (the beauty of maths!) where white = +1 and black = -1.



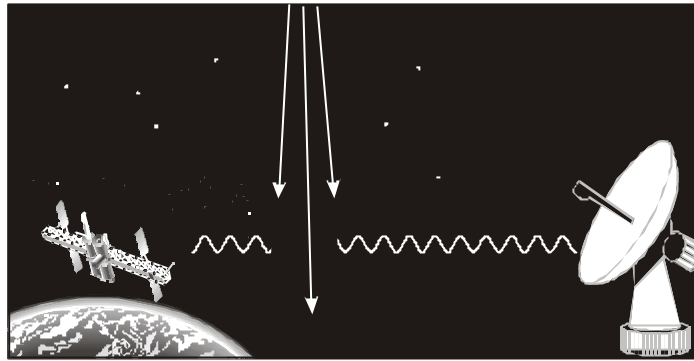
7.2.2 Application of the *Hadamard* transformation:

Hadamard transforms are used first of all because they are simple! Furthermore, they have the property that they spread the information content of a single pixel over two columns of its *Hadamard* transform. Hence, they represent an easy way to encode image information content over ‘more or less’ the entire image („holographic“).

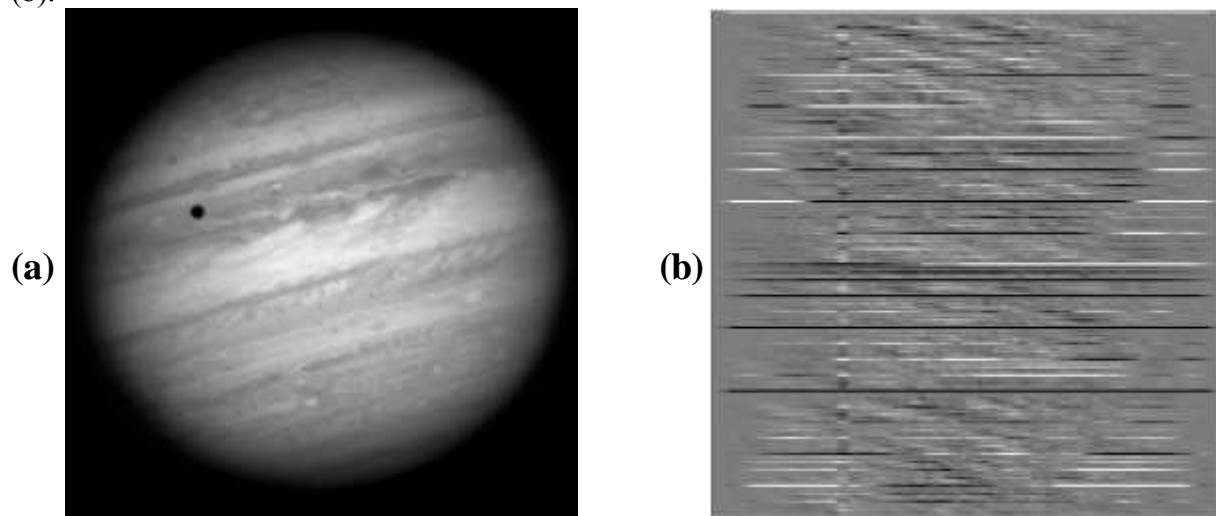
This can be used to ensure improved image reconstruction in case of data loss (When the data are send row after row!).

Imagine a space-probe is send to another planet or moon, from which it is sending images. The images (grey scales) could be transmitted as bytes (8-bit) for instance row after row until all data is transferred. However, it is quite likely that due to some events (e.g. solar wind, objects moving in the path, reflections etc.) there will be a interruption of the data flow.

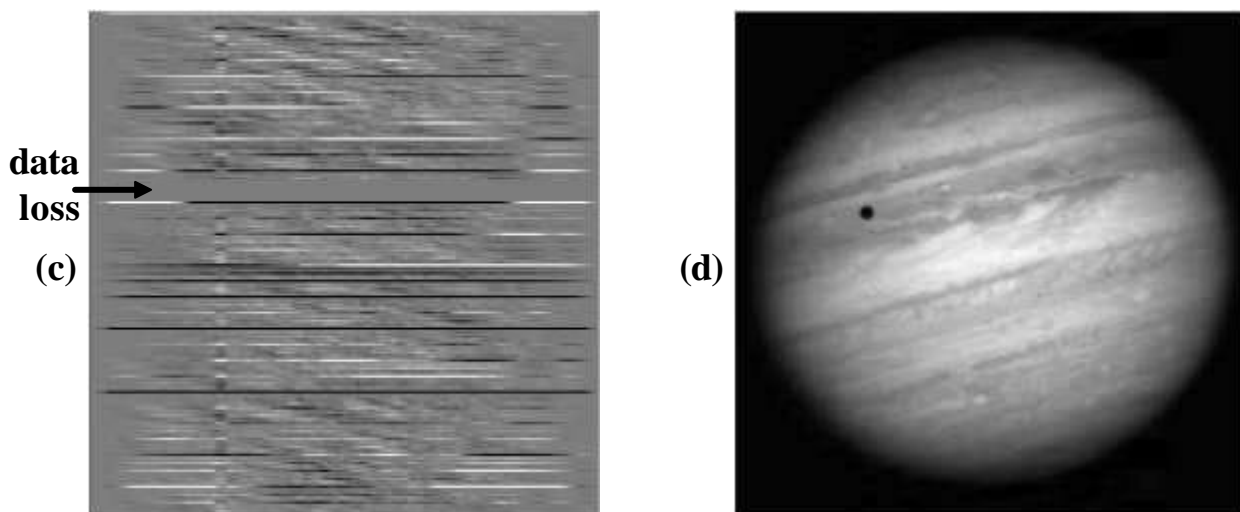
Hadamard transforms can be utilised to minimise the data-loss in such a situation.



The space-probe obtains the following image of Jupiter (a) and performs its *Hadamard*-transform (b).



Then the data in (b) is send row after row to the receiver. If some event corrupts the stream of data (c) (here represented by zeroes between 86th and 96th row) the inverse *Hadamard*-transform restores the image (d) as a whole (the loss of 4% data is almost invisible). If we wouldn't have used the *Hadamard*-transform in this case, the presence of the moon would have been lost in the image.



Other matrix transforms (e. g. *Walsh*, *Slant*, *Hotelling*...) can be found in (R. C. Gonzalez and R. E. Woods: „*Digital Image Processing*“, Addison-Wesley Publ., New York 1993, page 128 ff.).

Acknowledgements:

I would like to thank Dr. Morley R. Halse, who gave this course before me, for his excellent notes which became the foundation of this script.

Additionally, Prof. Dr. Hans-Wolfgang Spieß and Prof. Dr. Bernhard Blümich have to be acknowledged, who 'forced' me into convolution with the subject of this course during my very enjoyable time as their Ph.D. student.