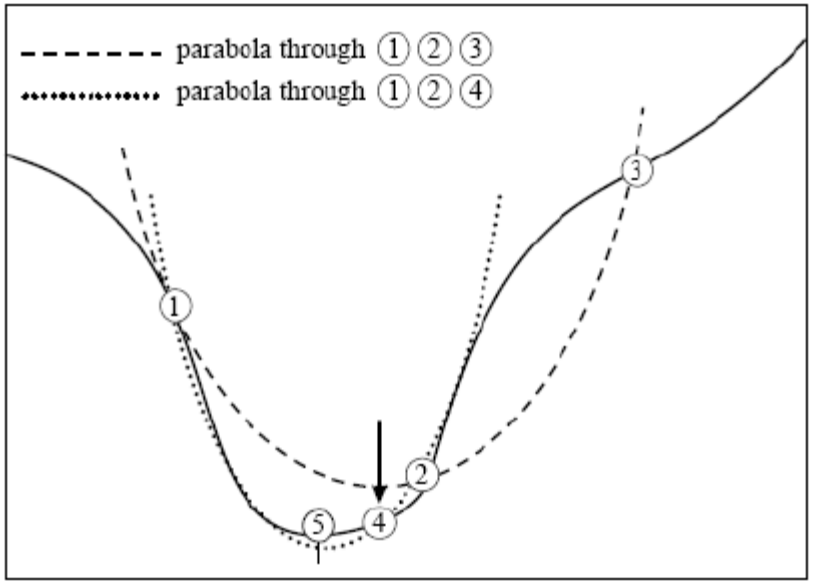
Lecture 7

Fourier Analysis

Summary Lecture 6

- Minima and maxima
  - 1 dimension :
    - Bracket minima : 3 values of f(x) : f(2) < f(1) and f(2)<f(3)
    - Minimum can be hunted down: e.g. Golden Mean (linear convergence).
    - Better: Brents method, parabolic interpolation



Summary Lecture 6

Multiple dimensions:

Complicated. Simplex method to enclose a minimum in N dimensions. Can be hunted down.

Use of derivatives: often known with less precision than function itself!

Multiple dimensions: linearization along all gradients. Powells method.

Strategies depend on order in which mi
schemes. NR: Fletcher-Rieves-Polak-Rib

Conjugate directions.

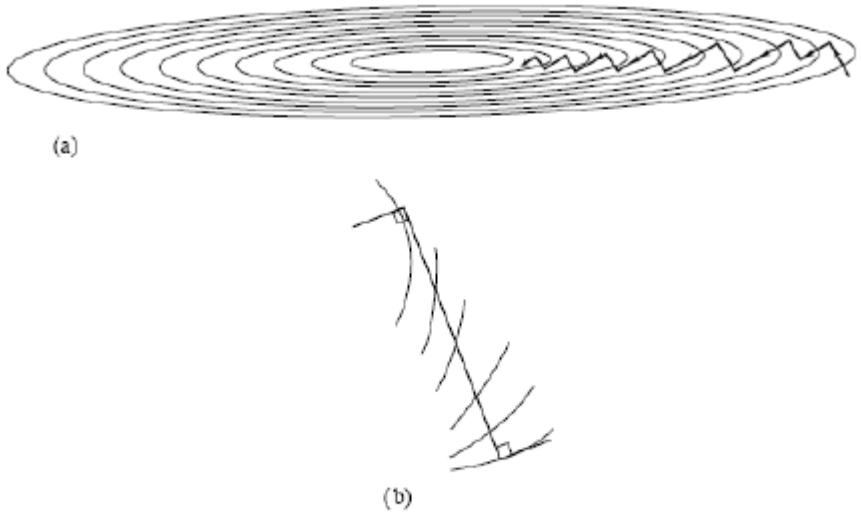
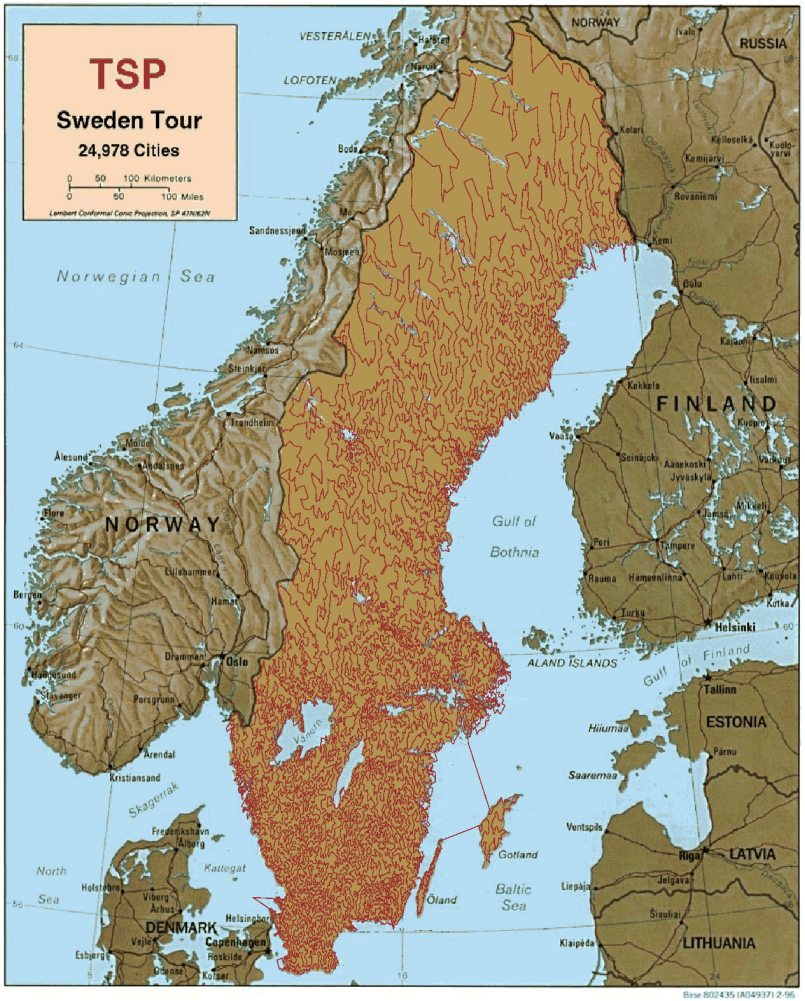


Figure 10.6.1. (a) Steepest descent method in a long, narrow "valley." While more efficient than the strategy of Figure 10.5.1, steepest descent is nonetheless an inefficient strategy, taking many steps to reach the valley floor. (b) Magnified view of one step: A step starts off in the local gradient direction, perpendicular to the contour lines, and traverses a straight line until a local minimum is reached, where the traverse is parallel to the local contour lines.

Summary Lecture 6

- Traveling salesman – like problems
- Simulated annealing : changing configurations in a random manner to obtain optimal result
  - needs an objective function (e.g. the total round-trip pathlength) to be minimized
  - needs a way to change paths:
    - E.g. swapping the order of 2 cities
  - Local minima can only be avoided by accepting “up-hill” steps. (accept longer paths with probability exp(-dL/T), T a “temperature”, e.g. 50 km
  - Schemes need experimenting.



Fourier Analysis

- Many applications:
  - switch to conjugate coordinates
    - time – frequency (electronics)
    - Classical/quantum mechanics -> coordinate – momentum space
  - signal processing
    - lock-in amplification
    - filtering
    - sounds : MP3, cd 44 kHz
    - image processing.
    - data reduction
    - wavelets
    - gravitational waves : Virgo/Ligo can detect up to 10^{-27} deformations in spacetime by measuring for a year.
  - convolution of functions
- all thanks to fast Fourier transform (FFT)

Fourier transformation

$$h(t) \Leftrightarrow H(f) \quad , \quad H(f) = \int_{-\infty}^{\infty} h(t) e^{2\pi i f t} dt \quad , \quad h(t) = \int_{-\infty}^{\infty} H(f) e^{-2\pi i f t} df$$

linear: $\{h(t) + g(t)\} \Leftrightarrow \{H(f) + G(f)\}$

- Symmetries:

$h(t)$ real $\Leftrightarrow$ $H(-f) = [H(f)]^*$

$h(t)$ imaginary $\Leftrightarrow$ $H(-f) = -[H(f)]^*$

$h(t)$ even $\Leftrightarrow$ $H(-f) = H(f)$

$h(t)$ odd $\Leftrightarrow$ $H(-f) = -H(f)$

$h(t)$ real, even $\Leftrightarrow$ $H(f)$ real, even

$h(t)$ real, odd $\Leftrightarrow$ $H(f)$ imaginary, odd

$h(t)$ imaginary, even $\Leftrightarrow$ $H(f)$ imaginary, even

$h(t)$ imaginary, odd $\Leftrightarrow$ $H(f)$ real, odd

Transformation Laws

- scaling: $h(at) \Leftrightarrow \frac{1}{|a|} H\left(\frac{f}{a}\right)$
- translation: $h(t-t_0) \Leftrightarrow H(f)\, e^{2\pi i t_0 f}$

  $H(f-f_0) \Leftrightarrow h(t) e^{-2\pi i f_0 t}$
- Convolution: $g \otimes h = \int_{-\infty}^{\infty} g(\tau) h(t-\tau) d\tau$

  $g \otimes h \Leftrightarrow G(f) H(f)$
- Correlation (lag) $Corr(g,h) = \int_{-\infty}^{\infty} g(t+\tau) h(\tau) d\tau$

  $Corr(g,h) \Leftrightarrow G(f) H^*(f)$

Fourier transformation

- Power in functions:

$$\text{Total power: } \int_{-\infty}^{\infty} |h(t)|^2 dt = \int_{-\infty}^{\infty} |H(f)|^2 df$$

- power between f and (f+df): typically one uses the one-sided power,

$$P_h(f) = |H(f)|^2 + |H(-f)|^2 \quad \Leftrightarrow \quad \text{total power} = \int_0^{\infty} P_h(f) df$$

- Note, in FFT algorithms typically df and dt are unknown and the normalization 1/N should be applied by the user.
- this gives a factor of two for real h(t):

$$\int_{-\infty}^{\infty} |H(f)|^2 df = \int_0^{\infty} P(f) df \Leftrightarrow P(f) = 2|H(f)|^2$$

Sampling, Nyquist frequency

- Often, h(t) is sampled in intervals delta:

$$h(t) \to h_n \quad h_n = h(n\Delta)$$

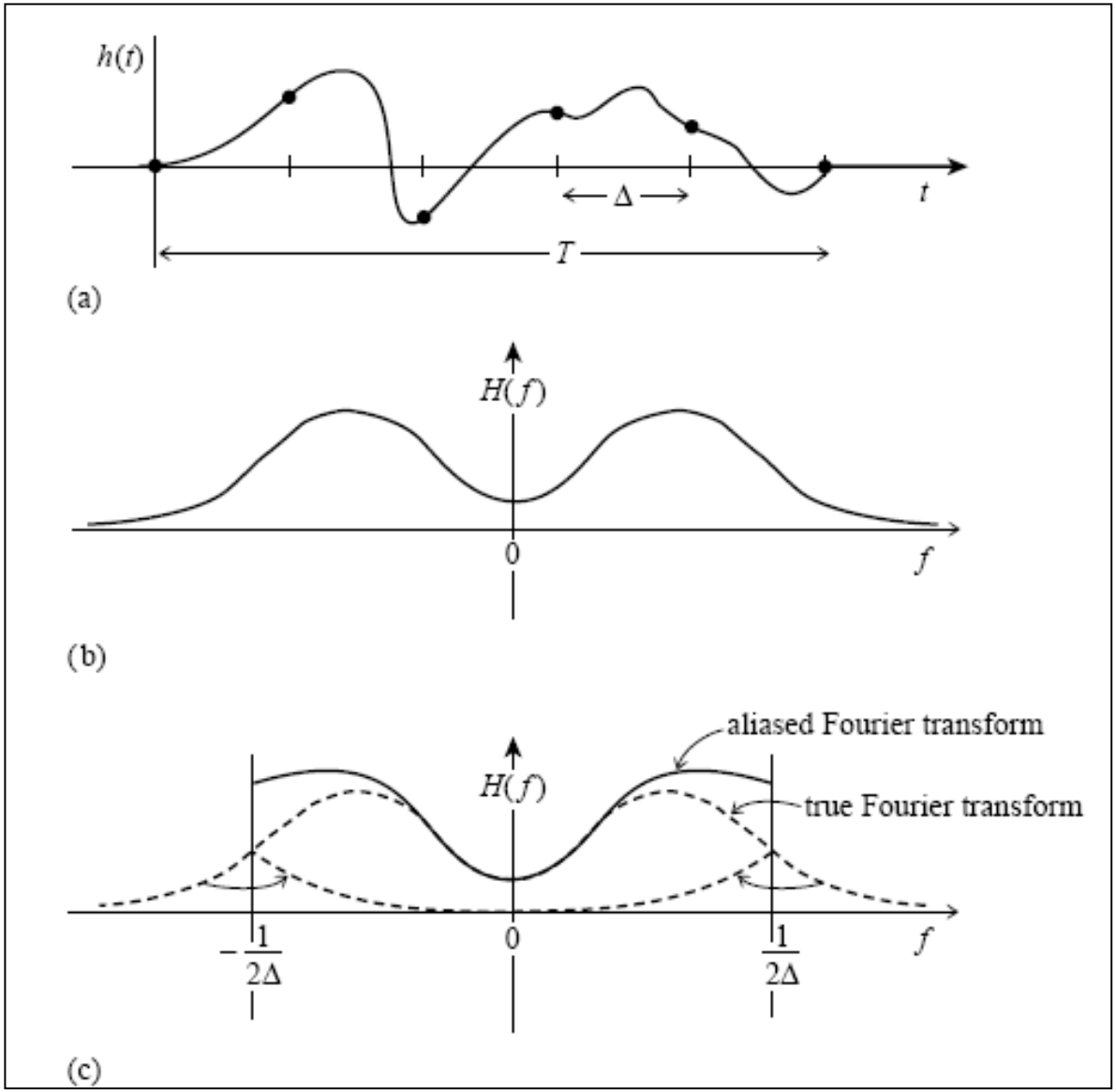
$$f_c \equiv \frac{1}{2\Delta}$$

- f_c is the so-called Nyquist frequency. If a function is band-width limited to frequencies below the Nyquist frequency, then the sampled function h_n is an EXACT representation of the original function (infinite information compression)

$$h(t) = \Delta \sum_{n=-\infty}^{\infty} h_n \frac{\sin(2\pi f_c(t - n\Delta))}{\pi(t - n\Delta)}$$

- If h(t) is NOT bandwidth-limited, the higher frequencies will be spuriously moved to lower frequencies.

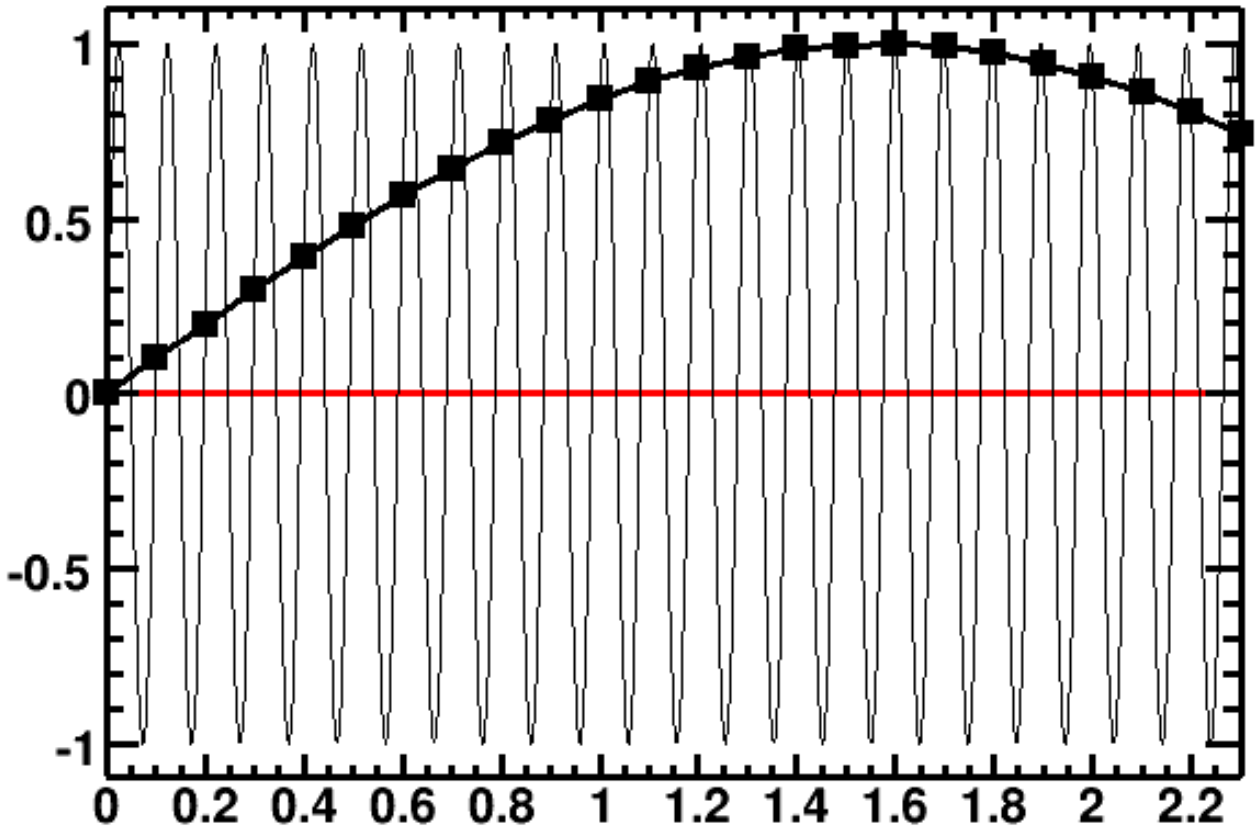
Sampling, aliasing



aliasing

Sin(x+20πx) sampled with 1000Hz and 10Hz.

At 10Hz, sin(x) and sin(x+20πx) look identical



Discrete Fourier transform

- Go from discrete number of samples h_n to the discrete Fourier transform H_n:

$$\text{N samples } h_n \text{ define N frequencies } f_n = \frac{n}{N\Delta}, \quad n = -\frac{N}{2}, \ldots, \frac{N}{2}$$

$$H(f_n) = \int h(t) e^{2\pi i t f_n} dt \approx \sum_{k=0}^{N-1} h_k e^{2\pi i t_k f_n} \Delta = \Delta \sum_{k=0}^{N-1} h_k e^{2\pi i k n / N} \equiv \Delta H_n$$

- H_n is periodic with period N; $H_{-n} = H_{N-n}$
- one usually replaces negative frequencies with frequencies shifted up by the Nyquist frequency (n runs from 0 to N instead of –N/2 to +N/2).
- reverse Fourier transform:

$$h_k = \frac{1}{N} \sum_{k=0}^{N-1} H_n e^{-2\pi i k n / N}$$

Fast Fourier Transform

- Normal Fourier transform: N numbers h_n need to be transformed to N numbers H_n, for each number H you need N multiplications. Expected $O(N^2)$ matrix multiplications to transform vector h in vector H.
- This implies e.g. for a 1000x1000 pixel photo: 10^{12} operations.
- FFT: O(NlogN) operations (Cooley-Tukey, 1960-s).
- already mentioned by Gauss, 1805.
- Danielson-Lanczos: Fourier transform of N steps can be written in terms of 2 Fourier transforms with N/2 steps each. One of even-numbered, one of odd-numbered points

Fast Fourier Transform

Split Fourier transform into 2 halves:

$$F_k = \sum_{j=0}^{N-1} e^{2\pi ijk/N} f_j = \sum_{j=0}^{N/2-1} e^{2\pi ik(2j)/N} f_{2j} + \sum_{j=0}^{N/2-1} e^{2\pi ik(2j+1)/N} f_{2j+1}$$

$$= \sum_{j=0}^{N/2-1} e^{2\pi ikj/(N/2)} f_{2j} + \left(e^{2\pi i/N}\right)^k \sum_{j=0}^{N/2-1} e^{2\pi ikj/(N/2)} f_{2j+1} = F_k^e + \left(e^{2\pi i/N}\right)^k F_k^o$$

do this recursively:

$$F_k = F_k^e + \left(e^{2\pi i/N}\right)^k F_k^o = F_k^{ee} + W^{eo} F_k^{eo} + W^{oe} F_k^{oe} + W^{oo} F_k^{oo}$$

$$= .....F^{eeoeoeooe} + F^{eeoeoeooo} + ...$$

Start with $N=2^k$ points (else zero-pad the remainder).

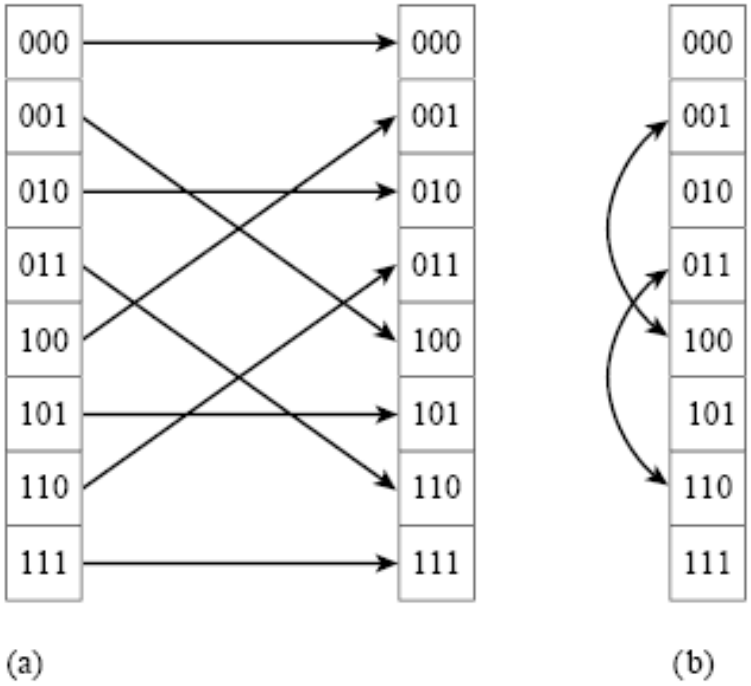
Fourier-transform of length 1: just a copy of the number ($F^{eeoeoeooe} = f_n$)

Fast Fourier Transform

- Which number n corresponds to $F^{eeoeoeoeoooeooee}$?
  - successive divisions of the data in an even and an odd part test for the lowest significant bit: the k^{th} index is o if $n\%2^k = 1$, e if it is 0.
  - reverse the pattern of e and o
    - set e=0, o=1
  - then you have the index n in binary.

FFT transform

- first step: reorder array by bit-reversing:

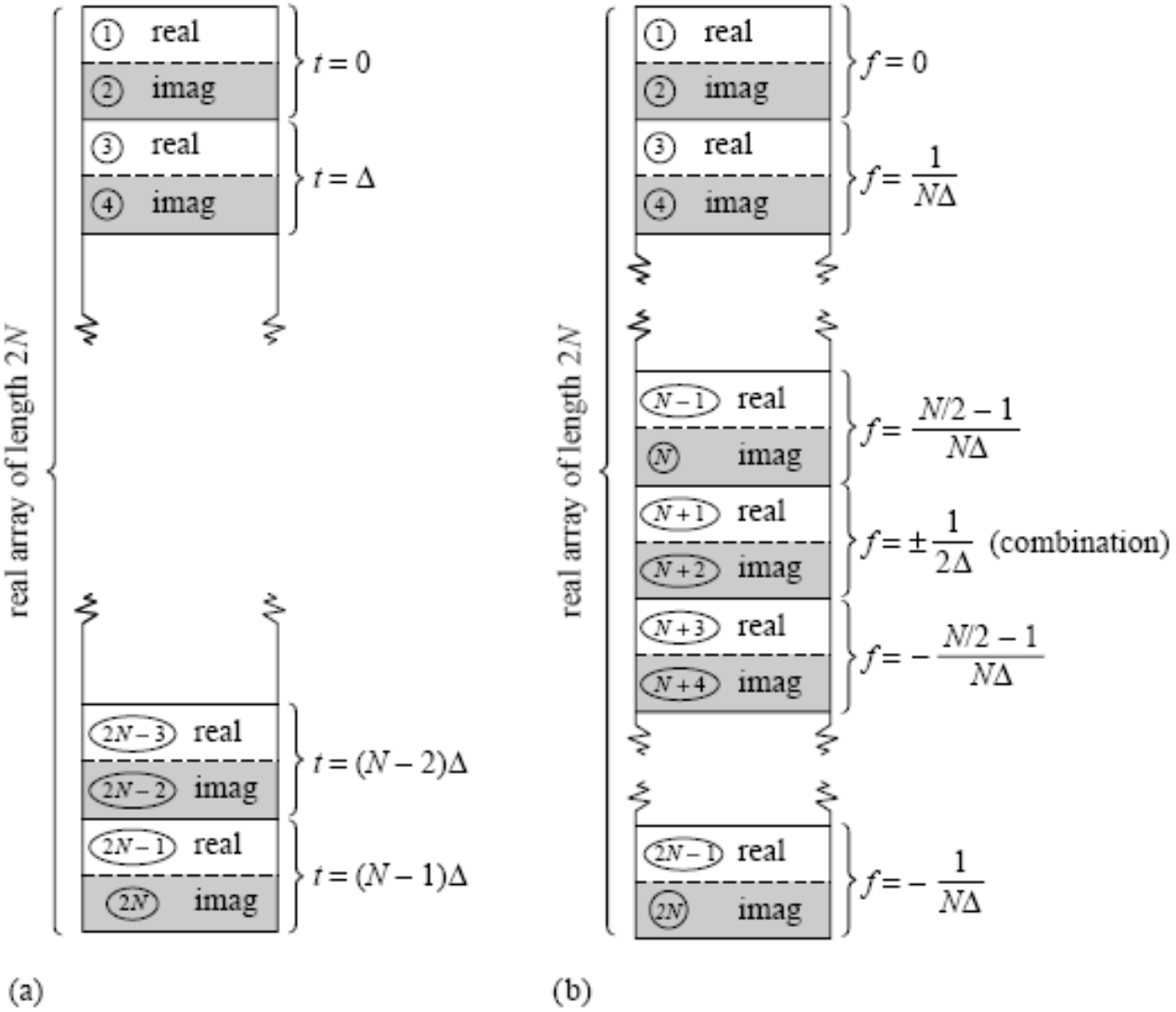


- second step: combine 2 adjacent points:

$$F_k^{xxx\ldots x} = F_k^{xxx\ldots xe} + w^k F_k^{xxx\ldots xo}$$

FFT

- repeat this step $^2\log(N)$ times, until the final FFT is obtained.
- Input and output definitions (general feature of FFT libraries, such as FTTW3):



FFT transforms

- Typically, complex number arithmetic is not implemented as efficient as real number arithmetic, and algorithms are fed with real arrays with 2N numbers instead of N complex numbers. If your function has a symmetry, one can reduce the time needed by the Fourier transform to fill the arrays differently. (E.g. the function is real: all imaginary terms are 0)
- Better: either do 2 FFT's at once (e.g. with convolution, filtering. Pack one function in the real numbers, the second in the imaginary numbers. To separate the resulting transformed complex functions, use $F_{N-n}=F_n^*$, $G_{N-n}=-G_n^*$.
- Alternatively: split odd and even samples, put odd samples in imaginary, and do the last step of the FFT yourself. Numerical recipes provides an algorithm for that too

multiple dimensions

- The numerical FFT can be obviously extended from one to more dimensions, e.g. $h(k_1,k_2)$ <-> $H(n_1,n_2)$:

$$h(k_1,k_2) \quad for \quad k_1 = 0,...,N_1-1 \quad k_2 = 0,...,N_2-1$$

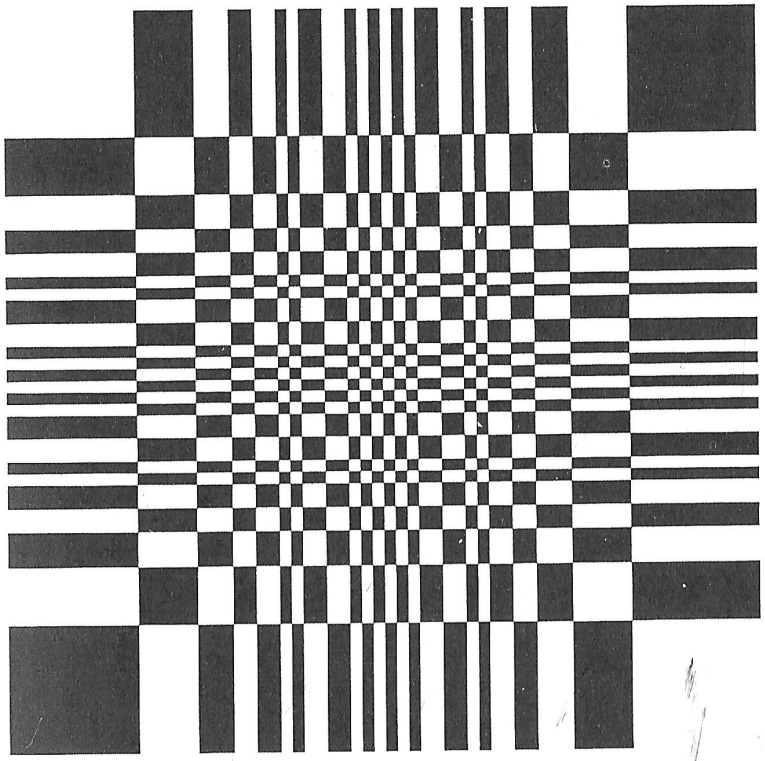
$$H(n_1,n_2) = \sum_{k_2=0}^{N_2-1} \sum_{k_1=0}^{N_1-1} e^{2\pi i k_2 n_2 / N_2} e^{2\pi i k_1 n_1 / N_1} h(k_1,k_2)$$

$$H(n_1,n_2) = FFT_2\{FFT_1[h(k_1,k_2)]\} = FFT_1\{FFT_2[h(k_1,k_2)]\}$$

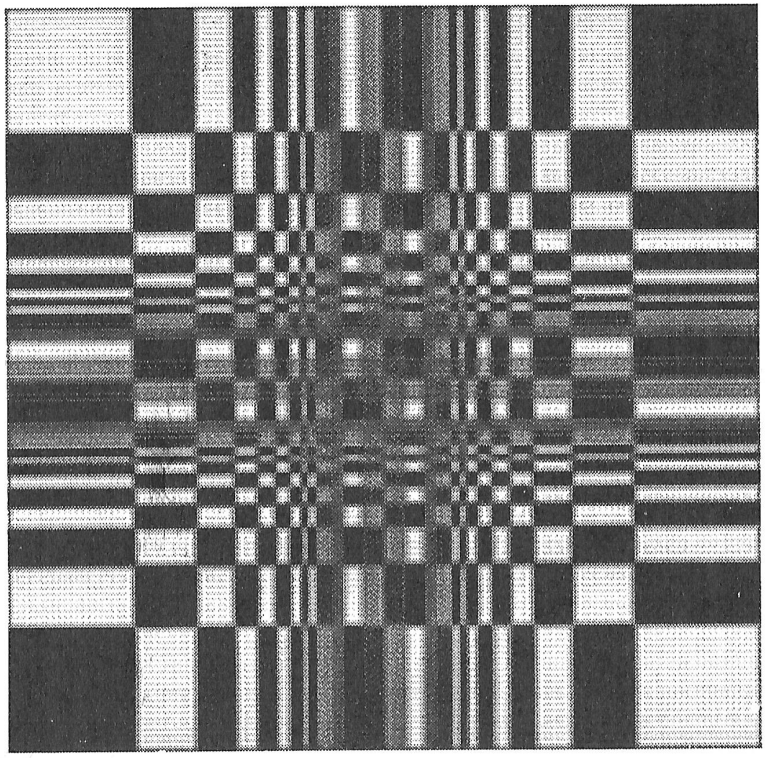
- lots of bookkeeping due to nested Fourier transforms; use a multi-dimensional FFT algorithm (NR::fourn).
- inverse transform: just normalize with the factor $1/(N_1N_2...N_x)$ after the transformation, and specify with switch that the complex factors are e^{-ic} instead of e^{+ic}

Two dimensions

image processing typically uses FFT techniques. The image pixels are presented as 2-dimensional functions. One could for instance “brighten” the whole picture by changing the value of H(0,0) and transforming back. One can sharpen the picture by deconvolution, or run it through a filter that filters out high or low frequency components.



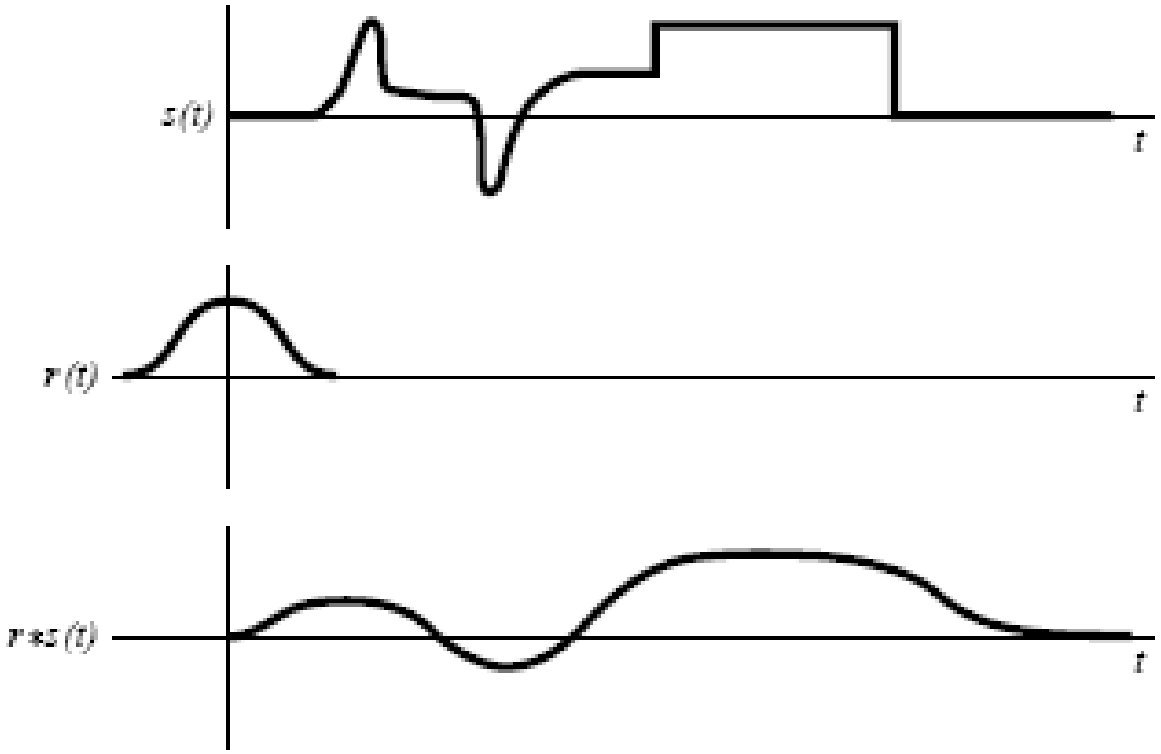
(a)



(b)

Convolution

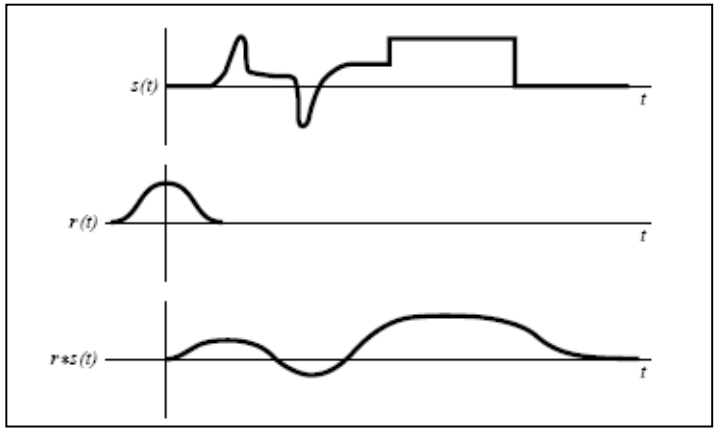
- r*s=s*r
- often, r and s are different, s signal (continuous data stream), r response (e.g. smearing, low-pass filter, ...r

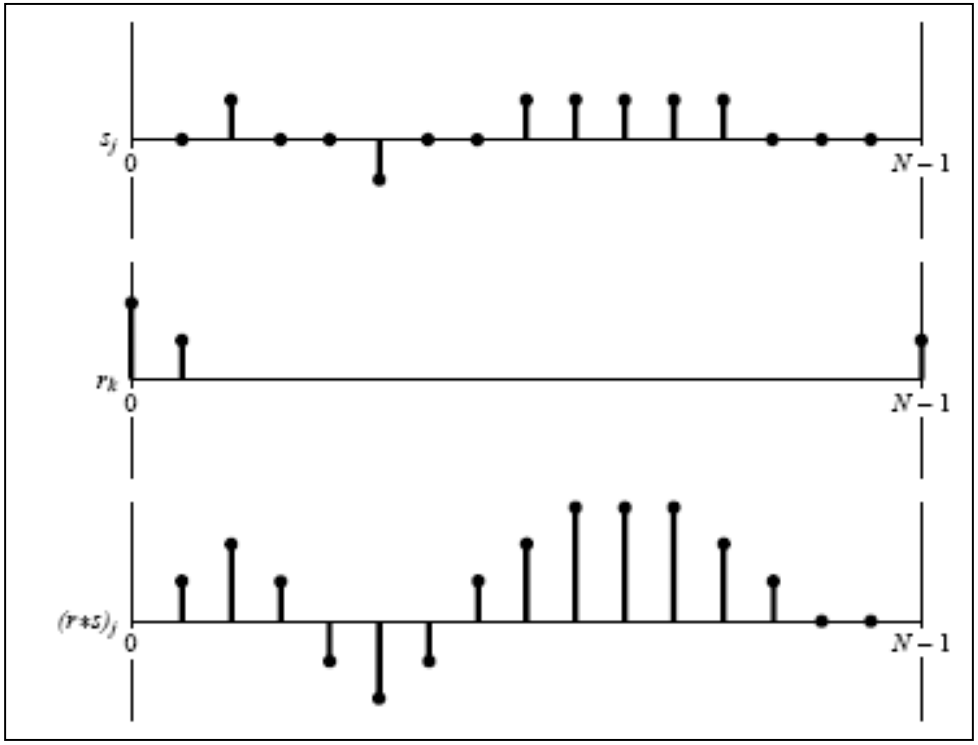


Convolution

Typically, the signal $s(t_0)$ is not transferred exactly as a delta function $r=\text{delta}(t-t_0)$, but smeared by a function $r(t-t_0)$.

Discrete case: s and r are sampled in steps Delta:





Convolution

- Hence, r_0 copies the input into the output, r_1 tells you which fraction of the input is smeared towards the next bin, $r_{-1} = R_{N-1}$ gives you how much is put into the previous bin, etc.
- discrete convolution:
- if r is non-zero only in finite range M, it is called a finite-impulse response (FIR)

$$r \otimes s = \int_{-\infty}^{\infty} r(\tau) s(t-\tau) d\tau$$

$$(r \otimes s)_j = \sum_{k=-M/2+1}^{M/2} s_{j-k} r_k$$

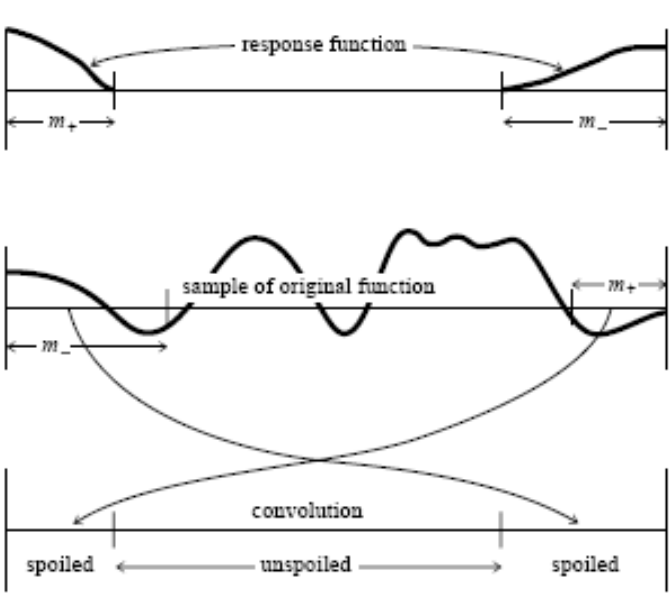
- discrete convolution theorem: if signal periodic with period N and FIR has length M<=N, then

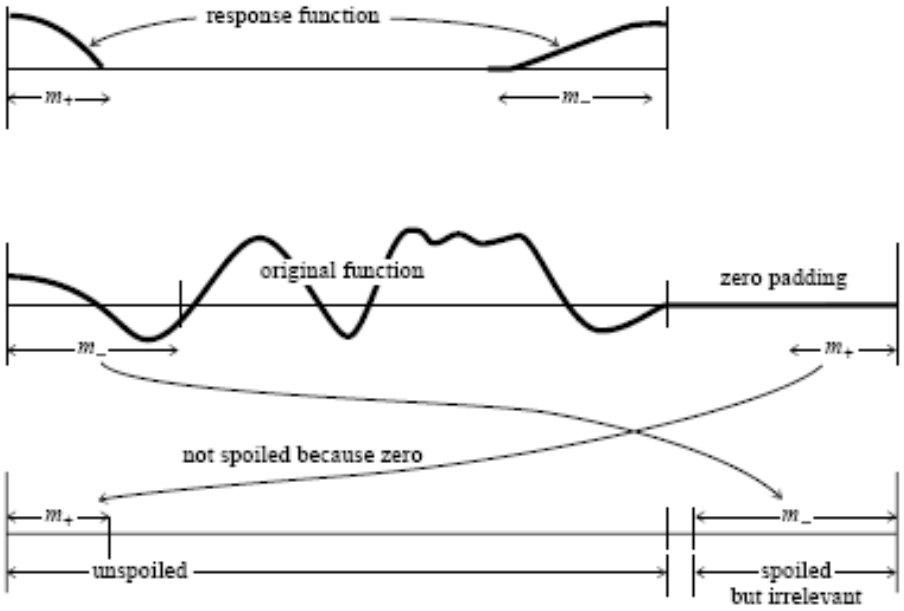
$$\sum_{k=-N/2+1}^{N/2} s_{j-k} r_k \Leftrightarrow S_n R_n$$

Convolution

- usually, FIR has much shorter length M. Treat this by inserting zeros.
- The index runs from 0 to N-1, just as before: s(-N/2)=s(N/2), s(-1)=s(N-1).
- The signal may not be periodic. Then, the result at s(0) is influenced by the measurement at s(N-1). Here, zero padding helps too. Insert after the last non-zero measurement at least M zero's (M is duration of FIR), then the results at s(0) are not influenced any more. discard the results for the last M values of s, the padded zero's. (They are determined by the measurements at s(0)...s(M-1)).

Convolution





- left: discrete convolution of non-periodic signal s. The regions at the end are spoiled.
- right: zero-padded result.

Convolution

- procedure:
  - data, with zero padding, consists of pairs s_j, r_j, j=0,...,N-1
  - discrete convolution is given by taking the Fourier transform of r and s, multiplying them term by term, then take the inverse Fourier transform. The result is (r*s)
- deconvolution: take transform of measured data and of response function. divide the transform of the measured data by the transform of the response function, and you obtain the deconvolved signal!
- if $R_n = 0$, this implies that the knowledge of the frequency component S_n is completely lost, so a reconstruction is not possible.

Correlation

- in correlation, the data sets are typically similar in shape. E.g. you search for a predefined signal, such as a gravitational wave signal in a binary collapse, in a data stream (matched filtering).
- Corr(g,h)(t)=Corr(h,g)(-t)
- discrete:

$$Corr(g,h)_j \equiv \sum_{k=0}^{N-1} g_{j+k} h_k = FFT(G_k H_k^*)$$

- zero padding is done as in the convolution method.

Optimal filtering

- Noise removal: signal u is modified by a response r and a noise contribution.
  - true signal u, convolved with response r gives signal s, with noise n added gives measurement c: c=(u*r)+n

$$s(t)=\int r(z)u(t-z)\,dz \Leftrightarrow S(f)=R(f)U(f)$$

$$c(t)=s(t)=n(t)$$

$$U(f)\approx\tilde{U}(f)=\frac{C(f)Z(f)}{R(f)}$$

- Z(f) is optimal filter when U-tilde is as close as possible to true function U:

$$\int |U^2(f)-\tilde{U}^2(f)|df$$ is minimized

Optimal filtering

- signal and noise are uncorrelated, so their cross-product is zero:

$$\int |U^2(f) - \tilde{U}^2(f)|df = \int |\frac{[S(f)+N(f)]Z(f)}{R(f)} - \frac{S(f)}{R(f)}|^2$$

$$= \int |R(f)|^{-2}\{|S(f)|^2|1-Z(f)|^2 + |N(f)|^2|Z(f)|^2\}df$$

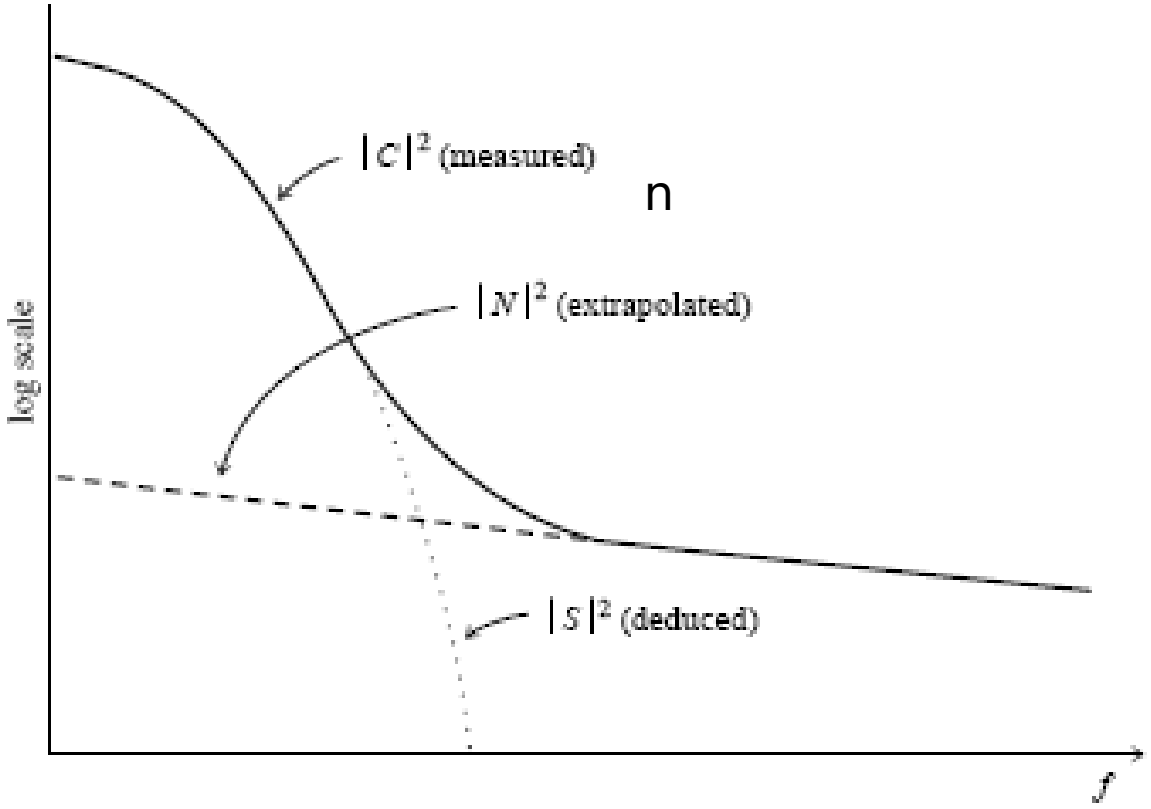
- The integrand is minimized if the derivative to Z(f) is zero.This leads to the optimal filter Z :

$$Z(f) = \frac{|S(f)|^2}{|S(f)|^2 + |N(f)|^2}$$

- Noise needs to be estimated; this is usually possible.

Optimal filtering

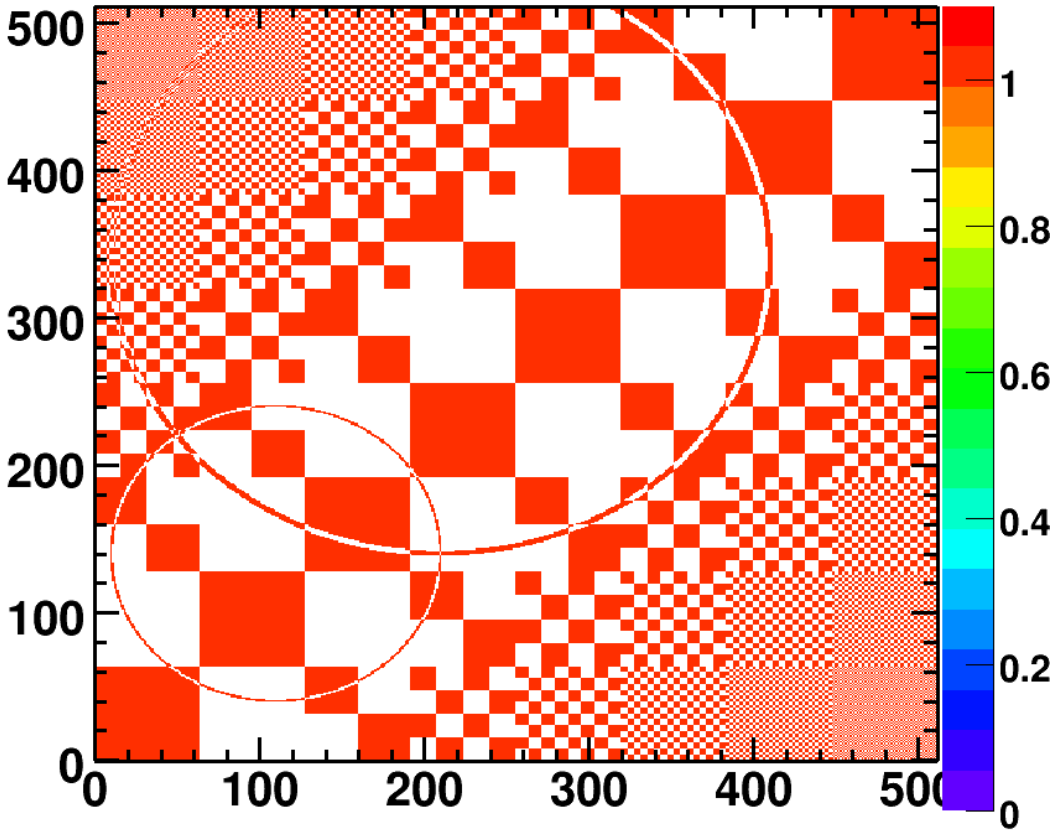
- Noise : either measured in absence signal, or extrapolated under signal region.



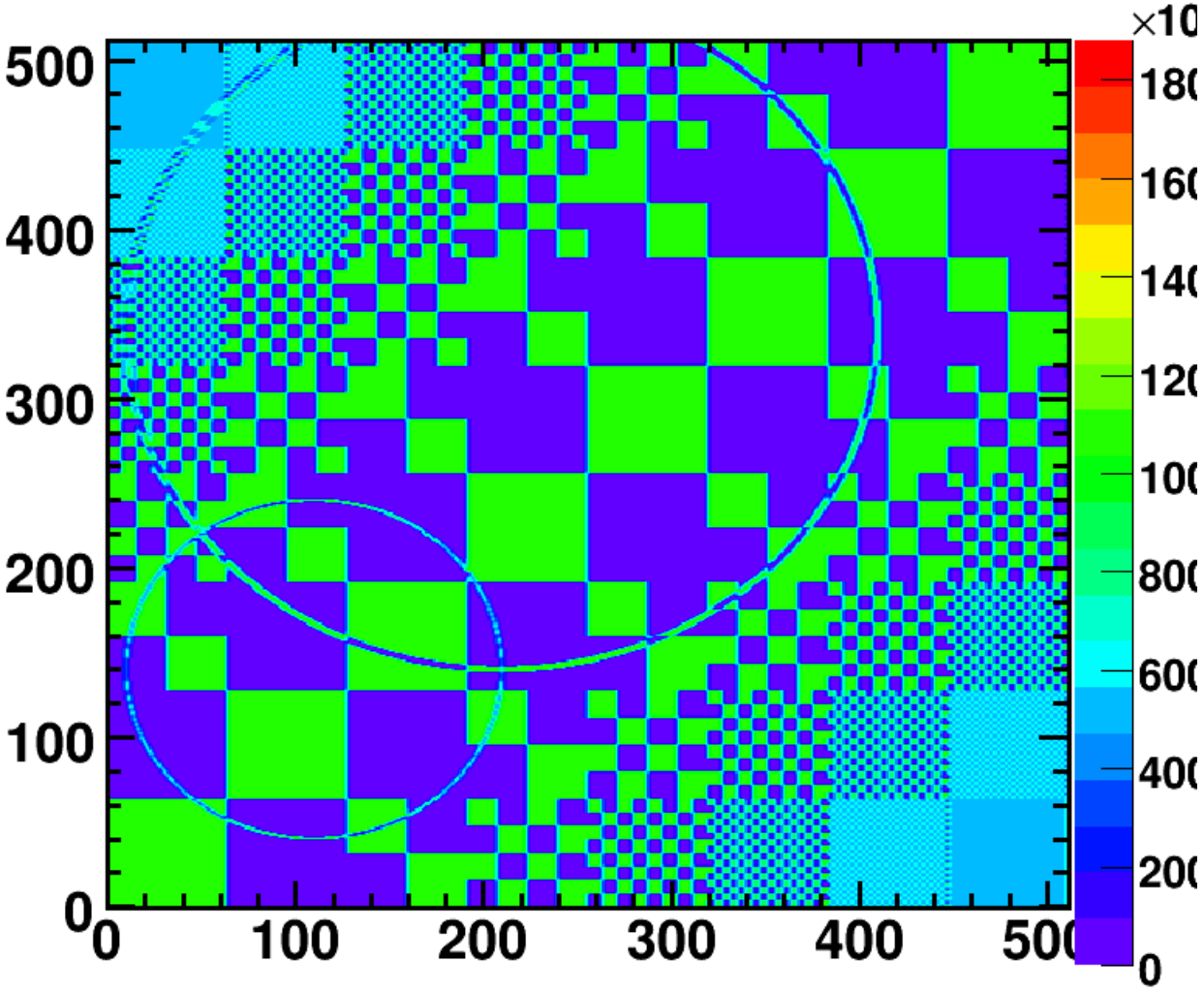
Example deconvolution

- To show how the algorithms works I made an example, creating a 512x512 pixel plot. The code creates files with 3 columns, x,y, and brightness. I plotted it by making 2D histograms in root.
- The original picture is smeared with a Gaussian distribution with different RMS in both directions, by making a smearing function in coordinate space, Fourier-transforming it and multiplying with the Fourier-transformed picture. This would in coordinate space constitute to 10^{12} operations (about 10 hours), but with the FFT it is limited to 10^{7} operations (fraction of a second)
- The result is deconvolved as well.
- The code produces files orig, smeared, and deconv

Original plot



Smeared plot

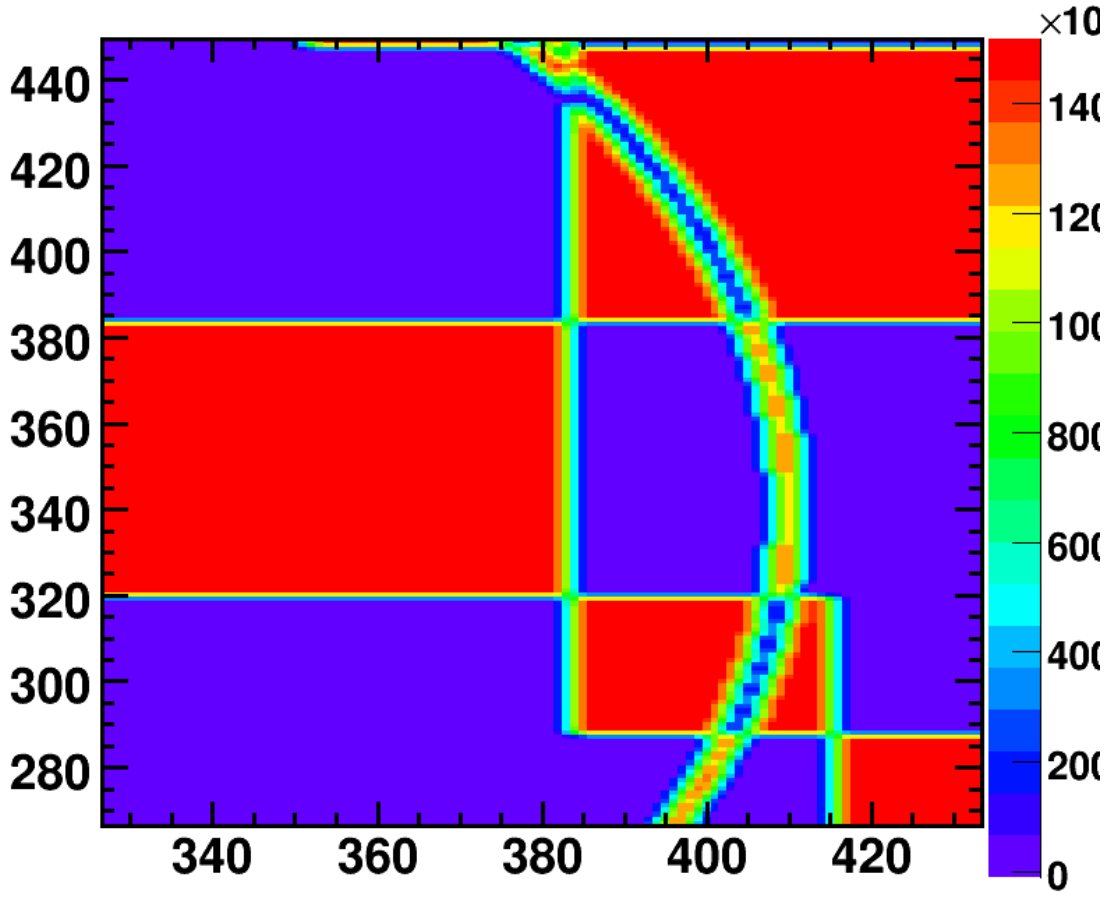


Note, that the Fourier-transform does not apply the normalization factor 1/N or 1/sqrt(N). This is typical for FFT libraries. The distance dt and df are not known, so it is up to the user to normalize.

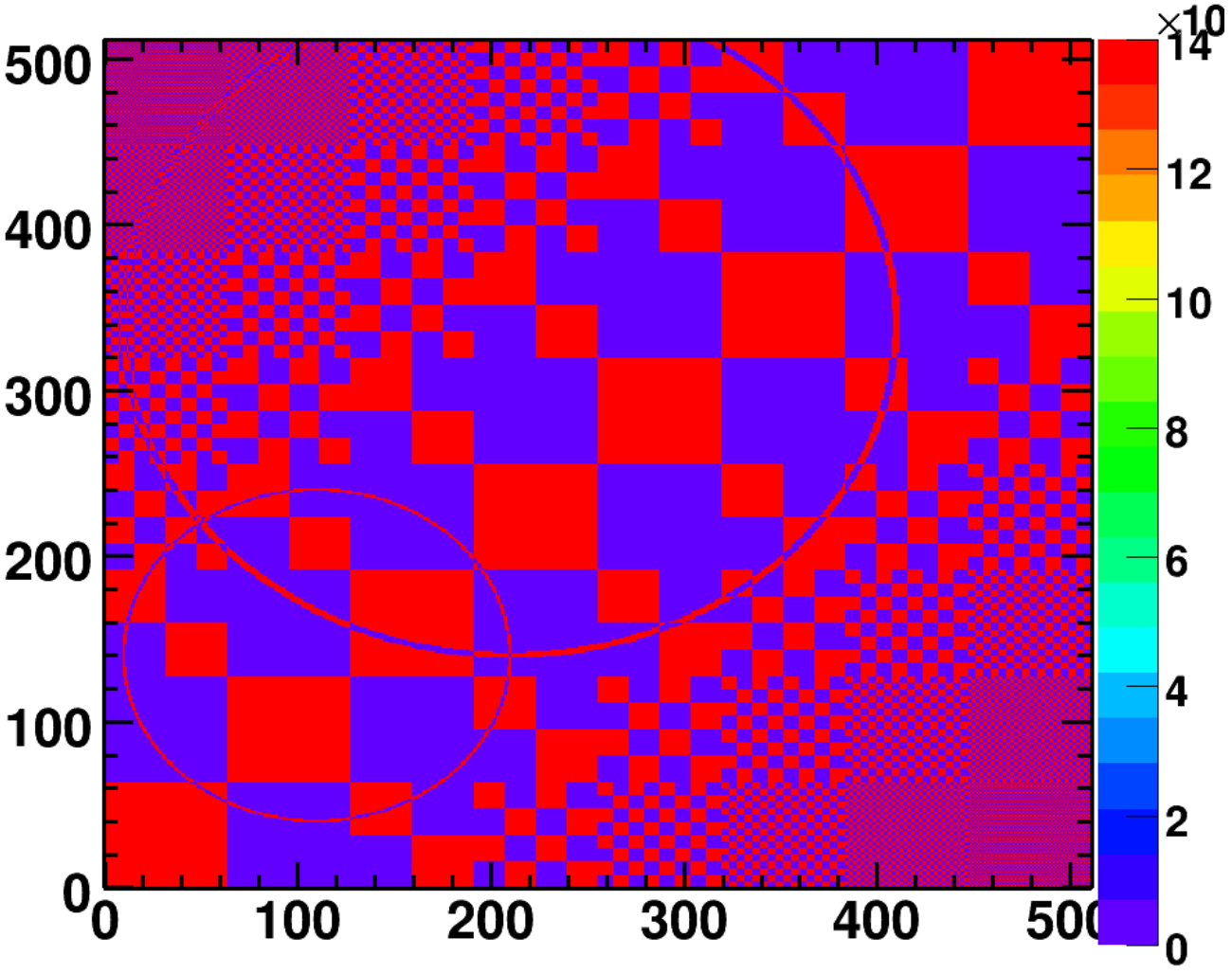
In order to have the sums of $(s(t)^2$ and $S(f)^2)$ the same, one could divide S_n by 1/N.

Verify the normalization of your FFTs yourself. (Apply it twice, make sure you get the original dataset back, and that the summed powers equal in time and frequency domain

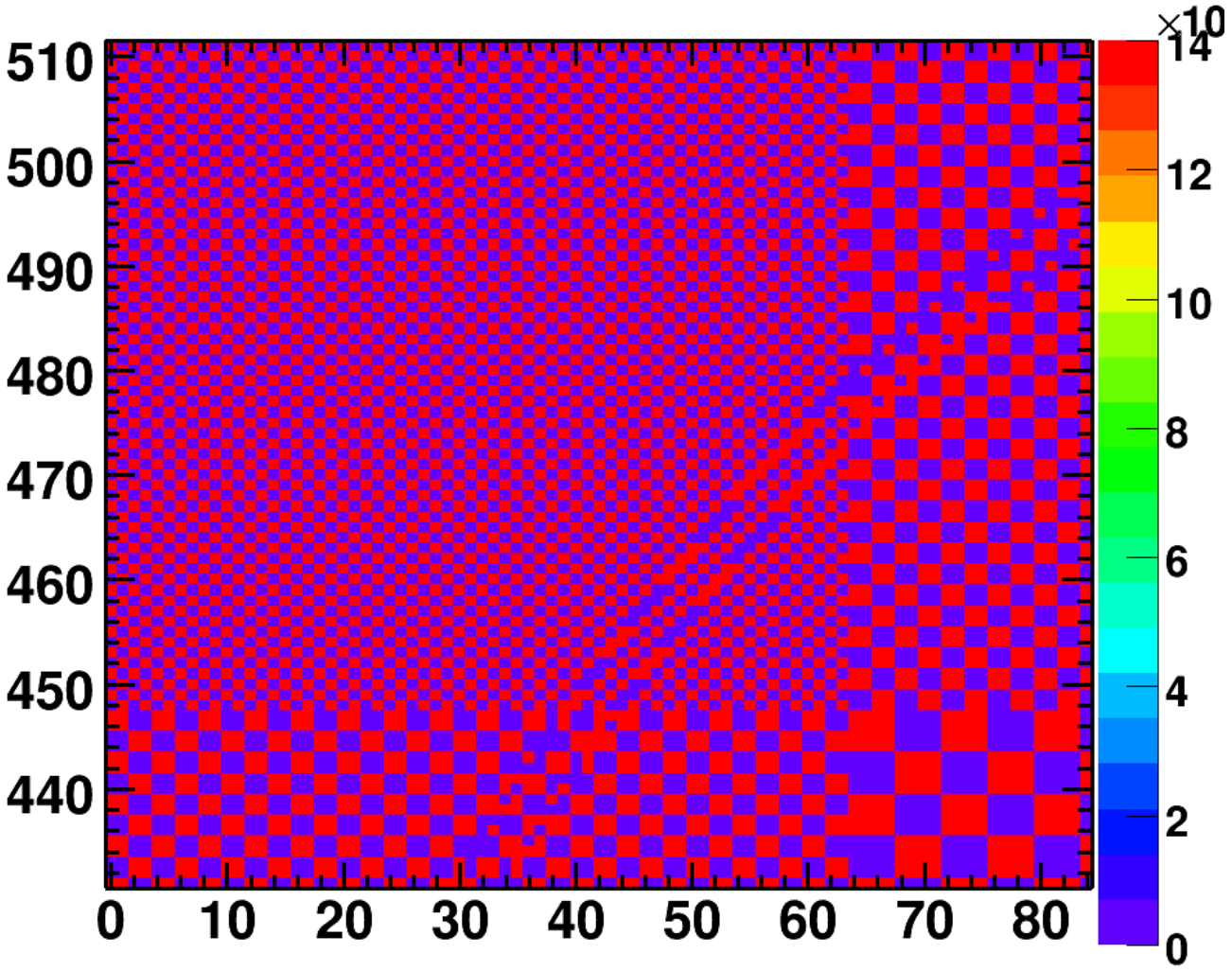
Smeared plot, detail



Deconvolved plot.



Deconvolved plot, detail



Exercise 6

- Exercise 6 will be about optimal filtering with the use of the Fourier transform.
- I will give more background on that next lecture; aspects like power spectral density and statistical details need another lecture.