

# Digital Signal Processing

Markus Kuhn

Department of Computer Science and Technology  
University of Cambridge

<https://www.cl.cam.ac.uk/teaching/2627/{DSP,L314}/>

Michaelmas 2026  
CST Part II/Part III/MPhil ACS module

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

- ▶ flow of information
- ▶ measured quantity that varies with time (or position)
- ▶ electrical signal received from a transducer (microphone, thermometer, accelerometer, antenna, etc.)
- ▶ electrical signal that controls a process

**Continuous-time signals:** voltage, current, temperature, speed, ...

**Discrete-time signals:** daily minimum/maximum temperature, lap intervals in races, sampled continuous signals, ...

Electronics (unlike optics) can only deal easily with time-dependent signals. Spatial signals, such as images, are typically first converted into a time signal with a scanning process (TV, fax, etc.).

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

Signals may have to be transformed in order to

- ▶ amplify or filter out embedded information
- ▶ detect patterns
- ▶ prepare the signal to survive a transmission channel
- ▶ prevent interference with other signals sharing a medium
- ▶ undo distortions contributed by a transmission channel
- ▶ compensate for sensor deficiencies
- ▶ find information encoded in a different domain

To do so, we also need

- ▶ methods to measure, characterise, model and simulate transmission channels
- ▶ mathematical tools that split common channels and transformations into easily manipulated building blocks

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

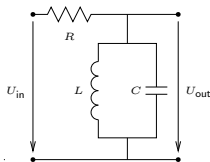
Random signals

Digital communication

Audiovisual data compression

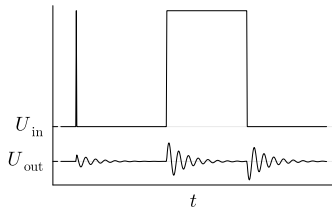
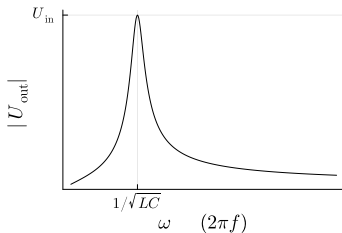
Outlook

Passive networks (resistors, capacitors, inductances, crystals, SAW filters), non-linear elements (diodes, ...), (roughly) linear operational amplifiers



## Advantages:

- ▶ passive networks are highly linear over a very large dynamic range and large bandwidths
- ▶ analog signal-processing circuits require little or no power
- ▶ analog circuits cause little additional interference



$$\frac{U_{in} - U_{out}}{R} = \frac{1}{L} \int_{-\infty}^t U_{out} d\tau + C \frac{dU_{out}}{dt}$$

## Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

Analog/digital and digital/analog converter, CPU, DSP, ASIC, FPGA.

## Advantages:

- ▶ noise is easy to control after initial quantization
- ▶ highly linear (within limited dynamic range)
- ▶ complex algorithms fit into a single chip
- ▶ flexibility, parameters can easily be varied in software
- ▶ digital processing is insensitive to component tolerances, aging, environmental conditions, electromagnetic interference

## But:

- ▶ discrete-time processing artifacts (aliasing)
- ▶ can require significantly more power (battery, cooling)
- ▶ digital clock and switching cause interference

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

## communication systems

modulation/demodulation, channel equalization, echo cancellation

## consumer electronics

perceptual coding of audio and video (DAB, DVB, DVD), speech synthesis, speech recognition

## music

synthetic instruments, audio effects, noise reduction

## medical diagnostics

magnetic-resonance and ultrasonic imaging, X-ray computed tomography, ECG, EEG, MEG, AED, audiology

## geophysics

seismology, oil exploration

## astronomy

VLBI, speckle interferometry

## transportation

radar, radio navigation

## security

steganography, digital watermarking, biometric identification, surveillance systems, signals intelligence, electronic warfare

## engineering

control systems, feature extraction for pattern recognition, sensor-data evaluation

### Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

By the end of the course, you should be able to

- ▶ apply basic properties of time-invariant linear systems
- ▶ understand sampling, aliasing, convolution, filtering, the pitfalls of spectral estimation
- ▶ explain the above in time and frequency domain representations
- ▶ use filter-design software
- ▶ visualise and discuss digital filters in the  $z$ -domain
- ▶ use the FFT for convolution, deconvolution, filtering
- ▶ implement, apply and evaluate simple DSP applications, e.g. in Julia
- ▶ apply transforms that reduce correlation between several signal sources
- ▶ understand the basic principles of several widely-used modulation and image-coding techniques.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

- ▶ R.G. Lyons: *Understanding digital signal processing*. 3rd ed., Prentice-Hall, 2010. (£73)
- ▶ Thomas Holton: *Digital signal processing – principles and applications*. Cambridge University Press, 2021. (£85)
- ▶ A.V. Oppenheim, R.W. Schaffer: *Discrete-time signal processing*. 3rd ed., Prentice-Hall, 2007. (£47)
- ▶ J. Stein: *Digital signal processing – a computer science perspective*. Wiley, 2000. (£133)
- ▶ S.W. Smith: *Digital signal processing – a practical guide for engineers and scientists*. Newness, 2003. (£48)
- ▶ K. Steiglitz: *A digital signal processing primer – with applications to digital audio and computer music*. Addison-Wesley, 1996. (£67)

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

- 1 Sequences and systems
- 2 Convolution
- 3 Fourier transform
- 4 Sampling
- 5 Discrete Fourier transform
- 6 Deconvolution
- 7 Spectral estimation
- 8 Digital filters
- 9 IIR filters
- 10 Random signals
- 11 Digital communication
- 12 Audiovisual data compression

Introduction

**Sequences and systems**

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

A *discrete sequence*  $\{x_n\}_{n=-\infty}^{\infty}$  is a sequence of numbers

$$\dots, x_{-2}, x_{-1}, x_0, x_1, x_2, \dots$$

where  $x_n$  denotes the  $n$ -th number in the sequence ( $n \in \mathbb{Z}$ ). A discrete sequence maps integer numbers onto real (or complex) numbers.

We normally abbreviate  $\{x_n\}_{n=-\infty}^{\infty}$  to  $\{x_n\}$ , or to  $\{x_n\}_n$  if the running index is not obvious. The notation is not well standardized. Some authors write  $x[n]$  instead of  $x_n$ , others  $x(n)$ .

Where a discrete sequence  $\{x_n\}$  samples a continuous function  $x(t)$  as

$$x_n = x(t_s \cdot n) = x(n/f_s),$$

we call  $t_s$  the *sampling period* and  $f_s = 1/t_s$  the *sampling frequency*.

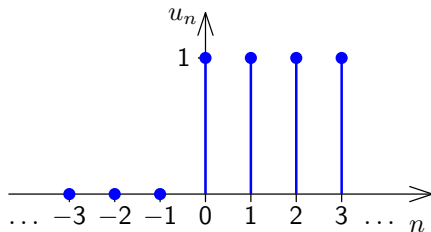
A *discrete system*  $T$  receives as input a sequence  $\{x_n\}$  and transforms it into an output sequence  $\{y_n\} = T\{x_n\}$ :

$$\dots, x_2, x_1, x_0, x_{-1}, \dots \longrightarrow \boxed{\text{discrete system } T} \longrightarrow \dots, y_2, y_1, y_0, y_{-1}, \dots$$

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

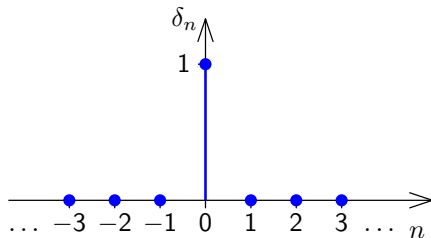
Unit-step sequence:

$$u_n = \begin{cases} 0, & n < 0 \\ 1, & n \geq 0 \end{cases}$$



Impulse sequence:

$$\begin{aligned} \delta_n &= \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases} \\ &= u_n - u_{n-1} \end{aligned}$$

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

A cosine wave, amplitude  $A$ , frequency  $f$ , phase offset  $\varphi$ :

$$x(t) = A \cdot \cos(\underbrace{2\pi f t + \varphi}_{\text{phase}})$$

Sampling it at sampling rate  $f_s$  results in the discrete sequence  $\{x_n\}$ :

$$x_n = A \cdot \cos(2\pi f n / f_s + \varphi) = A \cdot \cos(\dot{\omega} n + \varphi)$$

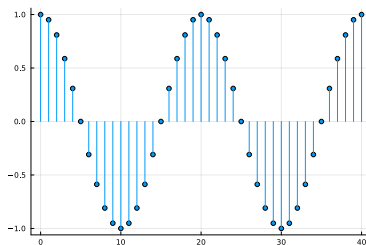
where  $\dot{\omega} = 2\pi f / f_s$  is the normalized angular frequency in radians/sample.

## Julia example:

```
n = 0:40; fs = 8000  
f = 400; x = cos.(2pi*f*n/fs)  
sticks(n, x; shape=:circle)
```

This shows 41 samples ( $\approx 1/200 \text{ s} = 5 \text{ ms}$ ) of an  $f = 400 \text{ Hz}$  sine wave, sampled at  $f_s = 8 \text{ kHz}$ .

**Exercise:** Try  $f = 0, 1000, 2000, 3000, 4000, 5000 \text{ Hz}$ . Try negative  $f$ . Try sine instead of cosine. Try adding phase offsets  $\varphi$  of  $\pm\pi/4$ ,  $\pm\pi/2$ , and  $\pm\pi$ .

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

A sequence  $\{x_n\}$  is

$$\textit{periodic} \Leftrightarrow \exists k > 0 : \forall n \in \mathbb{Z} : x_n = x_{n+k}$$

Is a continuous function with period  $t_p$  still periodic after sampling?

$$\textit{absolutely summable} \Leftrightarrow \sum_{n=-\infty}^{\infty} |x_n| < \infty$$

$$\textit{square summable} \Leftrightarrow \underbrace{\sum_{n=-\infty}^{\infty} |x_n|^2}_{\text{"energy"}} < \infty \Leftrightarrow \text{"energy signal"}$$

$$\underbrace{0 < \lim_{k \rightarrow \infty} \frac{1}{1+2k} \sum_{n=-k}^k |x_n|^2}_{\text{"average power"}} < \infty \Leftrightarrow \text{"power signal"}$$

This energy/power terminology reflects that if  $U$  is a voltage supplied to a load resistor  $R$ , then  $P = UI = U^2/R$  is the power consumed, and  $\int P(t) dt$  the energy. It is used even if we drop physical units (e.g., volts) for simplicity in calculations.

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

A sequence  $\{x_n\}$  is

$$\textit{periodic} \Leftrightarrow \exists k > 0 : \forall n \in \mathbb{Z} : x_n = x_{n+k}$$

Is a continuous function with period  $t_p$  still periodic after sampling? Only if  $t_p/t_s \in \mathbb{Q}$ .

$$\textit{absolutely summable} \Leftrightarrow \sum_{n=-\infty}^{\infty} |x_n| < \infty$$

$$\textit{square summable} \Leftrightarrow \underbrace{\sum_{n=-\infty}^{\infty} |x_n|^2}_{\text{"energy"}} < \infty \Leftrightarrow \text{"energy signal"}$$

$$0 < \underbrace{\lim_{k \rightarrow \infty} \frac{1}{1+2k} \sum_{n=-k}^k |x_n|^2}_{\text{"average power"}} < \infty \Leftrightarrow \text{"power signal"}$$

This energy/power terminology reflects that if  $U$  is a voltage supplied to a load resistor  $R$ , then  $P = UI = U^2/R$  is the power consumed, and  $\int P(t) dt$  the energy. It is used even if we drop physical units (e.g., volts) for simplicity in calculations.

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

# A brief excursion into measuring signal intensity

## Root-mean-square (RMS) signal strength

DC = direct current (constant), AC = alternating current (zero mean)

Consider a time-variable signal  $f(t)$  over time interval  $[t_1, t_2]$ :

$$\text{DC component} = \text{mean voltage} = \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} f(\tau) d\tau$$

$$\text{AC component} = f(t) - \text{DC component}$$

## How can we state the strength of an AC signal?

The root-mean-square signal strength (voltage, etc.)

$$\text{rms} = \sqrt{\frac{1}{t_2 - t_1} \int_{t_1}^{t_2} f^2(\tau) d\tau}$$

is the strength of a DC signal of equal average power.

RMS of a sine wave:

$$\sqrt{\frac{1}{2\pi k} \int_0^{2\pi k} [A \cdot \sin(\tau + \varphi)]^2 d\tau} = \frac{A}{\sqrt{2}} \quad \text{for all } k \in \mathbb{N}, A, \varphi \in \mathbb{R}$$

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

# Perception of signal strength

Sensation limit (SL) = lowest intensity stimulus that can still be perceived

Difference limit (DL) = smallest perceivable stimulus difference at given intensity level

## Weber's law

Difference limit  $\Delta\phi$  is proportional to the intensity  $\phi$  of the stimulus (except for a small correction constant  $a$ , to describe deviation of experimental results near SL):

$$\Delta\phi = c \cdot (\phi + a)$$

## Fechner's scale

Define a perception intensity scale  $\psi$  using the sensation limit  $\phi_0$  as the origin and the respective difference limit  $\Delta\phi = c \cdot \phi$  as a unit step. The result is a logarithmic relationship between stimulus intensity and scale value:

$$\psi = \log_c \frac{\phi}{\phi_0}$$

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

Fechner's scale matches older subjective intensity scales that follow differentiability of stimuli, e.g. the astronomical magnitude numbers for star brightness introduced by Hipparchus ( $\approx 150$  BC).

## Stevens' power law

A sound that is 20 DL over SL is perceived as more than twice as loud as one that is 10 DL over SL, i.e. Fechner's scale does not describe well perceived intensity. A rational scale attempts to reflect subjective relations perceived between different values of stimulus intensity  $\phi$ . Stanley Smith Stevens observed that such rational scales  $\psi$  follow a power law:

$$\psi = k \cdot (\phi - \phi_0)^a$$

Example coefficients  $a$ : brightness 0.33, loudness 0.6, heaviness 1.45, temperature (warmth) 1.6.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

Communications engineers often use logarithmic units:

- ▶ Quantities often vary over many orders of magnitude → difficult to agree on a common SI prefix (nano, micro, milli, kilo, etc.)
- ▶ Quotient of quantities (amplification/attenuation) usually more interesting than difference
- ▶ Signal strength usefully expressed as field quantity (voltage, current, pressure, etc.) or power, but quadratic relationship between these two ( $P = U^2/R = I^2 R$ ) rather inconvenient
- ▶ Perception is logarithmic (Weber/Fechner law → slide 14)

Plus: Using magic special-purpose units has its own odd attractions (→ typographers, navigators)

**Neper (Np)** denotes the natural logarithm of the quotient of a field quantity  $F$  and a reference value  $F_0$ . (rarely used today)

**Bel (B)** denotes the base-10 logarithm of the quotient of a power  $P$  and a reference power  $P_0$ . Common prefix: 10 decibel (dB) = 1 bel.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

Where  $P$  is some power and  $P_0$  a 0 dB reference power, or equally where  $F$  is a field quantity and  $F_0$  the corresponding reference level:

$$10 \text{ dB} \cdot \log_{10} \frac{P}{P_0} = 20 \text{ dB} \cdot \log_{10} \frac{F}{F_0}$$

Common reference values are indicated with a suffix after “dB”:

$$0 \text{ dBW} = 1 \text{ W}$$

$$0 \text{ dBm} = 1 \text{ mW} = -30 \text{ dBW}$$

$$0 \text{ dB}\mu\text{V} = 1 \mu\text{V}$$

$$0 \text{ dBu} = 0.775 \text{ V} = \sqrt{600 \Omega \times 1 \text{ mW}}$$

$$0 \text{ dB}_{\text{SPL}} = 20 \mu\text{Pa} \quad (\text{sound pressure level})$$

$$0 \text{ dB}_{\text{SL}} = \text{perception threshold (sensation limit)}$$

$$0 \text{ dBFS} = \text{full scale (clipping limit of analog/digital converter)}$$

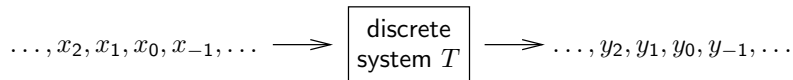
Remember:

$$3 \text{ dB} = 2\times \text{ power}, \quad 6 \text{ dB} = 2\times \text{ voltage/pressure/etc.}$$

$$10 \text{ dB} = 10\times \text{ power}, \quad 20 \text{ dB} = 10\times \text{ voltage/pressure/etc.}$$

W.H. Martin: Decibel – the new name for the transmission unit. Bell Syst. Tech. J., Jan. 1929.  
ITU-R Recommendation V.574-4: Use of the decibel and neper in telecommunication.

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)



A *causal system* cannot look into the future:

$$y_n = f(x_n, x_{n-1}, x_{n-2}, \dots)$$

A *memory-less system* depends only on the current input value:

$$y_n = f(x_n)$$

A *delay system* shifts a sequence in time:

$$y_n = x_{n-d}$$

$T$  is a *time-invariant system* if for any  $d$

$$\{y_n\} = T\{x_n\} \iff \{y_{n-d}\} = T\{x_{n-d}\}.$$

$T$  is a *linear system* if for any pair of sequences  $\{x_n\}$  and  $\{x'_n\}$

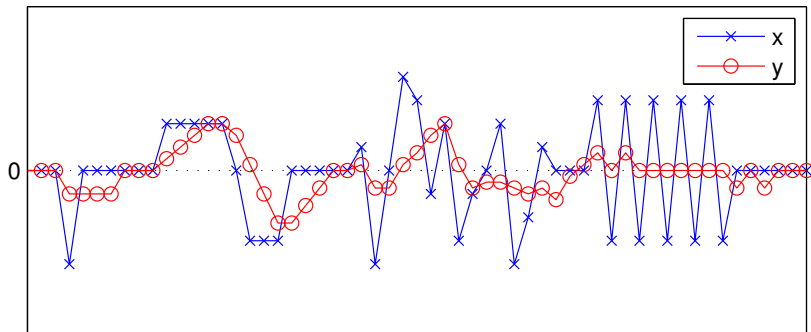
$$T\{a \cdot x_n + b \cdot x'_n\} = a \cdot T\{x_n\} + b \cdot T\{x'_n\}.$$

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

# Example: $M$ -point moving average system

$$y_n = \frac{1}{M} \sum_{k=0}^{M-1} x_{n-k} = \frac{x_{n-M+1} + \cdots + x_{n-1} + x_n}{M}$$

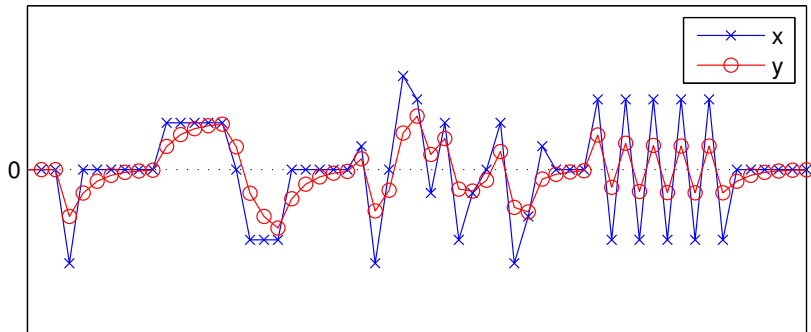
It is causal, linear, time-invariant, with memory. With  $M = 4$ :

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

# Example: exponential averaging system

$$y_n = \alpha \cdot x_n + (1 - \alpha) \cdot y_{n-1} = \alpha \sum_{k=0}^{\infty} (1 - \alpha)^k \cdot x_{n-k}$$

It is causal, linear, time-invariant, with memory. With  $\alpha = \frac{1}{2}$ :



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

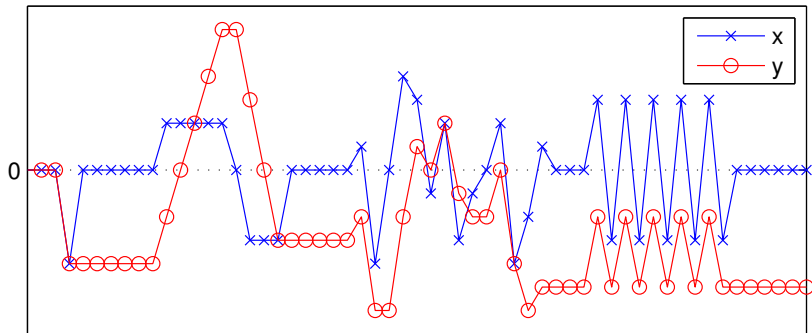
Audiovisual data compression

Outlook

# Example: accumulator system

$$y_n = \sum_{k=-\infty}^n x_k$$

It is causal, linear, time-invariant, with memory.



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

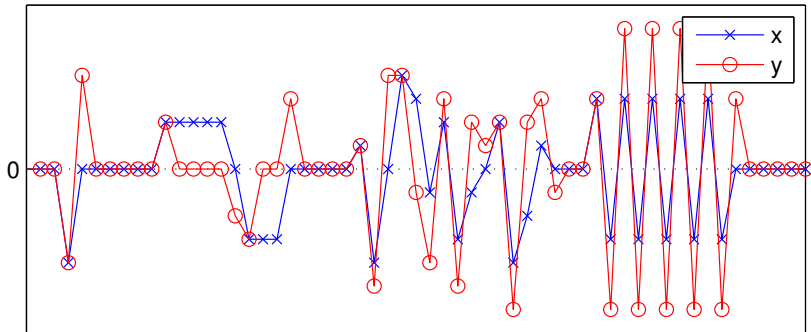
Audiovisual data compression

Outlook

# Example: backward difference system

$$y_n = x_n - x_{n-1}$$

It is causal, linear, time-invariant, with memory.

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

Time-invariant non-linear memory-less systems:

$$y_n = x_n^2, \quad y_n = \log_2 x_n, \quad y_n = \max\{\min\{\lfloor 256x_n \rfloor, 255\}, 0\}$$

Linear but not time-invariant systems:

$$y_n = \begin{cases} x_n, & n \geq 0 \\ 0, & n < 0 \end{cases} = x_n \cdot u_n$$

$$y_n = x_{\lfloor n/4 \rfloor}$$

$$y_n = x_n \cdot \Re(e^{j\omega n})$$

$k$ -times expansion/decimation:

$$y_n = \begin{cases} x_{n/k}, & k \mid n \\ 0, & k \nmid n \end{cases}$$

$$y_n = x_{kn}$$

Linear time-invariant non-causal systems:

$$y_n = \frac{1}{2}(x_{n-1} + x_{n+1})$$

$$y_n = \sum_{k=-9}^9 x_{n+k} \cdot \frac{\sin(\pi k \omega)}{\pi k \omega} \cdot [0.5 + 0.5 \cdot \cos(\pi k/10)]$$

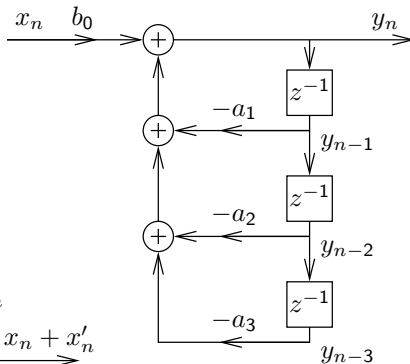
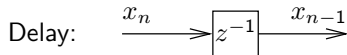
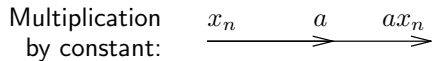
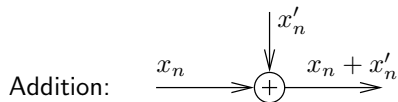
[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

# Constant-coefficient difference equations

Of particular practical interest are causal linear time-invariant systems of the form

$$y_n = b_0 \cdot x_n - \sum_{k=1}^N a_k \cdot y_{n-k}$$

Block diagram representation of sequence operations:

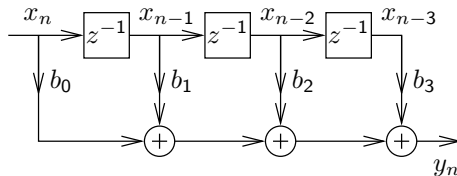


The  $a_k$  and  $b_m$  are constant coefficients.

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

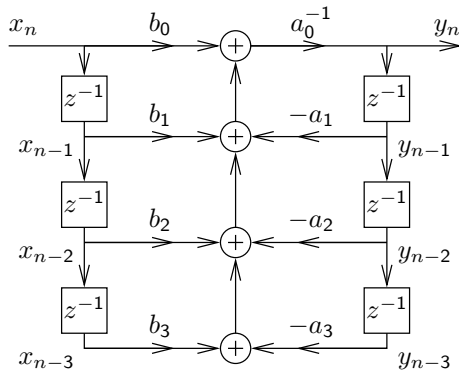
or

$$y_n = \sum_{m=0}^M b_m \cdot x_{n-m}$$



or the combination of both:

$$\sum_{k=0}^N a_k \cdot y_{n-k} = \sum_{m=0}^M b_m \cdot x_{n-m}$$



Implementations: DSP.jl's `filt(b, a, x)`, MATLAB's `filter`, `scipy.signal.lfilter`.

- ① Sequences and systems
- ② **Convolution**
- ③ Fourier transform
- ④ Sampling
- ⑤ Discrete Fourier transform
- ⑥ Deconvolution
- ⑦ Spectral estimation
- ⑧ Digital filters
- ⑨ IIR filters
- ⑩ Random signals
- ⑪ Digital communication
- ⑫ Audiovisual data compression

Introduction

Sequences and systems

**Convolution**

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

Another example of an LTI system is

$$y_n = \sum_{k=-\infty}^{\infty} a_k \cdot x_{n-k}$$

where  $\{a_k\}$  is a suitably chosen sequence of coefficients.

This operation over sequences is called *convolution* and is defined as

$$\{p_n\} * \{q_n\} = \{r_n\} \quad \Longleftrightarrow \quad \forall n \in \mathbb{Z} : r_n = \sum_{k=-\infty}^{\infty} p_k \cdot q_{n-k}.$$

Introduction

Sequences and systems

**Convolution**

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

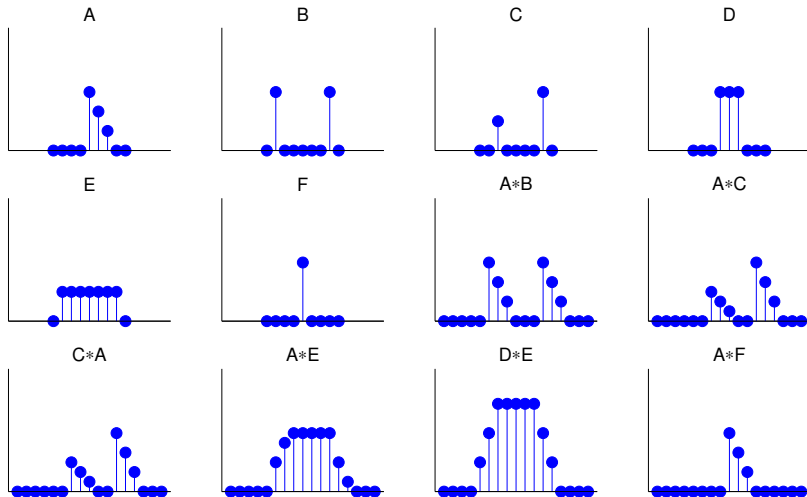
IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook



Introduction

Sequences and systems

**Convolution**

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

Another example of an LTI system is

$$y_n = \sum_{k=-\infty}^{\infty} a_k \cdot x_{n-k}$$

where  $\{a_k\}$  is a suitably chosen sequence of coefficients.

This operation over sequences is called *convolution* and is defined as

$$\{p_n\} * \{q_n\} = \{r_n\} \quad \Longleftrightarrow \quad \forall n \in \mathbb{Z} : r_n = \sum_{k=-\infty}^{\infty} p_k \cdot q_{n-k}.$$

Introduction

Sequences and systems

**Convolution**

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

Another example of an LTI system is

$$y_n = \sum_{k=-\infty}^{\infty} a_k \cdot x_{n-k}$$

where  $\{a_k\}$  is a suitably chosen sequence of coefficients.

This operation over sequences is called *convolution* and is defined as

$$\{p_n\} * \{q_n\} = \{r_n\} \iff \forall n \in \mathbb{Z} : r_n = \sum_{k=-\infty}^{\infty} p_k \cdot q_{n-k}.$$

If  $\{y_n\} = \{a_n\} * \{x_n\}$  is a representation of an LTI system  $T$ , with  $\{y_n\} = T\{x_n\}$ , then we call the sequence  $\{a_n\}$  the *impulse response* of  $T$ , because  $\{a_n\} = T\{\delta_n\}$ , as  $\{a_n\} * \{\delta_n\} = \{a_n\}$ ,  $\sum_k a_k \cdot \delta_{n-k} = a_n$ .

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

For arbitrary sequences  $\{p_n\}$ ,  $\{q_n\}$ ,  $\{r_n\}$  and scalars  $a$ ,  $b$ :

- Convolution is associative

$$(\{p_n\} * \{q_n\}) * \{r_n\} = \{p_n\} * (\{q_n\} * \{r_n\})$$

- Convolution is commutative

$$\{p_n\} * \{q_n\} = \{q_n\} * \{p_n\}$$

- Convolution is linear

$$\{p_n\} * \{a \cdot q_n + b \cdot r_n\} = a \cdot (\{p_n\} * \{q_n\}) + b \cdot (\{p_n\} * \{r_n\})$$

- The impulse sequence (slide 10) is neutral under convolution

$$\{p_n\} * \{\delta_n\} = \{\delta_n\} * \{p_n\} = \{p_n\}$$

- Sequence shifting is equivalent to convolving with a shifted impulse

$$\{p_{n-d}\}_n = \{p_n\} * \{\delta_{n-d}\}_n$$

Introduction

Sequences and systems

**Convolution**

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

Another example of an LTI system is

$$y_n = \sum_{k=-\infty}^{\infty} a_k \cdot x_{n-k}$$

where  $\{a_k\}$  is a suitably chosen sequence of coefficients.

This operation over sequences is called *convolution* and is defined as

$$\{p_n\} * \{q_n\} = \{r_n\} \iff \forall n \in \mathbb{Z} : r_n = \sum_{k=-\infty}^{\infty} p_k \cdot q_{n-k}.$$

If  $\{y_n\} = \{a_n\} * \{x_n\}$  is a representation of an LTI system  $T$ , with  $\{y_n\} = T\{x_n\}$ , then we call the sequence  $\{a_n\}$  the *impulse response* of  $T$ , because  $\{a_n\} = T\{\delta_n\}$ , as  $\{a_n\} * \{\delta_n\} = \{a_n\}$ ,  $\sum_k a_k \cdot \delta_{n-k} = a_n$ .

Introduction

Sequences and systems

**Convolution**

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

Another example of an LTI system is

$$y_n = \sum_{k=-\infty}^{\infty} a_k \cdot x_{n-k}$$

where  $\{a_k\}$  is a suitably chosen sequence of coefficients.

This operation over sequences is called *convolution* and is defined as

$$\{p_n\} * \{q_n\} = \{r_n\} \iff \forall n \in \mathbb{Z} : r_n = \sum_{k=-\infty}^{\infty} p_k \cdot q_{n-k}.$$

If  $\{y_n\} = \{a_n\} * \{x_n\}$  is a representation of an LTI system  $T$ , with  $\{y_n\} = T\{x_n\}$ , then we call the sequence  $\{a_n\}$  the *impulse response* of  $T$ , because  $\{a_n\} = T\{\delta_n\}$ , as  $\{a_n\} * \{\delta_n\} = \{a_n\}$ ,  $\sum_k a_k \cdot \delta_{n-k} = a_n$ .

If  $f$  and  $g$  are continuous functions, their convolution is defined similarly as the integral

$$(f * g)(t) = \int_{-\infty}^{\infty} f(s)g(t-s)ds.$$

But what is the continuous equivalent of  $\{\delta_n\}$ ? More on that later . . .

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

# All LTI systems just apply convolution

## Proof:

Any sequence  $\{x_n\}$  can be decomposed into a weighted sum of shifted impulse sequences:

$$\{x_n\} = \sum_{k=-\infty}^{\infty} x_k \cdot \{\delta_{n-k}\}$$

Let's see what happens if we apply a linear<sup>(\*)</sup> time-invariant<sup>(\*\*)</sup> system  $T$  to such a decomposed sequence:

$$\begin{aligned} T\{x_n\} &= T\left(\sum_{k=-\infty}^{\infty} x_k \cdot \{\delta_{n-k}\}\right) \stackrel{(*)}{=} \sum_{k=-\infty}^{\infty} x_k \cdot T\{\delta_{n-k}\} \\ &\stackrel{(**)}{=} \sum_{k=-\infty}^{\infty} x_k \cdot \{\delta_{n-k}\} * T\{\delta_n\} = \left(\sum_{k=-\infty}^{\infty} x_k \cdot \{\delta_{n-k}\}\right) * T\{\delta_n\} \\ &= \{x_n\} * T\{\delta_n\} \quad \text{q.e.d.} \end{aligned}$$

$\Rightarrow$  The impulse response  $T\{\delta_n\}$  fully characterizes an LTI system.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

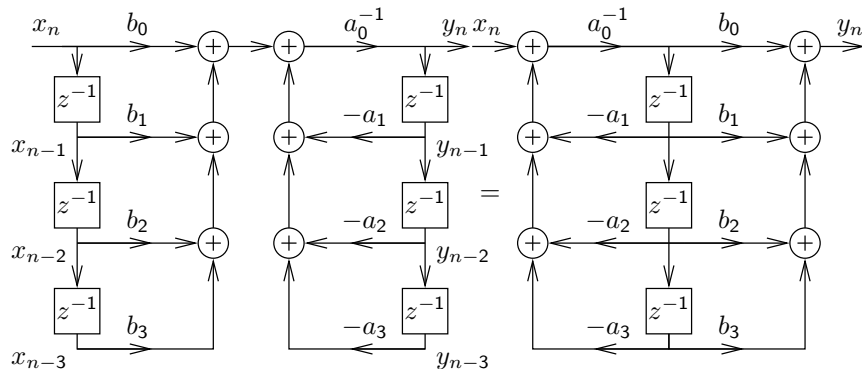
IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook



The block diagram representation of the constant-coefficient difference equation on slide 25 is called the *direct form I implementation*.

The number of delay elements can be halved by using the commutativity of convolution to swap the two feedback loops, leading to the *direct form II implementation* of the same LTI system.

These two forms are only equivalent with ideal arithmetic (no rounding errors and range limits).

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

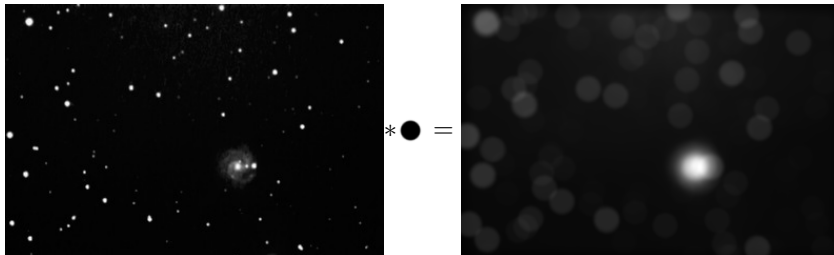
Digital communication

Audiovisual data compression

Outlook

# Convolution: optics example

If a projective lens is out of focus, the blurred image is equal to the original image convolved with the aperture shape (e.g., a filled circle):

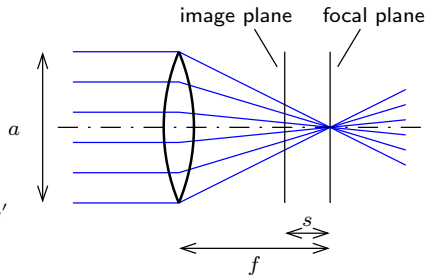


Point-spread function  $h$  (disk,  $r = \frac{a s}{2f}$ ):

$$h(x, y) = \begin{cases} \frac{1}{r^2 \pi}, & x^2 + y^2 \leq r^2 \\ 0, & x^2 + y^2 > r^2 \end{cases}$$

Original image  $I$ , blurred image  $B = I * h$ , i.e.

$$B(x, y) = \iint I(x - x', y - y') \cdot h(x', y') \cdot dx' dy'$$



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

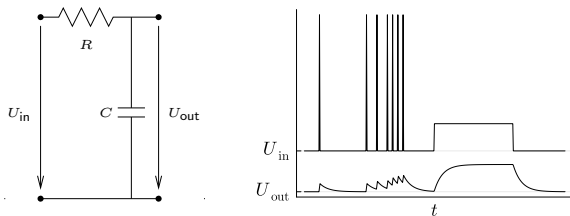
IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook



Any passive network (resistors, capacitors, inductors) convolves its input voltage  $U_{in}$  with an *impulse response function*  $h$ , leading to  $U_{out} = U_{in} * h$ , that is

$$U_{out}(t) = \int_{-\infty}^{\infty} U_{in}(t - \tau) \cdot h(\tau) \cdot d\tau$$

In the above example:

$$\frac{U_{in} - U_{out}}{R} = C \cdot \frac{dU_{out}}{dt}, \quad h(t) = \begin{cases} \frac{1}{RC} \cdot e^{\frac{-t}{RC}}, & t \geq 0 \\ 0, & t < 0 \end{cases}$$

Introduction

Sequences and systems

**Convolution**

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

- 1 Sequences and systems
- 2 Convolution
- 3 Fourier transform**
- 4 Sampling
- 5 Discrete Fourier transform
- 6 Deconvolution
- 7 Spectral estimation
- 8 Digital filters
- 9 IIR filters
- 10 Random signals
- 11 Digital communication
- 12 Audiovisual data compression

Introduction

Sequences and systems

Convolution

**Fourier transform**

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

Adding together sine waves of equal frequency, but arbitrary amplitude and phase, results in another sine wave of the same frequency:

$$A_1 \cdot \sin(\omega t + \varphi_1) + A_2 \cdot \sin(\omega t + \varphi_2) = A \cdot \sin(\omega t + \varphi)$$

Why?

Introduction

Sequences and systems

Convolution

**Fourier transform**

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

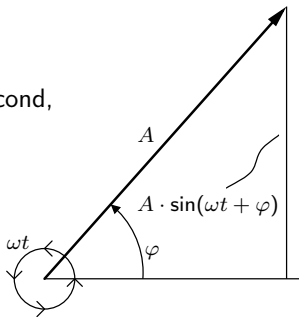
# Adding sine waves

Adding together sine waves of equal frequency, but arbitrary amplitude and phase, results in another sine wave of the same frequency:

$$A_1 \cdot \sin(\omega t + \varphi_1) + A_2 \cdot \sin(\omega t + \varphi_2) = A \cdot \sin(\omega t + \varphi)$$

Why?

Think of  $A \cdot \sin(\omega t + \varphi)$  as the height of an arrow of length  $A$ , rotating  $\frac{\omega}{2\pi}$  times per second, with start angle  $\varphi$  (radians) at  $t = 0$ .

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

# Adding sine waves

Adding together sine waves of equal frequency, but arbitrary amplitude and phase, results in another sine wave of the same frequency:

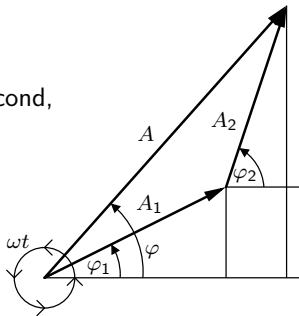
$$A_1 \cdot \sin(\omega t + \varphi_1) + A_2 \cdot \sin(\omega t + \varphi_2) = A \cdot \sin(\omega t + \varphi)$$

Why?

Think of  $A \cdot \sin(\omega t + \varphi)$  as the height of an arrow of length  $A$ , rotating  $\frac{\omega}{2\pi}$  times per second, with start angle  $\varphi$  (radians) at  $t = 0$ .

Consider two more such arrows, of length  $A_1$  and  $A_2$ , with start angles  $\varphi_1$  and  $\varphi_2$ .

$A_1$  and  $A_2$  stuck together are as high as  $A$ , all three rotating at the same frequency.

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

Adding together sine waves of equal frequency, but arbitrary amplitude and phase, results in another sine wave of the same frequency:

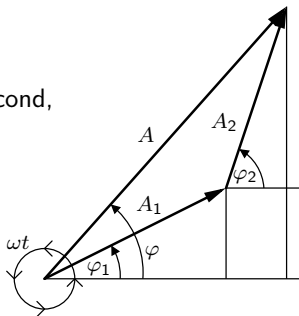
$$A_1 \cdot \sin(\omega t + \varphi_1) + A_2 \cdot \sin(\omega t + \varphi_2) = A \cdot \sin(\omega t + \varphi)$$

Why?

Think of  $A \cdot \sin(\omega t + \varphi)$  as the height of an arrow of length  $A$ , rotating  $\frac{\omega}{2\pi}$  times per second, with start angle  $\varphi$  (radians) at  $t = 0$ .

Consider two more such arrows, of length  $A_1$  and  $A_2$ , with start angles  $\varphi_1$  and  $\varphi_2$ .

$A_1$  and  $A_2$  stuck together are as high as  $A$ , all three rotating at the same frequency.



But adding sine waves as vectors  $(A_1, \varphi_1)$  and  $(A_2, \varphi_2)$  in polar coordinates is cumbersome:

$$A = \sqrt{A_1^2 + A_2^2 + 2A_1A_2 \cos(\varphi_2 - \varphi_1)}, \quad \tan \varphi = \frac{A_1 \sin \varphi_1 + A_2 \sin \varphi_2}{A_1 \cos \varphi_1 + A_2 \cos \varphi_2}$$

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

# Cartesian coordinates for sine waves

Sine waves of any amplitude  $A$  and phase (start angle)  $\varphi$  can be represented as linear combinations of  $\sin(\omega t)$  and  $\cos(\omega t)$ :

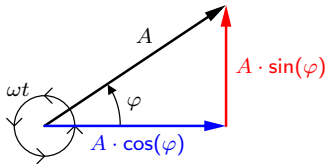
$$A \cdot \sin(\omega t + \varphi) = x \cdot \sin(\omega t) + y \cdot \cos(\omega t) \quad \cos(\omega t) = \sin(\omega t + 90^\circ)$$

where

$$x = A \cdot \cos(\varphi), \quad y = A \cdot \sin(\varphi)$$

and

$$A = \sqrt{x^2 + y^2}, \quad \tan \varphi = \frac{y}{x}.$$



Base: two rotating arrows with start angles  $0^\circ$  [height =  $\sin(\omega t)$ ] and  $90^\circ$  [height =  $\cos(\omega t)$ ].

Adding two sine waves as vectors in Cartesian coordinates is simple:

$$f_1(t) = x_1 \cdot \sin(\omega t) + y_1 \cdot \cos(\omega t)$$

$$f_2(t) = x_2 \cdot \sin(\omega t) + y_2 \cdot \cos(\omega t)$$

$$f_1(t) + f_2(t) = (x_1 + x_2) \cdot \sin(\omega t) + (y_1 + y_2) \cdot \cos(\omega t)$$

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

# Why are sine waves useful?

## 1) Sine-wave sequences form a family of discrete sequences that is closed under convolution with arbitrary sequences.

Convolution of a discrete sequence  $\{x_n\}$  with another sequence  $\{h_n\}$  is nothing but adding together scaled and delayed copies of  $\{x_n\}$ .

Think again of  $\{h_n\}$  as decomposed into a sum of impulses:

$$\begin{aligned}\{x_n\} * \{h_n\} &= \{x_n\} * \sum_k h_k \cdot \{\delta_{n-k}\}_n = \sum_k h_k \cdot (\{x_n\} * \{\delta_{n-k}\}_n) \\ &= \sum_k h_k \cdot \{x_{n-k}\}_n\end{aligned}$$

If  $\{x_n\}$  is a sampled sine wave of frequency  $f$ , i.e.

$$x_n = A_x \cdot \sin(2\pi f n / f_s + \phi_x)$$

then  $\{y_n\} = \{x_n\} * \{h_n\} = \sum_k h_k \cdot \{x_{n-k}\}_n$  is another sampled sine wave of frequency  $f$ , i.e. for each  $\{h_n\}$  there exists a pair  $(A_y, \phi_y)$  with

$$y_n = A_y \cdot \sin(2\pi f n / f_s + \phi_y)$$

The equivalent applies for continuous sine waves and convolution.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

# Why are sine waves useful?

## 2) Sine waves are orthogonal to each other

The term “orthogonal” is used here in the context of an (infinitely dimensional) vector space, where the “vectors” are functions of the form  $f : \mathbb{R} \rightarrow \mathbb{R}$  (or  $f : \mathbb{R} \rightarrow \mathbb{C}$ ) and the scalar product is defined as

$$f \cdot g = \int f(t) \cdot g(t) dt.$$

Over integer (half-)periods:

$$m, n \in \mathbb{N}, m \neq n \quad \Rightarrow \quad \int_0^\pi \sin(nt) \sin(mt) dt = 0$$

$$m, n \in \mathbb{N} \quad \Rightarrow \quad \int_{-\pi}^\pi \sin(nt) \cos(mt) dt = 0$$

We can even (with some handwaving) extend this to improper integrals:

$$\int_{-\infty}^{\infty} \sin(\omega_1 t + \varphi_1) \cdot \sin(\omega_2 t + \varphi_2) dt \text{ “=” } 0$$

$$\iff \omega_1 \neq \omega_2 \quad \vee \quad \varphi_1 - \varphi_2 = (2k + 1)\pi/2 \quad (k \in \mathbb{Z})$$

They can be used to form an orthogonal function basis for a transform.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

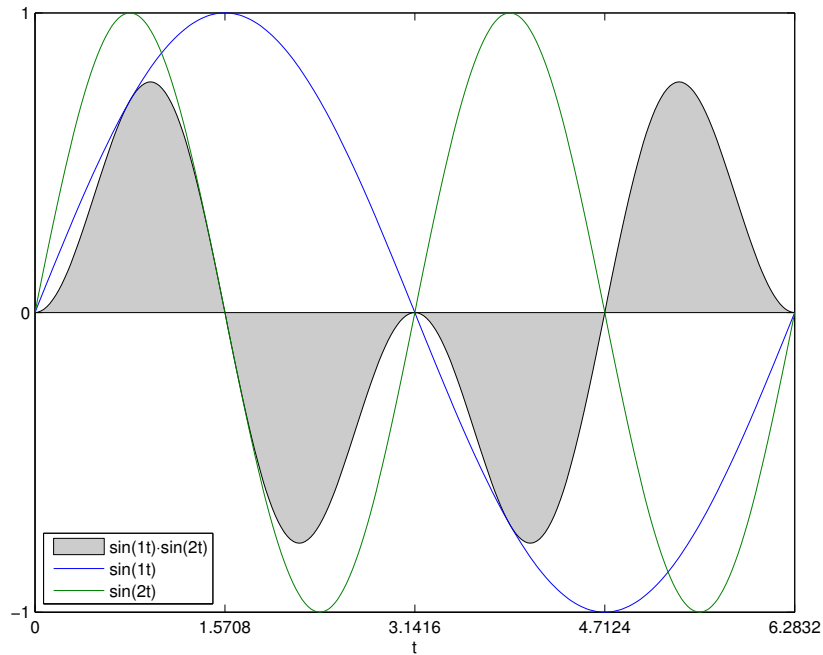
IIR filters

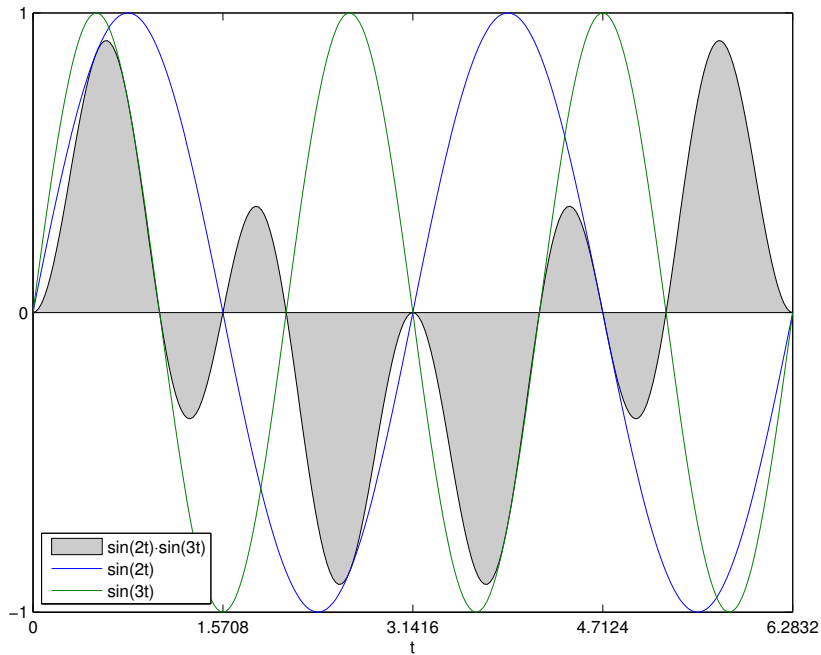
Random signals

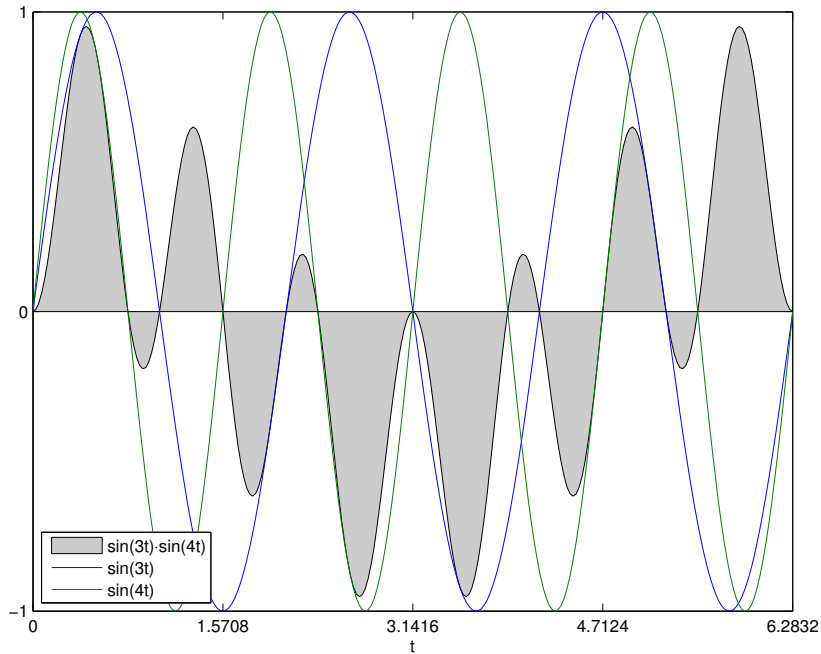
Digital communication

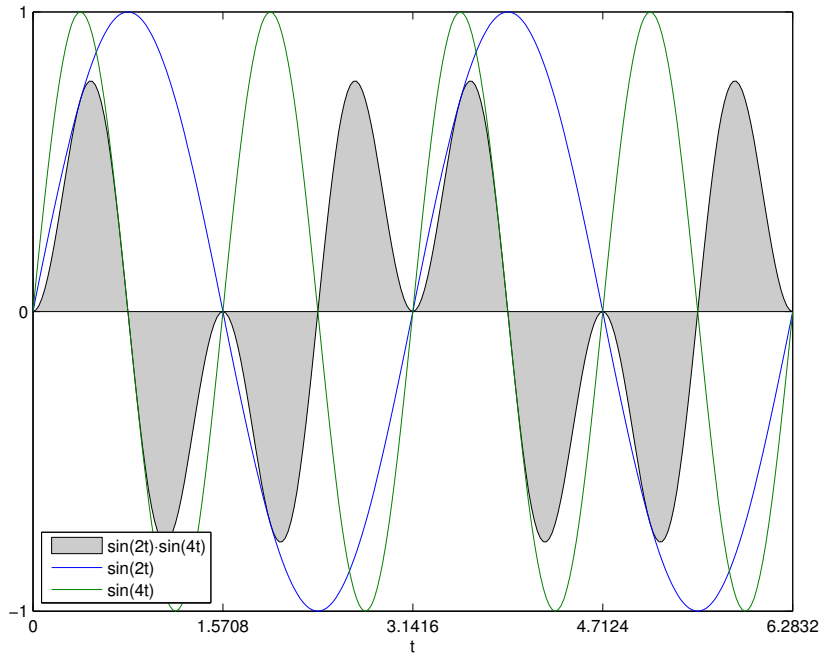
Audiovisual data compression

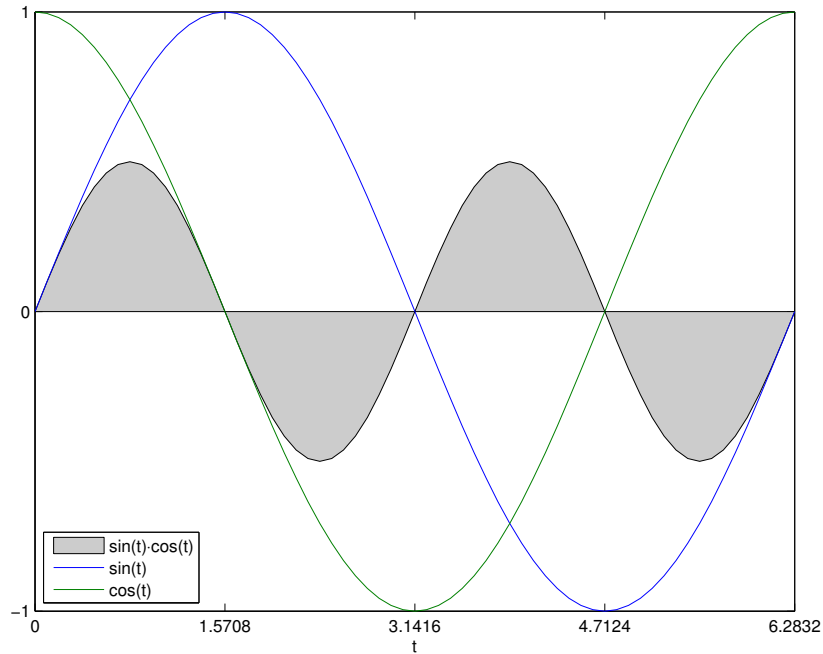
Outlook

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

# Why are exponential functions useful?

Adding together two exponential functions with the same base  $z$ , but different scale factor and offset, results in another exponential function with the same base:

$$\begin{aligned} A_1 \cdot z^{t+\varphi_1} + A_2 \cdot z^{t+\varphi_2} &= A_1 \cdot z^t \cdot z^{\varphi_1} + A_2 \cdot z^t \cdot z^{\varphi_2} \\ &= (A_1 \cdot z^{\varphi_1} + A_2 \cdot z^{\varphi_2}) \cdot z^t = A \cdot z^t \end{aligned}$$

Likewise, if we convolve a sequence  $\{x_n\}$  of values

$$\dots, z^{-3}, z^{-2}, z^{-1}, 1, z, z^2, z^3, \dots$$

$x_n = z^n$  with an arbitrary sequence  $\{h_n\}$ , we get  $\{y_n\} = \{z^n\} * \{h_n\}$ ,

$$y_n = \sum_{k=-\infty}^{\infty} x_{n-k} \cdot h_k = \sum_{k=-\infty}^{\infty} z^{n-k} \cdot h_k = z^n \cdot \sum_{k=-\infty}^{\infty} z^{-k} \cdot h_k = z^n \cdot H(z)$$

where  $H(z)$  is independent of  $n$ .

**Exponential sequences are closed under convolution with arbitrary sequences.**

The same applies in the continuous case.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

# Why are complex numbers so useful?

- 1) They give us all  $n$  solutions ( “roots” ) of equations involving polynomials up to degree  $n$  (the “  $\sqrt{-1} = j$  ” story).
- 2) They give us the “great unifying theory” that combines sine and exponential functions:

$$\cos(\theta) = \frac{1}{2} (e^{j\theta} + e^{-j\theta})$$

$$\sin(\theta) = \frac{1}{2j} (e^{j\theta} - e^{-j\theta})$$

or

$$\cos(\omega t + \varphi) = \frac{1}{2} (e^{j(\omega t + \varphi)} + e^{-j(\omega t + \varphi)})$$

or

$$\cos(\dot{\omega} n + \varphi) = \Re(e^{j(\dot{\omega} n + \varphi)}) = \Re[(e^{j\dot{\omega}})^n \cdot e^{j\varphi}]$$

$$\sin(\dot{\omega} n + \varphi) = \Im(e^{j(\dot{\omega} n + \varphi)}) = \Im[(e^{j\dot{\omega}})^n \cdot e^{j\varphi}]$$

Notation:  $\Re(a + jb) := a$ ,  $\Im(a + jb) := b$  and  $(a + jb)^* := a - jb$ , where  $j^2 = -1$  and  $a, b \in \mathbb{R}$ .  
Then  $\Re(x) = \frac{1}{2}(x + x^*)$  and  $\Im(x) = \frac{1}{2j}(x - x^*)$  for all  $x \in \mathbb{C}$ .

Introduction

Sequences and systems

Convolution

**Fourier transform**

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

We can now represent sine waves as projections of a rotating complex vector. This allows us to represent sine-wave sequences as exponential sequences with basis  $e^{j\omega}$ .

A phase shift in such a sequence corresponds to a rotation of a complex vector.

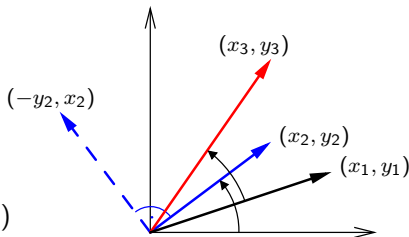
3) Complex multiplication allows us to modify the amplitude and phase of a complex rotating vector using a single operation and value.

Rotation of a 2D vector in  $(x, y)$ -form is notationally slightly messy, but fortunately  $j^2 = -1$  does exactly what is required here:

$$\begin{pmatrix} x_3 \\ y_3 \end{pmatrix} = \begin{pmatrix} x_2 & -y_2 \\ y_2 & x_2 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \\ = \begin{pmatrix} x_1x_2 - y_1y_2 \\ x_1y_2 + x_2y_1 \end{pmatrix}$$

$$z_1 = x_1 + jy_1, \quad z_2 = x_2 + jy_2$$

$$z_1 \cdot z_2 = x_1x_2 - y_1y_2 + j(x_1y_2 + x_2y_1)$$



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

Amplitude and phase are two distinct characteristics of a sine function that are inconvenient to keep separate notationally.

Complex functions (and discrete sequences) of the form

$$(A \cdot e^{j\varphi}) \cdot e^{j\omega t} = A \cdot e^{j(\omega t + \varphi)} = A \cdot [\cos(\omega t + \varphi) + j \cdot \sin(\omega t + \varphi)]$$

(where  $j^2 = -1$ ) are able to represent both amplitude  $A \in \mathbb{R}^+$  and phase  $\varphi \in [0, 2\pi)$  in one single algebraic object  $A \cdot e^{j\varphi} \in \mathbb{C}$ .

Thanks to complex multiplication, we can also incorporate in one single factor both a multiplicative change of amplitude and an additive change of phase of such a function. This makes discrete sequences of the form

$$x_n = e^{j\omega n}$$

*eigensequences* with respect to an LTI system  $T$ , because for each  $\omega$ , there is a complex number (eigenvalue)  $H(\omega)$  such that

$$T\{x_n\} = H(\omega) \cdot \{x_n\}$$

In the notation of slide 38, where the argument of  $H$  is the base, we would write  $H(e^{j\omega})$ .

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

We define the Fourier integral transform and its inverse as

$$\mathcal{F}\{g(t)\}(f) = G(f) = \int_{-\infty}^{\infty} g(t) \cdot e^{-2\pi j f t} dt$$

$$\mathcal{F}^{-1}\{G(f)\}(t) = g(t) = \int_{-\infty}^{\infty} G(f) \cdot e^{2\pi j f t} df$$

Many equivalent forms of the Fourier transform are used in the literature. There is no strong consensus on whether the forward transform uses  $e^{-2\pi j f t}$  and the backwards transform  $e^{2\pi j f t}$ , or vice versa. The above form uses the *ordinary frequency*  $f$ , whereas some authors prefer the *angular frequency*  $\omega = 2\pi f$ :

$$\mathcal{F}\{h(t)\}(\omega) = H(\omega) = \alpha \int_{-\infty}^{\infty} h(t) \cdot e^{\mp j \omega t} dt$$

$$\mathcal{F}^{-1}\{H(\omega)\}(t) = h(t) = \beta \int_{-\infty}^{\infty} H(\omega) \cdot e^{\pm j \omega t} d\omega$$

This substitution introduces factors  $\alpha$  and  $\beta$  such that  $\alpha\beta = 1/(2\pi)$ . Some authors set  $\alpha = 1$  and  $\beta = 1/(2\pi)$ , to keep the convolution theorem free of a constant prefactor; others prefer the unitary form  $\alpha = \beta = 1/\sqrt{2\pi}$ , in the interest of symmetry.

Introduction

Sequences and systems

Convolution

**Fourier transform**

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

If

$$x(t) \longleftrightarrow X(f) \quad \text{and} \quad y(t) \longleftrightarrow Y(f)$$

are pairs of functions that are mapped onto each other by the Fourier transform, then so are the following pairs.

Linearity:

$$ax(t) + by(t) \longleftrightarrow aX(f) + bY(f)$$

Time scaling:

$$x(at) \longleftrightarrow \frac{1}{|a|} X\left(\frac{f}{a}\right)$$

Frequency scaling:

$$\frac{1}{|a|} x\left(\frac{t}{a}\right) \longleftrightarrow X(af)$$

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

Time shifting:

$$x(t - \Delta t) \quad \bullet \longleftrightarrow \quad X(f) \cdot e^{-2\pi j f \Delta t}$$

Frequency shifting:

$$x(t) \cdot e^{2\pi j \Delta f t} \quad \bullet \longleftrightarrow \quad X(f - \Delta f)$$

Time reversal:

$$x(-t) \quad \bullet \longleftrightarrow \quad X(-f)$$

Complex conjugate:

$$\begin{aligned} x^*(t) &\quad \bullet \longleftrightarrow \quad X^*(-f) \\ x^*(-t) &\quad \bullet \longleftrightarrow \quad X^*(f) \end{aligned}$$

Parseval's theorem (total energy):

$$\int_{-\infty}^{\infty} |x(t)|^2 dt = \int_{-\infty}^{\infty} |X(f)|^2 df$$

Introduction

Sequences and systems

Convolution

**Fourier transform**

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

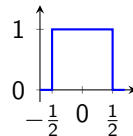
Digital communication

Audiovisual data compression

Outlook

The Fourier transform of the “rectangular function”

$$\text{rect}(t) = \begin{cases} 1 & \text{if } |t| < \frac{1}{2} \\ \frac{1}{2} & \text{if } |t| = \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$



is the “(normalized) sinc function”

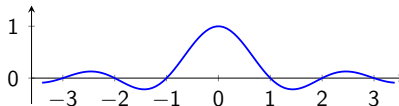
$$\mathcal{F}\{\text{rect}(t)\}(f) = \int_{-\frac{1}{2}}^{\frac{1}{2}} e^{-2\pi j f t} dt = \frac{\sin \pi f}{\pi f} = \text{sinc}(f)$$

and vice versa

$$\mathcal{F}\{\text{sinc}(t)\}(f) = \text{rect}(f).$$

Some noteworthy properties of these functions:

- ▶  $\int_{-\infty}^{\infty} \text{sinc}(t) dt = 1 = \int_{-\infty}^{\infty} \text{rect}(t) dt$
- ▶  $\text{sinc}(0) = 1 = \text{rect}(0)$
- ▶  $\forall n \in \mathbb{Z} \setminus \{0\} : \text{sinc}(n) = 0$

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

Convolution in the time domain is equivalent to (complex) scalar multiplication in the frequency domain:

$$\mathcal{F}\{(f * g)(t)\} = \mathcal{F}\{f(t)\} \cdot \mathcal{F}\{g(t)\}$$

Proof:  $z(r) = \int_s x(s)y(r-s)ds \iff \int_r z(r)e^{-j\omega r}dr = \int_r \int_s x(s)y(r-s)e^{-j\omega r}dsdr =$   
 $\int_s x(s) \int_r y(r-s)e^{-j\omega r}drds = \int_s x(s)e^{-j\omega s} \int_r y(r-s)e^{-j\omega(r-s)}drds \stackrel{t:=r-s}{=} \int_s x(s)e^{-j\omega s} \int_t y(t)e^{-j\omega t}dtds = \int_s x(s)e^{-j\omega s}ds \cdot \int_t y(t)e^{-j\omega t}dt.$

Convolution in the frequency domain corresponds to scalar multiplication in the time domain:

$$\mathcal{F}\{f(t) \cdot g(t)\} = \mathcal{F}\{f(t)\} * \mathcal{F}\{g(t)\}$$

This second form is also called “modulation theorem”, as it describes what happens in the frequency domain with amplitude modulation of a signal (see slide 53).

The proof is very similar to the one above.

Both equally work for the inverse Fourier transform:

$$\mathcal{F}^{-1}\{(F * G)(f)\} = \mathcal{F}^{-1}\{F(f)\} \cdot \mathcal{F}^{-1}\{G(f)\}$$
$$\mathcal{F}^{-1}\{F(f) \cdot G(f)\} = \mathcal{F}^{-1}\{F(f)\} * \mathcal{F}^{-1}\{G(f)\}$$

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

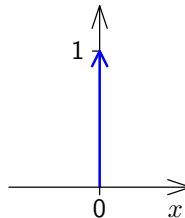
Digital communication

Audiovisual data compression

Outlook

The continuous equivalent of the impulse sequence  $\{\delta_n\}$  is known as Dirac delta function  $\delta(x)$ . It is a generalized function, defined such that

$$\delta(x) = \begin{cases} 0, & x \neq 0 \\ \infty, & x = 0 \end{cases}$$
$$\int_{-\infty}^{\infty} \delta(x) dx = 1$$



and can be thought of as the limit of function sequences such as

$$\delta(x) = \lim_{n \rightarrow \infty} \begin{cases} 0, & |x| \geq 1/n \\ n/2, & |x| < 1/n \end{cases}$$

or

$$\delta(x) = \lim_{n \rightarrow \infty} \frac{n}{\sqrt{\pi}} e^{-n^2 x^2}$$

The delta function is mathematically speaking not a function, but a *distribution*, that is an expression that is only defined when integrated.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

Some properties of the Dirac delta function:

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

$$\int_{-\infty}^{\infty} e^{\pm 2\pi j x a} dx = \delta(a)$$

$$\sum_{i=-\infty}^{\infty} e^{\pm 2\pi j i x a} = \frac{1}{|a|} \sum_{i=-\infty}^{\infty} \delta(x - i/a)$$

$$\delta(ax) = \frac{1}{|a|} \delta(x)$$

Fourier transform:

$$\mathcal{F}\{\delta(t)\}(f) = \int_{-\infty}^{\infty} \delta(t) \cdot e^{-2\pi j f t} dt = e^0 = 1$$

$$\mathcal{F}^{-1}\{1\}(t) = \int_{-\infty}^{\infty} 1 \cdot e^{2\pi j f t} df = \delta(t)$$

Introduction

Sequences and systems

Convolution

**Fourier transform**

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

The Fourier transform of 1 follows from the Dirac delta's ability to sample inside an integral:

$$\begin{aligned} g(t) &= \mathcal{F}^{-1}(\mathcal{F}(g))(t) \\ &= \int_{-\infty}^{\infty} \left( \int_{-\infty}^{\infty} g(s) \cdot e^{-2\pi j f s} \cdot ds \right) \cdot e^{2\pi j f t} \cdot df \\ &= \int_{-\infty}^{\infty} \left( \int_{-\infty}^{\infty} e^{-2\pi j f s} \cdot e^{2\pi j f t} \cdot df \right) \cdot g(s) \cdot ds \\ &= \int_{-\infty}^{\infty} \underbrace{\left( \int_{-\infty}^{\infty} e^{-2\pi j f (s-t)} \cdot df \right)}_{\delta(s-t)} \cdot g(s) \cdot ds \end{aligned}$$

So if  $\delta$  has the property

$$g(t) = \int_{-\infty}^{\infty} \delta(s-t) \cdot g(s) \cdot ds$$

then

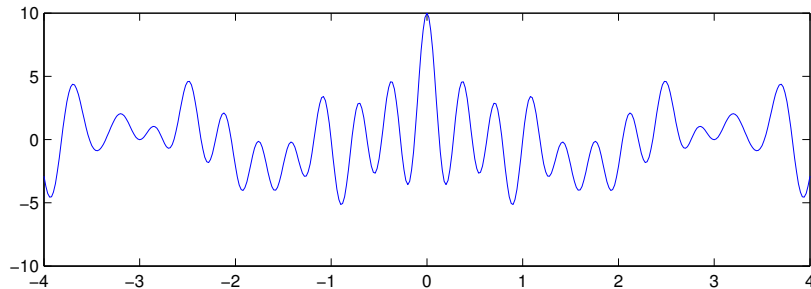
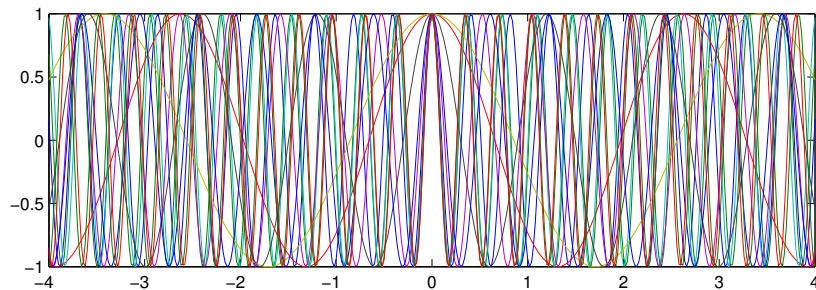
$$\int_{-\infty}^{\infty} e^{-2\pi j f (s-t)} df = \delta(s-t)$$

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

$$\int_{-\infty}^{\infty} e^{2\pi j t f} df = \delta(t)$$

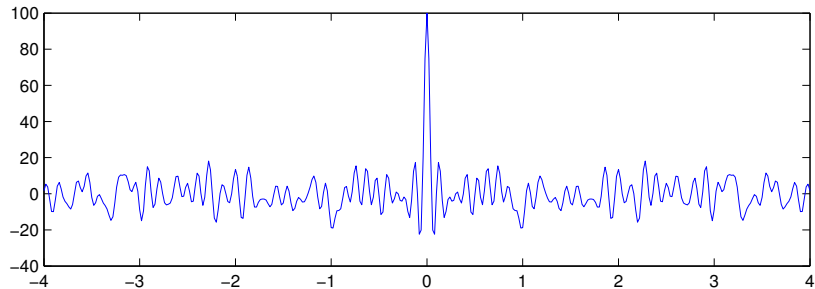
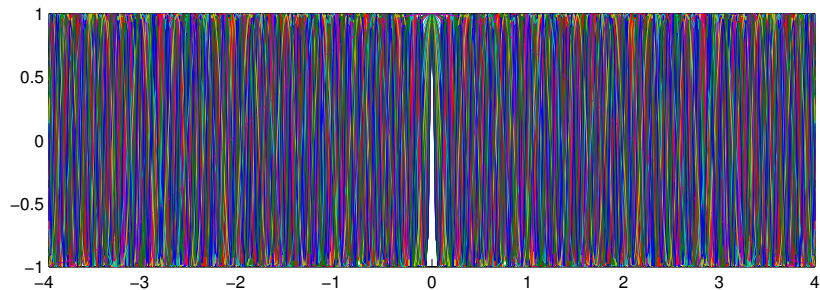
$$\sum_{i=1}^{10} \cos(2\pi f_i t) \approx \delta(t)$$

$f_1, \dots, f_{10} \in [0, 3]$  chosen uniformly at random

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

$$\int_{-\infty}^{\infty} e^{2\pi j t f} df = \delta(t) \quad \sum_{i=1}^{100} \cos(2\pi f_i t) \approx \delta(t)$$

$f_1, \dots, f_{100} \in [0, 10]$  chosen uniformly at random

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

$$\sum_{n=-\infty}^{\infty} e^{\pm 2\pi j n t} = \sum_{n=-\infty}^{\infty} \delta(t - n)$$

Introduction

Sequences and systems

Convolution

**Fourier transform**

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

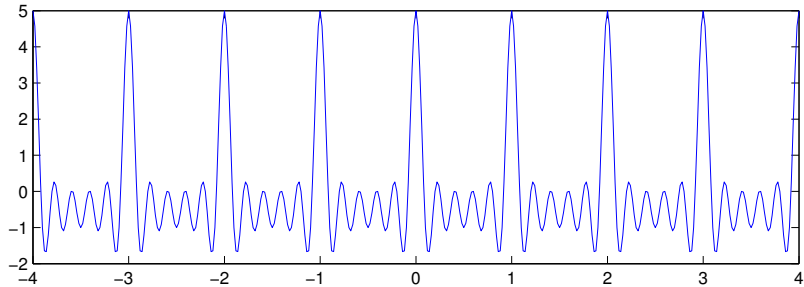
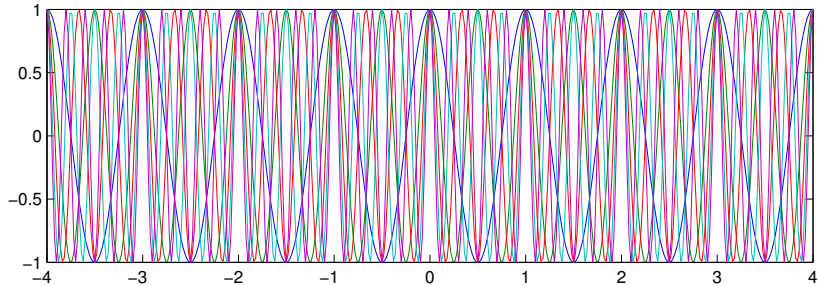
Random signals

Digital communication

Audiovisual data compression

Outlook

$$\sum_{n=1}^5 \cos(2\pi nt) \approx \sum_{n=-\infty}^{\infty} \delta(t - n)$$



Introduction

Sequences and systems

Convolution

**Fourier transform**

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

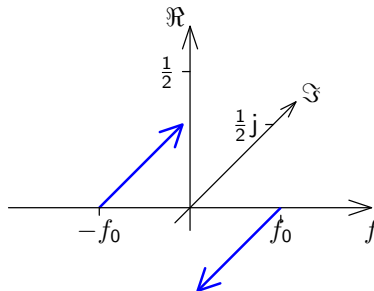
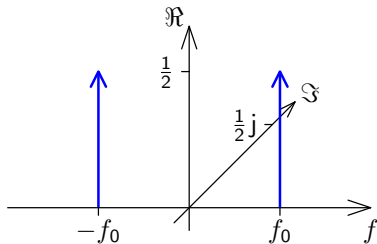
Outlook

# Sine and cosine in the frequency domain

$$\cos(2\pi f_0 t) = \frac{1}{2} e^{2\pi j f_0 t} + \frac{1}{2} e^{-2\pi j f_0 t} \quad \sin(2\pi f_0 t) = \frac{1}{2j} e^{2\pi j f_0 t} - \frac{1}{2j} e^{-2\pi j f_0 t}$$

$$\mathcal{F}\{\cos(2\pi f_0 t)\}(f) = \frac{1}{2} \delta(f - f_0) + \frac{1}{2} \delta(f + f_0)$$

$$\mathcal{F}\{\sin(2\pi f_0 t)\}(f) = -\frac{j}{2} \delta(f - f_0) + \frac{j}{2} \delta(f + f_0)$$



As any  $x(t) \in \mathbb{R}$  can be decomposed into sine and cosine functions, the spectrum of any real-valued signal will show the symmetry  $X(-f) = [X(f)]^*$ , where  $*$  denotes the complex conjugate (i.e., negated imaginary part).

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

We call a function  $x(t)$

$$\text{odd if } x(-t) = -x(t)$$

$$\text{even if } x(-t) = x(t)$$

and  $\cdot^*$  is the complex conjugate, such that  $(a + jb)^* = (a - jb)$ .

Then

|                              |                                              |
|------------------------------|----------------------------------------------|
| $x(t)$ is real               | $\Leftrightarrow X(-f) = [X(f)]^*$           |
| $x(t)$ is imaginary          | $\Leftrightarrow X(-f) = -[X(f)]^*$          |
| $x(t)$ is even               | $\Leftrightarrow X(f)$ is even               |
| $x(t)$ is odd                | $\Leftrightarrow X(f)$ is odd                |
| $x(t)$ is real and even      | $\Leftrightarrow X(f)$ is real and even      |
| $x(t)$ is real and odd       | $\Leftrightarrow X(f)$ is imaginary and odd  |
| $x(t)$ is imaginary and even | $\Leftrightarrow X(f)$ is imaginary and even |
| $x(t)$ is imaginary and odd  | $\Leftrightarrow X(f)$ is real and odd       |

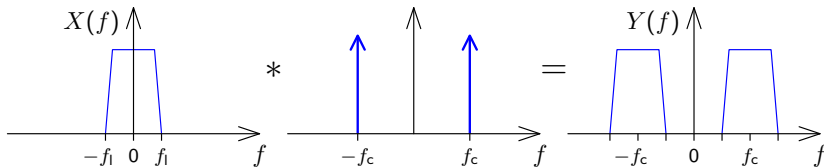
[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

# Example: amplitude modulation

Communication channels usually permit only the use of a given frequency interval, such as 300–3400 Hz for the analog phone network or 590–598 MHz for TV channel 36. Modulation with a carrier frequency  $f_c$  shifts the spectrum of a signal  $x(t)$  into the desired band.

Amplitude modulation (AM):

$$y(t) = A \cdot \cos(2\pi t f_c) \cdot x(t)$$



The spectrum of the baseband signal in the interval  $-f_l < f < f_l$  is shifted by the modulation to the intervals  $\pm f_c - f_l < f < \pm f_c + f_l$ .

How can such a signal be demodulated?

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

- 1 Sequences and systems
- 2 Convolution
- 3 Fourier transform
- 4 Sampling**
- 5 Discrete Fourier transform
- 6 Deconvolution
- 7 Spectral estimation
- 8 Digital filters
- 9 IIR filters
- 10 Random signals
- 11 Digital communication
- 12 Audiovisual data compression

Introduction

Sequences and systems

Convolution

Fourier transform

**Sampling**

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

The loss of information in the sampling process that converts a continuous function  $x(t)$  into a discrete sequence  $\{x_n\}$  defined by

$$x_n = x(t_s \cdot n) = x(n/f_s)$$

can be modelled through multiplying  $x(t)$  by a comb of Dirac impulses

$$s(t) = t_s \cdot \sum_{n=-\infty}^{\infty} \delta(t - t_s \cdot n)$$

to obtain the sampled function

$$\hat{x}(t) = x(t) \cdot s(t)$$

The function  $\hat{x}(t)$  now contains exactly the same information as the discrete sequence  $\{x_n\}$ , but is still in a form that can be analysed using the Fourier transform on continuous functions.

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[\*\*Sampling\*\*](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

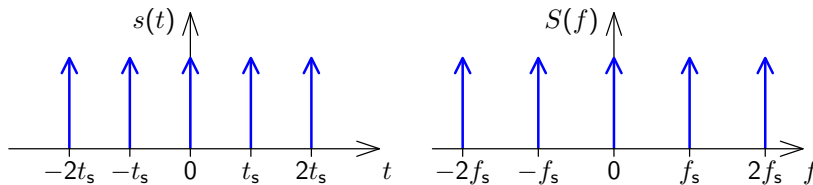
## The Fourier transform of a Dirac comb

$$s(t) = t_s \cdot \sum_{n=-\infty}^{\infty} \delta(t - t_s \cdot n) = \sum_{n=-\infty}^{\infty} e^{2\pi j n t / t_s}$$

is another Dirac comb

$$S(f) = \mathcal{F} \left\{ t_s \cdot \sum_{n=-\infty}^{\infty} \delta(t - t_s n) \right\} (f) =$$

$$t_s \cdot \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \delta(t - t_s n) e^{-2\pi j f t} dt = \sum_{n=-\infty}^{\infty} \delta \left( f - \frac{n}{t_s} \right).$$



Introduction

Sequences and systems

Convolution

Fourier transform

**Sampling**

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

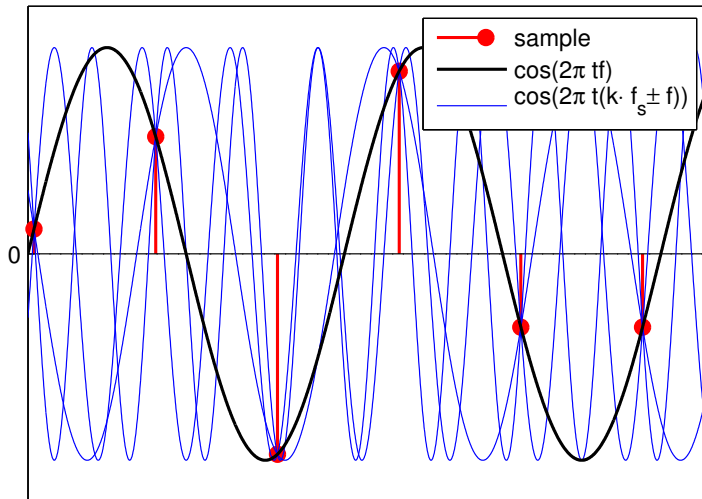
IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook



Sampled at frequency  $f_s$ , the function  $\cos(2\pi tf)$  cannot be distinguished from  $\cos[2\pi t(kf_s \pm f)]$  for any  $k \in \mathbb{Z}$ .

Introduction

Sequences and systems

Convolution

Fourier transform

**Sampling**

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

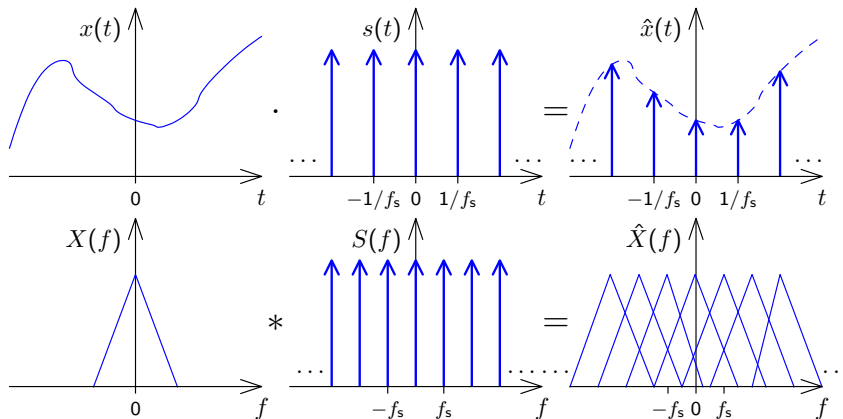
IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook



Sampling a signal in the time domain corresponds in the frequency domain to convolving its spectrum with a Dirac comb. The resulting copies of the original signal spectrum in the spectrum of the sampled signal are called “images”.

Introduction

Sequences and systems

Convolution

Fourier transform

**Sampling**

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

The Fourier transform of a sampled signal

$$\hat{x}(t) = t_s \cdot \sum_{n=-\infty}^{\infty} x_n \cdot \delta(t - t_s \cdot n)$$

is

$$\mathcal{F}\{\hat{x}(t)\}(f) = \hat{X}(f) = \int_{-\infty}^{\infty} \hat{x}(t) \cdot e^{-2\pi j f t} dt = t_s \cdot \sum_{n=-\infty}^{\infty} x_n \cdot e^{-2\pi j \frac{f}{f_s} n}$$

The inverse transform is

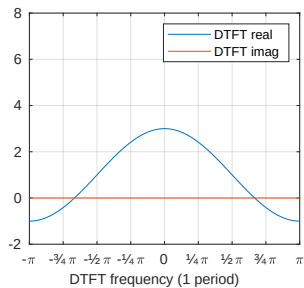
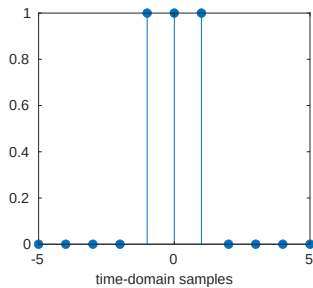
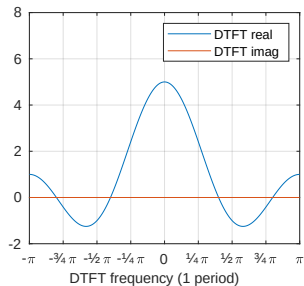
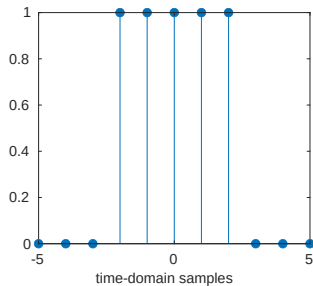
$$\hat{x}(t) = \int_{-\infty}^{\infty} \hat{X}(f) \cdot e^{2\pi j f t} df \quad \text{or} \quad x_m = \int_{-f_s/2}^{f_s/2} \hat{X}(f) \cdot e^{2\pi j \frac{f}{f_s} m} df.$$

The DTFT is also commonly expressed using the normalized frequency  $\hat{\omega} = 2\pi \frac{f}{f_s}$  (radians per sample), and the notation

$$X(e^{j\hat{\omega}}) = \sum_n x_n \cdot e^{-j\hat{\omega}n}$$

is customary, to highlight both the periodicity of the DTFT and its relationship with the  $z$ -transform of  $\{x_n\}$  (see slide 124).

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)



Introduction

Sequences and systems

Convolution

Fourier transform

**Sampling**

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

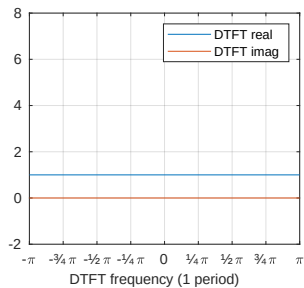
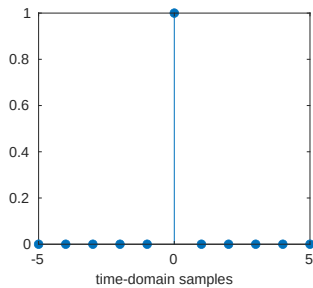
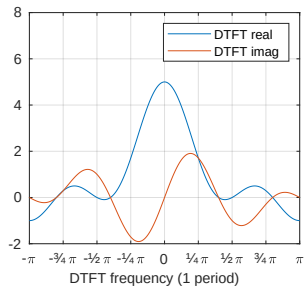
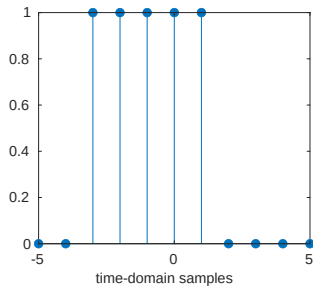
IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook



Introduction

Sequences and systems

Convolution

Fourier transform

**Sampling**

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

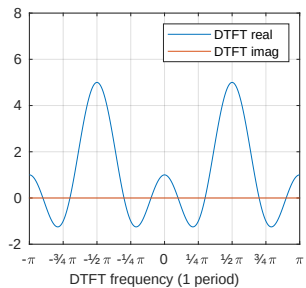
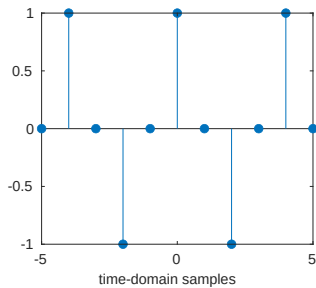
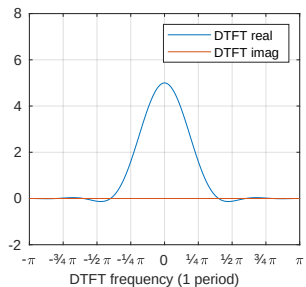
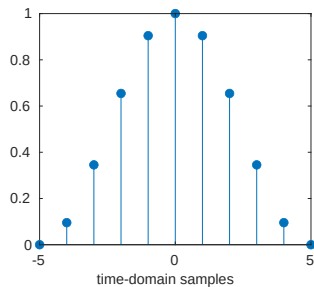
IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook



Introduction

Sequences and systems

Convolution

Fourier transform

**Sampling**

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

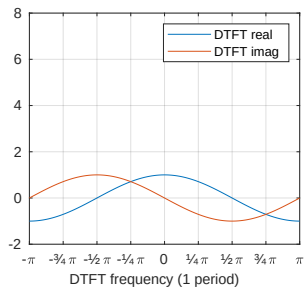
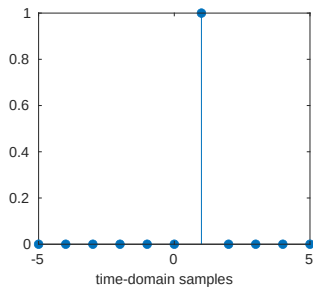
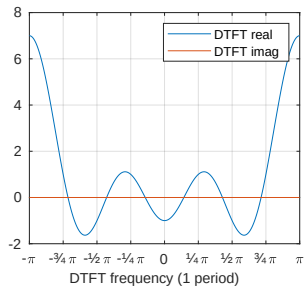
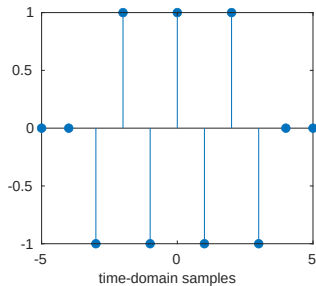
IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook



Introduction

Sequences and systems

Convolution

Fourier transform

**Sampling**

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

The DTFT is periodic:

$$\hat{X}(f) = \hat{X}(f + kf_s) \quad \text{or} \quad X(e^{j\omega}) = X(e^{j(\omega+2\pi k)}) \quad \forall k \in \mathbb{Z}$$

Beyond that, the DTFT is just the Fourier transform applied to a discrete sequence, and inherits the properties of the continuous Fourier transform, e.g.

- ▶ Linearity
- ▶ Symmetries
- ▶ Convolution and modulation theorem:

$$\{x_n\} * \{y_n\} = \{z_n\} \quad \Longleftrightarrow \quad X(e^{j\omega}) \cdot Y(e^{j\omega}) = Z(e^{j\omega})$$

and

$$x_n \cdot y_n = z_n \quad \Longleftrightarrow \quad \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega'}) \cdot Y(e^{j(\omega-\omega')}) d\omega' = Z(e^{j\omega})$$

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

If the (double-sided) bandwidth of a signal to be sampled is larger than the sampling frequency  $f_s$ , the images of the signal that emerge during sampling may overlap with the original spectrum.

Such an overlap will hinder reconstruction of the original continuous signal by removing the aliasing frequencies with a *reconstruction filter*.

Therefore, it is advisable to limit the bandwidth of the input signal to the sampling frequency  $f_s$  before sampling, using an *anti-aliasing filter*.

In the common case of a real-valued base-band signal (with frequency content down to 0 Hz), all frequencies  $f$  that occur in the signal with non-zero power should be limited to the interval  $-f_s/2 < f < f_s/2$ .

The upper limit  $f_s/2$  for the single-sided bandwidth of a baseband signal is known as the “Nyquist limit”.

Introduction

Sequences and systems

Convolution

Fourier transform

**Sampling**

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

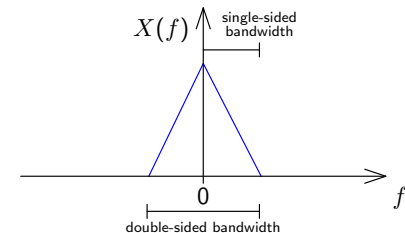
Random signals

Digital communication

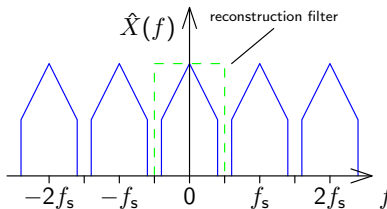
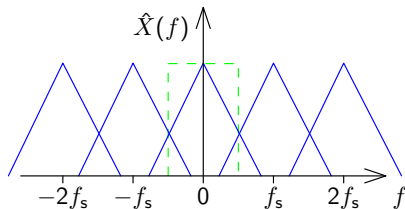
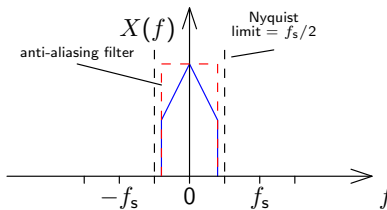
Audiovisual data compression

Outlook

## Without anti-aliasing filter



## With anti-aliasing filter



Anti-aliasing and reconstruction filters both suppress frequencies outside  $|f| < f_s/2$ .

Introduction

Sequences and systems

Convolution

Fourier transform

**Sampling**

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

The ideal anti-aliasing filter for eliminating any frequency content above  $f_s/2$  before sampling with a frequency of  $f_s$  has the Fourier transform

$$H(f) = \begin{cases} 1 & \text{if } |f| < \frac{f_s}{2} \\ 0 & \text{if } |f| > \frac{f_s}{2} \end{cases} = \text{rect}(t_s f).$$

This leads, after an inverse Fourier transform, to the impulse response

$$h(t) = f_s \cdot \frac{\sin \pi t f_s}{\pi t f_s} = \frac{1}{t_s} \cdot \text{sinc}\left(\frac{t}{t_s}\right).$$

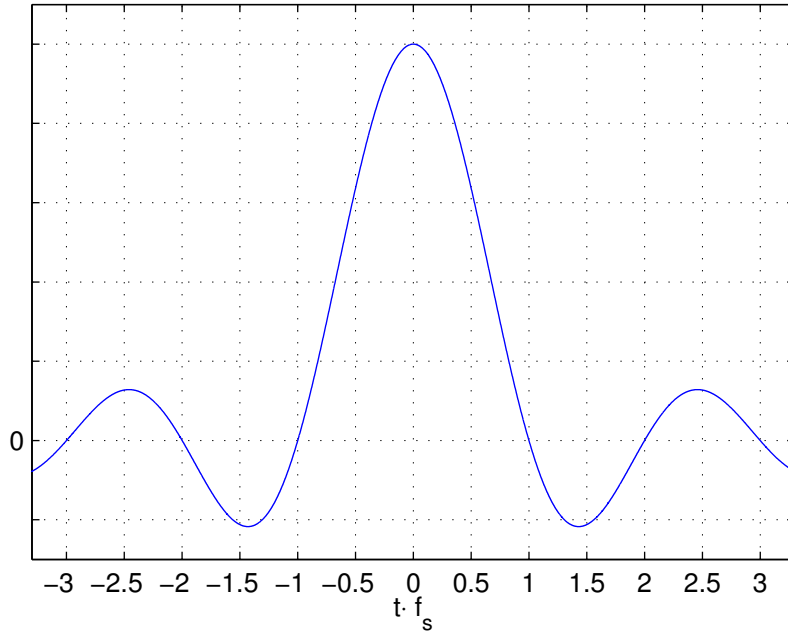
The original band-limited signal can be reconstructed by convolving this with the sampled signal  $\hat{x}(t)$ , which eliminates the periodicity of the frequency domain introduced by the sampling process:

$$x(t) = h(t) * \hat{x}(t)$$

Note that sampling  $h(t)$  gives the impulse function:  $h(t) \cdot s(t) = \delta(t)$ .

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

## Impulse response of ideal low-pass filter with cut-off frequency $f_s/2$ :



Introduction

Sequences and systems

Convolution

Fourier transform

**Sampling**

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

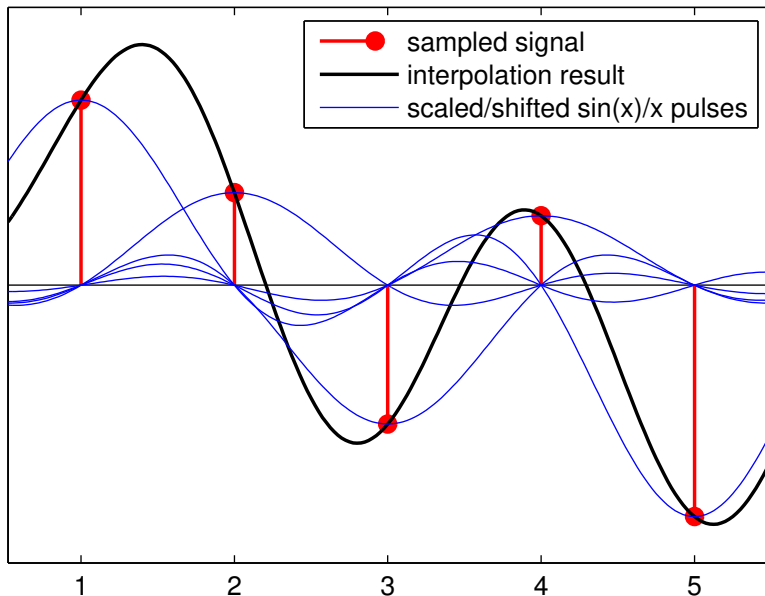
Random signals

Digital communication

Audiovisual data compression

Outlook

# Reconstruction filter example



Introduction

Sequences and systems

Convolution

Fourier transform

**Sampling**

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

If before being sampled with  $x_n = x(n/f_s)$  the signal  $x(t)$  satisfied the Nyquist limit

$$\mathcal{F}\{x(t)\}(f) = \int_{-\infty}^{\infty} x(t) \cdot e^{-2\pi j f t} dt = 0 \quad \text{for all } |f| \geq \frac{f_s}{2}$$

then it can be reconstructed by interpolation with  $h(t) = \frac{1}{t_s} \text{sinc}\left(\frac{t}{t_s}\right)$ :

$$\begin{aligned} x(t) &= \int_{-\infty}^{\infty} h(s) \cdot \hat{x}(t-s) \cdot ds \\ &= \int_{-\infty}^{\infty} \frac{1}{t_s} \text{sinc}\left(\frac{s}{t_s}\right) \cdot t_s \sum_{n=-\infty}^{\infty} x_n \cdot \delta(t-s-t_s \cdot n) \cdot ds \\ &= \sum_{n=-\infty}^{\infty} x_n \cdot \int_{-\infty}^{\infty} \text{sinc}\left(\frac{s}{t_s}\right) \cdot \delta(t-s-t_s \cdot n) \cdot ds \\ &= \sum_{n=-\infty}^{\infty} x_n \cdot \text{sinc}\left(\frac{t-t_s \cdot n}{t_s}\right) = \sum_{n=-\infty}^{\infty} x_n \cdot \text{sinc}(t/t_s - n) \\ &= \sum_{n=-\infty}^{\infty} x_n \cdot \frac{\sin \pi(t/t_s - n)}{\pi(t/t_s - n)} \end{aligned}$$

The mathematically ideal form of a reconstruction filter for suppressing aliasing frequencies interpolates the sampled signal  $x_n = x(t_s \cdot n)$  back into the continuous waveform

$$x(t) = \sum_{n=-\infty}^{\infty} x_n \cdot \frac{\sin \pi(t/t_s - n)}{\pi(t/t_s - n)}.$$

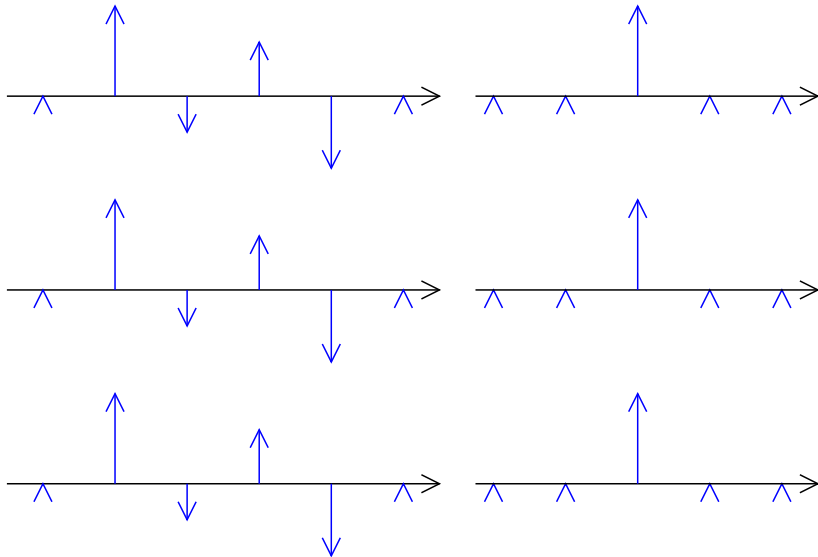
## Choice of sampling frequency

Due to causality and economic constraints, practical analog filters can only approximate such an ideal low-pass filter. Instead of a sharp transition between the “pass band” ( $< f_s/2$ ) and the “stop band” ( $> f_s/2$ ), they feature a “transition band” in which their signal attenuation gradually increases.

The sampling frequency is therefore usually chosen somewhat higher than twice the highest frequency of interest in the continuous signal (e.g.,  $4\times$ ). On the other hand, the higher the sampling frequency, the higher are CPU, power and memory requirements. Therefore, the choice of sampling frequency is a tradeoff between signal quality, analog filter cost and digital subsystem expenses.

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

# Interpolation through convolution



Introduction

Sequences and systems

Convolution

Fourier transform

**Sampling**

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

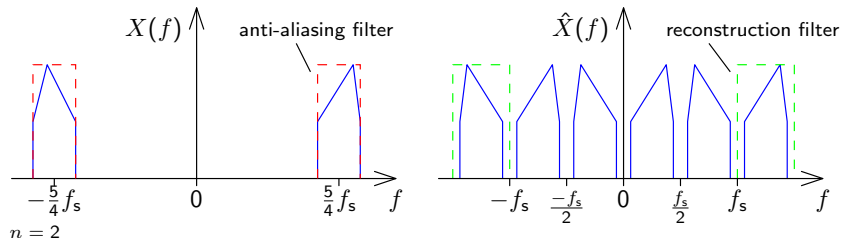
Random signals

Digital communication

Audiovisual data compression

Outlook

Sampled signals can also be reconstructed if their spectral components remain entirely within the interval  $n \cdot f_s/2 < |f| < (n+1) \cdot f_s/2$  for some  $n \in \mathbb{N}$ . (The baseband case discussed so far is just  $n = 0$ .)



In this case, the aliasing copies of the positive and the negative frequencies will interleave instead of overlap, and can therefore be removed again later by a reconstruction filter.

The ideal reconstruction filter for this sampling technique will only allow frequencies in the interval  $[n \cdot f_s/2, (n+1) \cdot f_s/2]$  to pass through. The impulse response of such a *band-pass filter* can be obtained by amplitude modulating a low-pass filter, or by subtracting two low-pass filters:

$$h(t) = f_s \frac{\sin \pi t f_s / 2}{\pi t f_s / 2} \cdot \cos \left( 2\pi t f_s \frac{2n+1}{4} \right) = (n+1) f_s \frac{\sin \pi t (n+1) f_s}{\pi t (n+1) f_s} - n f_s \frac{\sin \pi t n f_s}{\pi t n f_s}.$$

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

- ① Sequences and systems
- ② Convolution
- ③ Fourier transform
- ④ Sampling
- ⑤ **Discrete Fourier transform**
  - FFT
  - FFT-based convolution
- ⑥ Deconvolution
- ⑦ Spectral estimation
- ⑧ Digital filters
- ⑨ IIR filters
- ⑩ Random signals
- ⑪ Digital communication
- ⑫ Audiovisual data compression

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

**Discrete Fourier transform**

FFT

FFT-based convolution

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

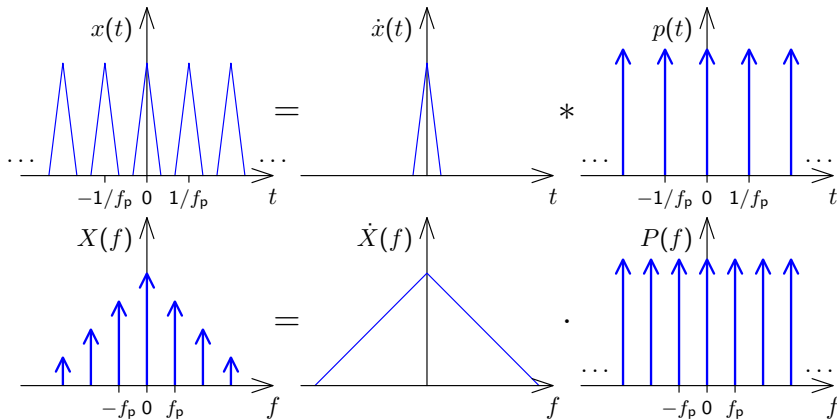
Digital communication

Audiovisual data compression

Outlook

# Spectrum of a periodic signal

A signal  $x(t)$  that is periodic with frequency  $f_p$  can be factored into a single period  $\dot{x}(t)$  convolved with an impulse comb  $p(t)$ . This corresponds in the frequency domain to the multiplication of the spectrum of the single period with a comb of impulses spaced  $f_p$  apart.



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

FFT

FFT-based convolution

Deconvolution

Spectral estimation

Digital filters

IIR filters

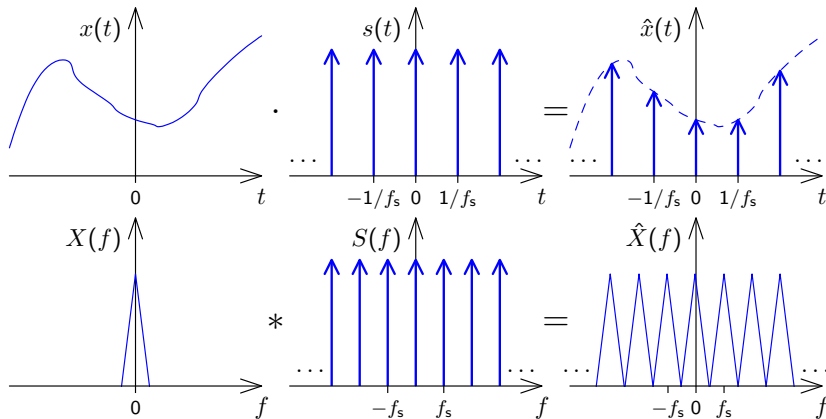
Random signals

Digital communication

Audiovisual data compression

Outlook

A signal  $x(t)$  that is sampled with frequency  $f_s$  has a spectrum that is periodic with a period of  $f_s$ .



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

FFT

FFT-based convolution

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

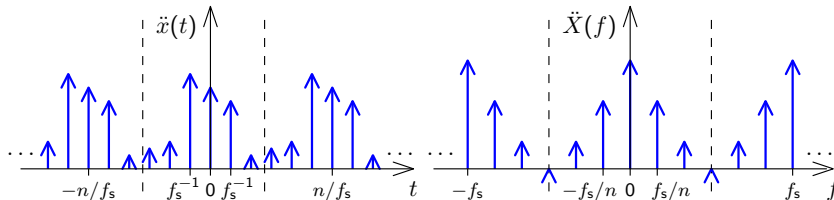
Audiovisual data compression

Outlook

- ▶ Sampling a **continuous** signal makes its spectrum **periodic**
- ▶ A **periodic** signal has a **sampled** spectrum

We **sample** a signal  $x(t)$  with  $f_s$ , getting  $\hat{x}(t)$ . We take  $n$  consecutive samples of  $\hat{x}(t)$  and **repeat** these periodically, getting a new signal  $\ddot{x}(t)$  with period  $n/f_s$ . Its spectrum  $\ddot{X}(f)$  is **sampled** (i.e., has non-zero value) at frequency intervals  $f_s/n$  and **repeats** itself with a period  $f_s$ .

Now both  $\ddot{x}(t)$  and its spectrum  $\ddot{X}(f)$  are **finite** vectors of length  $n$ .



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

FFT

FFT-based convolution

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

If  $x(t)$  has period  $t_p = n \cdot t_s$ , then after sampling we have

$$\ddot{x}(t) = x(t) \cdot s(t) = t_s \cdot \sum_{i=-\infty}^{\infty} x_i \cdot \delta(t - t_s \cdot i) = t_s \cdot \sum_{l=-\infty}^{\infty} \sum_{i=0}^{n-1} x_i \cdot \delta(t - t_s \cdot (i + nl))$$

and the Fourier transform of that is

$$\begin{aligned} \mathcal{F}\{\ddot{x}(t)\}(f) &= \ddot{X}(f) = \int_{-\infty}^{\infty} \ddot{x}(t) \cdot e^{-2\pi j f t} dt \\ &= t_s \cdot \sum_{l=-\infty}^{\infty} \sum_{i=0}^{n-1} x_i \cdot e^{-2\pi j \frac{f}{f_s} \cdot (i + nl)} = t_s \cdot \underbrace{\sum_{l=-\infty}^{\infty} e^{-2\pi j \frac{f}{f_s} \cdot nl}}_{\frac{1}{t_s n} \sum_l \delta(f - \frac{l}{n} f_s)} \cdot \sum_{i=0}^{n-1} x_i \cdot e^{-2\pi j \frac{f}{f_s} \cdot i} \end{aligned}$$

Recall that  $\sum_{i=-\infty}^{\infty} e^{\pm 2\pi j i x a} = \frac{1}{|a|} \sum_{i=-\infty}^{\infty} \delta(x - i/a)$  and map  $x = f$ ,  $a = \frac{n}{f_s}$  and  $i = l$ .

After substituting  $k := \frac{f}{f_p} = \frac{f}{f_s} n$ , i.e.  $\frac{f}{f_s} = \frac{k}{n}$  and  $f = k f_p$

$$\begin{aligned} \ddot{X}(k f_p) &= \frac{1}{n} \cdot \underbrace{\sum_{l=-\infty}^{\infty} \delta(k f_p - l f_p)}_{\substack{\delta(0) \quad \text{if } k \in \mathbb{Z} \\ 0 \quad \text{if } k \notin \mathbb{Z}}} \cdot \underbrace{\sum_{i=0}^{n-1} x_i \cdot e^{-2\pi j \frac{k i}{n}}}_{X_k} \\ &= \begin{cases} \delta(0) & \text{if } k \in \mathbb{Z} \\ 0 & \text{if } k \notin \mathbb{Z} \end{cases} \end{aligned}$$

Show that  $X_k = X_{k \pm n}$  for all  $k \in \mathbb{Z}$ .

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

FFT

FFT-based convolution

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

$$X_k = \sum_{i=0}^{n-1} x_i \cdot e^{-2\pi j \frac{ik}{n}} \quad x_k = \frac{1}{n} \cdot \sum_{i=0}^{n-1} X_i \cdot e^{2\pi j \frac{ik}{n}}$$

The  $n$ -point DFT multiplies a vector with an  $n \times n$  matrix

$$F_n = \begin{pmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & e^{-2\pi j \frac{1}{n}} & e^{-2\pi j \frac{2}{n}} & e^{-2\pi j \frac{3}{n}} & \dots & e^{-2\pi j \frac{n-1}{n}} \\ 1 & e^{-2\pi j \frac{2}{n}} & e^{-2\pi j \frac{4}{n}} & e^{-2\pi j \frac{6}{n}} & \dots & e^{-2\pi j \frac{2(n-1)}{n}} \\ 1 & e^{-2\pi j \frac{3}{n}} & e^{-2\pi j \frac{6}{n}} & e^{-2\pi j \frac{9}{n}} & \dots & e^{-2\pi j \frac{3(n-1)}{n}} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & e^{-2\pi j \frac{n-1}{n}} & e^{-2\pi j \frac{2(n-1)}{n}} & e^{-2\pi j \frac{3(n-1)}{n}} & \dots & e^{-2\pi j \frac{(n-1)(n-1)}{n}} \end{pmatrix}$$

$$F_n \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{n-1} \end{pmatrix} = \begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ \vdots \\ X_{n-1} \end{pmatrix}, \quad \frac{1}{n} \cdot F_n^* \cdot \begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ \vdots \\ X_{n-1} \end{pmatrix} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{n-1} \end{pmatrix}$$

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

FFT

FFT-based convolution

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

$$\begin{pmatrix}
 \begin{array}{c} \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \end{array} &
 \begin{array}{c} \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \end{array} &
 \begin{array}{c} \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \end{array} &
 \begin{array}{c} \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \end{array} &
 \begin{array}{c} \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \end{array} &
 \begin{array}{c} \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \end{array} &
 \begin{array}{c} \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \end{array} &
 \begin{array}{c} \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \\ \text{phasor} \end{array}
 \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} = \begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \end{pmatrix}$$

The  $n$ -point DFT of a signal  $\{x_i\}$  sampled at frequency  $f_s$  contains in the elements  $X_0$  to  $X_{n/2}$  of the resulting frequency-domain vector the frequency components  $0, f_s/n, 2f_s/n, 3f_s/n, \dots, f_s/2$ , and contains in  $X_{n-1}$  down to  $X_{n/2}$  the corresponding negative frequencies. Note that for a real-valued input vector, both  $X_0$  and  $X_{n/2}$  will be real, too.

Why is there no phase information recovered at  $f_s/2$ ?

[Introduction](#)
[Sequences and systems](#)
[Convolution](#)
[Fourier transform](#)
[Sampling](#)
[Discrete Fourier transform](#)
[FFT](#)
[FFT-based convolution](#)
[Deconvolution](#)
[Spectral estimation](#)
[Digital filters](#)
[IIR filters](#)
[Random signals](#)
[Digital communication](#)
[Audiovisual data compression](#)
[Outlook](#)

$$\frac{1}{8} \cdot \begin{pmatrix} \text{8 DFT basis vectors} \end{pmatrix} \cdot \begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \end{pmatrix} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix}$$

The diagram illustrates the inverse Discrete Fourier Transform (IDFT) process. It shows a matrix of 8 DFT basis vectors (each a circle with a horizontal axis and a vertical axis, with arrows indicating the direction of the basis function) multiplied by a vector of 8 frequency components  $X_0$  through  $X_7$ . The result is a vector of 8 time-domain samples  $x_0$  through  $x_7$ . The scaling factor  $\frac{1}{8}$  is shown to the left of the basis vector matrix.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

**Discrete Fourier transform**

FFT

FFT-based convolution

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

$$\begin{aligned} \left( \mathcal{F}_n \{x_i\}_{i=0}^{n-1} \right)_k &= \sum_{i=0}^{n-1} x_i \cdot e^{-2\pi j \frac{ik}{n}} \\ &= \sum_{i=0}^{\frac{n}{2}-1} x_{2i} \cdot e^{-2\pi j \frac{ik}{n/2}} + e^{-2\pi j \frac{k}{n}} \sum_{i=0}^{\frac{n}{2}-1} x_{2i+1} \cdot e^{-2\pi j \frac{ik}{n/2}} \\ &= \begin{cases} \left( \mathcal{F}_{\frac{n}{2}} \{x_{2i}\}_{i=0}^{\frac{n}{2}-1} \right)_k + e^{-2\pi j \frac{k}{n}} \cdot \left( \mathcal{F}_{\frac{n}{2}} \{x_{2i+1}\}_{i=0}^{\frac{n}{2}-1} \right)_k, & k < \frac{n}{2} \\ \left( \mathcal{F}_{\frac{n}{2}} \{x_{2i}\}_{i=0}^{\frac{n}{2}-1} \right)_{k-\frac{n}{2}} + e^{-2\pi j \frac{k}{n}} \cdot \left( \mathcal{F}_{\frac{n}{2}} \{x_{2i+1}\}_{i=0}^{\frac{n}{2}-1} \right)_{k-\frac{n}{2}}, & k \geq \frac{n}{2} \end{cases} \end{aligned}$$

The DFT over  $n$ -element vectors can be reduced to two DFTs over  $n/2$ -element vectors plus  $n$  multiplications and  $n$  additions, leading to  $\log_2 n$  rounds and  $n \log_2 n$  additions and multiplications overall, compared to  $n^2$  for the equivalent matrix multiplication.

A high-performance FFT implementation in C with many processor-specific optimizations and support for non-power-of-2 sizes is available at <https://www.fftw.org/>. Julia wrapper: `FFTW.jl`

Some CPU vendors offer even faster ones, such as the *Intel Math Kernel Library (MKL)* or *Arm Performance Libraries*. Hardware implementations: <https://www.spiral.net/>.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

**FFT**

FFT-based convolution

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

The symmetry properties of the Fourier transform applied to the discrete Fourier transform  $\{X_i\}_{i=0}^{n-1} = \mathcal{F}_n\{x_i\}_{i=0}^{n-1}$  have the form

$$\forall i : x_i = \Re(x_i) \iff \forall i : X_{n-i} = X_i^*$$

$$\forall i : x_i = j \cdot \Im(x_i) \iff \forall i : X_{n-i} = -X_i^*$$

These two symmetries, combined with the linearity of the DFT, allow us to calculate two real-valued  $n$ -point DFTs

$$\{X'_i\}_{i=0}^{n-1} = \mathcal{F}_n\{x'_i\}_{i=0}^{n-1} \quad \{X''_i\}_{i=0}^{n-1} = \mathcal{F}_n\{x''_i\}_{i=0}^{n-1}$$

simultaneously in a single complex-valued  $n$ -point DFT, by composing its input as

$$x_i = x'_i + j \cdot x''_i$$

and decomposing its output as

$$X'_i = \frac{1}{2}(X_i + X_{n-i}^*) \quad X''_i = \frac{1}{2j}(X_i - X_{n-i}^*)$$

where  $X_n = X_0$ .

To optimize the calculation of a single real-valued FFT, use this trick to calculate the two half-size real-valued FFTs that occur in the first round.

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)**[FFT](#)**[FFT-based convolution](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

Calculating the product of two complex numbers as

$$(a + jb) \cdot (c + jd) = (ac - bd) + j(ad + bc)$$

involves four (real-valued) multiplications and two additions.

The alternative calculation

$$(a + jb) \cdot (c + jd) = (\alpha - \beta) + j(\alpha + \gamma) \quad \text{with} \quad \begin{aligned} \alpha &= a(c + d) \\ \beta &= d(a + b) \\ \gamma &= c(b - a) \end{aligned}$$

provides the same result with three multiplications and five additions.

The latter may perform faster on CPUs where multiplications take three or more times longer than additions.

This “Karatsuba multiplication” is most helpful on simpler microcontrollers. Specialized signal-processing CPUs (DSPs) feature 1-clock-cycle multipliers. High-end desktop processors use pipelined multipliers that stall where operations depend on each other.

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[FFT](#)[FFT-based convolution](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

## Fourier transform:

$$X(\omega) = \int_{-\infty}^{\infty} x(t) \cdot e^{-j\omega t} dt$$

- ▶ time domain: continuous
- ▶ freq. domain: continuous

**Discrete-time Fourier transform (DTFT):**  $X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x_n \cdot e^{-j\omega n}$

- ▶ time domain: discrete sequence
- ▶ freq. domain: continuous

## Discrete Fourier transform (DFT):

$$X_k = \sum_{n=0}^{l-1} x_n \cdot e^{-2\pi j \frac{nk}{l}}$$

- ▶ time domain: periodic discrete-sequence (degree- $l$  vector)
- ▶ freq. domain: periodic discrete-sequence (degree- $l$  vector)
- ▶ also: the result of sampling the DTFT of an  $l$ -sample finite-support sequence  $\{x_n\}_{n=0}^{l-1}$  at frequencies  $\omega = 2\pi \frac{k}{l}$  for  $k \in \{0, \dots, l-1\}$

## Fast Fourier transform (FFT):

- ▶ a fast algorithm for calculating the DFT (in  $n \log n$  steps)

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

**FFT**

FFT-based convolution

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

Calculating the convolution of two finite sequences  $\{x_i\}_{i=0}^{m-1}$  and  $\{y_i\}_{i=0}^{n-1}$  of lengths  $m$  and  $n$  via

$$z_i = \sum_{j=\max\{0, i-(n-1)\}}^{\min\{m-1, i\}} x_j \cdot y_{i-j}, \quad 0 \leq i < m + n - 1$$

takes  $mn$  multiplications.

Can we apply the FFT and the convolution theorem to calculate the convolution faster, in just  $O(m \log m + n \log n)$  multiplications?

$$\{z_i\} = \mathcal{F}^{-1}(\mathcal{F}\{x_i\} \cdot \mathcal{F}\{y_i\})$$

There is obviously no problem if this condition is fulfilled:

$\{x_i\}$  and  $\{y_i\}$  are periodic, with equal period lengths

In this case, the fact that the DFT interprets its input as a single period of a periodic signal will do exactly what is needed, and the FFT and inverse FFT can be applied directly as above.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

FFT

FFT-based convolution

Deconvolution

Spectral estimation

Digital filters

IIR filters

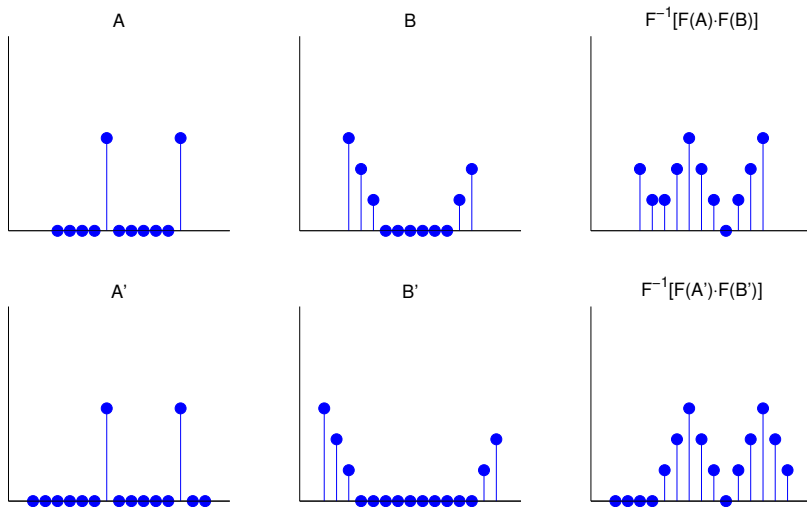
Random signals

Digital communication

Audiovisual data compression

Outlook

In the general case, measures have to be taken to prevent a wrap-over:



Both sequences are padded with zero values to a length of at least  $m + n - 1$ . This ensures that the start and end of the resulting sequence do not overlap.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

FFT

FFT-based convolution

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

Zero padding is usually applied to extend both sequence lengths to the next higher power of two ( $2^{\lceil \log_2(m+n-1) \rceil}$ ), which facilitates the FFT.

With a causal sequence, simply append the padding zeros at the end.

With a non-causal sequence, values with a negative index number are wrapped around the DFT block boundaries and appear at the right end. In this case, zero-padding is applied in the center of the block, between the last and first element of the sequence.

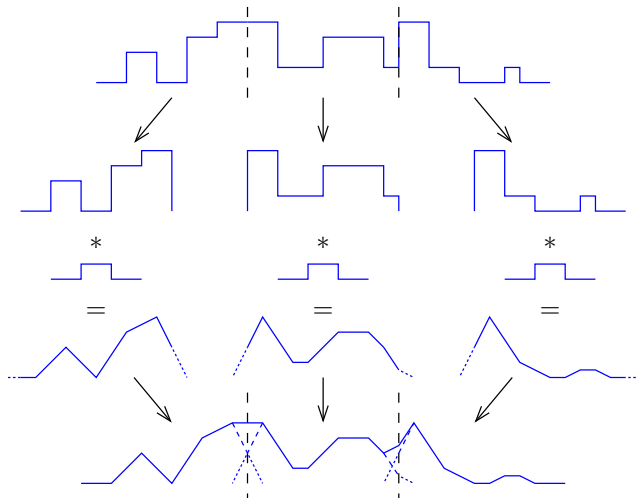
Thanks to the periodic nature of the DFT, zero padding at both ends has the same effect as padding only at one end.

If both sequences can be loaded entirely into RAM, the FFT can be applied to them in one step. However, one of the sequences might be too large for that. It could also be a real-time waveform (e.g., a telephone signal) that cannot be delayed until the end of the transmission.

In such cases, the sequence has to be split into shorter blocks that are separately convolved and then added together with a suitable overlap.

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[FFT](#)[FFT-based convolution](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

Each block is zero-padded at both ends and then convolved as before:



The regions originally added as zero padding are, after convolution, aligned to overlap with the unpadded ends of their respective neighbour blocks. The overlapping parts of the blocks are then added together.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

FFT

FFT-based convolution

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

- 1 Sequences and systems
- 2 Convolution
- 3 Fourier transform
- 4 Sampling
- 5 Discrete Fourier transform
- 6 Deconvolution**
- 7 Spectral estimation
- 8 Digital filters
- 9 IIR filters
- 10 Random signals
- 11 Digital communication
- 12 Audiovisual data compression

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

**Deconvolution**

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

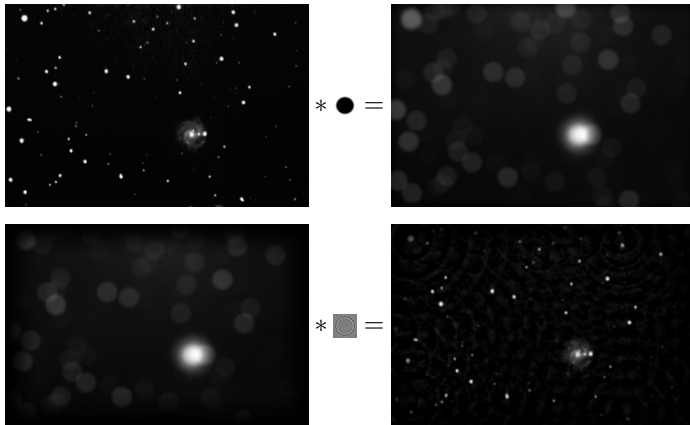
Audiovisual data compression

Outlook

# Deconvolution

A signal  $u(t)$  was distorted by convolution with a known impulse response  $h(t)$  (e.g., through a transmission channel or a sensor problem). The “smeared” result  $s(t)$  was recorded.

Can we undo the damage and restore (or at least estimate)  $u(t)$ ?



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

**Deconvolution**

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

The convolution theorem turns the problem into one of multiplication:

$$s(t) = \int u(t - \tau) \cdot h(\tau) \cdot d\tau$$

$$s = u * h$$

$$\mathcal{F}\{s\} = \mathcal{F}\{u\} \cdot \mathcal{F}\{h\}$$

$$\mathcal{F}\{u\} = \mathcal{F}\{s\} / \mathcal{F}\{h\}$$

$$u = \mathcal{F}^{-1}\{\mathcal{F}\{s\} / \mathcal{F}\{h\}\}$$

In practice, we also record some noise  $n(t)$  (quantization, etc.):

$$c(t) = s(t) + n(t) = \int u(t - \tau) \cdot h(\tau) \cdot d\tau + n(t)$$

**Problem** – At frequencies  $f$  where  $\mathcal{F}\{h\}(f)$  approaches zero, the noise will be amplified (potentially enormously) during deconvolution:

$$\tilde{u} = \mathcal{F}^{-1}\{\mathcal{F}\{c\} / \mathcal{F}\{h\}\} = u + \mathcal{F}^{-1}\{\mathcal{F}\{n\} / \mathcal{F}\{h\}\}$$

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

**Deconvolution**

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

Typical workarounds:

- ▶ Modify the Fourier transform of the impulse response, such that  $|\mathcal{F}\{h\}(f)| > \epsilon$  for some experimentally chosen threshold  $\epsilon$ .
- ▶ If estimates of the signal spectrum  $|\mathcal{F}\{s\}(f)|$  and the noise spectrum  $|\mathcal{F}\{n\}(f)|$  can be obtained, then we can apply the “Wiener filter” (“optimal filter”)

$$W(f) = \frac{|\mathcal{F}\{s\}(f)|^2}{|\mathcal{F}\{s\}(f)|^2 + |\mathcal{F}\{n\}(f)|^2}$$

before deconvolution:

$$\tilde{u} = \mathcal{F}^{-1}\{W \cdot \mathcal{F}\{c\} / \mathcal{F}\{h\}\}$$

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

- ① Sequences and systems
- ② Convolution
- ③ Fourier transform
- ④ Sampling
- ⑤ Discrete Fourier transform
- ⑥ Deconvolution
- ⑦ Spectral estimation**
  - Window functions
  - Padding
- ⑧ Digital filters
- ⑨ IIR filters
- ⑩ Random signals
- ⑪ Digital communication
- ⑫ Audiovisual data compression

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

**Spectral estimation**

Window functions

Padding

Digital filters

IIR filters

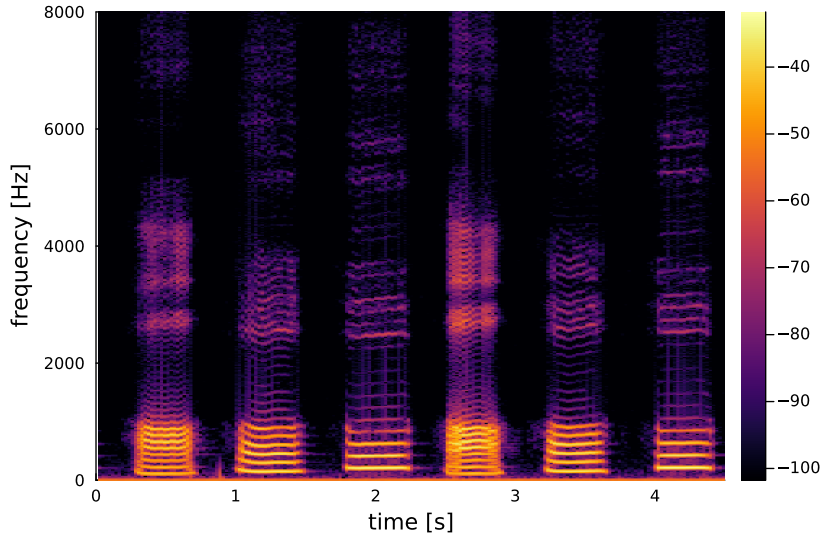
Random signals

Digital communication

Audiovisual data compression

Outlook

## Vowel "A" sung at varying pitch



```
(w, fs, bits) = wavread("sing.wav")
s = spectrogram(w[:,1], 2048; fs, window=hamming)
ps = 10*log10.(power(s)); mx = maximum(ps)
heatmap(time(s), freq(s), ps; xlabel="time [s]", ylabel="frequency [Hz]",
        xlim=(0, 4.5), ylim=(0, 8000.0), clim=(mx-70, mx))
```

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

**Spectral estimation**

Window functions

Padding

Digital filters

IIR filters

Random signals

Digital communication

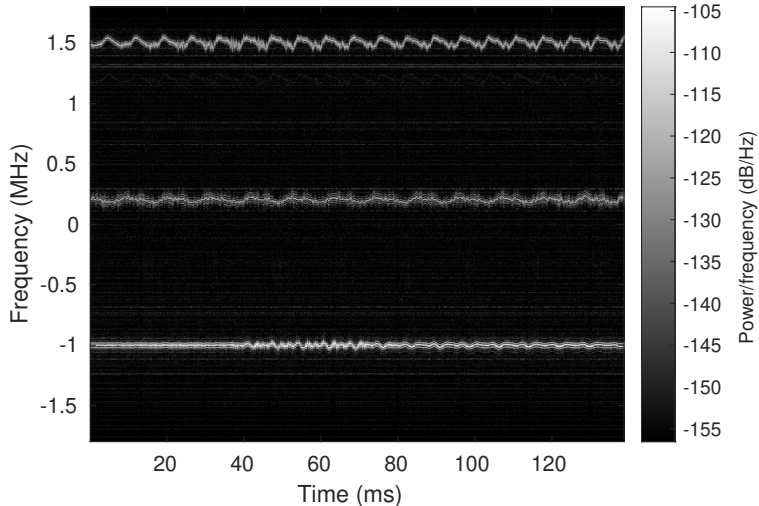
Audiovisual data compression

Outlook

## aeiou.wav

Markus Kuhn





```
f = fopen('iq-fm-97M-3.6M.dat', 'r', 'ieee-le');
c = fread(f, [2,inf], '*float32');
fclose(f);
z = complex(c(1,:), c(2,:));
fs = 3.6e6; % IQ sampling frequency
fcfq = 97e6; % center frequency of IQ downconverter
spectrogram(z(1:5e5), 1024, 512, 1024, fs, 'centered', 'yaxis');
colormap(gray)
```

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

**Spectral estimation**

Window functions

Padding

Digital filters

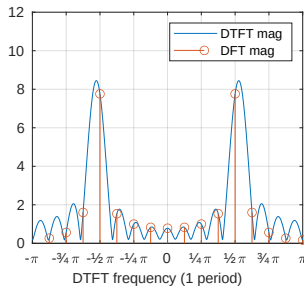
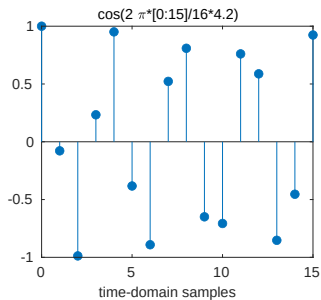
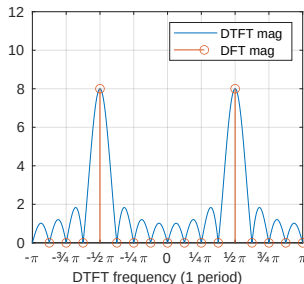
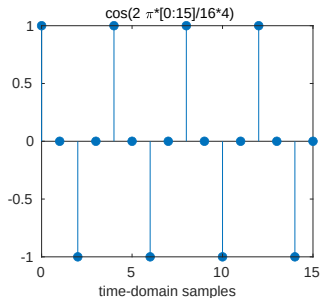
IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

**Spectral estimation**

Window functions

Padding

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

We introduced the DFT as a special case of the continuous Fourier transform, where the input is sampled *and periodic*.

If the input is sampled, but not periodic, the DFT can still be used to calculate an approximation of the Fourier transform of the original continuous signal. However, there are two effects to consider. They are particularly visible when analysing pure sine waves.

Sine waves whose frequency is a multiple of the base frequency ( $f_s/n$ ) of the DFT are identical to their periodic extension beyond the size of the DFT. They are, therefore, represented exactly by a single sharp peak in the DFT. All their energy falls into one single frequency “bin” in the DFT result.

Sine waves with other frequencies, which do not match exactly one of the output frequency bins of the DFT, are still represented by a peak at the output bin that represents the nearest integer multiple of the DFT's base frequency. However, such a peak is distorted in two ways:

- ▶ Its amplitude is lower (down to 63.7%).
- ▶ Much signal energy has “leaked” to other frequencies.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

**Spectral estimation**

Window functions

Padding

Digital filters

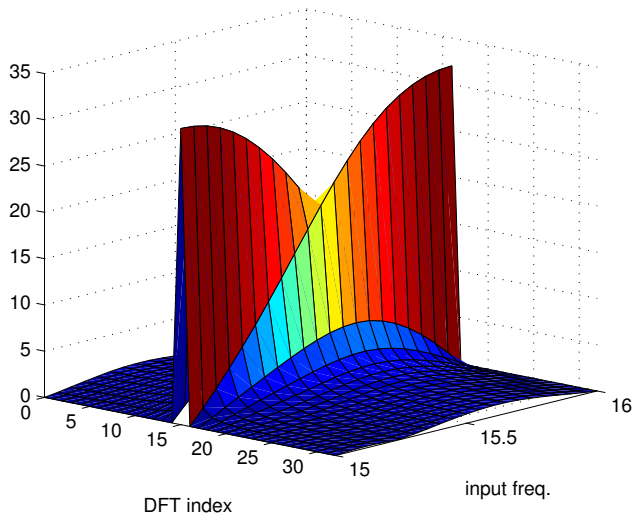
IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook



The *leakage* of energy to other frequency bins not only blurs the estimated spectrum. The peak amplitude also changes significantly as the frequency of a tone changes from that associated with one output bin to the next, a phenomenon known as *scalloping*. In the above graphic, an input sine wave gradually changes from the frequency of bin 15 to that of bin 16 (only positive frequencies shown).

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

**Spectral estimation**

Window functions

Padding

Digital filters

IIR filters

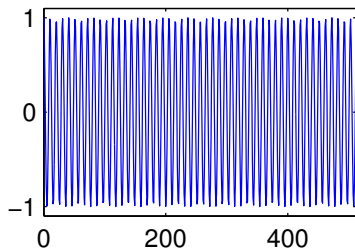
Random signals

Digital communication

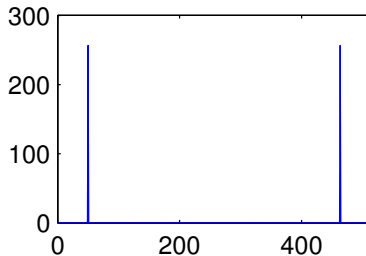
Audiovisual data compression

Outlook

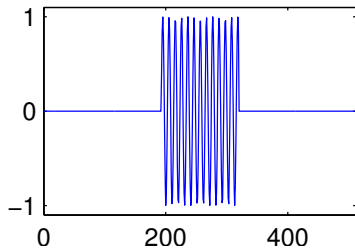
Sine wave



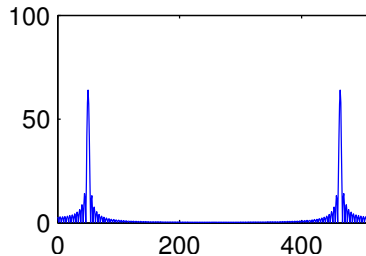
Discrete Fourier Transform



Sine wave multiplied with window function



Discrete Fourier Transform



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Window functions

Padding

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

The reason for the leakage and scalloping losses is easy to visualize with the help of the convolution theorem:

The operation of cutting a sequence of the size of the DFT input vector out of a longer original signal (the one whose continuous Fourier spectrum we try to estimate) is equivalent to multiplying this signal with a rectangular function. This destroys all information and continuity outside the “window” that is fed into the DFT.

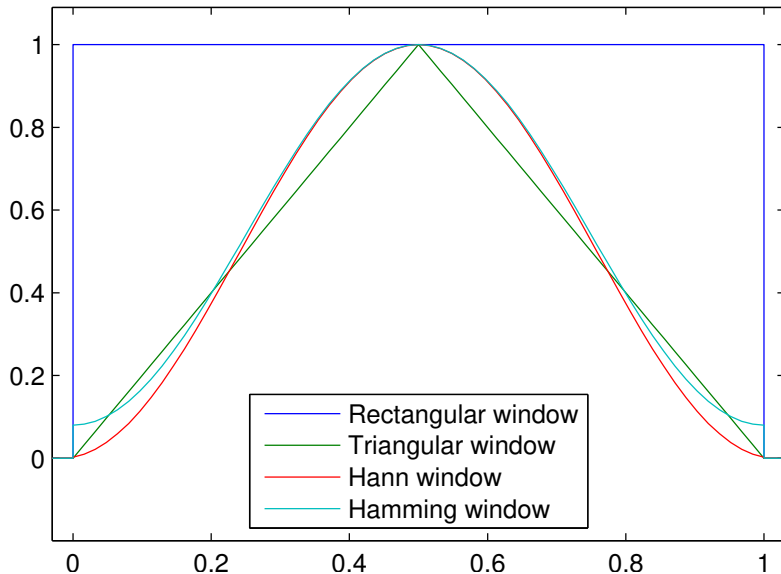
Multiplication with a rectangular window of length  $T$  in the time domain is equivalent to convolution with  $\sin(\pi f T)/(\pi f T)$  in the frequency domain.

The subsequent interpretation of this window as a periodic sequence by the DFT leads to sampling of this convolution result (sampling meaning multiplication with a Dirac comb whose impulses are spaced  $f_s/n$  apart).

Where the window length was an exact multiple of the original signal period, sampling of the  $\sin(\pi f T)/(\pi f T)$  curve leads to a single Dirac pulse, and the windowing causes no distortion. In all other cases, the effects of the convolution become visible in the frequency domain as leakage and scalloping losses.

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Window functions](#)[Padding](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

# Some better window functions



All these functions are 0 outside the interval  $[0,1]$ .

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Window functions

Padding

Digital filters

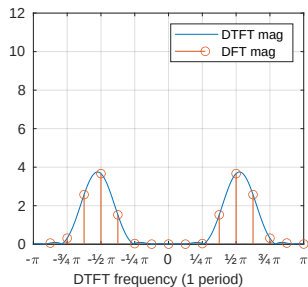
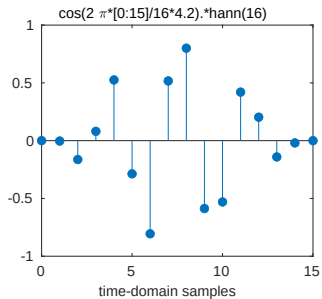
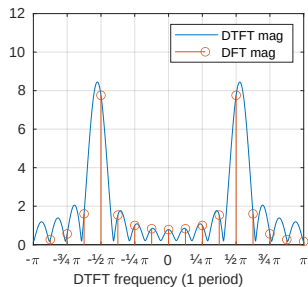
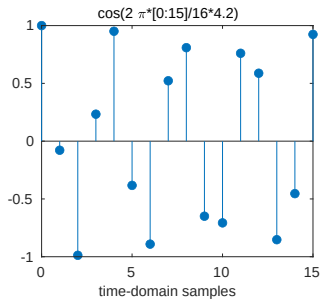
IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Window functions

Padding

Digital filters

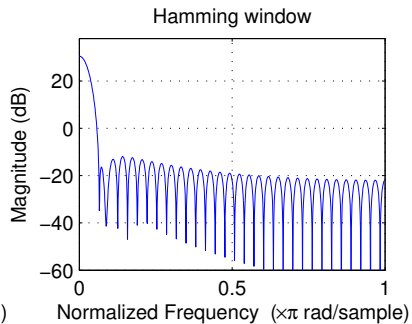
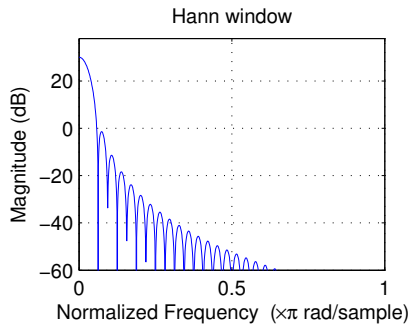
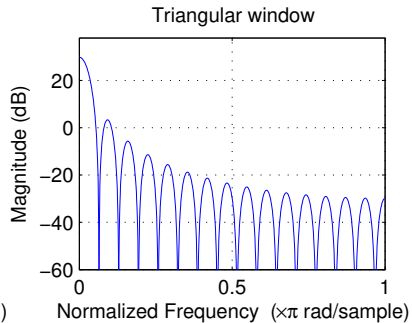
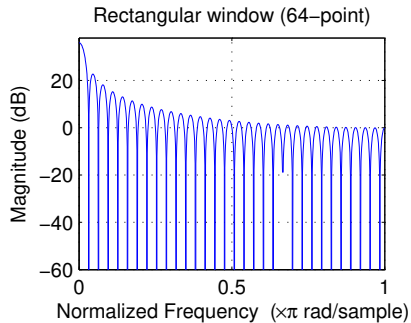
IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Window functions](#)[Padding](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

Numerous alternatives to the rectangular window have been proposed that reduce leakage and scalloping in spectral estimation. These are vectors multiplied element-wise with the input vector before applying the DFT to it. They all force the signal amplitude smoothly down to zero at the edge of the window, thereby avoiding the introduction of sharp jumps in the signal when it is extended periodically by the DFT.

Three examples of such window vectors  $\{w_i\}_{i=0}^{n-1}$  are:

**Triangular window** (Bartlett window):

$$w_i = 1 - \left| 1 - \frac{i}{n/2} \right|$$

**Hann window** (raised-cosine window, Hanning window):

$$w_i = 0.5 - 0.5 \times \cos \left( 2\pi \frac{i}{n-1} \right)$$

**Hamming window**:

$$w_i = 0.54 - 0.46 \times \cos \left( 2\pi \frac{i}{n-1} \right)$$

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Window functions](#)[Padding](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

# Does zero padding increase DFT resolution?

The two figures below show two spectra of the 16-element sequence

$$s_i = \cos(2\pi \cdot 3i/16) + \cos(2\pi \cdot 4i/16), \quad i \in \{0, \dots, 15\}.$$

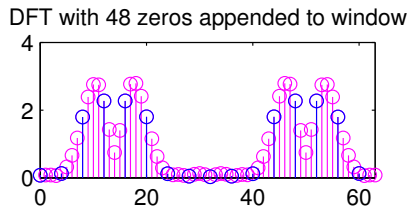
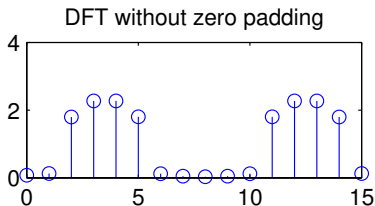
The left plot shows the DFT of the windowed sequence

$$x_i = s_i \cdot w_i, \quad i \in \{0, \dots, 15\}$$

and the right plot shows the DFT of the zero-padded windowed sequence

$$x'_i = \begin{cases} s_i \cdot w_i, & i \in \{0, \dots, 15\} \\ 0, & i \in \{16, \dots, 63\} \end{cases}$$

where  $w_i = 0.54 - 0.46 \times \cos(2\pi i/15)$  is the Hamming window.



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Window functions

Padding

Digital filters

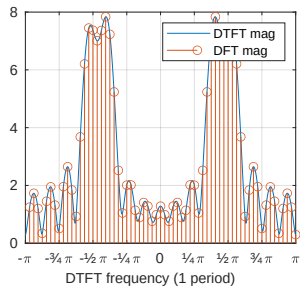
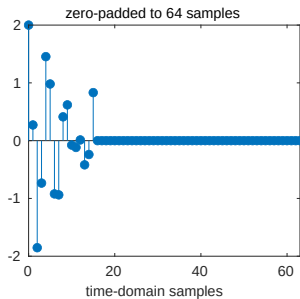
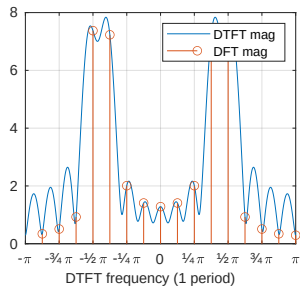
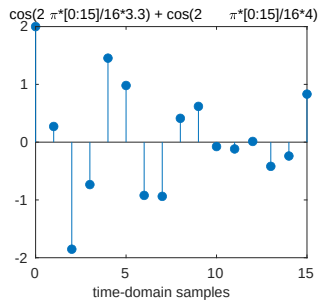
IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Window functions

Padding

Digital filters

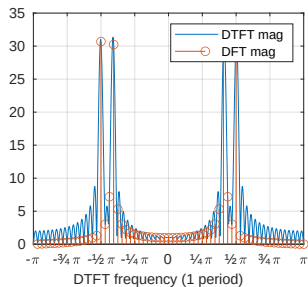
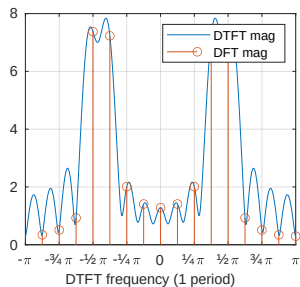
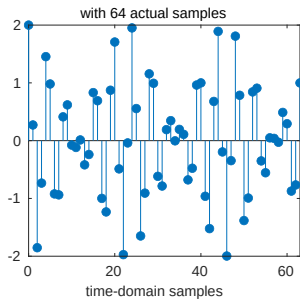
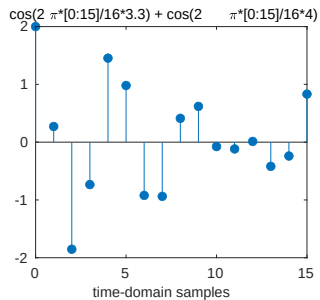
IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Window functions

Padding

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

Applying the discrete Fourier transform (DFT) to an  $n$ -element long real-valued sequence samples the DTFT of that sequence at  $n/2 + 1$  discrete frequencies.

The DTFT spectrum has already been distorted by multiplying the (hypothetically longer) signal with a windowing function that limits its length to  $n$  non-zero values and forces the waveform down to zero outside the window. Therefore, appending further zeros outside the window will not affect the DTFT.

The frequency resolution of the DFT is the sampling frequency divided by the block size of the DFT. Zero padding can therefore be used to increase the frequency resolution of the DFT, to sample the DTFT at more places. But that does not change the limit imposed on the frequency resolution (i.e., blurriness) of the DTFT by the length of the window.

Note that zero padding does *not* add any additional information to the signal. The DTFT has already been “low-pass filtered” by being convolved with the spectrum of the windowing function. Zero padding in the time domain merely causes the DFT to sample the same underlying DTFT spectrum at a higher resolution, thereby making it easier to visually distinguish spectral lines and to locate their peak more precisely.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Window functions

**Padding**

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

- ① Sequences and systems
- ② Convolution
- ③ Fourier transform
- ④ Sampling
- ⑤ Discrete Fourier transform
- ⑥ Deconvolution
- ⑦ Spectral estimation
- ⑧ **Digital filters**
  - FIR filters
- ⑨ IIR filters
- ⑩ Random signals
- ⑪ Digital communication
- ⑫ Audiovisual data compression

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

**Digital filters**

FIR filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

Filter: suppresses (removes, attenuates) unwanted signal components.

- ▶ **low-pass filter** – suppress all frequencies above a cut-off frequency
- ▶ **high-pass filter** – suppress all frequencies below a cut-off frequency, including DC (direct current = 0 Hz)
- ▶ **band-pass filter** – suppress signals outside a frequency interval (= passband)
- ▶ **band-stop filter** (aka: band-reject filter) – suppress signals inside a single frequency interval (= stopband)
- ▶ **notch filter** – narrow band-stop filter, ideally suppressing only a single frequency

The term “filter” is sometimes extended to other LTI systems, e.g.

- ▶ **all-pass filter** – maintains amplitude for all frequencies, but modifies phase
- ▶ **comb filter** – adds an echo to create frequency-dependent interference

For digital filters, we also distinguish

- ▶ **finite impulse response (FIR)** filters
- ▶ **infinite impulse response (IIR)** filters

depending on how far their memory reaches back in time.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

**Digital filters**

FIR filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

Recall that the ideal continuous low-pass filter with cut-off frequency  $f_c$  has the frequency characteristic

$$H(f) = \begin{cases} 1 & \text{if } |f| < f_c \\ 0 & \text{if } |f| > f_c \end{cases} = \text{rect}\left(\frac{f}{2f_c}\right)$$

and the impulse response

$$h(t) = 2f_c \frac{\sin 2\pi t f_c}{2\pi t f_c} = 2f_c \cdot \text{sinc}(2f_c \cdot t).$$

Sampling this impulse response with the sampling frequency  $f_s$  of the signal to be processed will lead to a periodic frequency characteristic, that matches the periodic spectrum of the sampled signal.

There are two problems though:

- ▶ the impulse response is infinitely long
- ▶ this filter is not causal, that is  $h(t) \neq 0$  for  $t < 0$

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

**FIR filters**

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

Solutions:

- ▶ Make the impulse response finite by multiplying the sampled  $h(t)$  with a windowing function
- ▶ Make the impulse response causal by adding a delay of half the window size

The impulse response of an  $n$ -th order low-pass filter is then chosen as

$$h_i = 2f_c/f_s \cdot \frac{\sin[2\pi(i - n/2)f_c/f_s]}{2\pi(i - n/2)f_c/f_s} \cdot w_i$$

where  $\{w_i\}$  is a windowing sequence, such as the Hamming window

$$w_i = 0.54 - 0.46 \times \cos(2\pi i/n)$$

with  $w_i = 0$  for  $i < 0$  and  $i > n$ , that is  $n + 1$  coefficients or “taps”.

Note that for  $f_c = f_s/4$ , we have  $h_i = 0$  wherever  $i - n/2$  is even and non-zero, that is for every second coefficient outside the centre. Therefore, this special case requires only half the number of multiplications during the convolution. Such “half-band” FIR filters are used, for example, as anti-aliasing filters wherever a sampling rate needs to be halved.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

**FIR filters**

IIR filters

Random signals

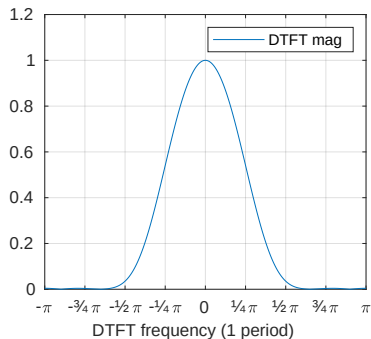
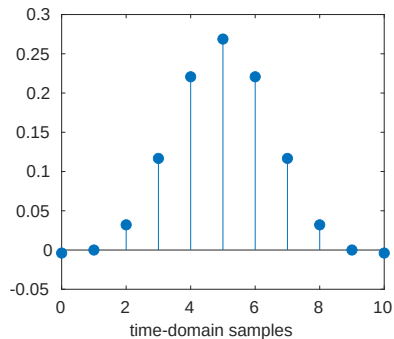
Digital communication

Audiovisual data compression

Outlook

# FIR low-pass filter design examples

order  $n = 10$



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

**FIR filters**

IIR filters

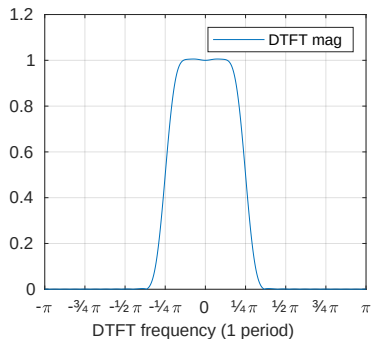
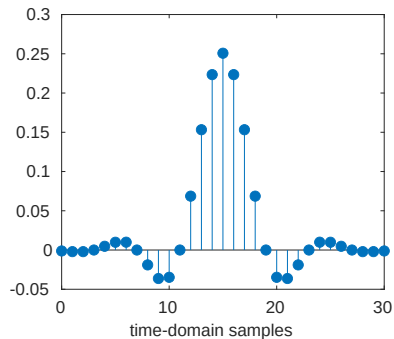
Random signals

Digital communication

Audiovisual data compression

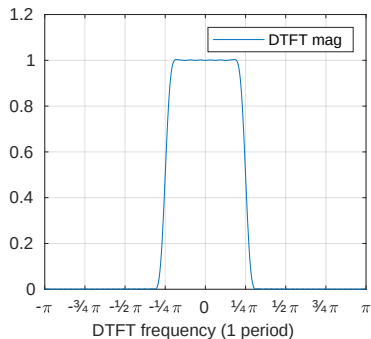
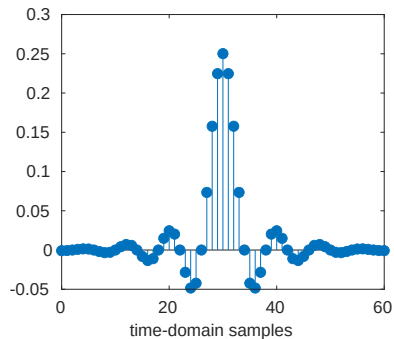
Outlook

order  $n = 30$

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)**[FIR filters](#)**[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

# FIR low-pass filter design examples

order  $n = 60$



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

**FIR filters**

IIR filters

Random signals

Digital communication

Audiovisual data compression

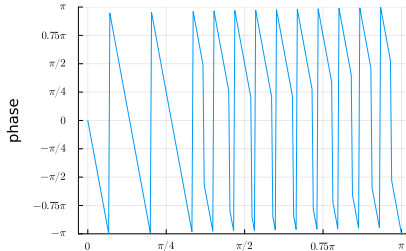
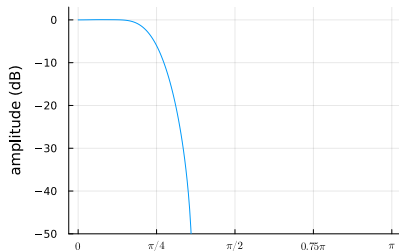
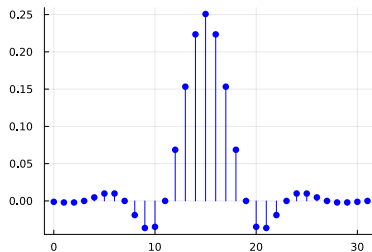
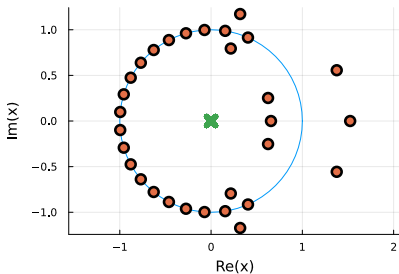
Outlook

# FIR low-pass filter design example (DSP.jl)

order:  $n = 30$ , cut-off frequency ( $-6$  dB):  $f_c = 0.25 \times f_s/2$ , window: Hamming

using DSP; b = digitalfilter(Lowpass(0.25), FIRWindow(hamming(31)))

f = convert(ZeroPoleGain, PolynomialRatio(b, [1])); H, w = freqresp(f)

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[FIR filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

An ideal filter has a gain of 1 in the pass-band and a gain of 0 in the stop band, and nothing in between.

A practical filter will have

- ▶ frequency-dependent gain near 1 in the passband
- ▶ frequency-dependent gain below a threshold in the stopband
- ▶ a transition band between the pass and stop bands

We truncate the ideal, infinitely-long impulse response by multiplication with a window sequence.

In the frequency domain, this will convolve the rectangular frequency response of the ideal low-pass filter with the frequency characteristic of the window.

The width of the main lobe determines the width of the transition band, and the side lobes cause ripples in the passband and stopband.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

**FIR filters**

IIR filters

Random signals

Digital communication

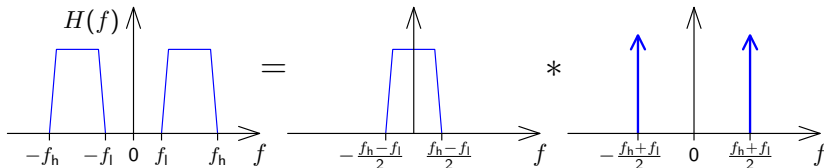
Audiovisual data compression

Outlook

# Low-pass to band-pass filter conversion (modulation)

To obtain a band-pass filter that attenuates all frequencies  $f$  outside the range  $f_l < f < f_h$ , we first design a low-pass filter with a cut-off frequency  $(f_h - f_l)/2$ . We then multiply its impulse response with a sine wave of frequency  $(f_h + f_l)/2$ , effectively amplitude modulating it, to shift its centre frequency. Finally, we apply a window function:

$$h_i = (f_h - f_l)/f_s \cdot \frac{\sin[\pi(i - n/2)(f_h - f_l)/f_s]}{\pi(i - n/2)(f_h - f_l)/f_s} \cdot \cos[\pi i(f_h + f_l)/f_s] \cdot w_i$$



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

FIR filters

IIR filters

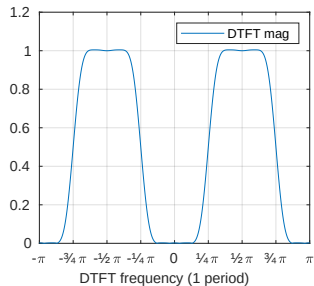
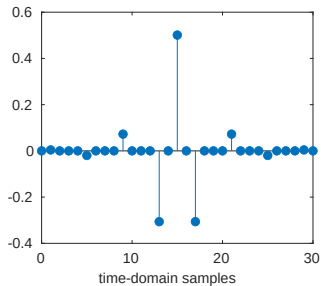
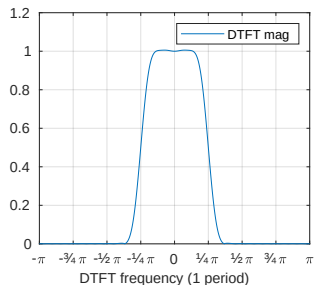
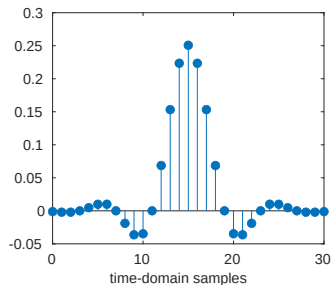
Random signals

Digital communication

Audiovisual data compression

Outlook

# Band-pass filter example (modulation)



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

**FIR filters**

IIR filters

Random signals

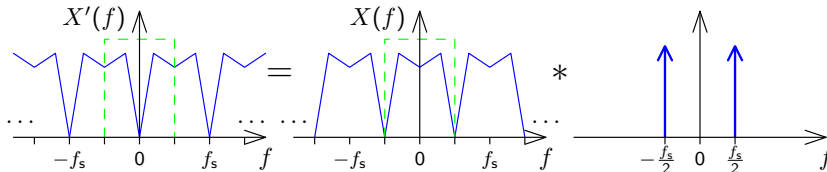
Digital communication

Audiovisual data compression

Outlook

# Low-pass to high-pass filter conversion (freq. inversion)

In order to turn the spectrum  $X(f)$  of a real-valued signal  $x_i$  sampled at  $f_s$  into an inverted spectrum  $X'(f) = [X(f_s/2 - f)]^* = X(f \pm f_s/2)$ , we merely have to shift the periodic spectrum by  $f_s/2$ :



This can be accomplished by multiplying the sampled sequence  $x_i$  with  $y_i = \cos \pi f_s t = \cos \pi i = e^{j\pi i}$ , which is nothing but multiplication with the sequence

$$\dots, 1, -1, 1, -1, 1, -1, 1, -1, \dots$$

So in order to design a discrete high-pass filter that attenuates all frequencies  $f$  outside the range  $f_c < |f| < f_s/2$ , we merely have to design a low-pass filter that attenuates all frequencies outside the range  $-f_c < f < f_c$ , and then multiply every second value of its impulse response with  $-1$ .

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

**FIR filters**

IIR filters

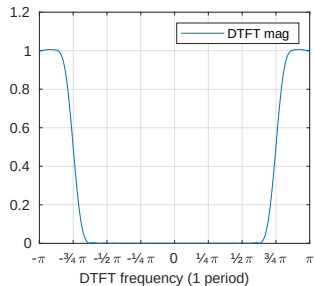
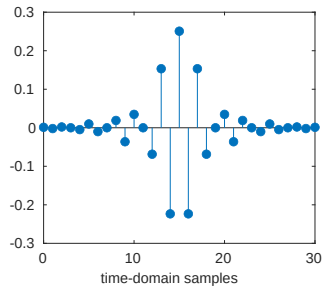
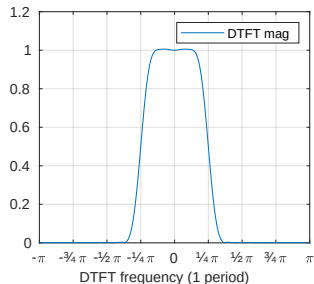
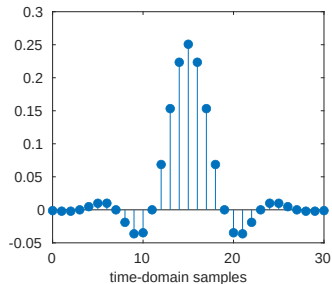
Random signals

Digital communication

Audiovisual data compression

Outlook

# High-pass filter example (freq. inversion)



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

**FIR filters**

IIR filters

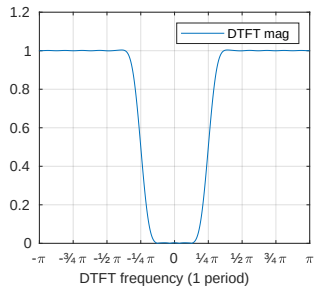
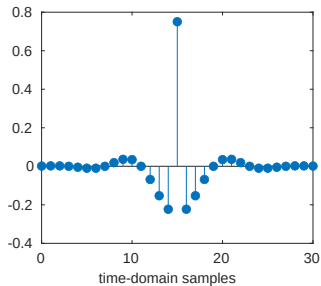
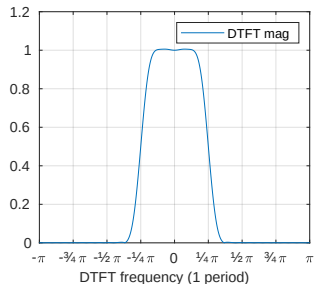
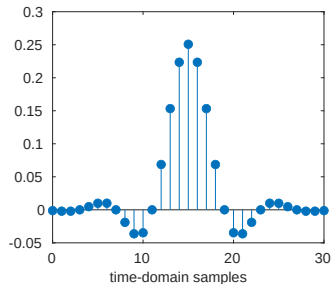
Random signals

Digital communication

Audiovisual data compression

Outlook

# High-pass filter example (subtract from impulse seq.)



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

**FIR filters**

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

A filter where the Fourier transform  $H(f)$  of its impulse response  $h(t)$  is real-valued will not affect the phase of the filtered signal at any frequency. Only the amplitudes will be affected.

$$\forall f \in \mathbb{R} : H(f) \in \mathbb{R} \iff \forall t \in \mathbb{R} : h(t) = [h(-t)]^*$$

A phase-neutral filter with a real-valued frequency response will have an even impulse response, and will therefore usually be non-causal.

To make such a filter causal, we have to add a delay  $\Delta t$  (half the length of the impulse response). This corresponds to multiplication with  $e^{-2\pi j f \Delta t}$  in the frequency domain:

$$h(t - \Delta t) \longleftrightarrow H(f) \cdot e^{-2\pi j f \Delta t}$$

Filters that delay the phase of a signal at each frequency by the time  $\Delta t$  therefore add to the phase angle a value  $-2\pi f \Delta t$ , which increases linearly with  $f$ . They are therefore called *linear-phase filters*.

This is the closest one can get to phase-neutrality with causality.

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[FIR filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

- ① Sequences and systems
- ② Convolution
- ③ Fourier transform
- ④ Sampling
- ⑤ Discrete Fourier transform
- ⑥ Deconvolution
- ⑦ Spectral estimation
- ⑧ Digital filters
- ⑨ IIR filters**
  - z-transform
  - Filter design
- ⑩ Random signals
- ⑪ Digital communication
- ⑫ Audiovisual data compression

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

**IIR filters**

z-transform

Filter design

Random signals

Digital communication

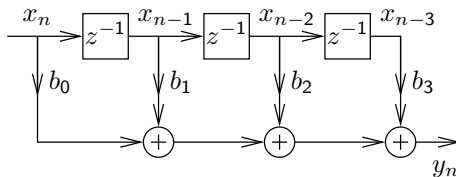
Audiovisual data compression

Outlook

# Finite impulse response (FIR) filter

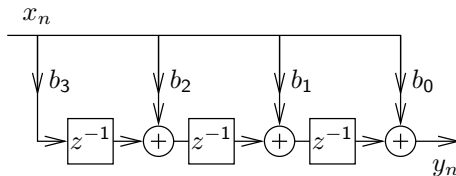
$$y_n = \sum_{m=0}^M b_m \cdot x_{n-m}$$

$M = 3$ :



(see slide 25)

Transposed implementation:



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

**IIR filters**

z-transform

Filter design

Random signals

Digital communication

Audiovisual data compression

Outlook

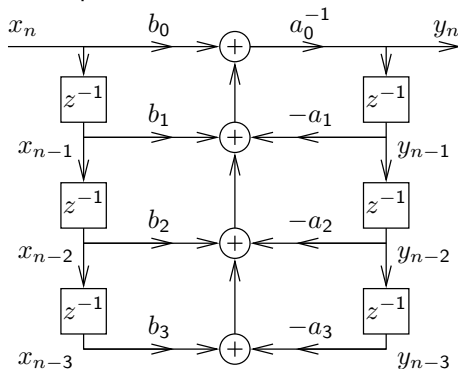
# Infinite impulse response (IIR) filter

$$\sum_{k=0}^N a_k \cdot y_{n-k} = \sum_{m=0}^M b_m \cdot x_{n-m}$$

Usually normalize:  $a_0 = 1$

$$y_n = \left( \sum_{m=0}^M b_m \cdot x_{n-m} - \sum_{k=1}^N a_k \cdot y_{n-k} \right) / a_0$$

Direct form I implementation:



$$\max\{M, N\} = \text{"filter order"}$$

with

$$M + N + 1 \text{ coefficients}$$

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

**IIR filters**

z-transform

Filter design

Random signals

Digital communication

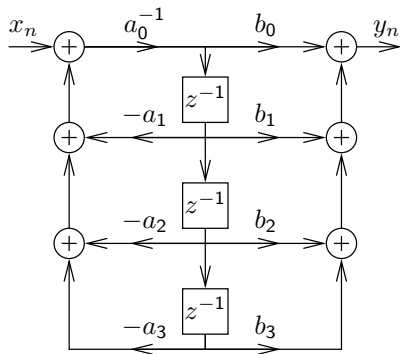
Audiovisual data compression

Outlook

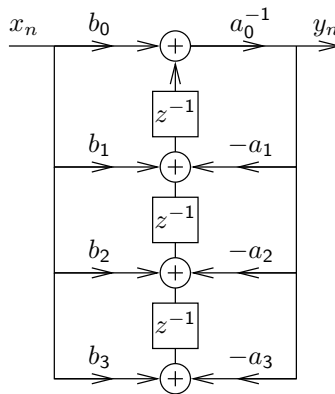
# Infinite impulse response (IIR) filter – direct form II

$$y_n = \left( \sum_{m=0}^M b_m \cdot x_{n-m} - \sum_{k=1}^N a_k \cdot y_{n-k} \right) / a_0$$

Direct form II:



Transposed direct form II:



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

**IIR filters**

z-transform

Filter design

Random signals

Digital communication

Audiovisual data compression

Outlook

We can represent sequences  $\{x_n\}$  as polynomials:

$$X(v) = \sum_{n=-\infty}^{\infty} x_n v^n$$

Example of polynomial multiplication:

$$\begin{array}{r} (1 + 2v + 3v^2) \cdot (2 + 1v) \\ \hline 2 + 4v + 6v^2 \\ + \quad \quad 1v + 2v^2 + 3v^3 \\ \hline = 2 + 5v + 8v^2 + 3v^3 \end{array}$$

Compare this with the convolution of two sequences (in Julia):

```
conv([1 2 3], [2 1]) == [2 5 8 3]
```

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[z-transform](#)[Filter design](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

Convolution of sequences is equivalent to polynomial multiplication:

$$\begin{aligned}
 \{h_n\} * \{x_n\} &= \{y_n\} \Rightarrow y_n = \sum_{k=-\infty}^{\infty} h_k \cdot x_{n-k} \\
 \downarrow \quad \downarrow & \\
 H(v) \cdot X(v) &= \left( \sum_{n=-\infty}^{\infty} h_n v^n \right) \cdot \left( \sum_{n=-\infty}^{\infty} x_n v^n \right) \\
 &= \sum_{n=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} h_k \cdot x_{n-k} \cdot v^n
 \end{aligned}$$

Note how the Fourier transform of a sequence can be accessed easily from its polynomial form:

$$X(e^{-j\omega}) = \sum_{n=-\infty}^{\infty} x_n e^{-j\omega n}$$

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

z-transform

Filter design

Random signals

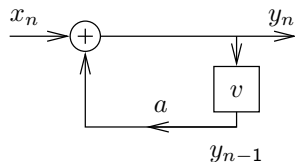
Digital communication

Audiovisual data compression

Outlook

Example of polynomial division:

$$\frac{1}{1 - av} = 1 + av + a^2v^2 + a^3v^3 + \dots = \sum_{n=0}^{\infty} a^n v^n$$



$$\begin{array}{r}
 1 - av \overline{) \begin{array}{l} 1 + av + a^2v^2 + \dots \\ 1 - av \\ \hline av - a^2v^2 \\ \hline a^2v^2 - a^3v^3 \\ \hline \dots \end{array} }
 \end{array}$$

Rational functions (quotients of two polynomials) can provide a convenient closed-form representation for infinitely-long exponential sequences, in particular the impulse responses of IIR filters.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

z-transform

Filter design

Random signals

Digital communication

Audiovisual data compression

Outlook

The  $z$ -transform of a sequence  $\{x_n\}$  is defined as:

$$X(z) = \sum_{n=-\infty}^{\infty} x_n z^{-n}$$

Note that this differs only in the sign of the exponent from the polynomial representation discussed on the preceding slides.

Recall that the above  $X(z)$  is exactly the factor with which an exponential sequence  $\{z^n\}$  is multiplied, if it is convolved with  $\{x_n\}$ :

$$\{z^n\} * \{x_n\} = \{y_n\}$$

$$\Rightarrow y_n = \sum_{k=-\infty}^{\infty} z^{n-k} x_k = z^n \cdot \sum_{k=-\infty}^{\infty} z^{-k} x_k = z^n \cdot X(z)$$

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[z-transform](#)[Filter design](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

The  $z$ -transform defines for each sequence a continuous complex-valued surface over the complex plane  $\mathbb{C}$ .

For finite sequences, its value is defined across the entire complex plane (except possibly at  $z = 0$  or  $|z| = \infty$ ).

For infinite sequences, it can be shown that the  $z$ -transform converges only for the region

$$\lim_{n \rightarrow \infty} \left| \frac{x_{n+1}}{x_n} \right| < |z| < \lim_{n \rightarrow -\infty} \left| \frac{x_{n+1}}{x_n} \right|$$

The  $z$ -transform identifies a sequence unambiguously only in conjunction with a given *region of convergence*. In other words, there exist different sequences, that have the same expression as their  $z$ -transform, but that converge for different amplitudes of  $z$ .

The  $z$ -transform is a generalization of the discrete-time Fourier transform, which it contains on the complex unit circle ( $|z| = 1$ ):

$$t_s^{-1} \cdot \mathcal{F}\{\hat{x}(t)\}(f) = X(e^{j\dot{\omega}}) = \sum_{n=-\infty}^{\infty} x_n e^{-j\dot{\omega}n}$$

where  $\dot{\omega} = 2\pi \frac{f}{f_s}$ .

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

z-transform

Filter design

Random signals

Digital communication

Audiovisual data compression

Outlook

# Properties of the $z$ -transform

If  $X(z)$  is the  $z$ -transform of  $\{x_n\}$ , we write here  $\{x_n\} \bullet\!\!\!\circ X(z)$ .

If  $\{x_n\} \bullet\!\!\!\circ X(z)$  and  $\{y_n\} \bullet\!\!\!\circ Y(z)$ , then:

## Linearity:

$$\{ax_n + by_n\} \bullet\!\!\!\circ aX(z) + bY(z)$$

## Convolution:

$$\{x_n\} * \{y_n\} \bullet\!\!\!\circ X(z) \cdot Y(z)$$

## Time shift:

$$\{x_{n+k}\} \bullet\!\!\!\circ z^k X(z)$$

Remember in particular: delaying by one sample is multiplication with  $z^{-1}$ .

[Introduction](#)
[Sequences and systems](#)
[Convolution](#)
[Fourier transform](#)
[Sampling](#)
[Discrete Fourier transform](#)
[Deconvolution](#)
[Spectral estimation](#)
[Digital filters](#)
[IIR filters](#)
[z-transform](#)
[Filter design](#)
[Random signals](#)
[Digital communication](#)
[Audiovisual data compression](#)
[Outlook](#)

**Time reversal:**

$$\{x_{-n}\} \bullet \circ X(z^{-1})$$

**Multiplication with exponential:**

$$\{a^{-n}x_n\} \bullet \circ X(az)$$

**Complex conjugate:**

$$\{x_n^*\} \bullet \circ X^*(z^*)$$

**Real/imaginary value:**

$$\{\Re\{x_n\}\} \bullet \circ \frac{1}{2}(X(z) + X^*(z^*))$$

$$\{\Im\{x_n\}\} \bullet \circ \frac{1}{2j}(X(z) - X^*(z^*))$$

**Initial value:**

$$x_0 = \lim_{z \rightarrow \infty} X(z) \quad \text{if } x_n = 0 \text{ for all } n < 0$$

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

z-transform

Filter design

Random signals

Digital communication

Audiovisual data compression

Outlook

## Some example sequences and their $z$ -transforms:

| $x_n$                                 | $X(z)$                                                                           |
|---------------------------------------|----------------------------------------------------------------------------------|
| $\delta_n$                            | 1                                                                                |
| $u_n$                                 | $\frac{z}{z-1} = \frac{1}{1-z^{-1}}$                                             |
| $a^n u_n$                             | $\frac{z}{z-a} = \frac{1}{1-az^{-1}}$                                            |
| $nu_n$                                | $\frac{z}{(z-1)^2}$                                                              |
| $n^2 u_n$                             | $\frac{z(z+1)}{(z-1)^3}$                                                         |
| $e^{an} u_n$                          | $\frac{z}{z-e^a}$                                                                |
| $\binom{n-1}{k-1} e^{a(n-k)} u_{n-k}$ | $\frac{1}{(z-e^a)^k}$                                                            |
| $\sin(\omega n + \varphi) u_n$        | $\frac{z^2 \sin(\varphi) + z \sin(\omega - \varphi)}{z^2 - 2z \cos(\omega) + 1}$ |

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

**z-transform**

Filter design

Random signals

Digital communication

Audiovisual data compression

Outlook

## Example:

What is the  $z$ -transform of the impulse response  $\{h_n\}$  of the discrete system  $y_n = x_n + ay_{n-1}$ ?

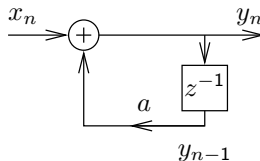
$$y_n = x_n + ay_{n-1}$$

$$Y(z) = X(z) + az^{-1}Y(z)$$

$$Y(z) - az^{-1}Y(z) = X(z)$$

$$Y(z)(1 - az^{-1}) = X(z)$$

$$\frac{Y(z)}{X(z)} = \frac{1}{1 - az^{-1}} = \frac{z}{z - a}$$



Since  $\{y_n\} = \{h_n\} * \{x_n\}$ , we have  $Y(z) = H(z) \cdot X(z)$  and therefore

$$H(z) = \frac{Y(z)}{X(z)} = \frac{z}{z - a} = 1 + az^{-1} + a^2z^{-2} + \dots$$

where polynomial long division returns the causal impulse response

$$h_0 = 1, h_1 = a, h_2 = a^2, \dots, h_n = a^n \text{ for all } n \geq 0$$

We have applied here the linearity of the  $z$ -transform, and its time-shift and convolution properties.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

z-transform

Filter design

Random signals

Digital communication

Audiovisual data compression

Outlook

# $z$ -transform of recursive filter structures

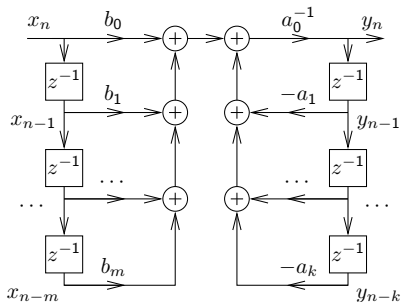
Consider the discrete system defined by

$$\sum_{l=0}^k a_l \cdot y_{n-l} = \sum_{l=0}^m b_l \cdot x_{n-l}$$

or equivalently

$$a_0 y_n + \sum_{l=1}^k a_l \cdot y_{n-l} = \sum_{l=0}^m b_l \cdot x_{n-l}$$

$$y_n = a_0^{-1} \cdot \left( \sum_{l=0}^m b_l \cdot x_{n-l} - \sum_{l=1}^k a_l \cdot y_{n-l} \right)$$



What is the  $z$ -transform  $H(z)$  of its impulse response  $\{h_n\}$ , where  $\{y_n\} = \{h_n\} * \{x_n\}$ ?

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

**$z$ -transform**

Filter design

Random signals

Digital communication

Audiovisual data compression

Outlook

Using the linearity and time-shift property of the  $z$ -transform:

$$\sum_{l=0}^k a_l \cdot y_{n-l} = \sum_{l=0}^m b_l \cdot x_{n-l}$$

$$\sum_{l=0}^k a_l z^{-l} \cdot Y(z) = \sum_{l=0}^m b_l z^{-l} \cdot X(z)$$

$$Y(z) \sum_{l=0}^k a_l z^{-l} = X(z) \sum_{l=0}^m b_l z^{-l}$$

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{l=0}^m b_l z^{-l}}{\sum_{l=0}^k a_l z^{-l}}$$

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \dots + b_m z^{-m}}{a_0 + a_1 z^{-1} + a_2 z^{-2} + \dots + a_k z^{-k}}$$

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[z-transform](#)[Filter design](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

The  $z$ -transform of the impulse response  $\{h_n\}$  of the causal LTI system defined by

$$\sum_{l=0}^k a_l \cdot y_{n-l} = \sum_{l=0}^m b_l \cdot x_{n-l}$$

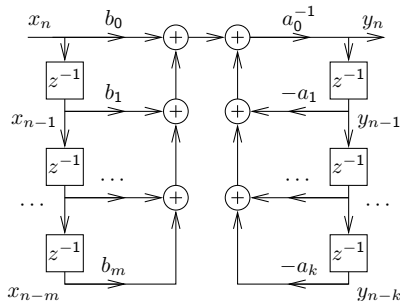
with  $\{y_n\} = \{h_n\} * \{x_n\}$  is the rational function

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \dots + b_m z^{-m}}{a_0 + a_1 z^{-1} + a_2 z^{-2} + \dots + a_k z^{-k}}$$

( $b_m \neq 0$ ,  $a_k \neq 0$ ) which can also be written as

$$H(z) = \frac{z^k \sum_{l=0}^m b_l z^{m-l}}{z^m \sum_{l=0}^k a_l z^{k-l}} = \frac{z^k}{z^m} \cdot \frac{b_0 z^m + b_1 z^{m-1} + b_2 z^{m-2} + \dots + b_m}{a_0 z^k + a_1 z^{k-1} + a_2 z^{k-2} + \dots + a_k}.$$

$H(z)$  has  $m$  zeros and  $k$  poles at non-zero locations in the  $z$  plane, plus  $k - m$  zeros (if  $k > m$ ) or  $m - k$  poles (if  $m > k$ ) at  $z = 0$ .



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

z-transform

Filter design

Random signals

Digital communication

Audiovisual data compression

Outlook

This function can be converted into the form

$$H(z) = \frac{b_0}{a_0} \cdot \frac{\prod_{l=1}^m (1 - c_l \cdot z^{-1})}{\prod_{l=1}^k (1 - d_l \cdot z^{-1})} = \frac{b_0}{a_0} \cdot z^{k-m} \cdot \frac{\prod_{l=1}^m (z - c_l)}{\prod_{l=1}^k (z - d_l)}$$

where the  $c_l$  are the non-zero positions of zeros ( $H(c_l) = 0$ ) and the  $d_l$  are the non-zero positions of the poles (i.e.,  $z \rightarrow d_l \Rightarrow |H(z)| \rightarrow \infty$ ) of  $H(z)$ . Except for a constant factor,  $H(z)$  is entirely characterized by the position of these zeros and poles.

On the unit circle  $z = e^{j\omega}$ ,  $H(e^{j\omega})$  is the discrete-time Fourier transform of  $\{h_n\}$  ( $\omega = \pi f / \frac{f_s}{2}$ ). The DTFT amplitude can also be expressed in terms of the relative position of  $e^{j\omega}$  to the zeros and poles:

$$|H(e^{j\omega})| = \left| \frac{b_0}{a_0} \right| \cdot \frac{\prod_{l=1}^m |e^{j\omega} - c_l|}{\prod_{l=1}^k |e^{j\omega} - d_l|}$$

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

z-transform

Filter design

Random signals

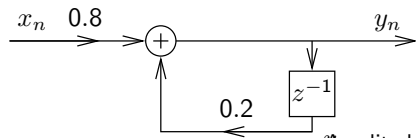
Digital communication

Audiovisual data compression

Outlook

# Example: a single-pole filter

Consider this IIR filter:



Its  $z$ -transform

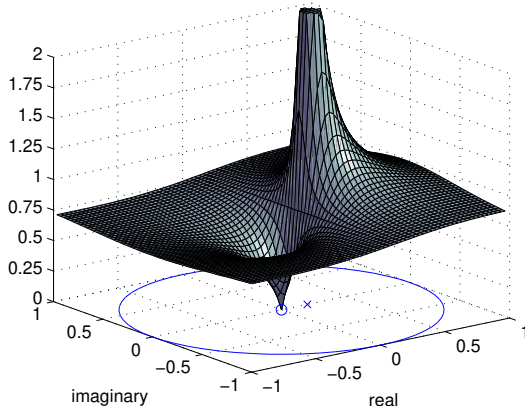
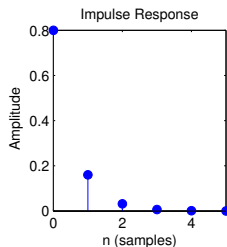
$$H(z) = \frac{0.8}{1 - 0.2 \cdot z^{-1}} = \frac{0.8z}{z - 0.2}$$

has one *pole* at  $z = d_1 = 0.2$  and one *zero* at  $z = 0$ .

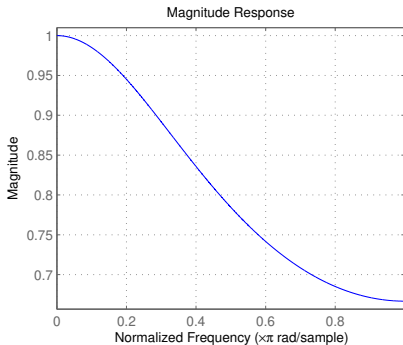
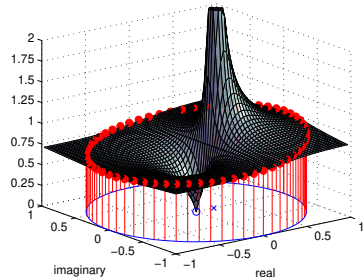
Amplitude  $|H(z)|$ :

$$a_0 = 1, a_1 = -0.2, \\ b_0 = 0.8$$

$$x_n = \delta_n \Rightarrow y_n =$$

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[z-transform](#)[Filter design](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

$$H(z) = \frac{0.8}{1-0.2 \cdot z^{-1}} = \frac{0.8z}{z-0.2} \text{ (cont'd)}$$



Run this LTI filter at sampling frequency  $f_s$  and test it with sinusoidal input (frequency  $f$ , amplitude 1):  $x_n = \cos(2\pi f n / f_s)$

Output:  $y_n = A(f) \cdot \cos(2\pi f n / f_s + \theta(f))$

What are the *gain*  $A(f)$  and *phase delay*  $\theta(f)$  at frequency  $f$ ?

Answer:

$$A(f) = |H(e^{j2\pi f / f_s})|$$

$$\theta(f) = \angle H(e^{j2\pi f / f_s}) = \tan^{-1} \frac{\Im\{H(e^{j2\pi f / f_s})\}}{\Re\{H(e^{j2\pi f / f_s})\}} + k\pi$$

angle  
atan2  
unwrap

Example:  $f_s = 8 \text{ kHz}$ ,  $f = 2 \text{ kHz}$  (normalized frequency  $f / f_s = 0.5$ )  $\Rightarrow$  Gain  $A(2 \text{ kHz}) =$

$$|H(e^{j\pi/2})| = |H(j)| = \left| \frac{0.8j}{j-0.2} \right| = \left| \frac{0.8j(-j-0.2)}{(j-0.2)(-j-0.2)} \right| = \left| \frac{0.8-0.16j}{1+0.04} \right| = \sqrt{\frac{0.8^2+0.16^2}{1.04^2}} = 0.784 \dots$$

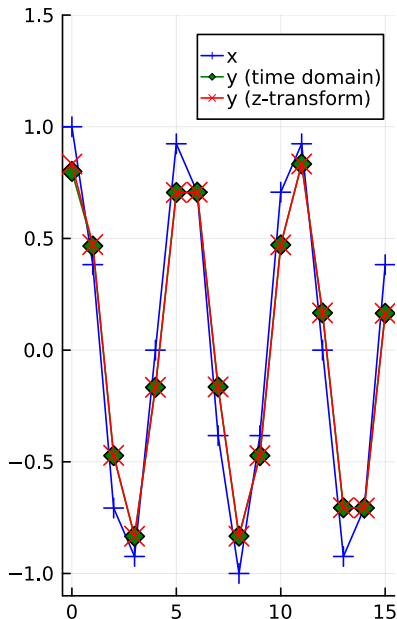
## Visual verification in Julia:

```

n = 0:15; fs = 8000
f = 1500
x = cos.(2pi*f*n/fs)
b = [0.8]; a = [1, -0.2]
y1 = filt(b, a, x)
z = exp(1im*2pi*f/fs)
H = 0.8 * z / (z-0.2)
A = abs(H)
theta = atan(imag(H), real(H))
y2 = A * cos.(2pi*f*n/fs.+theta)

plot(n, [x y1 y2];
      color=[:blue :green :red],
      shape=[:+ :diamond :x],
      msiz=6, mswidth = 4,
      label=["x" "y (time domain)"
            "y (z-transform)"],
      ylim=(-1.1, 1.5),
      size=(250, 400))

```



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

z-transform

Filter design

Random signals

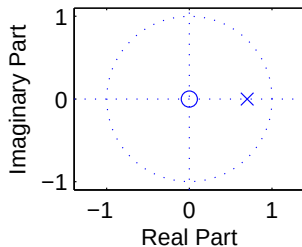
Digital communication

Audiovisual data compression

Outlook

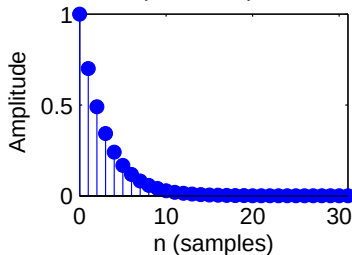
$$H(z) = \frac{z}{z-0.7} = \frac{1}{1-0.7 \cdot z^{-1}}$$

z Plane



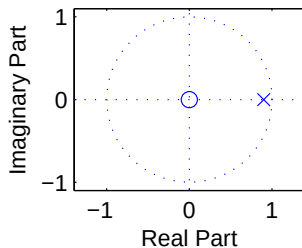
## How do poles affect time domain?

Impulse Response

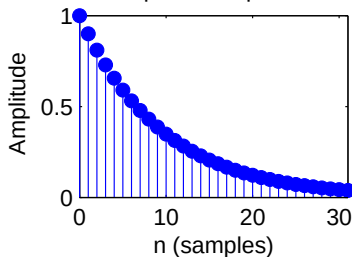


$$H(z) = \frac{z}{z-0.9} = \frac{1}{1-0.9 \cdot z^{-1}}$$

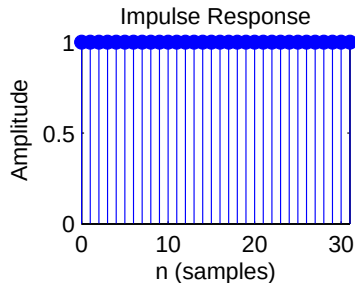
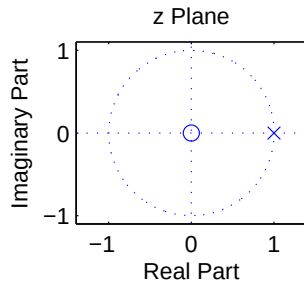
z Plane



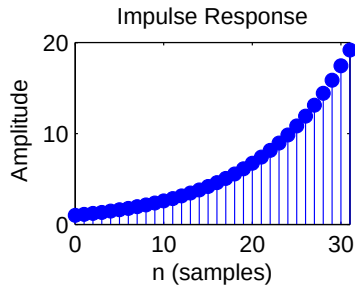
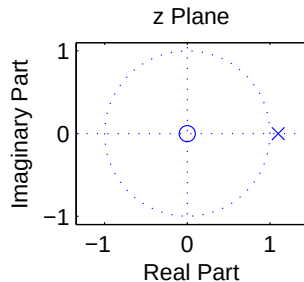
Impulse Response



$$H(z) = \frac{z}{z-1} = \frac{1}{1-z^{-1}}$$



$$H(z) = \frac{z}{z-1.1} = \frac{1}{1-1.1 \cdot z^{-1}}$$



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

z-transform

Filter design

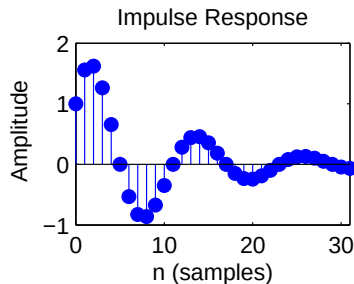
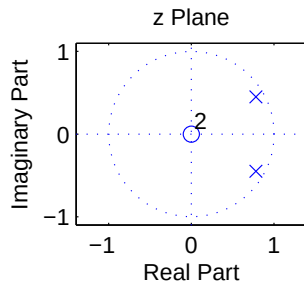
Random signals

Digital communication

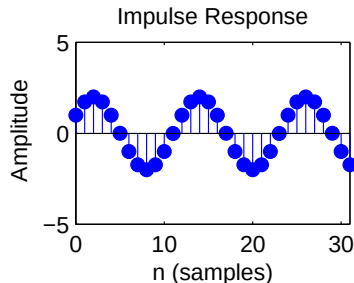
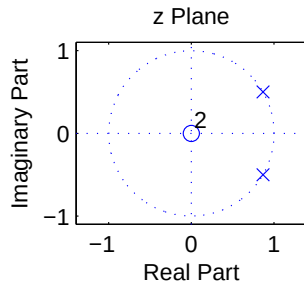
Audiovisual data compression

Outlook

$$H(z) = \frac{z^2}{(z - 0.9 \cdot e^{j\pi/6}) \cdot (z - 0.9 \cdot e^{-j\pi/6})} = \frac{1}{1 - 1.8 \cos(\pi/6)z^{-1} + 0.9^2 \cdot z^{-2}}$$



$$H(z) = \frac{z^2}{(z - e^{j\pi/6}) \cdot (z - e^{-j\pi/6})} = \frac{1}{1 - 2 \cos(\pi/6)z^{-1} + z^{-2}}$$



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

z-transform

Filter design

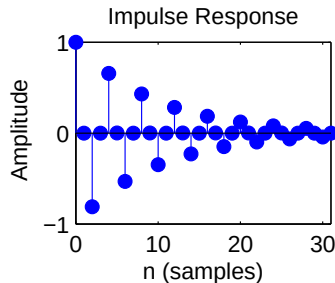
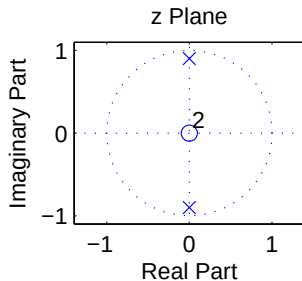
Random signals

Digital communication

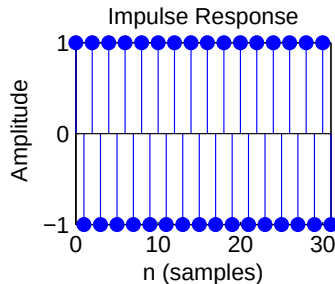
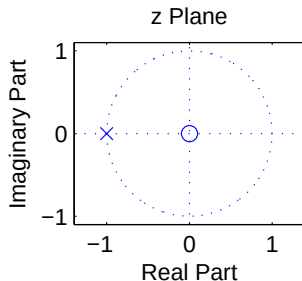
Audiovisual data compression

Outlook

$$H(z) = \frac{z^2}{(z-0.9 \cdot e^{j\pi/2}) \cdot (z-0.9 \cdot e^{-j\pi/2})} = \frac{1}{1-1.8 \cos(\pi/2)z^{-1}+0.9^2 \cdot z^{-2}} = \frac{1}{1+0.9^2 \cdot z^{-2}}$$



$$H(z) = \frac{z}{z+1} = \frac{1}{1+z^{-1}}$$



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

z-transform

Filter design

Random signals

Digital communication

Audiovisual data compression

Outlook

The design of a filter starts with specifying the desired parameters:

- ▶ The *passband* is the frequency range where we want to approximate a gain of one.
- ▶ The *stopband* is the frequency range where we want to approximate a gain of zero.
- ▶ The *order* of a filter is the maximum of the number of zeros or poles it has in the  $z$ -domain, which is the maximum delay (in samples) needed to implement it.
- ▶ Both passband and stopband will in practice not have gains of exactly one and zero, respectively, but may show several deviations from these ideal values, and these *ripples* may have a specified maximum quotient between the highest and lowest gain.
- ▶ There will in practice not be an abrupt change of gain between passband and stopband, but a *transition band* where the frequency response will gradually change from its passband to its stopband value.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

z-transform

Filter design

Random signals

Digital communication

Audiovisual data compression

Outlook

The designer can then trade off conflicting goals such as: small transition band, low order, low ripple amplitude or absence of ripples.

Design techniques for making these tradeoffs for analog filters (involving capacitors, resistors, coils) can also be used to design digital IIR filters:

**Butterworth filters:** Have no ripples, gain falls monotonically across the pass and transition band. Within the passband, the gain drops slowly down to  $\sqrt{1/2}$  ( $-3$  dB). Outside the passband, it drops asymptotically by a factor  $2^N$  per octave ( $N \cdot 20$  dB/decade).

**Chebyshev type I filters:** Distribute the gain error uniformly throughout the passband (equiripples) and drop off monotonically outside.

**Chebyshev type II filters:** Distribute the gain error uniformly throughout the stopband (equiripples) and drop off monotonically in the passband.

**Elliptic filters (Cauer filters):** Distribute the gain error as equiripples both in the passband and stopband. This type of filter is optimal in terms of the combination of the passband-gain tolerance, stopband-gain tolerance, and transition-band width that can be achieved at a given filter order.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

z-transform

Filter design

Random signals

Digital communication

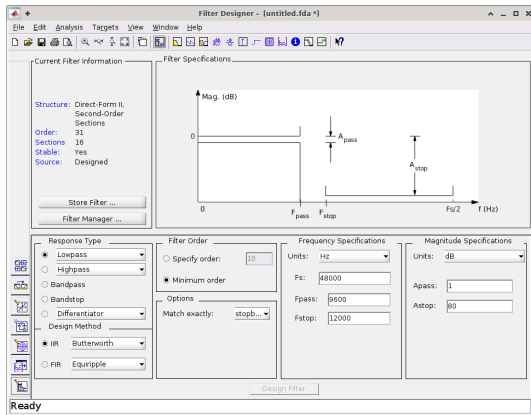
Audiovisual data compression

Outlook

The aforementioned filter-design techniques are implemented in the MATLAB Signal Processing Toolbox in the functions `butter`, `cheby1`, `cheby2`, and `ellip`. They output the coefficients  $a_n$  and  $b_n$  of the difference equation that describes the filter.

These can be applied with `filter` to a sequence, or can be visualized with `zplane` as poles/zeros in the  $z$ -domain, with `impz` as an impulse response, and with `freqz` as an amplitude and phase spectrum.

Call `filterDesigner` for an interactive GUI.



MATLAB Filter Designer

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

z-transform

Filter design

Random signals

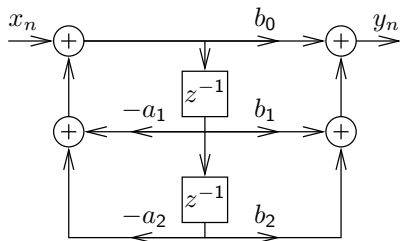
Digital communication

Audiovisual data compression

Outlook

Higher-order IIR filters can be numerically unstable (quantization noise).

A commonly used trick is to split a higher-order IIR filter design into a cascade of  $l$  second-order (biquad) filter sections of the form:



$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

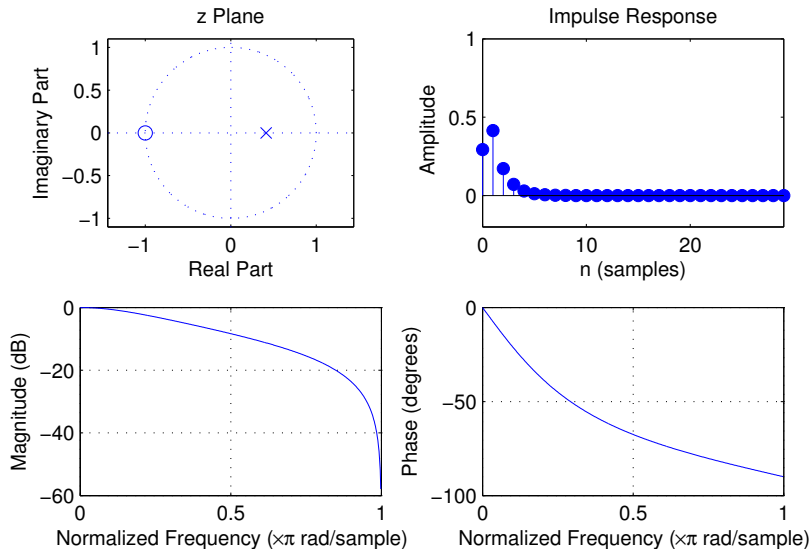
Filter sections  $H_1, H_2, \dots, H_l$  are then applied sequentially to the input sequence, resulting in a filter

$$H(z) = \prod_{k=1}^l H_k(z) = \prod_{k=1}^l \frac{b_{k,0} + b_{k,1} z^{-1} + b_{k,2} z^{-2}}{1 + a_{k,1} z^{-1} + a_{k,2} z^{-2}}$$

Each section implements one pair of poles and one pair of zeros. Jackson's algorithm for pairing poles and zeros into sections: pick the pole pair closest to the unit circle, and place it into a section along with the nearest pair of zeros; repeat until no poles are left.

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[z-transform](#)[Filter design](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

# Butterworth filter design example



order: 1, cut-off frequency ( $-3$  dB):  $0.25 \times f_s/2$

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

z-transform

**Filter design**

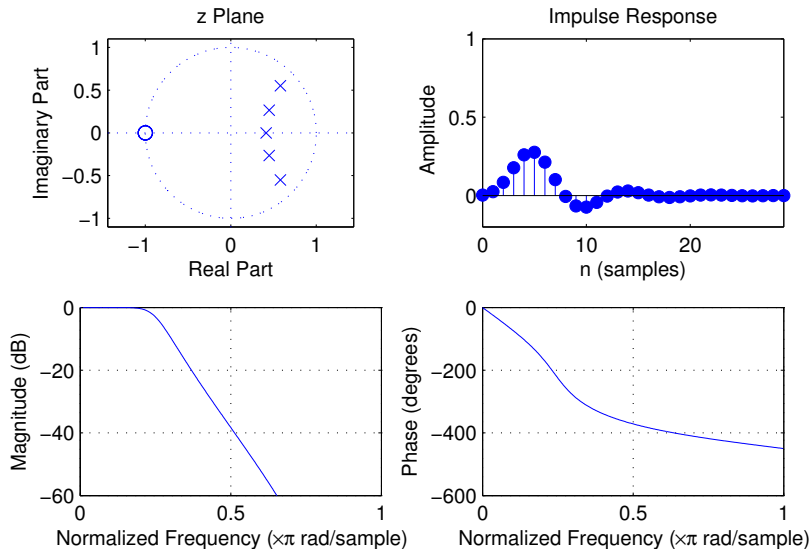
Random signals

Digital communication

Audiovisual data compression

Outlook

# Butterworth filter design example



order: 5, cut-off frequency ( $-3$  dB):  $0.25 \times f_s/2$

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

z-transform

**Filter design**

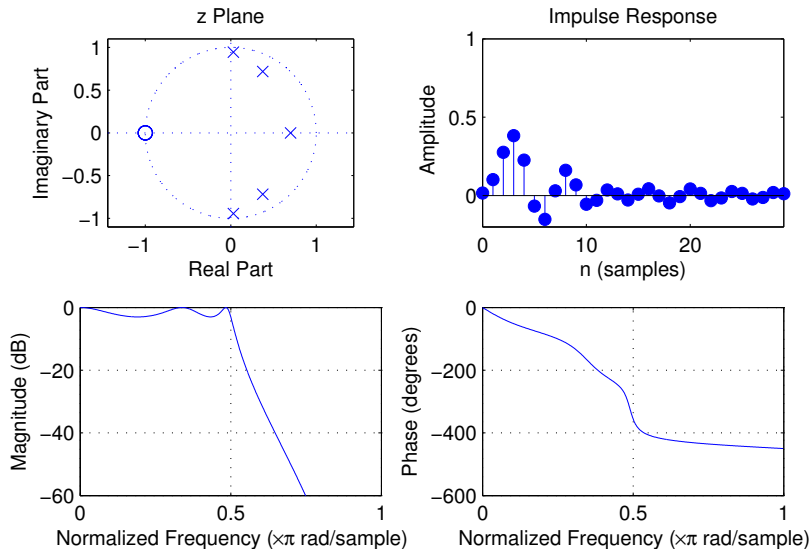
Random signals

Digital communication

Audiovisual data compression

Outlook

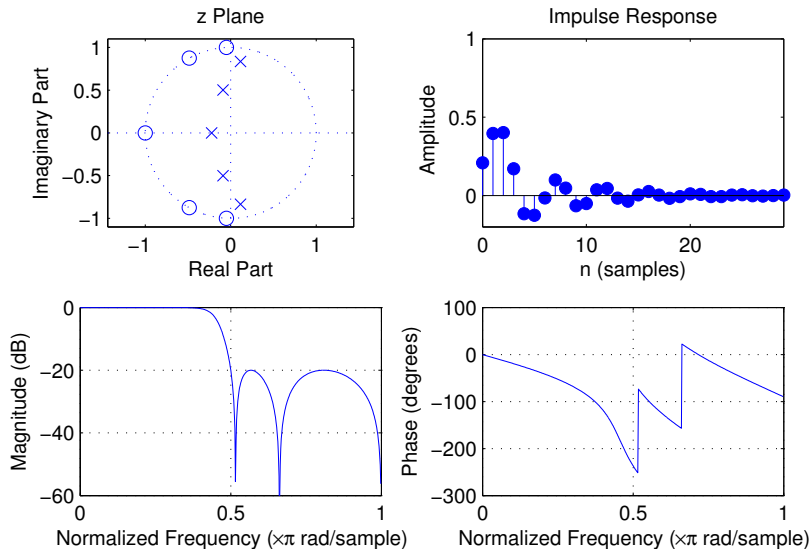
# Chebyshev type I filter design example



order: 5, cut-off frequency:  $0.5 \times f_s/2$ , pass-band ripple:  $-3$  dB

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[z-transform](#)[Filter design](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

# Chebyshev type II filter design example



order: 5, cut-off frequency:  $0.5 \times f_s/2$ , stop-band ripple: -20 dB

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

z-transform

**Filter design**

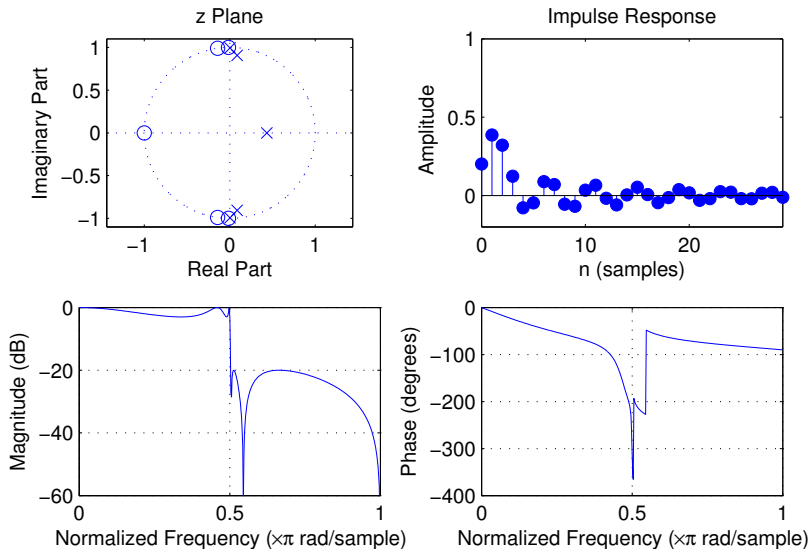
Random signals

Digital communication

Audiovisual data compression

Outlook

# Elliptic filter design example



order: 5, cut-off frequency:  $0.5 \times f_s/2$ , pass-band ripple:  $-3$  dB, stop-band ripple:  $-20$  dB

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

z-transform

**Filter design**

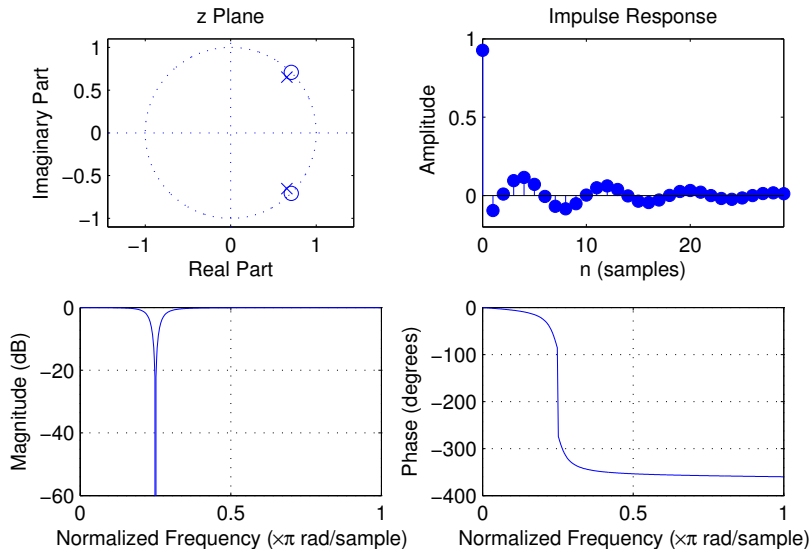
Random signals

Digital communication

Audiovisual data compression

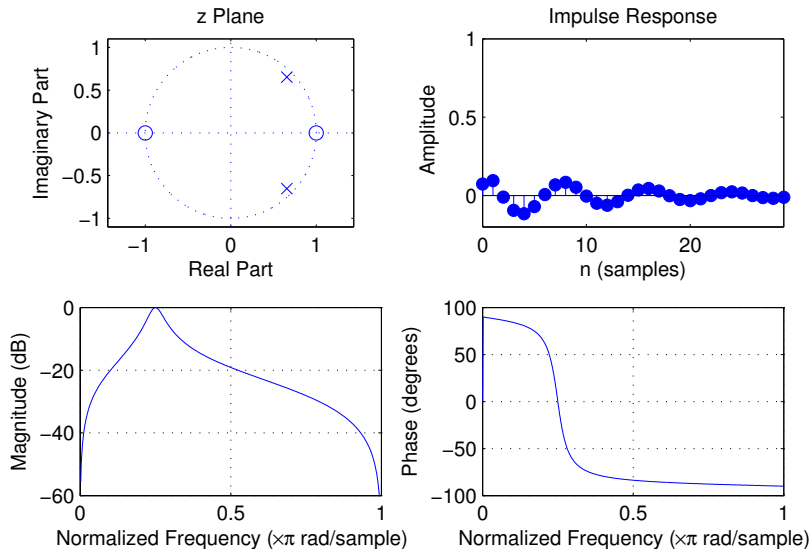
Outlook

# Notch filter design example



order: 2, cut-off frequency:  $0.25 \times f_s/2$ , -3 dB bandwidth:  $0.05 \times f_s/2$

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[z-transform](#)[Filter design](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)



order: 2, cut-off frequency:  $0.25 \times f_s/2$ , -3 dB bandwidth:  $0.05 \times f_s/2$

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[z-transform](#)[Filter design](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

## FIR filters:

- + easy to construct linear-phase filters (symmetric impulse response)
- + numerically stable
  - No poles means: none can get dangerously close to the unit circle.
- higher order, i.e. computationally expensive

## IIR filters:

- + can achieve given transition bands with lower order, i.e. computationally less expensive, as a few multiplications and delays can achieve long impulse responses (slowly decaying oscillations)
- can become numerically unstable (i.e., impulse response not absolutely summable)
- generally not linear phase, and less control over phase behaviour

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

z-transform

Filter design

Random signals

Digital communication

Audiovisual data compression

Outlook

# Zero-phase IIR filtering (`filtfilt`)

In non-real-time applications, where the entire input sequence is available in advance, a simple trick can be used to apply an IIR filter  $H$  without causing any phase change in the filtered signal.

- ① apply the (causal) filter  $H$  normally in forward direction
- ② time-reverse the resulting sequence
- ③ apply the filter  $H$  again (i.e., in backwards direction)
- ④ time-reverse the resulting sequence again

This is equivalent to applying the filter twice, once normally and once with a time-reversed impulse response.

Reversing a real-valued sequence in the time domain corresponds to taking the complex conjugate in the frequency domain.

Resulting filter  $G$  (for  $h_n \in \mathbb{R}$ ):

$$\{g_n\} = \{h_n\} * \{h_{-n}\}$$

$$G(e^{j\omega}) = H(e^{j\omega}) \cdot H(e^{-j\omega}) = H(e^{j\omega}) \cdot H^*(e^{j\omega}) = |H(e^{j\omega})|^2$$

Basic idea in Julia (omitting any optimization, padding, initialization):

```
filtfilt(b, a, x) = reverse(filt(b, a, reverse(filt(b, a, x))))
```

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

z-transform

Filter design

Random signals

Digital communication

Audiovisual data compression

Outlook

- 1 Sequences and systems
- 2 Convolution
- 3 Fourier transform
- 4 Sampling
- 5 Discrete Fourier transform
- 6 Deconvolution
- 7 Spectral estimation
- 8 Digital filters
- 9 IIR filters
- 10 Random signals**
- 11 Digital communication
- 12 Audiovisual data compression

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

**Random signals**

Digital communication

Audiovisual data compression

Outlook

Let  $S$  be the set of all possible outcomes (results) of some experiment. We call  $S$  the *sample space* of that experiment.

A *random variable*  $\mathbf{X}$  is a function

$$\mathbf{X} : S \rightarrow E$$

that assigns to each outcome  $\zeta \in S$  a value  $\mathbf{X}(\zeta) \in E$ , where usually  $E \subseteq \mathbb{R}$  or  $E \subseteq \mathbb{C}$ .

A *random vector*  $\vec{\mathbf{X}}(\zeta) = (\mathbf{x}_1(\zeta), \mathbf{x}_2(\zeta), \dots, \mathbf{x}_n(\zeta))^T$  is a vector of  $n$  random variables, or equivalently a random variable that outputs vectors, e.g.  $\vec{\mathbf{X}}(\zeta) \in \mathbb{R}^n$ .

A *continuous-time random process*  $\mathbf{X} : S \rightarrow E^{\mathbb{R}}$  is a function that maps each experimental outcome  $\zeta \in S$  onto a continuous-time function  $x_\zeta(t)$ , and a *discrete-time random process*  $\mathbf{X} : S \rightarrow E^{\mathbb{Z}}$  maps each outcome  $\zeta$  onto a discrete sequence  $\{x_{\zeta,n}\}_n$ .

The *ensemble* of a random process is the set of all functions (or sequences) from which it picks its output.

In the following, we will usually omit outcome parameter  $\zeta$  from random variables, etc., for notational convenience, and use boldface roman to distinguish random variables from samples.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

A *discrete-time random process* or *random sequence*  $\{\mathbf{x}_n\}$  can also be thought of as a discrete sequence of random variables

$$\dots, \mathbf{x}_{-2}, \mathbf{x}_{-1}, \mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \dots$$

Each time we repeat an experiment, we observe one *realization* or *sample sequence*

$$\dots, x_{-2}, x_{-1}, x_0, x_1, x_2, \dots$$

of that random process. (We cannot observe the outcome  $\zeta$  directly.)

Each individual random variable  $\mathbf{x}_n$  in a random sequence is characterized by its probability distribution function

$$P_{\mathbf{x}_n}(a) = \text{Prob}(\mathbf{x}_n \leq a)$$

and the entire random process is characterized completely by all joint probability distribution functions

$$P_{\mathbf{x}_{n_1}, \dots, \mathbf{x}_{n_k}}(a_1, \dots, a_k) = \text{Prob}(\mathbf{x}_{n_1} \leq a_1 \wedge \dots \wedge \mathbf{x}_{n_k} \leq a_k)$$

for all possible sets  $\{\mathbf{x}_{n_1}, \dots, \mathbf{x}_{n_k}\}$  and all  $k > 0$ .

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

**Random signals**

Digital communication

Audiovisual data compression

Outlook

Two random variables  $\mathbf{x}_n$  and  $\mathbf{x}_m$  are called *independent* if

$$P_{\mathbf{x}_n, \mathbf{x}_m}(a, b) = P_{\mathbf{x}_n}(a) \cdot P_{\mathbf{x}_m}(b)$$

The derivative  $p_{\mathbf{x}_n}(a) = P'_{\mathbf{x}_n}(a)$  is called the *probability density function*.

This helps us to define quantities such as the

- ▶ *expected value*  $E(\mathbf{x}_n) = \int a p_{\mathbf{x}_n}(a) da$
- ▶ *mean-square value* (average power)  $E(|\mathbf{x}_n|^2) = \int |a|^2 p_{\mathbf{x}_n}(a) da$
- ▶ *variance*  $\text{Var}(\mathbf{x}_n) = E[|\mathbf{x}_n - E(\mathbf{x}_n)|^2] = E(|\mathbf{x}_n|^2) - |E(\mathbf{x}_n)|^2$
- ▶ *correlation*  $\text{Cor}(\mathbf{x}_n, \mathbf{x}_m) = E(\mathbf{x}_n \cdot \mathbf{x}_m^*)$
- ▶ *covariance*  $\text{Cov}(\mathbf{x}_n, \mathbf{x}_m) = E[(\mathbf{x}_n - E(\mathbf{x}_n)) \cdot (\mathbf{x}_m - E(\mathbf{x}_m))^*] = E(\mathbf{x}_n \mathbf{x}_m^*) - E(\mathbf{x}_n) E(\mathbf{x}_m)^*$

The expected value  $E(\cdot)$  is a linear operator:  $E(a\mathbf{x}) = aE(\mathbf{x})$  and  $E(\mathbf{x} + \mathbf{y}) = E(\mathbf{x}) + E(\mathbf{y})$ .

Variance is not linear, but  $\text{Var}(a\mathbf{x}) = a^2 \text{Var}(\mathbf{x})$  and, if  $\mathbf{x}$  and  $\mathbf{y}$  are independent,  $\text{Var}(\mathbf{x} + \mathbf{y}) = \text{Var}(\mathbf{x}) + \text{Var}(\mathbf{y})$ .

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

A random sequence is called *strict-sense stationary* if

$$P_{\mathbf{x}_{n_1+l}, \dots, \mathbf{x}_{n_k+l}}(a_1, \dots, a_k) = P_{\mathbf{x}_{n_1}, \dots, \mathbf{x}_{n_k}}(a_1, \dots, a_k)$$

for any shift  $l$  and any number  $k$ , that is if all joint probability distributions are time invariant.

If the above condition holds at least for  $k = 1$ , then the mean

$$E(\mathbf{x}_n) = m_x,$$

and variance

$$E(|\mathbf{x}_n - m_x|^2) = \sigma_x^2$$

are constant over all  $n$ . ( $\sigma_x$  is also called *standard deviation*).

If the above condition holds in addition also for  $k = 2$ , we call the random sequence *wide-sense stationary (WSS)*.

If a sequence is strict-sense stationary, it is always also wide-sense stationary, but not vice versa.

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

A wide-sense stationary random process  $\{\mathbf{x}_n\}$  can not only be characterized by its mean  $m_x = E(\mathbf{x}_n)$  and variance  $\sigma_x^2 = E(|\mathbf{x}_n - m_x|^2)$  over all sample positions  $n$ .

It can, in addition, also be characterized by its *autocorrelation sequence*

$$\phi_{xx}(k) = E(\mathbf{x}_{n+k} \cdot \mathbf{x}_n^*)$$

The autocorrelation sequence of a zero-mean version of a sequence is called the *autocovariance sequence*

$$\gamma_{xx}(k) = E[(\mathbf{x}_{n+k} - m_x) \cdot (\mathbf{x}_n - m_x)^*] = \phi_{xx}(k) - |m_x|^2$$

where  $\gamma_{xx}(0) = \sigma_x^2$ .

A pair of stationary random processes  $\{\mathbf{x}_n\}$  and  $\{\mathbf{y}_n\}$  can, in addition, be *jointly wide-sense stationary* and therefore be characterized by their *crosscorrelation sequence*

$$\phi_{xy}(k) = E(\mathbf{x}_{n+k} \cdot \mathbf{y}_n^*)$$

Their *crosscovariance sequence* is then

$$\gamma_{xy}(k) = E[(\mathbf{x}_{n+k} - m_x) \cdot (\mathbf{y}_n - m_y)^*] = \phi_{xy}(k) - m_x m_y^*$$

The complex conjugates  $*$  are only needed with complex-valued sequences.

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

If  $\dots, \mathbf{x}_{-2}, \mathbf{x}_{-1}, \mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \dots$  is a WSS random sequence, then we can estimate the mean value and auto-correlation sequence from these random variables from any location  $n$  as

$$m_x = E(\mathbf{x}_n)$$
$$\phi_{xx}(k) = E(\mathbf{x}_{n+k} \mathbf{x}_n^*)$$

What if we have just one sample sequence  $\dots, x_{-2}, x_{-1}, x_0, x_1, x_2, \dots$ ?  
If we still can estimate mean and auto-correlation from that as

$$m_x = \lim_{L \rightarrow \infty} \frac{1}{2L+1} \sum_{n=-L}^L x_n \approx \frac{1}{N} \sum_{n=1}^N x_n \quad \text{for large } N$$
$$\phi_{xx}(k) = \lim_{L \rightarrow \infty} \frac{1}{2L+1} \sum_{n=-L}^L x_{n+k} x_n^* \approx \frac{1}{N} \sum_{n=1}^N x_{n+k} x_n^*$$

then we call the process *mean ergodic* and *correlation ergodic*, resp.

Ergodicity means that single-sample-sequence time averages are identical to averages over the entire ensemble for a random process, or, in other words, variation along the time axis looks similar to variation across the ensemble.

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

For deterministic finite-energy sequences  $\{x_n\}$  and  $\{y_n\}$ , we can define their *crosscorrelation sequence* as

$$c_{xy}(k) = \sum_{i=-\infty}^{\infty} x_{i+k} \cdot y_i^* = \sum_{i=-\infty}^{\infty} x_i \cdot y_{i-k}^*.$$

If  $\{x_n\}$  is similar to  $\{y_n\}$ , but lags  $l$  elements behind ( $x_n \approx y_{n-l}$ ), then  $c_{xy}(l)$  will be a peak in the crosscorrelation sequence. It can therefore be used to locate shifted versions of a known sequence in another one.

Swapping the input sequences mirrors the output sequence:  $c_{xy}(k) = c_{yx}^*(-k)$ .

This crosscorrelation sequence is essentially just convolution, with the second input sequence mirrored:

$$\{c_{xy}(n)\} = \{x_n\} * \{y_{-n}^*\}$$

It can therefore be calculated equally easily via the DTFT:

$$C_{xy}(e^{j\omega}) = X(e^{j\omega}) \cdot Y^*(e^{j\omega})$$

DSP.jl's `xcorr` function calculates the crosscorrelation sequence for two finite sequences (vectors), equivalent to `xcorr(x,y) = conv(x,reverse(conj(y)))`

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

# Using `xcorr` to estimate the crosscorrelation

Given two  $m$ -samples long finite sequences  $\{x_n\}_{n=1}^m$  and  $\{y_n\}_{n=1}^m$  sampled from two jointly correlation-ergodic WSS processes  $\{\mathbf{x}_n\}$  and  $\{\mathbf{y}_n\}$ , we can estimate their crosscorrelation sequence

$$\phi_{xy}(k) = E(\mathbf{x}_{n+k} \cdot \mathbf{y}_n^*)$$

for lags  $-m < k < m$  using the estimator

$$\hat{\phi}_{xy}(k) = \frac{1}{m - |k|} \sum_{n=\max\{1, 1-k\}}^{\min\{m, m-k\}} (x_{n+k} \cdot y_n^*)$$

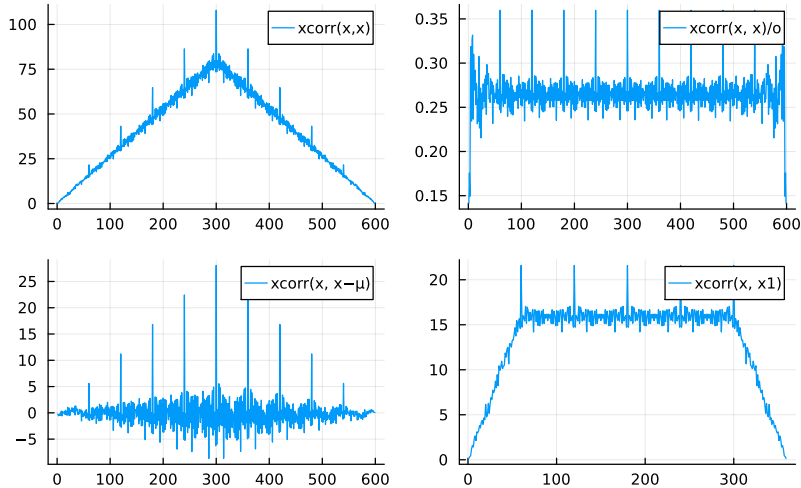
In other words, we calculate the deterministic cross-correlation sequence of both sample sequences, and then divide the result for each lag  $k$  by the length of the overlap,  $m - |k|$ , e.g. as in

```
xcorr(x,y) ./ xcorr(ones(length(x)), ones(length(y))) ==  
xcorr(x,y) ./ [1:m; m-1:-1:1]
```

But as  $k$  approaches  $\pm m$  the overlap drops and the variance of the estimate rises! For a fixed variance, keep the overlap fixed.

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

Example: estimating the auto-correlation/covariance of a periodic signal with xcorr



```
x1 = rand(60); x = repeat(x1, 5); m = length(x)
mu = mean(x); o = [1:m; m-1:-1:1]
plot(plot.([xcorr(x, x), xcorr(x, x)./o, xcorr(x, x.-mu), xcorr(x, x1)])...;
    label=["xcorr(x,x)" "xcorr(x, x)/o" "xcorr(x, x\u2212\u03bc)" "xcorr(x, x1)"],
    layout=(2,2))
```

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

**Random signals**

Digital communication

Audiovisual data compression

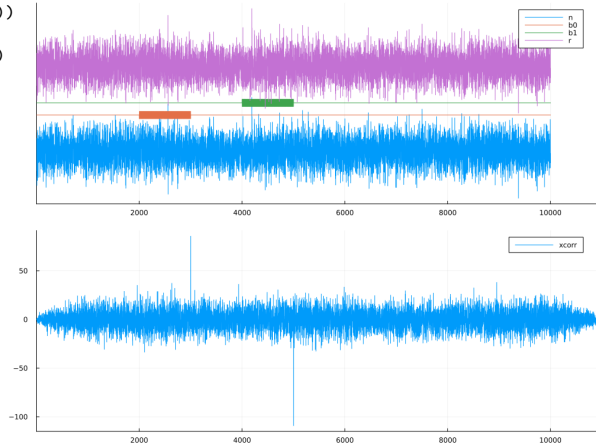
Outlook

# Demonstration of covert spread-spectrum communication:

```
n = randn(10000); a = 0.3; l = 1000
pattern = rand((-a, a), l)
b0 = [zeros(2000); pattern; zeros(7000)]
b1 = [zeros(4000); -pattern; zeros(5000)]
r = n .+ b0 .+ b1
f1 = plot([n b0 b1 r] .- [0 -3 -4 -7]; label = ["n" "b0" "b1" "r"], yticks= [])
```

```
x = conv(r,reverse(pattern))
# or: x = xcorr(r,pattern)
f2 = plot(x; label="xcorr")
```

```
xlims!(f1, 1, length(n)+1)
xlims!(f2, 1, length(n)+1)
plot(f1, f2; layout=(2,1))
```



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

Equivalently, we define the deterministic autocorrelation sequence in the time domain as

$$c_{xx}(k) = \sum_{i=-\infty}^{\infty} x_{i+k} x_i^*$$

This is just the sequence convolved with a time-reversed version of itself:

$$\{c_{xx}(k)\} = \{x_i\} * \{x_{-i}^*\}$$

This corresponds in the frequency domain to

$$C_{xx}(e^{j\omega}) = X(e^{j\omega}) \cdot X^*(e^{j\omega}) = |X(e^{j\omega})|^2.$$

In other words, the DTFT  $C_{xx}(e^{j\omega})$  of the autocorrelation sequence  $\{c_{xx}(n)\}$  of a sequence  $\{x_n\}$  is identical to the squared amplitudes of the DTFT, or *power spectrum*, of  $\{x_n\}$ .

This suggests that the DTFT of the autocorrelation sequence of a random process might be a suitable way for defining the power spectrum of that random process.

What can we say about the phase in the Fourier spectrum of a time-invariant random process?

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

For a zero-mean wide-sense stationary random sequence  $\{\mathbf{x}_n\}$  with absolutely summable autocorrelation sequence

$$\phi_{xx}(k) = E(\mathbf{x}_{n+k} \cdot \mathbf{x}_n^*)$$

we call the DTFT

$$\Phi_{xx}(e^{j\omega}) = \sum_{n=-\infty}^{\infty} \phi_{xx}(n) \cdot e^{-j\omega n}$$

of its autocorrelation sequence the *power density spectrum (PDS)* or *power spectrum* of  $\{\mathbf{x}_n\}$ .

The power spectrum is real, even<sup>†</sup>, non-negative and periodic.

<sup>†</sup> for real-valued sequences

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

The autocorrelation of a sequence  $\{\mathbf{x}_n\}$  with power spectrum  $\Phi_{xx}(e^{j\omega})$  is

$$\phi_{xx}(k) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \Phi_{xx}(e^{j\omega}) e^{jk\omega} d\omega$$

Since the variance of  $\{\mathbf{x}_n\}$  is

$$\text{Var}(\mathbf{x}_n) = \phi_{xx}(0) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \Phi_{xx}(e^{j\omega}) d\omega$$

we can interpret

$$\frac{1}{\pi} \int_{2\pi \frac{f_l}{f_s}}^{2\pi \frac{f_h}{f_s}} \Phi_{xx}(e^{j\omega}) d\omega$$

as the variance of the output of an ideal band-pass filter applied to  $\{\mathbf{x}_n\}$  with cut-off frequencies  $0 \leq f_l < f_h$ .

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

Let  $\{\mathbf{x}_n\}$  be a random sequence from a WSS random process. The output

$$\mathbf{y}_n = \sum_{k=-\infty}^{\infty} h_k \cdot \mathbf{x}_{n-k} = \sum_{k=-\infty}^{\infty} h_{n-k} \cdot \mathbf{x}_k$$

of an LTI system is then another random sequence, characterized by

$$m_y = E(\mathbf{y}_n) = E\left(\sum_{k=-\infty}^{\infty} h_k \cdot \mathbf{x}_{n-k}\right) = \sum_{k=-\infty}^{\infty} h_k \cdot E(\mathbf{x}_{n-k}) = m_x \sum_{k=-\infty}^{\infty} h_k$$

and

$$\phi_{yy}(k) = \sum_{i=-\infty}^{\infty} \phi_{xx}(k-i) c_{hh}(i), \quad \text{where} \quad \begin{aligned} \phi_{xx}(k) &= E(\mathbf{x}_{n+k} \cdot \mathbf{x}_n^*) \\ c_{hh}(k) &= \sum_{i=-\infty}^{\infty} h_{i+k} h_i^* \end{aligned}$$

In other words:

$$\begin{aligned} \{\mathbf{y}_n\} = \{h_n\} * \{\mathbf{x}_n\} &\Rightarrow \begin{aligned} \{\phi_{yy}(n)\} &= \{c_{hh}(n)\} * \{\phi_{xx}(n)\} \\ \Phi_{yy}(e^{j\omega}) &= |H(e^{j\omega})|^2 \cdot \Phi_{xx}(e^{j\omega}) \end{aligned} \end{aligned}$$

Similarly:

$$\begin{aligned} \{\mathbf{y}_n\} = \{h_n\} * \{\mathbf{x}_n\} &\Rightarrow \begin{aligned} \{\phi_{yx}(n)\} &= \{h_n\} * \{\phi_{xx}(n)\} \\ \Phi_{yx}(e^{j\omega}) &= H(e^{j\omega}) \cdot \Phi_{xx}(e^{j\omega}) \end{aligned} \end{aligned}$$

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

## Summary:

$$\{\mathbf{y}_n\} = \{h_n\} * \{\mathbf{x}_n\} \Rightarrow \{\phi_{xx}(n)\} \xrightarrow{* \{h_n\}} \{\phi_{yx}(n)\} \xrightarrow{* \{h_n^*\}} \{\phi_{yy}(n)\}$$

## Proofs:

$$\begin{aligned}\phi_{yx}(l) &= \mathbb{E}(\mathbf{x}_n^* \cdot \mathbf{y}_{n+l}) = \mathbb{E}\left(\mathbf{x}_n^* \cdot \sum_{k=-\infty}^{\infty} h_k \cdot \mathbf{x}_{n+l-k}\right) = \\ &= \sum_{k=-\infty}^{\infty} h_k \cdot \mathbb{E}(\mathbf{x}_n^* \cdot \mathbf{x}_{n+l-k}) \stackrel{\text{WSS}}{=} \sum_{k=-\infty}^{\infty} h_k \cdot \phi_{xx}(l-k)\end{aligned}$$

$$\begin{aligned}\phi_{yy}(l) &= \mathbb{E}(\mathbf{y}_n^* \cdot \mathbf{y}_{n+l}) = \mathbb{E}\left(\sum_{k=-\infty}^{\infty} h_k^* \cdot \mathbf{x}_{n-k}^* \cdot \sum_{m=-\infty}^{\infty} h_m \cdot \mathbf{x}_{n+l-m}\right) = \\ &= \sum_{k=-\infty}^{\infty} h_k^* \cdot \sum_{m=-\infty}^{\infty} h_m \cdot \mathbb{E}(\mathbf{x}_{n-k}^* \cdot \mathbf{x}_{n+l-m}) \stackrel{\text{WSS}}{=} \\ &= \sum_{k=-\infty}^{\infty} h_k^* \cdot \sum_{m=-\infty}^{\infty} h_m \cdot \phi_{xx}(l+k-m) \stackrel{i:=m-k}{=} \\ &= \sum_{k=-\infty}^{\infty} h_k^* \cdot \sum_{i=-\infty}^{\infty} h_{k+i} \cdot \phi_{xx}(l-i) = \sum_{i=-\infty}^{\infty} \phi_{xx}(l-i) \underbrace{\sum_{k=-\infty}^{\infty} h_k^* \cdot h_{k+i}}_{c_{hh}(i)}\end{aligned}$$

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

A random sequence  $\{\mathbf{x}_n\}$  is a *white noise* signal, if  $m_x = 0$  and

$$\phi_{xx}(k) = \sigma_x^2 \delta_k.$$

The power spectrum of a white noise signal is flat:

$$\Phi_{xx}(e^{j\omega}) = \sigma_x^2.$$

A commonly used form of white noise is *white Gaussian noise (WGN)*, where each random variable  $\mathbf{x}_n$  is independent and identically distributed (i.i.d.) according to the normal-distribution probability density function

$$p_{\mathbf{x}_n}(x) = \frac{1}{\sqrt{2\pi\sigma_x^2}} e^{-\frac{(x-m_x)^2}{2\sigma_x^2}}$$

Application example:

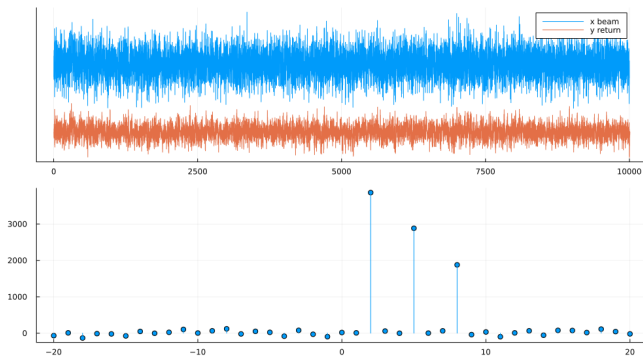
Where an LTI system  $\{\mathbf{y}_n\} = \{\mathbf{h}_n\} * \{\mathbf{x}_n\}$  can be observed to operate on white noise  $\{\mathbf{x}_n\}$  with  $\phi_{xx}(k) = \sigma_x^2 \delta_k$ , the crosscorrelation between input and output will reveal the impulse response of the system:

$$\phi_{yx}(k) = \sigma_x^2 \cdot h_k$$

where  $\phi_{yx}(k) = \phi_{xy}^*(-k) = E(\mathbf{y}_{n+k} \cdot \mathbf{x}_n^*)$ .

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

# Demonstration of covert spread-spectrum radar:



```
x = randn(10000)                # outgoing radar beam
h = [0, 0, 0.4, 0, 0, 0.3, 0, 0, 0.2, 0, 0] # target impulse response
y = conv(x, h)                  # return signal
f1 = plot(1:length(x), x,      label = "x beam")
plot(f1, 1:length(y), y ./ 5, label = "y return", yticks=[])
c = conv(reverse(x), y)         # detected target echos
lags = -20:20
f2 = sticks(lags, c[(length(c)-length(h))÷2 .+ lags .+ 1];
            markershape=:circle, legend=false)
plot(f1, f2; layout=(2,1))
```

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

# Dot product on complex vectors and sequences

Given vectors  $x, y \in \mathbb{R}^n$ , the dot product (or scalar product)

$$x \cdot y = x^T y = \sum_{i=1}^n x_i y_i$$

leads to the Euclidean norm  $\sqrt{x \cdot x} = \sqrt{x^T x} = \|x\| \geq 0$  with:

$$x \cdot x = x^T x = 0 \quad \Rightarrow \quad x = (0 \dots 0)^T.$$

But if  $x, y \in \mathbb{C}^n$ , this (“positive definiteness”) no longer works. Example:

$$(1 \ j) \begin{pmatrix} 1 \\ j \end{pmatrix} = 1 - 1 = 0$$

**Solution:** define dot product over complex vectors as

$$x \cdot y = x^T y^* = y^H x = \sum_{i=1}^n x_i y_i^*$$

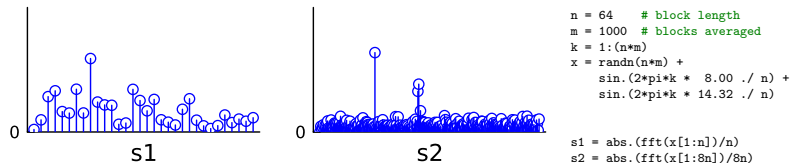
such that  $\|x\|^2 = x \cdot x = x^H x = \sum x_i x_i^* = \sum |x_i|^2$ .

Similarly for cross-correlation of random variables and sequences.

[Introduction](#)
[Sequences and systems](#)
[Convolution](#)
[Fourier transform](#)
[Sampling](#)
[Discrete Fourier transform](#)
[Deconvolution](#)
[Spectral estimation](#)
[Digital filters](#)
[IIR filters](#)
[Random signals](#)
[Digital communication](#)
[Audiovisual data compression](#)
[Outlook](#)

Estimate amplitude spectrum of the noisy discrete sequence

$$x_k = \sin(2\pi k \times 8/64) + \sin(2\pi k \times 14.32/64) + n_k \text{ with } \phi_{nn}(i) = 4\delta_i$$



**s1** Absolute values of a single 64-element DFT of  $\{x_n\}_{n=1}^{64}$  (rect. window).

The flat spectrum of white noise is only an expected value. In a single discrete Fourier transform of such a sequence, the significant variance of the noise spectrum becomes visible. It almost drowns the two peaks from the sine waves.

**s2** Absolute values of a single 512-element DFT of  $\{x_n\}_{n=1}^{512}$  (rect. window).

With an  $8\times$  larger window, the bandwidth of each frequency bin is now reduced  $8\times$ , so the sine functions stand out better from the noise. However, the variance in each frequency bin relative to the expected value remains the same.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

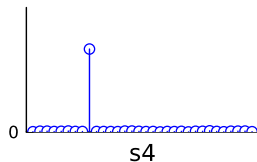
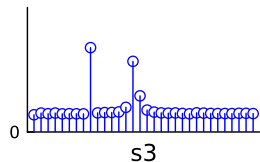
Digital communication

Audiovisual data compression

Outlook

Estimate amplitude spectrum of the noisy discrete sequence

$$x_k = \sin(2\pi k \times 8/64) + \sin(2\pi k \times 14.32/64) + n_k \text{ with } \phi_{nn}(i) = 4\delta_i$$



```
n = 64      # block length
m = 1000    # blocks averaged
k = 1:(n*m)
x = randn(n*m) +
    sin.(2*pi*k * 8.00 ./ n) +
    sin.(2*pi*k * 14.32 ./ n)
xx = reshape(x, n, m)

s3 = mean(abs.(fft(xx, 1)/n), dims=2)
s4 = abs.(mean(fft(xx, 1)/n, dims=2))
```

- s3**  $\{x_n\}_{n=1}^{64000}$  cut into 1000 consecutive 64-sample windows, showing the average of the absolute values of the DFT of each window.

*Non-coherent averaging*: discard phase information first.

This better approximates the shape of the power spectrum: with a flat noise floor.

- s4** Same 1000 windows, but this time the complex values of the DFTs averaged *before* the absolute value was taken  $\Rightarrow$  *coherent averaging*.

Because DFT is linear, this is identical to first averaging all 1000 windows and then applying a single DFT and taking its absolute value.

The windows start 64 samples apart. Only periodic waveforms with a period length that divides 64 are not averaged away. This periodic averaging step suppresses both the noise and the second sine wave.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

“Periodogram”: Single-rectangular-window DTFT power spectrum of a random sequence  $\{x_n\}$ :  $|X(\dot{\omega})|^2$  with  $X(\dot{\omega}) = \sum_{n=0}^{N-1} x_n \cdot e^{-j\dot{\omega}n}$ .

Problem:  $\frac{\text{Var}[|X(\dot{\omega})|^2]}{\mathbb{E}[|X(\dot{\omega})|^2]}$  does not drop with increasing window length  $N$ .

“Welch's method” for estimating the PSD makes three improvements:

- ▶ Reduce leakage using a non-rectangular window sequence  $\{w_i\}$  (“modified periodogram”)
- ▶ To reduce the variance, average  $K$  periodograms of length  $N$ .
- ▶ Triangular, Hamming, Hann, etc. windows can be used with 50% overlap ( $L = N/2$ ), such that all samples contribute with equal weight.

$$x_{k,n} = x_{k \cdot L + n} \cdot w_n, \quad \begin{array}{l} 0 \leq k < K \\ 0 \leq n < N \end{array}$$

$$X_k(\dot{\omega}) = \sum_{n=0}^{N-1} x_{k,n} \cdot e^{-j\dot{\omega}n}$$

$$P(\dot{\omega}) = \frac{1}{K} \sum_{k=0}^{K-1} |X_k(\dot{\omega})|^2$$

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

If a signal  $x(t)$  has a periodic component with period length  $t_p$ , then we can isolate this periodic component from discrete sequence  $x_n = x(n/f_s)$  by *periodic averaging*

$$\bar{x}_n = \lim_{L \rightarrow \infty} \frac{1}{2L+1} \sum_{i=-L}^L x_{n+pi} \approx \frac{1}{N} \sum_{i=1}^N x_{n+pi}, \quad n \in \{0, \dots, p-1\}$$

but only if the period length in samples  $p = t_p \cdot f_s$  is an integer.

Otherwise  $\{x_n\}$  may need to be interpolated and resampled at an integer multiple of  $t_p^{-1}$  first.

Periodic averaging of  $x(t)$  corresponds in the time domain to convolution with a windowed Dirac comb  $a(t) = w(t) \cdot \sum_i \delta(t - t_p i)$ :

$$\bar{x}(t) = \int_s x(t-s) \cdot a(s) ds$$

In the frequency domain, this means multiplication with a  $t_p^{-1}$ -spaced Dirac comb that has been convolved with  $W(f)$ .

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Outlook](#)

# Parametric models of the power spectrum

If we understand the physical process that generates a random sequence, we may be able to model and estimate its power spectrum more accurately, with fewer parameters.

If  $\{\mathbf{x}_n\}$  can be modelled as white noise filtered by an LTI system  $H(e^{j\omega})$ , then

$$\Phi_{xx}(e^{j\omega}) = \sigma_w^2 |H(e^{j\omega})|^2.$$

Often such an LTI system can be modelled as an IIR filter with

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \dots + b_m z^{-m}}{a_0 + a_1 z^{-1} + a_2 z^{-2} + \dots + a_k z^{-k}} \quad \text{with } z = e^{j\omega}.$$

The *auto-regressive moving-average model*  $ARMA(k, m)$  is

$$\mathbf{x}_n = \sum_{l=0}^m b_l \cdot \mathbf{w}_{n-l} - \sum_{l=1}^k a_l \cdot \mathbf{x}_{n-l}$$

where  $\{\mathbf{w}_n\}$  is stationary white noise with variance  $\sigma_w^2$ .

There is also the simpler *AR(k) model*  $\mathbf{x}_n = \mathbf{w}_n - \sum_{l=1}^k a_l \cdot \mathbf{x}_{n-l}$ .

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

- ① Sequences and systems
- ② Convolution
- ③ Fourier transform
- ④ Sampling
- ⑤ Discrete Fourier transform
- ⑥ Deconvolution
- ⑦ Spectral estimation
- ⑧ Digital filters
- ⑨ IIR filters
- ⑩ Random signals
- ⑪ Digital communication**
  - IQ sampling
  - AM/FM demodulation
  - Modems
- ⑫ Audiovisual data compression

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

**Digital communication**

IQ sampling

AM/FM demodulation

Modems

Audiovisual data compression

Outlook

Consider signal  $x(t) \in \mathbb{R}$  in which only frequencies  $f_l < |f| < f_h$  are of interest. This band has a centre frequency of  $f_c = (f_l + f_h)/2$  and a bandwidth  $B = f_h - f_l$ . It can be sampled efficiently (at the lowest possible sampling frequency) by *downconversion*:

- ▶ Shift its spectrum by  $-f_c$ :

$$y(t) = x(t) \cdot e^{-2\pi j f_c t}$$

- ▶ Low-pass filter it with a cut-off frequency of  $B/2$ :

$$z(t) = B \int_{-\infty}^{\infty} y(\tau) \cdot \text{sinc}((t - \tau)B) \cdot d\tau \quad \bullet \circ \quad Z(f) = Y(f) \cdot \text{rect}(f/B)$$

- ▶ Sample the result at sampling frequency  $f_s \geq B$ :

$$z_n = z(n/f_s)$$

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

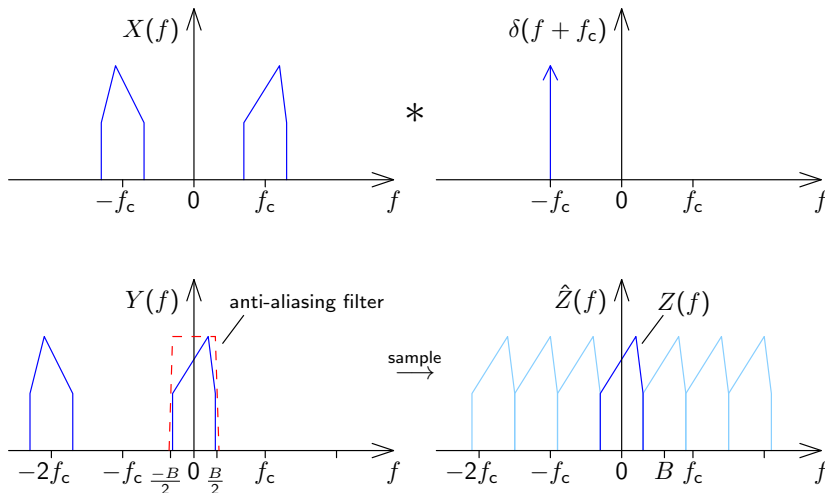
IQ sampling

AM/FM demodulation

Modems

Audiovisual data compression

Outlook



Shifting the center frequency  $f_c$  of the interval of interest to 0 Hz (DC) makes the spectrum asymmetric. This leads to a complex-valued time-domain representation

$$(\exists f : Z(f) \neq [Z(-f)]^* \implies \exists t : z(t) \in \mathbb{C} \setminus \mathbb{R}).$$

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

IQ sampling

AM/FM demodulation

Modems

Audiovisual data compression

Outlook

Given a discrete sequence of downconverted samples  $z_n \in \mathbb{C}$  recorded with sampling frequency  $f_s$  at centre frequency  $f_c$  (as on slide 177), how can we reconstruct a continuous waveform  $\tilde{x}(t) \in \mathbb{R}$  that matches the original signal  $x(t)$  within the frequency interval  $f_l$  to  $f_h$ ?

Reconstruction steps:

- Interpolation of complex baseband signal (remove aliases):

$$\tilde{z}(t) = \sum_{n=-\infty}^{\infty} z_n \cdot \text{sinc}(t \cdot f_s - n)$$

- Upconvert by modulating a complex phasor at carrier frequency  $f_c$ . Then discard the imaginary part (to reconstruct the negative frequency components of the original real-valued signal):

$$\begin{aligned}\tilde{x}(t) &= 2\Re\left(\tilde{z}(t) \cdot e^{2\pi j f_c t}\right) \\ &= 2\Re\left(\left(\Re(\tilde{z}(t)) + j\Im(\tilde{z}(t))\right) \cdot (\cos 2\pi f_c t + j \sin 2\pi f_c t)\right) \\ &= 2\Re(\tilde{z}(t)) \cdot \cos 2\pi f_c t - 2\Im(\tilde{z}(t)) \cdot \sin 2\pi f_c t\end{aligned}$$

Recall that  $2\Re(c) = c + c^*$  for all  $c \in \mathbb{C}$ .

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

IQ sampling

AM/FM demodulation

Modems

Audiovisual data compression

Outlook

# Example: IQ downconversion of a sine wave

What happens if we downconvert the input signal

$$x(t) = A \cdot \cos(2\pi f t + \phi) = \frac{A}{2} \cdot e^{2\pi j f t + j\phi} + \frac{A}{2} \cdot e^{-2\pi j f t - j\phi}$$

using centre frequency  $f_c$  and bandwidth  $B < 2f_c$  with  $|f - f_c| < B/2$ ?

After frequency shift:

$$y(t) = x(t) \cdot e^{-2\pi j f_c t} = \frac{A}{2} \cdot e^{2\pi j (f - f_c) t + j\phi} + \frac{A}{2} \cdot e^{-2\pi j (f + f_c) t - j\phi}$$

After low-pass filter with cut-off frequency  $B/2 < f_c < f + f_c$ :

$$z(t) = \frac{A}{2} \cdot e^{2\pi j (f - f_c) t + j\phi}$$

After sampling:

$$z_n = \frac{A}{2} \cdot e^{2\pi j (f - f_c) n / f_s + j\phi}$$

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

IQ sampling

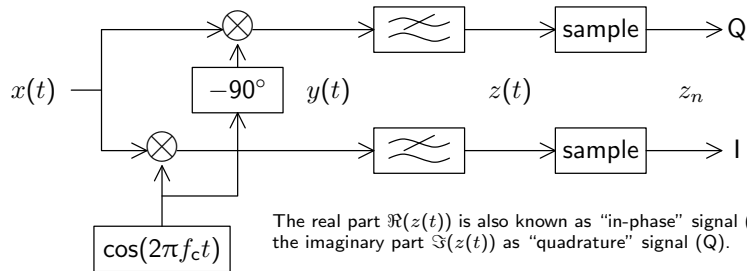
AM/FM demodulation

Modems

Audiovisual data compression

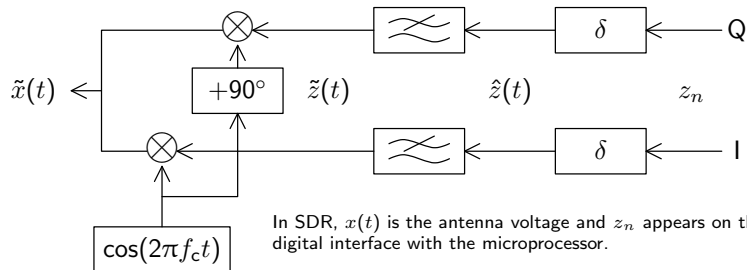
Outlook

IQ downconversion in SDR receiver:



The real part  $\Re(z(t))$  is also known as “in-phase” signal (I) and the imaginary part  $\Im(z(t))$  as “quadrature” signal (Q).

IQ upconversion in SDR transmitter:



In SDR,  $x(t)$  is the antenna voltage and  $z_n$  appears on the digital interface with the microprocessor.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

IQ sampling

AM/FM demodulation

Modems

Audiovisual data compression

Outlook

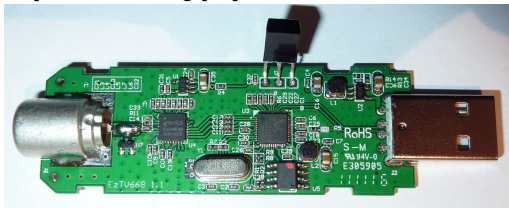
# SDR front-end hardware examples

Low-cost USB-dongle receivers:  $\approx$ £20

Realtek RTL2832U/R820T (RTL-SDR)

USB2,  $f_s < 2.5$  MHz,  $f_c = 24$ –1776 MHz, 8-bit IQ samples

<https://osmocom.org/projects/rtl-sdr/wiki>



SDR front ends are also commonly used today in military radios, spectrum surveillance, amateur-radio stations, mobile-phone base stations, MRI machines, radars, etc.

Mid range transceivers: £250–£2k

HackRF One, Ettus USRP B200/N200, etc.

USB3 or 1-Gbit Ethernet,  $f_s = 10$ –50 MHz,

$f_c = 0$ –6 GHz, 16-bit IQ samples



High-end measurement kit: £3k–£40k

National Instruments (NI), Rohde&Schwarz, etc.

10 Gbit/PCIe, FPGA,  $B$ ,  $f_s = 60$ –1000 MHz,

$f_c = 0$ –14 GHz, float32 IQ samples in volts



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

IQ sampling

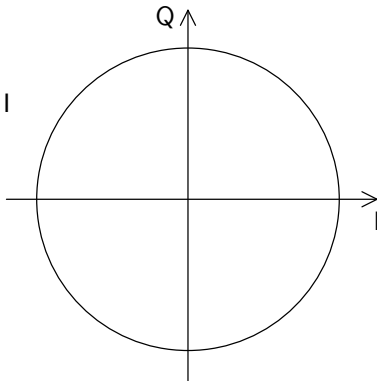
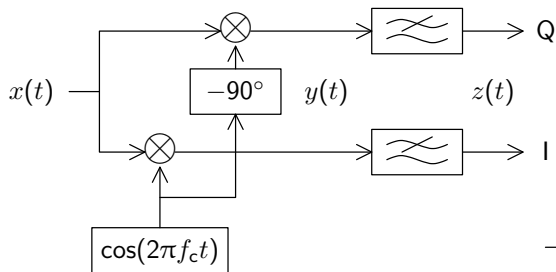
AM/FM demodulation

Modems

Audiovisual data compression

Outlook

# Visualization of IQ representation of sine waves



Recall these products of sine and cosine functions:

- ▶  $\cos(x) \cdot \cos(y) = \frac{1}{2} \cos(x - y) + \frac{1}{2} \cos(x + y)$
- ▶  $\sin(x) \cdot \sin(y) = \frac{1}{2} \cos(x - y) - \frac{1}{2} \cos(x + y)$
- ▶  $\sin(x) \cdot \cos(y) = \frac{1}{2} \sin(x - y) + \frac{1}{2} \sin(x + y)$

Consider: (with  $x = 2\pi f_c t$ )

- ▶  $\sin(x) = \cos(x - \frac{1}{2}\pi)$
- ▶  $\cos(x) \cdot \cos(x) = \frac{1}{2} + \frac{1}{2} \cos 2x$
- ▶  $\sin(x) \cdot \sin(x) = \frac{1}{2} - \frac{1}{2} \cos 2x$
- ▶  $\sin(x) \cdot \cos(x) = 0 + \frac{1}{2} \sin 2x$
- ▶  $\cos(x) \cdot \cos(x - \varphi) = \frac{1}{2} \cos(\varphi) + \frac{1}{2} \cos(2x - \varphi)$
- ▶  $\sin(x) \cdot \cos(x - \varphi) = \frac{1}{2} \sin(\varphi) + \frac{1}{2} \sin(2x - \varphi)$

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[IQ sampling](#)[AM/FM demodulation](#)[Modems](#)[Audiovisual data compression](#)[Outlook](#)

# IQ representation of amplitude-modulated signals

Assume voice signal  $s(t)$  contains only frequencies below  $B/2$ .

Antenna signal amplitude-modulated with carrier frequency  $f_c$ :

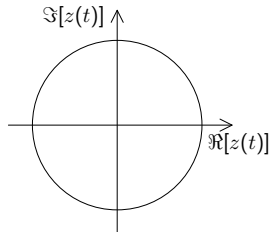
$$x(t) = s(t) \cdot A \cdot \cos(2\pi f_c t + \varphi)$$

After IQ downconversion with centre frequency  $f'_c \approx f_c$ :

$$z(t) = \frac{A}{2} \cdot s(t) \cdot e^{2\pi j(f_c - f'_c)t + j\varphi}$$

With perfect receiver tuning  $f'_c = f_c$ :

$$z(t) = \frac{A}{2} \cdot s(t) \cdot e^{j\varphi}$$



Reception techniques:

- ▶ Non-coherent demodulation (requires  $s(t) \geq 0$ ):

$$s(t) = \frac{2}{A} |z(t)|$$

- ▶ Coherent demodulation (requires knowing  $\varphi$  and  $f'_c = f_c$ ):

$$s(t) = \frac{2}{A} \Re[z(t) \cdot e^{-j\varphi}]$$

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[IQ sampling](#)[AM/FM demodulation](#)[Modems](#)[Audiovisual data compression](#)[Outlook](#)

In frequency modulation, the voice signal  $s(t)$  changes the carrier frequency  $f_c$ :  $f_c(t) = f_c + k \cdot s(t)$

Compared to a constant-frequency carrier signal  $\cos(2\pi f_c t + \varphi)$ , to allow variable frequency, we need to replace the phase-accumulating term  $2\pi f_c t$  with an integral  $2\pi \int f_c(t) dt$ .

Frequency-modulated antenna signal:

$$\begin{aligned} x(t) &= A \cdot \cos \left[ 2\pi \cdot \int_0^t [f_c + k \cdot s(\tau)] d\tau + \varphi \right] \\ &= A \cdot \cos \left[ 2\pi f_c t + 2\pi k \cdot \int_0^t s(\tau) d\tau + \varphi \right] \end{aligned}$$

After IQ downconversion from centre frequency  $f_c$ :

$$z(t) = \frac{A}{2} \cdot e^{2\pi j k \int_0^t s(\tau) d\tau + j\varphi}$$

Therefore,  $s(t)$  is proportional to the rotational rate of  $z(t)$ .

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[IQ sampling](#)[AM/FM demodulation](#)[Modems](#)[Audiovisual data compression](#)[Outlook](#)

Determine  $s(t)$  from downconverted signal  $z(t) = \frac{A}{2} \cdot e^{2\pi jk \int_0^t s(\tau) d\tau + j\varphi}$ .

First idea: measure the angle  $\angle z(t)$ , where the angle operator  $\angle$  is defined such that  $\angle a e^{j\phi} = \phi$  ( $a, \phi \in \mathbb{R}, a > 0$ ). Then take its derivative:

$$s(t) = \frac{1}{2\pi k} \frac{d}{dt} \angle z(t)$$

Problem: angle ambiguity,  $\angle$  works only for  $-\pi \leq \phi < \pi$ .

Ugly hack: MATLAB function `unwrap` removes  $2\pi$  jumps from sample sequences

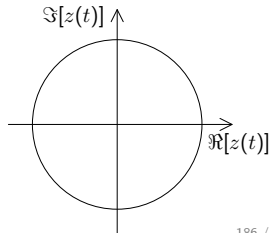
Better idea: first take the complex derivative

$$\frac{dz(t)}{dt} = \frac{A}{2} \cdot 2\pi jk \cdot s(t) \cdot e^{2\pi jk \int_0^t s(\tau) d\tau + j\varphi}$$

then divide by  $z(t)$ :  $\frac{dz(t)}{dt} / z(t) = 2\pi jk \cdot s(t)$

Other practical approaches:

- ▶  $s(t) \propto \Im \left[ \frac{dz(t)}{dt} \cdot z^*(t) \right] / |z(t)|^2$
- ▶  $s(t) \propto \angle \frac{z(t)}{z(t-\Delta t)} / \Delta t$



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

IQ sampling

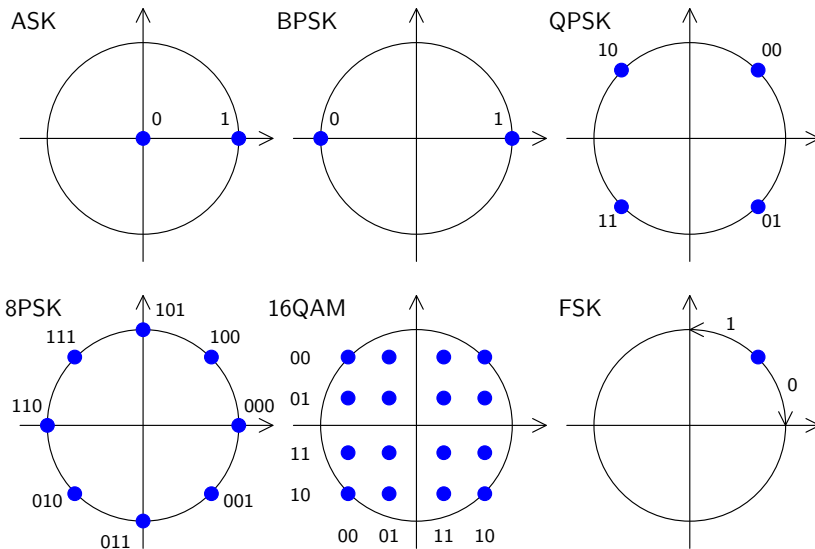
AM/FM demodulation

Modems

Audiovisual data compression

Outlook

Pick  $z_n \in \mathbb{C}$  from a *constellation* of  $2^k$  symbols to encode  $k$  bits:



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

IQ sampling

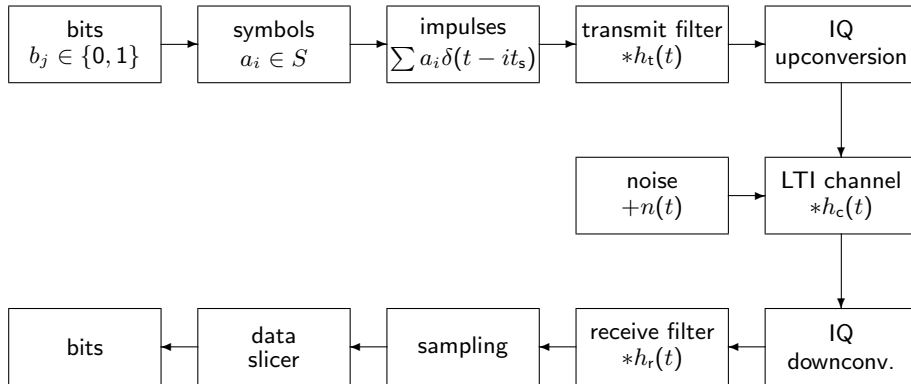
AM/FM demodulation

Modems

Audiovisual data compression

Outlook

# Basic model of a modem



IQ up/down conversion: only required for pass-band channels (e.g., radio)

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

IQ sampling

AM/FM demodulation

Modems

Audiovisual data compression

Outlook

Baseband transmission (e.g., for wires), no IQ up/down conversion

- ▶ binary PAM:  $a_i \in S = \{-A, A\} \subset \mathbb{R}$   
1 bit/symbol  $\Rightarrow$  bit rate (bit/s) = symbol rate (baud)
- ▶  $m$ -ary PAM:  $a_i \in S = \{A_1, \dots, A_m\} \subset \mathbb{R}$   
 $k = \log_2 m$  bit/symbol  $\Rightarrow$  bit rate (bit/s)  $>$  symbol rate (baud)
- ▶ bit sequence  $\{b_j\} \rightarrow$  symbol sequence  $\{a_i\}$ ,  
 $a_i = f(b_{ki}, \dots, b_{ki+k-1})$

Pulse generator (symbol rate  $f_s = t_s^{-1}$ ):

$$\hat{x}(t) = \sum_i a_i \cdot \delta(t - it_s)$$

Transmit filter:  $x = \hat{x} * h_t$ ,  $X(f) = \hat{X}(f) \cdot H_t(f)$

$$x(t) = \sum_i a_i \cdot h_t(t - it_s)$$

Channel:

$$z(t) = \int_0^\infty h_c(s)x(t-s)ds + n(t)$$

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[IQ sampling](#)[AM/FM demodulation](#)[Modems](#)[Audiovisual data compression](#)[Outlook](#)

Receive filter applied to channel output  $z(t)$ :

$$y(t) = \int_0^{\infty} h_r(s)z(t-s)ds$$

Initial symbol pulses  $\hat{x}(t)$  have now passed through three LTIs:

$$y = h * \hat{x}$$

$$h = h_t * h_c * h_r$$

$$H(f) = H_t(f) \cdot H_c(f) \cdot H_r(f)$$

Sample  $y(t)$  at times  $t_n = nt_s + t_d$  with delay  $t_d$  where pulse magnitude is largest:

$$y_n = y(nt_s + t_d) = \sum_i a_i h((n-i)t_s + t_d) + v_n$$

where  $v_n = v(nt_s + t_d)$  is the sampled noise  $v = n * h_r$ .

Data slicer: compare  $y_n$  against thresholds and convert detected nearest symbol  $a'_n \in S$  back into bits  $b'_{kn}, \dots, b'_{kn+k-1}$ .

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

IQ sampling

AM/FM demodulation

Modems

Audiovisual data compression

Outlook

The receiver needs to know the times  $t_n$  when to sample  $y(t)$ .

- ▶ Local clock generators have temperature-dependent frequency drift.
- ▶ In some transmission systems, the transmitter provides the sample clock on a separate wire (or wire pair).

For example: DVI and HDMI video cables contain four wire pairs: three transmitting red/green/blue pixel bytes (using an 8b/10b line encoding), and one providing a pixel clock signal, which the receiver multiplies  $10\times$  to get a bit clock.

- ▶ More commonly, the receiver has to extract the sample clock from the received signal, for example by tracking the phase of transitions (phase locked loop, PLL).

This works reliably only if there are regular transitions.

- Some systems use a **line encoding** (e.g., Manchester code, 8b/10b encoding) to ensure regular transitions.  
Some line encodings add a spectral line at the symbol rate, which the receiver can extract with a band-pass filter, others first require a non-linear step, e.g. squaring.
- Others use a **scrambler**: the data bits  $b_i$  are XORed with the output of synchronized deterministic random-bit generators (e.g., a maximum-length linear feedback shift register), in both the sender and recipient, to make long runs of the same symbol unlikely.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

IQ sampling

AM/FM demodulation

Modems

Audiovisual data compression

Outlook

For notational convenience: set  $t_d = 0$  and allow  $h(t)$  to be non-causal.

$$y_n = a_n h(0) + \sum_{i \neq n} a_i h((n - i)t_s) + v_n$$

Ideally, we want

$$h(it_s) = \begin{cases} 1, & i = 0 \\ 0, & i \neq 0 \end{cases}$$

otherwise  $y_n$  will depend on other (mainly previous) symbols, not just on  $a_n \Rightarrow$  intersymbol interference. (See also: interpolation function)

## Nyquist ISI criterion

$$\begin{aligned} y_n = a_n + v_n &\Leftrightarrow h(t) \cdot \sum_i \delta(t - it_s) = \delta(t) \\ &\Leftrightarrow H(f) * f_s \sum_i \delta(f - if_s) = 1 \\ &\Leftrightarrow \sum_i H(f - if_s) = t_s \end{aligned}$$

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[IQ sampling](#)[AM/FM demodulation](#)[Modems](#)[Audiovisual data compression](#)[Outlook](#)

# Some possible pulse-shape choices

- ▶  $h(t) = \text{rect}(t/t_s) \bullet \circ H(f) = t_s \text{sinc}(f/f_s)$

Rectangular pulses may be practical on fibre optics and short cables, where there are no bandwidth restrictions. Not suitable for radio: bandwidth high compared to symbol rate.

- ▶  $h(t) = \text{sinc}(t/t_s) \bullet \circ H(f) = t_s \text{rect}(f/f_s)$

Most bandwidth efficient pulse shape, but very long symbol waveform, very sensitive to clock synchronization errors.

- ▶ Raised-cosine filter: rectangle with half-period cosine transitions

$$H(f) = \begin{cases} t_s, & |f| \leq f_s/2 - \beta \\ t_s \cos^2 \left[ \frac{\pi}{4\beta} (|f| - f_s/2 + \beta) \right], & f_s/2 - \beta < |f| \leq f_s/2 + \beta \\ 0, & |f| > f_s/2 + \beta \end{cases}$$

$$h(t) = \text{sinc}(t/t_s) \frac{\cos 2\pi\beta t}{1 - (4\beta t)^2}$$

Transition half-width  $\beta$  with  $0 \leq \beta \leq f_s/2$ ; for  $\beta = 0$  this is  $H(f) = t_s \text{rect}(f/f_s)$ .

- ▶ Gaussian filter: both  $h(t)$  and  $H(f)$  are Gaussians (self-Fourier)

Fastest transition without overshoot in either time or frequency domain, but does not satisfy Nyquist ISI criterion.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

IQ sampling

AM/FM demodulation

Modems

Audiovisual data compression

Outlook

# Optimal transmit and receive filtering

Nyquist ISI criterion dictates  $H(f) = H_t(f) \cdot H_c(f) \cdot H_r(f)$ .

Bandwidth limits guide choice of  $H_t(f)$ , and channel dictates  $H_c(f)$  and  $N(f)$ .

How should we then choose  $H_t(f)$  and  $H_r(f)$ ?

Select a received pulse spectrum  $P_r(f)$ , e.g. raised cosine. Then for some arbitrary gain factor  $k > 0$ :

$$H(f) = H_t(f) \cdot H_c(f) \cdot H_r(f) = k \cdot P_r(f)$$

## Optimal filters

Minimize noise variance  $\text{Var}(v_n) = \int N(f) |H_r(f)|^2 df$  at slicer relative to symbol distance.

$$|H_r(f)| = \left| \frac{P_r(f)}{\sqrt{N(f)} H_c(f)} \right|^{\frac{1}{2}}$$

$$|H_t(f)| = k \left| \frac{P_r(f) \sqrt{N(f)}}{H_c(f)} \right|^{\frac{1}{2}}$$

If  $N(f)$  and  $H_c(f)$  are flat:  $|H_r(f)| = |H_t(f)|/k'$ , e.g. root raised cosine.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

IQ sampling

AM/FM demodulation

Modems

Audiovisual data compression

Outlook

- ① Sequences and systems
- ② Convolution
- ③ Fourier transform
- ④ Sampling
- ⑤ Discrete Fourier transform
- ⑥ Deconvolution
- ⑦ Spectral estimation
- ⑧ Digital filters
- ⑨ IIR filters
- ⑩ Random signals
- ⑪ Digital communication
- ⑫ **Audiovisual data compression**
  - Entropy coding
  - Transform coding
  - Decorrelation

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

**Audiovisual data compression**

Entropy coding

Transform coding

Decorrelation

Discrete Cosine Transform

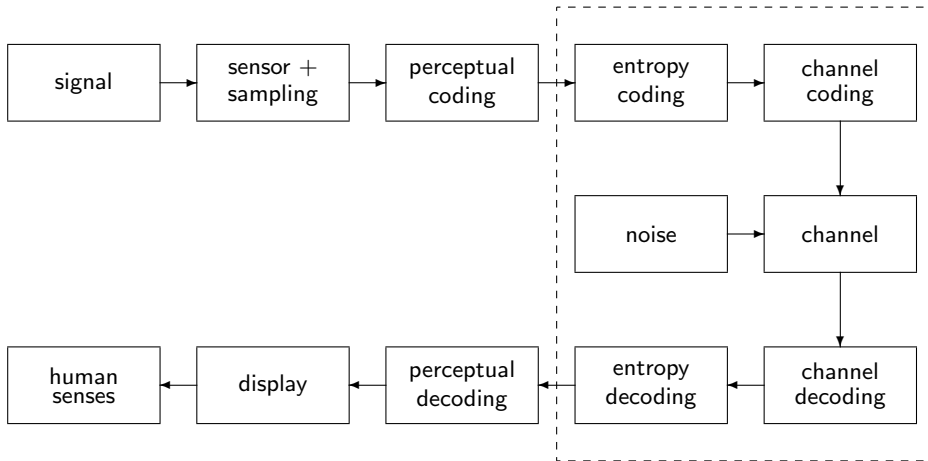
Colour spaces

Quantization

JPEG

Outlook

Structure of modern audiovisual communication systems:



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

**Audiovisual data compression**

Entropy coding

Transform coding

Decorrelation

Discrete Cosine Transform

Colour spaces

Quantization

JPEG

Outlook

Audiovisual lossy coding today typically consists of these steps:

- ▶ A *transducer* converts the original stimulus into a voltage.
- ▶ This analog signal is then *sampled and quantized*.  
The digitization parameters (sampling frequency, quantization levels) are preferably chosen generously beyond the ability of human senses or output devices.
- ▶ The digitized sensor-domain signal is then *transformed into a perceptual domain*.  
This step often mimics some of the first neural processing steps in humans.
- ▶ This signal is *quantized* again, based on a *perceptual model* of what level of quantization-noise humans can still sense.
- ▶ The resulting quantized levels may still be highly statistically dependent. A *prediction or decorrelation transform* exploits this and produces a less dependent symbol sequence of lower entropy.
- ▶ An *entropy coder* turns that into an apparently-random bit string, whose length approximates the remaining entropy.

The first neural processing steps in humans are in effect often a kind of decorrelation transform; our eyes and ears were optimized like any other AV communications system. This allows us to use the same transform for decorrelating and transforming into a perceptually relevant domain.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Entropy coding

Transform coding

Decorrelation

Discrete Cosine Transform

Colour spaces

Quantization

JPEG

Outlook

- ▶ Quick review of entropy coding
- ▶ Transform coding: techniques for converting sequences of highly-dependent symbols into less-dependent lower-entropy sequences.
  - run-length coding
  - decorrelation, Karhunen–Loève transform (PCA)
  - Discrete cosine transform
- ▶ Introduction to some characteristics and limits of human senses
  - perceptual scales and sensitivity limits
  - colour vision
- ▶ Quantization techniques to remove information that is irrelevant to human senses

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

**Audiovisual data compression**

Entropy coding

Transform coding

Decorrelation

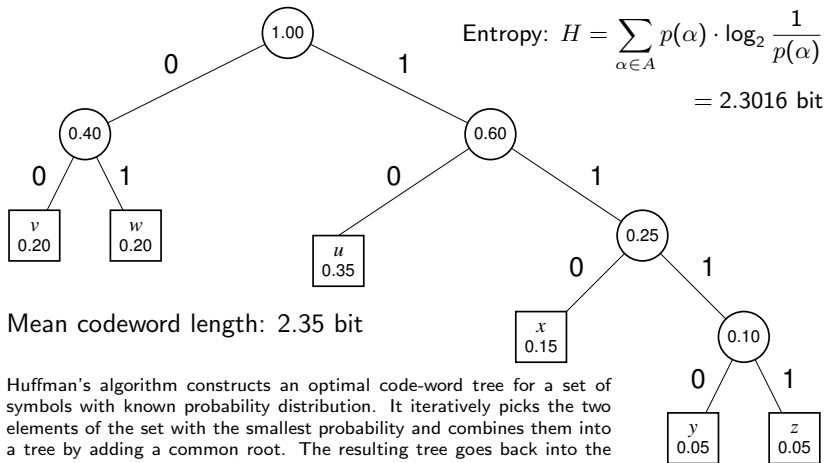
Discrete Cosine Transform

Colour spaces

Quantization

JPEG

Outlook



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

**Entropy coding**

Transform coding

Decorrelation

Discrete Cosine Transform

Colour spaces

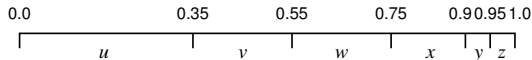
Quantization

JPEG

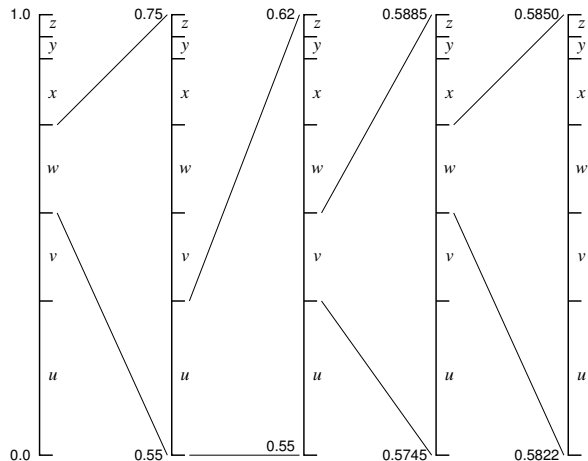
Outlook

# Entropy coding review – arithmetic coding

Partition  $[0,1]$  according  
to symbol probabilities:



Encode text  $wuvw \dots$  as numeric value  $(0.58\dots)$  in nested intervals:



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

**Entropy coding**

Transform coding

Decorrelation

Discrete Cosine Transform

Colour spaces

Quantization

JPEG

Outlook

Several advantages:

- ▶ Length of output bitstring can approximate the theoretical information content of the input to within 1 bit.
- ▶ Performs well with probabilities  $> 0.5$ , where the information per symbol is less than one bit.
- ▶ Interval arithmetic makes it easy to change symbol probabilities (no need to modify code-word tree)  $\Rightarrow$  convenient for adaptive coding

Can be implemented efficiently with fixed-length arithmetic by rounding probabilities and shifting out leading digits as soon as leading zeros appear in interval size. Usually combined with adaptive probability estimation.

Huffman coding remains popular because of its simplicity and lack of patent-licence issues.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

**Entropy coding**

Transform coding

Decorrelation

Discrete Cosine Transform

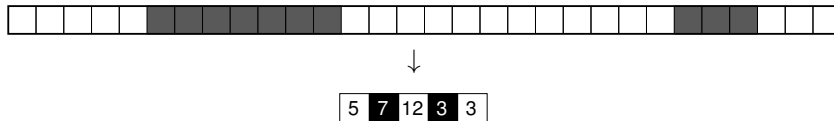
Colour spaces

Quantization

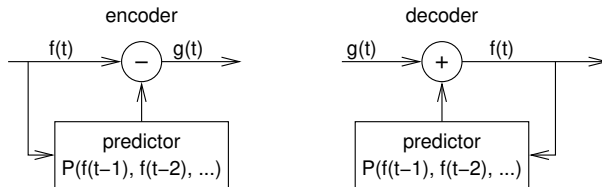
JPEG

Outlook

Run-length coding:



Predictive coding:



Delta coding (DPCM):

$$P(x) = x$$

Linear predictive coding:

$$P(x_1, \dots, x_n) = \sum_{i=1}^n a_i x_i$$

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Entropy coding

**Transform coding**

Decorrelation

Discrete Cosine Transform

Colour spaces

Quantization

JPEG

Outlook

# Old (Group 3 MH) fax code

- ▶ Run-length encoding plus modified Huffman code
- ▶ Fixed code table (from eight sample pages)
- ▶ separate codes for runs of white and black pixels
- ▶ *termination code* in the range 0–63 switches between black and white code
- ▶ *makeup code* can extend length of a run by a multiple of 64
- ▶ termination run length 0 needed where run length is a multiple of 64
- ▶ single white column added on left side before transmission
- ▶ makeup codes above 1728 equal for black and white
- ▶ 12-bit end-of-line marker: 000000000001 (can be prefixed by up to seven zero-bits to reach next byte boundary)

Example: line with 2 w, 4 b, 200 w, 3 b, EOL →  
1000|011|010111|10011|10|000000000001

| pixels | white code | black code    |
|--------|------------|---------------|
| 0      | 00110101   | 0000110111    |
| 1      | 000111     | 010           |
| 2      | 0111       | 11            |
| 3      | 1000       | 10            |
| 4      | 1011       | 011           |
| 5      | 1100       | 0011          |
| 6      | 1110       | 0010          |
| 7      | 1111       | 00011         |
| 8      | 10011      | 000101        |
| 9      | 10100      | 000100        |
| 10     | 00111      | 0000100       |
| 11     | 01000      | 0000101       |
| 12     | 001000     | 0000111       |
| 13     | 000011     | 00000100      |
| 14     | 110100     | 00000111      |
| 15     | 110101     | 000011000     |
| 16     | 101010     | 0000010111    |
| ...    | ...        | ...           |
| 63     | 00110100   | 000001100111  |
| 64     | 11011      | 0000001111    |
| 128    | 10010      | 000011001000  |
| 192    | 010111     | 000011001001  |
| ...    | ...        | ...           |
| 1728   | 010011011  | 0000001100101 |

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Entropy coding

**Transform coding**

Decorrelation

Discrete Cosine Transform

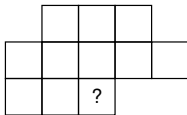
Colour spaces

Quantization

JPEG

Outlook

Performs context-sensitive arithmetic coding of binary pixels. Both encoder and decoder maintain statistics on how the black/white probability of each pixel depends on these 10 previously transmitted neighbours:



Based on the counted numbers  $n_{\text{black}}$  and  $n_{\text{white}}$  of how often each pixel value has been encountered so far in each of the 1024 contexts, the probability for the next pixel being black is estimated as

$$p_{\text{black}} = \frac{n_{\text{black}} + 1}{n_{\text{white}} + n_{\text{black}} + 2}$$

The encoder updates its estimate only after the newly counted pixel has been encoded, such that the decoder knows the exact same statistics.

Joint Bi-level Expert Group: International Standard ISO 11544, 1993.

Example implementation: <https://www.cl.cam.ac.uk/~mgk25/jbigkit/>

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Entropy coding

**Transform coding**

Decorrelation

Discrete Cosine Transform

Colour spaces

Quantization

JPEG

Outlook

Random variables  $\mathbf{X}, \mathbf{Y}$  are *dependent* iff  $\exists x, y$ :

$$P(\mathbf{X} = x \wedge \mathbf{Y} = y) \neq P(\mathbf{X} = x) \cdot P(\mathbf{Y} = y).$$

If  $\mathbf{X}, \mathbf{Y}$  are dependent, then

$$\Rightarrow \exists x, y : P(\mathbf{X} = x \mid \mathbf{Y} = y) \neq P(\mathbf{X} = x) \vee \\ P(\mathbf{Y} = y \mid \mathbf{X} = x) \neq P(\mathbf{Y} = y)$$

$$\Rightarrow H(\mathbf{X} \mid \mathbf{Y}) < H(\mathbf{X}) \vee H(\mathbf{Y} \mid \mathbf{X}) < H(\mathbf{Y})$$

## Application

Where  $x$  is the value of the next symbol to be transmitted and  $y$  is the vector of all symbols transmitted so far, accurate knowledge of the conditional probability  $P(\mathbf{X} = x \mid \mathbf{Y} = y)$  will allow a transmitter to remove all redundancy.

An application example of this approach is JBIG, but there  $y$  is limited to 10 past single-bit pixels and  $P(\mathbf{X} = x \mid \mathbf{Y} = y)$  is only an estimate.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Entropy coding

**Transform coding**

Decorrelation

Discrete Cosine Transform

Colour spaces

Quantization

JPEG

Outlook

The practical estimation of conditional probabilities, in their most general form, based on statistical measurements of example signals, quickly reaches practical limits. JBIG needs an array of only  $2^{11} = 2048$  counting registers to maintain estimator statistics for its 10-bit context.

If we wanted to encode each 24-bit pixel of a colour image based on its statistical dependence on the full colour information of just ten previous neighbour pixels, the required number of

$$(2^{24})^{11} \approx 3 \times 10^{79}$$

registers for storing each probability will exceed the estimated number of particles in this universe. (Neither will we encounter enough pixels to record statistically significant occurrences in all  $(2^{24})^{10}$  contexts.)

This example is far from excessive. It is easy to show that in colour images, pixel values show statistically significant dependence across colour channels, and across locations more than eight pixels apart.

A simpler approximation of dependence is needed: correlation.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Entropy coding

**Transform coding**

Decorrelation

Discrete Cosine Transform

Colour spaces

Quantization

JPEG

Outlook

Two random variables  $\mathbf{X} \in \mathbb{R}$  and  $\mathbf{Y} \in \mathbb{R}$  are *correlated* iff

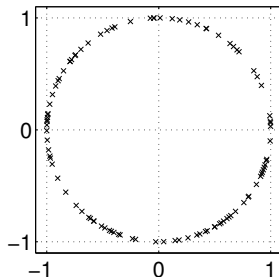
$$E\{[\mathbf{X} - E(\mathbf{X})] \cdot [\mathbf{Y} - E(\mathbf{Y})]\} \neq 0$$

where  $E(\cdots)$  denotes the *expected value* of a random-variable term.

Correlation implies dependence, but dependence does not always lead to correlation (see example to the right).

However, most dependency in audio-visual data is a consequence of correlation, which is algorithmically much easier to exploit.

Dependent but not correlated:

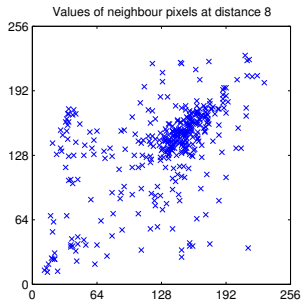
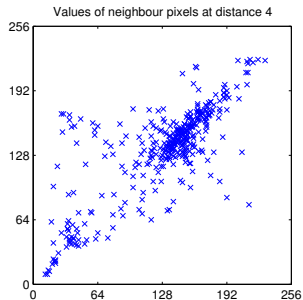
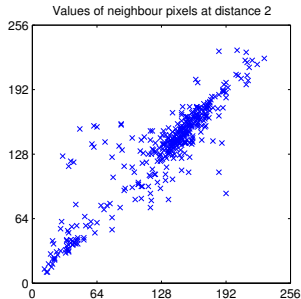
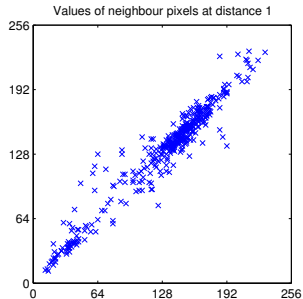


Positive correlation: higher  $\mathbf{X} \Leftrightarrow$  higher  $\mathbf{Y}$ , lower  $\mathbf{X} \Leftrightarrow$  lower  $\mathbf{Y}$

Negative correlation: lower  $\mathbf{X} \Leftrightarrow$  higher  $\mathbf{Y}$ , higher  $\mathbf{X} \Leftrightarrow$  lower  $\mathbf{Y}$

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Entropy coding](#)[Transform coding](#)[Decorrelation](#)[Discrete Cosine Transform](#)[Colour spaces](#)[Quantization](#)[JPEG](#)[Outlook](#)

# Correlation of neighbour pixels



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Entropy coding

Transform coding

**Decorrelation**

Discrete Cosine Transform

Colour spaces

Quantization

JPEG

Outlook

We define the *covariance* of two random variables  $\mathbf{X}$  and  $\mathbf{Y}$  as

$$\text{Cov}(\mathbf{X}, \mathbf{Y}) = \text{E}\{[\mathbf{X} - \text{E}(\mathbf{X})] \cdot [\mathbf{Y} - \text{E}(\mathbf{Y})]\} = \text{E}(\mathbf{X} \cdot \mathbf{Y}) - \text{E}(\mathbf{X}) \cdot \text{E}(\mathbf{Y})$$

and the *variance* as  $\text{Var}(\mathbf{X}) = \text{Cov}(\mathbf{X}, \mathbf{X}) = \text{E}\{[\mathbf{X} - \text{E}(\mathbf{X})]^2\}$ .

The *Pearson correlation coefficient*

$$\rho_{\mathbf{X}, \mathbf{Y}} = \frac{\text{Cov}(\mathbf{X}, \mathbf{Y})}{\sqrt{\text{Var}(\mathbf{X}) \cdot \text{Var}(\mathbf{Y})}}$$

is a normalized form of the covariance. It is limited to the range  $[-1, 1]$ .

If the correlation coefficient has one of the values  $\rho_{\mathbf{X}, \mathbf{Y}} = \pm 1$ , this implies that  $\mathbf{X}$  and  $\mathbf{Y}$  are exactly linearly dependent, i.e.  $\mathbf{Y} = a\mathbf{X} + b$ , with  $a = \text{Cov}(\mathbf{X}, \mathbf{Y})/\text{Var}(\mathbf{X})$  and  $b = \text{E}(\mathbf{Y}) - a\text{E}(\mathbf{X})$ .

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Entropy coding

Transform coding

**Decorrelation**

Discrete Cosine Transform

Colour spaces

Quantization

JPEG

Outlook

For a random vector  $\vec{X} = (\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_n) \in \mathbb{R}^n$  we define the *covariance matrix*

$$\text{Cov}(\vec{X}) = E \left( (\vec{X} - E(\vec{X})) \cdot (\vec{X} - E(\vec{X}))^T \right) = (\text{Cov}(\mathbf{X}_i, \mathbf{X}_j))_{i,j} =$$
$$\begin{pmatrix} \text{Cov}(\mathbf{X}_1, \mathbf{X}_1) & \text{Cov}(\mathbf{X}_1, \mathbf{X}_2) & \text{Cov}(\mathbf{X}_1, \mathbf{X}_3) & \cdots & \text{Cov}(\mathbf{X}_1, \mathbf{X}_n) \\ \text{Cov}(\mathbf{X}_2, \mathbf{X}_1) & \text{Cov}(\mathbf{X}_2, \mathbf{X}_2) & \text{Cov}(\mathbf{X}_2, \mathbf{X}_3) & \cdots & \text{Cov}(\mathbf{X}_2, \mathbf{X}_n) \\ \text{Cov}(\mathbf{X}_3, \mathbf{X}_1) & \text{Cov}(\mathbf{X}_3, \mathbf{X}_2) & \text{Cov}(\mathbf{X}_3, \mathbf{X}_3) & \cdots & \text{Cov}(\mathbf{X}_3, \mathbf{X}_n) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \text{Cov}(\mathbf{X}_n, \mathbf{X}_1) & \text{Cov}(\mathbf{X}_n, \mathbf{X}_2) & \text{Cov}(\mathbf{X}_n, \mathbf{X}_3) & \cdots & \text{Cov}(\mathbf{X}_n, \mathbf{X}_n) \end{pmatrix}$$

The elements of a random vector  $\vec{X}$  are uncorrelated if and only if  $\text{Cov}(\vec{X})$  is a diagonal matrix.

$\text{Cov}(\mathbf{X}, \mathbf{Y}) = \text{Cov}(\mathbf{Y}, \mathbf{X})$ , so all covariance matrices are *symmetric*:  
 $\text{Cov}(\vec{X}) = \text{Cov}^T(\vec{X})$ .

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Entropy coding

Transform coding

**Decorrelation**

Discrete Cosine Transform

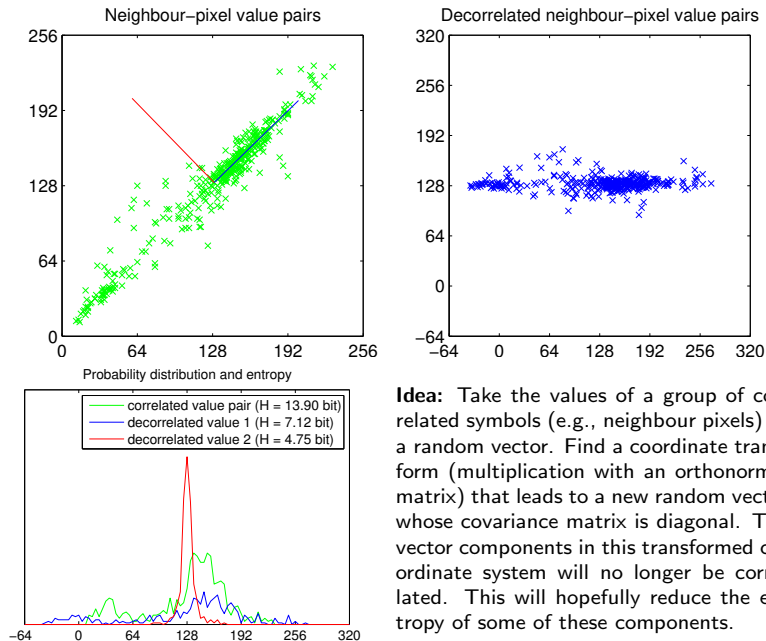
Colour spaces

Quantization

JPEG

Outlook

# Decorrelation by coordinate transform



**Idea:** Take the values of a group of correlated symbols (e.g., neighbour pixels) as a random vector. Find a coordinate transform (multiplication with an orthonormal matrix) that leads to a new random vector whose covariance matrix is diagonal. The vector components in this transformed coordinate system will no longer be correlated. This will hopefully reduce the entropy of some of these components.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Entropy coding

Transform coding

**Decorrelation**

Discrete Cosine Transform

Colour spaces

Quantization

JPEG

Outlook

**Theorem:** Let  $\vec{X} \in \mathbb{R}^n$  and  $\vec{Y} \in \mathbb{R}^n$  be random vectors that are linearly dependent with  $\vec{Y} = A\vec{X} + b$ , where  $A \in \mathbb{R}^{n \times n}$  and  $b \in \mathbb{R}^n$  are constants. Then

$$\begin{aligned} E(\vec{Y}) &= A \cdot E(\vec{X}) + b \\ \text{Cov}(\vec{Y}) &= A \cdot \text{Cov}(\vec{X}) \cdot A^T \end{aligned}$$

**Proof:** The first equation follows from the linearity of the expected-value operator  $E(\cdot)$ , as does  $E(A \cdot \vec{X} \cdot B) = A \cdot E(\vec{X}) \cdot B$  for matrices  $A, B$ . With that, we can transform

$$\begin{aligned} \text{Cov}(\vec{Y}) &= E\left((\vec{Y} - E(\vec{Y})) \cdot (\vec{Y} - E(\vec{Y}))^T\right) \\ &= E\left((A\vec{X} - AE(\vec{X})) \cdot (A\vec{X} - AE(\vec{X}))^T\right) \\ &= E\left(A(\vec{X} - E(\vec{X})) \cdot (\vec{X} - E(\vec{X}))^T A^T\right) \\ &= A \cdot E\left((\vec{X} - E(\vec{X})) \cdot (\vec{X} - E(\vec{X}))^T\right) \cdot A^T \\ &= A \cdot \text{Cov}(\vec{X}) \cdot A^T \end{aligned}$$

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Entropy coding

Transform coding

**Decorrelation**

Discrete Cosine Transform

Colour spaces

Quantization

JPEG

Outlook

We are given a square matrix  $A \in \mathbb{R}^{n \times n}$ . The vector  $x \in \mathbb{R}^n$  is an *eigenvector* of  $A$  if there exists a scalar value  $\lambda \in \mathbb{R}$  such that

$$Ax = \lambda x.$$

The corresponding  $\lambda$  is the *eigenvalue* of  $A$  associated with  $x$ .

The length of an eigenvector is irrelevant, as any multiple of it is also an eigenvector. Eigenvectors are in practice normalized to length 1.

## Spectral decomposition

Any real, *symmetric* matrix  $A = A^T \in \mathbb{R}^{n \times n}$  can be diagonalized into the form

$$A = U\Lambda U^T,$$

where  $\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$  is the diagonal matrix of the ordered eigenvalues of  $A$  (with  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$ ), and the columns of  $U$  are the  $n$  corresponding orthonormal eigenvectors of  $A$ .

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Entropy coding

Transform coding

**Decorrelation**

Discrete Cosine Transform

Colour spaces

Quantization

JPEG

Outlook

We are given a random vector variable  $\vec{\mathbf{X}} \in \mathbb{R}^n$ . The correlation of the elements of  $\vec{\mathbf{X}}$  is described by the covariance matrix  $\text{Cov}(\vec{\mathbf{X}})$ .

How can we find a transform matrix  $A$  that decorrelates  $\vec{\mathbf{X}}$ , i.e. that turns  $\text{Cov}(A\vec{\mathbf{X}}) = A \cdot \text{Cov}(\vec{\mathbf{X}}) \cdot A^T$  into a diagonal matrix?  $A$  would provide us the transformed representation  $\vec{\mathbf{Y}} = A\vec{\mathbf{X}}$  of our random vector, in which all elements are mutually uncorrelated.

Note that  $\text{Cov}(\vec{\mathbf{X}})$  is symmetric. It therefore has  $n$  real eigenvalues  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$  and a set of associated mutually orthogonal eigenvectors  $b_1, b_2, \dots, b_n$  of length 1 with

$$\text{Cov}(\vec{\mathbf{X}})b_i = \lambda_i b_i.$$

We convert this set of equations into matrix notation using the matrix  $B = (b_1, b_2, \dots, b_n)$  that has these eigenvectors as columns and the diagonal matrix  $D = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$  that consists of the corresponding eigenvalues:

$$\text{Cov}(\vec{\mathbf{X}})B = BD$$

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Entropy coding](#)[Transform coding](#)[Decorrelation](#)[Discrete Cosine Transform](#)[Colour spaces](#)[Quantization](#)[JPEG](#)[Outlook](#)

$B$  is orthonormal, that is  $BB^T = I$ .

Multiplying the above from the right with  $B^T$  leads to the *spectral decomposition*

$$\text{Cov}(\vec{X}) = BDB^T$$

of the covariance matrix. Similarly multiplying instead from the left with  $B^T$  leads to

$$B^T \text{Cov}(\vec{X}) B = D$$

and therefore shows with

$$\text{Cov}(B^T \vec{X}) = D$$

that the eigenvector matrix  $B^T$  is the wanted transform.

The *Karhunen–Loève transform* (also known as *Hotelling transform* or *Principal Component Analysis*) is the multiplication of a correlated random vector  $\vec{X}$  with the orthonormal eigenvector matrix  $B^T$  from the spectral decomposition  $\text{Cov}(\vec{X}) = BDB^T$  of its covariance matrix. This leads to a decorrelated random vector  $B^T \vec{X}$  whose covariance matrix is diagonal.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Entropy coding

Transform coding

**Decorrelation**

Discrete Cosine Transform

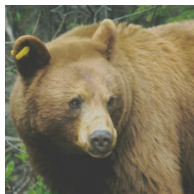
Colour spaces

Quantization

JPEG

Outlook

# Karhunen–Loève transform example I



colour image



red channel



green channel



blue channel

The colour image (left) has  $m = r^2$  pixels, each of which is an  $n = 3$ -dimensional RGB vector

$$I_{x,y} = (r_{x,y}, g_{x,y}, b_{x,y})^T$$

The three rightmost images show each of these colour planes separately as a black/white image.

We want to apply the KLT on a set of such  $\mathbb{R}^n$  colour vectors. Therefore, we reformat the image  $I$  into an  $n \times m$  matrix of the form

$$S = \begin{pmatrix} r_{1,1} & r_{1,2} & r_{1,3} & \cdots & r_{r,r} \\ g_{1,1} & g_{1,2} & g_{1,3} & \cdots & g_{r,r} \\ b_{1,1} & b_{1,2} & b_{1,3} & \cdots & b_{r,r} \end{pmatrix}$$

We can now define the mean colour vector

$$\bar{S}_c = \frac{1}{m} \sum_{i=1}^m S_{c,i}, \quad \bar{S} = \begin{pmatrix} 0.4839 \\ 0.4456 \\ 0.3411 \end{pmatrix}$$

and the covariance matrix

$$C_{c,d} = \frac{1}{m-1} \sum_{i=1}^m (S_{c,i} - \bar{S}_c)(S_{d,i} - \bar{S}_d)$$
$$C = \begin{pmatrix} 0.0328 & 0.0256 & 0.0160 \\ 0.0256 & 0.0216 & 0.0140 \\ 0.0160 & 0.0140 & 0.0109 \end{pmatrix}$$

[“ $m-1$ ” because  $\bar{S}_c$  only estimates the mean]

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Entropy coding

Transform coding

Decorrelation

Discrete Cosine Transform

Colour spaces

Quantization

JPEG

Outlook

The resulting covariance matrix  $C$  has three eigenvalues 0.0622, 0.0025, and 0.0006:

$$\begin{pmatrix} 0.0328 & 0.0256 & 0.0160 \\ 0.0256 & 0.0216 & 0.0140 \\ 0.0160 & 0.0140 & 0.0109 \end{pmatrix} \begin{pmatrix} 0.7167 \\ 0.5833 \\ 0.3822 \end{pmatrix} = 0.0622 \begin{pmatrix} 0.7167 \\ 0.5833 \\ 0.3822 \end{pmatrix}$$
$$\begin{pmatrix} 0.0328 & 0.0256 & 0.0160 \\ 0.0256 & 0.0216 & 0.0140 \\ 0.0160 & 0.0140 & 0.0109 \end{pmatrix} \begin{pmatrix} -0.5509 \\ 0.1373 \\ 0.8232 \end{pmatrix} = 0.0025 \begin{pmatrix} -0.5509 \\ 0.1373 \\ 0.8232 \end{pmatrix}$$
$$\begin{pmatrix} 0.0328 & 0.0256 & 0.0160 \\ 0.0256 & 0.0216 & 0.0140 \\ 0.0160 & 0.0140 & 0.0109 \end{pmatrix} \begin{pmatrix} -0.4277 \\ 0.8005 \\ -0.4198 \end{pmatrix} = 0.0006 \begin{pmatrix} -0.4277 \\ 0.8005 \\ -0.4198 \end{pmatrix}$$

It can thus be diagonalized as

$$\begin{pmatrix} 0.0328 & 0.0256 & 0.0160 \\ 0.0256 & 0.0216 & 0.0140 \\ 0.0160 & 0.0140 & 0.0109 \end{pmatrix} = C = U \cdot D \cdot U^T =$$
$$\begin{pmatrix} 0.7167 & -0.5509 & -0.4277 \\ 0.5833 & 0.1373 & 0.8005 \\ 0.3822 & 0.8232 & -0.4198 \end{pmatrix} \begin{pmatrix} 0.0622 & 0 & 0 \\ 0 & 0.0025 & 0 \\ 0 & 0 & 0.0006 \end{pmatrix} \begin{pmatrix} 0.7167 & 0.5833 & 0.3822 \\ -0.5509 & 0.1373 & 0.8232 \\ -0.4277 & 0.8005 & -0.4198 \end{pmatrix}$$

(e.g. using MATLAB's singular-value decomposition function `svd`).

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Entropy coding](#)[Transform coding](#)[Decorrelation](#)[Discrete Cosine Transform](#)[Colour spaces](#)[Quantization](#)[JPEG](#)[Outlook](#)

Before KLT:



red

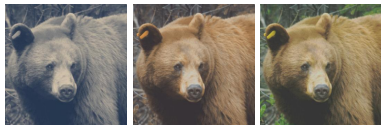
green

blue

After KLT:

 $u$  $v$  $w$ 

Projections on eigenvector subspaces:

 $v = w = 0$  $w = 0$ 

original

We finally apply the orthogonal  $3 \times 3$  transform matrix  $U$ , which we just used to diagonalize the covariance matrix, to the entire image:

$$T = U^T \cdot \left[ S - \begin{pmatrix} \bar{S}_1 & \bar{S}_1 & \cdots & \bar{S}_1 \\ \bar{S}_2 & \bar{S}_2 & \cdots & \bar{S}_2 \\ \bar{S}_3 & \bar{S}_3 & \cdots & \bar{S}_3 \end{pmatrix} \right] + \begin{pmatrix} \bar{S}_1 & \bar{S}_1 & \cdots & \bar{S}_1 \\ \bar{S}_2 & \bar{S}_2 & \cdots & \bar{S}_2 \\ \bar{S}_3 & \bar{S}_3 & \cdots & \bar{S}_3 \end{pmatrix}$$

The resulting transformed image

$$T = \begin{pmatrix} u_{1,1} & u_{1,2} & u_{1,3} & \cdots & u_{r,r} \\ v_{1,1} & v_{1,2} & v_{1,3} & \cdots & v_{r,r} \\ w_{1,1} & w_{1,2} & w_{1,3} & \cdots & w_{r,r} \end{pmatrix}$$

consists of three new “colour” planes whose pixel values no longer have any correlation to the pixels at the same coordinates in another plane. [The bear disappeared from the last of these ( $w$ ), which represents mostly some of the green grass in the background.]

The previous example used the Karhunen–Loève transform in order to eliminate correlation between colour planes. While this is of some relevance for image compression, far more correlation can be found between neighbour pixels within each colour plane.

In order to exploit such correlation using the KLT, the sample set has to be extended from individual pixels to entire images. The underlying calculation is the same as in the preceding example, but this time the columns of  $S$  are entire (monochrome) images, namely the different images found in the set of test images that we use to examine typical correlations between neighbour pixels.

In other words, we use the same formulas as in the previous example, but this time  $n$  is the number of pixels per image and  $m$  is the number of sample images. The Karhunen–Loève transform is here no longer a rotation in a 3-dimensional colour space, but it operates now in a *much* larger vector space that has as many dimensions as an image has pixels.

To keep things simple, we look in the next experiment only at  $m = 9000$  1-dimensional “images” with  $n = 32$  pixels each. As a further simplification, we use not real images, but random noise that was filtered such that its amplitude spectrum is proportional to  $1/f$ , where  $f$  is the frequency. The result would be similar in a sufficiently large collection of real test images.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Entropy coding

Transform coding

Decorrelation

**Discrete Cosine Transform**

Colour spaces

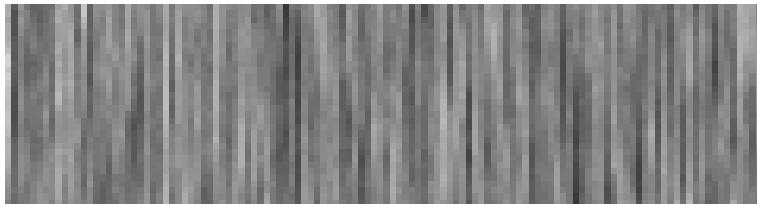
Quantization

JPEG

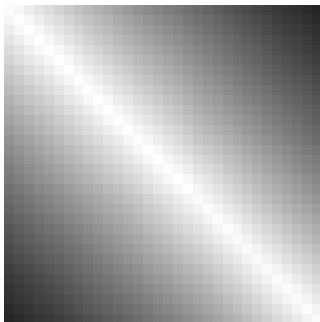
Outlook

# Karhunen–Loève transform example II

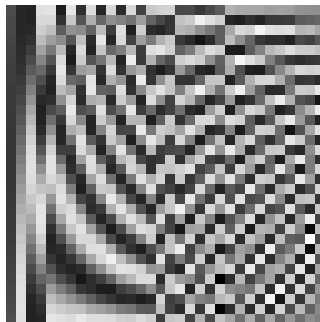
Matrix columns of  $S$  filled with samples of  $1/f$  filtered noise



Covariance matrix  $C$



Matrix  $U$  with eigenvector columns



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Entropy coding

Transform coding

Decorrelation

**Discrete Cosine Transform**

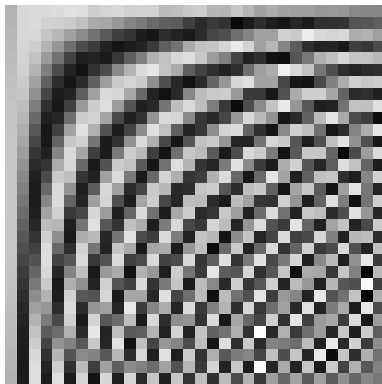
Colour spaces

Quantization

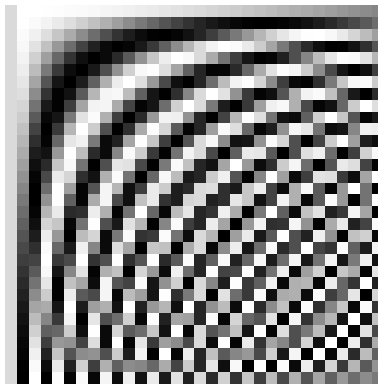
JPEG

Outlook

Matrix  $U'$  with normalised KLT  
eigenvector columns



Matrix with Discrete Cosine  
Transform base vector columns



**Breakthrough:** Ahmed/Natarajan/Rao discovered the DCT as an excellent approximation of the KLT for typical photographic images, but far more efficient to calculate.

Ahmed, Natarajan, Rao: Discrete Cosine Transform. IEEE Transactions on Computers, Vol. 23, January 1974, pp. 90–93.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Entropy coding

Transform coding

Decorrelation

**Discrete Cosine Transform**

Colour spaces

Quantization

JPEG

Outlook

The forward and inverse discrete cosine transform

$$S_u = \frac{C_u}{\sqrt{N/2}} \sum_{x=0}^{N-1} s_x \cos \frac{(2x+1)u\pi}{2N}$$
$$s_x = \sum_{u=0}^{N-1} \frac{C_u}{\sqrt{N/2}} S_u \cos \frac{(2x+1)u\pi}{2N}$$

with

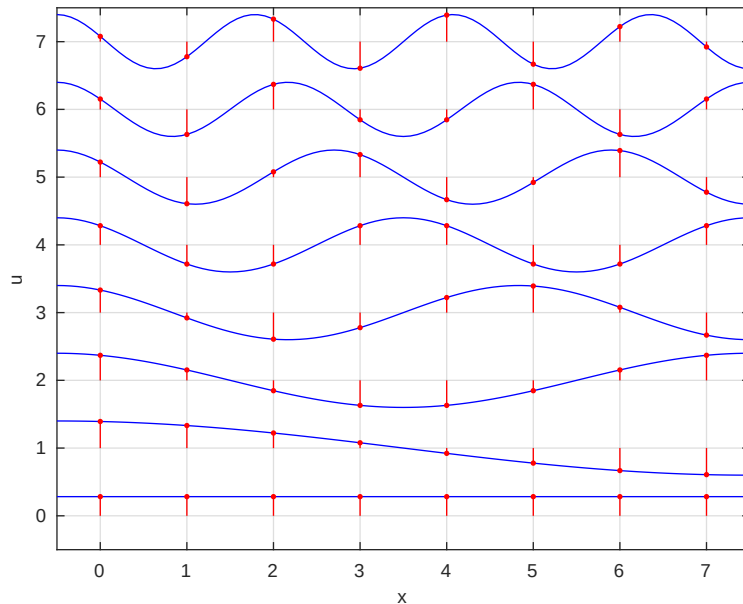
$$C_u = \begin{cases} \frac{1}{\sqrt{2}} & u = 0 \\ 1 & u > 0 \end{cases}$$

is an orthonormal transform:

$$\sum_{x=0}^{N-1} \frac{C_u}{\sqrt{N/2}} \cos \frac{(2x+1)u\pi}{2N} \cdot \frac{C_{u'}}{\sqrt{N/2}} \cos \frac{(2x+1)u'\pi}{2N} = \begin{cases} 1 & u = u' \\ 0 & u \neq u' \end{cases}$$

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Entropy coding](#)[Transform coding](#)[Decorrelation](#)[Discrete Cosine Transform](#)[Colour spaces](#)[Quantization](#)[JPEG](#)[Outlook](#)

## DCT base vectors for $N = 8$ :



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Entropy coding

Transform coding

Decorrelation

**Discrete Cosine Transform**

Colour spaces

Quantization

JPEG

Outlook

The 2-dimensional variant of the DCT applies the 1-D transform on both rows and columns of an image:

$$S_{u,v} = \frac{C_u}{\sqrt{N/2}} \frac{C_v}{\sqrt{N/2}} \cdot \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} s_{x,y} \cos \frac{(2x+1)u\pi}{2N} \cos \frac{(2y+1)v\pi}{2N}$$
$$s_{x,y} = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \frac{C_u}{\sqrt{N/2}} \frac{C_v}{\sqrt{N/2}} \cdot S_{u,v} \cos \frac{(2x+1)u\pi}{2N} \cos \frac{(2y+1)v\pi}{2N}$$

A range of fast algorithms have been found for calculating 1-D and 2-D DCTs (e.g., Ligtenberg/Vetterli).

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Entropy coding

Transform coding

Decorrelation

**Discrete Cosine Transform**

Colour spaces

Quantization

JPEG

Outlook

# Whole-image DCT

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Entropy coding

Transform coding

Decorrelation

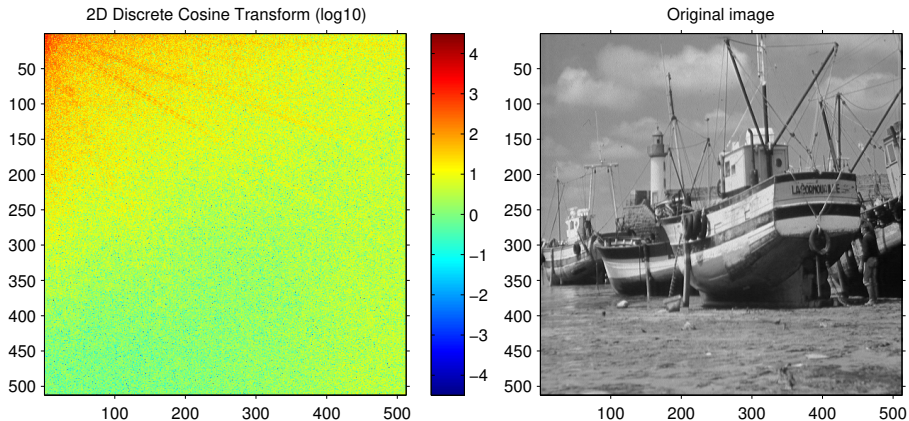
**Discrete Cosine Transform**

Colour spaces

Quantization

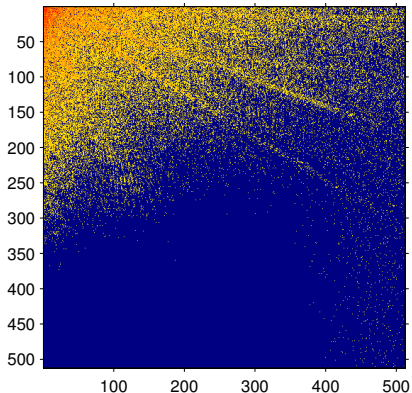
JPEG

Outlook

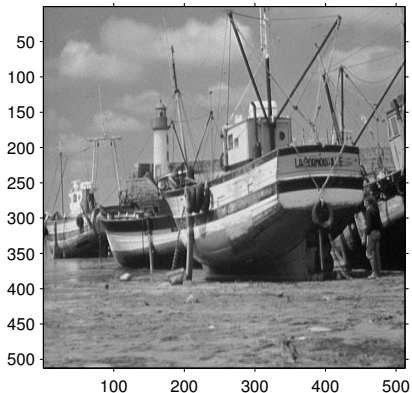


# Whole-image DCT, 80% coefficient cut-off

80% truncated 2D DCT (log10)



80% truncated DCT: reconstructed image



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Entropy coding

Transform coding

Decorrelation

**Discrete Cosine Transform**

Colour spaces

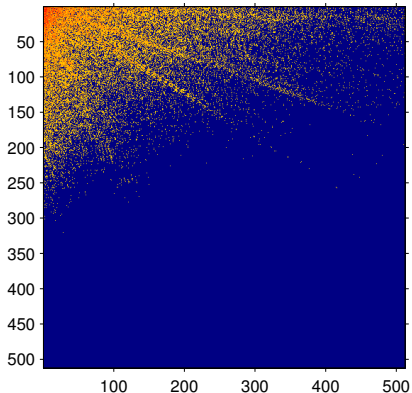
Quantization

JPEG

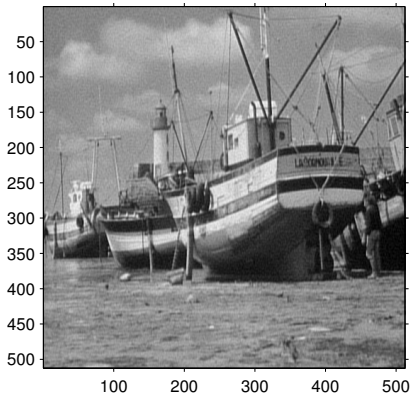
Outlook

# Whole-image DCT, 90% coefficient cut-off

90% truncated 2D DCT (log10)



90% truncated DCT: reconstructed image



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Entropy coding

Transform coding

Decorrelation

**Discrete Cosine Transform**

Colour spaces

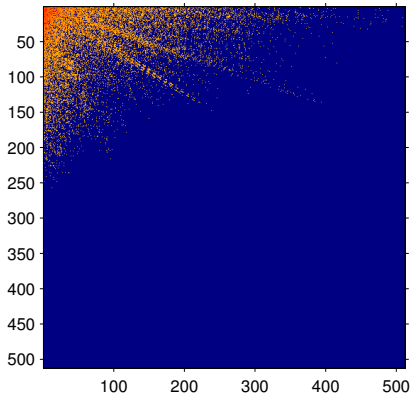
Quantization

JPEG

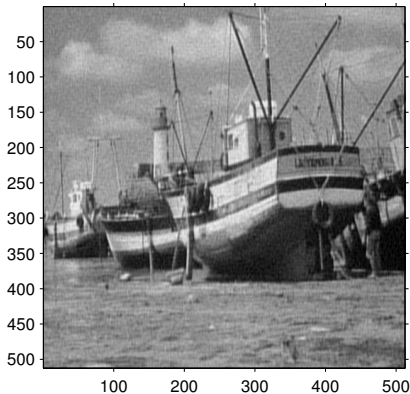
Outlook

# Whole-image DCT, 95% coefficient cut-off

95% truncated 2D DCT (log10)



95% truncated DCT: reconstructed image



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Entropy coding

Transform coding

Decorrelation

**Discrete Cosine Transform**

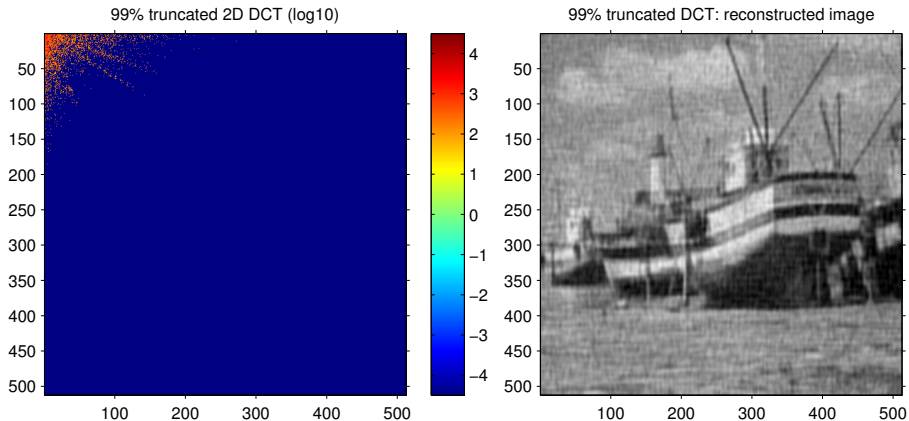
Colour spaces

Quantization

JPEG

Outlook

# Whole-image DCT, 99% coefficient cut-off



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Entropy coding

Transform coding

Decorrelation

**Discrete Cosine Transform**

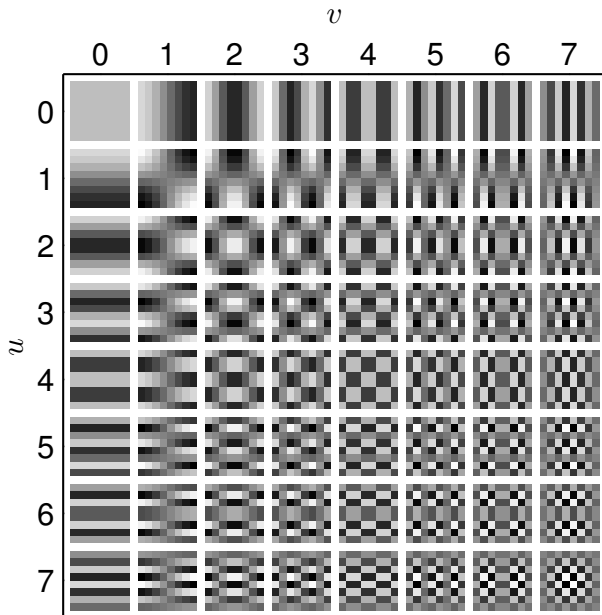
Colour spaces

Quantization

JPEG

Outlook

# Base vectors of 8×8 DCT



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Entropy coding

Transform coding

Decorrelation

**Discrete Cosine Transform**

Colour spaces

Quantization

JPEG

Outlook

Hardware interface (VGA): red, green, blue signals with 0–0.7 V

Electron-beam current and photon count of cathode-ray displays are roughly proportional to  $(v - v_0)^\gamma$ , where  $v$  is the video-interface or control-grid voltage and  $\gamma$  is a device parameter that is typically in the range 1.5–3.0. In broadcast TV, this CRT non-linearity is compensated in cameras (gamma compression,  $(\dots)^{1/\gamma}$ ). A welcome side effect is that it approximates Stevens' scale and therefore helps to reduce perceived noise.

Software interfaces map RGB voltage linearly to  $\{0, 1, \dots, 255\}$  or 0–1.

How numeric RGB values map to colour and luminosity can depend on the hardware, operating system or device driver.

The “sRGB” standard aims to standardize the meaning of an RGB value with the parameter  $\gamma = 2.2$  and with standard colour coordinates of the three primary colours.

<https://www.w3.org/Graphics/Color/sRGB>, IEC 61966-2-1 at <https://bsol.bsigroup.com/>

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Entropy coding

Transform coding

Decorrelation

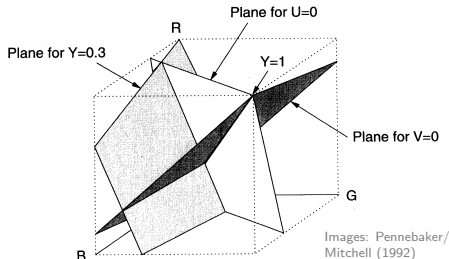
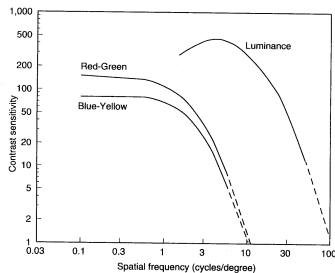
Discrete Cosine Transform

Colour spaces

Quantization

JPEG

Outlook



The human eye processes colour and luminosity at different resolutions. To exploit this phenomenon, many image transmission systems use a colour space with a “luminance” coordinate

$$Y = 0.3R + 0.6G + 0.1B$$

If based on gamma-compressed  $R'$ ,  $G'$ ,  $B'$  then  $Y' = 0.3R' + 0.6G' + 0.1B'$  is called “luma”.

The remaining “chrominance” colour information can be encoded as “chroma” coordinates  $U$  and  $V$ :

$$V = R' - Y' = 0.7R' - 0.6G' - 0.1B'$$

$$U = B' - Y' = -0.3R' - 0.6G' + 0.9B'$$

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Entropy coding

Transform coding

Decorrelation

Discrete Cosine Transform

Colour spaces

Quantization

JPEG

Outlook

# YUV transform example



original



Y channel



U and V channels

The centre image shows only the luminance channel as a black/white image. In the right image, the luminance channel (Y) was replaced with a constant, such that only the chrominance information remains.

This example and the next make only sense when viewed in colour. On a black/white printout of this slide, only the Y channel information will be present.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Entropy coding

Transform coding

Decorrelation

Discrete Cosine Transform

Colour spaces

Quantization

JPEG

Outlook

# Y versus UV sensitivity example



original



blurred U and V



blurred Y channel

In the centre image, the chrominance channels have been severely low-pass filtered (Gaussian impulse response ). But the human eye perceives this distortion as far less severe than if the exact same filtering is applied to the luminance channel (right image).

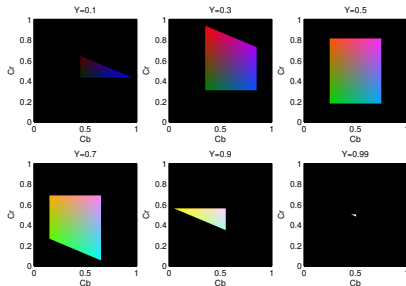
Photo courtesy of  
Karel de Gendre

- Introduction
- Sequences and systems
- Convolution
- Fourier transform
- Sampling
- Discrete Fourier transform
- Deconvolution
- Spectral estimation
- Digital filters
- IIR filters
- Random signals
- Digital communication
- Audiovisual data compression
  - Entropy coding
  - Transform coding
  - Decorrelation
  - Discrete Cosine Transform
- Colour spaces
- Quantization
- JPEG
- Outlook

Since  $-0.7 \leq V \leq 0.7$  and  $-0.9 \leq U \leq 0.9$ , a more convenient normalized encoding of chrominance is:

$$Cb = \frac{U}{2.0} + 0.5$$

$$Cr = \frac{V}{1.6} + 0.5$$



Many image-compression methods operate on  $Y'$ ,  $Cr$ ,  $Cb$  channels separately, using half the resolution of  $Y'$  for storing  $Cr$ ,  $Cb$ .

Some digital-television engineering terminology:

If each pixel is represented by its own  $Y'$ ,  $Cr$  and  $Cb$  byte, this is called a “4:4:4” format. In the more compact “4:2:2” format, a  $Cr$  and  $Cb$  value is transmitted only for every second pixel, reducing the horizontal chrominance resolution by a factor two. The “4:2:0” format transmits in alternating lines either  $Cr$  or  $Cb$  for every second pixel, thus halving the chrominance resolution both horizontally and vertically. The “4:1:1” format reduces the chrominance resolution horizontally to a quarter and “4:1:0” does so in both directions. [ITU-R BT.601]

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Entropy coding

Transform coding

Decorrelation

Discrete Cosine Transform

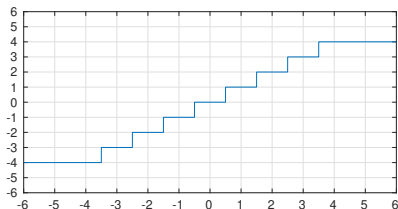
Colour spaces

Quantization

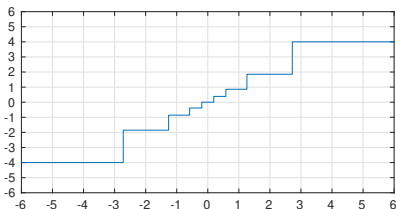
JPEG

Outlook

## Uniform/linear quantization:



## Non-uniform quantization:



Quantization is the mapping from a continuous or large set of values (e.g., analog voltage, floating-point number) to a smaller set of (typically  $2^8$ ,  $2^{10}$ ,  $2^{12}$ ,  $2^{14}$ ,  $2^{16}$ , or  $2^{24}$ ) values.

This introduces two types of error:

- ▶ the amplitude of *quantization noise* reaches up to half the maximum difference between neighbouring quantization levels
- ▶ *clipping* occurs where the input amplitude exceeds the value of the highest (or lowest) quantization level

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Entropy coding

Transform coding

Decorrelation

Discrete Cosine Transform

Colour spaces

Quantization

JPEG

Outlook

Example of a linear quantizer (resolution  $R$ , peak value  $V$ ):

$$y = \max \left\{ -V, \min \left\{ V, R \left\lfloor \frac{x}{R} + \frac{1}{2} \right\rfloor \right\} \right\}$$

Adding a noise signal that is uniformly distributed on  $[0, 1]$  instead of adding  $\frac{1}{2}$  helps to spread the frequency spectrum of the quantization noise more evenly. This is known as *dithering*.

Variant with even number of output values (no zero):

$$y = \max \left\{ -V, \min \left\{ V, R \left( \left\lfloor \frac{x}{R} \right\rfloor + \frac{1}{2} \right) \right\} \right\}$$

Improving the resolution by a factor of two (i.e., adding 1 bit) reduces the quantization noise by 6 dB.

Linearly quantized signals are easiest to process, but analog input levels need to be adjusted carefully to achieve a good tradeoff between the signal-to-quantization-noise ratio and the risk of clipping. Non-uniform quantization can reduce quantization noise where input values are not uniformly distributed and can approximate human perception limits.

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Entropy coding](#)[Transform coding](#)[Decorrelation](#)[Discrete Cosine Transform](#)[Colour spaces](#)[Quantization](#)[JPEG](#)[Outlook](#)

Rounding the logarithm of the signal amplitude makes the quantization error scale-invariant and is used where the signal level is not very predictable. Two alternative schemes are widely used to make the logarithm function odd and linearize it across zero before quantization:

$\mu$ -law:

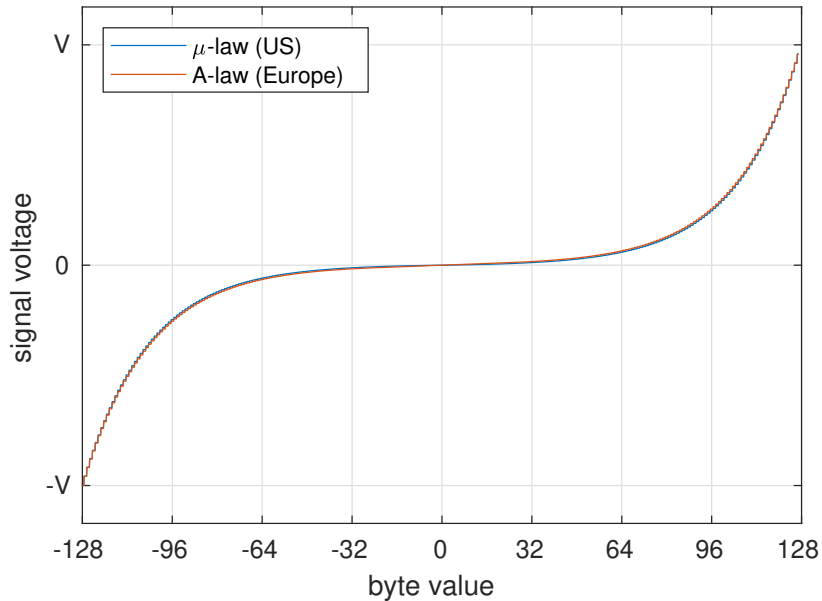
$$y = \frac{V \log(1 + \mu|x|/V)}{\log(1 + \mu)} \operatorname{sgn}(x) \quad \text{for } -V \leq x \leq V$$

A-law:

$$y = \begin{cases} \frac{A|x|}{1+\log A} \operatorname{sgn}(x) & \text{for } 0 \leq |x| \leq \frac{V}{A} \\ \frac{V(1+\log \frac{A|x|}{V})}{1+\log A} \operatorname{sgn}(x) & \text{for } \frac{V}{A} \leq |x| \leq V \end{cases}$$

European digital telephone networks use A-law quantization ( $A = 87.6$ ), North American ones use  $\mu$ -law ( $\mu = 255$ ), both with 8-bit resolution and 8 kHz sampling frequency (64 kbit/s). [ITU-T G.711]

[Introduction](#)[Sequences and systems](#)[Convolution](#)[Fourier transform](#)[Sampling](#)[Discrete Fourier transform](#)[Deconvolution](#)[Spectral estimation](#)[Digital filters](#)[IIR filters](#)[Random signals](#)[Digital communication](#)[Audiovisual data compression](#)[Entropy coding](#)[Transform coding](#)[Decorrelation](#)[Discrete Cosine Transform](#)[Colour spaces](#)[Quantization](#)[JPEG](#)[Outlook](#)



Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Entropy coding

Transform coding

Decorrelation

Discrete Cosine Transform

Colour spaces

**Quantization**

JPEG

Outlook

Working group “ISO/TC97/SC2/WG8 (Coded representation of picture and audio information)” was set up in 1982 by the International Organization for Standardization.

## Goals:

- ▶ continuous tone gray-scale and colour images
- ▶ recognizable images at 0.083 bit/pixel
- ▶ useful images at 0.25 bit/pixel
- ▶ excellent image quality at 0.75 bit/pixel
- ▶ indistinguishable images at 2.25 bit/pixel
- ▶ feasibility of 64 kbit/s (ISDN fax) compression with late 1980s hardware (16 MHz Intel 80386).
- ▶ workload equal for compression and decompression

The JPEG standard (ISO 10918) was finally published in 1994.

William B. Pennebaker, Joan L. Mitchell: JPEG still image compression standard. Van Nostrand Reinhold, New York, ISBN 0442012721, 1993.

Gregory K. Wallace: The JPEG Still Picture Compression Standard. Communications of the ACM 34(4)30–44, April 1991, <https://dl.acm.org/doi/10.1145/103085.103089>

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Entropy coding

Transform coding

Decorrelation

Discrete Cosine Transform

Colour spaces

Quantization

JPEG

Outlook

The most widely used lossy method from the JPEG standard:

- ▶ Colour component transform: 8-bit RGB  $\rightarrow$  8-bit  $Y'CrCb$
- ▶ Reduce resolution of  $Cr$  and  $Cb$  by a factor 2
- ▶ For the rest of the algorithm, process  $Y'$ ,  $Cr$  and  $Cb$  components independently (like separate gray-scale images)

The above steps are obviously skipped where the input is a gray-scale image.

- ▶ Split each image component into  $8 \times 8$  pixel blocks

Partial blocks at the right/bottom margin may have to be padded by repeating the last column/row until a multiple of eight is reached. The decoder will remove these padding pixels.

- ▶ Apply the  $8 \times 8$  forward DCT on each block

On unsigned 8-bit input, the resulting DCT coefficients will be signed 11-bit integers.

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Entropy coding

Transform coding

Decorrelation

Discrete Cosine Transform

Colour spaces

Quantization

JPEG

Outlook

- Quantization: divide each DCT coefficient by the corresponding value from an  $8 \times 8$  table, then round to the nearest integer:

The two standard quantization-matrix examples for luminance and chrominance are:

|    |    |    |    |     |     |     |     |    |    |    |    |    |    |    |    |
|----|----|----|----|-----|-----|-----|-----|----|----|----|----|----|----|----|----|
| 16 | 11 | 10 | 16 | 24  | 40  | 51  | 61  | 17 | 18 | 24 | 47 | 99 | 99 | 99 | 99 |
| 12 | 12 | 14 | 19 | 26  | 58  | 60  | 55  | 18 | 21 | 26 | 66 | 99 | 99 | 99 | 99 |
| 14 | 13 | 16 | 24 | 40  | 57  | 69  | 56  | 24 | 26 | 56 | 99 | 99 | 99 | 99 | 99 |
| 14 | 17 | 22 | 29 | 51  | 87  | 80  | 62  | 47 | 66 | 99 | 99 | 99 | 99 | 99 | 99 |
| 18 | 22 | 37 | 56 | 68  | 109 | 103 | 77  | 99 | 99 | 99 | 99 | 99 | 99 | 99 | 99 |
| 24 | 35 | 55 | 64 | 81  | 104 | 113 | 92  | 99 | 99 | 99 | 99 | 99 | 99 | 99 | 99 |
| 49 | 64 | 78 | 87 | 103 | 121 | 120 | 101 | 99 | 99 | 99 | 99 | 99 | 99 | 99 | 99 |
| 72 | 92 | 95 | 98 | 112 | 100 | 103 | 99  | 99 | 99 | 99 | 99 | 99 | 99 | 99 | 99 |

- apply DPCM coding to quantized DC coefficients from DCT
- read remaining quantized values from DCT in zigzag pattern
- locate sequences of zero coefficients (run-length coding)
- apply Huffman coding on zero run-lengths and magnitude of AC values
- add standard header with compression parameters

<https://jpeg.org/>

Example implementation: <https://www.ijg.org/>

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Entropy coding

Transform coding

Decorrelation

Discrete Cosine Transform

Colour spaces

Quantization

JPEG

Outlook

Further topics that we have not covered in this brief introductory tour through DSP, but for the understanding of which you should now have a good theoretical foundation:

- ▶ multirate systems
- ▶ effects of rounding errors
- ▶ adaptive filters
- ▶ DSP hardware architectures
- ▶ sound effects

If you find any typo or mistake in these lecture notes, please email [Markus.Kuhn@cl.cam.ac.uk](mailto:Markus.Kuhn@cl.cam.ac.uk).

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook

## Some final thoughts about redundancy ...

Aoccdrnig to rsceearh at Cmabrigde Uinervtisy, it deosn't mtt aer in waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is taht the frist and lsat ltteer be at the rghit pclae. The rset can be a total mses and you can sitll raed it wouthit porbelm. Tihs is bcuseae the huamn mnid deos not raed ervey lteter by istlef, but the wrod as a wlohe.

## ... and perception

Count how many Fs there are in this text:

FINISHED FILES ARE THE RE-  
SULT OF YEARS OF SCIENTIF-  
IC STUDY COMBINED WITH THE  
EXPERIENCE OF YEARS

Introduction

Sequences and systems

Convolution

Fourier transform

Sampling

Discrete Fourier transform

Deconvolution

Spectral estimation

Digital filters

IIR filters

Random signals

Digital communication

Audiovisual data compression

Outlook