Signals

► 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.).
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
Analog electronics

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

\[
\frac{U_{in} - U_{out}}{R} = \frac{1}{L} \int_{-\infty}^{t} U_{out} \, d\tau + C \frac{dU_{out}}{dt}
\]
Digital signal processing

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
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
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 in MATLAB
- apply transforms that reduce correlation between several signal sources
- understand the basic principles of several widely-used modulation and image-coding techniques.
▶ K. Steiglitz: *A digital signal processing primer – with applications to digital audio and computer music*. Addison-Wesley, 1996. (£67)
Outline

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

\[
\ldots, x_{-2}, x_{-1}, x_0, x_1, x_2, \ldots
\]

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 \( \{ y_n \} = T \{ x_n \} \):

\[
\ldots, x_2, x_1, x_0, x_{-1}, \ldots \xrightarrow{\text{discrete system } T} \ldots, y_2, y_1, y_0, y_{-1}, \ldots
\]
Some simple sequences

Unit-step sequence:

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

Impulse sequence:

\[ \delta_n = \begin{cases} 
1, & n = 0 \\
0, & n \neq 0 
\end{cases} \]

\[ = u_n - u_{n-1} \]
Sinusoidal sequences

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

$$x(t) = A \cdot \cos \left( 2\pi ft + \varphi \right)$$

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

$$x_n = A \cdot \cos \left( 2\pi fn/f_s + \varphi \right) = A \cdot \cos (\omega n + \varphi)$$

where $\omega = 2\pi f / f_s$ is the frequency expressed in radians per sample.

MATLAB/Octave example:

```matlab
n=0:40; fs=8000;
f=400; x=cos(2*pi*f*n/fs);
stem(n, x); ylim([-1.1 1.1])
```

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

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

A sequence \( \{x_n\} \) is

\[
\text{periodic} \iff \exists k > 0 : \forall n \in \mathbb{Z} : x_n = x_{n+k}
\]

Is a continuous function with period \( t_p \) still periodic after sampling?

\[
\begin{align*}
\text{absolutely summable} & \iff \sum_{n=-\infty}^{\infty} |x_n| < \infty \\
\text{square summable} & \iff \sum_{n=-\infty}^{\infty} |x_n|^2 < \infty \iff \text{“energy signal”} \\
0 & < \lim_{k \to \infty} \frac{1}{1 + 2k} \sum_{n=-k}^{k} |x_n|^2 < \infty \iff \text{“power signal”}
\end{align*}
\]

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.
Properties of sequences

A sequence \( \{x_n\} \) is

\[
\text{periodic} \iff \exists k > 0 : \forall n \in \mathbb{Z} : x_n = x_{n+k}
\]

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

- \( \text{absolutely summable} \iff \sum_{n=-\infty}^{\infty} |x_n| < \infty \)

- \( \text{square summable} \iff \sum_{n=-\infty}^{\infty} |x_n|^2 < \infty \iff \text{“energy signal”} \)

\[
0 < \lim_{k \to \infty} \frac{1}{1 + 2k} \sum_{n=-k}^{k} |x_n|^2 < \infty \iff \text{“power signal”}
\]

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.
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 $[t_1, t_2]$

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

AC component = $f(t) -$ 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:

$$\sqrt{\frac{1}{2\pi k} \int_{0}^{2\pi k} [A \cdot \sin(\tau + \phi)]^2 \, d\tau} = \frac{A}{\sqrt{2}} \quad \text{for all } k \in \mathbb{N}, A, \phi \in \mathbb{R}$$
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:

$$\psi = \log_c \frac{\phi}{\phi_0}$$
Fechner’s scale matches older subjective intensity scales that follow differentiability of stimuli, e.g. the astronomical magnitude numbers for star brightness introduced by Hipparchos (≈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.
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 \( P = U^2/R = I^2R \) 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.
Decibel

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”:

- $0 \text{ dBW} = 1 \text{ W}$
- $0 \text{ dBm} = 1 \text{ mW} = -30 \text{ dBW}$
- $0 \text{ dB} \mu\text{V} = 1 \mu\text{V}$
- $0 \text{ dB} \text{SPL} = 20 \mu\text{Pa}$ (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:

- $3 \text{ dB} = 2 \times \text{power}$
- $6 \text{ dB} = 2 \times \text{voltage/pressure/etc.}$
- $10 \text{ dB} = 10 \times \text{power}$
- $20 \text{ dB} = 10 \times \text{voltage/pressure/etc.}$

Types of discrete systems

\[
\ldots, x_2, x_1, x_0, x_{-1}, \ldots \rightarrow \text{discrete system } T \rightarrow \ldots, y_2, y_1, y_0, y_{-1}, \ldots
\]

A causal system cannot look into the future:

\[
y_n = f(x_n, x_{n-1}, x_{n-2}, \ldots)
\]

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 \(d\)

\[
\{y_n\} = T\{x_n\} \quad \iff \quad \{y_{n-d}\} = T\{x_{n-d}\}.
\]

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

\[
T\{a \cdot x_n + b \cdot x'_n\} = a \cdot T\{x_n\} + b \cdot T\{x'_n\}.
\]
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 $M = 4$:
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 \( \alpha = \frac{1}{2} \):
Example: accumulator system

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

It is causal, linear, time-invariant, with memory.
Example: backward difference system

\[ y_n = x_n - x_{n-1} \]

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

Time-invariant non-linear memory-less systems:

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

Linear but not time-invariant systems:

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

\[ y_n = x_{\lfloor n/4 \rfloor} \]

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

Linear time-invariant non-causal systems:

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

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

**Addition:**

**Multiplication by constant:**

**Delay:**

The \( a_k \) and \( b_m \) are constant coefficients.
or

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

or the combination of both:

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

The MATLAB function \texttt{filter} is an efficient implementation of the last variant.
Outline

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
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

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

If \( \{y_n\} = \{a_n\} \ast \{x_n\} \) 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 \( \{a_n\} = T\{\delta_n\} \).
Convolution examples

Convolution examples

A B C D E F A ∗ B A ∗ C C ∗ A A ∗ E D ∗ E A ∗ F
Properties of convolution

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

- Convolution is associative
  \[
  (\{p_n\} \ast \{q_n\}) \ast \{r_n\} = \{p_n\} \ast (\{q_n\} \ast \{r_n\})
  \]

- Convolution is commutative
  \[
  \{p_n\} \ast \{q_n\} = \{q_n\} \ast \{p_n\}
  \]

- Convolution is linear
  \[
  \{p_n\} \ast \{a \cdot q_n + b \cdot r_n\} = a \cdot (\{p_n\} \ast \{q_n\}) + b \cdot (\{p_n\} \ast \{r_n\})
  \]

- The impulse sequence (slide 10) is neutral under convolution
  \[
  \{p_n\} \ast \{\delta_n\} = \{\delta_n\} \ast \{p_n\} = \{p_n\}
  \]

- Sequence shifting is equivalent to convolving with a shifted impulse
  \[
  \{p_{n-d}\} = \{p_n\} \ast \{\delta_{n-d}\}
  \]
Proof: all LTI systems just apply convolution

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\((*)\) time-invariant\((**)\) system \(T\) to such a decomposed sequence:

\[
T\{x_n\} = T\left( \sum_{k=-\infty}^{\infty} x_k \cdot \{\delta_{n-k}\} \right) \stackrel{(*)}{=} \sum_{k=-\infty}^{\infty} x_k \cdot T\{\delta_{n-k}\}
\]

\[
\stackrel{(**)}{=} \sum_{k=-\infty}^{\infty} x_k \cdot \{\delta_{n-k}\} * T\{\delta_n\} = \left( \sum_{k=-\infty}^{\infty} x_k \cdot \{\delta_{n-k}\} \right) * T\{\delta_n\}
\]

\[
= \{x_n\} * T\{\delta_n\} \quad \text{q.e.d.}
\]

\(\Rightarrow\) The impulse response \(T\{\delta_n\}\) fully characterizes an LTI system.
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).
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, $r = \frac{as}{2f}$):

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

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

$$B(x, y) = \int\int I(x-x', y-y') \cdot h(x', y') \cdot dx' \, dy'$$
Any passive network (resistors, capacitors, inductors) convolves its input voltage $U_{\text{in}}$ with an *impulse response function* $h$, leading to $U_{\text{out}} = U_{\text{in}} * h$, that is

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

In the above example:

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

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
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?
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 \( t = 0 \).
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.
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.

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

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

$$A \cdot \sin(\omega t + \varphi) = x \cdot \sin(\omega t) + y \cdot \cos(\omega t)$$

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 = $\sin(\omega)$] and $90^\circ$ [height = $\cos(\omega)$].

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

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

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

$$f_1(t) + f_2(t) = (x_1 + x_2) \cdot \sin(\omega) + (y_1 + y_2) \cdot \cos(\omega).$$
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 \( \{y_n\} \) is nothing but adding together scaled and delayed copies of \( \{x_n\} \).
(Think of \( \{y_n\} \) decomposed into a sum of impulses.)
If \( \{x_n\} \) is a sampled sine wave of frequency \( f \), so is \( \{x_n\} \ast \{y_n\} \).

The same applies for continuous sine waves and convolution.

2) Sine waves are orthogonal to each other

\[
\int_{-\infty}^{\infty} \sin(\omega_1 t + \varphi_1) \cdot \sin(\omega_2 t + \varphi_2) \, dt \quad "=" \quad 0
\]

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

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

The term “orthogonal” is used here in the context of an (infinitely dimensional) vector space, where the “vectors” are functions of the form \( f : \mathbb{R} \to \mathbb{R} \) (or \( f : \mathbb{R} \to \mathbb{C} \)) and the scalar product is defined as \( f \cdot g = \int_{-\infty}^{\infty} f(t) \cdot g(t) \, dt \).
\[ \sin(2t) \cdot \sin(3t) \]
\[ \sin(2t) \cdot \sin(4t) \]
$\sin(t) \cdot \cos(t)$
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:

\[
A_1 \cdot z^{t+\varphi_1} + A_2 \cdot z^{t+\varphi_2} = A_1 \cdot z^t \cdot z^{\varphi_1} + A_2 \cdot z^t \cdot z^{\varphi_2} \\
= (A_1 \cdot z^{\varphi_1} + A_2 \cdot z^{\varphi_2}) \cdot z^t = A \cdot z^t
\]

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

\[
\ldots, z^{-3}, z^{-2}, z^{-1}, 1, z, z^2, z^3, \ldots
\]

\( x_n = z^n \) with an arbitrary sequence \( \{h_n\} \), we get \( \{y_n\} = \{z^n\} \ast \{h_n\} \),

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

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

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

The same applies in the continuous case.
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:

\[
\begin{align*}
\cos(\theta) &= \frac{1}{2} \left( e^{j\theta} + e^{-j\theta} \right) \\
\sin(\theta) &= \frac{1}{2j} \left( e^{j\theta} - e^{-j\theta} \right)
\end{align*}
\]

or

\[
\cos(\omega t + \varphi) = \frac{1}{2} \left( e^{j(\omega t + \varphi)} + e^{-j(\omega t + \varphi)} \right)
\]

or

\[
\begin{align*}
\cos(\omega n + \varphi) &= \Re(e^{j(\omega n + \varphi)}) = \Re[(e^{j\omega})^n \cdot e^{j\varphi}] \\
\sin(\omega n + \varphi) &= \Im(e^{j(\omega n + \varphi)}) = \Im[(e^{j\omega})^n \cdot e^{j\varphi}]
\end{align*}
\]

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 \( x \in \mathbb{C} \).
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:

\[
\begin{pmatrix}
  x_3 \\
  y_3
\end{pmatrix}
= \begin{pmatrix}
  x_2 & -y_2 \\
  y_2 & x_2
\end{pmatrix} \cdot \begin{pmatrix}
  x_1 \\
  y_1
\end{pmatrix}
= \begin{pmatrix}
  x_1x_2 - y_1y_2 \\
  x_1y_2 + x_2y_1
\end{pmatrix} = (-y_2, x_2)
\]

\[z_1 = x_1 + jy_1, \quad z_2 = x_2 + jy_2\]

\[z_1 \cdot z_2 = x_1x_2 - y_1y_2 + j(x_1y_2 + x_2y_1)\]
Complex phasors

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 37, where the argument of \(H\) is the base, we would write \(H(e^{j\omega})\).
We define the Fourier integral transform and its inverse as

\[
\mathcal{F}\{g(t)\}(f) = G(f) = \int_{-\infty}^{\infty} g(t) \cdot e^{-2\pi jft} \, dt
\]

\[
\mathcal{F}^{-1}\{G(f)\}(t) = g(t) = \int_{-\infty}^{\infty} G(f) \cdot e^{2\pi jft} \, df
\]

Many equivalent forms of the Fourier transform are used in the literature. There is no strong consensus on whether the forward transform uses \(e^{-2\pi jft}\) and the backwards transform \(e^{2\pi jft}\), 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.
Properties of the Fourier transform

If

\[ x(t) \quad \rightarrow \quad X(f) \quad \text{and} \quad y(t) \quad \rightarrow \quad Y(f) \]

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

Linearity:

\[ ax(t) + by(t) \quad \rightarrow \quad aX(f) + bY(f) \]

Time scaling:

\[ x(at) \quad \rightarrow \quad \frac{1}{|a|} X \left( \frac{f}{a} \right) \]

Frequency scaling:

\[ \frac{1}{|a|} x \left( \frac{t}{a} \right) \quad \rightarrow \quad X(af) \]
Time shifting:

\[ x(t - \Delta t) \longrightarrow X(f) \cdot e^{-2\pi j f \Delta t} \]

Frequency shifting:

\[ x(t) \cdot e^{2\pi j f \Delta t} \longrightarrow X(f - \Delta f) \]

Parseval’s theorem (total energy):

\[
\int_{-\infty}^{\infty} |x(t)|^2 dt = \int_{-\infty}^{\infty} |X(f)|^2 df
\]
Fourier transform example: rect and sinc

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”

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

and vice versa

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

Some noteworthy properties of these functions:

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

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

\[ \mathcal{F}\{(f \ast g)(t)\} = \mathcal{F}\{f(t)\} \cdot \mathcal{F}\{g(t)\} \]

Proof:

\[
\begin{align*}
z(r) &= \int_s x(s)y(r-s)ds \\
&\iff \int_r z(r)e^{-j\omega r}dr = \int_r \int_s x(s)y(r-s)e^{-j\omega r}dsdr = \\
&\int_s x(s)\int_r y(r-s)e^{-j\omega r}dr ds = \int_s x(s)e^{-j\omega s} \int_r y(r-s)e^{-j\omega(r-s)}dr ds \\
&= \int_s x(s)e^{-j\omega s} \int_t y(t)e^{-j\omega t} dt ds = \int_s x(s)e^{-j\omega s} ds \cdot \int_t y(t)e^{-j\omega t} dt.
\end{align*}
\]

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

\[ \mathcal{F}\{f(t) \cdot g(t)\} = \mathcal{F}\{f(t)\} \ast \mathcal{F}\{g(t)\} \]

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 52).

The proof is very similar to the one above.

Both equally work for the inverse Fourier transform:

\[
\begin{align*}
\mathcal{F}^{-1}\{(F \ast G)(f)\} &= \mathcal{F}^{-1}\{F(f)\} \cdot \mathcal{F}^{-1}\{G(f)\} \\
\mathcal{F}^{-1}\{F(f) \cdot G(f)\} &= \mathcal{F}^{-1}\{F(f)\} \ast \mathcal{F}^{-1}\{G(f)\}
\end{align*}
\]
The continuous equivalent of the impulse sequence $\{\delta_n\}$ is known as the Dirac delta function $\delta(x)$. It is a generalized function, defined such that

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

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

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

or

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

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

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

\[ \mathcal{F}^{-1}\{1\}(t) = \int_{-\infty}^{\infty} 1 \cdot e^{2\pi j ft} \, df \quad = \delta(t) \]
The Fourier transform of 1 follows from the Dirac delta's ability to sample inside an integral:

\[ g(t) = \mathcal{F}^{-1}(\mathcal{F}(g))(t) \]

\[ = \int_{-\infty}^{\infty} \left( \int_{-\infty}^{\infty} g(s) \cdot e^{-2\pi jfs} \cdot ds \right) \cdot e^{2\pi jft} \cdot df \]

\[ = \int_{-\infty}^{\infty} \left( \int_{-\infty}^{\infty} e^{-2\pi jfs} \cdot e^{2\pi jft} \cdot df \right) \cdot g(s) \cdot ds \]

\[ = \int_{-\infty}^{\infty} \left( \int_{-\infty}^{\infty} e^{-2\pi jf(t-s)} \cdot df \right) \cdot g(s) \cdot ds \]

So if \( \delta \) has the property

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

then

\[ \int_{-\infty}^{\infty} e^{-2\pi jf(t-s)} df = \delta(t - s) \]
\[ \int_{-\infty}^{\infty} e^{2\pi j ft} df = \delta(t) \quad \sum_{i=1}^{10} \cos(2\pi f_i t) \approx \delta(t) \]

\[ f_1, \ldots, f_{10} \in [0, 3] \text{ chosen uniformly at random} \]
\[ \int_{-\infty}^{\infty} e^{2\pi j f} \, df = \delta(t) \]
\[ \sum_{i=1}^{100} \cos(2\pi f_i t) \approx \delta(t) \]
\[ f_1, \ldots, f_{100} \in [0, 10] \text{ chosen uniformly at random} \]
\[ \sum_{n=-\infty}^{\infty} e^{\pm 2\pi jnt} = \sum_{n=-\infty}^{\infty} \delta(t - n) \]
\[ \sum_{n=1}^{5} \cos(2\pi nt) \approx \sum_{n=-\infty}^{\infty} \delta(t - n) \]
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).
Fourier transform symmetries

We call a function \( x(t) \)

odd if \( x(-t) = -x(t) \)
even if \( x(-t) = x(t) \)

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

Then

\[
\begin{align*}
\text{\( x(t) \) is real} & \quad \iff \quad X(-f) = [X(f)]^* \\
\text{\( x(t) \) is imaginary} & \quad \iff \quad X(-f) = -[X(f)]^* \\
\text{\( x(t) \) is even} & \quad \iff \quad X(f) \text{ is even} \\
\text{\( x(t) \) is odd} & \quad \iff \quad X(f) \text{ is odd} \\
\text{\( x(t) \) is real and even} & \quad \iff \quad X(f) \text{ is real and even} \\
\text{\( x(t) \) is real and odd} & \quad \iff \quad X(f) \text{ is imaginary and odd} \\
\text{\( x(t) \) is imaginary and even} & \quad \iff \quad X(f) \text{ is imaginary and even} \\
\text{\( x(t) \) is imaginary and odd} & \quad \iff \quad X(f) \text{ is real and odd}
\end{align*}
\]
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?
Outline

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
Sampling using a Dirac comb

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.
The Fourier transform of a Dirac comb

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

is another Dirac comb

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

\[ t_s \cdot \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \delta(t - t_s n) e^{-2\pi jft} dt = \sum_{n=-\infty}^{\infty} \delta \left( f - \frac{n}{t_s} \right). \]
Sampled at frequency $f_s$, the function $\cos(2\pi tf)$ cannot be distinguished from $\cos[2\pi t(k \cdot f_s \pm f)]$ for any $k \in \mathbb{Z}$.
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”.
The Fourier transform of a sampled signal

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

is

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

The inverse transform is

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

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

\[ X(e^{j\dot{\omega}}) = \sum_{n} x_n \cdot e^{-j\dot{\omega}n} \]

is customary, to highlight both the periodicity of the DTFT and its relationship with the \( z \)-transform of \( \{x_n\} \) (see slide 117).
Properties of the DTFT

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:**

\[ \{x_n\} * \{y_n\} = \{z_n\} \iff X(e^{j\omega}) \cdot Y(e^{j\omega}) = Z(e^{j\omega}) \]

and

\[ x_n \cdot y_n = z_n \iff \int_{-\pi}^{\pi} X(e^{j\omega'}) \cdot Y(e^{j(\omega-\omega')}) d\omega' = Z(e^{j\omega}) \]
Nyquist limit and anti-aliasing filters

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”.
Nyquist limit and anti-aliasing filters

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

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

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:

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

Note that sampling $h(t)$ gives the impulse function: $h(t) \cdot s(t) = \delta(t)$. 
Impulse response of ideal low-pass filter with cut-off frequency $f_s/2$: 

\[ t \cdot f_s \]
Reconstruction filter example

sampled signal
interpolation result
scaled/shifted sin(x)/x pulses
If before being sampled with \( x_n = x(t/f_s) \) the signal \( x(t) \) satisfied the Nyquist limit

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

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

\[
x(t) = \int_{-\infty}^{\infty} h(s) \cdot \hat{x}(t - s) \cdot ds
\]

\[
= \int_{-\infty}^{\infty} \frac{1}{t_s} \text{sinc}\left(\frac{s}{t_s}\right) \cdot t_s \sum_{n=-\infty}^{\infty} x_n \cdot \delta(t - s - t_s \cdot n) \cdot ds
\]

\[
= \sum_{n=-\infty}^{\infty} x_n \cdot \int_{-\infty}^{\infty} \text{sinc}\left(\frac{s}{t_s}\right) \cdot \delta(t - s - t_s \cdot n) \cdot ds
\]

\[
= \sum_{n=-\infty}^{\infty} x_n \cdot \text{sinc}\left(\frac{t - t_s \cdot n}{t_s}\right) = \sum_{n=-\infty}^{\infty} x_n \cdot \text{sinc}(t/t_s - n)
\]

\[
= \sum_{n=-\infty}^{\infty} x_n \cdot \frac{\sin \pi(t/t_s - n)}{\pi(t/t_s - n)}
\]
Reconstruction filters

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.
Band-pass signal sampling

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:

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

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
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.

$$x(t) = \dot{x}(t) * p(t)$$

```
\begin{align*}
X(f) &= \dot{X}(f) \\
\dot{X}(f) &= P(f)
\end{align*}
```
A signal $x(t)$ that is sampled with frequency $f_s$ has a spectrum that is periodic with a period of $f_s$. 

$$x(t)$$ 

$$X(f)$$ 

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

$$X(f) * S(f) = \hat{X}(f)$$
Continuous vs discrete Fourier transform

- 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 \( n \).
If \( x(t) \) has period \( t_p = n \cdot t_s \), then after sampling it at rate \( t_s \) we have

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

and the Fourier transform of that is

\[
\mathcal{F}\{\ddot{x}(t)\}(f) = \ddot{X}(f) = \int_{-\infty}^{\infty} \ddot{x}(t) \cdot e^{-2\pi j ft} dt
\]

\[
= t_s \cdot \sum_{l=-\infty}^{\infty} \sum_{i=0}^{n-1} x_i \cdot e^{-2\pi j \frac{f}{f_s} \cdot (i+n l)} = t_s \cdot \sum_{l=-\infty}^{\infty} e^{-2\pi j \frac{f}{f_s} \cdot n l} \cdot \sum_{i=0}^{n-1} x_i \cdot e^{-2\pi j \frac{f}{f_s} \cdot i}
\]

\[
= \frac{1}{t_s n} \sum_l \delta(f - \frac{l}{n} f_s)
\]

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 \( f = kf_p \)

\[
\ddot{X}(kf_p) = \frac{1}{n} \cdot \sum_{l=-\infty}^{\infty} \delta(k f_p - l f_p) \cdot \sum_{i=0}^{n-1} x_i \cdot e^{-2\pi j \frac{k i}{n}}
\]

\[
= \begin{cases} 
\delta(0) & \text{if } k \in \mathbb{Z} \\
0 & \text{if } k \notin \mathbb{Z}
\end{cases}
\]

Show that \( X_k = X_{k \pm n} \) for all \( k \in \mathbb{Z} \).
Discrete Fourier Transform (DFT)

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

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

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

\[ F_n \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{n-1} \end{pmatrix} = \begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ \vdots \\ X_{n-1} \end{pmatrix}, \quad \frac{1}{n} \cdot F_n^* \cdot \begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ \vdots \\ X_{n-1} \end{pmatrix} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{n-1} \end{pmatrix} \]
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 $0, f_s/n, 2f_s/n, 3f_s/n, \ldots, f_s/2$, 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 $f_s/2$?
Inverse DFT visualized
Fast Fourier Transform (FFT)

\[
\left( F_n \{ x_i \}_{i=0}^{n-1} \right)_k = \sum_{i=0}^{n-1} x_i \cdot e^{-2\pi j \frac{ik}{n}}
\]

\[
= \sum_{i=0}^{\frac{n}{2}-1} x_{2i} \cdot e^{-2\pi j \frac{ik}{n/2}} + e^{-2\pi j \frac{k}{n}} \sum_{i=0}^{\frac{n}{2}-1} x_{2i+1} \cdot e^{-2\pi j \frac{ik}{n/2}}
\]

\[
= \begin{cases} 
(F_{\frac{n}{2}} \{ x_{2i} \}_{i=0}^{\frac{n}{2}-1})_k + e^{-2\pi j \frac{k}{n}} \cdot (F_{\frac{n}{2}} \{ x_{2i+1} \}_{i=0}^{\frac{n}{2}-1})_k, & k < \frac{n}{2} \\
(F_{\frac{n}{2}} \{ x_{2i} \}_{i=0}^{\frac{n}{2}-1})_{k-\frac{n}{2}} + e^{-2\pi j \frac{k}{n}} \cdot (F_{\frac{n}{2}} \{ x_{2i+1} \}_{i=0}^{\frac{n}{2}-1})_{k-\frac{n}{2}}, & k \geq \frac{n}{2}
\end{cases}
\]

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 http://www.fftw.org/.
Efficient real-valued FFT

The symmetry properties of the Fourier transform applied to the discrete Fourier transform $\{X_i\}_{i=0}^{n-1} = F_n \{x_i\}_{i=0}^{n-1}$ 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, allows us to calculate two real-valued $n$-point DFTs

$$\{X_i'\}_{i=0}^{n-1} = F_n \{x_i'\}_{i=0}^{n-1} \quad \{X_i''\}_{i=0}^{n-1} = F_n \{x_i''\}_{i=0}^{n-1}$$

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-value FFTs that occur in the first round.
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

\[(a + jb) \cdot (c + jd) = (\alpha - \beta) + j(\alpha + \gamma) \quad \text{with} \quad \alpha = a(c + d), \quad \beta = d(a + b), \quad \gamma = c(b - a)\]

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.
FFT-based convolution

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

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

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

\[
F^{-1}[F(A) \cdot F(B)]
\]

\[
F^{-1}[F(A') \cdot F(B')]
\]

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

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
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) $u(t)$?
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:

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

$$\tilde{u} = \mathcal{F}^{-1}\{W \cdot \mathcal{F}\{c\}/\mathcal{F}\{h\}\}$$
Exercise 13: Use MATLAB to deconvolve the blurred stars from slide 31. The files stars-blurred.png with the blurred-stars image and stars-psf.png with the impulse response (point-spread function) are available on the course-material web page. You may find the MATLAB functions imread, double, imagesc, circshift, fft2, ifft2 of use.

Try different ways to control the noise (slide 87) and distortions near the margins (windowing). [The MATLAB image processing toolbox provides ready-made “professional” functions deconvwnr, deconvreg, deconvlucy, edgetaper, for such tasks. Do not use these, except perhaps to compare their outputs with the results of your own attempts.]
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
[w,fs, bits] = auread('sing.au');
specgram(w,2048,fs);
ylim([0 8e3]); xlim([0 4.5]);
saveas(gcf, 'sing.eps', 'eps2c');
Different vowels at constant pitch

Time
Frequency (Hz)
0.5 1 1.5 2 2.5 3 3.5 4
0
1000
2000
3000
4000
5000
6000
7000
8000
f = fopen('iq-fm-97M-3.6M.dat', 'r', 'ieee-le');
c = fread(f, [2,inf], '*float32');
close(f);
z = c(1,:) + j*c(2,:);
fs = 3.6e6; % IQ sampling frequency
fciq = 97e6; % center frequency of IQ downconverter
spectrogram(double(z(1:5e5)), 1024, 512, 1024, fs, 'yaxis');
colormap(gray)
Spectral estimation

\[ \cos(2 \pi \cdot [0:15]/16 \cdot 4) \]

\[ \cos(2 \pi \cdot [0:15]/16 \cdot 4.2) \]
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.
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 *scalloppling*. 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).
Windowing

Sine wave

Discrete Fourier Transform

Sine wave multiplied with window function

Discrete Fourier Transform
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 $\sin(\pi fT)/(\pi fT)$ 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 $\sin(\pi fT)/(\pi fT)$ 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.
Some better window functions

All these functions are 0 outside the interval [0,1].
\[ \cos(2 \pi \frac{[0:15]}{16} \times 4.2) \]

**Time-domain samples:**
-0.5 0 0.5 1

**DTFT frequency (1 period):**
-\(\pi\) -\(\frac{3\pi}{4}\) -\(\frac{\pi}{2}\) -\(\frac{\pi}{4}\) 0 \(\frac{\pi}{4}\) \(\frac{\pi}{2}\) \(\frac{3\pi}{4}\) \(\pi\)

**DTFT magnitude**: Not shown.

**DFT magnitude**: Not shown.

\[ \cos(2 \pi \frac{[0:15]}{16} \times 4.2) \times \text{hann}(16) \]

**Time-domain samples:**
-0.5 0 0.5 1

**DTFT frequency (1 period):**
-\(\pi\) -\(\frac{3\pi}{4}\) -\(\frac{\pi}{2}\) -\(\frac{\pi}{4}\) 0 \(\frac{\pi}{4}\) \(\frac{\pi}{2}\) \(\frac{3\pi}{4}\) \(\pi\)

**DTFT magnitude**: Not shown.

**DFT magnitude**: Not shown.
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**:

\[
w_i = 0.54 - 0.46 \times \cos \left( 2\pi \frac{i}{n-1} \right)
\]
Does zero padding increase DFT resolution?

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

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

The left plot shows the DFT of the windowed sequence

\[ x_i = s_i \cdot w_i, \quad i \in \{0, \ldots, 15\} \]

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

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

where \( w_i = 0.54 - 0.46 \times \cos(2\pi i/15) \) is the Hamming window.
\[
\cos(2 \pi \frac{[0:15]}{16}\cdot3.3) + \cos(2 \pi \frac{[0:15]}{16}\cdot4)
\]
\[ \cos(2 \pi \frac{[0:15]}{16 \times 3.3}) + \cos(2 \pi \frac{[0:15]}{16 \times 4}) \]
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.
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
Filter: supresses (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

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.
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

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

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 $t < 0$
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

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

with $w_i = 0$ for $i < 0$ and $i > n$.

Note that for $f_c = f_s/4$, we have $h_i = 0$ for all even values of $i$. 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.
FIR low-pass filter design example

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

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.
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:

$$h_i = \frac{(f_h - f_l)}{f_s} \cdot \frac{\sin[\pi(i - n/2)(f_h - f_l)/f_s]}{\pi(i - n/2)(f_h - f_l)/f_s} \cdot \cos[\pi i (f_h + f_l)/f_s] \cdot w_i$$
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 $X'(f) = X(f_s/2 - f)$, we merely have to shift the periodic spectrum by $f_s/2$:

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

$$\ldots, 1, -1, 1, -1, 1, -1, 1, -1, \ldots$$

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 $-1$. 

\[109 / 144\]
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:

\[ h(t - \Delta t) \bullet\circ H(f) \cdot e^{-2\pi j f \Delta t} \]

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

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

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
Finite impulse response (FIR) filter

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

\( M = 3: \)

Transposed implementation:

(see slide 25)
Infinite impulse response (IIR) filter

\[ \sum_{k=0}^{N} a_k \cdot y_{n-k} = \sum_{m=0}^{M} b_m \cdot x_{n-m} \quad \text{Usually normalize: } a_0 = 1 \]

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

Direct form I implementation:
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:
We can represent sequences \( \{x_n\} \) as polynomials:

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

Example of polynomial multiplication:

\[
(1 + 2v + 3v^2) \cdot (2 + 1v)
\]

\[
= 2 + 5v + 8v^2 + 3v^3
\]

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

\[
\text{conv([1 2 3], [2 1])} \quad \text{equals} \quad [2 5 8 3]
\]
Convolution of sequences is equivalent to polynomial multiplication:

$$\{h_n\} \ast \{x_n\} = \{y_n\} \Rightarrow y_n = \sum_{k=-\infty}^{\infty} h_k \cdot x_{n-k}$$

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

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

$$X(e^{-j\omega}) = \sum_{n=-\infty}^{\infty} x_n e^{-j\omega n}$$
Example of polynomial division:

\[
\frac{1}{1 - av} = 1 + av + a^2v^2 + a^3v^3 + \cdots = \sum_{n=0}^{\infty} a^n v^n
\]

Rational functions (quotients of two polynomials) can provide a convenient closed-form representations for infinitely-long exponential sequences, in particular the impulse responses of IIR filters.
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 $\{x_n\}$:

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

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

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

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 ($|z| = 1$):

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

where $\omega = 2\pi \frac{f}{f_s}$. 
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 $z^{-1}$. 
Time reversal:

\[ \{x_{-n}\} \rightarrow X(\bar{z}^{-1}) \]

Multiplication with exponential:

\[ \{a^{-n}x_n\} \rightarrow X(az) \]

Complex conjugate:

\[ \{x_n^*\} \rightarrow X^*(\bar{z}^*) \]

Real/imaginary value:

\[ \mathcal{R}\{x_n\} \rightarrow \frac{1}{2}(X(z) + X^*(\bar{z}^*)) \]

\[ \mathcal{I}\{x_n\} \rightarrow \frac{1}{2j}(X(z) - X^*(\bar{z}^*)) \]

Initial value:

\[ x_0 = \lim_{z \to \infty} X(z) \quad \text{if } x_n = 0 \text{ for all } n < 0 \]
Some example sequences and their $z$-transforms:

<table>
<thead>
<tr>
<th>$x_n$</th>
<th>$X(z)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta_n$</td>
<td>$\frac{z}{z-1} = \frac{1}{1-z^{-1}}$</td>
</tr>
<tr>
<td>$u_n$</td>
<td>$\frac{z}{z-a} = \frac{1}{1-az^{-1}}$</td>
</tr>
<tr>
<td>$a^n u_n$</td>
<td>$\frac{z}{(z-1)^2}$</td>
</tr>
<tr>
<td>$n u_n$</td>
<td>$\frac{z(z+1)}{(z-1)^3}$</td>
</tr>
<tr>
<td>$e^{an} u_n$</td>
<td>$\frac{z}{z-e^a}$</td>
</tr>
<tr>
<td>$\left(\frac{n-1}{k-1}\right) e^{a(n-k)} u_{n-k}$</td>
<td>$\frac{1}{(z-e^a)^k}$</td>
</tr>
<tr>
<td>$\sin(\omega n + \varphi) u_n$</td>
<td>$\frac{z^2 \sin(\varphi) + z \sin(\omega - \varphi)}{z^2 - 2z \cos(\omega) + 1}$</td>
</tr>
</tbody>
</table>
Example:
What is the $z$-transform of the impulse response $\{h_n\}$ of the discrete system $y_n = x_n + a y_{n-1}$?

$$y_n = x_n + a y_{n-1}$$

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

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

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

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

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

$$H(z) = \frac{Y(z)}{X(z)} = \frac{z}{z - a}$$

We have applied here the linearity of the $z$-transform, and its time-shift and convolution properties.
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 $\{y_n\} = \{h_n\} \ast \{x_n\}$?
Using the linearity and time-shift property of the $z$-transform:

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

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

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

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

\[
H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_m z^{-m}}{a_0 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_k z^{-k}}
\]
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 $\{y_n\} = \{h_n\} \ast \{x_n\}$ is the rational function

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

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

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

$H(z)$ 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 $z = 0$. 

\[ \text{Diagram of the causal LTI system} \]
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., $z \to d_l \Rightarrow |H(z)| \to \infty$) 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 $\{h_n\}$ ($\omega = \pi f / f_s$). The DTFT amplitude can also be expressed in terms of the relative position of $e^{j\omega}$ to the zeros and poles:

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

Consider this IIR filter:

\[ x_n \quad 0.8 \]

\[ \quad + \quad \]

\[ \quad 0.2 \quad z^{-1} \]

\[ y_n \]

\[ y_{n-1} \]

\[ a_0 = 1, \quad a_1 = -0.2, \quad b_0 = 0.8 \]

\[ x_n = \delta_n \Rightarrow y_n = \]

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 \( z = 0 \).
\[ H(z) = \frac{0.8}{1-0.2z^{-1}} = \frac{0.8z}{z-0.2} \text{ (cont'd)} \]

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:
\[
A(f) = |H(e^{j2\pi f / f_s})|
\]
\[
\theta(f) = \angle H(e^{j2\pi f / f_s}) = \tan^{-1} \frac{\Im\{H(e^{j\pi f / f_s})\}}{\Re\{H(e^{j\pi f / f_s})\}}
\]

Example:
\( f_s = 8 \text{ kHz}, f = 2 \text{ kHz} \) (normalized frequency \( f / f_s \frac{1}{2} = 0.5 \)) \Rightarrow \text{Gain } A(2 \text{ kHz}) =
\[
|H(e^{j\pi / 2})| = |H(j)| = |\frac{0.8j}{j-0.2}| = |\frac{0.8j(-j-0.2)}{(j-0.2)(-j-0.2)}| = |\frac{0.8-0.16j}{1+0.04} | = \sqrt{\frac{0.8^2+0.16^2}{1.04^2}} = 0.784\ldots
\]
Visual verification in MATLAB:

\[
\begin{align*}
n & = 0:15; \\
fs & = 8000; \\
f & = 1500; \\
x & = \cos(2\pi f n/\fs); \\
b & = [0.8]; \ a = [1 \ -0.2]; \\
y1 & = \text{filter}(b, a, x); \\
z & = \exp(j 2\pi f/\fs); \\
H & = 0.8 z/(z-0.2); \\
A & = \text{abs}(H); \\
theta & = \text{atan}(\text{imag}(H)/\text{real}(H)); \\
y2 & = A \cos(2\pi f n/\fs + \theta); \\
\text{plot}(n, x, 'bx-', \\
\quad n, y1, 'go-', \\
\quad n, y2, 'r+-') \\
\text{legend('x', ...} \\
\quad 'y (time domain)', ... \\
\quad 'y (z\text{-transform}') \\
\text{ ylim([-1.1 1.8])}
\end{align*}
\]
$$H(z) = \frac{z}{z-0.7} = \frac{1}{1-0.7 \cdot z^{-1}}$$

How do poles affect time domain?

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

\[ H(z) = \frac{z}{z-1.1} = \frac{1}{1-1.1 \cdot z^{-1}} \]
\[ H(z) = \frac{z^2}{(z-0.9\cdot e^{j\pi/6})\cdot(z-0.9\cdot e^{-j\pi/6})} = \frac{1}{1-1.8\cos(\pi/6)z^{-1}+0.9^2z^{-2}} \]
\[ H(z) = \frac{z^2}{(z - 0.9 \cdot e^{j\pi/2})(z - 0.9 \cdot e^{-j\pi/2})} = \frac{1}{1 - 1.8 \cos(\pi/2)z^{-1} + 0.9^2 z^{-2}} = \frac{1}{1 + 0.9^2 z^{-2}} \]
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 number of poles it uses in the $z$-domain, and equivalently the number of delay elements necessary 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.
IIR filter design techniques

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 $1 - \sqrt{1/2}$ (−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.
The aforementioned filter-design techniques are implemented in the MATLAB Signal Processing Toolbox in the functions \texttt{butter}, \texttt{cheby1}, \texttt{cheby2}, and \texttt{ellip}. They output the coefficients $a_n$ and $b_n$ of the difference equation that describes the filter. These can be applied with \texttt{filter} to a sequence, or can be visualized with \texttt{zplane} as poles/zeros in the $z$-domain, with \texttt{impz} as an impulse response, and with \texttt{freqz} as an amplitude and phase spectrum. The commands \texttt{sptool} and \texttt{fdatool} provide interactive GUIs to design digital filters.
Cascade of filter sections

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 $H_1, H_2, \ldots, H_l$ 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.
Butterworth filter design example

order: 1, cutoff frequency (−3 dB): $0.25 \times \frac{f_s}{2}$
Butterworth filter design example

order: 5, cutoff frequency (−3 dB): $0.25 \times f_s/2$
Chebyshev type I filter design example

order: 5, cutoff frequency: $0.5 \times f_s/2$, pass-band ripple: $-3$ dB
Chebyshev type II filter design example

order: 5, cutoff frequency: $0.5 \times f_s/2$, stop-band ripple: $-20$ dB
Elliptic filter design example

order: 5, cutoff frequency: \(0.5 \times f_s/2\), pass-band ripple: \(-3\) dB, stop-band ripple: \(-20\) dB
Notch filter design example

**z Plane**

**Impulse Response**

order: 2, cutoff frequency: $0.25 \times f_s/2$, $-3$ dB bandwidth: $0.05 \times f_s/2$
Peak filter design example

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