# Digital Signal Processing

Markus Kuhn

UNIVERSITY OF
CAMBRIDGE

Computer Laboratory

http://www.cl.cam.ac.uk/Teaching/2003/DigSigProc/

Easter 2004 – Part II

# Textbooks

- R.G. Lyons: *Understanding digital signal processing.* Prentice-Hall, 2004. (£45)
- A.V. Oppenheim, R.W. Schafer: *Discrete-time signal processing.* 2nd ed., Prentice-Hall, 1999. (£47)
- J. Stein: *Digital signal processing – a computer science perspective.* Wiley, 2000. (£74)
- S.W. Smith: *Digital signal processing – a practical guide for engineers and scientists.* Newness, 2003. (£40)
- K. Steiglitz: *A digital signal processing primer – with applications to digital audio and computer music.* Addison-Wesley, 1996. (£40)
- Sanjit K. Mitra: *Digital signal processing – a computer-based approach.* McGraw-Hill, 2002. (£38)

# Digital signal processing

Concerned with algorithms to interpret, transform, and model waveforms and the information they contain. Some typical applications:

- → geophysics
  seismology, oil exploration
- → astronomy
  speckle interferometry, VLBI
- → experimental physics
  sensor-data evaluation
- → communication systems
  modulation/demodulation, channel equalization, echo cancellation
- → aviation
  radar, radio navigation
- → music
  synthetic instruments, audio effects, noise reduction
- → medical diagnostics
  ECG, EEG, MEG, audiology, computer tomography, magnetic-resonance imaging
- → consumer electronics
  perceptual coding of audio and video on DVDs, speech synthesis, speech recognition
- → security
  steganography, digital watermarking, biometric identification, SIGINT
- → engineering
  control systems, feature extraction for pattern recognition

# Sequences and systems

A *discrete sequence* $\{x_n\}$ 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.

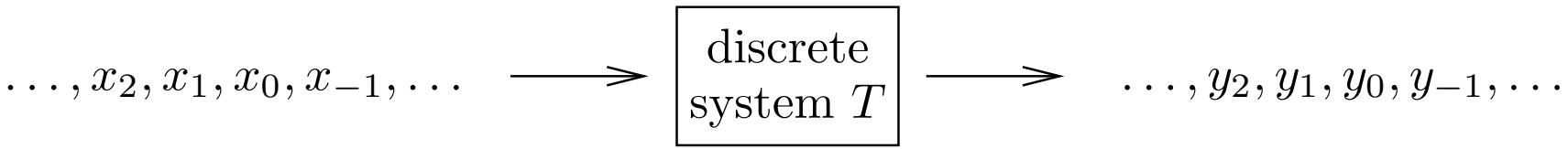
The notation is not well standardized. Some authors write $x[n]$ instead of $x_n$, others $x(n)$.

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

$$x_n = x(t_\mathrm{s} \cdot n) = x(n/f_\mathrm{s}),$$

we call $t_\mathrm{s}$ the *sampling period* and $f_\mathrm{s} = 1/t_\mathrm{s}$ the *sampling frequency*.

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



# Properties of sequences

A sequence $\{x_n\}$ is

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

$$\textit{square summable} \quad \Leftrightarrow \quad \sum_{n=-\infty}^{\infty} |x_n|^2 < \infty$$

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

A square-summable sequence is also called an *energy signal*, and

$$\sum_{n=-\infty}^{\infty} |x_n|^2$$

its energy. This terminology reflects that if $U$ is a voltage supplied to a load resistor $R$, then $P = UI = U^2/R$ is the power consumed.

So even where we drop physical units (e.g., volts) for simplicity in sequence calculations, it is still customary to refer to the squared values of a sequence as *power* and to its sum or integral over time as *energy*.

A non-square-summable sequence is a *power signal* if its average power

$$\lim_{k\to\infty} \frac{1}{1+2k} \sum_{n=-k}^{k} |x_n|^2$$

exists.

# Special sequences

Unit-step sequence:

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

Impulse sequence:

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

# Types of discrete systems

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 $d$

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

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

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

## Examples:

The *accumulator system*

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

is a causal, linear, time-invariant system with memory, as are the *backward difference system*

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

the *M-point moving average system*

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

and the *exponential averaging system*

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

Examples for time-invariant non-linear memory-less systems are

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

examples for linear but not time-invariant systems are

$$\begin{aligned} y_n &= \begin{cases} x_n, & n \geq 0 \\ 0, & n < 0 \end{cases} = x_n \cdot u_n \\ y_n &= x_{\lfloor n/4 \rfloor} \\ y_n &= x_n \cdot \Re(e^{\omega \mathrm{j} n}) \end{aligned}$$

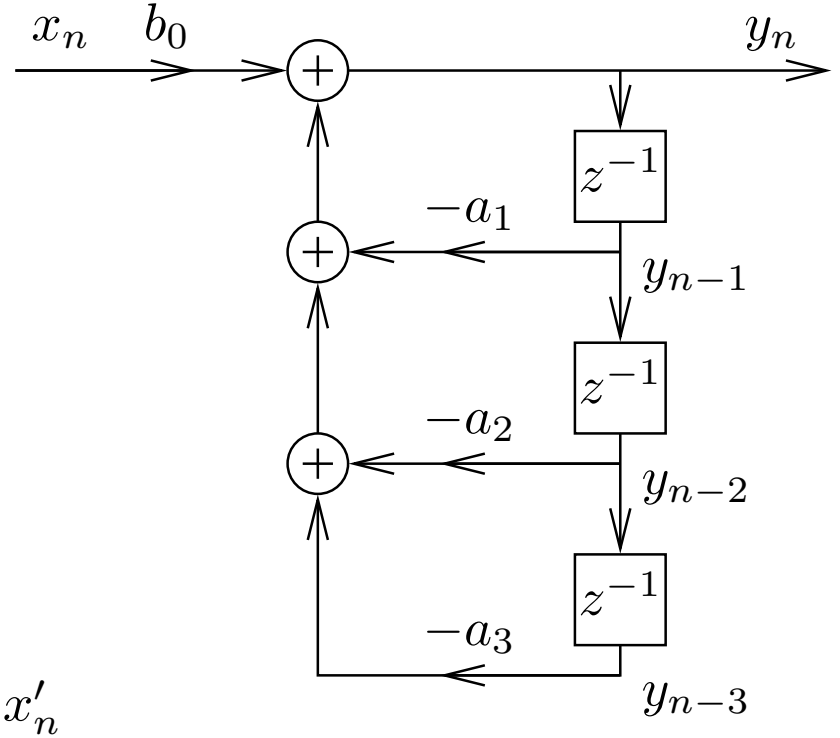
and examples for linear time-invariant non-causal system are

$$\begin{aligned} y_n &= \frac{1}{2}(x_{n-1} + x_{n+1}) \\ y_n &= \sum_{k=-9}^{9} x_{n+k} \cdot \frac{\sin(\pi k \omega)}{\pi k \omega} \cdot [0.5 + 0.5 \cdot \cos(\pi k / 10)] \end{aligned}$$

# Constant-coefficient difference equations

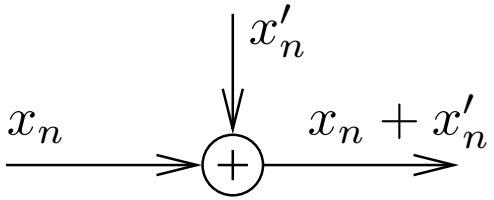
Of particular practical interest are causal linear time-invariant systems of the form

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

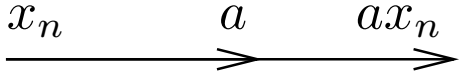



Block diagram representation of sequence operations:

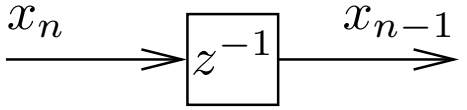
Addition:




Multiplication by constant:



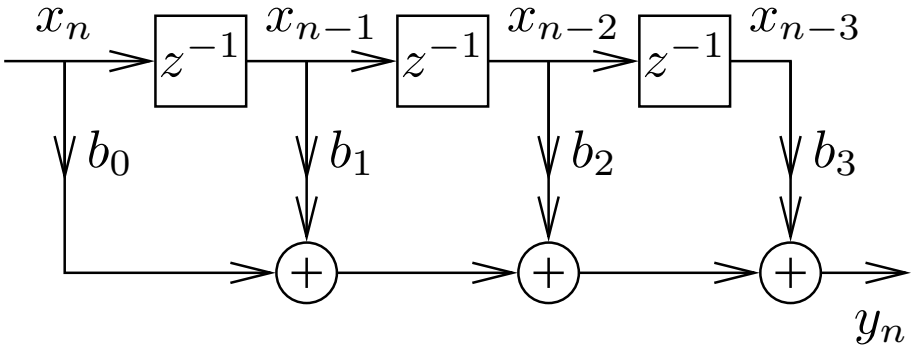

Delay:




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

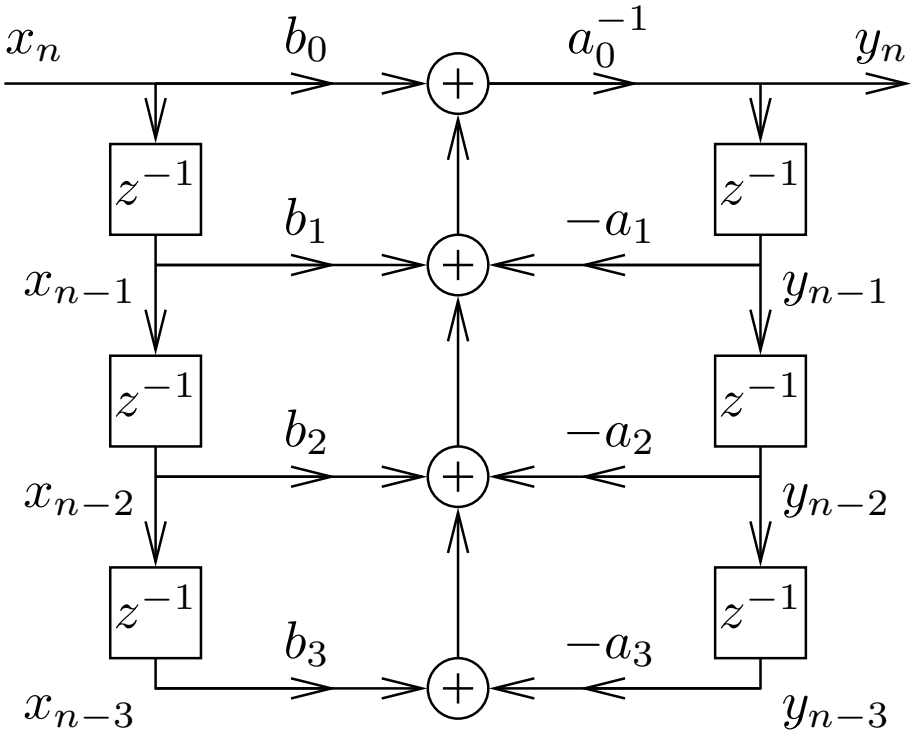
or

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




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 `filter` is an efficient implementation of the last variant.

# Convolution

All linear time-invariant (LTI) systems can be represented in the form

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

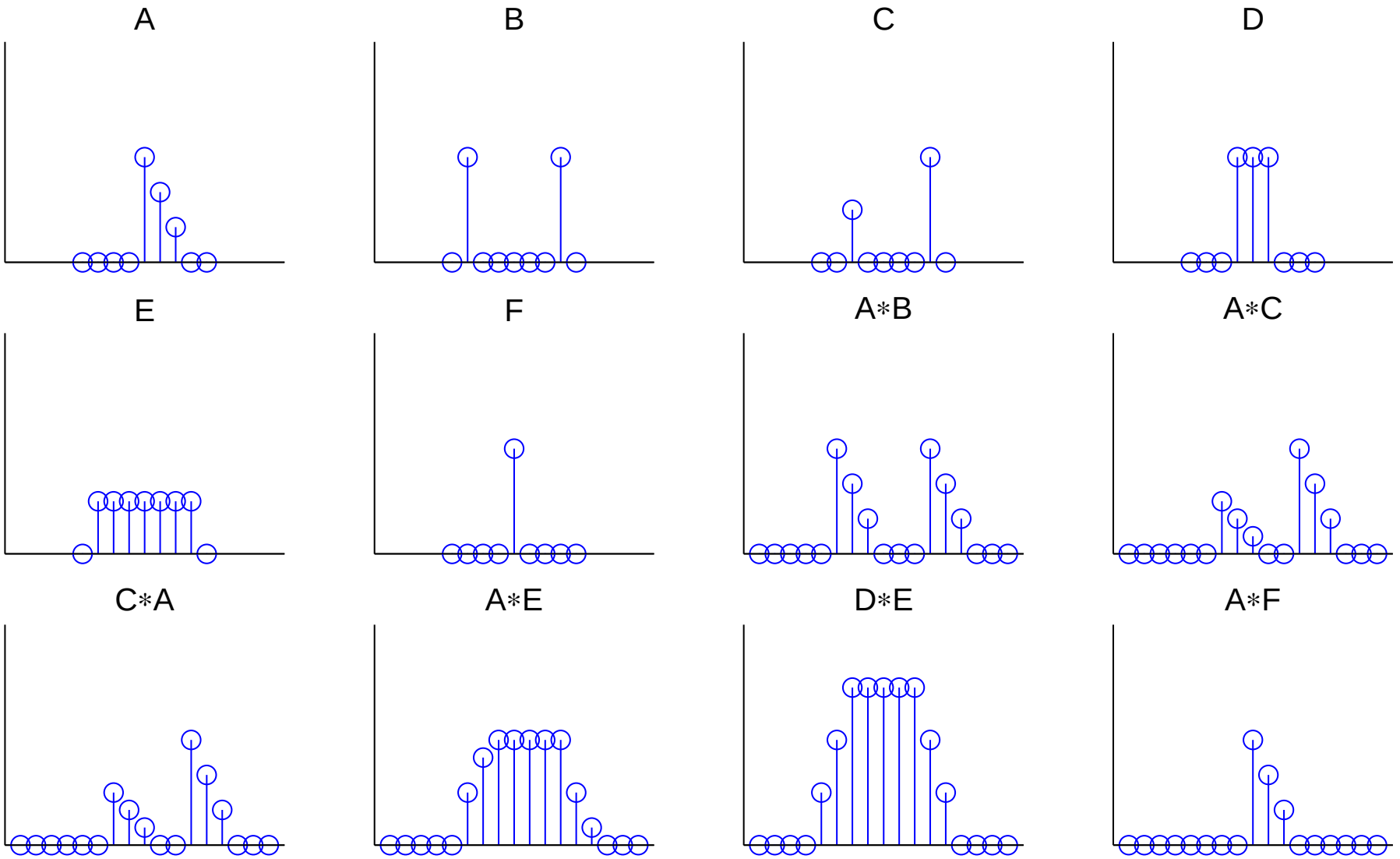
where $\{a_k\}$ is a suitably chosen sequence of coefficients.

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

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

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 $\{a_n\} = T\{\delta_n\}$.

# Convolution examples



# Properties of convolution

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

$\longrightarrow$ Convolution is associative

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

$\longrightarrow$ Convolution is commutative

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

$\longrightarrow$ Convolution is linear

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

$\longrightarrow$ The impulse sequence (slide 6) is neutral under convolution

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

$\longrightarrow$ Sequence shifting is equivalent to convolving with a shifted impulse

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

# Can all LTI systems be represented by 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:

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

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

**Exercise 1** A *finite-length sequence* is non-zero only at a finite number of positions. If $m$ and $n$ are the first and last non-zero positions, respectively, then we call $n-m+1$ the *length* of that sequence. What maximum length can the result of convolving two sequences of length $k$ and $l$ have?

**Exercise 2** The length-3 sequence $a_0 = -3$, $a_1 = 2$, $a_2 = 1$ is convolved with a second sequence $\{b_n\}$ of length 5.

(a) Write down this linear operation as a matrix multiplication involving a matrix $A$, a vector $\vec{b} \in \mathbb{R}^5$, and a result vector $\vec{c}$.

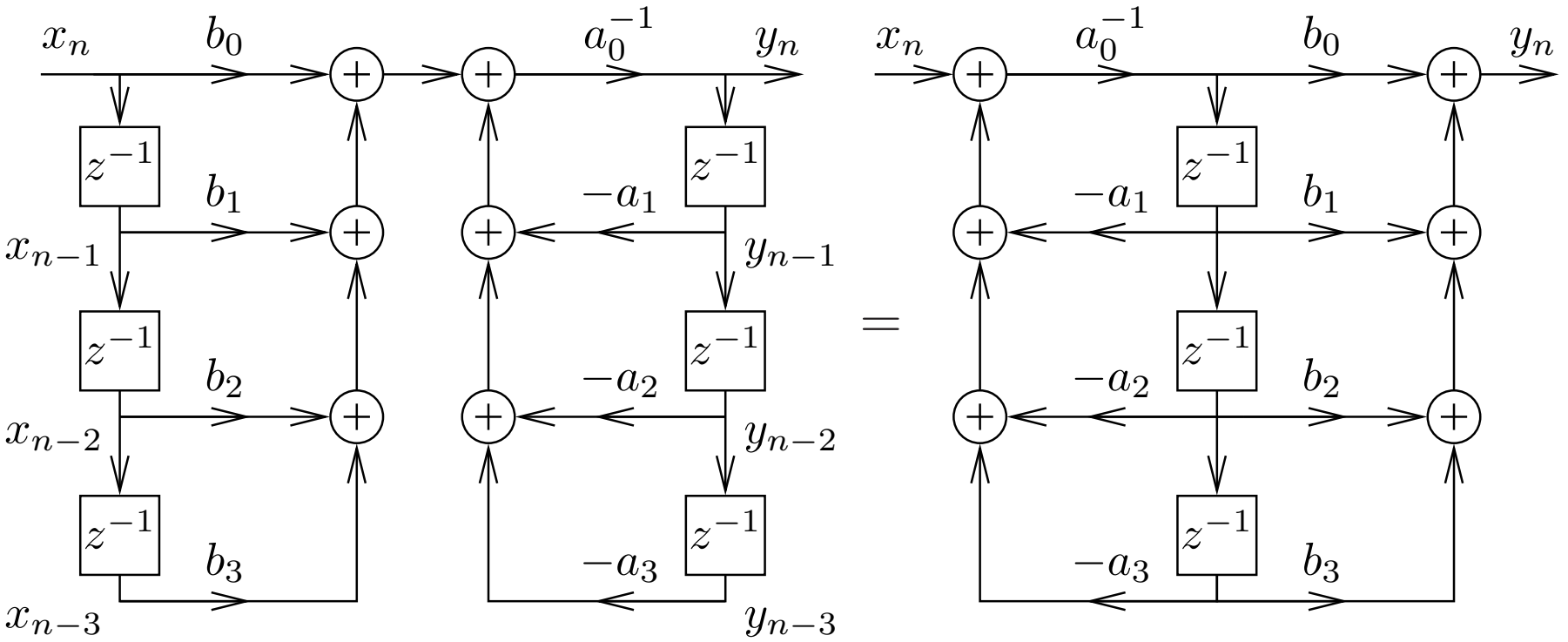
(b) Use MATLAB to multiply your matrix by the vector $\vec{b} = (1, 0, 0, 2, 2)$ and compare the result with that of using the `conv` function.

(c) Use the MATLAB facilities for solving systems of linear equations to undo the above convolution step.

**Exercise 3** (a) Find a pair of sequences $\{a_n\}$ and $\{b_n\}$, where either contains at least three different values and where the convolution $\{a_n\} * \{b_n\}$ results in an all-zero sequence.

(b) Does every LTI system $T$ have an inverse LTI system $T^{-1}$ such that $\{x_n\} = T^{-1}T\{x_n\}$ for all sequences $\{x_n\}$? Why?

# Direct form I and II implementations




The block diagram representation of the constant-coefficient difference equation on slide 11 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.

# Why are sine waves useful?

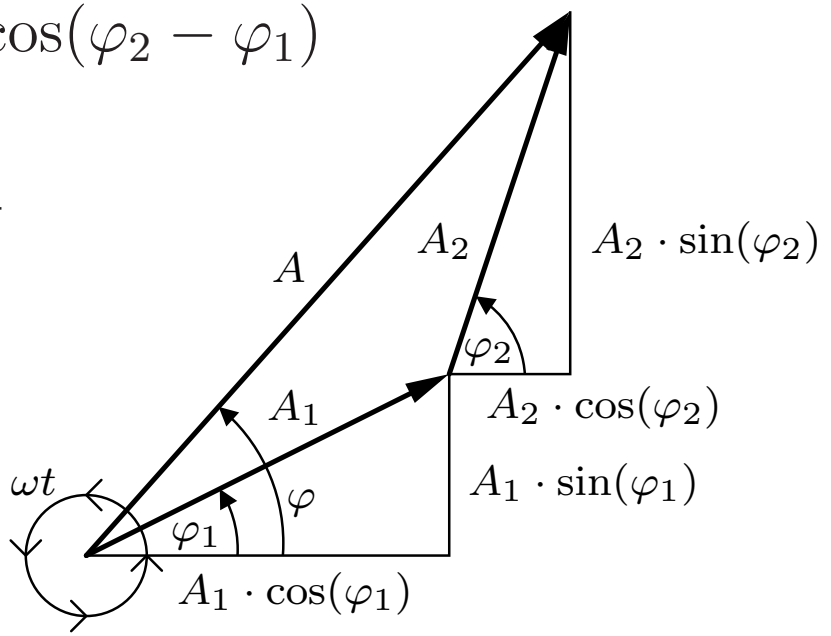
Adding together two sine waves of equal frequency, but arbitrary amplitude and phase, results in another sine wave of the same frequency:

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

with

$$A \;=\; \sqrt{A_1^2 + A_2^2 + 2A_1 A_2 \cos(\varphi_2 - \varphi_1)}$$

$$\tan\varphi \;=\; \frac{A_1 \sin\varphi_1 + A_2 \sin\varphi_2}{A_1 \cos\varphi_1 + A_2 \cos\varphi_2}$$




Also

$$A \cdot \sin(\omega t + \varphi) =$$
$$a \cdot \sin(\omega t) + b \cdot \cos(\omega t)$$

with $A = \sqrt{a^2 + b^2}$ and $\tan\varphi = \frac{b}{a}$.

Convolution of a discrete sequence $\{x_n\}$ with another sequence $\{y_n\}$ is nothing but adding together scaled and delayed copies of $\{x_n\}$, according to a decomposition of $\{y_n\}$ into a sum of impulses. If $\{x_n\}$ was a sampled sine wave of frequency $f$, so will $\{x_n\} * \{y_n\}$ be.

$\Longrightarrow$ Sine-wave sequences form a family of discrete sequences that is closed under convolution with arbitrary sequences.

Sine waves are orthogonal to each other

$$\int_{-\infty}^{\infty} \sin(\omega_1 t + \varphi_1) \cdot \sin(\omega_2 t + \varphi_2)\,\mathrm{d}t = 0$$

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

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

# 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

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

(where $\mathrm{j}^2 = -1$) are able to represent both amplitude and phase in one single algebraic object.

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

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

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

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

# Recall: Fourier transform

The Fourier integral transform and its inverse are defined as

$$\mathcal{F}\{g(t)\}(\omega) \;=\; G(\omega) \;=\; \alpha \int_{-\infty}^{\infty} g(t) \cdot \mathrm{e}^{\mp \mathrm{j}\omega t}\,\mathrm{d}t$$

$$\mathcal{F}^{-1}\{G(\omega)\}(t) \;=\; g(t) \;=\; \beta \int_{-\infty}^{\infty} G(\omega) \cdot \mathrm{e}^{\pm \mathrm{j}\omega t}\,\mathrm{d}\omega$$

where $\alpha$ and $\beta$ are constants chosen such that $\alpha\beta = 1/(2\pi)$.

Many equivalent forms of the Fourier transform are used in the literature, and there is no strong consensus of whether the forward transform uses $\mathrm{e}^{-\mathrm{j}\omega t}$ and the backwards transform $\mathrm{e}^{\mathrm{j}\omega t}$, or vice versa. Some authors set $\alpha = 1$ and $\beta = 1/(2\pi)$, to keep the convolution theorem free of a constant prefactor, others use $\alpha = \beta = 1/\sqrt{2\pi}$, in the interest of symmetry.

The substitution $\omega = 2\pi f$ leads to a form without prefactors:

$$\mathcal{F}\{h(t)\}(f) \;=\; H(f) \;=\; \int_{-\infty}^{\infty} h(t) \cdot \mathrm{e}^{\mp 2\pi \mathrm{j} f t}\,\mathrm{d}t$$

$$\mathcal{F}^{-1}\{H(f)\}(t) \;=\; h(t) \;=\; \int_{-\infty}^{\infty} H(f) \cdot \mathrm{e}^{\pm 2\pi \mathrm{j} f t}\,\mathrm{d}f$$

Another notation is in the continuous case

$$\begin{aligned}
\mathcal{F}\{h(t)\}(\omega) &= H(\mathrm{e}^{\mathrm{j}\omega}) &&= \int_{-\infty}^{\infty} h(t) \cdot \mathrm{e}^{-\mathrm{j}\omega t}\,\mathrm{d}t \\
\mathcal{F}^{-1}\{H(\mathrm{e}^{\mathrm{j}\omega})\}(t) &= h(t) &&= \frac{1}{2\pi}\int_{-\infty}^{\infty} H(\mathrm{e}^{\mathrm{j}\omega}) \cdot \mathrm{e}^{\mathrm{j}\omega t}\,\mathrm{d}\omega
\end{aligned}$$

and in the discrete-sequence case

$$\begin{aligned}
\mathcal{F}\{h_n\}(\omega) &= H(\mathrm{e}^{\mathrm{j}\omega}) &&= \sum_{n=-\infty}^{\infty} h_n \cdot \mathrm{e}^{-\mathrm{j}\omega n} \\
\mathcal{F}^{-1}\{H(\mathrm{e}^{\mathrm{j}\omega})\}(t) &= h_n &&= \frac{1}{2\pi}\int_{-\pi}^{\pi} H(\mathrm{e}^{\mathrm{j}\omega}) \cdot \mathrm{e}^{\mathrm{j}\omega n}\,\mathrm{d}\omega
\end{aligned}$$

which treats the Fourier transform as a special case of the $z$-transform to be introduced later.

# Convolution theorem

Continuous form:

$$\begin{aligned} \mathcal{F}\{(f * g)(t)\} &= \mathcal{F}\{f(t)\} \cdot \mathcal{F}\{g(t)\} \\ \mathcal{F}\{f(t) \cdot g(t)\} &= \mathcal{F}\{f(t)\} * \mathcal{F}\{g(t)\} \end{aligned}$$

Discrete form:

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

Convolution in the time domain is equivalent to (complex) scalar multiplication in the frequency domain, and convolution in the frequency domain corresponds to scalar multiplication in the time domain.

# Dirac's delta function

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

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

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

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

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 Dirac's delta function:

$$\int_{-\infty}^{\infty} f(x)\delta(x-a)\,\mathrm{d}x = f(a)$$

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

$$\frac{1}{2\pi}\int_{-\infty}^{\infty} \mathrm{e}^{\pm \mathrm{j}\omega t}\mathrm{d}\omega = \delta(t)$$

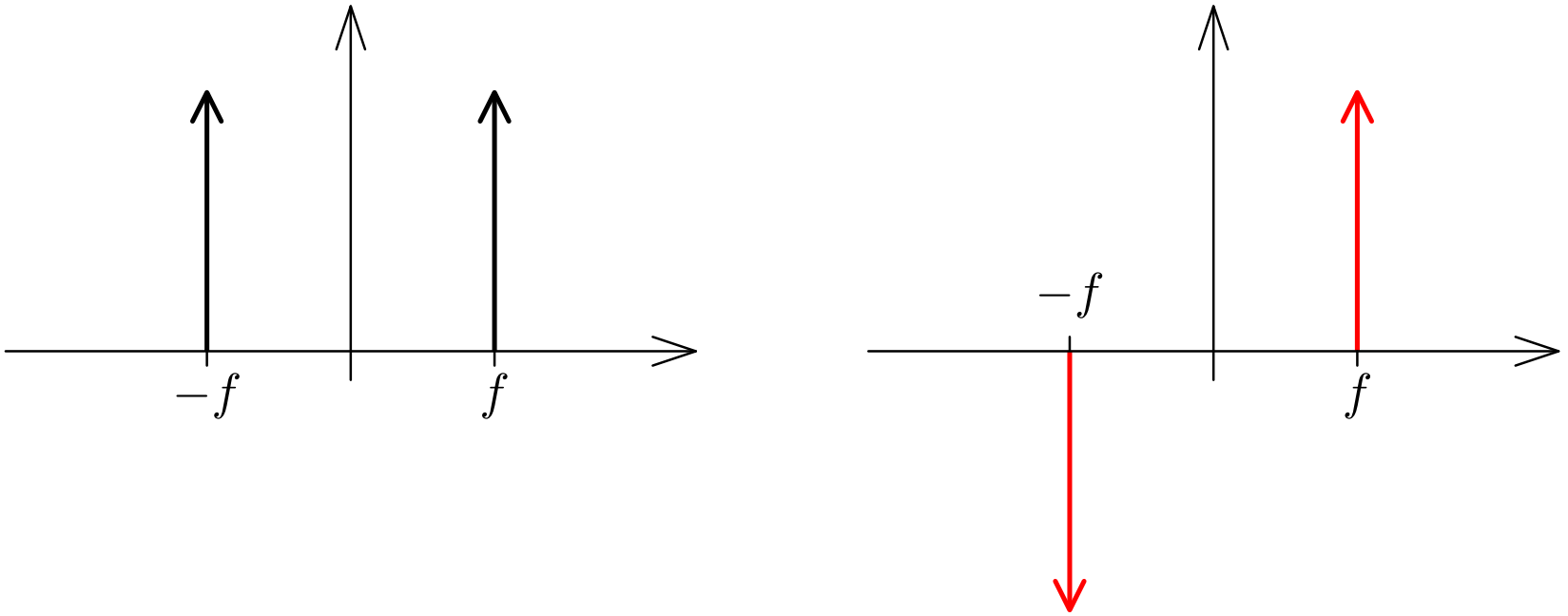
Fourier transform:

$$\mathcal{F}\{\delta(t)\}(\omega) = \int_{-\infty}^{\infty} \delta(t)\cdot \mathrm{e}^{-\mathrm{j}\omega t}\,\mathrm{d}t = \mathrm{e}^{0} = 1$$

$$\mathcal{F}^{-1}\{1\}(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} 1 \cdot \mathrm{e}^{\mathrm{j}\omega t}\,\mathrm{d}\omega = \delta(t)$$

# Sine and cosine in the frequency domain

$$\cos(2\pi f t) = \frac{1}{2}\,\mathrm{e}^{2\pi \mathrm{j} f t} + \frac{1}{2}\,\mathrm{e}^{-2\pi \mathrm{j} f t} \qquad \sin(2\pi f t) = \frac{1}{2\mathrm{j}}\,\mathrm{e}^{2\pi \mathrm{j} f t} - \frac{1}{2\mathrm{j}}\,\mathrm{e}^{-2\pi \mathrm{j} f t}$$




As any real-valued signal $x(t)$ can be represented as a combination of sine and cosine functions, the spectrum of any real-valued signal will show the symmetry $X(e^{\mathrm{j}\omega}) = [X(e^{-\mathrm{j}\omega})]^*$, where $^*$ denotes the complex conjugate (i.e., negated imaginary part).

# 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

$$x_n = x(t_\mathrm{s} \cdot n) = x(n/f_\mathrm{s})$$

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

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

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.

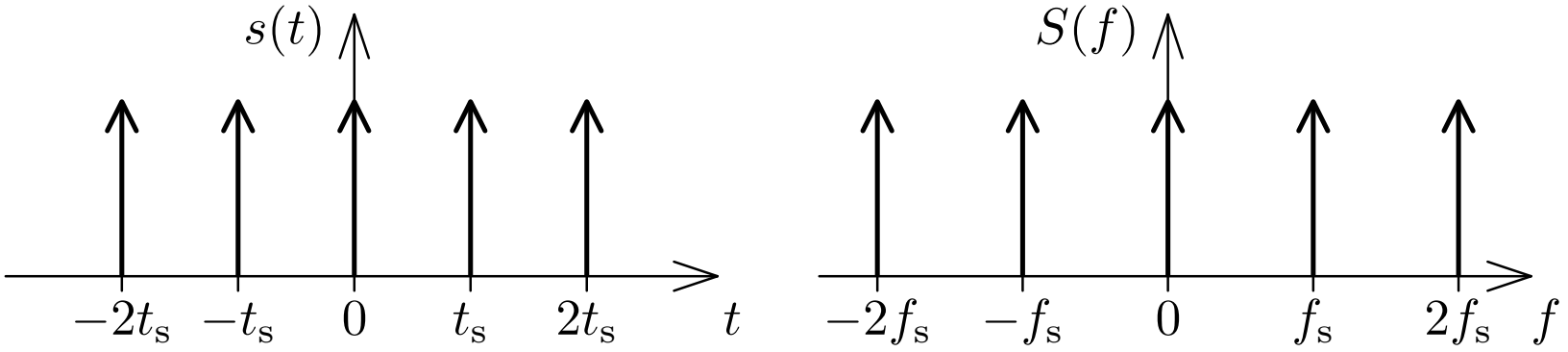
The Fourier transform of a Dirac comb

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

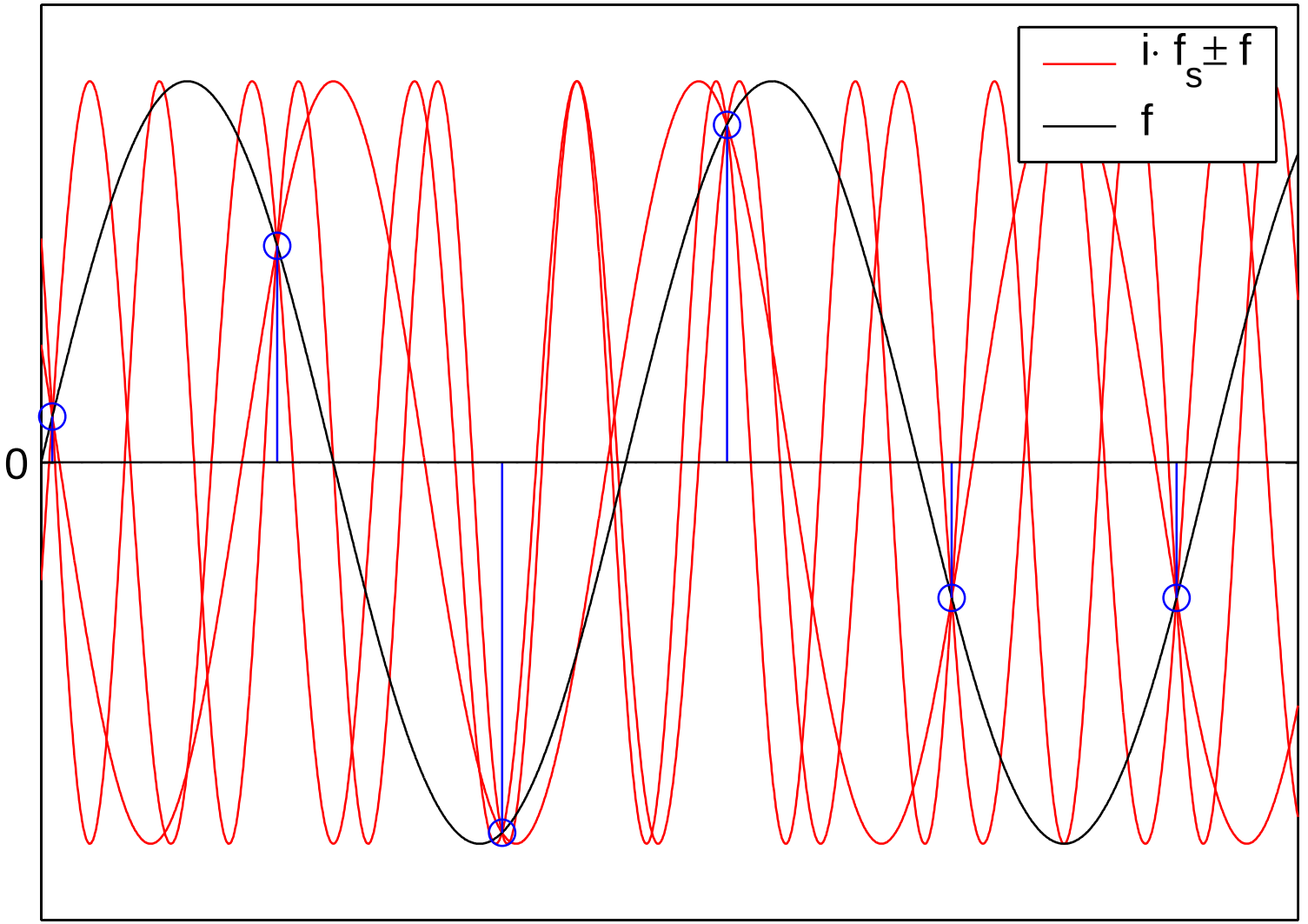
is another Dirac comb

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

$$\int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \delta(t - t_\mathrm{s} n)\, \mathrm{e}^{2\pi \mathrm{j} f t} \mathrm{d}t = \frac{1}{t_\mathrm{s}} \sum_{n=-\infty}^{\infty} \delta\left(f - \frac{n}{t_\mathrm{s}}\right).$$



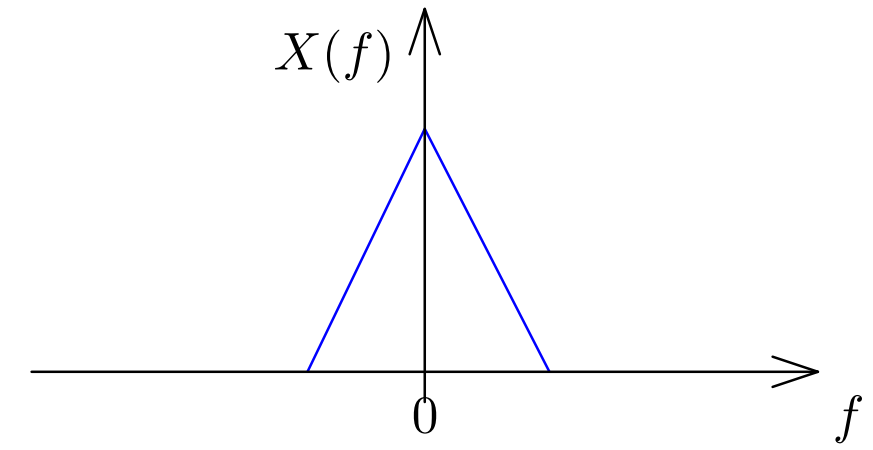
# Sampling, aliasing and Nyquist limit



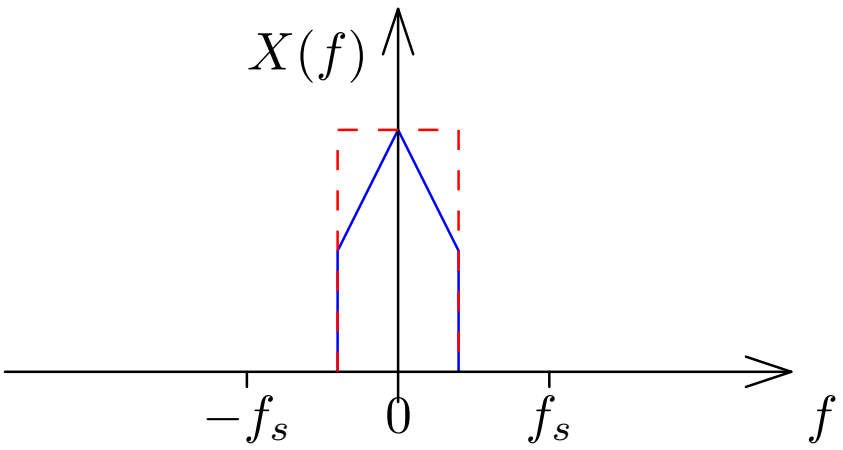

A wave $\cos(2\pi t f)$ sampled at frequency $f_\mathrm{s}$ cannot be distinguished from $\cos(2\pi t(kf_\mathrm{s} \pm f))$ for any $k \in \mathbb{Z}$, therefore ensure $|f| < f_s/2$.

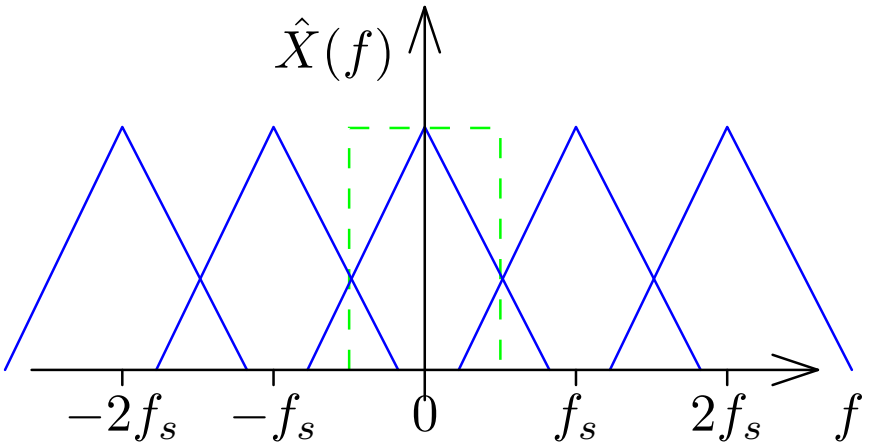
# Frequency-domain view of sampling

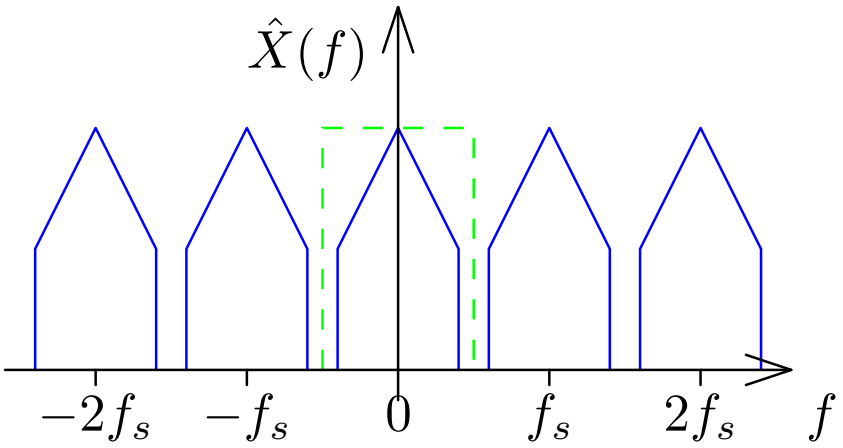
Without anti-aliasing filter




With anti-aliasing filter



## Exercise 4

- Generate a one second long Gaussian noise sequence $\{r_n\}$ (using MATLAB function `randn`) with a sampling rate of 300 Hz.
- Use the `fir1(50, 45/150)` function to design a finite impulse response low-pass filter with a cut-off frequency of 45 Hz. Use the `filtfilt` function in order to apply that filter to the generated noise signal, resulting in the filtered noise signal $\{x_n\}$.
- Then sample $\{x_n\}$ at 100 Hz by setting all but every third sample value to zero, resulting in sequence $\{y_n\}$.
- Generate another lowpass filter with a cut-off frequency of 50 Hz and apply it to $\{y_n\}$, in order to interpolate the reconstructed filtered noise signal $\{z_n\}$. Multiply the result by three, to compensate the energy lost during sampling.
- Plot $\{x_n\}$, $\{y_n\}$, and $\{z_n\}$. Finally compare $\{x_n\}$ and $\{z_n\}$.

Why should the first filter have a lower cut-off frequency than the second?

This page is intentionally left blank.

# Reconstruction of a continuous band-limited waveform

The ideal anti-aliasing filter for eliminating any frequency content above $f_\mathrm{s}/2$ before sampling with a frequency of $f_\mathrm{s}$ has the Fourier transform

$$H(f) = \begin{cases} 1 & \text{if } -\frac{f_\mathrm{s}}{2} < f \leq \frac{f_\mathrm{s}}{2} \\ 0 & \text{otherwise} \end{cases} .$$

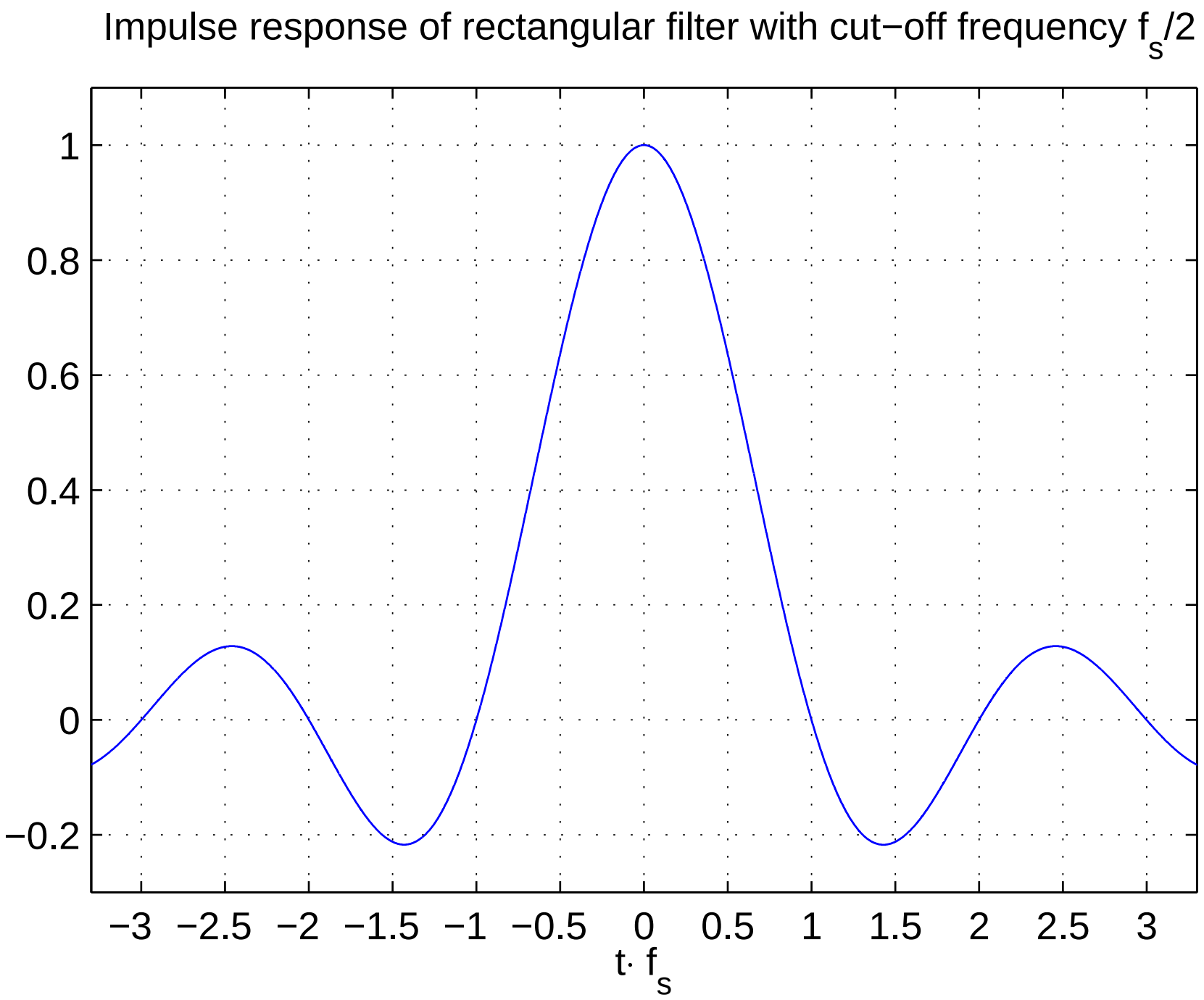
This leads, after an inverse Fourier transform, to the impulse response

$$h(t) = \frac{\sin \pi t f_\mathrm{s}}{\pi t f_\mathrm{s}}.$$

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: $h(t) \cdot s(t) = \delta(t)$.

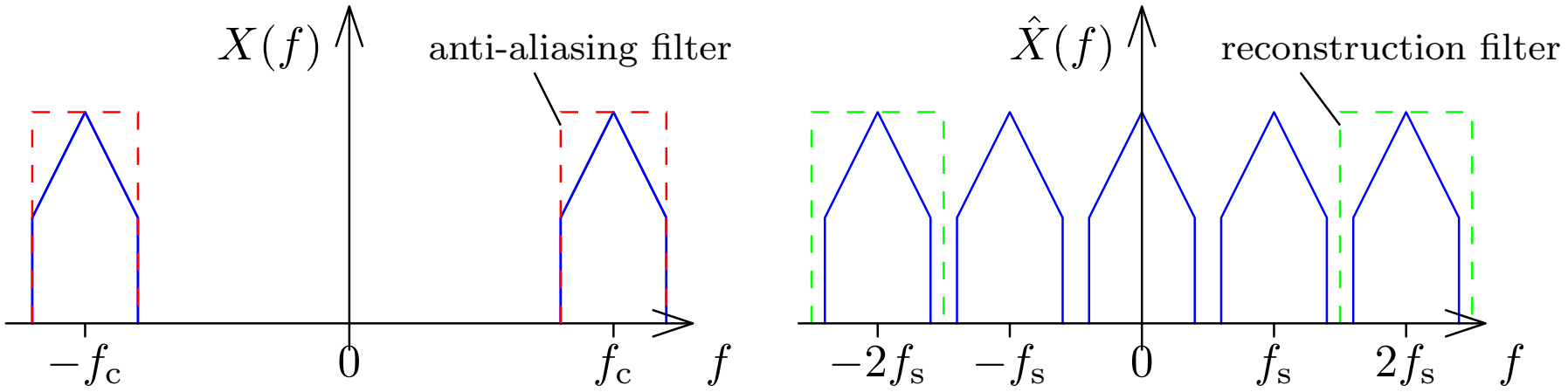

Impulse response of rectangular filter with cut−off frequency $f_s/2$
1
0.8
0.6
0.4
0.2
0
−0.2
−3
−2.5
−2
−1.5
−1
−0.5
0
0.5
1
1.5
2
2.5
3
$t \cdot f_s$

# Bandpass sampling

Sampled signals can be reconstructed as long as their doublesided bandwidth (counting both the positive and negative frequency range) does not exceed the sampling frequency. The anti-aliasing filter used can also be a band-pass filter instead of a low-pass filter.

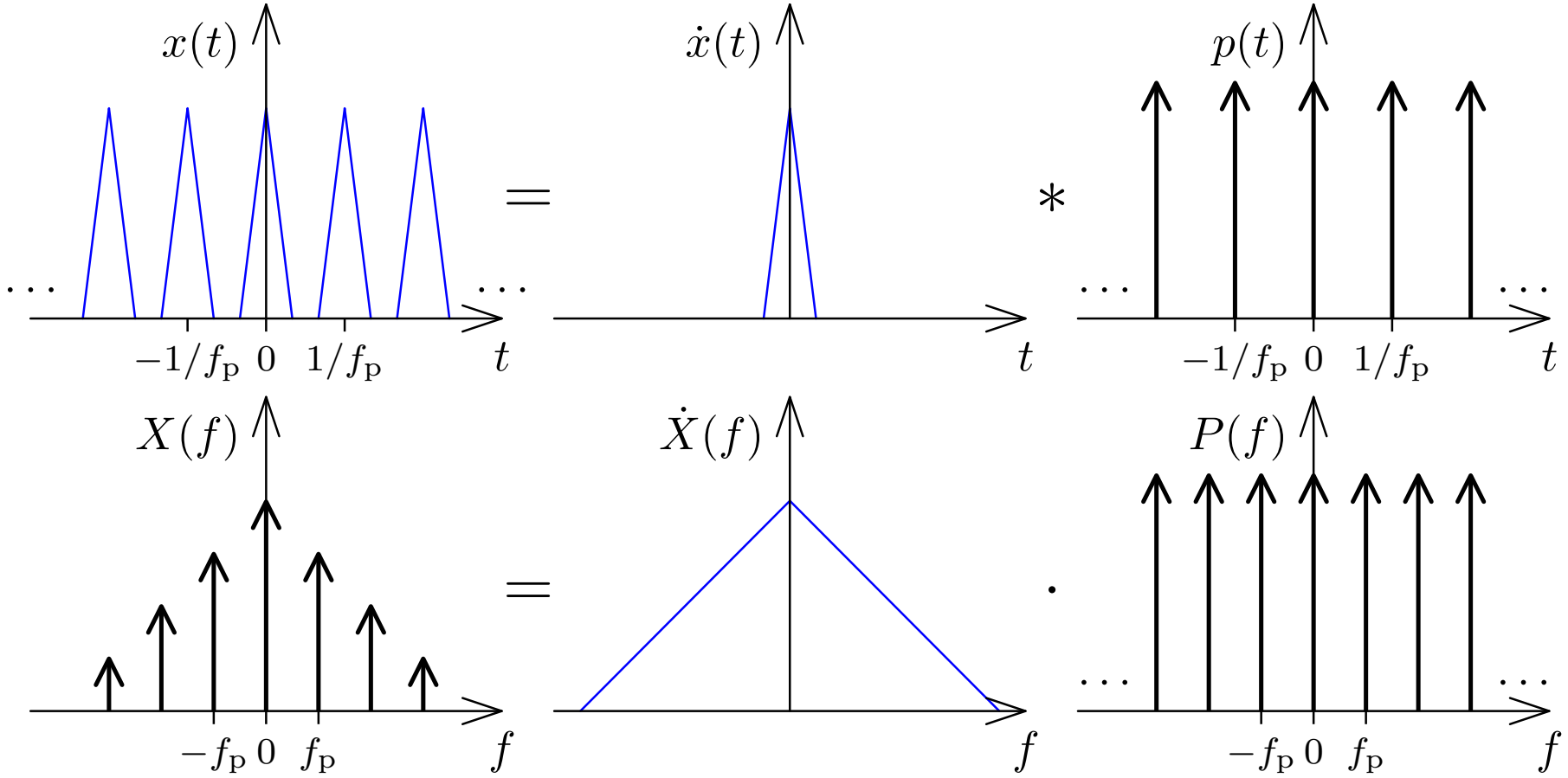
A suitably bandlimited signal $x(t)$ with center frequency $f_\mathrm{c}$ sampled at frequency $f_\mathrm{s}$ can be reconstructed by convolution with

$$h(t) = \frac{\sin \pi t f_\mathrm{s}}{\pi t f_\mathrm{s}} \cdot \cos(2\pi t f_\mathrm{c}).$$



# Spectrum of a periodic signal

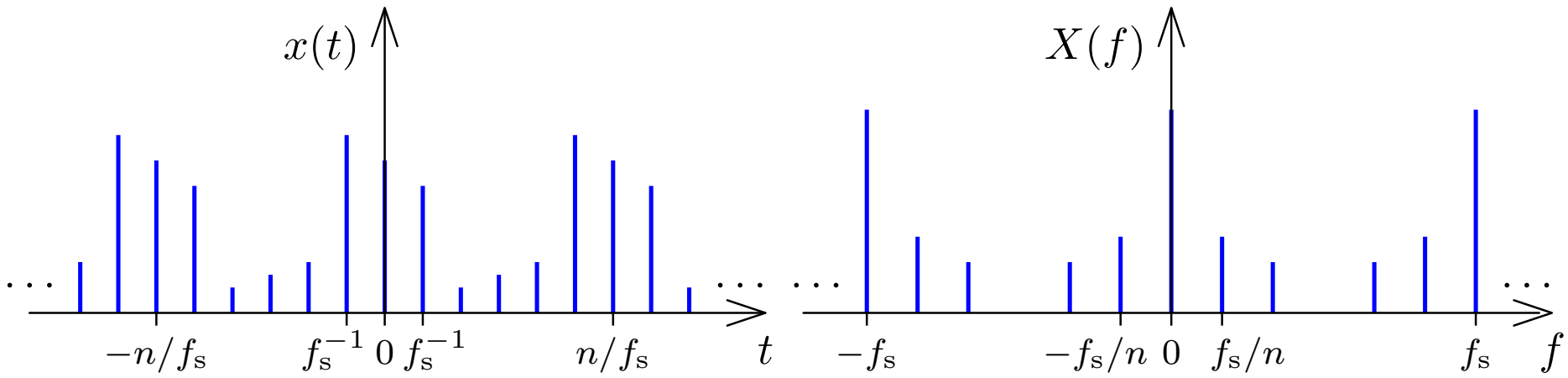
A signal $x(t)$ that is periodic with frequency $f_\mathrm{p}$ can be factored into a single period $\dot{x}(t)$ convolved with an impulse comb $p(t)$. This corresponds in the frequency domain to the multiplication of the spectrum of the single period with a comb of impulses spaced $f_\mathrm{p}$ apart.



# Continuous versus discrete Fourier transform

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

$\Longrightarrow$ By taking $n$ consecutive samples of a signal sampled with $f_\mathrm{s}$ and repeating these periodically, we obtain a signal with period $n/f_\mathrm{s}$, the spectrum of which is sampled at frequency intervals $f_\mathrm{s}/n$ and repeats itself with a period $f_\mathrm{s}$. Now both the signal and its spectrum are finite vectors of equal length.



# Properties of the Fourier transform

If

$$x(t) \quad \circ\!\!-\!\!\bullet \quad X(f) \qquad \text{and} \qquad y(t) \quad \circ\!\!-\!\!\bullet \quad Y(f)$$

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

Linearity:

$$ax(t) + by(t) \quad \circ\!\!-\!\!\bullet \quad aX(f) + bY(f)$$

Time scaling:

$$x(at) \quad \circ\!\!-\!\!\bullet \quad \frac{1}{|a|} X\left(\frac{f}{a}\right)$$

Frequency scaling:

$$\frac{1}{|a|} x\left(\frac{t}{a}\right) \quad \circ\!\!-\!\!\bullet \quad X(af)$$

Time shifting:

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

Frequency shifting:

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

Parseval's theorem (total power):

$$\int_{-\infty}^{\infty} |x(t)|^2 \mathrm{d}t \quad = \quad \int_{-\infty}^{\infty} |X(f)|^2 \mathrm{d}f$$

# Fourier transform symmetries

We call a function $x(t)$

$$\begin{aligned} \text{odd:} \quad x(-t) &= -x(t) \\ \text{even:} \quad x(-t) &= x(t) \end{aligned}$$

and $\cdot^*$ is the complex conjugate, such that $(a + \mathrm{j}b)^* = (a - \mathrm{j}b)$.

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 |

# Discrete Fourier Transform (DFT)

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

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

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

$$F \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{n-1} \end{pmatrix} = \begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ \vdots \\ X_{n-1} \end{pmatrix}$$

# Fast Fourier Transform (FFT)

$$\left(\mathcal{F}_n\{x_i\}_{i=0}^{n-1}\right)_k = \sum_{i=0}^{n-1} x_i \cdot \mathrm{e}^{2\pi \mathrm{j} \frac{ik}{n}}$$

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

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

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 $\{X_i\}_{i=0}^{n-1} = \mathcal{F}_n\{x_i\}_{i=0}^{n-1}$ have the form

$$\begin{aligned} \forall i : x_i = \quad \Re(x_i) \quad &\Longleftrightarrow \quad \forall i : X_{n-i} = \quad X_i^* \\ \forall i : x_i = \mathrm{j} \cdot \Im(x_i) \quad &\Longleftrightarrow \quad \forall i : X_{n-i} = -X_i^* \end{aligned}$$

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

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

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

$$x_i = x'_i + \mathrm{j} \cdot x''_i$$

and decomposing its output as

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

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.

# Fast complex multiplication

The calculation of the product

$$(a + \mathrm{j}b) \cdot (c + \mathrm{j}d) = (ac - bd) + \mathrm{j}(ad + bc)$$

of two complex numbers costs four real-valued multiplications and two additions. On small processors, where multiplication is significantly slower than addition, the calculation

$$(\alpha - \beta) + \mathrm{j}(\alpha + \gamma) \qquad \text{with} \qquad \begin{aligned} \alpha &= a(c + d) \\ \beta &= d(a + b) \\ \gamma &= c(b - a) \end{aligned}$$

provides the same result with just three multiplications, in exchange for using three more (five) additions.

# Polynomial representation of sequences

Example of polynomial multiplication:

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

Convolution:

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

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

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

Convolution of sequences then becomes polynomial multiplication:

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

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

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

Example of polynomial division:

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

$$\begin{array}{r|l}
 & 1 \;+\; av \;+\; a^2v^2 \;+\; \cdots \\
\cline{2-2}
1-av & 1 \\
 & \underline{1 \;-\; av} \\
 & \quad av \\
 & \quad \underline{av \;-\; a^2v^2} \\
 & \qquad a^2v^2 \\
 & \qquad \underline{a^2v^2 \;-\; a^3v^3} \\
 & \qquad\quad \cdots
\end{array}$$

Rational functions (quotients of two polynomials) can provide a convenient closed-form representations for infinitely-long exponential sequences, as they are produced by IIR filters.

# The $z$-transform

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

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

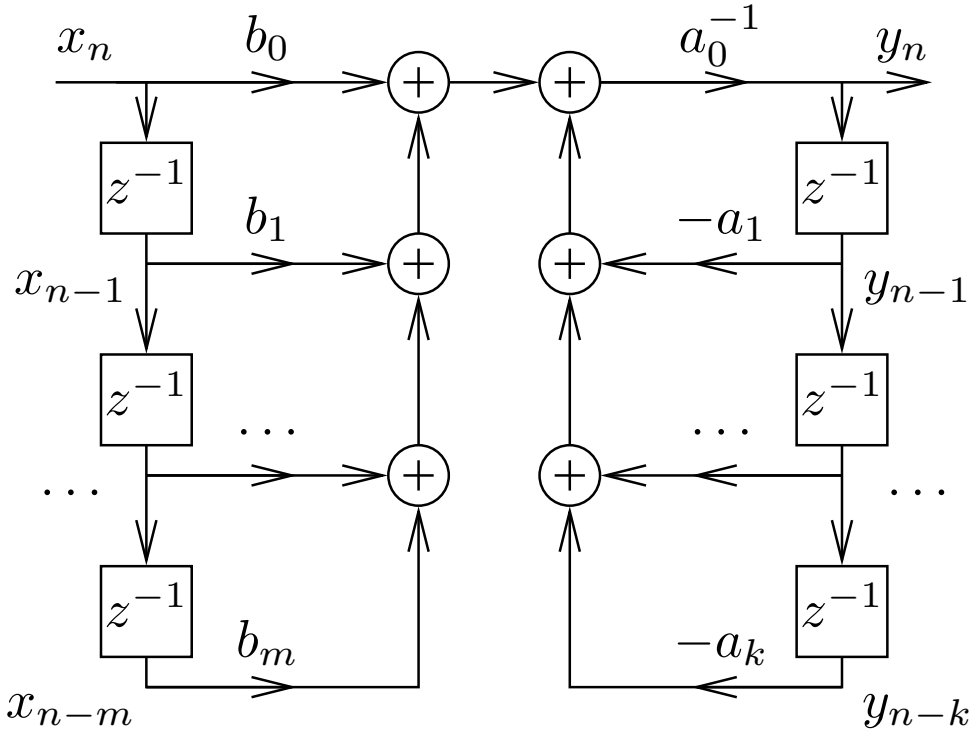
It defines for each sequence a continuous complex surface over the complex plane. The $z$-transform is a generalization of the Fourier transform, which it contains on the complex unit circle ($|z| = 1$):

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

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

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

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




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

which can also be written as

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

$H(z)$ will have $m$ zeros and $n$ poles at non-zero location in the $z$ plane, plus $n - m$ zeros (if $n > m$) or $m - n$ poles (if $m > n$) at $z = 0$.

This function can be converted into the form

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

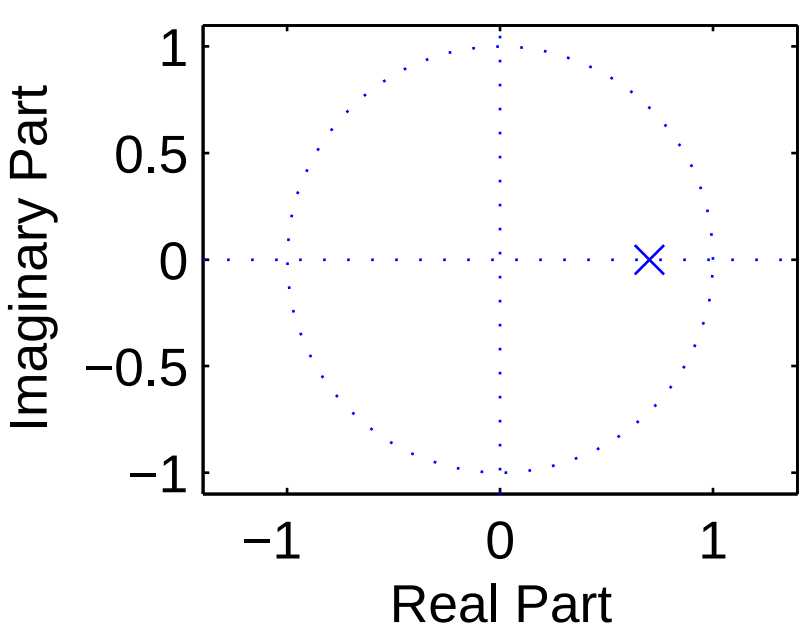
where the $c_l$ are the non-zero zeros and the $d_l$ the non-zero poles of $H(z)$. Except for a constant factor, $H(z)$ is entirely characterized by the position of these zeros and poles.

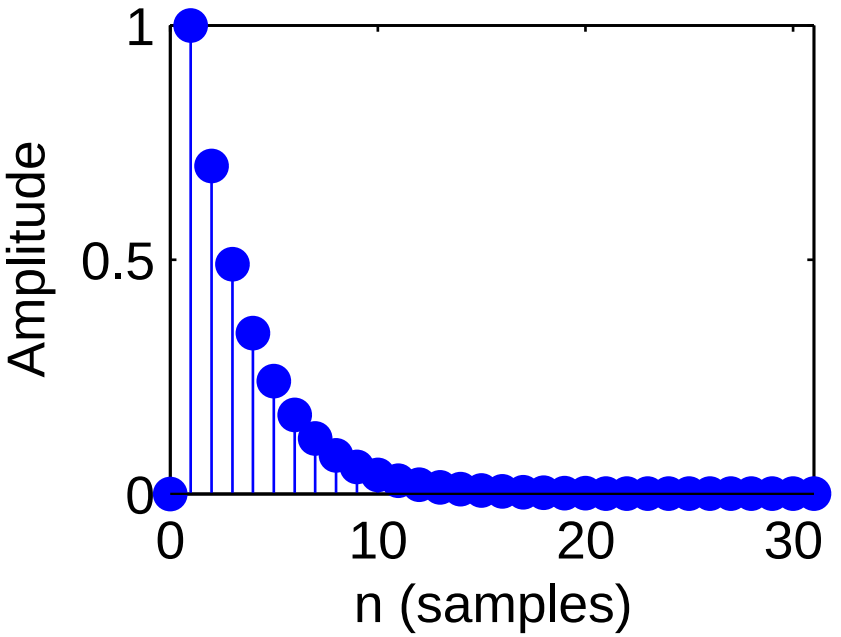
As with the Fourier transform, convolution in the time domain corresponds to complex multiplication in the $z$-domain:

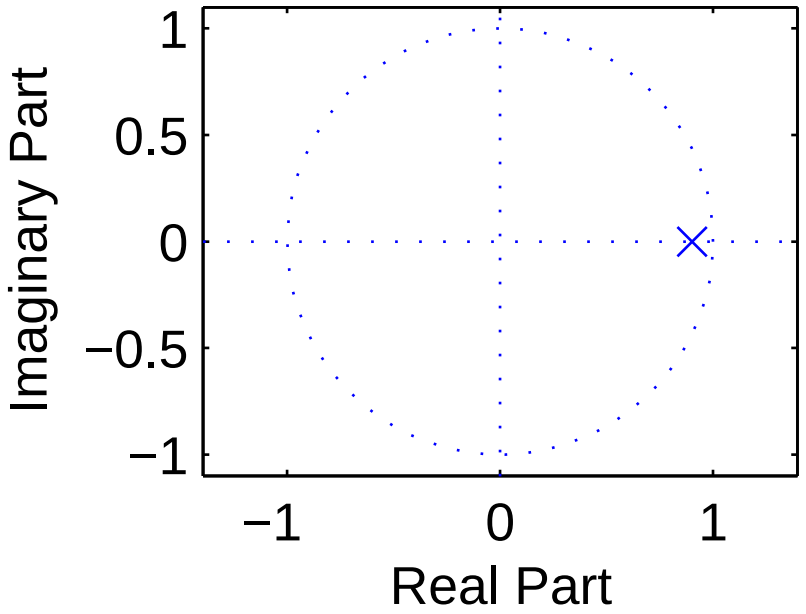
$$\{x_n\} \circ\!\!-\!\!\bullet\ X(z),\ \{y_n\} \circ\!\!-\!\!\bullet\ Y(z) \Rightarrow \{x_n\} * \{y_n\} \circ\!\!-\!\!\bullet\ X(z) \cdot Y(z)$$

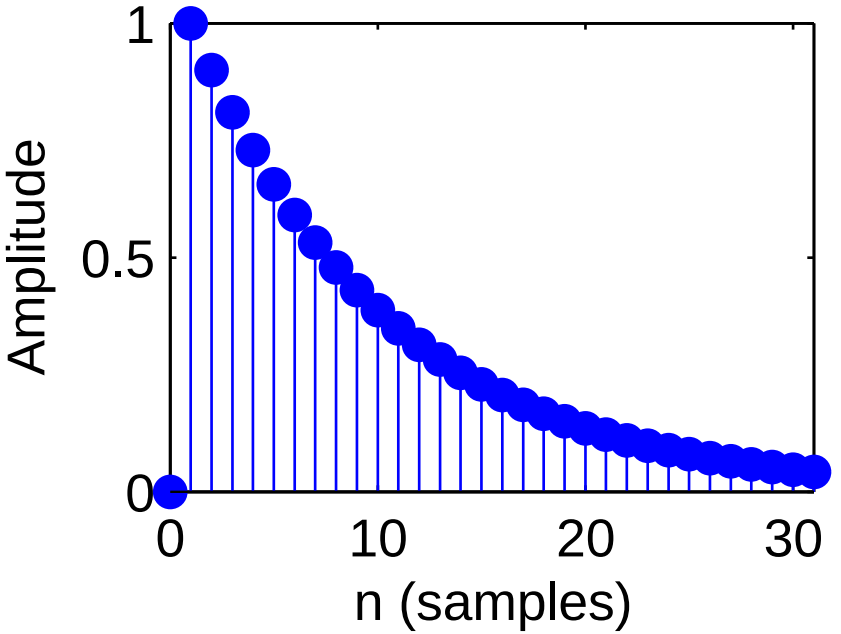
Delaying a sequence by one corresponds in the $z$-domain to multiplication with $z^{-1}$:

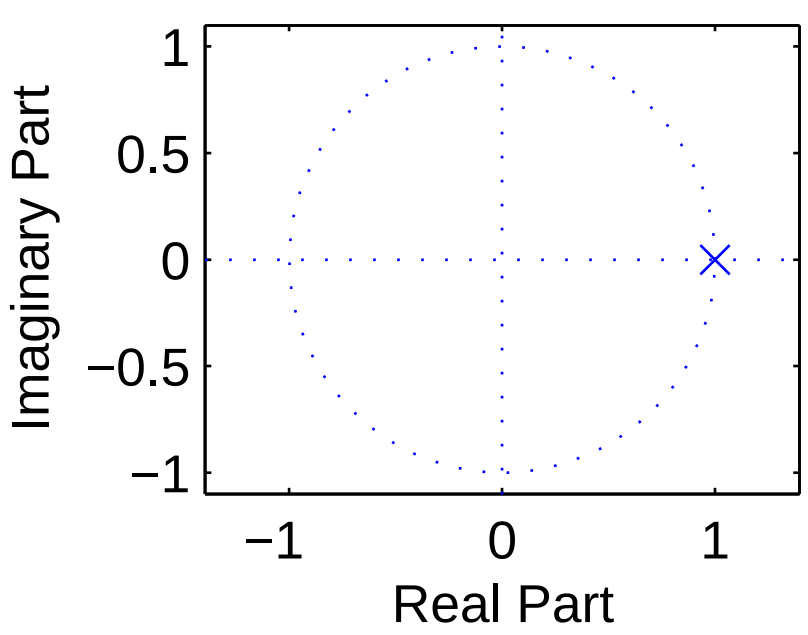
$$\{x_{n-\Delta n}\} \circ\!\!-\!\!\bullet\ X(z) \cdot z^{-\Delta n}$$

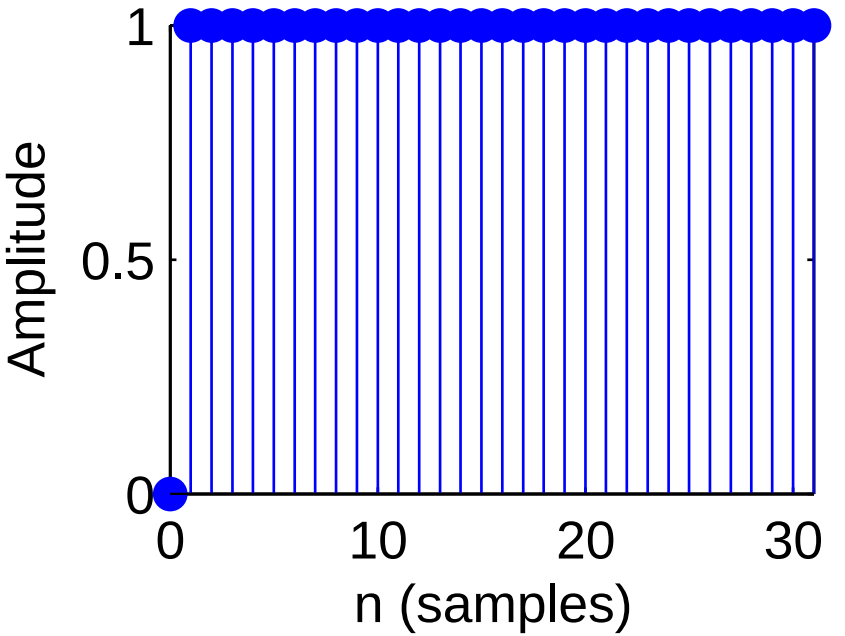

Imaginary Part
1
0.5
0
−0.5
−1
−1
0
1
Real Part

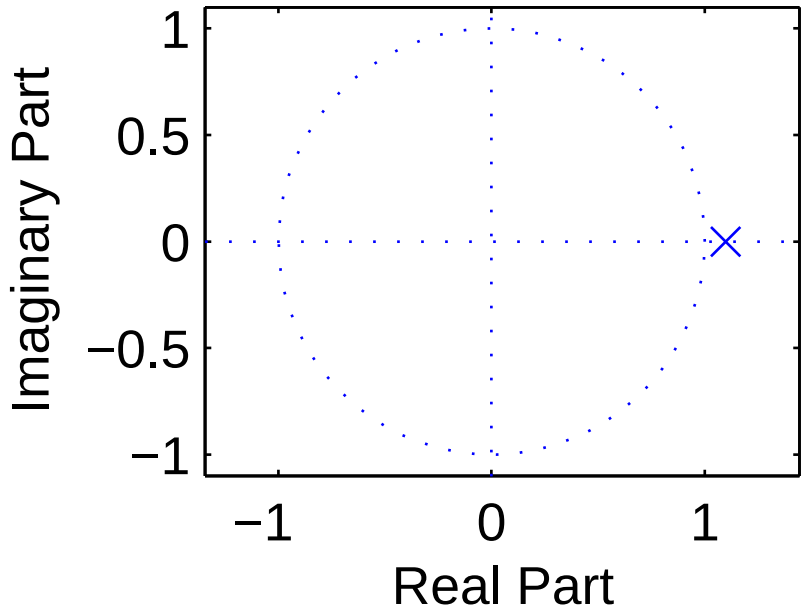

Amplitude
1
0.5
0
0
10
20
30
n (samples)

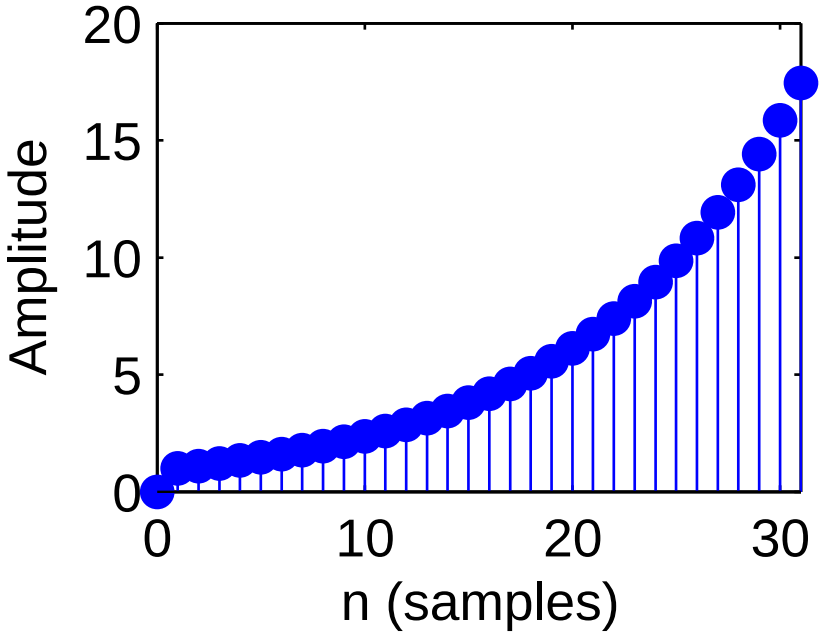

Imaginary Part
1
0.5
0
−0.5
−1
−1
0
1
Real Part

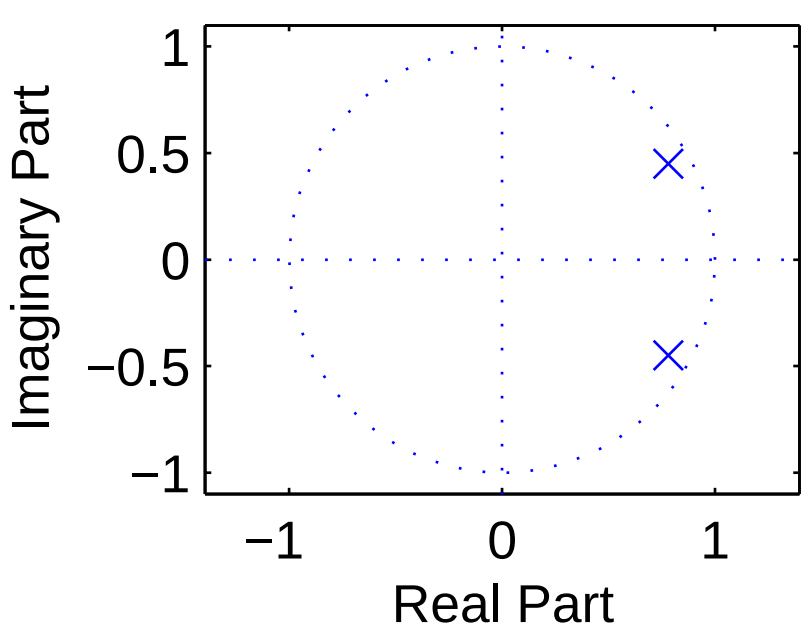

Amplitude
1
0.5
0
0
10
20
30
n (samples)

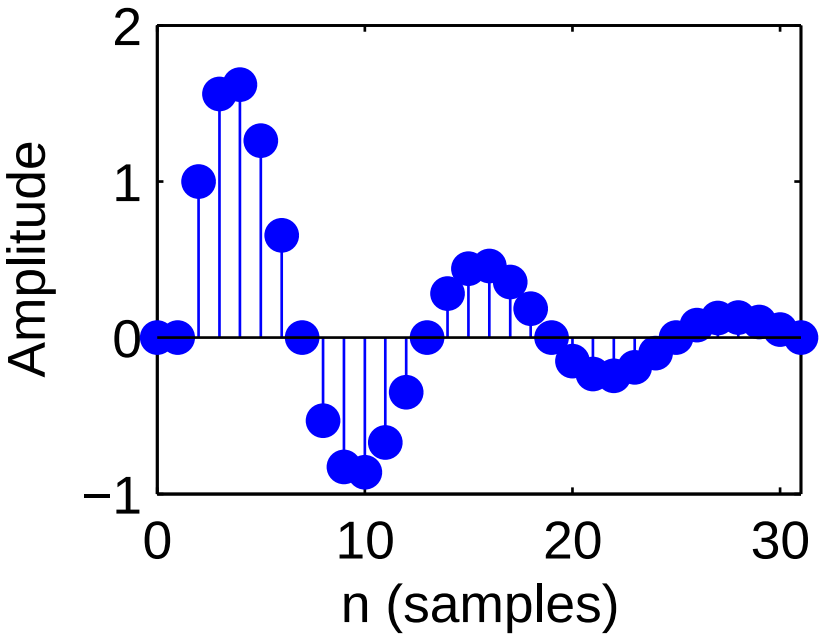

Imaginary Part
Real Part
1
0.5
0
−0.5
−1
−1
0
1

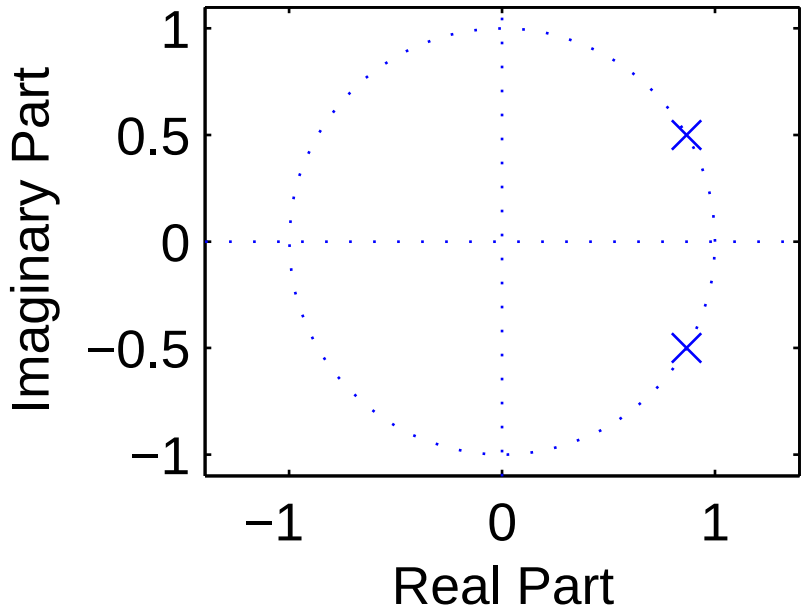

Amplitude
n (samples)
1
0.5
0
0
10
20
30

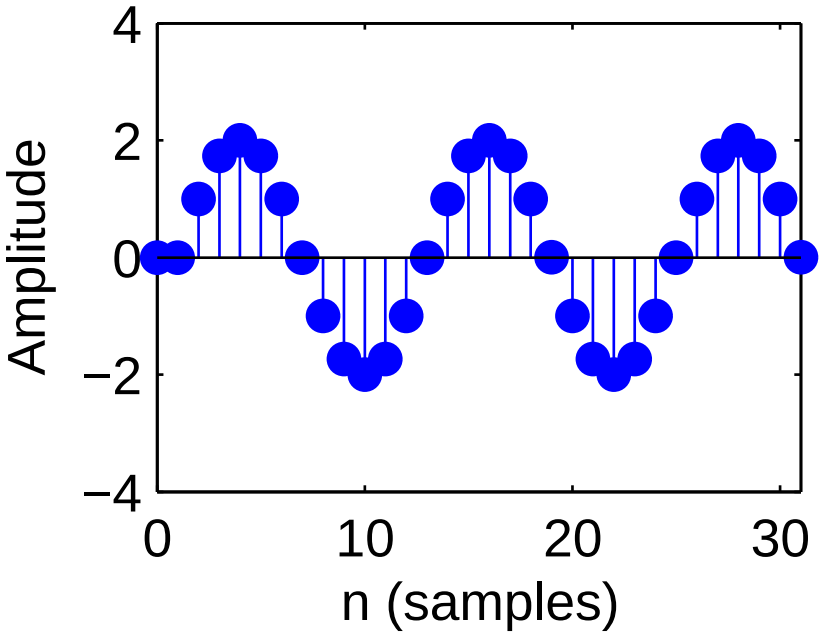

Imaginary Part
Real Part
1
0.5
0
−0.5
−1
−1
0
1

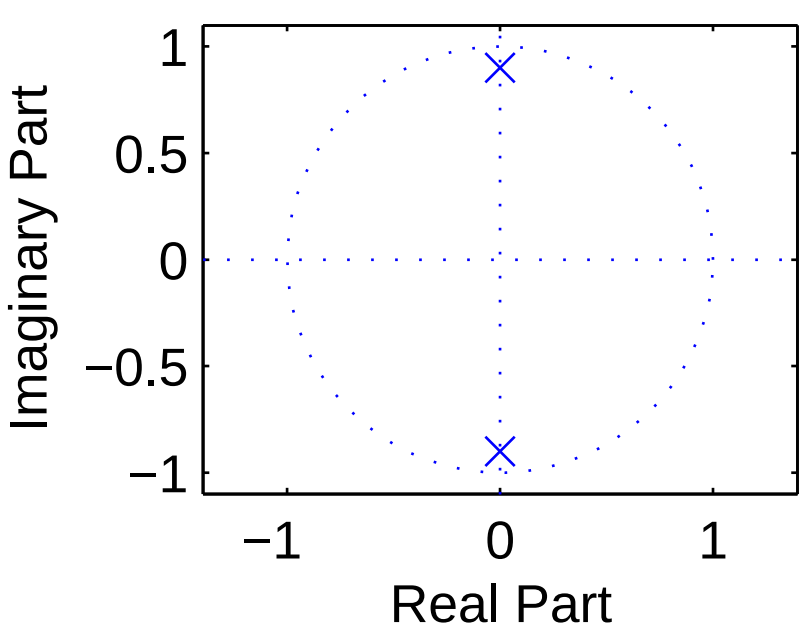

Amplitude
n (samples)
20
15
10
5
0
0
10
20
30

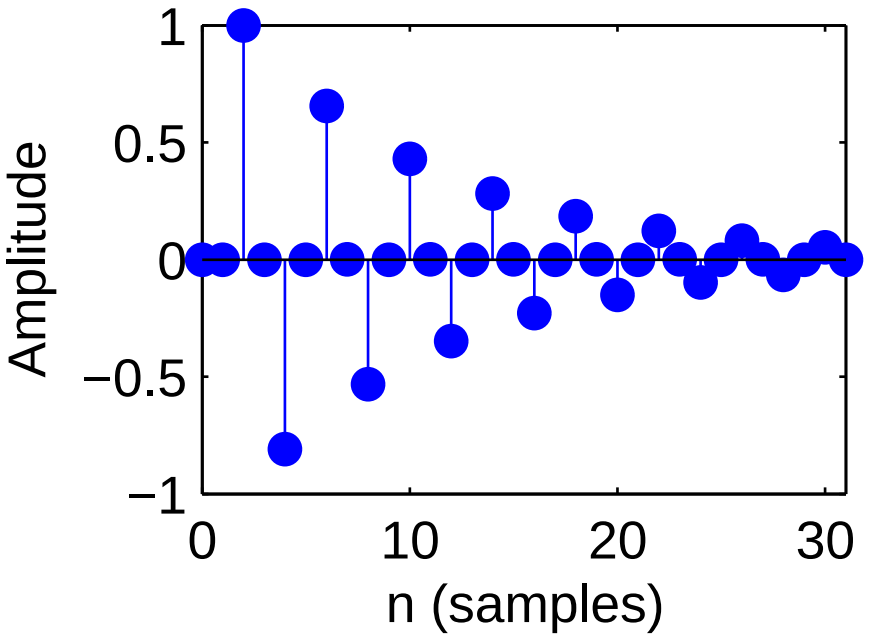

Imaginary Part
Real Part
1
0.5
0
−0.5
−1
−1
0
1

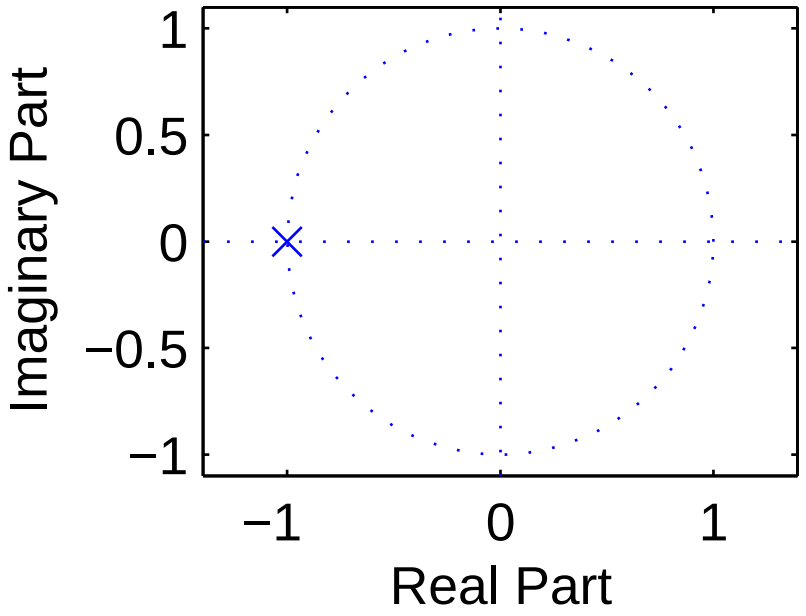

Amplitude
n (samples)
2
1
0
−1
0
10
20
30

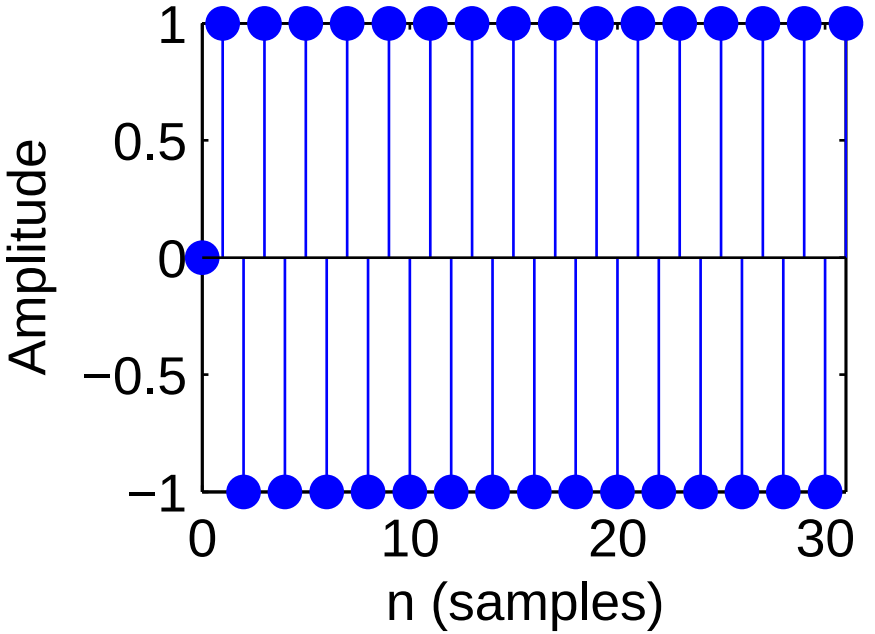

Imaginary Part
Real Part
1
0.5
0
−0.5
−1
−1
0
1


Amplitude
n (samples)
4
2
0
−2
−4
0
10
20
30

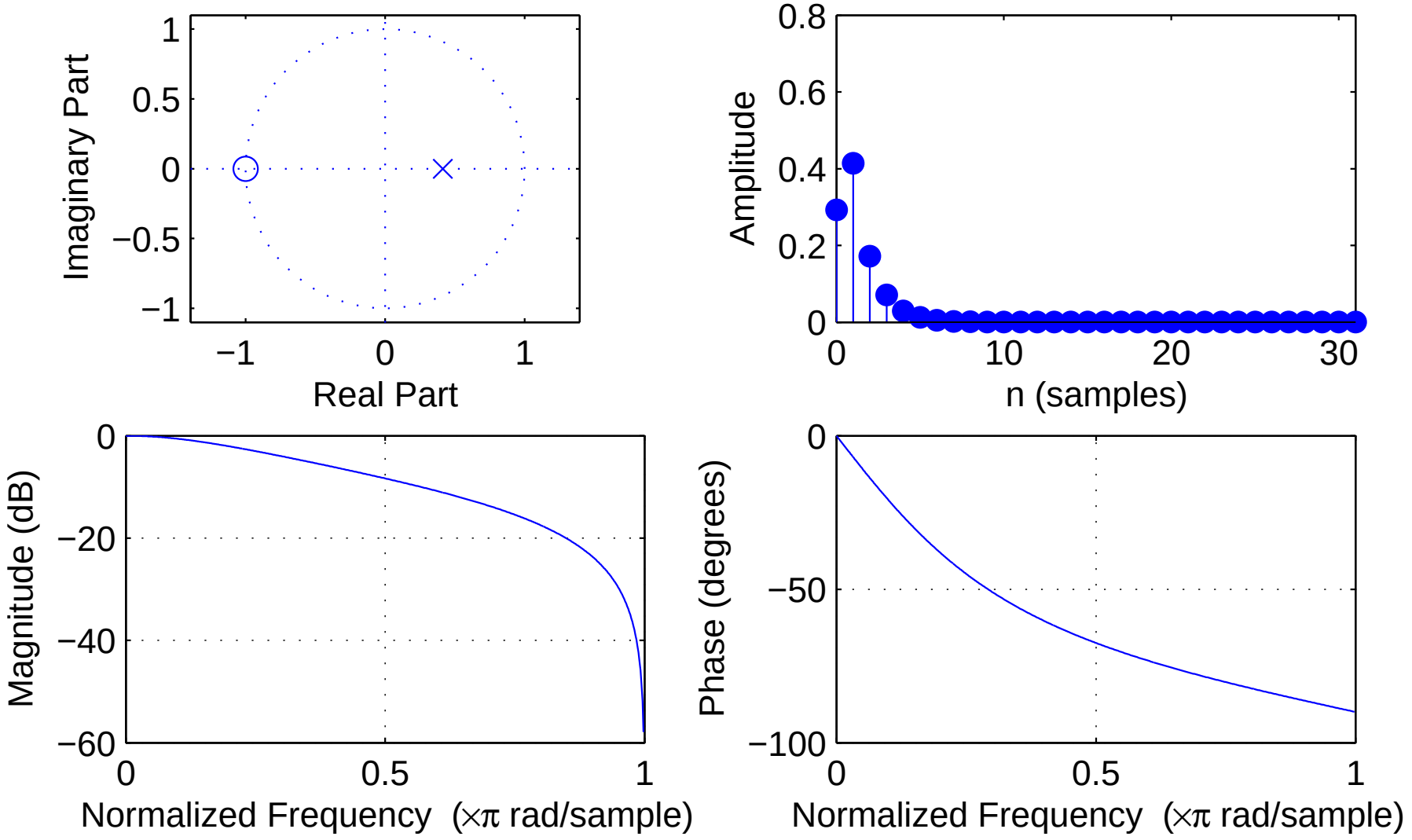

Imaginary Part
Real Part
1
0.5
0
−0.5
−1
−1
0
1


Amplitude
n (samples)
1
0.5
0
−0.5
−1
0
10
20
30


Imaginary Part
Real Part
1
0.5
0
−0.5
−1
−1
0
1

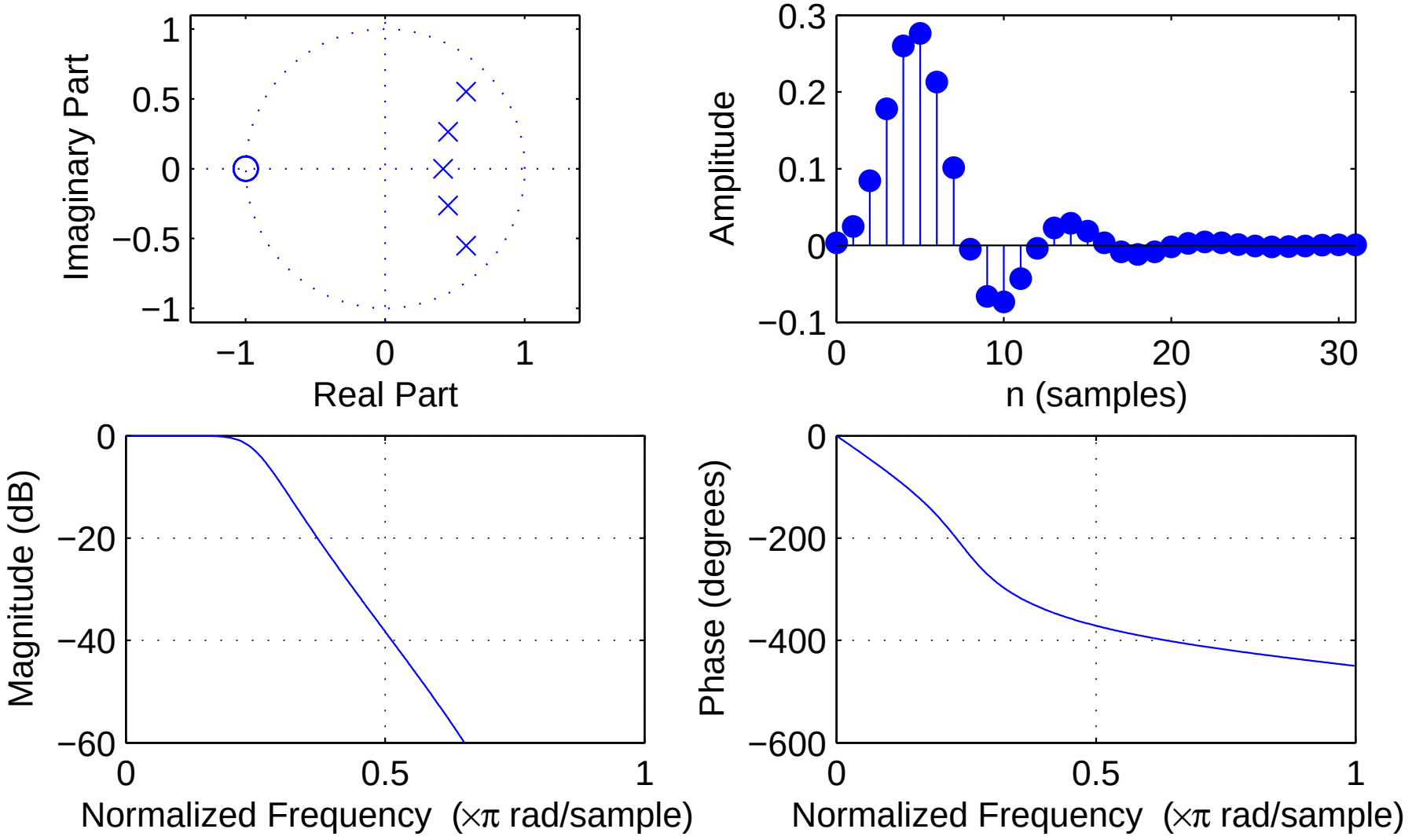

Amplitude
n (samples)
1
0.5
0
−0.5
−1
0
10
20
30

# Butterworth filter design example



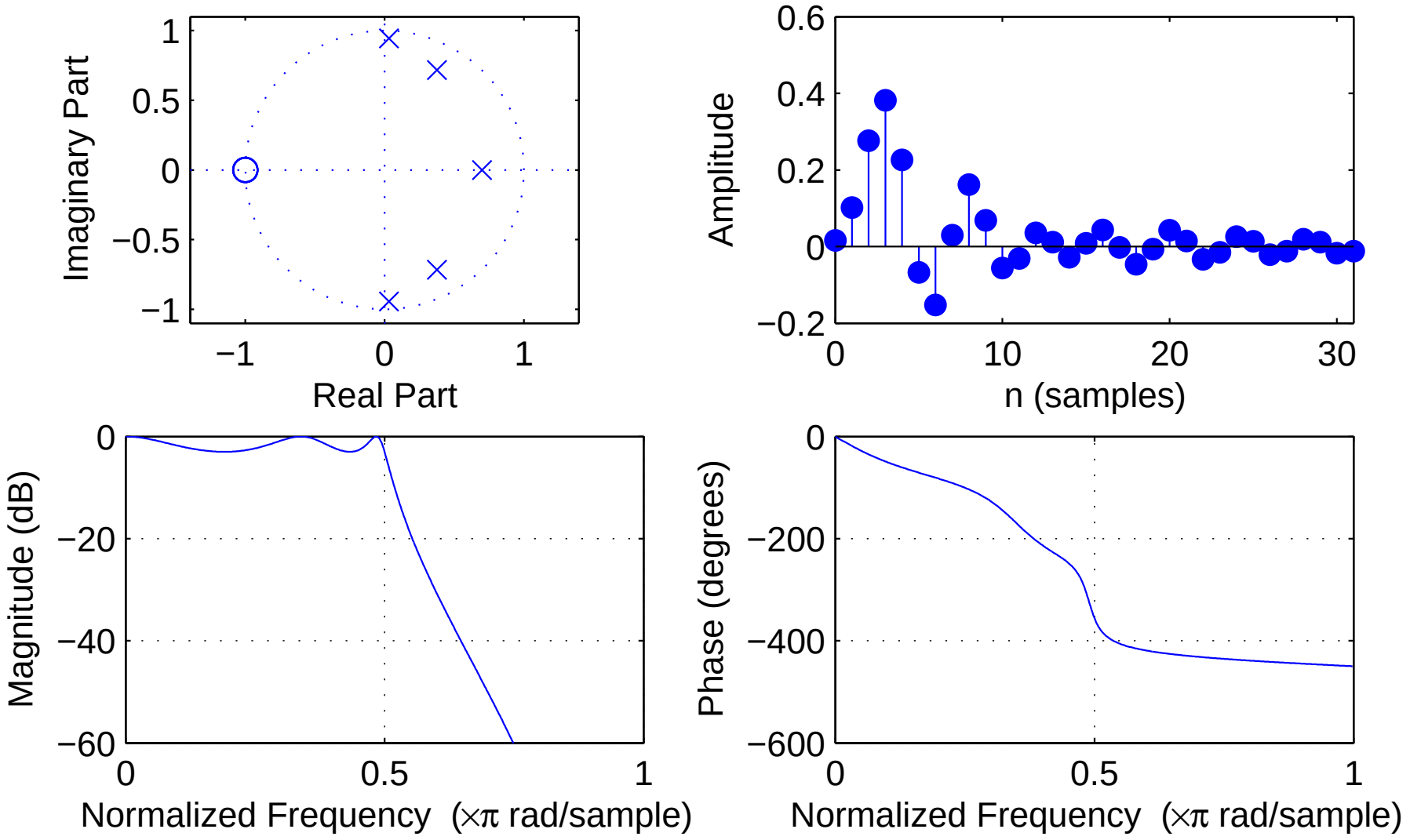

order: 1, cutoff frequency ($-3$ dB): $0.25 \times f_s/2$

# Butterworth filter design example



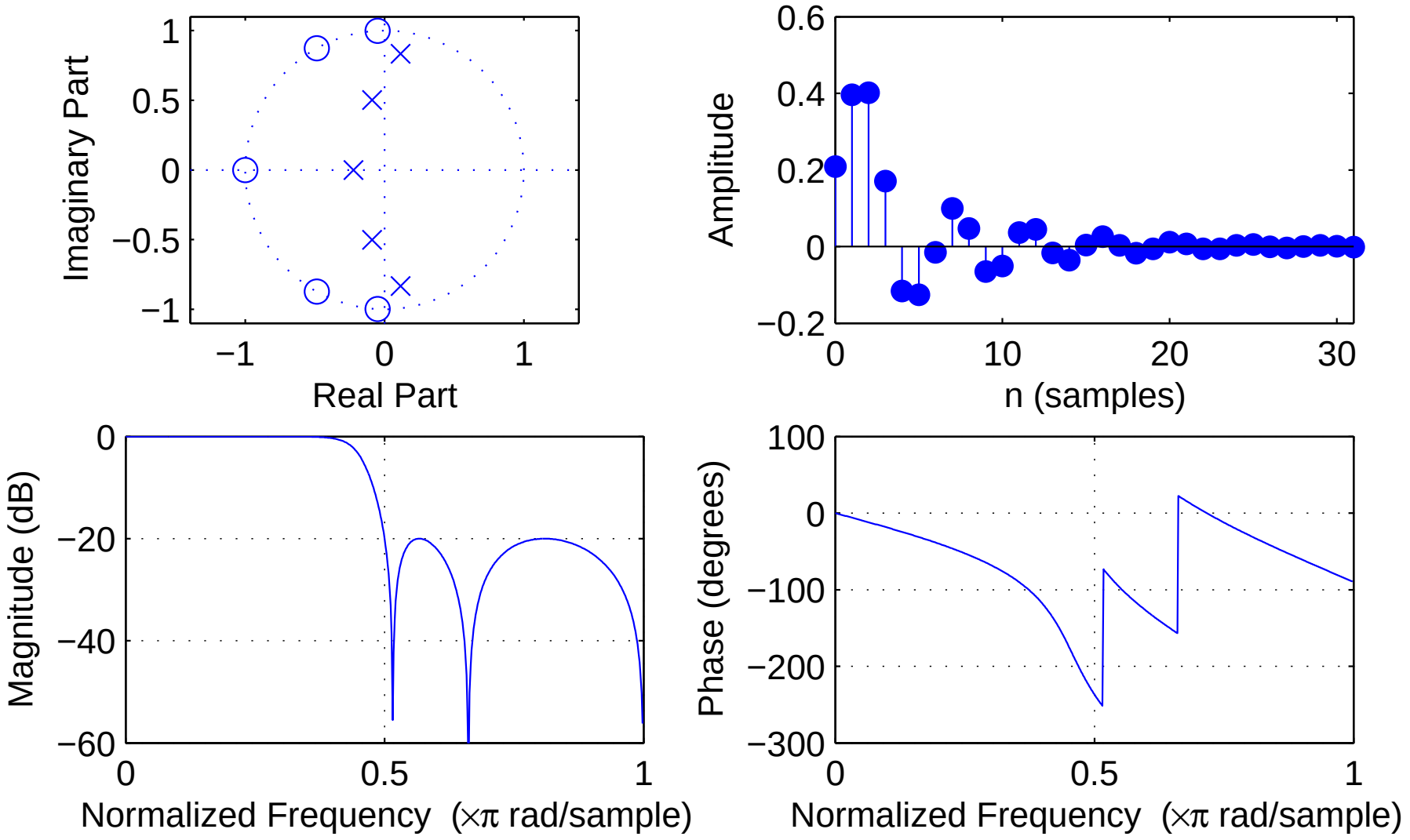

order: 5, cutoff frequency ($-3$ dB): $0.25 \times f_s/2$

# Chebyshev type I filter design example



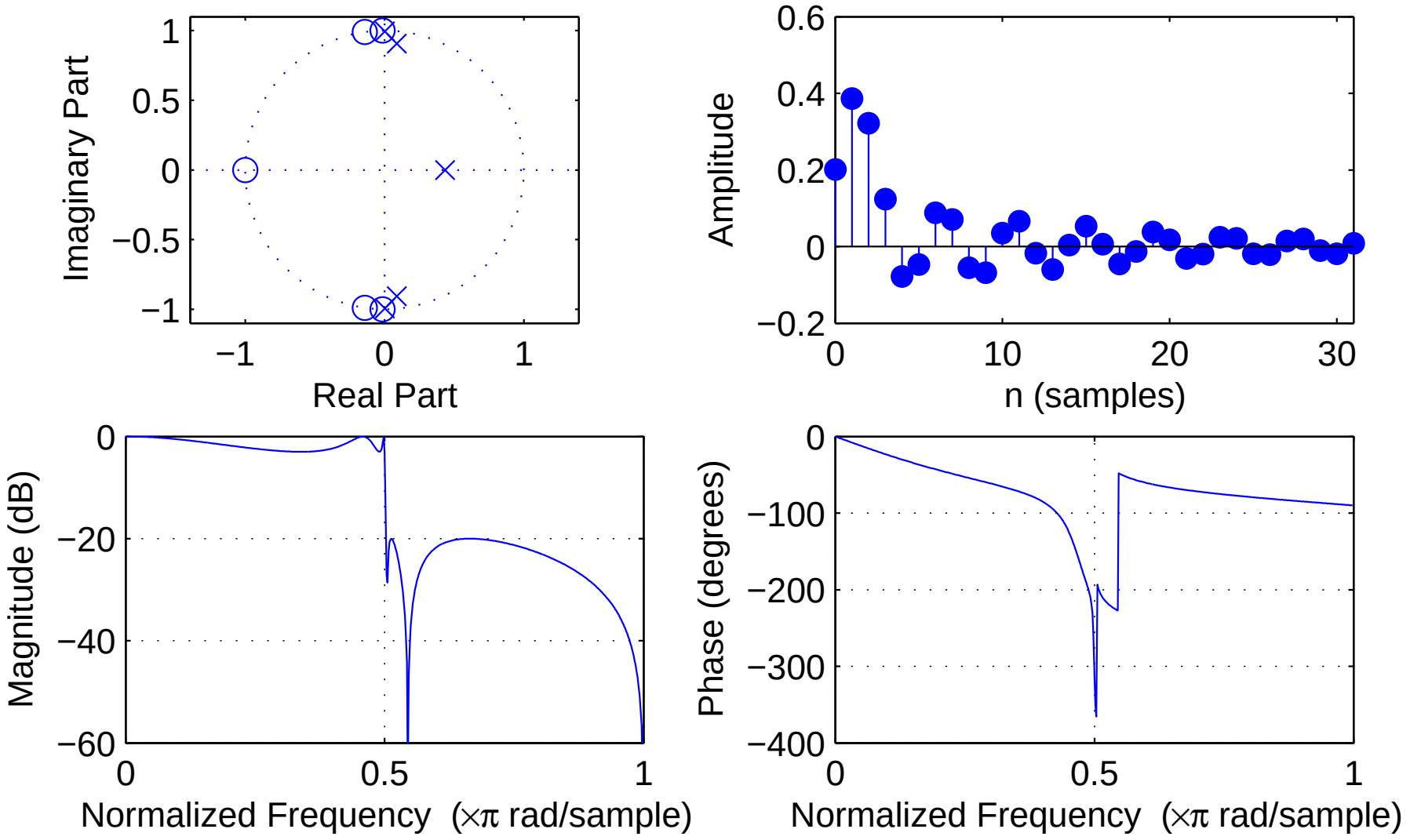

order: 5, cutoff frequency: $0.5 \times f_s/2$, pass-band ripple: $-3$ dB

# 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

# IIR Filter design techniques

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.

$\longrightarrow$ 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.

The designer can then trade off conflicting goals such as a small transition band, a low order, a low ripple amplitude, or even an 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

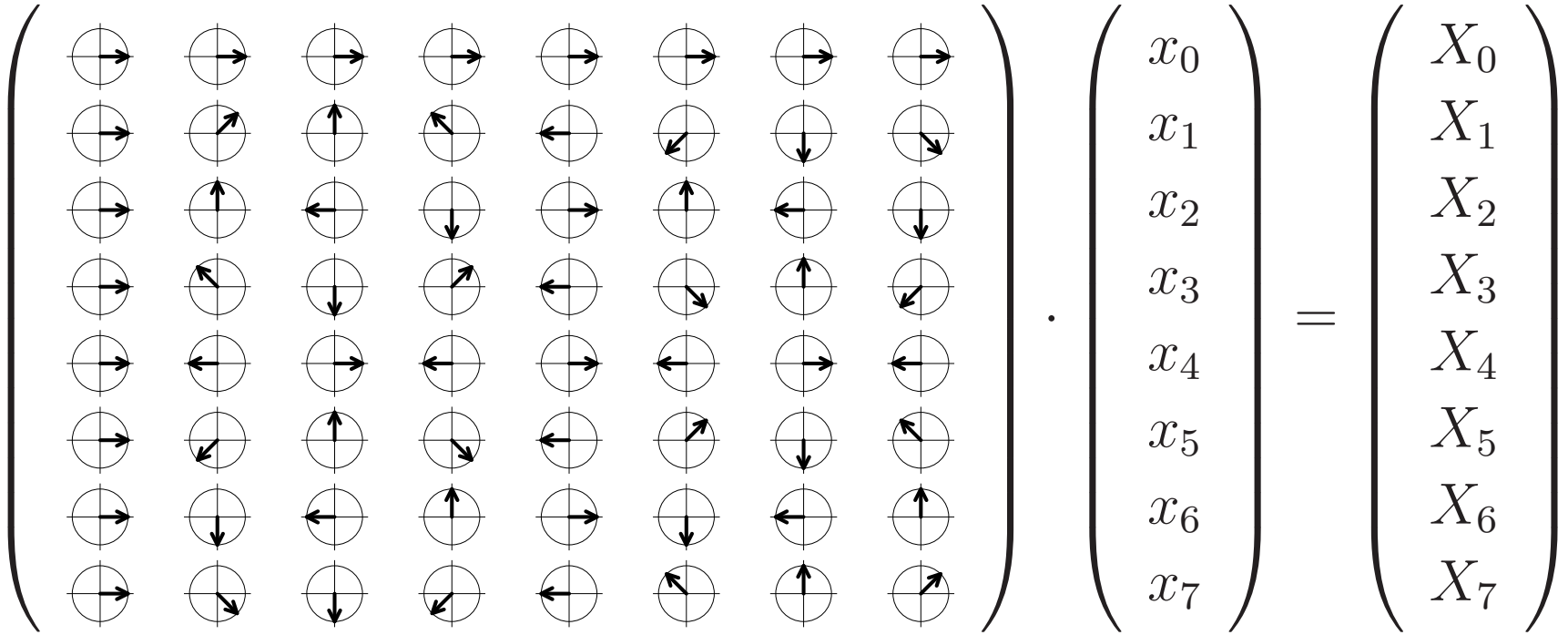
Distribute the gain error uniformly throughout the stopband (equiripples) and drop off monotonically in the passband.

## 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.

All these filter design techniques are implemented in the MATLAB Signal Processing Toolbox in the functions `butter`, `cheby1`, `cheby2`, and `ellip`, which 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. The commands `sptool` and `fdatool` provide interactive GUIs to design digital filters.

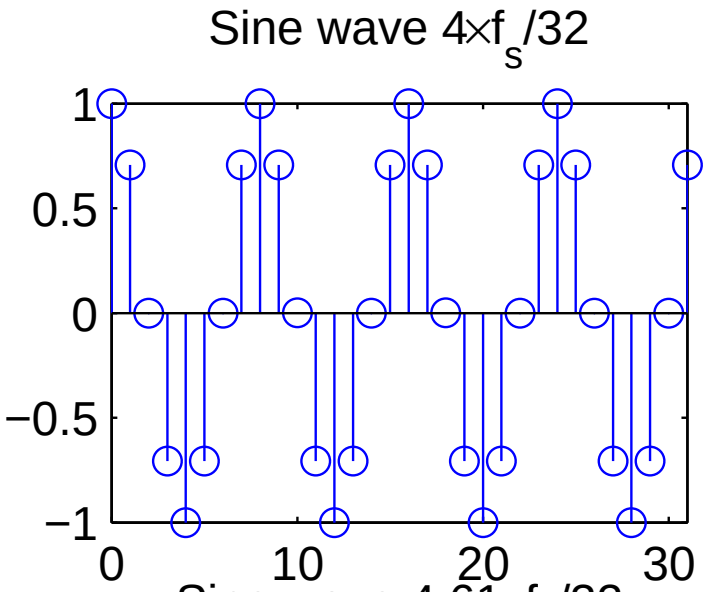
# Discrete Fourier Transform visualized

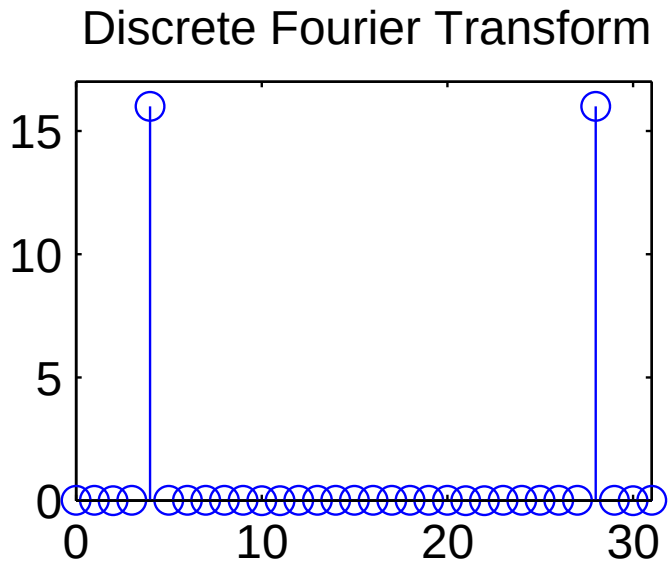


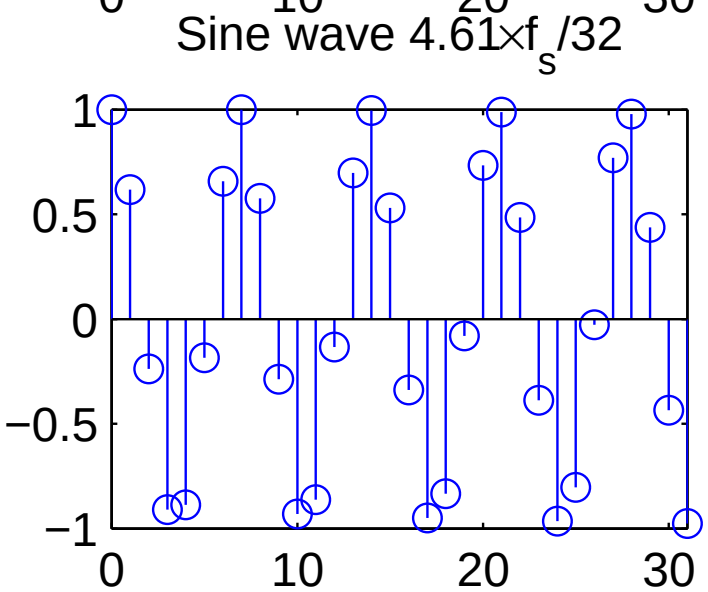
The $n$-point DFT of a signal $\{x_i\}$ sampled at frequency $f_\mathrm{s}$ contains in the elements $X_0$ to $X_{n/2}$ of the resulting frequency-domain vector the frequency components $0$, $f_\mathrm{s}/n$, $2f_\mathrm{s}/n$, $3f_\mathrm{s}/n$, $\ldots$, $f_\mathrm{s}/2$, and contains in $X_{n-1}$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

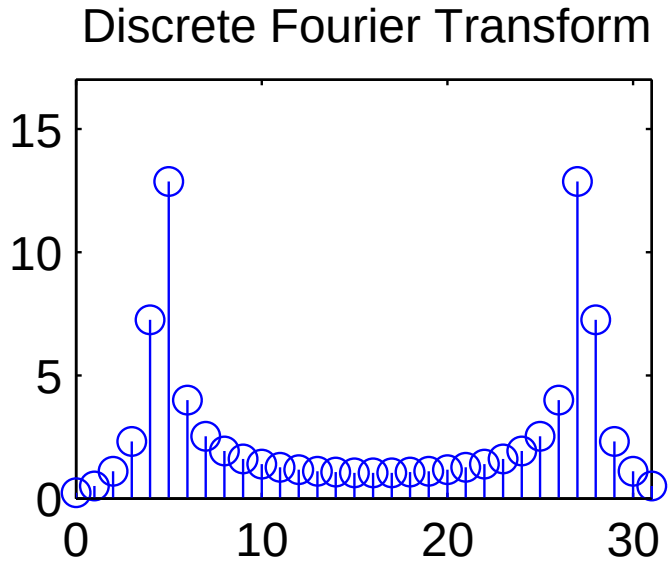
Why is there no phase information recovered at $f_\mathrm{s}/2$?

# Spectral estimation



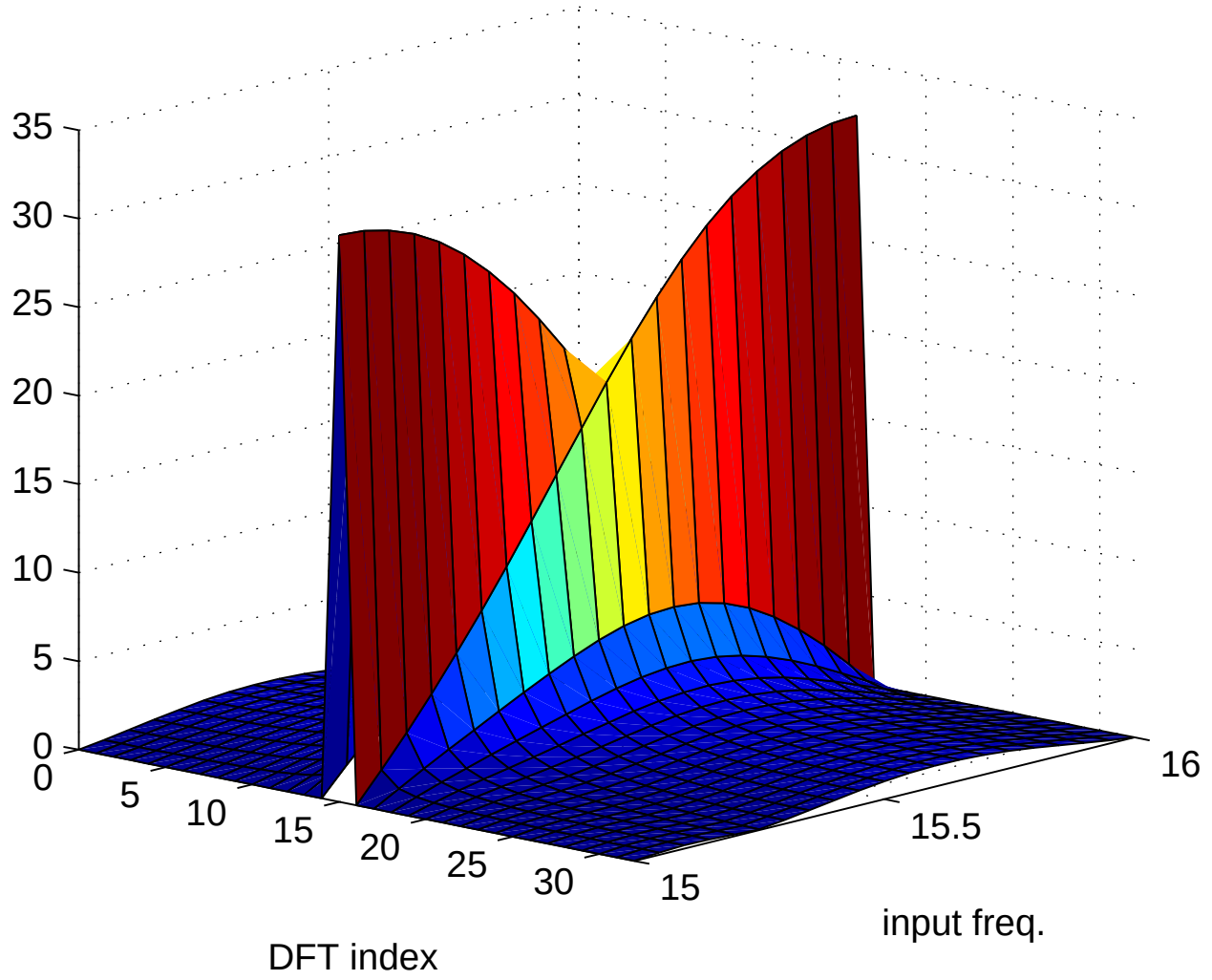
We introduced the DFT as a special case of the continuous Fourier transform, where the input is sampled *and periodic*.

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 ($f_\mathrm{s}/n$) 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.

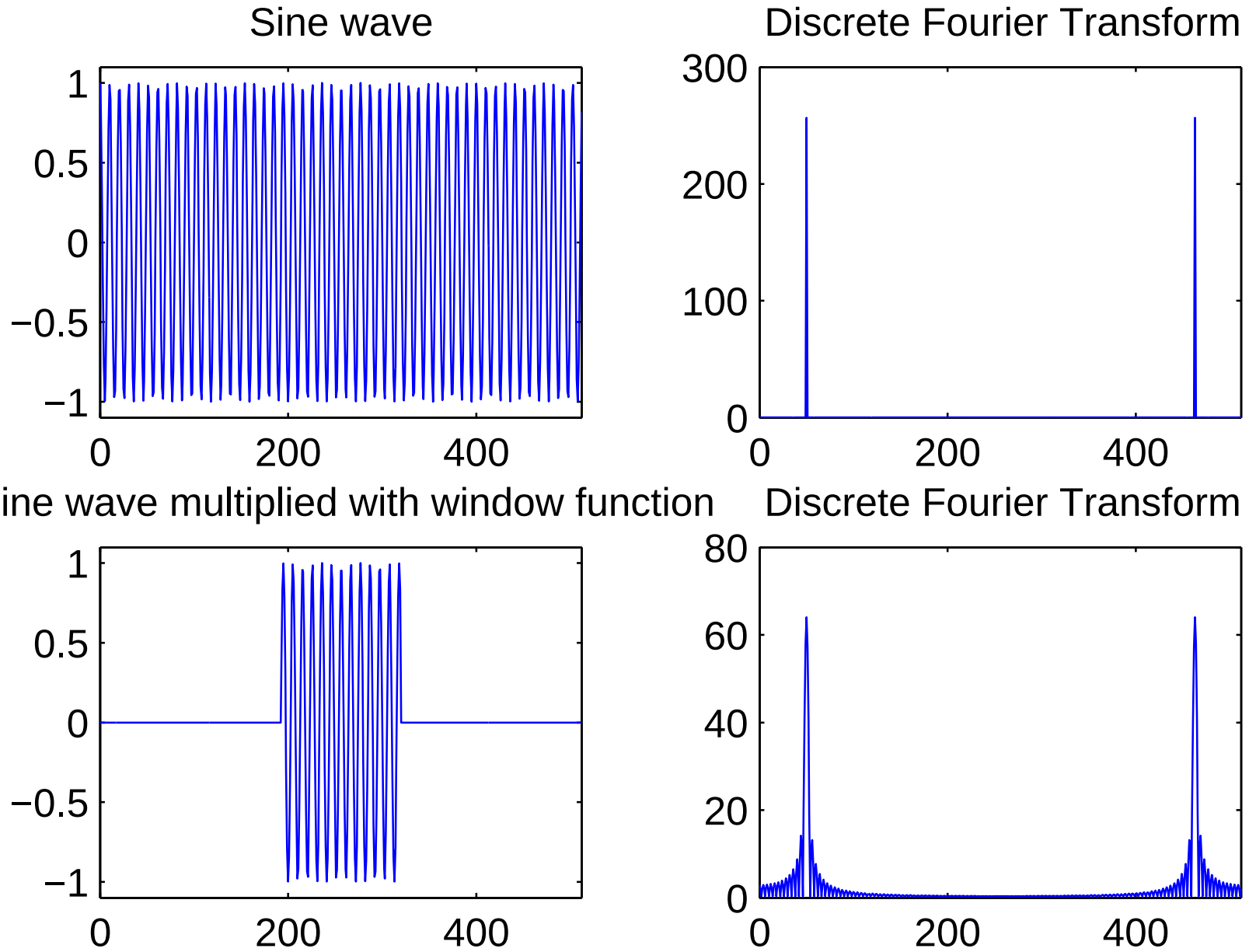
Sine waves with other frequencies that do not match exactly one of the output frequency bins of the DFT are still represented by a peak at the output bin that represents the nearest integer multiple of the DFT's base frequency. However, such a peak is distorted in two ways:

- $\longrightarrow$ Its amplitude is lower (down to 63.7%)
- $\longrightarrow$ 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 associates 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).

# Windowing


Sine wave
Discrete Fourier Transform
Sine wave multiplied with window function
Discrete Fourier Transform
1
0.5
0
−0.5
−1
0
200
400
300
200
100
0
80
60
40
20
0
200
400

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

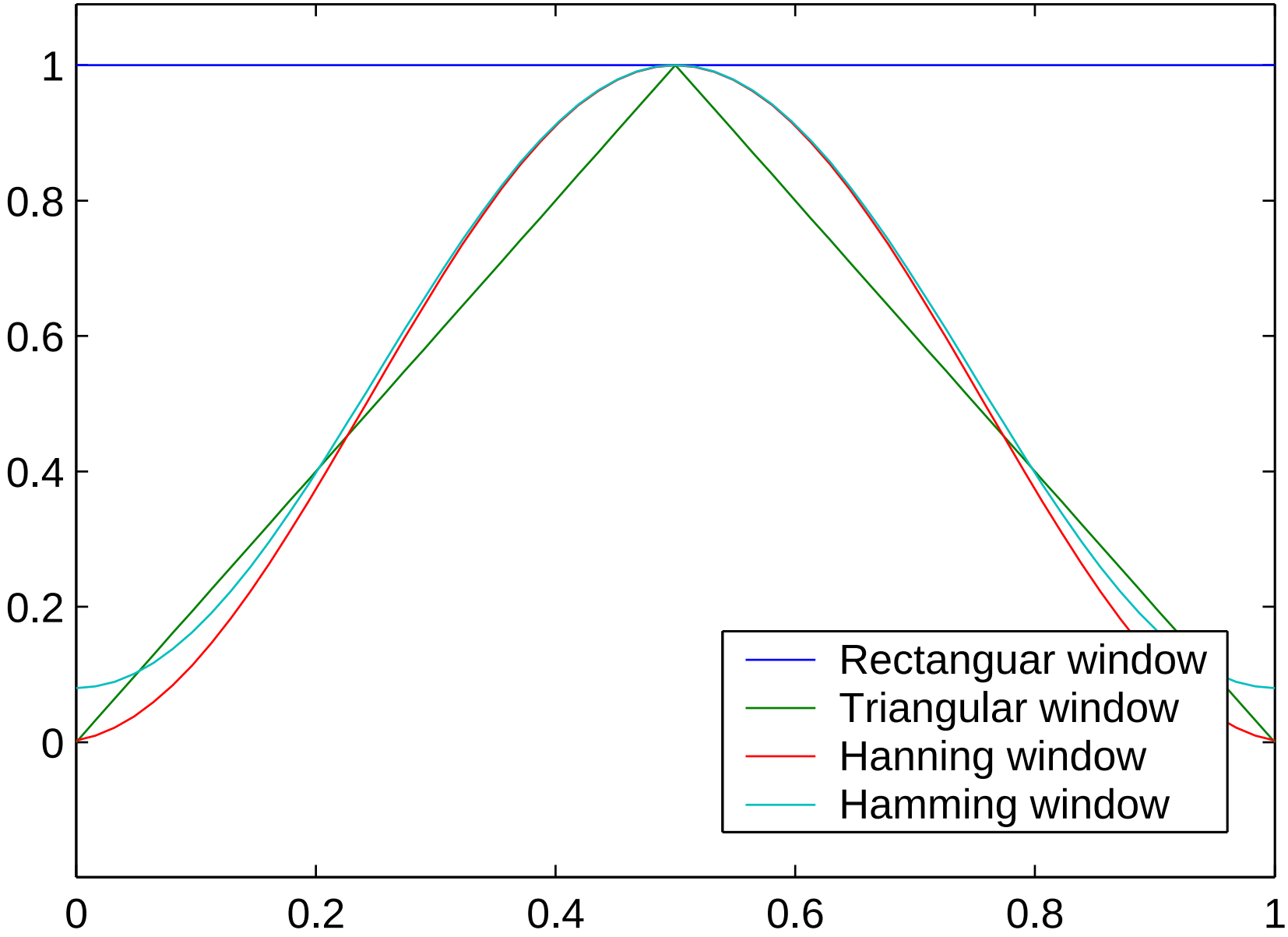
The operation of cutting a sequence with the size of the DFT input vector out of a longer original signal (the one whose continuous Fourier spectrum we try to estimate) is equivalent to multiplying this signal with a rectangular function. This destroys all information and continuity outside the "window"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

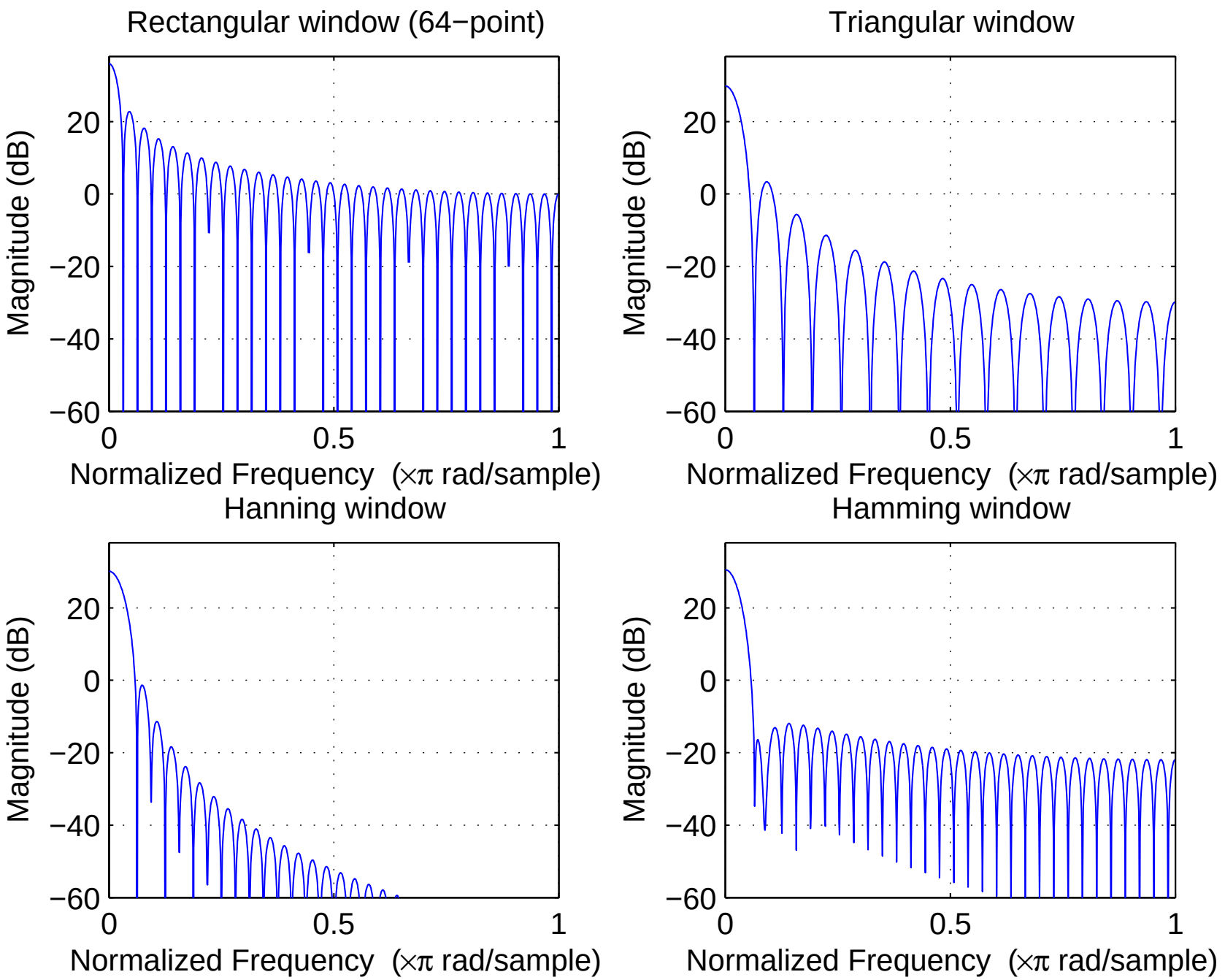
The subsequent interpretation of this window as a periodic sequence by the DFT leads to sampling of this convolution result (sampling meaning multiplication with a Dirac comb whose impulses are spaced $f_\mathrm{s}/n$ apart).

Where the window length was an exact multiple of the original signal period, sampling of the $\sin(\pi f T)/(\pi f T)$ 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].


Rectangular window (64−point)
Magnitude (dB)
20
0
−20
−40
−60
0
0.5
1
Normalized Frequency (×π rad/sample)
Triangular window
Magnitude (dB)
20
0
−20
−40
−60
0
0.5
1
Normalized Frequency (×π rad/sample)
Hanning window
Magnitude (dB)
20
0
−20
−40
−60
0
0.5
1
Normalized Frequency (×π rad/sample)
Hamming window
Magnitude (dB)
20
0
−20
−40
−60
0
0.5
1
Normalized Frequency (×π rad/sample)

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):

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

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

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

**Hamming window**:

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

**Exercise 5** Explain the difference between the DFT, FFT, and FFTW.

**Exercise 6** Push-button telephones use a combination of two sine tones to signal, which button is currently being pressed:

| | 1209 Hz | 1336 Hz | 1477 Hz | 1633 Hz |
|---|---|---|---|---|
| 697 Hz | 1 | 2 | 3 | A |
| 770 Hz | 4 | 5 | 6 | B |
| 852 Hz | 7 | 8 | 9 | C |
| 941 Hz | * | 0 | # | D |

(a) You receive a digital telephone signal with a sampling frequency of 8 kHz. You cut a 256-sample window out of this sequence, multiply it with a windowing function and apply a 256-point DFT. What are the indices where the resulting vector $(X_0, X_1, \ldots, X_{255})$ will show the highest amplitude if button $\boxed{9}$ was pushed at the time of the recording?

(b) Use MATLAB to determine, which button sequence was typed in the touch tones recorded in

`http://www.cl.cam.ac.uk/Teaching/2003/DigSigProc/touchtone.wav`

**Exercise 7** Draw the direct form II block diagrams of the causal infinite-impulse response filters described by the following $z$-transforms and write down a formula describing their time-domain impulse responses:

(a) $H(z) = \dfrac{1}{1 - \frac{1}{2}z^{-1}}$

(b) $H'(z) = \dfrac{1 - \frac{1}{4^4}z^{-4}}{1 - \frac{1}{4}z^{-1}}$

(c) $H''(z) = \dfrac{1}{2} + \dfrac{1}{4}z^{-1} + \dfrac{1}{2}z^{-2}$

**Exercise 8** (a) Perform the polynomial division of the rational function given in exercise 7 (a) until you have found the coefficient of $z^{-5}$ in the result.

(b) Perform the polynomial division of the rational function given in exercise 7 (b) until you have found the coefficient of $z^{-10}$ in the result.

(c) Has one of the filters in exercise 7 actually a finite impulse response, and if so, what is its $z$-transform?

# Zero padding increases DFT resolution

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

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

The left plot shows the DFT of the windowed sequence

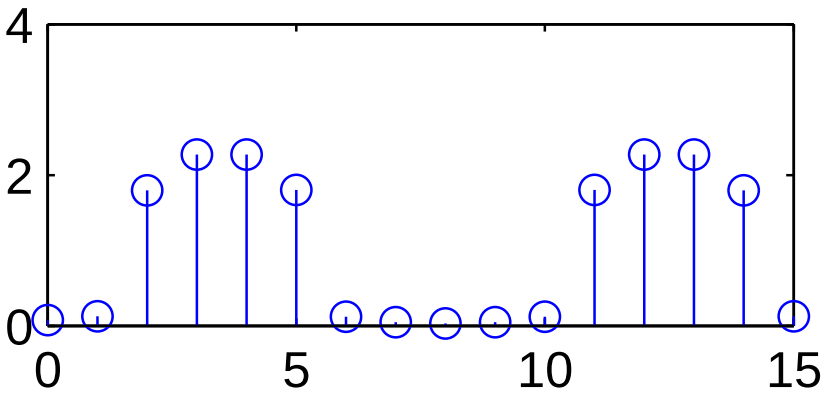
$$x_i = s_i \cdot w_i, \qquad i \in \{0, \ldots, 15\}$$

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

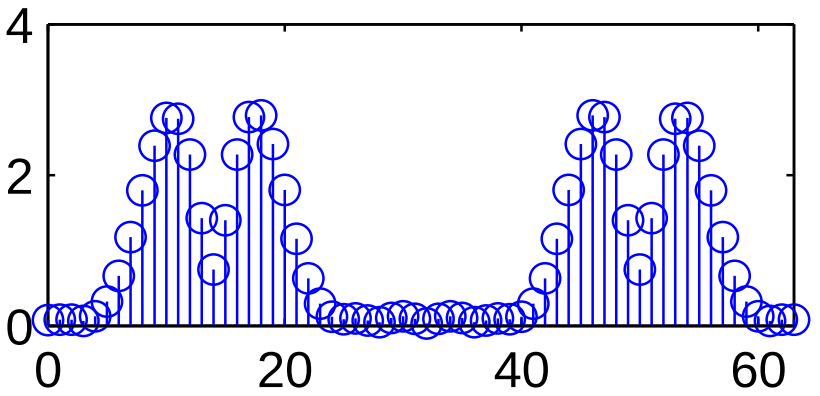
$$x'_i = \begin{cases} s_i \cdot w_i, & i \in \{0, \ldots, 15\} \\ 0, & i \in \{16, \ldots, 63\} \end{cases}$$

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

DFT without zero padding




DFT with 48 zeros appended to window



Applying the discrete Fourier transform to an $n$-element long real-valued sequence leads to a spectrum consisting of only $n/2+1$ discrete frequencies.

Since the resulting 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 smoothly down to zero at the window boundaries, appending further zeros outside the window will not distort the signal further.

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.

Note that zero padding does not add any additional information to the signal. The spectrum has already been "low-pass filtered" by being convolved with the spectrum of the windowing function. Zero padding in the time domain merely samples this spectrum blurred by the windowing step at a higher resolution, thereby making it easier to visually distinguish spectral lines and locating their peak more precisely.

# Window-based design of FIR filters

Recall that the ideal continuous low-pass filter with cut-off frequency $f_\mathrm{c}$ has the frequency characteristic

$$H(f) = \begin{cases} 1 & \text{if } -f_\mathrm{c} < f \leq f_\mathrm{c} \\ 0 & \text{otherwise} \end{cases}$$

and the impulse response

$$h(t) = \frac{\sin \pi t f_\mathrm{c}}{\pi t f_\mathrm{c}}.$$

Sampling this impulse response with the sampling frequency $f_\mathrm{s}$ 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:

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

Solutions:

- $\longrightarrow$ Make the impulse response finite by multiplying the sampled $h(t)$ with a windowing function
- $\longrightarrow$ 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

$$h_i = \frac{\sin[2\pi(i - n/2)f_\mathrm{c}/f_\mathrm{s}]}{2\pi(i - n/2)f_\mathrm{c}/f_\mathrm{s}} \cdot w_i$$

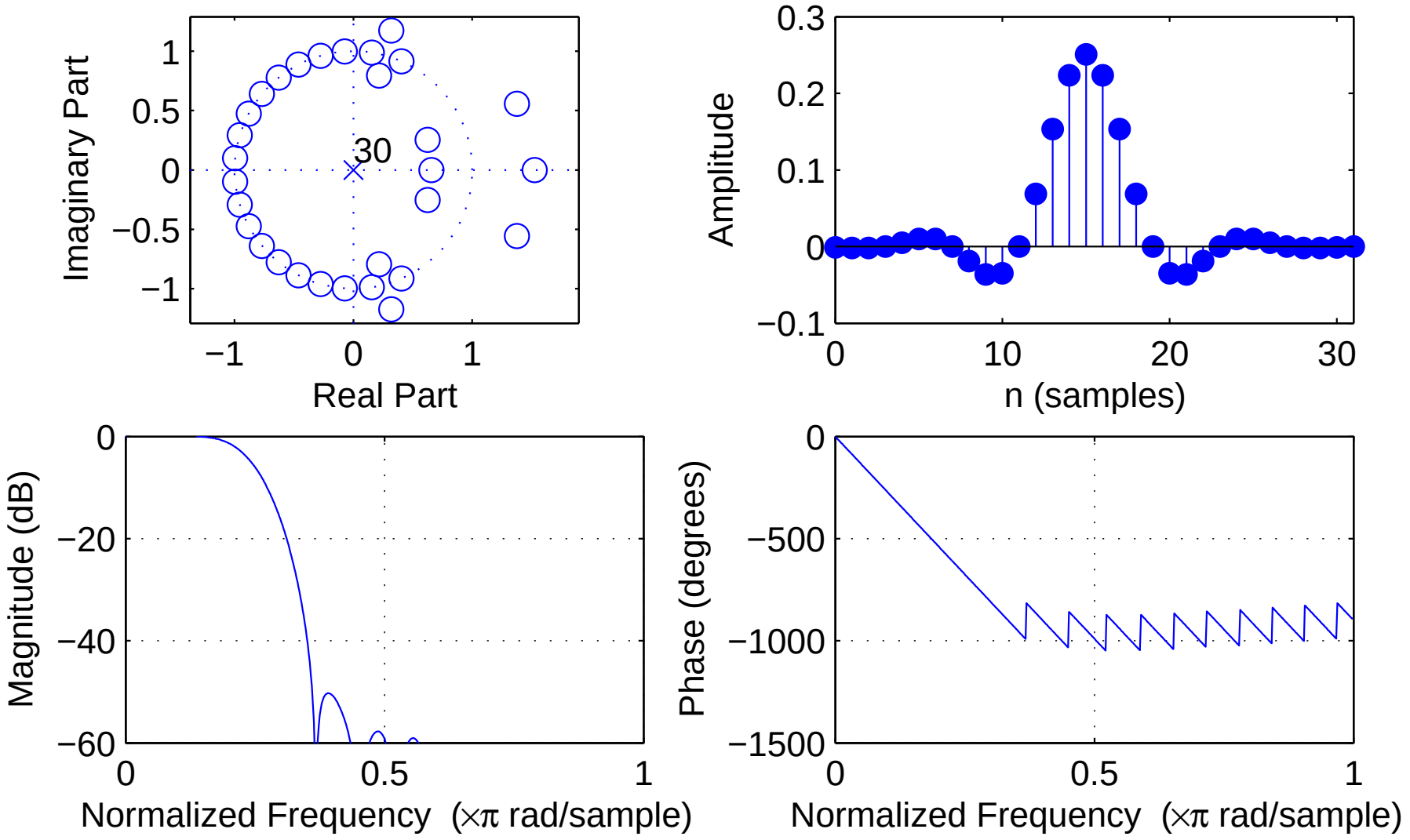
where $w_i$ is a window function, such as the Hamming window

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

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

Note that for $f_\mathrm{c} = f_\mathrm{s}/4$, 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 lowpass filter design example

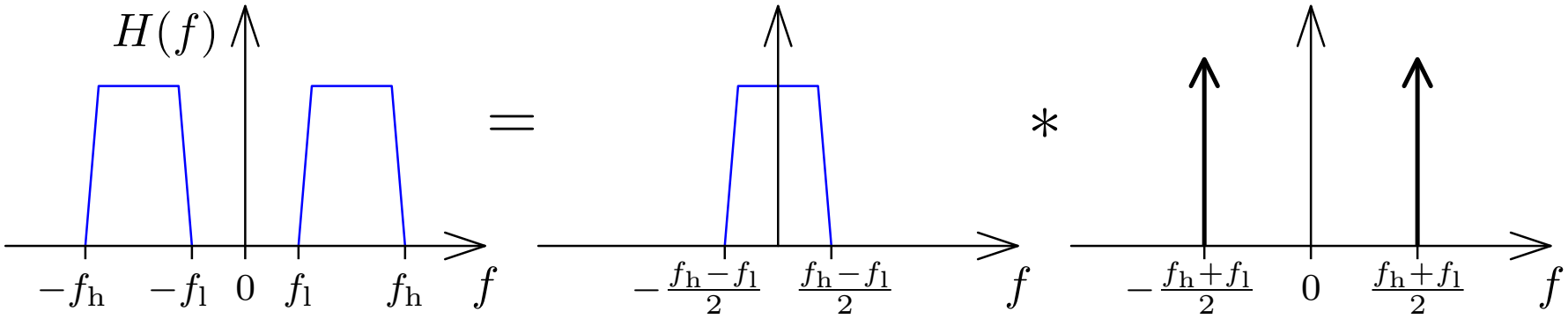



order: 30, cutoff frequency ($-6$ dB): $0.25 \times f_s/2$

Truncating the ideal infinitely-long impulse response by multiplication with a window sequence 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.

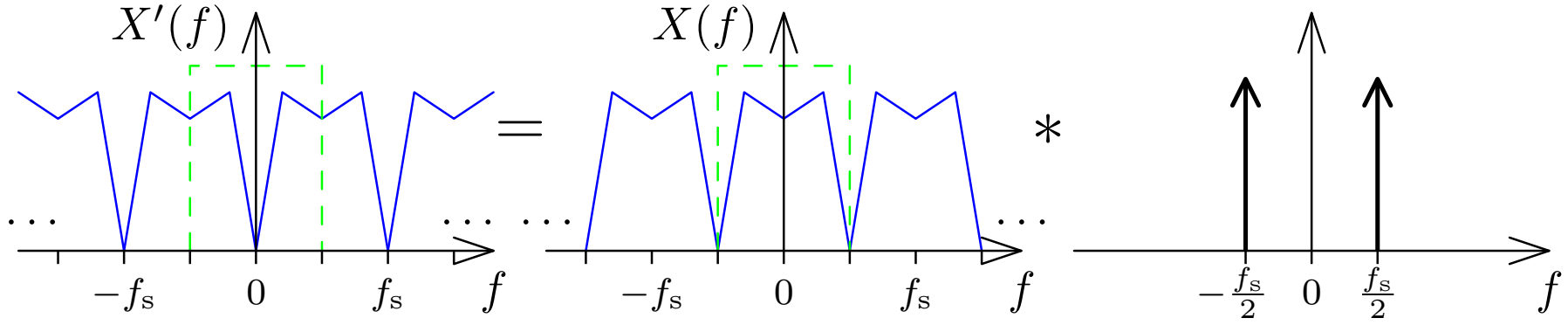
To design a passband filter that attenuates all frequencies $f$ outside the range $f_\mathrm{l} < f < f_\mathrm{h}$, we first design a low-pass filter with a cut-off frequency $(f_\mathrm{h} - f_\mathrm{l})/2$ and multiply its impulse response with a sine wave of frequency $(f_\mathrm{h} + f_\mathrm{l})/2$, before applying the usual windowing:

$$h_i = \frac{\sin[\pi(i - n/2)(f_\mathrm{h} - f_l)/f_\mathrm{s}]}{\pi(i - n/2)(f_\mathrm{h} - f_\mathrm{l})/f_\mathrm{s}} \cdot \sin[\pi(f_\mathrm{h} + f_\mathrm{l})] \cdot w_i$$



# Frequency inversion

In order to turn the spectrum $X(f)$ of a real-valued signal $x_i$ sampled at $f_\mathrm{s}$ into an inverted spectrum $X'(f) = X(f_\mathrm{s}/2 - f)$, we merely have to shift the periodic spectrum by $f_\mathrm{s}/2$:




This can be accomplished by multiplying the sampled sequence $x_i$ with $y_i = \cos \pi f_\mathrm{s} t = \cos \pi i$, 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 $f_\mathrm{c} < |f| < f_\mathrm{s}/2$, we merely have to design a low-pass filter that attenuates all frequencies outside the range $-f_\mathrm{c} < f < f_\mathrm{c}$, and then multiply every second value of its impulse response with $-1$.

# 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

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

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?

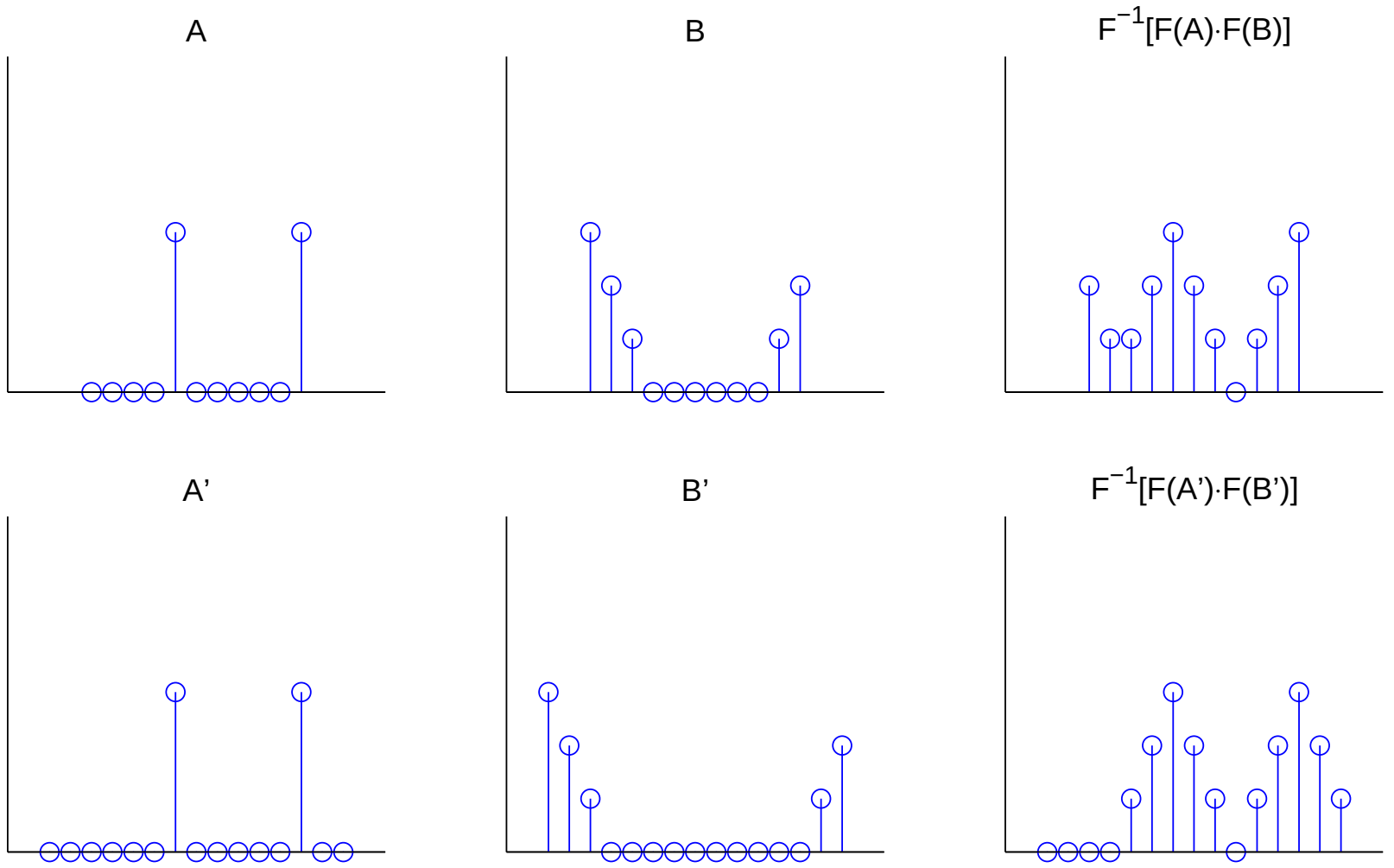
$$\{z_i\} = \mathcal{F}^{-1}\left(\mathcal{F}\{x_i\} \cdot \mathcal{F}\{y_i\}\right)$$

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 taked to prevent a wrap-over:




Both sequences are padded by appending zero values to a length of at least $m + n - 1$, to ensure that the start and end of the resulting sequence to not overlap.

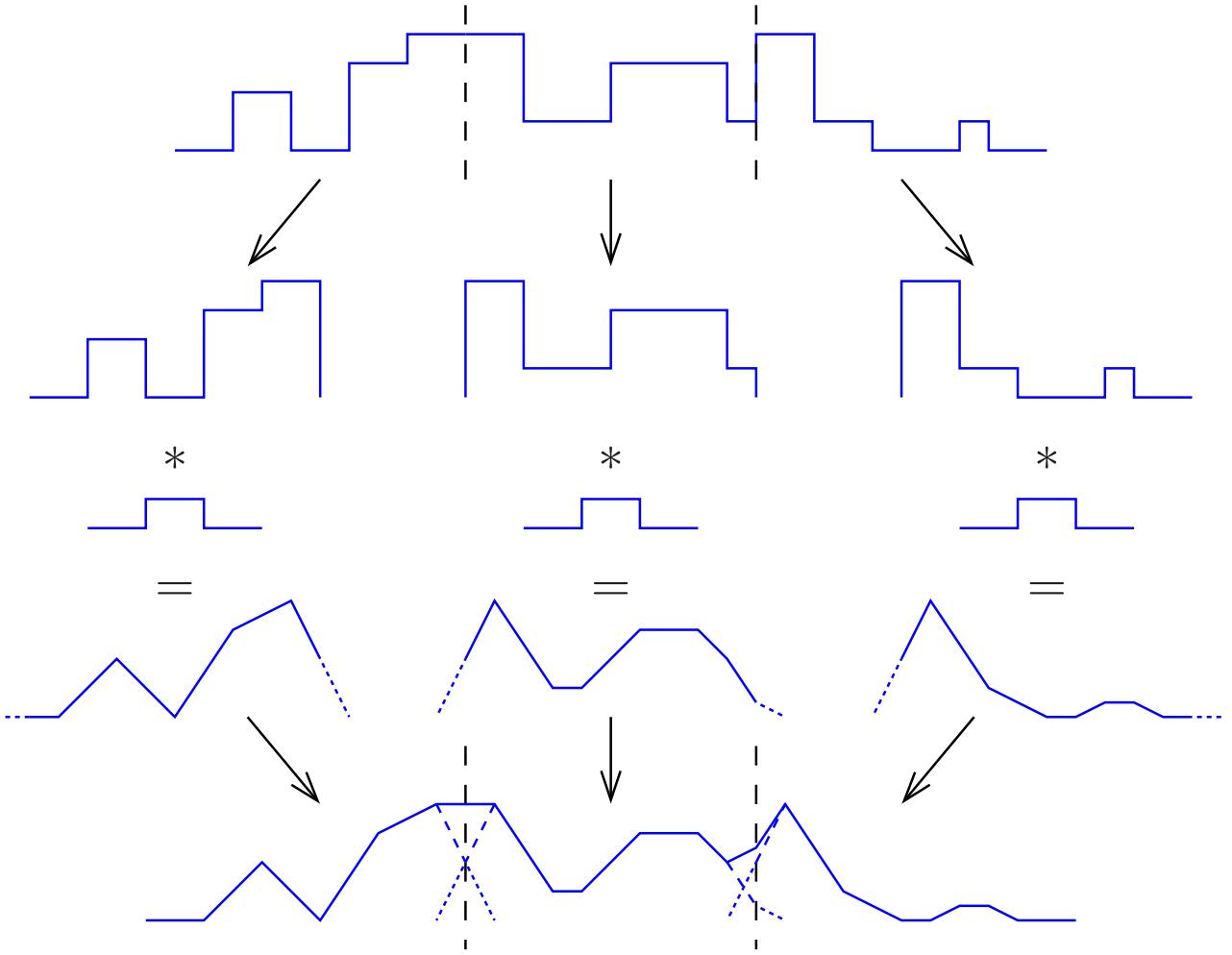
Zero padding is typically applied to extend the length of both sequences to the next larger power of two ($2^{\lceil \log_2(m+n-1) \rceil}$), to facilitate use of the FFT.

If one of the sequences is non-causal, 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 is in practice the same 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 must not 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.

# Random sequences and noise

A *discrete random sequence* $\{x_n\}$ is a sequence of numbers

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

where each value $x_n$ is the outcome of a random variable $\mathbf{x}_n$ in a corresponding sequence of random variables

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

Such a collection of random variables is called a *random process*. Each individual random variable $\mathbf{x}_n$ is characterized by its probability distribution function

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

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

$$P_{\mathbf{x}_{n_1},\ldots,\mathbf{x}_{n_k}}(a_1, \ldots, a_k) = \mathrm{Prob}(x_{n_1} \leq a_1 \wedge \ldots \wedge x_{n_k} \leq a_k)$$

for all possible sets $\{\mathbf{x}_{n_1}, \ldots, \mathbf{x}_{n_k}\}$.

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

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

and a random process is called *stationary* if

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

for all $l$, in other words, if the probability distributions are time invariant.

The derivative $p_{\mathbf{x}_n}(a) = P'_{\mathbf{x}_n}(a)$ is called the *probability density function*, and helps us to define quantities such as

- $\longrightarrow$ the *expected value* $\mathcal{E}(\mathbf{x}_n) = \int a p_{\mathbf{x}_n}(a)\,\mathrm{d}a$
- $\longrightarrow$ the *mean-square value* (average power) $\mathcal{E}(|\mathbf{x}_n|^2)$
- $\longrightarrow$ the *variance* $\mathrm{Var}(\mathbf{x}_n) = \mathcal{E}[|\mathbf{x}_n - \mathcal{E}(\mathbf{x}_n)|^2] = \mathcal{E}(|\mathbf{x}_n|^2) - |\mathcal{E}(\mathbf{x}_n)|^2$
- $\longrightarrow$ the *correlation* $\mathrm{Cor}(\mathbf{x}_n,\mathbf{x}_m) = \mathcal{E}(\mathbf{x}_n \cdot \mathbf{x}_m^*)$

Remember that $\mathcal{E}(\cdot)$ is linear, that is $\mathcal{E}(a\mathbf{x}) = a\mathcal{E}(\mathbf{x})$ and $\mathcal{E}(\mathbf{x}+\mathbf{y}) = \mathcal{E}(\mathbf{x}) + \mathcal{E}(\mathbf{y})$. Also, $\mathrm{Var}(a\mathbf{x}) = a^2\mathrm{Var}(\mathbf{x})$ and, if $\mathbf{x}$ and $\mathbf{y}$ are independent, $\mathrm{Var}(\mathbf{x}+\mathbf{y}) = \mathrm{Var}(\mathbf{x}) + \mathrm{Var}(\mathbf{y})$.

A stationary random process $\{\mathbf{x}_n\}$ can be characterized by its mean value

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

its variance

$$\sigma_x^2 = \mathcal{E}(|\mathbf{x}_n - m_x|^2) = \gamma_{xx}(0)$$

($\sigma_x$ is also called *standard deviation*), its *autocorrelation sequence*

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

and its *autocovariance sequence*

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

A stationary pair of random processes $\{\mathbf{x}_n\}$ and $\{\mathbf{y}_n\}$ can, in addition, be characterized by its *crosscorrelation sequence*

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

and its *crosscovariance sequence*

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

# Deterministic crosscorrelation sequence

For deterministic sequences $\{x_n\}$ and $\{y_n\}$, the *crosscorrelation sequence* is

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

After dividing through the overlapping length of the finite sequences involved, $c_{xy}(k)$ can be used to estimate from a finite sample of a stationary random sequence the underlying $\phi_{xy}(k)$. MATLAB's `xcorr` function does that with option `unbiased`.

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 is therefore widely calculated to locate shifted versions of a known sequence in another one.

The crosscorrelation is a close cousin of the convolution, with just the second input sequence mirrored:

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

It can therefore be calculated equally easily via the Fourier transform:

$$C_{xy}(f) = X(f) \cdot Y^*(f)$$

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

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

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

which corresponds in the frequency domain to

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

In other words, the Fourier transform $C_{xx}(f)$ of the autocorrelation sequence $\{c_{xx}(n)\}$ of a sequence $\{x_n\}$ is identical to the squared amplitudes of the Fourier transform or *power spectrum* of $\{x_n\}$.

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

# Filtered random sequences

Let $\{x_n\}$ be a random sequence from a stationary random process. The output

$$y_n = \sum_{k=-\infty}^{\infty} h_k \cdot x_{n-k} = \sum_{k=-\infty}^{\infty} h_{n-k} \cdot x_k$$

of an LTI applied to it will then be another random sequence, characterized by

$$m_y = m_x \sum_{k=-\infty}^{\infty} h_k$$

and

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

In other words:

$$\{y_n\} = \{h_n\} * \{x_n\} \quad \Rightarrow \quad \begin{aligned} \{\phi_{yy}(n)\} &= \{c_{hh}(n)\} * \{\phi_{xx}(n)\} \\ \Phi_{yy}(f) &= |H(f)|^2 \cdot \Phi_{xx}(f) \end{aligned}$$

Similarly:

$$\{y_n\} = \{h_n\} * \{x_n\} \quad \Rightarrow \quad \begin{aligned} \{\phi_{xy}(n)\} &= \{h_n\} * \{\phi_{xx}(n)\} \\ \Phi_{xy}(f) &= H(f) \cdot \Phi_{xx}(f) \end{aligned}$$

# White noise

A random sequence $\{x_n\}$ 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:

$$\Phi_{xx}(f) = \sigma_x^2.$$

Application example:

Where an LTI $\{y_n\} = \{h_n\} * \{x_n\}$ can be observed to operate on white noise $\{x_n\}$ with $\phi_{xx}(k) = \sigma_x^2 \delta_k$, the crosscorrelation between input and output will reveal the impulse response of the system:

$$\phi_{xy}(n) = \sigma_x^2 \cdot h_n.$$

# Averaging and noise reduction

Often an original signal $\{x_i\}$ is only accessible with some added noise

$$\{y_i\} = \{x_i\} + \{n_i\}$$

which turns a deterministic sequence into a random sequence. The signal-to-noise ratio (SNR) of the received signal $\{y_i\}$ is the square root of the power ratio of these components: $\mathrm{SNR}_y = \sqrt{\mathcal{E}(|x_i|^2)/\mathcal{E}(|n_i|^2)}$.

As an SNR might also be given in terms of a power ratio, it is commonly expressed in decibels, to avoid any confusion between power and voltage ratios: $10\ \mathrm{dB} \cdot \log_{10} \mathcal{E}(|x_i|^2)/\mathcal{E}(|n_i|^2) = 20\ \mathrm{dB} \cdot \log_{10} \sqrt{\mathcal{E}(|x_i|^2)/\mathcal{E}(|n_i|^2)}$.

The simplest noise reduction technique is averaging. If we know that the $k$ signal values $x_1, \ldots, x_k$ are identical, and the noise average is $m_n = 0$, then we can calculate an average value

$$\bar{y} = \frac{1}{k} \sum_{i=1}^{k} y_i$$

as an approximation for the true value $x_1 = \cdots = x_k$.

# What noise level remains after averaging?

The $k$ identical signal values $x_1, \ldots, x_k$ are characterized by $m_x = x_i$ $(i = 1, \ldots, k)$ and $\sigma_x^2 = 0$. The average signal power is $\mathcal{E}(|x_i|^2) = m_x^2$.

We assume that the $k$ noise values $n_1, \ldots, n_k$ are statistically independent and are the output of a stationary process with mean $m_n = 0$ and variance $\sigma_n^2$. The average noise power is $\mathcal{E}(|n_i|^2) = \sigma_n^2$.

The averaging result $\bar{y}$ can be split up into a signal component $\bar{x}$ and a noise component $\bar{n}$:

$$\bar{y} = \frac{1}{k}\sum_{i=1}^{k} y_i = \frac{1}{k}\sum_{i=1}^{k} x_i + \frac{1}{k}\sum_{i=1}^{k} n_i = \bar{x} + \bar{n}$$

The corresponding average power values are

$$\mathcal{E}(|\bar{x}|^2) = \mathcal{E}\left(\left|\frac{1}{k}\sum_{i=1}^{k} x_i\right|^2\right) = \mathcal{E}\left(\left|\frac{1}{k}\sum_{i=1}^{k} m_x\right|^2\right) = m_x^2$$

and

$$\mathcal{E}(|\bar{n}|^2) = \mathrm{Var}(\bar{n}) = \mathrm{Var}\left(\frac{1}{k}\sum_{i=1}^{k} n_i\right) = \frac{1}{k^2}\sum_{i=1}^{k}\mathrm{Var}(n_i) = \frac{1}{k}\sigma_n^2.$$

We can now compare the signal-to-noise ratio of the original noisy sequence

$$\mathrm{SNR}_y = \sqrt{\frac{\mathcal{E}(|x_i|^2)}{\mathcal{E}(|n_i|^2)}} = \frac{m_x}{\sigma_n}$$

with that of the averaging result

$$\mathrm{SNR}_{\bar{y}} = \sqrt{\frac{\mathcal{E}(|\bar{x}|^2)}{\mathcal{E}(|\bar{n}|^2)}} = \sqrt{k}\cdot\frac{m_x}{\sigma_n}.$$

Averaging $k$ samples of identical signal values with added independent zero-mean noise values will increase the signal-to-noise ratio by the factor $\sqrt{k}$.

Remember that adding identical values $x_1$ and $x_2$ will double their value and therefore quadruple their power. On the other hand, adding independent zero-mean noise values $n_1$ and $n_2$ will – on average – only double their power (= variance).

# Noise reduction filters

Added independent noise values with $\phi_{nn}(k) = \sigma_n^2 \delta_k$ have a flat spectrum $\Phi_{nn}(f) = \sigma_n^2$, which is added in $\{y_i\} = \{x_i\} + \{n_i\}$ across the spectrum of the noise-free signal $\{x_i\}$.

Knowledge of the power spectrum $X(f)$ of the original signal can help to design a filter that attenuates much of the noise in $\{y_i\}$, without degrading the wanted signal $\{x_i\}$ too much. If $\{x_i\}$ changes very slowly, most of its energy will be at low frequencies, and noise can be reduced with a suitably chosen low-pass filter.

If there are no significant changes in $\{x_i\}$ during $k$ consecutive samples, then convolution with a $k$ samples wide rectangular window will not affect the wanted signal.

We have already seen that this will reduce the noise amplitude by a factor $\sqrt{k}$, or equivalently the noise power by a factor of $k$.

How does a general LTI filter

$$\bar{y}_i = \sum_{k=-\infty}^{\infty} h_k \cdot y_{i-k} = \sum_{k=-\infty}^{\infty} h_k \cdot (x_{i-k} + n_{i-k}) = \bar{x}_i + \bar{n}_i$$

affect the signal-to-noise ratio $\mathrm{SNR_y} = \sqrt{\mathcal{E}(|x_i|^2)/\mathcal{E}(|n_i|^2)}$?
With $m_x \approx x_i$, we get

$$\mathcal{E}(|\bar{x}_i|^2) = \mathcal{E}\left(\left|\sum_{k=-\infty}^{\infty} h_k x_{i-k}\right|^2\right) = m_x^2 \left|\sum_{k=-\infty}^{\infty} h_k\right|^2$$

and

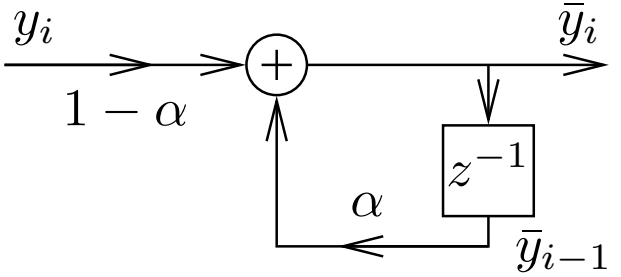
$$\mathcal{E}(|\bar{n}_i|^2) = \mathrm{Var}(\bar{n}_i) =$$

$$\mathrm{Var}\left(\sum_{k=-\infty}^{\infty} h_k n_{i-k}\right) = \sum_{k=-\infty}^{\infty} h_k^2 \cdot \mathrm{Var}(n_{i-k}) = \sigma_n^2 \sum_{k=-\infty}^{\infty} h_k^2$$

$$\Rightarrow \mathrm{SNR_{\bar{y}_i}}/\mathrm{SNR_{y_i}} = \left|\textstyle\sum_{k=-\infty}^{\infty} h_k\right| \Big/ \sqrt{\textstyle\sum_{k=-\infty}^{\infty} h_k^2}$$

# Exponential averaging

A particularly easy to implement IIR low-pass filter, with only one delay element, is the *exponential averaging filter*




$$\bar{y}_i = (1-\alpha)y_i + \alpha\bar{y}_{i-1} = (1-\alpha)\sum_{k=0}^{\infty}\alpha^k y_{i-k}, \qquad (0 \le \alpha < 1).$$

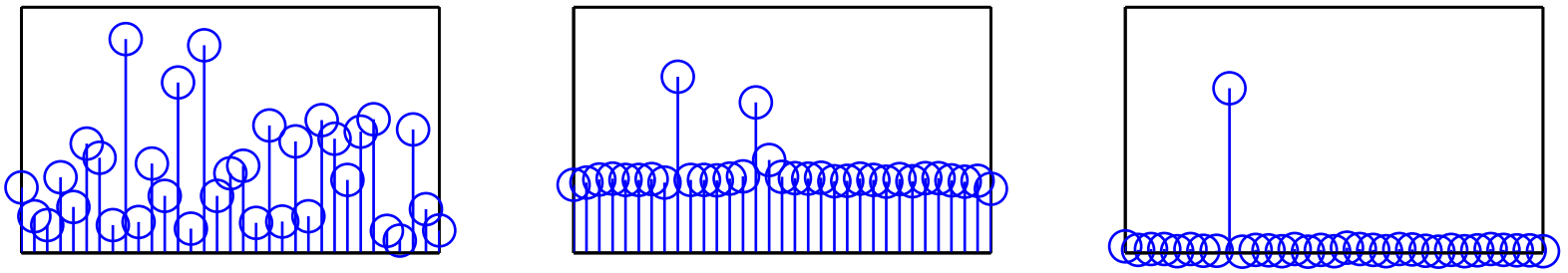
When applied as a noise filter on a sequence $\{y_i\} = \{x_i\} + \{n_i\}$ with $\sigma_x^2 = 0$, $m_n = 0$, and mutually independent noise values $\{n_i\}$, we get

$$\sigma^2_{\bar{y}_i} = (1-\alpha)^2\sum_{k=0}^{\infty}\alpha^{2k}\sigma^2_{y_{i-k}} = \sigma_n^2(1-\alpha)^2\sum_{k=0}^{\infty}\alpha^{2k} =$$

$$\sigma_n^2(1-\alpha)^2\cdot\frac{1}{1-\alpha^2} = \sigma_n^2\cdot\frac{1-\alpha}{1+\alpha}$$

and therefore $\mathrm{SNR}_{\bar{y}_i}/\mathrm{SNR}_{y_i} = \dfrac{m_x}{\sigma_{\bar{y}_i}}\Big/\dfrac{m_x}{\sigma_n} = \sqrt{\dfrac{1+\alpha}{1-\alpha}}.$

# DFT averaging



The above diagrams show different type of spectral estimates of a sequence $x_i = \sin(2\pi \mathrm{j} \times 8/64) + \sin(2\pi \mathrm{j} \times 14.32/64) + n_i$ with $\phi_{nn}(i) = 4\delta_i$.

Left is a single 64-element DFT of $\{x_i\}$ (with rectangular 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 sine waves.

After cutting $\{x_i\}$ into 1000 windows of 64 elements each, calculating their DFT, and plotting the average of their absolute values, the centre figure shows an approximation of the expected value of the amplitude spectrum, with a flat noise floor. Taking the absolute value before averaging is called *incoherent averaging*, as the phase information is thrown away.

The rightmost figure was generated from the same set of 1000 windows, but this time the complex values of the DFTs were averaged before the

absolute value was taken. This is called *coherent averaging* and, because of the linearity of the DFT, identical to first averaging the 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 that divides 64 are not averaged away. This periodic averaging step suppresses both the noise and the second sine wave.

# Periodic averaging

If a zero-mean signal $\{x_i\}$ has a periodic component with period $p$, the periodic component can be isolated by *periodic averaging*:

$$\bar{x}_i = \lim_{k\to\infty} \frac{1}{2k+1} \sum_{n=-k}^{k} x_{i+pn}$$

Periodic averaging corresponds in the time domain to convolution with a Dirac comb $\sum_n \delta_{i-pn}$. In the frequency domain, this means multiplication with a Dirac comb that eliminates all frequencies but multiples of $1/p$.

**Exercise 9** A tumble dryer measures the air humidity $x_i$ in the drum twice per second, but the control software uses only a smoothed value $y_i = \sum_{k=0}^{15} x_{i-k}$ to decide when to stop the drying programme.

(a) Assuming that humidity fluctuations in a tumble dryer have components that are periodic with the rotational frequency of the drum, for which rotational speeds would the above smoothing filter suppress these periodic components particularly well?

(b) It is the year 2020, and a cartel of semiconductor manufacturers recently managed to double the cost of memory every 18 months. Your job is to redesign products to eliminate unnecessary use of precious memory. Replace the above data smoothing algorithm with one that requires the least amount of memory possible, but that reduces the random fluctuations caused in the measurements by tumbling clothes by exactly the same factor. You now can assume that these fluctuations are not correlated between different measurements $x_i$.

(c) First predict the approximate shape of the frequency characteristic of both the old and the new smoothing filter, then use MATLAB to plot it.